



The
University
Of
Sheffield.

Crowdsourced Testing Approach For Mobile Compatibility Testing

By:
Qamar H. Naith

*A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the*

Faculty of Engineering
Department of Computer Science

June 2021

I dedicate my PhD thesis to my parents who have encouraged me in all stages of my life and are my greatest source of inspiration.

Declaration

I hereby declare that this thesis with the title “Novel Crowdsourced Compatibility Testing Strategy for Addressing Mobile Device Fragmentation Issues” is entirely my own work, and I have submitted my thesis to the University of Sheffield for the degree of Doctor of Philosophy. I confirm that:

- The work presented in this thesis has been composed by myself under the supervision of Prof. Ciravegna, without any collaboration with other researchers.
- The text of this thesis includes no material previously published or written by any other researchers except where particular reference is made to the work of others.
- The content of this thesis has not been submitted in whole or in part for obtaining a qualification or an academic degree in this university or any other university.
- Some parts of this thesis are published in peer-reviewed conferences and journals, as specified in Section [1.5](#).

Qamar H. Naith
June 2021

Acknowledgements

This PhD thesis extends beyond an individual experience to encompass a larger social context within which this work would be beneficial and includes others whom have helped me finish my research. I want to take this opportunity to acknowledge and thank those without whom this work would not have been the same.

First and foremost, I would like to express my gratitude to **my PhD supervisor, Professor Fabio Ciravegna**, who was not only my source of guidance throughout my four years of research and publications but a life mentor. His insightful outlooks on researches and passion for teaching and academia constantly created motivational avenues for my career as a researcher. He presented me with teaching opportunities that further strengthened my abilities as an academic, and his discussions were nothing short of stimulating, engaging, and encouraging despite my perplexities. Professor Fabio Ciravegna was also an inspirational mentor who constantly supported me and assisted me in building a positive attitude during the difficult times of my illness within the duration of research and the COVID-19 pandemic. In all honesty, I feel blessed to have had Professor Fabio Ciravegna as my supervisor.

Secondly, I would also like to acknowledge the support I received from the staff at the Department of Computer Science at the University of Sheffield, especially the heads of department **Professor Guy Brown** and **Karen Barker**, regarding all procedural paperwork that was resultant from project bottlenecks and health-related delays.

In addition, I am extremely grateful to the following people who accepted to be part of the panel of experts **Professor.Mohammed Assaf, Professor.Haroun Sabra, Dr.Hamid Ghaeini, Dr.Salam Al-Jurani, Dr.Mutaz Al-bajaq, Eng.Mohammed Borhan, Eng. Hussam Yahia** who remained by my side and provided me with their encouraging words. Also, I would like to extend a special thank you to my colleagues in **The OAK Research Group** for sharing this research journey with me and providing me with advice regarding numerous practical issues, especially **Dr.Suvodeep Mazumdar** and **Anne Hayes**.

I am also deeply grateful to those people who work at the Saudi Arabian Cultural Bureau (SACB) **Dr Amra, Dr.Adel Al-Yobi, Mr.Mahdi Mohamed, Mr.Othman Alodhibi, and Ms.Amel Trabelsi** and at the University of Jeddah, **Dr.Fatmah Assiri, Areej Alshutayri, and Atyah Albushri** for their help, encouragement, support, as well as for providing me with this opportunity by medically and financially sponsoring my PhD.

I also wish to send my immeasurable appreciation to the **NHS staff and the University Health Service (GP)** for also manifesting their admirable resilience throughout the coronavirus pandemic. Their continuous commitment to their patients, one of whom was myself, has given me comfort and peace of mind during an immensely difficult time with my prolonged illness, and I will always be grateful for these heroes.

Finally, I want to acknowledge the efforts of **the receptionists in The Diamond Library Team, and the engineering staff on the 3rd floor** for going above and beyond their duties to ensure both my physical health and safety along with my fellow colleagues during my utilisation of the library and learning commons space. They patiently and laboriously attended to all my needs during those times from technical printing issues to valuable kind words during stressful periods.

Crowdsourced Testing Approach For Mobile Compatibility Testing

Qamar H. Naith

Abstract

The frequent release of mobile devices and operating system versions bring several compatibility issues to mobile applications. This thesis addresses fragmentation-induced compatibility issues. The thesis comprises three main phases. The first of these involves an in-depth review of relevant literature that identifies the main challenges of existing compatibility testing approaches. The second phase reflects on the conduction of an in-depth exploratory study on Android/iOS developers in academia and industry to gain further insight into their actual needs in testing environments whilst gauging their willingness to work with public testers with varied experience. The third phase relates to implementing a new manual crowdtesting approach that supports large-scale distribution of tests and execution by public testers and real users on a larger number of devices in a short time. The approach is designed based on a direct crowdtesting workflow to bridge the communication gap between developers and testers. The approach supports performing the three dimensions of compatibility testing. This approach helps explore different behaviours of the app and the users of the app to identify all compatibility issues. Two empirical evaluation studies were conducted on iOS/Android developers and testers to gauge developers' and testers' perspectives regarding the benefits, satisfaction, and effectiveness of the proposed approach. Our findings show that the approach is effective and improves on current state-of-the-art approaches. The findings also show that the approach met the several unmet needs of different groups of developers and testers. The evaluation proved that the different groups of developers and testers were satisfied with the approach. Importantly, the level of satisfaction was especially high in small and medium-sized enterprises that have limited access to traditional testing infrastructures, which are instead present in large enterprises. This is the first research that provides insights for future research into the actual needs of each group of developers and testers.

Table of contents

List of figures	xvii
List of tables	xxi
Nomenclature	xxiii
1 Introduction	1
1.1 Research Problems and Motivation	2
1.2 Research Aims and Objectives	4
1.3 Research Questions	5
1.4 Research Contributions	6
1.5 Publication Summary of the Thesis	8
1.5.1 Publications Arising from this Research	8
1.6 Thesis Structure	10
2 A State-of-the-Art Review	13
2.1 Introduction	13
2.2 Mobile Compatibility Testing: Dimensions, Challenges, and Requirements . .	14
2.3 Compatibility Testing Solutions	15
2.3.1 Existing Mobile Compatibility Testing Solutions	16
2.3.2 The Limitation of Existing Compatibility Testing Solutions	18
2.4 An Overview of Current Crowdttesting Workflows	19
2.4.1 Manual Crowdttesting workflow (M-CT)	19
2.4.1.1 Limitations of Manual Crowdttesting workflow (M-CT)	22
2.4.2 Automated Crowdttesting Workflow (Auto-CT)	22
2.4.2.1 limitations of Automated Crowdttesting Workflow (Auto-CT)	24
2.5 An overview of Crowdsourced Testing Research	26
2.6 Current Crowdsourced Compatibility Testing Solutions	29
2.7 Current Industrial Mobile App Crowdttesting Platforms	30
2.7.1 Limitations of Current Crowdsourced Compatibility Practices	37

2.8	Challenges Faced by Developers and Testers in Crowdtesting	38
2.9	Place of this Research in the Literature	39
2.10	Chapter Summary	40
3	Research Design and Methodology	43
3.1	Introduction	43
3.2	Research Design and Methodology	43
3.2.1	Phase 1: Key Requirements for the Crowdtesting Approach	44
3.2.1.1	Applied Methods	46
3.2.2	Phase 2: Development of the Crowdtesting Approach	47
3.2.3	Phase 3: Evaluation of the Crowdtesting Approach	47
3.2.3.1	Applied Methods	48
3.3	Research Validation	50
3.3.1	Construct Validity	50
3.3.2	Internal Validity	51
3.3.3	External Validity	52
3.4	Ethical considerations	54
3.5	Chapter Summary	54
I	<u>Research Phase 1: Key Requirements for Crowdtesting Approach</u>	57
4	Critical Requirements of Proposed Crowdtesting Approach	59
4.1	Introduction	59
4.2	Data Collection Method	60
4.2.1	Questionnaire Survey Design	60
4.2.2	Questionnaire Quality Evaluation	62
4.3	Population and Sampling Method	65
4.4	Procedures	65
4.5	Data Analysis Method	66
4.6	Study Results	69
4.7	Main Discussion	80
4.8	Chapter Summary	91
II	<u>Research Phase 2: Development of Crowdtesting Approach</u>	95
5	Design and Development of the Proposed Crowdtesting Approach	97
5.1	Introduction	97
5.2	Distinctive Features of the Proposed Approach	98
5.3	Proposed Crowdtesting Approach Working Mechanism	99

5.4	Design and Implementation of the Proposed Approach	102
5.4.1	Tasks Defining and Distribution	102
5.4.2	Testers and Task Selection	103
5.4.3	Results Submission	109
5.4.4	Report Tracking and Evaluation	113
5.4.5	Tester Motivation/Reward	117
5.4.6	Repository/Wiki	118
5.5	Chapter Summary	124
 III <u>Research Phase 3: Empirical Assessment of the Developed Crowdtesting Approach</u>		125
6	Empirical Evaluation Studies: Data Collection, & Data Analysis Methods	127
6.1	Introduction	127
6.2	Data Collection Method	128
6.2.1	Measurement Variables	128
6.2.2	Scale of Measurement	132
6.2.3	Questionnaire Surveys Structure	133
6.2.4	Evaluation of Questionnaires and Measurement Quality	133
6.2.4.1	Iterative Question Refinement	133
6.2.4.2	Reliability and Validity Assessment	134
6.3	Sampling Methods and Data Collection Procedure	142
6.4	Data Analysis Method	143
6.4.1	Data Coding	144
6.4.2	Data Checking and Cleaning	145
6.4.3	Descriptive Statistical Analysis	146
6.4.4	Inference Statistical Analysis	148
6.5	Chapter Summary	150
7	Empirical Study(I): Developers Experience With Crowdtesting Approach	153
7.1	Introduction	153
7.2	Developers Demographic Information	154
7.3	Descriptive Statistics Results	155
7.4	Results I: Effectiveness	160
7.4.1	Overall Effectiveness of the Approach	160
7.4.2	Effectiveness of the each processes in the Approach	162
7.5	Results II: Benefits/Advantages	167
7.5.1	Benefits of the Approach	167
7.5.2	The Difference Between Developers Groups Regarding Benefits	168

7.5.3	Received Benefits by Each Group	168
7.6	Results III: Satisfaction	174
7.6.1	Overall Satisfaction of the Approach	174
7.6.2	Satisfaction for Different Developers Groups	174
7.7	Open Question Analysis Results	176
7.8	Chapter Summary	176
8	Empirical Study(II): Testers Experience With Crowdtesting Approach	177
8.1	Introduction	177
8.2	Testers Demographic Information	178
8.3	Descriptive Statistics Results	180
8.4	Results I: Effectiveness	183
8.4.1	Overall Effectiveness of the Approach	183
8.4.2	Effectiveness of the each processes in the approach	184
8.5	Results II: Benefits/Advantages	189
8.5.1	Benefits of the Approach	189
8.5.2	The Difference Between Testers Groups Regarding Benefits	189
8.5.3	Received Benefits by Each Group	191
8.6	Results III: Satisfaction	197
8.6.1	Overall Satisfaction of the Approach	197
8.6.2	Satisfaction for Different Testers Groups	198
8.7	Open Question Analysis Results	199
8.8	Chapter Summary	200
9	Results Discussion and Related Implications	201
9.1	Introduction	201
9.2	Effectiveness of the Proposed Crowdtesting Approach	202
9.3	Benefit of the Proposed Crowdtesting Approach	228
9.4	Satisfaction with the Proposed Crowdtesting Approach	236
9.5	Chapter Summary	237
10	Conclusion and Future Work	239
10.1	Introduction	239
10.2	Conclusion of the Thesis	239
10.3	Guiding Principles for Researchers Interested in Crowdsourcing Contexts	242
10.4	Research Limitations	245
10.5	Future Research Directions	246
	Bibliography	249

Appendix A Ethical Considerations	267
Appendix B Requirements Gathering Questionnaire	271
Appendix C Empirical Evaluation Questionnaires	281
Appendix D Additional Results of Both Studies	291

List of figures

2.1	The activities of the manual crowdtesting workflow (M-CT) [adapted from Alyahya and Alrugebh (2017)]	21
2.2	The activities of the Automated crowdtesting workflow (Auto-CT) [adapted from Alyahya and Alrugebh (2017)]	25
3.1	Illustration of the overall research design and methodology	45
3.2	Mapping of research objectives and research questions across the research methodological phase	53
4.1	Structural design of the questionnaire	63
4.2	Analysis of participant developers responses: (a) identification and encoding participants' feedback based on the color-coding of the relevant category; and b) hand-drawn tally marks helped to count and summarize the number of category repetitions based on survey responses.	68
5.1	Detailed direct-interaction workflow of proposed crowdtesting approach (DT-CT)	100
5.2	Task defining and distributing mechanism	104
5.3	The process of distributing the test and selecting the specialised testers	106
5.4	The method of selecting tasks by testers	108
5.5	The methods of tracking published tasks' status	109
5.6	Results reporting mechanism	111
5.7	Mobile device data detection method	112
5.8	The two ways of filtering and organising test reports	113
5.9	The tracking method of reports/issues status from the developer and tester side	114
5.10	Report quality evaluation criteria	116
5.11	Activities summary dashboard for the tester	117
5.12	The rewarding process from the developer and tester side	119
5.13	Display of potential reward value based on quality level within task requirement	120
5.14	The two implemented methods for tracking non-paid testers	121

5.15	The implemented wiki for documenting the mobile device compatibility testing knowledge	122
5.16	The auto-tagging system and searching mechanism of the implemented wiki	123
6.1	The Descriptive and Inferential Statistics Data Analysis Methods	144
6.2	Screenshots of the SCA method applied for analysing <i>open-ended</i> question in both studies.	147
6.3	A histogram of the normality distribution of the developers' data	149
6.4	A histogram of the normality distribution of the testers' data	149
7.1	The effectiveness percentage for each process in the approach, which achieves the overall effectiveness percentage (77.4%).	161
7.2	Effectiveness percentage for each characteristic within all processes (sub-dimensions) of the proposed approach	164
7.3	Comparison between different developers' groups regarding benefit/advantage of the approach	169
7.4	The percentages and mean scores for the most important benefits received by Android and iOS developers.	170
7.5	The percentages and mean scores for the most important benefits received by Employees and Freelancers developers.	172
7.6	The percentages and mean scores for the most important benefits received by Small Organisations and Big Organisations developers.	173
7.7	Comparison between different developers' groups regarding overall satisfaction of the approach	175
8.1	The effectiveness percentage for each process in the approach, which achieves the overall effectiveness percentage (82.2%).	185
8.2	Effectiveness percentage for each characteristic within all processes (sub-dimensions) of the proposed approach	187
8.3	Comparison between different contexts of testers regarding benefit/advantage of the approach	191
8.4	The percentages and mean scores for the most important benefits received by Android and iOS testers.	192
8.5	The percentages and mean scores for the most important benefits received by Employees and Freelancers testers.	194
8.6	The percentages and mean scores for the most important benefits received by Small Organisations and Big Organisations testers.	195
8.7	The percentages and mean scores for the most important benefits received by Beginners, Mid-level, and Experts testers.	197
8.8	Comparison between testers' group regarding overall satisfaction of the approach	200

B.1	Screenshots of the developers' answers collected by Google Form.	279
C.1	Screenshots of the online questionnaire for developers	285
C.2	Screenshots of the online questionnaire for Testers	289
D.1	Screenshots of the developers' answers collected by Google Form.	292
D.2	Screenshots of the testers' answers collected by Google Form.	293

List of tables

1.1	Published works under each chapter and names of journals and conferences. . .	9
2.1	List of exiting industrial crowdsourced testing platforms	32
2.2	Comparison between existing Industrial Crowdsourced Testing Platforms . .	33
2.3	Comparison between existing Industrial Crowdsourced Testing Platforms, Cont...	34
2.4	Comparison between existing Industrial Crowdsourced Testing Platforms, Cont...	35
2.5	Comparison between existing Industrial Crowdsourced Testing Platforms, Cont...	36
2.6	Summary of the identified challenges	40
4.1	The overall outcomes of "Inter-rater Reliability" test	64
4.2	Examples of some generated categories from participant responses by the main researcher and co-researcher	70
4.3	The proportional use of mobile apps testing methods from the participants' perspective	71
4.4	The proportional use of the three crowdsourced programming websites Stack Overflow, GitHub and Stack Exchange from the participants' perspective . .	71
4.5	Important factors used as evidence on the accuracy of results	76
4.6	A list of non-functional requirements for the public crowdtesting approach . .	92
4.7	A list of functional requirements for defining and distributing tasks on a large-scale	93
6.1	Summary of the two pilot studies	136
6.2	Internal Consistency Reliability (α) and Construct Validity of the CSTE-Q/D questions	138
6.3	Internal Consistency Reliability (α) and Construct Validity for the CSTE-Q/T questions	140
6.4	List of tasks related to developers	143
6.5	List of tasks related to testers	143
6.6	The variables of demographic questions and their coding	145
7.1	Developers demographic information (N = 34).	155

7.2	Developers' answers to all questions of the CSTE-Q/D questionnaire regarding Effectiveness	156
7.3	Developers' answers to all questions of the CSTE-Q/D questionnaire regarding Benefits and Satisfaction	160
7.4	The mean score and standard deviation values of participant developers according to the "Overall Effectiveness".	161
7.5	Summary of the mean score analysis of each question (feature) within sub-dimensions of effectiveness.	162
7.6	The mean score and standard deviation values of participant developers according to the "Benefit//advantage"	168
7.7	Summary of the independent samples t-test results (regarding benefits) . . .	169
7.8	The mean score and standard deviation values of participant developers according to the "Overall Satisfaction".	174
7.9	Summary of the independent samples t-test results (regarding satisfaction) .	175
8.1	Testers demographic information (N = 31).	179
8.2	Testers' Answeres to all questions of the CSTE-Q/T questionnaire	182
8.3	The mean score and standard deviation values of participant testers according to the "Overall Effectiveness".	184
8.4	Summary of the mean values of all questions within effectiveness sub-dimensions.	185
8.5	The mean score and standard deviation values of participant testers according to the "Benefit//advantage"	190
8.6	Summary of the independent samples t-test results (regarding benefits) . . .	190
8.7	Summary of the one-way ANOVA test results (regarding benefits)	191
8.8	The mean score and standard deviation values of participant testers according to the "Overall Satisfaction".	198
8.9	Summary of the independent samples t-test results (regarding satisfactions) .	199
8.10	Summary of the one-way ANOVA test results (regarding satisfactions)	199
C.1	Questionnaire Questions for Developer (CSTE-Q/D)	283
C.2	Questionnaire Questions for Tester (CSTE-Q/T)	287
C.3	Questions deleted after reliability and validity tests	290
C.4	Testing the normality distribution of the developers' and testers' responses using "One-Sample Kolmogorov-Smirnov Test".	290

Nomenclature

Acronyms / Abbreviations

API Application Programming Interface

Apps Applications

CCTF Conceptual Crowdtesting Framework

CI Correlated Item

CSTE-Q Crowdsourced Testing Experience Questionnaire

GUI Graphical User Interface

HW Hardware

I-CVI Item Content Validity Index

IRR Inter-Rater Reliability

OS Operating System

PCA Principal Component Analysis

rbi Point Biserial Correlation Coefficient

S-CVI/UA Scale Content Validity Index/ Universal Agreement

SMEs Small/Medium-sized Enterprises

UCD User-Centered Design

1

Introduction

In recent years, mobile device diffusion and penetration have grown exponentially. This tremendous growth has led to the emergence of the fragmentation phenomenon in mobile devices and operating system (OS) versions (Park et al., 2013). Device fragmentation refers to the vast diversity in mobile device models, accompanied by different hardware (HW) characteristics and API's level configurations. At the same time, OS fragmentation refers to the large variety of OS versions (Kamran et al., 2016). Consequently, the fragmentation has led to a reduced standardisation by different manufacturers, especially in the Android world due to the broader diversity of Android device manufacturers (Huang, 2014). The presence of the fragmentation issue has subsequently led to the testing process becoming much more difficult for app developers.

Compatibility testing is a solution proposed for resolving fragmentation issues (Park et al., 2013). It requires developers to test their apps on a large range of mobile device models and OS versions before the app launch, to ensure their portability, test the correct behaviour of the different functionalities, and hence prevent a poor user experience. Given the high cost and effort required by compatibility testing Liu (2019); Zhang et al. (2015), there has been an increased effort towards the development of novel cost-effective test methods and tools to support compatibility testing.

Among the proposed methods, crowdsourcing has gained momentum as an emerging trend (Mao et al., 2017; Sari and Alptekin, 2017). Crowdsourcing has been adopted in various activities within different software development phases and include: planning, requirements extraction and analysis, design, implementation (programming/coding), and software testing (called crowdtesting). Crowdtesting focuses on exploiting distributed human effort to manually perform mobile app testing on various platforms, devices, and system configurations from different regions or countries under real-world conditions (Mao et al., 2017; Wu et al., 2017). In crowdtesting, many users and experts with different mobile devices are engaged to participate

remotely and use their devices to accomplish testing tasks under various realistic platforms instead of testing internally. Once issues are identified, crowd testers submit testing reports to developers through appropriate platforms. Crowd testers are often rewarded based on the severity of the reported issue (Wang et al., 2016). More importantly, crowdtesting can also help to solve the bottleneck of knowledge between app developers and testers to the large-scale deployment of mobile apps.

Crowdtesting has become an alternative to the traditional testing methods in respect of ensuring faster time to market. It has been used by both academia and industry to perform several types of testing, such as graphical user interface (GUI), functionality, usability, load, localisation, and security. However, the available literature demonstrates that there has been 'little work' done on crowdtesting in respect of compatibility testing (Alyahya and Alrughbh, 2017). This motivates the need for identifying the main research problem in this area in Section 1.1 and investigating this problem in the following Section 1.2.

1.1 Research Problems and Motivation

Several prior studies have investigated the fragmentation-induced compatibility issues (Huang, 2014; Lanui and Chiew, 2019; Liu, 2018, 2019; Moran et al., 2018; Starov and Vilkomir, 2013; Villanes et al., 2015; Zhang et al., 2015). These studies proposed automatically performing compatibility testing on as many mobile devices as possible, through artificial testing environments. Nevertheless, the solutions proposed in these studies still have low coverage and are insufficient to address the main issues because many compatibility issues cannot be fully explored using artificial environments and automatic testing tools (Knott, 2015b; Wu et al., 2017). Consequently, developers moved towards cooperating with traditional testing organisations and crowdsourced testing platforms belonging to large organisations and thus allocating a high amount of the budget to tests. Unfortunately, even after using testing organisations and platforms, compatibility testing as a solution for fragmentation issue remains a substantial challenge for app developers, especially for individual developers or small/medium-sized enterprises (SMEs). Therefore, further research into this area is required to investigate the reasons that led to difficulty in compatibility testing and to explore a more effective way of conducting compatibility tests that would be suitable for all developers. The literature reveals five main causes (gaps) behind the difficulties involved in compatibility testing and that hampered the success of prior studies in addressing fragmentation-induced compatibility issues.

- **G1:** The inability to fulfill large-scale app deployment and to cover the full breadth of global mobile devices the markets produce because of the high costs involved (Gao et al., 2019; Linares-Vásquez et al., 2017; Vilkomir, 2018).

- **G2:** A lack of knowledge about the behaviour and interaction of real users with apps causes issues with development (Dubey et al., 2017). Thus, developing a better understanding of human factors can help reveal compatibility issues earlier in order to help to ensure a higher quality of the apps when they are delivered (Alnawas and Aburub, 2016).
- **G3:** To date, architectures of mobile device platforms, such as Android, have not been standardised across mobile device manufacturers, and many of the differences in the architectures and the characteristics of the components of these devices are largely undocumented and not yet available for the public even in the "Android Development Source"¹ (Wnuk and Garrepalli, 2018). Evidently there is a need for more resources to aid the quick identification of those common issues relating to diverse architectures of mobile devices or OS versions, and how to solve such issues in reality (Wei et al., 2016, 2018).
- **G4:** It is challenging to generate an ideal coverage of test cases and/or scenarios that cover all possible behaviour of humans and the diverse hardware devices (Gao et al., 2019; Kong et al., 2018). More efficient testing techniques are therefore needed, to provide testers with better guidelines and instructions so as to perform effective testing to achieve more accurate results (Kowalczyk et al., 2018).
- **G5:** To date, most of the existing testing approaches have relied on indirect communication between developers and testers (through a middleman), which can result in several problems that lead to difficulties in the testing process. There is a strong need for a direct communication channel and effective coordination among developers and testers (Cruzes et al., 2016; Machado et al., 2016; Otolu, 2016).

No previous study has provided a compatibility testing solution addressing all the above gaps. Consequently, this has led to an increase in the fragmentation-induced compatibility issues over the past three years. Wei et al. (2016) emphasised that crowdtesting can play an effective role in finding a solution to this albeit that research studies in this area are still very limited (Alyahya and Alrugebh, 2017). It is still not very clear as to what extent crowdtesting can be used for compatibility testing, as it requires a considerable amount of human effort to conduct testing and record results for each device. On the basis of this, we have been greatly inspired to perform further investigations to explore the possibility of crowdtesting to help developers in performing effective large-scale compatibility testing which could address this research problem. Designing an appropriate "crowdsourced-based compatibility testing" approach to fully address all five aforementioned gaps could have a significant impact on preventing the rise of new compatibility issues and lead to the improvement of the overall

¹<https://source.android.com/devices/sensors/index.html>

outcome of the apps developed in the future. The next chapter will present the state-of-the-art work addressing the fragmentation-induced compatibility issues including existing compatibility testing and crowdsourced testing approaches, and their limitations in order to generate a novel solution for compatibility testing that would contribute to the existing knowledge in this field.

1.2 Research Aims and Objectives

This thesis aims to propose a novel crowdsourced-based compatibility testing approach to address comprehensively the issues mentioned in the previous section. The approach has been evaluated in terms of effectiveness, benefits (advantages), and overall satisfaction (intention to crowdsourcing/intention to participate), as well as drawing comparisons with the current state of the art. The proposed approach is mainly focused on five core aspects:

- **A1:** Testing features of mobile devices and/or OS versions to make sure they meet app standards. This will help understand the extent to which a specific feature on a mobile device can support a particular app functionality (Covering G3).
- **A2:** Investigating a new method that support detection of issues resulted from different human behaviour and interaction with apps (Covering G2).
- **A3:** Improving communication and cooperation between developers and testers during in the app development and testing process and increase Industry-Academia work collaboration (covering G5)
- **A4:** Distribution of compatibility testing to large-scale global testing communities to cover a broad set of mobile devices with a minimal cost, time, and effort (Covering G1).
- **A5:** Provide an insight into the architectures of mobile device platforms, useful testing scenarios, compatibility issues, causes, and possible solutions (Covering G3 and G4).

To accomplish the above aims, the following objectives need to be fulfilled:

- **OBJ1:** Review the literature concerning the current compatibility testing techniques and crowdtesting approaches in relation to the fragmentation issues;
- **OBJ2:** Identify the main criteria and requirements that must be included in the proposed crowdsourced compatibility testing approach and which will be the focus of this research.
- **OBJ3:** Investigate the developers' acceptability of working with public and anonymous crowd testers, with varied experience in the testing process and identify the main factors that could reduce their concerns regarding dealing with these crowd testers.

- **OBJ4:** Design and implementation of a public crowdtesting approach and its basic processes covering comprehensively G1-G5.
- **OBJ5:** Develop an appropriate assessment tool that supports the evaluation of the essential characteristics of the approach.
- **OBJ6:** Evaluate the benefits, satisfaction, and effectiveness of the public crowdtesting approach in addressing the fragmentation and compatibility issues in different contexts of developers and testers.
- **OBJ7:** Identify the potential advantages of the proposed public crowdtesting approach in comparison to the existing crowdtesting approaches.
- **OBJ8:** Summarise the findings of this thesis and formulate recommendations and guidelines to help other researchers interested in designing efficient crowdsourced-based solutions.

1.3 Research Questions

This thesis intends to address the following research questions:

- **RQ1 (Acceptability):** To what extent are mobile app developers keen to work with unknown testers with various levels of experience to perform their testing tasks? And what are their critical requirements to use the crowdtesting approach and the key factors to be considered to mitigate their concerns?
- **RQ2 (Effectiveness):** How effective is the proposed public crowdtesting approach in addressing the fragmentation-induced compatibility issues and allowing effective compatibility testing in comparison to the state-of-the-art approaches?
 - How effective is each process of the proposed crowdtesting approach?
- **RQ3 (Benefits):** Does the proposed public crowdtesting approach benefit both developers and testers? If so, is there a specific context where the public crowdtesting approach is particularly beneficial for developers and testers?
 - To what extent does our approach meet the need of the different developer and tester groups in their everyday work environment?
- **RQ4: (Satisfaction))** To what extent are developers and testers satisfied and keen to use the proposed crowdtesting approach??

1.4 Research Contributions

This thesis makes the following contributions:

- **C1:** It provides a comprehensive survey of the different technologies published in academic literature and the industry, that have been used to address the fragmentation-induced issues in the period (2017 to 2021). The body of knowledge of this survey produces a state-of-the-art comparison of compatibility testing and crowdtesting approaches, including the achievements, and the salient challenges of each (Chapter 2). As a result, two publications have been derived from this contribution: [P1](#) and [P2](#).
- **C2:** To our knowledge this is the first in-depth exploratory study into understanding developers' needs and measuring their willingness to work with members of the public as testers (Chapter 4). Through this exploratory study, we have managed to introduce some new knowledge and guidelines based on the main findings of this research in order to improve the crowdtesting process including:
 - The identification of the critical requirements of the public crowdtesting approach and factors that would address developers' concerns when working with unknown testers and help increase their confidence in them. These factors could significantly increase the developers' likelihood of working with public testers. Our requirements would also help the testing companies to leverage crowdtesting more effectively and efficiently. As a result of this, two publications have resulted from the C2 contribution: [P3](#) and [P4](#).
- **C3:** We proposed a new crowd-based compatibility testing that exploits public testers' effort, such as freelance and contracted testers and a community of end-users with different experience, with their devices to promote effective manual compatibility testing of Android and iOS platforms. The approach includes designing a public platform that enables the large-scale distribution of tests to enhance the overall mobile device test coverage with lower human effort, cost, and time compared to other approaches. In particular, the approach focuses on testing the features of all local mobile devices and OS versions to detect all potential issues that can affect apps. This aspect is deemed to have been overlooked by current approaches that focus more on testing the functionalities of apps only. The approach can also help discover other issues related to human behaviour and interaction with apps, potentially avoiding problems when end-users use the app. None of the existing practical approaches currently considers such an issue. This contribution has resulted in a publication: [P1](#).
 - We proposed a novel crowdtesting workflow that bridges the gap of direct communication and work collaboration between developers and testers (without a crowd

manager or leader) and supports the performance of an effective crowdsourced compatibility testing process in substantially less time and with less effort than current approaches. To the best of our knowledge, this is the first work that investigates the direct cooperation and interaction between developers and testers on mobile app testing.

- **C4:** Empirical evidence of the effectiveness of the proposed crowdtesting approach in addressing fragmentation-induced compatibility issues and facilitating large scale compatibility testing and its benefit and developers' and testers' satisfaction to use the approach in the future. To the best of our knowledge, this is the first empirical study that fully evaluates the crowdtesting approach and its sub-processes from both sides. This would be a base for other researchers in the future to evaluate the new crowdsourced-based approaches from both seekers' and crowd workers' sides. Through the analysis of the empirical studies results, we managed to demonstrate promising evaluation results, which act as the main findings in this research (see Chapter 7, 8, and 9):
 - Effectivity of the approach in large-scale mobile device compatibility testing, within a flexible and scalable manner, for tackling fragmentation-induced compatibility issues as compared to other state-of-the-art approaches.
 - The approach is beneficial and improves on state-of-the-art compatibility testing approaches for mobile apps regarding the reduction of human effort, cost, and time which are the crucial factors for the success of apps in current app markets, as well as delivering more in-depth insight into the compatibility issues through the implemented wiki.
 - High levels of satisfaction and acknowledgement of the significant benefits of using the approach within the different contexts of developers and testers.
 - We have revealed interesting facts during evaluation of our approach (for more details see Chapter 7 and 8).
 - * Our results demonstrate that the diversity of end-users' and testers' experience is beneficial when aiming to obtain more realistic results and to deliver higher-quality apps.
 - * Bridging the gap and increase Industry-Academia work collaboration.
- **C5:** This research contributes to crowdsourcing research and practices in both academia and industry by providing a set of guiding principles for researchers and practitioners to advance the useful and practical design of crowdsourced-based solutions and improve the conclusiveness of communication, work collaboration, and knowledge sharing support on crowdsourcing platforms. These principles are extracted from our experience building

the proposed crowdtesting approach and the findings and discussion of our empirical evaluation studies (see Chapter 10, Section 10.3).

1.5 Publication Summary of the Thesis

The contributions derived from this thesis have been published in peer-reviewed journals and conferences. Table 1.1 provides a detailed description of parts of the thesis published under each chapter, the names of journals/conferences, and the awards given to these publications.

1.5.1 Publications Arising from this Research

The conference and journal papers published from this thesis are ordered by year of publication and listed as follows:

- **P1:** Naith, Q. and Ciravegna, F., 2018, July. Hybrid crowd-powered approach for compatibility testing of mobile devices and applications. In Proceedings of the 3rd International Conference on Crowd Science and Engineering (pp. 1-8).
- **P2:** Naith, Q. and Ciravegna, F., 2018. Mobile devices compatibility testing strategy via crowdsourcing. International Journal of Crowd Science.
- **P3:** Naith, Q. and Ciravegna, F., 2019, October. The Key Considerations In Building A Crowd-testing Platform For Software Developers. In Proceedings of the 4th International Conference on Crowd Science and Engineering (pp. 50-57).
- **P4:** Naith, Q. and Ciravegna, F., 2020. Definitive guidelines toward effective mobile devices crowdtesting approach. International Journal of Crowd Science.

Table 1.1 Published works under each chapter and names of journals and conferences.

Chapter	Published work	Journal or Conference
-	Comparison between existing compatibility testing approaches and their limitations.	The 3rd International Conference on Crowd Science and Engineering (ICCSE 2018), for which it was awarded "Best Student Paper Award" .
Ch: 2	- State-of-the-art comparison between existing crowdtesting workflows and their shortcomings. - Comparison between current crowdsourcing models.	- The International Journal of Crowd Science (IJCS 2018).
Ch: 4	The exploratory study for gathering developers' needs and its results (fully published and presented).	The 4th International Conference on Crowd Science and Engineering (ICCSE 2019). and then extended and published in The International Journal of Crowd Science (IJCS 2020).
Ch: 5	The proposed crowdsourced compatibility testing workflow and its features and functionalities.	The 3rd International Conference on Crowd Science and Engineering (ICCSE 2018), for which it was awarded "Best Student Paper Award" .
Ch: 6	The CSTS-Q questionnaire tool for evaluating crowdtesting approach and its development and evaluation steps.	The ACM Conference on Human-Computer Interaction (CHI 2021) (Sept. 10, 2020).
Ch: 7, 8 and 9	The two empirical studies for evaluating benefits, satisfaction, and effectiveness of our approach, as well as their finding and discussion. (fully published).	The 24th ACM Conference on Computer-Supported Cooperative Work (CSCW 15-Oct 2021). The 43rd International Conference on Software Engineering (ICSE 18-Oct 2021).

1.6 Thesis Structure

The structure of this thesis consists of ten chapters. The details of each chapter are described below:

Chapter 2, (Literature Review), provides a review of the relevant literature work with the aim of identifying the current challenges associated with mobile app testing, focusing especially on fragmentation-induced compatibility issues. An assessment of the current state-of-the-art is thus provided, which arguing that the literature currently fails to provide a testing approach that can fully address these issues. The subsequent section gives a critical review of all the limitations found in the previous research that hamper the resolution of fragmentation- induced compatibility issues. The chapter also focuses on presenting an overview of the crowdsourcing's role in software engineering research, especially in software testing. A review of the literature regarding crowdsourcing for mobile app testing, alongside its contributions to reductions in fragmentation-induced compatibility issues, is also included. Additionally, an overview of the most popular crowdtesting platforms used for mobile apps testing was presented. Lastly, the state-of-the-art crowdtesting workflows used by these platforms or other testing organisations are supplied to both academia and industry. An understanding of these topics gives an insight into the research objectives, questions, and the appropriate approach for investigating the identified research questions.

Chapter 3, (Research Methodology), introduces and justifies the research design and methodology, including the methods (conducted studies) used to investigate our research questions aiming to help developers perform more effective compatibility testing. The chapter also provides a brief discussion of the data collection and data analysis methods for the studies conducted in this research.

Chapter 4, (Requirements of Crowdtesting Approach), presents and discusses the exploratory study conducted for understanding and gathering developers' needs to use the proposed approach. The methods used for collecting data and its analysis are also described. Then, a presentation of the results and the main finding of the exploratory study is given. A clear discussion of the results is provided regarding the developers' acceptance to work with public and testers with varied experiences. Lastly, the chapter outlines the main functional and non-functional requirements necessary for building a more effective crowdsourced testing approach, reducing developers' concerns about dealing with public crowd testers.

Chapter 5, (Design and Implementation of Crowdtesting Approach), discusses the design and implementation of the proposed public crowdtesting approach in detail. It begins with an explanation of the entire crowdtesting working mechanism. Followed by a detailed description of the design and implementation of each process with its requirements.

Chapter 6, (Empirical Evaluation Studies: Data collection and Analysis), provides an explanation of the whole evaluation method. The chapter presents the underlying aspects required to assess the proposed crowdtesting approach. Then, a discussion of the main steps taken in developing the CSTE-Q tool is given. The processes used to evaluate the CSTE-Q are also discussed, including the question refinement process, reliability and validity tests. The chapter also describes the method of distributing the survey questionnaires, population sampling, and studies procedure. Finally, the chapter explains the analysis methods used for analysing data of both empirical studies.

Chapters 7 and 8, (Empirical Evaluation Results (Study I and II)), provide detailed information about the main findings of both empirical studies on evaluating the satisfaction of, benefits from, and effectiveness of, the proposed crowdtesting approach. Chapter 7 presents the main results and findings from the empirical study conducted with the developers. On the other hand, Chapter 8 presents the results and findings from the second empirical study, conducted with the testers. Similarly, both chapters offer a descriptive and inference statistical analysis of the relevant data.

Chapter 9 (General Discussion), provides a broader discussion of the main findings of both studies. An in-depth comparison is made between the different contexts of developers' and testers' towards assessing the benefits of public crowdtesting approach and thus their satisfaction with such approach is provided. The chapter also demonstrates the effectiveness of the proposed approach in reducing the effect of fragmentation-induced compatibility issues and its role in facilitating compatibility testing. Furthermore, the effect of direct communication between developers and testers (without a middleman) on the testing process is investigated. Lastly, the potential advantages derived from adopting our approach are identified in relation to enhancing the overall mobile app testing process, in comparison to other approaches.

Chapter 10 (Conclusion and Future Work), gives an overall summary of the thesis. A brief discussion of the conclusions drawn from the research is presented, including a discussion on the extent to which the findings can be generalised as well as the contribution of these findings to the field. Furthermore, the chapter also provides a list of principle guidelines and recommendations to assist other practitioners and researchers in the domain. The chapter then concludes by outlining the research limitations and possible directions for future work to elaborate on this research.

A State-of-the-Art Review

2.1 Introduction

The previous chapter presented the general research problem, the research aims and objectives, and research questions. This chapter explores the problem in-depth, assesses current practices and activities, and identifies the existing challenges within the previous related studies. The chapter started with an overview of the concept of mobile compatibility testing, including dimensions, challenges, and critical requirements for developing effective compatibility testing solutions. An evaluation of the current state-of-the-art compatibility testing solutions is provided. This is followed by providing the critical limitations found in the previous research, which hamper the resolution of fragmentation-induced compatibility issues. The chapter also provides a review of the literature work regarding crowdsourcing for mobile app testing. Next, an overview of the crowdtesting workflows used by current approaches, tools, and industrial practices is provided. A clear description for the limitation of each workflow is also included. An in-depth review of current crowdsourced compatibility testing approaches is also exposed in this chapter. Additionally, a comparison between existing industrial crowdtesting platforms used for mobile apps testing is presented, including those that perform and those not performing compatibility testing. Then, the limitations of the current crowdtesting approaches published in the literature and the available industrial crowdtesting platform practices are described. Lastly, the chapter provides a clear description of the limitations and challenges faced by developers and testers in the crowdtesting process in general.

2.2 Mobile Compatibility Testing: Dimensions, Challenges, and Requirements

Mobile apps are often suffering from the appearance of many compatibility issues, which can hinder the apps from successfully running across different mobile devices, in particular Android devices. (Kong et al., 2018; Wei et al., 2016; Zhang and Cai, 2019). This is mainly due to the mobile device and operating system (OS) fragmentation, especially in the Android ecosystem brought by its open-source nature, where every mobile device manufacturer (e.g., Huawei, Samsung, LG) can have its own customized mobile device (e.g., for supporting specific low-level hardware) from the original Android systems (Fazzini and Orso, 2017; Wei et al., 2016). Compatibility testing is a solution that validates whether the mobile apps can work well and effectively on all different mobile devices with various OS versions, appliances and features, and in different environments before releasing them to the markets (Amen et al., 2015; Cheng et al., 2015; Ham and Park, 2011; Wei et al., 2016). Mobile compatibility testing focuses on three key compatibility dimensions (Zhang et al., 2015); **Platform/OS compatibility** is related to ensure the mobile apps work correctly on different mobile platforms and with different OS versions, such as Windows mobile, Android, and iOS, etc. **Device feature compatibility** is related to ensuring that the mobile app is compatible with different hardware features of mobile devices such as RAM, CPU, GPS, sensors, screen size, camera, net connection, etc. **Native API compatibility** refers to whether a mobile app is compatible with various versions of the native API programs, such as interfaces of other software development kits. As pointed out Vilkomir and Amstutz (2014); Wei et al. (2018); Zhang et al. (2015), it is costly, exhausting, and time-consuming to perform compatibility testing of apps across all existing mobile device and OS versions.

According to the review of relevant works in literature, we observed four major root causes for the difficulty to perform effective compatibility testing and the continued arising of compatibility issues.

- Most mobile app developers are individual developers or small and medium-sized teams. Therefore, it may be difficult and expensive for them to have many mobile device models and OS versions to be tested (Fazzini and Orso, 2017; Wei et al., 2016)
- Lack of knowledge about internal complexity of mobile devices and/or the guidance for testing specific apps functionalities (especially that related to sensing, education, health monitoring, banking, tracking apps, etc) (Samuel and Pfahl, 2016; Wnuk and Garrepalli, 2018);
- The existing testing approaches and tools do not consider the difference of users behaviours and their interaction with mobile app (especially the app that is using sensors) (Leotta et al., 2019);

- Inconsideration of all three dimension of compatibility testing; mobile platform compatibility, mobile device feature compatibility, and native API compatibility.
- The existing automated testing approaches and tools are insufficient to test all possible real-life scenarios or to capture the different aspects of mobile devices ([Knott, 2015a,b](#)).

Through the literature review conducted in this area, we found that although several published research studies have investigated fragmentation issue and mobile compatibility testing, little is known on the main requirements that need to be considered when developing new compatibility testing solutions. The below requirements which we observed would help to provide better compatibility testing solutions that:

- Support the global-scale distribution of the test.
- Cover all three different mobile compatibility testing dimensions: platform, device feature, and native API compatibility.
- Perform the two types of compatibility testing; Forward Compatibility Testing (FCT) to check whether the app is compatible with the newer device models and OS versions. Backward Compatibility Testing (FCT) to verify if the app is compatible with the older device models and OS version ([Professional QA Group,2020, 2020](#)).
- Support the accessibility of individual developers, small teams, and small and medium-sized organisations to all different types of mobile devices with a variety of OS versions.
- Support sharing knowledge and guidelines relevant to compatibility testing to assist developers in the mobile apps development lifecycle and improve testing knowledge among developers and testers.
- Support all different types of mobile apps and capture all real-life testing scenarios.
- Capture all unexpected compatibility issues that may result from different behaviours and interaction of real users with the app.
- Contain a realistic testing environment, not a virtual/artificial environment to perform the test.
- Perform the test in two different levels: testing the physical feature of the mobile device and the piece of the source code of the app.

2.3 Compatibility Testing Solutions

Several research studies have proposed solutions for addressing fragmentation-induced compatibility issues and facilitating compatibility testing in different ways. Some of them developed

as automated testing frameworks; other methods have been developed to prioritize devices for compatibility testing, others developed to perform automated testing, and others to perform the test over the cloud server. Section 2.3.1 describes these different proposed solutions, while Section 2.3.2 outlines the main challenges of the discussed solutions.

2.3.1 Existing Mobile Compatibility Testing Solutions

To address the fragmentation, facilitate compatibility testing, and ensure that new developed Android mobile apps will work well on a variety of devices, several automated testing frameworks have been proposed such as Robotium ¹, Appium ², UIAutomator ³, and Android Espresso ⁴. These testing frameworks allow developers to automatically run their test cases for the different apps on different mobile devices under multiple mobile platforms, after encoding them.

To further alleviate fragmentation issues and allow effecting compatibility testing in another way, Google has provided the manufactures with a set of Compatibility Test Suites (CTS). This contains a huge number of test cases to assist them to ensure whether their Android mobile devices are in compliance with the basic Android compatibility standards (Source, 2021). However, testing all of these test cases would take more time to complete the CTS tests (Liu et al., 2016). Therefore, LIU et al. (2018) proposed an approach to split the compatibility test suites provided by google into multiple testing tasks that can be executed on different Android devices concurrently through a developed cloud-based testing platform. They found that this approach reduced the time taken to perform CTS tests and increased the number of tested devices.

Other studies proposed techniques to reduce the problem of fragmentation and aid developers in selecting the best devices to test their new mobile apps on (Khalid et al., 2014; Lu et al., 2016; Wei et al., 2019). For example, Khalid et al. (2014) proposed a method to prioritize and select mobile devices for testing according to the user ratings/reviews of other mobile apps from the same category. The findings demonstrated that the approach could effectively reduce the developers' efforts to test the new app on several devices since it provides them with a list of mobile devices that greatly impact user ratings. Lu et al. (2016) proposed a PRADA technique to select and prioritize a set of major Android mobile device models to test different types of app based on the extracted usage data of similar apps. They found that the PRADA can effectively work over Game and Media apps and improve the work of (Khalid et al., 2014). A different prioritisation solution was provided for API device correlations. Wei et al. (2019) proposed a PIVOT technique that automatically detects API-devices correlations of fragmentation-induced compatibility issues from existing Android

¹<https://github.com/RobotiumTech/robotium>

²<https://appium.io/docs/en/writing-running-appium/running-tests/>

³<https://developer.android.com/training/testing/ui-automator>

⁴<https://developer.android.com/training/testing/espresso>

mobile apps. The PIVOT technique extracts and prioritizes correlations of API-devices from the provided set of Android apps. The evaluation shows that the PIVOT technique can effectively prioritize valid correlations of API-device for Android app corpora gathered at different time.

Another type of solution provided is related to automated compatibility testing approaches. A number of research studies have proposed an automated compatibility testing approach (Karlsson et al., 2021; Ki et al., 2019; us Saqib and Shahzad, 2018). For instance, us Saqib and Shahzad (2018) proposed a model-based approach (MBT) named TriTest to perform automated compatibility testing of the mobile app. In TriTest, the test cases are easily generated based on the developers' created model. They conclude that developers can be used the TriTest to perform compatibility testing on existing and new developing mobile apps. Ki et al. (2019) proposed an automated UI compatibility testing tool called Mimic. It has designed to test the UI behaviour of mobile app and capture the visual differences of mobile apps' UI during runtime across different Android devices and OS versions. Mimic allows developers to execute forward and backward compatibility testing easily for their Android apps. Additionally, Mimic supports different testing strategies such as parallel, randomized or sequential testing. They found that Mimic is effective in handling the UI compatibility issues correctly in real Android apps and minimizes the development burden for app developers. Karlsson et al. (2021) proposed a model-based approach to execute GUI automated compatibility testing of mobile apps. This approach quickly generates test cases that can be executed concurrently on different real mobile devices hosted in the cloud. They found that their approach produces a high coverage of all parts of the mobile app related to user interactions.

The work on mobile fragmentation also led to the proposal of several cloud-based compatibility testing approaches (Chen et al., 2018a; Lanui and Chiew, 2019; Liu, 2018). For example, Chen et al. (2018a) proposed a new compatibility testing tool named ICAT for testing both Android and iOS apps. The ICAT consist of the main cloud server and different IoT devices, including gateway, heterogeneous mobile devices with different OS versions, sensors, and Wi-Fi. It enables the tester to select the target environment, a particular version combination of the IoT devices and then performs compatibility testing for the registered devices. A test report that is describing the issues is offered once the testing is completed. Their experimental results demonstrated that the ICAT could enhance test coverage of mobile devices and reduce test costs. To ensure the clarity of content presentation and correct interaction behaviours of the multimedia apps on Android devices with less time and effort, (Liu, 2018) proposed another cloud-based testing (CTP) platform to automatically test Android multimedia apps against a scalable number of real mobile devices in parallel. This approach allows developers to automatically automate the user interactions with the target app and receive a record video and screenshots regarding the compatibility once the test is completed to ease the identification of uncovering potential issues in the apps. Their

evaluation shows that the CTP platform can effectively reduce the time and effort required for testing and ensuring the compatibility of multimedia apps on various Android devices. [Lanui and Chiew \(2019\)](#) proposed a cloud testing method for addressing the fragmentation issue and performing remote automated compatibility testing on numerous Android devices. This method allows the public members to share their mobile devices on the central system to be accessible for the developers. The developer who needs to perform the compatibility testing will connect to the system and run the test under any mobile devices connected to the system. It also allows the developer to reuse the existing tests on any device at any time. Furthermore, this method allows device providers to switch their access to networks in real-time to adapt to the testing requirements. [Lanui and Chiew \(2019\)](#) found that this method has the potential to address the challenges of compatibility testing of Android devices in terms of covering more devices and OS versions in a flexible manner compared to previous cloud-based compatibility solutions like CTOMS ([Starov, 2013](#)), AppACTS ([Huang, 2014](#)), and RTMS ([Huang and Gong, 2012](#)).

2.3.2 The Limitation of Existing Compatibility Testing Solutions

Although all the compatibility testing solutions discussed in Section 2.3.1 have attempted to address the fragmentation-induced compatibility issues and aid app developers to perform the compatibility testing process easier, there are still several challenges and limitations that hamper their complete success. This section exposes these challenges and limitations as follows:

- The automated testing frameworks are difficult as it requires special skills and a massive effort from testers or developers to manually encode the special test scripts for each testing ([Wu et al., 2017](#)).
- The coordination issue is the main drawback of cloud-based testing solutions where such solutions do not provide the developers or testers enough control in the selection of the devices that are required to be tested which are connected to the cloud ([Lanui and Chiew, 2019](#)). In our opinion, this is because the registered devices might be underused by other developers, which can cause delays in the testing process.
- The mimic system provided by [Ki et al. \(2019\)](#) is not suitable for detecting UI compatibility issues for all types of mobile apps such as sensor-centric apps (e.g., games and tracking)
- The automated and cloud-based solutions can not find all compatibility issues as they do not consider all different real-life testing scenarios. These compatibility issues that result from different mobile device configuration and different users interactions with the app. ([Dubey et al., 2017](#); [Gao et al., 2019](#); [Knott, 2015a,b](#); [Kong et al., 2018](#); [Leotta et al., 2019](#)).

- It is challenging to ensure app compatibility through the test prioritization approach as the app behaves slightly differently on the devices with various software or hardware environments, even those devices from the same manufacturer (Ham and Park, 2014; Samuel and Pfahl, 2016).
- Most of these solutions are focused on handling only one dimension of compatibility testing: mobile platform compatibility, mobile device feature compatibility, and native API compatibility.
- Mobile apps can not be tested sufficiently by cloud-based testing services, as they can not provide the complete set of all mobile devices and OS versions used by actual users, leading to many compatibility issues after releasing the app (Wu et al., 2017).
- The method provided by Lamui and Chiew (2019) does not support compatibility testing of all Android OS version and do not provide sufficient and clear summary of results, which are required developers more time to analyse and verify the test results.
- Most of the cloud-based testing services provide record/replay techniques to reduce testers effort, accelerate the testing process, and reduce spending of the budget by developers. However, the test cases that are automatically generated by these techniques are not sufficient and effective to detect all compatibility issues of the app as it may become weaker or break when replayed on different platforms (Knott, 2015b; Wu et al., 2017).

2.4 An Overview of Current Crowdtesting Workflows

The review of the current relevant literature (Alyahya, 2020; Alyahya and Alrugebh, 2017; Alyahya and Alsayyari, 2020; Gao et al., 2019; Liu et al., 2019; Vishwakarma et al., 2020) yielded that there are two crowdtesting workflows: "Manual Crowdtesting Workflow" and "Automated Crowdtesting Workflow" which can be seen and understood from a simple high-level view as appears in Figure 2.1 and 2.2. In general, these two crowdtesting workflows rely on indirect interaction between the developer who provides the mobile app for testing and the crowd tester who performs the test. This means that several activities are required to be completed by the middleman/QA member (leader and/or manager) to provide feedback or essential information to testers and to collect testing results. More information about these two workflows can be found in Section 2.4.1 and 2.4.2).

2.4.1 Manual Crowdtesting workflow (M-CT)

We now present the "Manual Crowdtesting workflow (M-CT)", which involves four main pillars: the developers, testers, managers and leaders. In this workflow, most of the processes

are done manually and the platform does not have a role in automatically performing any activities without developer, leader, or manager interference. According to [Gao et al. \(2019\)](#); [Vishwakarma et al. \(2020\)](#) this workflow has been used by most of the current crowdtesting companies reported in ([Alyahya, 2020](#); [Alyahya and Alrugebh, 2017](#)), which are presented in Section 2.7. Figure 2.1 summarises all the activities of the M-CT workflows. All activities within this workflow are described below:

A) Submitting the project The developers define the mobile app and specify the app's business needs and specifications including the targeted app, types of testing required, mobile platforms, OS versions, and required testing automation tools. The developer then selects the appropriate manager from the pool of manager within the community. Once the manager receives the project, the manager must send back to the developer their agreement to work on it.

B) Reviewing the project and selecting testers: Once the manager is selected to supervise the project. The manager will need to perform a set of processes: (1) reviews the project's requirements; (2) convert the project to a set of tasks; (3) design the testing plan, which involves information about the estimation of the testing time and the payment cost; (4) select a set of appropriate testers from the community to perform the test; (5) announce the test to the testers.

C) Reviewing tasks requirements: Once the testers receive testing tasks, they will review the test requirements and decide to accept or reject the test. If the tester accepts the test, s/he will submit a confirmation message to the manager that includes the estimation time needed to perform the test.

D) Assigning test team leader: Once the manager receives the confirmation messages from all the testers who accept performing the test, the manager will directly create a team and assign a leader from the community for that team and send information about the leader to the testers.

E) Performing the test and submitting reports: Once the leader receives the invitation, s/he will immediately ask testers to perform the test and then submit the results to the leader through the management tool (e.g., JIRA) determined by the leader and manager.

F) Reviewing and validating testing reports: Once the testers perform the tests and submit all the results, the leader will review these reports and evaluate them based on the priority/severity of the issues and send the feedback of the evaluation, including the rating score of each tester, to the manager.

G) Preparing final report: Once the manager receives the evaluation feedback, the manager will need to update the testers rating and then prepare the final reports, specifying the payment cost for each tester and then sending the final report to the developer.

H) Reviewing final report and rewarding testers: Once the developer receives the

final report, the developer will validate each result, pay testers for each valid issue and then finish the testing lifecycle.

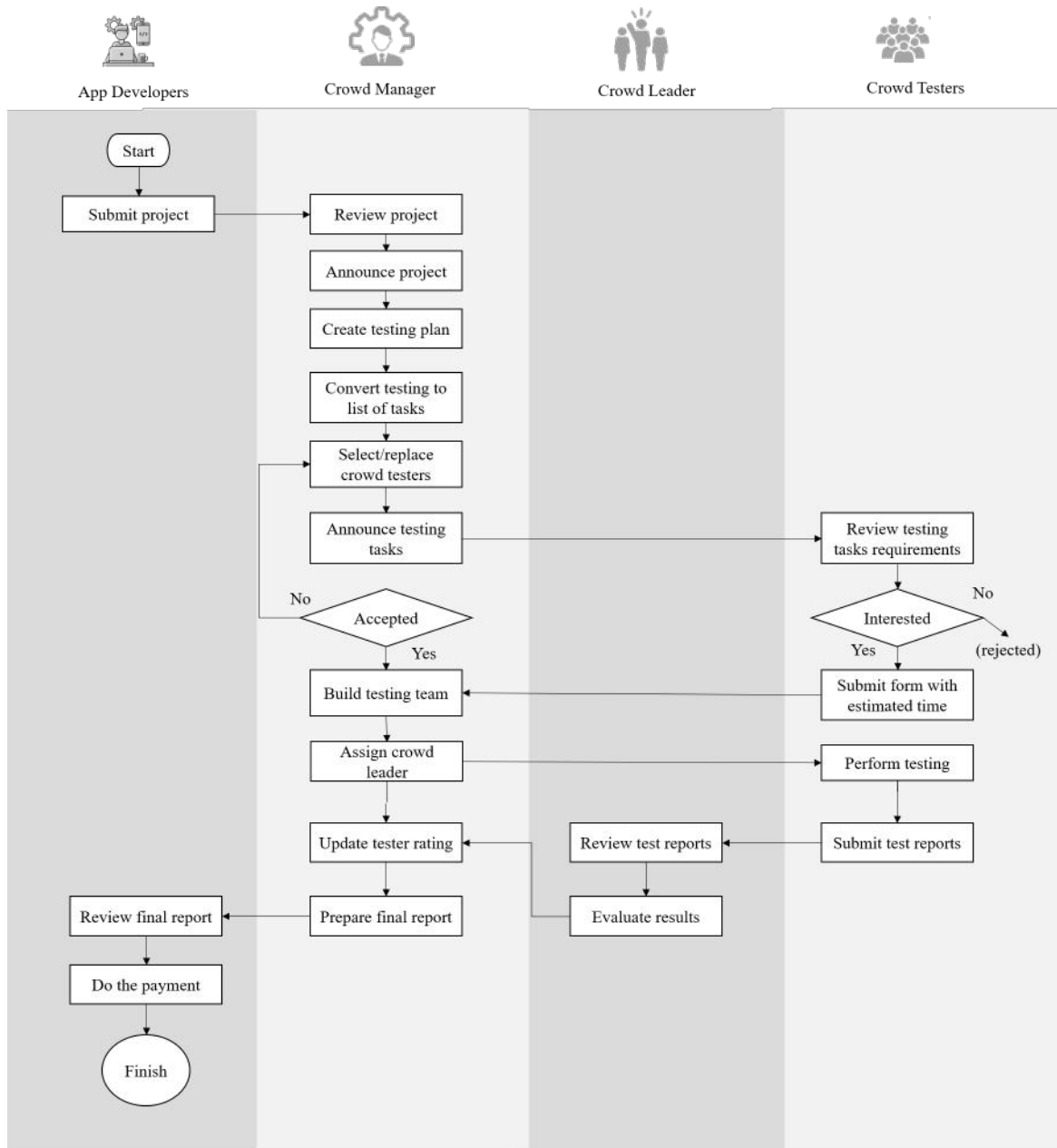


Fig. 2.1 The activities of the manual crowdtesting workflow (M-CT) [adapted from [Alyahya and Alrugebh \(2017\)](#)]

2.4.1.1 Limitations of Manual Crowdttesting workflow (M-CT)

According to [Alsayyari and Alyahya \(2018\)](#); [Alyahya and Alrugebh \(2017\)](#); [Vishwakarma et al. \(2020\)](#) and the analysis of the M-CT crowdtesting workflow, three main challenges were identified which are barriers to the success of the crowdtesting process:

- Lacks a method for monitoring and detecting the availability of the manager or leader to supervise newly announced testing projects. This could lead to inefficient distribution of the projects to available managers or leaders.
- The testing team and suitable testers for each project are chosen manually, which most likely lead to time-consuming and several delays.
- Lacks a mechanism to control the testing environment and the number of projects assigned for each tester. Testers may agree to participate in the testing life cycle but fail to send their test report on time (e.g., work pressure), which can negatively affect the whole project's testing time and its delivery to the market.

2.4.2 Automated Crowdttesting Workflow (Auto-CT)

Here we discuss the "Automated Crowdttesting workflow (Auto-CT)". The authors in ([Alyahya and Alrugebh, 2017](#)) carried out some improvements on the M-CT workflow to provide better crowdtesting services by proposing some processes that automatically done by the platform as described in Figure 2.2. These improvements are:

- Proposing a service that automatically assigns managers and leaders to the newly defined and announced projects based on the best availability of the leader or manager.
- Enabling an automated allocation method of testers according to the project requirements.
- Providing a service that automatically controls the testing environment by preventing test invitations from being sent to non-active or non-productive testers and notifying invited testers to perform the accepted task within a specific time.

Based on these improvements, this workflow now involves five main pillars: developer, manager, leader, testers, and platform. In this workflow, most of the activities are done automatically by the platform. Based on our literature review, we found that some well-known crowdtesting companies recently started following the Auto-CT workflow such as uTest, Crowdsprint, Crowd4test, Testbirds, and TestArmy. The main processes of this workflow which adapted from ([Alyahya and Alrugebh, 2017](#)) are discussed as follows:

A) Submitting the project: The developers define the whole app for testing with the identification of the business needs and goals. This is done by sending a testing request, including the required information: the targeted app, types of testing required, mobile platforms, OS versions, and required testing automation tools.

B) Selecting manager: The platform checks the availability of managers to supervise the new project. An automatic notification is sent to available managers based on testing requirements. Once they receive the notification, they must send their agreement to work on the new project.

C) Nominating manager: Once managers accept the project, they must provide an estimation for the actual time they shall be ready to start supervise the testing process of the project. Based on this, message send through the platform to the developers that includes the names of the managers who have accepted to supervise the test. The most suitable manager is selected by the developer to be the main manager of the new project.

D) Reviewing the submitted project by manager: The selected manager receives a notification message from the developer to supervise the project. The manager reviews the project's requirements and then designs the testing plan, which includes the estimation of the testing time that will be taken and the testing cost.

E) Dividing the project into a set of tasks: Once the manager reviews the requirements of the project and designs the testing plan, the manager will break the testing project into a set of small tasks to facilitate the testing process. These small tasks are later stored under that project.

F) Selecting testers from the community and announcing the tasks: After the manager has broken down the project into smaller tasks, the platform will automatically inspect the availability of testers and send invitations to the most appropriate testers based on several factors such as geographic location, the mobile device type, and OS versions.

G) Reviewing tasks specification by testers: Once testers receive the test invitation, they will review the tasks' requirements and then send their acceptance with the estimated time for conducting the test and submitting the test results.

H) Building test team: The manager will review the list of the available testers interested in performing the test and verify their information to ensure that they meet the required qualifications. After that, the manager builds the testing team and chooses the suitable testers for each task in the project.

I) Assigning test team leader: In this process, the manager selects a leader from the list of available leaders determined by the platform to work under the manager's guidance. After that, the selected leader receives a list that includes all selected testers to conduct the test.

J) Executing testing and submitting reports: After testers conducted the test, they will submit the testing reports, including discovered issues, to the leader using a management tool (e.g., JIRA), which is specified by the leader and manager of the project.

K) Reviewing and validating testing reports: Once all testers submitted their testing

reports, the leader will review and evaluate the priority/severity of the issues and then alter the report status to "Pending Approval" or "Pending Rejection".

L) Updating testers rating by manager: After the leader evaluates testing reports submitted by all testers and validates their reported issues, the leader will refer the issues to one of the following categories: Exceptionally Valuable, Very Valuable, Somewhat Valuable, or Rejected Issue. Based on the issue category, the manager is rating the testers' work and then storing the rating score in the platform.

M) Calculating the cost: The platform then calculates the total cost required for each tester based on their rating, sensitivity to the issue, and the possible price determined by developers.

N) Preparing final report by manager: According to the rating and calculation, the cost value needs to pay for each tester. The manager prepares the final report necessary information, including testers' names, results of each tester (which consist of the number of reported issues, issue priority), testers' rating score, and the payment value for each tester.

O) Reviewing final report and providing rewards: Once the developer receives the final report, they will inspect and validate each result. Then informs the manager about the "Approved" or "Rejected" issues. Once the manager updates the status of each report, the approved issue will be clustered and organised again in a new report with the cost of all approved report, and send it back to the developer. In this case, the developer will recheck it and then pay testers to finish the testing lifecycle.

2.4.2.1 limitations of Automated Crowdttesting Workflow (Auto-CT)

Our analysis of the current processes and activities used in Auto-CT workflow reveals several observations that we think can be a source of weaknesses of the workflow. These observations are:

- **More time to divided the project into smaller tasks:** The general concept of crowdttesting is breaking the big projects into small manageable tasks (Donepudi, 2020; Gao et al., 2019). In Auto-WF, this process requires more effort where the developers need to wait for the available manager to be selected and responsible for dividing the whole project/app into small tasks, which is time-consuming, especially when testing large projects.
- **Testing unnecessary tasks:** Partitioning the projects into multiple small tasks by managers might lead them to overlook testing some of the important tasks. It may also lead to executing unimportant test cases that will require developers to pay for it Haas et al. (2015).
- **Long time to assign team manager and leader:** Although assigning a manager or leader to each defined project is good and useful for the testing process in terms of

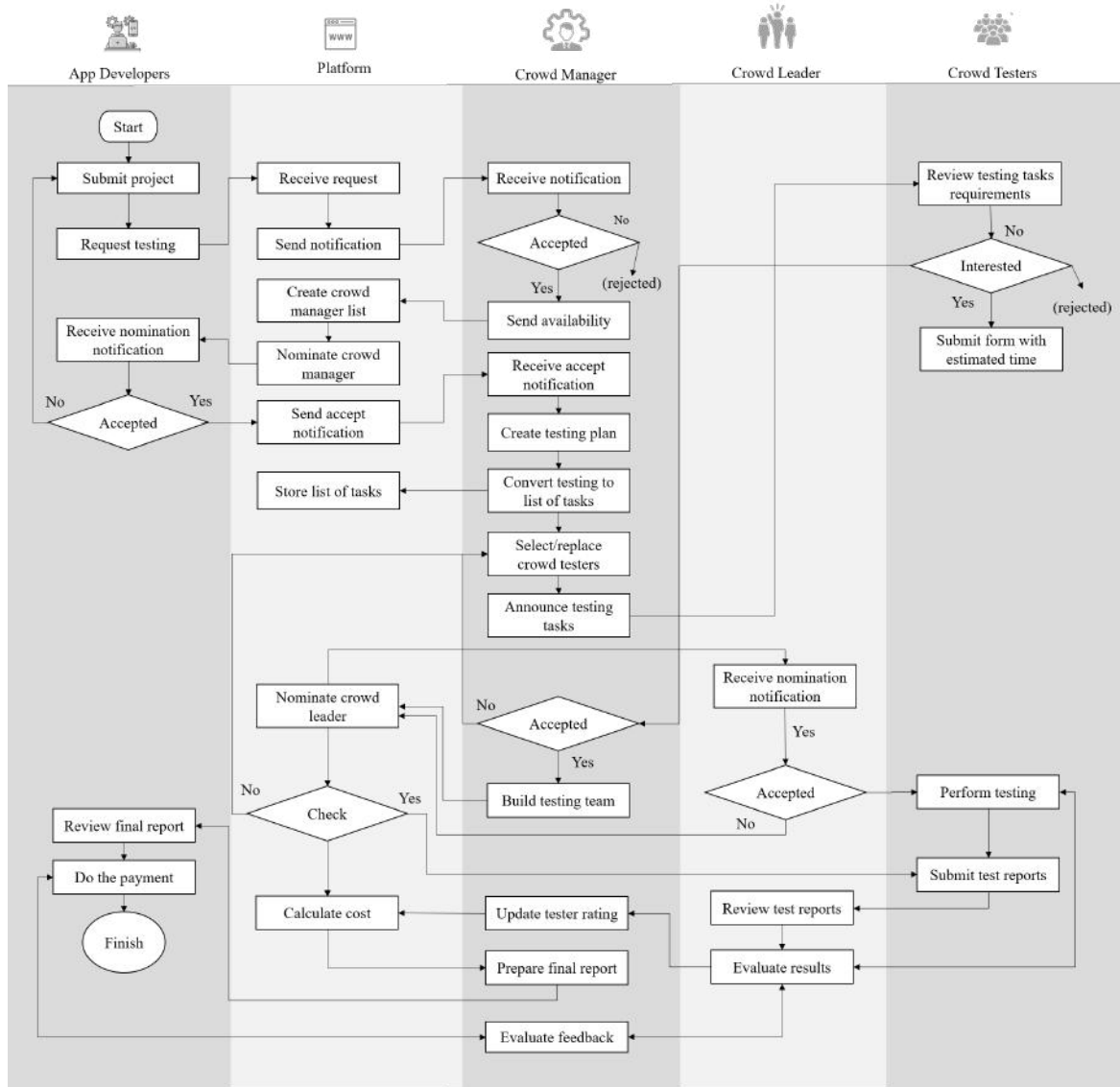


Fig. 2.2 The activities of the Automated crowdtesting workflow (Auto-CT) [adapted from Alyahya and Alrugebh (2017)]

organising the test. However, it might be a drawback due to two main reasons: (1) it causes more delay where the developer needs to wait more times for the manager and leader to be selected; (2) it is costly as it requires developers to spend more budget for their role in the test cycle, which might be expensive for the individual developers and small teams (Vishwakarma et al., 2020).

- **Lack of communication between developers and testers:** The Auto-CT workflow lacks direct interaction and communication between developers and testers. The developer first communicates with the manager, who communicates with the leader to reach the target tester and then provides the developers' feedback. This would, in turn, cause multiple delays and more effort to provide helpful information and feedback.
- **Lacks a method for tracking the published tasks' status:** For instance, the app might be developed for international use. In this case, the test needs to be achieved by a large-scale of different end-users worldwide. Additionally, the test may require to be tested on specific devices that run particular OS versions. Consequently, the developer needs to ensure that the test has covered the requirements and the target regions during the testing process to be able to alter task requirements. The Auto-CT workflow lacks such a service where the developer needs to wait for the manager to check each report and then prepare the final report to send to the developer, which can be time-consuming.

2.5 An overview of Crowdsourced Testing Research

Since 2006, crowdsourcing has rapidly become a new research domain. It has gained much attention in a variety of research fields. Several definitions of crowdsourcing have been proposed over recent years (Baba and Kashima, 2013; Erickson, 2011; Hosseini, 2014; Stolee and Elbaum, 2010), whereas the first definition was provided by Howe (2006). Most of these definitions imply that crowdsourcing is a process that enables tasks to be completed through the use of large groups of people from an online community rather than employees. Indeed, the use of crowdsourcing in software engineering has gained more attention over the last few years, especially in the domain of software testing. Mao et al. (2015a, 2017) provided an overview of crowdsourcing in the field of software engineering at that time. Another survey on the use of crowdsourcing technology in software testing has been provided by Leicht (2018), while a different survey has been provided by Liu et al. (2019) on existing research related to crowd-based mobile app testing, including the definition, advantages and disadvantages, workflow of mobile app crowd-based testing, and possible research directions for this area. Zhang et al. (2017b) provided a detailed discussion on the crowdtesting concept, services, common questions raised by practitioners, and the major issues and challenges of crowdtesting. In order to explore how crowdtesting and in-house testing can co-exist, Guaiani

and Muccini (2015) conducted a survey with crowd testers who work with different well-known crowdtesting companies to understand their perception on this matter.

To support coordination in crowdtesting activities, Alsayyari and Alyahya (2018) conducted a comprehensive review on existing research that aims to support coordination in crowdtesting activities in order to identify the coordination challenges in current industrial practices. The results reveal a lack of computer-based support for developers, testers, and project managers, as well as clients. As for comparing in-house testing with crowdsourced testing in terms of performance, Leicht et al. (2016b) provided two case studies regarding companies in different fields. The findings indicated that crowdsourced software testing offers more benefits in respect of speed, tester diversity and feedback in general, while the cost and quality of the testing were similar. In a later study conducted in 2017, Leicht et al. (2017) review crowdsourcing in practice. A clear comparison between crowdsourced testing and traditional lab-based testing of mobile app is also provided in (Zhang et al., 2017b). (Gao et al., 2019) provided an industrial study to investigate the use of crowdtesting for mobile apps compatibility testing through testing five Android apps using the Mootest and Kikbug platform. Based on their results, they provided an insightful analysis of the successes and challenges of using crowdtesting. To explore the crowdtesting concept from a broader perspective, Donepudi (2020) provided a comparative analysis between in-house testing and crowdtesting in terms of cost-benefit, level of expertise in the crowd, and presence of crowd manager or leader. The results showed that crowdtesting is very helpful regarding efficiency and cost-effectiveness. The results also showed the importance of a middleman figure for managing the connection between the crowdsourcer (e.g., a group, an individual, or a company) and crowd testers. The comparison between professional testers and beginner testers revealed that both of them have good capabilities based on their testing area.

As for the improvement of the crowdtesting process, Alyahya and Alrugebh (2017); Alyahya and Alsayyari (2020) investigated the current method of crowdtesting used by industrial platforms to discover the potential limitations and how to overcome them. The author provided some improvements for the detected issues to perform a better crowdtesting process. They evaluated the proposed improvements through scenario-based evaluation on domain experts. The results showed that the new improvements could strengthen the current crowdtesting practice. Moreover, a comprehensive review on crowdtesting has been carried out by Alyahya (2020) to determine the current research studies aiming to enhance and evaluate the value of using crowdtesting and the challenges that were determined by each study. Many research studies have also been published to enhance different sub-process of the crowdsourced testing process. For facilitating **testers selection process**, several testers recommendation methods have been proposed. Ye and Wang (2016) proposed a CrowdRec recommendation method to help developers select testers based on similarity of work and trust level of testers. (Xie et al., 2017) provided a Cocoon system to recommend testers

based on the test context (e.g., type of mobile devices and OS versions). Cui et al. (2017a) introduced a different testers recommendation method called ExReDiv, based on experience and capability of testers. (Cui et al., 2017b) proposed the MOOSE approach that considering the experience of testers and domain knowledge. Recently, (Wang et al., 2019b) introduced a recommendation method that improves all previous methods by considering the test context, experience, and capability of testers. As for **aggregating test reports and tracking the reported issues**, developers sometimes need more time to filter the submitted test reports and organise them in an orderly manner according to the priority of reported issues (Chen et al., 2020). To improve this process and the efficiency of inspecting each report, many widely used issue-tracking systems like Mantis ⁵, Jira/Atlassian ⁶, and Bugzilla ⁷, Zoho bug tracker ⁸, have offered keyword-search-based features to help developers filter and query similar reports. Most of the well-known industrial crowdtesting platforms have integrated at least one of these systems. Due to the different testing skills and experience of crowd testers, the quality of submitted test reports would be different, and also, some of these reports might be redundant Jiang et al. (2018).

To improve the **assessment process of test reports**, Jiang et al. (2018) proposed the Fuzzy Clustering Test Reports (TERFUR) approach to divide multi-issue test reports into multiple clusters and aggregate the reports that detail the same issue in one cluster in order to reduce the problem of having redundant reports. Instead of focusing on only clustering the reports that have similar issues in one group and detecting the duplicated reports automatically, based on textual descriptions, as in Jiang et al. (2018); Zhang et al. (2017a) found that the test reports may include short text descriptions, which will be insufficient for helping developers to understand the reports clearly. They suggested that identifying the information automatically from the screenshots would be much better. According to that, Wang et al. (2019a) provided a new approach that combines information from both TextUal descriptions and ScrEenshots, named SETU. The SETU help to automatically detect duplicate test reports submitted by testers over the crowdtesting process, reducing the triaging effort by developers. A different approach Crowdsourced-Test Report Aggregation and Summarization (CTRAS), has been presented by Hao et al. (2019) to improve the work in (Wang et al., 2019a) by leveraging duplicated test reports to enrich the content of issues descriptions and improve the efficiency of evaluating these reports. CTRAS can automatically collect the duplicated reports based on both screenshots and textual information and then summarizes these duplicate reports into a comprehensive report.

Regarding **incentivisation and rewarding of crowd testers**, in the literature, many incentivisation and rewarding scheme have been reported in various forms. An early study

⁵<https://www.mantisbt.org/>

⁶<https://www.atlassian.com/software/jira>

⁷<https://www.bugzilla.org/>

⁸<https://www.zoho.com/bugtracker/>

by [Muntés-Mulero et al. \(2013\)](#) summarised these rewarding models into four forms: the first is the competition model, which is based on *Best-gets-paid*; the second is *Pay-per-hour*, *pay per-day*, or *pay per-week* which is similar to the model used by several industrial crowdtesting platforms such as QA Mentor, Testlio, Rainforest, Passbrains, Bugwolf, and Crowdsprint; the third is *Pay-per-amount of work* as applied by Crowd4test platform; the last model is *Pay-per-quality of work* which is included within all of previous models in most of the platforms. During the last few years, two further models are introduced and used many current industrial crowdtesting platforms. Such models are *pay per-task* as in UserTesting, UserCrowd (UsabilityHub), TestArmy, Userfeel, Testbirds, and Ubertesters; *pay per-issue* which is the most popular model used by most of the existing platforms such as Applause (uTest), App Testify, Crowdsprint, TestUnity, Digivante (BugFinders), Bugcrowd, Test IO, StarDust, and DeviQA; *first-gets paid* as used only by MyCrowd QA.

2.6 Current Crowdsourced Compatibility Testing Solutions

Crowdtesting has been widely used in the mobile app testing domain. Several research studies have been conducted to develop crowdtesting solutions for addressing different types of mobile app testing, such as Functional testing, Localization testing, and Usability testing. A few research studies have focused on developing crowdtesting solutions to tackle the fragmentation issue and the difficulty of performing compatibility testing. ([Wu et al., 2017](#)) provided a crowdtesting tool named AppCheck for compatibility testing of mobile apps on android devices based on record/replay technique. This tool generates tests that automatically explore the behaviour of the apps on different mobile devices based on the event traces collected from the real users when they interact with the app over the internet through web browsers. It uses the collected events to re-run on the mobile devices of real users immediately. Their empirical evaluation results show that the AppCheck tool effectively detects different behaviours of the app and improves the state-of-the-art. [Almeida et al. \(2018\)](#) presented a new system named CHIMP that allows developers to test thousands of apps by thousands of real users worldwide through the virtualised mobile environment (on virtual Android devices) via a standard Web browser. CHIMP enables quick collection of human inputs for mobile apps such as user interactions, run-time traces, performance, network traffic, and user feedback. CHIMP demonstrated its feasibility and applicability to improve traditional app testing tasks. [Li et al. \(2019\)](#) developed a Crowdsourced Collaborative Testing (CoCoTest) platform that is specifically designed for testing android apps based on the concept of collective intelligence. The CoCoTest allows testers to capture a screenshot of the discovered issue and to write a short description creating a report for that issue. All submitted test reports are collected online and recommended to other testers in real-time to review, verify, and strengthen each others' reports. The results illustrated that CoCoTest could improve the quality of test reports submitted by testers who have different experiences and minimize

testing costs. [Binh et al. \(2020\)](#) presented TMACSTest platform for testing mobile apps using crowdsourcing. TMACSTest is model-based testing tool for mobile applications. TMACSTest provides volunteering internet testers with particular mobile app models to test and then collect reports with in-depth analysis of discovered issues to deduce relevant test results and increase test coverage to include more devices. Ongoing experimental work and evaluation will be completed in future work.

2.7 Current Industrial Mobile App Crowdttesting Platforms

The practice of crowdsourced testing has taken on great importance during recent years, which has created a sense of urgency for many industrial testing organisations to start using crowdttesting technology in their works. Through our in-depth investigation of current literature, 35 crowdttesting platforms were found that were developed to solve the issues of mobile apps testing from 2007 to 2021. All these platforms provide different testing services for both Android and iOS apps at the same time. In this research, 24 of 35 platforms have been discussed, and 11 of them have not been considered because (1) they were developed only for China and written in the Chinese language, which mad it difficult for us to understand the content (e.g., Testin, Baidu MTC, and Ce.wooyun, Mooctest and SoBug); (2) they were closed/unavailable, and we could not be able to find any resources for them at the time of investigation [Until Jun 2021] (Testcloud, Mob4Hire, Pay4Bugs, 99tests, crowdttesters and Testbats). Table 2.1 presents a list of all the 24 current crowdttesting platforms in order based on the year they founded in, headquarters, the number of available testers within their communities, and their download links. A comparison between these platforms is provided in Table 2.2. Many factors have been considered in this comparison, including the provided testing services, the testers invitation/selecting methods and selection criteria; the issue-tracking system used by each platform, and the payment method for the testers. All the information presented within Table 2.1 and Table 2.2 - 2.5 is extracted from the links/websites of these companies and from current literature ([Alyahya and Alrugebh, 2017](#); [Alyahya and Alsayyari, 2020](#); [Crowdttesting platforms, 2021](#); [Zhang et al., 2017b](#)).

From Table 2.2, 2.3, 2.4 and 2.5, it can be seen that only five of these platforms provide a service to perform compatibility testing in different ways: MyCrowd QA, QA Mentor, Crowd4test, TestUnity, and Crowdsprint. For example, **MyCrowd QA** provides a cloud-based compatibility testing service that runs the testing remotely through real devices of the testers registered in the cloud, which focuses more on performing UI compatibility testing. This service enables developers to upload mobile apps through the web interface, choose the target mobile device models and OS versions, and review the testing results report after testing is completed. **QA Mentor** provides a compatibility testing service named "lab compatibility testing". This means that the test is performed by testers with different experiences in a

testing lab on different types of real mobile devices with different OS versions for different mobile environments such as Blackberry, Android, and Apple iOS. Unlike MyCrowd QA and QA Mentor, the **TestUnity** provides an end-to-end cross-browser test automation solution to conduct compatibility testing in a virtual environment over a variety of mobile device models, OS versions, and API configurations. The test is performed by different testers from their testing community on registered devices over the cloud. The **Crowd4test** and **Crowdsprint** provides a similar service based on crowdsourced manual testing for compatibility testing of different platforms and mobile device specifications. These platforms perform two types of compatibility testing: backward and forward compatibility testing. It allows the developers to define the platform, the test environment that the application is expected to work on, and specify the number of mobile devices they want to test. The test is performed by crowd testers from their communities distributed across different locations based on the defined requirements and number or types of specified devices.

For the testers selection process, most of them followed the self-selection method for selecting and inviting testers from the community members of the platform by QA manager or leader based on matching testers' profiles with the task requirements. Only five of them are selecting suitable testers from the list of recommended and matched testers, such as Test IO, Testbirds, StarDust, Userfeel, and TestArmy. Three of them have expanded the selection to invite testers from the external communities (outside the platform) like Bugcrowd, Userfeel, and UserCrowd. As for the selection criteria, the majority of them considered the demographic information, test context, and experience as the main criteria for selecting the appropriate testers. Demographic information contains age, gender, language, location, daily hours online, technical proficiency, and employment status. Test context includes information regarding device models and OS versions. The experience involved information regarding the previous performed test and knowledge about the specific type of test (e.g., functional, security, usability).

Table 2.2 also shows that most of them integrated external issue tracking tools like JIRA to facilitate the testing process and aggregate and track reported issues during the crowdfunding process. Furthermore, most of these industrial platforms pay for testers per valid issue they discovered or spent time (e.g., hours, day, month); only a few of these platforms pay testers per completed task.

Table 2.1 List of exiting industrial crowdsourced testing platforms

	Platforms	Head Quarter	Year of Foundation	No. of Testers	Sources/URL
1	App Testify	Bangalore	2019	2,500	https://apptestify.com/
2	Bugwolf	Australia	2016	Unknown	https://bugwolf.com/
3	Crowdsprint	Australia	2016	100,000	https://crowdsprint.com/
4	Crowd4test	Bengaluru	2015	5,000	http://www.crowd4test.com/
5	TestUnity	Bangalore	2015	1,600	https://testunity.com/
6	MyCrowd QA	USA	2013	50,000	https://mycrowd.com/
7	Global App Testing	UK	2013	40,000	https://www.globalapptesting.com/
8	Passbrains	Switzerland	2012	50,000	https://www.passbrains.com/
9	Rainforest	USA	2012	50,000	https://www.rainforestqa.com/
10	Ubertesters	USA	2012	25,000	https://ubertesters.com/
11	Digivante (Formerly BugFinders)	UK	2012	55,000	http://www.bugfinders.com https://www.digivante.com/
12	Hiretesters (Formerly Crowdsourced Testing)	Canada	2012	77,200	https://hiretesters.com/
13	Testlio	USA	2012	4,000	https://testlio.com/
14	Test IO	Germany	2011	50,000	https://test.io/
15	Testbirds	Germany	2011	500,000	https://www.testbirds.com/
16	Bugcrowd	USA	2011	Unknown	https://www.bugcrowd.com/
17	StarDust	France	2011	Unknown	https://www.stardust-testing.com/
18	Userfeel	UK	2010	130,000	https://www.userfeel.com/
19	TestArmy	Poland	2010	7,000	https://testarmy.com/
20	DeviQA	Ukraine	2010	Unknown	https://www.deviqa.com/
21	QA Mentor	USA	2010	12,000	https://www.qamentor.com/
22	UserCrowd (Formerly UsabilityHub)	Australia	2008	100,000	https://www.usercrowd.com/
23	Applause (Formerly uTest)	USA	2007	500,000	https://www.utest.com/
24	UserTesting	USA	2007	Unknown	https://www.usertesting.com/

Table 2.2 Comparison between existing Industrial Crowdsourced Testing Platforms

Platforms	Testing Services	Tester Selection / Invitation	Selection Criteria	Reports/Issues Tracking System	Payment (Reward)
1 App Testify	Performance	Self-Selection (from the community)	Not mentioned	Atlassian Jira®	Pay-per-issue
2 Bugwolf	Acceptance	Self-Selection (from the community)	Tester Profile - Demographic Info - Device Info	Atlassian Jira® Bugzilla Trello	Pay-per-time (hour, day, month)
3 Crowdsprint	Functional Usability Security Regression Compatibility Localization Usability	Self-Selection + Suggested List (from the community)	Tester Profile - Demographic Info - Device Info - Experience - Skills Level	Atlassian Jira® HPE QC/ALM	Pay-per-issue Pay-per-hour
4 Crowd4test	Performance Localization Compatibility Crowdtesting Functional	Self-Selection + Suggested List (from the community)	Tester Profile - Device Info - Experience	Not mentioned	Pay-per-hour Pay-per-amount of work
5 TestUnity	Performance Security Usability Compatibility	Self-Selection (from the community)	Tester Profile - Demographic Info - Device Info - Experience - Domain Knowledge	TestUnity tool Atlassian Jira® Bugzilla	Pay-per-issue
6 MyCrowd QA	Usability Regression Functional Accessibility Compatibility	Self-Selection (from the community)	Tester Profile - Demographic Info - Device Info - Capabilities - Type of discovered issues	Not used	First-gets-paid

Table 2.3 Comparison between existing Industrial Crowdsourced Testing Platforms, Cont...

Platforms	Testing Services	Tester Selection / Invitation	Selection Criteria	Reports/Issues Tracking System	Payment (Reward)
7	Global App Testing Functional Localization Regression Usability	Self-Selection (from the community + external communities)	Tester Profile - Device Info - Domain Knowledge - Experience	Atlassian Jira® Trello DoneDone Asana	Not mentioned
8	Passbrains Functional Usability Security Performance Localization	Self-Selection (from the community)	Tester Profile - Demographic info - Device Info - Background & skills - Domain/experience	Atlassian Jira® HPE QC/ALM	Pay-per-month
9	Rainforest Functional	Self-Selection (from the community + external communities)	Not mentioned	Atlassian Jira® Pivotal Tracke	Pay-per-hour
10	Ubertesters Functional Usability Localization	Self-Selection (from the community)	Tester Profile - Demographic Info - Device Info - Experience	Atlassian Jira® YouTrack Mantis HP Quality	Pay-per-tasks Pay-per-time
11	Digivante (BugFinders) Functional Usability Security Accessibility Localization	Self-Selection (from the community)	Tester Profile - Device Info - Experience	Atlassian Jira®	Pay-per-issue
12	Hiretesters (Crowdsourced Testing) Functional Usability Localization	Self-Selection (from the community)	Tester Profile - Demographic Info - Device Info - Experience - Testers preferences - Skills level	Atlassian Jira®	Not mentioned

Table 2.4 Comparison between existing Industrial Crowdsourced Testing Platforms, Cont...

Platforms	Testing Services	Tester Selection / Invitation	Selection Criteria	Reports/Issues Tracking System	Payment (Reward)
13	Bugcrowd Security	Self-Selection (from the community + external communities)	Tester Profile - Demographic Info - Device Info	Atlassian Jira® Trello	Pay-per-issue
14	Testlio Functional Localization Usability Location Regression Accessibility	Self-Selection (from the community)	Tester Profile - Demographic Info - Device Info - Experience - Testers preferences - Previous work performance .	Atlassian Jira®	Pay-per-hour
15	Test IO Functional Usability Acceptance Regression	Suggested List (from the community)	Tester Profile - Demographic Info - Device Info - Experience - Capabilities (type of issue reported)	Atlassian Jira®	Pay-per-issue
16	Testbirds Usability Localization Performance	Suggested List (from the community)	Tester Profile - Demographic Info - Device Info - Experience	Atlassian Jira® Redmine	Pay-per-task (extra per issue)
17	StarDust Functional Regression Accessibility	Suggested List (from the community)	Tester Profile - Demographic Info - Device Info - Experience	Not mentioned	Pay-per-issue
18	Userfeel Usability User testing	Suggested List (from the community) External Testers (global communities)	Tester Profile - Demographic Info - Device Info - Experience	Atlassian Jira®	Pay-per-task

Table 2.5 Comparison between existing Industrial Crowdsourced Testing Platforms, Cont...

Platforms	Testing Services	Tester Selection / Invitation	Selection Criteria	Reports/Issues Tracking System	Payment (Reward)
19	TestArmy Functional Security Performance	Suggested List (from the community)	Tester Profile - Demographic Info - Device Info - Experience Tester Profile	Not mentioned	Pay-per-task (extra per issue)
20	DeviQA Functional Performance	Self-Selection (from the community)	- Demographic Info - Device Info - Experience	Atlassian Jira®	Pay-per-issue
21	QA Mentor Functional Usability Localization Security Load Acceptance Compatability	Self-Selection (from the community)	Tester Profile - Demographic Info - Device Info - Experience	Atlassian Jira® Mantis Bugzilla HP Quality Bug Genie	Pay-per-month
22	UserCrowd (UsabilityHub) Usability	Self-Selection (from the community + external communities)	Tester Profile - Demographic Info - Device Info	Not used	Pay-per-task (\$5 per issue)
23	Applause (uTest) Functional Usability Security Accessibility Performance Localization	Self-Selection + Suggested List (from the community)	Tester Profile - Demographic Info - Device Info - Experience	Utest bug tracking system	Pay-per-issue
24	UserTesting Usability	Self-Selection (from the community)	Tester Profile - Demographic Info - Device Info - Communication skills	Atlassian Jira®	Pay-per-task Pay-per-time

2.7.1 Limitations of Current Crowdsourced Compatibility Practices

According to the current crowdfunding approaches published in the literature (see Section ref) and practices used by industrial crowdfunding platforms (see Table 2.2 - 2.5) some of the potential limitations are identified. The discussion of these limitations would help researchers, developers, and testers to overcome them in the future and reach the best crowdfunding practice with additional features and options. These limitations are:

- MyCrowd QA, QA Mentor, executes, Crowd4test, TestUnity, and Crowdsprint are good and useful platforms in terms of offering compatibility testing services. However it could be a drawback at the same time due to its relatively higher cost for individual developers and small and medium-sized enterprises.
- According to [Almeida et al. \(2018\)](#), CHIMP can not be suitable for testing all mobile features, such as hardware (e.g., sensors) and input (e.g., multi-touch gestures).
- Running the test over the cloud as in the MyCrowd QA platform or through the lab as in QA Mentor can not capture all real-world testing scenarios such as mobility and physical activity tracking because the devices are allocated on the server or in the lab, and difficult to move them.
- The industrial platforms and the state-of-the-art approaches do not provide enough support for detecting the issues produced from the different behaviours and interactions of target users with the app.
- These practices do not provide testers access to all different published tasks and select the suitable tasks by themselves, limiting the distribution of the tests and their execution on more devices.
- All these practices have relied on an indirect workflow that depends on the existence of the crowd manager and/or leader to organise the testing process; the delay caused by them, in turn, may lead to several delays during the testing process ([Alyahya and Alrugebh, 2017](#); [Alyahya and Alsayyari, 2020](#))
- The compatibility testing services provided by the industrial platforms might be insufficient to cover all possible unexpected issues that may appear from users interaction with the app. This is because they only deal with testers who have a high level of experience from their community, while in some cases, it is necessary to perform the test by testers who have little-to-no experience or real users, which greatly can help in finding more issues quickly.
- Most state-of-the-art approaches and industrial practices do not support API configurations compatibility testing; they focus on testing the mobile platform and device

specification only. Whereas, it is essential to develop a solution that considered all of the three dimensions.

- Although the TestUnity platform provides a service to perform the test by end-users on their devices, this service is not sufficient due to it needs to execute over the cloud. Therefore, this service can not be effective for testing all types of apps, especially those relying on sensing or mobility tracking.
- The current literature works and industrial platforms do not provide a knowledge base (wiki) to document all the important information regarding testing which includes different test cases/scenarios, issues of mobile devices or OS versions, the causes of these issues, as well as the possible solutions or assistive guidelines for building similar apps functionalities in the future by other developers.

2.8 Challenges Faced by Developers and Testers in Crowdttesting

An in-depth review has been carried out on the relevant crowdtesting research studies ([Alsayyari and Alyahya, 2018](#); [Alyahya, 2020](#); [Chen et al., 2018b](#); [Donepudi, 2020](#); [Gao et al., 2019](#); [Liu et al., 2019](#); [Nguyen et al., 2020](#); [Vishwakarma et al., 2020](#); [Wang et al., 2018](#); ?). We found that despite crowdsourced testing is experiencing emerging success in the mobile app testing field, the developers and testers, and the middleman figures (leader and manager) are still encountered numerous challenges, which practitioners and researchers need to consider in future research to improve the value of using crowdtesting for mobile app testing. [Table 2.6](#) summarises the main challenges and limitations identified from the investigated studies. The following is an elucidation of the top challenges for developers and testers.

Regarding the defining and distribution of the test to the testers, [Donepudi \(2020\)](#) indicated that one of the main challenges associated with the presence of the crowdsourcing intermediaries (leader or manager) is how they can decompose the whole mobile app into smaller tests with the appropriate definition of the requirements of each, to ensure that the test can be executed in the right direction by any testers. As for controlling the overload work of testers, [Alsayyari and Alyahya \(2018\)](#) explored that the increase of work overload on testers is one of the biggest challenges in most current crowdtesting platforms, where different developers select testers for different projects at the same time. For example, one tester may receive several invitations at once, and thus, some crowd testers will be overloaded with work while others are available and waiting for a new job. This selection method can be inefficient causing delays in performing the test, and delivering the app to the market. For controlling the testing process, [Vishwakarma et al. \(2020\)](#) explored that updating of the testing information by the leader or manager will cause a problem for testers, and might

lead to misunderstanding of the test requirements, which can affect the tester's performance and results' accuracy. As for the reliability of testers, [Alyahya \(2020\)](#) observed that most of the current report aggregation methods within crowdtesting solutions do not take into account the rating/reputation of testers when organising and/or filtering the received test reports. This makes the evaluation process more challenging for developers, especially when receiving a large number of test reports. Moreover, [Leotta et al. \(2019\)](#) stated that one of the main drawbacks the developers face in crowdtesting is the difficulty to distinguish between reliable and unreliable testers. For monitoring the testers during the testing process, [Leotta et al. \(2019\)](#) mentioned that monitoring and managing crowd testers during crowdsourced testing process are one of the top challenges faced by developers due to differences in testers' locations, time zones, languages, and cultures.

Furthermore, [Liu et al. \(2019\)](#) mentioned that reducing the size of involved testers to reduce the testing cost negatively affects the time of test completion and efficiency of the test. Therefore, they think it would be better to find a solution to balance the testing devices and testers to dynamically meet the changing needs without increasing the testing cost or reducing testing efficiency. Regarding aggregating and evaluation of test reports, [Nguyen et al. \(2020\)](#) pointed out that finding a proper way to collect large numbers of test reports together without conflict and supply useful analysis about these reports for the developer is a challenge in the mobile app crowdtesting platform. In addition, [Chen et al. \(2018b\)](#) reported that assessment of the test reports is still challenging for the developers when they receive a considerable amount of reports, due to the available resources to review and assess these reports are insufficient. Although crowd testers are reporting a large number of issues, [Gao et al. \(2019\)](#) found that many of these received issues are incorrect. From our point of view, this might be due to the lack of domain knowledge or the insufficient and unclear definition of test requirements. In current practice, the decision to end the life-cycle of the crowdtesting task is done by guesswork that the work is completed. [Wang et al. \(2018\)](#), stated that one of the main challenges face the developers during the crowdtesting process is the lack of effective methods to automatically decide when to terminate the tasks when the test fully meet requirements.

2.9 Place of this Research in the Literature

As all existing solutions discussed in Section 2.3.1 and Section 2.6 did not fully meet all of the requirements of compatibility testing presented in Section 2.2; there is still a need for devising more scalable and efficient compatibility testing approaches, which can comprehensively cover all of these requirements. This PhD thesis proposes a manual compatibility testing approach that can fill the gap of state-of-the-art approaches by considering the following requirements: (1) distributing the test on a large-scale to cover more mobile devices; (2)

Table 2.6 Summary of the identified challenges

No.	Challenges	Resources
1	Insufficient resources to assess the submitted test reports.	Chen et al. (2018)
2	Lack of method to monitor and determine the suitable time to close the crowdsourced testing tasks.	Wang et al. (2018)
3	Updates to the testing information by team members.	Vishwakarma et al. (2020)
4	Task decomposition issue.	Donepudi (2020)
5	Lack of proper method to control the overload work of testers	Alsayyari and Alyahya (2018)
6	Lack of effective reliability classification method.	Alyahya (2020); Leotta et al. (2019)
7	Inability to directly communicate with testers.	Leotta et al. (2019)
8	Lack of proper method for monitoring testers during testing process.	Leotta et al. (2019)
9	Limited the participation of crowd testers to reduce the cost.	Liu et al. (2019)
10	Lack of proper method for collecting and evaluating test reports.	Chen et al. (2018); Nguyen et al. (2020)
11	Lack of knowledge/experience about software testing.	Gao et al. (2019)

performing manual compatibility testing; (3) testing different behaviours and interaction of users with the apps; (4) providing a method that can test all different mobile app types; (5) providing a compatibility testing approach that supports all three compatibility testing dimensions of the platform, device features, and API configurations; (6) providing more insight about the internal complexity and different architecture of mobile devices. More information about the approach is presented in Chapter 5.

2.10 Chapter Summary

This chapter narrows the research gap regarding studying the reasons behind fragmentation-induced compatibility. The chapter began with a discussion of the dimensions of compatibility testing, its challenges, and requirements for building better compatibility testing for tackling fragmentation issue in general and the difficulty of performing compatibility testing. This chapter has a review of existing compatibility testing approaches and their limitations. The literature review showed that automated testing and cloud-based testing of the mobile app had been a well-researched area in recent years. However, they did not fully meet the compatibility testing requirements. The chapter also provided an in-depth investigation of the relevant research work published on crowdtesting for mobile compatibility testing. Consequently, the review showed that the research in the crowdtesting area is still in its early stages. More specifically, the results showed that crowdtesting approaches in the literature and industry did not meet the complete compatibility testing requirements.

Additionally, the chapter provided a deep comparison between two common crowdtesting workflows used by the state-of-the-art approaches and industrial practices. The comparison

showed that both of these workflows have some limitations that need to be tackled in the future. An in-depth investigation provided an analysis of all existing crowdtesting platforms from 2007 until 2021. The investigation results show that 35 platforms exist, 11 of them had fully closed, and 24 still provide services. However, by reviewing the available 24 platforms, it was found that only 5 of these platforms provided support for the compatibility testing in different ways. Nevertheless, these platforms have some limitations to service all different groups of developers. The chapter concluded by providing a clear discussion about the challenges faced by developers and testers who use the crowdtesting for their work. The review of the relevant literature and state-of-the-art compatibility testing approaches of this chapter helped to discover a new solution that can meet all requirements of compatibility testing and cover all discussed limitations. It also helped identify the appropriate methodology to develop this solution and to answer our main research questions. The next chapter [3](#) explains the methodological steps of our methodology and the selected methods of each step in details.

3

Research Design and Methodology

3.1 Introduction

In the previous chapter, an investigation of the research topic and review of the relevant literature was conducted to explore the research problem in-depth and assess current compatibility testing solutions and identify the existing challenges within these solutions. The insights obtained from Chapter 2 helped to identify the main research questions, research objectives, and select the appropriate development methodology and methods that need to be developed to address the various shortcomings of existing literature and answer the research questions. Through the selected methodology and methods, this research can go beyond the current literature and make a contribution to the mobile app testing and crowdtesting literature. This chapter articulates the selected research methodology of this study. More specifically, it presents the methods employed to answer each question of the research and justifies the reason behind the choice of each method. Additionally, it describes the conducted studies and appropriate methods used regarding data collection, sample selection, and data analysis. The chapter also outlines the ethical considerations relating to participants, shedding light on the ethical guidelines considered and how the rights of participants have been protected, with the reference to the research ethics committee who granted the approval.

3.2 Research Design and Methodology

Given the research questions listed in Chapter 1 and reviewing the relevant literature in Chapter 2, we were able to identify the appropriate methodology, including the methodological research phases, and the selected approaches and methods of each phase, which will help to tackle the main research problem and answer the research questions as validly and accurately as possible. To develop a practical and effective crowdsourced compatibility testing approach

that satisfies the needs of developers and testers that were not met by other testing approaches, emphasis was placed on engaging developers and testers in the development phases of the proposed solution. Thus, the User-Centered Design (UCD) (Abrams et al., 2004) was adapted in this research as a basic approach that the entire research methodology depends on, which will ideally contribute to the acceptance and success of the approach. Based on this, three main research methodological phases were identified. The first phase was to understand the level of acceptance of developers to use the proposed crowdsourced compatibility testing approach, identifying their needs, and key requirements for implementing the approach. The second phase was the development of the proposed approach to make it available for use, and the final phase was to evaluate the experience of the users of the proposed approach by both developers and testers, in terms of its effectiveness, benefit, and satisfaction in addressing the main research problem. Mind mapping research methods (Crowe and Sheppard, 2012) were utilised in all methodological phases executed in this research to present an overview of the content and the major aspects of each phase as illustrated in Figure 3.1. The following sections present a brief explanation of the three sequential research methodological phases; highlighting the methodological approach and selected methods for gathering and analysing the data used in each phase, to answer the research questions. Each of these methodological phases and its selected methods is articulated in detail in the upcoming chapters.

3.2.1 Phase 1: Key Requirements for the Crowdtesting Approach

This phase aims to understand the concept of compatibility testing environment and limitations of current approaches to identify the key requirements for developing a more practical crowdtesting solution. Indeed, most of the existing solutions and practices have not been adequate to enter mainstream work practices in the industry. This is probably, as we assumed from the review of literature, because they have focused on reasoning the studies' literature and challenges that precede them to gather their basic requirements and develop their solutions. This might be why developers are still suffering from fragmentation issues and challenges to perform compatibility testing and they still need to cover the actual needs of the real work environment. According to Westwater and Johnson (1995) a user-based requirements collection method provides a greater value as far as the development of a new approach is concerned. For this reason, an exploratory study with a mixed quantitative-qualitative methodological approach was performed to investigate and measure to what extent developers would be willing to accept to the use of the proposed approach and gain in-depth knowledge about their needs and the ultimate requirements that need to be considered in our approach to achieve **OBJ2**, **OBJ3** and then investigate **RQ1**.

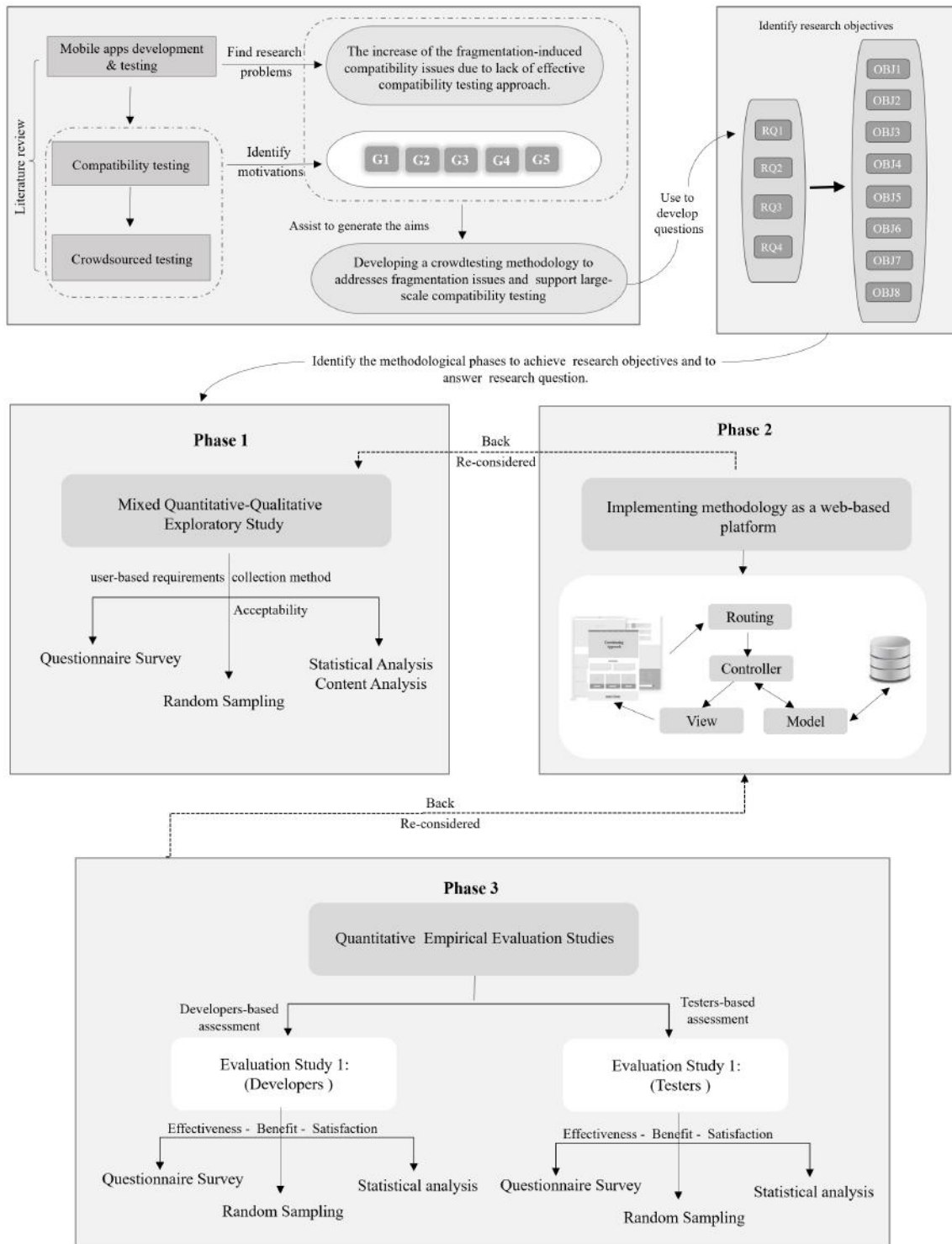


Fig. 3.1 Illustration of the overall research design and methodology

3.2.1.1 Applied Methods

A description of the methods used in the exploratory study in terms of data collection, sampling, and data analysis is provided below. More details of this exploratory survey study and these methods are presented in Chapter 4.

Data Collection Method:

Since this study focuses on the subjective opinions of developers regarding the level of acceptability to use the proposed crowdtesting approach to discover their actual work needs, which are unpublished in the literature and need to be covered in the proposed approach, a structured online questionnaire survey was used. The advantage of using the questionnaire in this study compared to other data collection methods is that the questionnaire is reasonably straightforward could help involve many developers worldwide and gather more information about the actual needs from different perspectives quickly (Dalati and Gómez, 2018; Müller and Sedley, 2015; Pratama, 2020)

Samples:

In this study, the "Random Sampling" method was used to select participant developers familiar with the Android and/or iOS platform with different levels of experience. The questionnaire's URL link was submitted to the participants and released online to the public for a period of time (for more details, see Chapter 4).

Data Analysis Method:

As this exploratory study collected two types of data, qualitative and quantitative, two data analysis methods were used. A descriptive statistical analysis method (frequencies and percentages) was used to analyse the quantitative data; and a summative content analysis method (Drisko and Maschi, 2016; Hsieh and Shannon, 2005) and coding method (Elliott, 2018; Mazumdar et al., 2017) were applied to analyse the qualitative data. After the analysis step, we divided the questions and the analysed data into two sets: the first set of questions successfully answered the first part of **RQ1** in regards to obtaining a better understanding of the subjective opinions of participant developers and their level of acceptability to use of our approach. The second group of questions has successfully addressed the second part of **RQ1** and identified the essential desires and needs of developers, which enable them to perform effective large-scale compatibility testing. The answer to both parts of question **RQ1** provided a foundation for the building of the proposed approach and identified the main aspects and requirements that must be implemented in our approach.

3.2.2 Phase 2: Development of the Crowdtesting Approach

This stage presents the design and implementation of the proposed crowdtesting approach. It discusses how the developed approach would be able to satisfy the need to provide the support identified as necessary in stage 1. The proposed approach was designed and implemented as a web-based crowdtesting platform to demonstrate the possibility of the services and features provided to meet the key requirements and addressing the main research problem discussed in Chapter 1. All processes were implemented to ensure the feasibility and effectiveness of the approach. The implementation of this approach relied on the proposed direct interaction workflow between developers and testers without the need for a crowd manager or leader as in the current state-of-the-art approaches. The web-based crowdtesting platform is developed through the "Laravel Framework" using the well-known languages PHP and JavaScript. The database was developed by MySQL because it is compatible with and supported by PHP. The reasons for selecting these languages and the database are their portability and popularity. The full description of the design and implementation of the main processes and features of the proposed crowdtesting approach and methods used is presented in detail in Chapter 5.

3.2.3 Phase 3: Evaluation of the Crowdtesting Approach

This phase aims to evaluate the success of the proposed crowdtesting approach. In our evaluation, we used empirical user-based evaluation ((end-user evaluation) approach (Dumas, 2002; Jay et al., 2008; Spink, 2002) to collect much more information regarding the experience of the use of services provided by the real practitioners who will use the approach in the future (Eraslan and Bailey, 2019; Jay et al., 2008). Also, it will help to identify whether the approach matches their work routine needs and determine whether there are further enhancements that need to be made in the approach. The proposed approach unlike other approaches in the literature, which assessed the effectiveness of their crowdtesting approach based on system-based evaluation. This means that their effectiveness evaluation was based on collecting results from the implemented approach, not through the real practitioners' perspectives as we did. This is likely be the main reason why developers and testers did not benefit much from previously existing testing approaches. In order to achieve our aim, two empirical evaluation studies were conducted to demonstrate the overall satisfaction, benefit, and overall effectiveness, taking into consideration the evaluation of the effectiveness of all features and sub-processes implemented within the approach from different perspectives of developers and testers, to complete **OBJ5**, **OBJ6**, **OBJ7** and investigate the three main research questions (**RQ2** **RQ3**, **RQ4**). These two studies have employed quantitative approach because we aim to evaluate the success of the approach and generalise results from the sample of a target population.

3.2.3.1 Applied Methods

An overview and summary of the methods used for data collection, sampling, and analysis in the two evaluation studies is described below, and more details are presented in Chapter 6.

Data Collection Method:

A structured data collection process was applied in this study. According to Vuolle et al. (2008) a questionnaire is the best possible method to evaluate the success of the innovations, systems, and approaches when still in the early stages of use. Using the questionnaire will make it easy to reach developers and crowd testers worldwide and obtain their point of view about the use and success of the proposed crowdtesting approach regardless of their physical location (Dalati and Gómez, 2018; Müller and Sedley, 2015). Also, for the purposes of this research, the use of the questionnaire will help to collect more honest answers about the evaluation of the approach from professionals and employees in companies because their identity remains anonymous when they answer, which provides them with more confidence and thus helps to provide more accurate feedback (Pratama, 2020). Therefore, an anonymous form of questionnaire was used to evaluate our approach in this research. To the best of our knowledge, there are no questionnaires yet available which could be used in the evaluation of all the important aspects of the entire crowdsourcing approach or features that we intend to measure when evaluating our approach. A review of the relevant literature on taxonomies and the main characteristics of the crowdsourcing and crowdtesting process was conducted. Based on this review, a set of all fundamental aspects that must be measured to assess our crowdtesting approach was identified (more details are presented in Chapter 6). Based on these aspects, two "Crowdsourced Testing Experience Questionnaires" were developed as tools to collect the data from the developers (CSTE-Q/D) and the testers (CSTE-Q/T) in our two empirical evaluations studies. The CSTE-Q can be adapted in any crowdsourced-based contexts in the future. An overview of the whole construction process of the CSTE-Q/D and CSTE-Q/T questionnaires and their validity checking are presented in Chapter 6.

Samples:

Participation invitations were widely distributed through different regions, and participants were selected based on the "Random Sampling" method according to the following criteria:

- English language speakers;
- Developers and testers with different years of experience;
- Developers and testers who are experienced with Android, iOS, or both;
- Developers who are experienced with the development of various types of mobile app;

- Testers who are experienced with the testing of various types of mobile app;
- Developers and testers who have knowledge about the crowdtesting process.

After selecting the participants, the implemented platform was launched and made available to participants for a limited period of time, and the participants were asked to perform a set of tasks (see Chapter 5). This was followed by, a feedback collection step using the two implemented questionnaires CSTE-Q/D and CSTE-Q/T. More details are presented in Chapter 6, and the size and characteristics of the developers and testers sample are outlined in Chapter 7 and 8.

Data Analysis Method:

The data collected from the two evaluation studies was analysed in a quantitative way. A statistical analysis, in particular, the "Mean Analysis" was used to assess the effectiveness of all sub-processes of the approach and hence answer the ten sub-questions (A - J) of **RQ2** (see Chapter 9). Each of the sub-questions provides evidence on the effectiveness of each sub-process within the approach. Through answering these questions, comparisons were drawn with state-of-the-art crowdtesting approaches in the literature relating to the academic and industry sectors from 2017 - 2021. It was found that most of the existing crowdtesting approaches were developed and used by companies, and none of them have published any details about their approach effectiveness or how they have been evaluated; only their limitations have been published by the researchers in the literature. Furthermore, only limited crowd-based solutions that have assessed the effectiveness of sub-processes within the crowdtesting approaches have been published in the literature. Therefore, the only means by which we could make comparisons between our work and these approaches was by comparing the effectiveness of the sub-processes of our approach with those published in the literature and the defects discovered and published by researchers in the literature. This enabled us to determine if our method improves on current compatibility testing approaches and covers their gaps and the main gaps mentioned in Chapter 1. The answers to all these sub-questions and the comparisons with the state-of-the-art literature, together formed the foundation for addressing the broader question **RQ2** regarding how effective the proposed approach is in addressing the problem of performing large-scale compatibility testing from different developers' and testers' perspectives.

The "Mean Analysis" also was used to assess and describe the benefits from and overall satisfaction with our approach. For further analysis, the inferential statistic tests "Differences Tests" were used to compare and describe the differences in the received benefits and satisfaction of the approach between the different groups of developers and testers. Additionally, we compared the benefits the developers and testers received from our approach with the list of

the recent challenges they faced (see Chapter 2) and their requirements gathered from the exploratory study (as presented in Chapter 4) to successfully answer the sub-question of **RQ3** in respect of identifying if our approach has addressed the unmet needs of all different groups of developers and testers. The answer to this sub-question was used as input to address **RQ3** and then answer **RQ4**.

3.3 Research Validation

Poorly designed research may produce invalid results and therefore, it is essential to validate the entire research design and its results. According to Yin (1994), the main criteria to judge the validity and quality of the research design and obtained results are: Construct Validity, Internal Validity and External Validity. The following sections explain how these criteria were applied to prove the validity and quality of our research results.

3.3.1 Construct Validity

Construct validity refers to whether the questionnaires used actually measure the aspects we intend to measure in terms of acceptability, effectiveness, benefit, and satisfaction of our approach (Leedy et al., 2014; Lucko and Rojas, 2010), in order to produce valid results. The following processes were followed to ensure the high construct validity of our data collection tools and our findings:

- **Check face and content validity:** We have inspected which questions on our questionnaires to ensure that they cover the construct that is being studied. The results showed that it has adequately measured the key concepts in the study (see Chapter 4 and Chapter 6). This is clear evidence of the validity of the questionnaires and collected results.
- **Pilot test/Pre-test:** As reported by Lucko and Rojas (2010), one way to accomplish the "construct validity" and ensure the results that will be obtained from the research are valid, is through a pilot test. For this reason, we performed pilot studies to fine-tune the questionnaires used in this research before their use in the actual data collection. The high construct validity value indicated the high validity of the results (see Chapter 6).
- **Reliability of the questionnaires:** We assessed the reliability of our questionnaires to ensure that the repetition of the experiments will produce consistent results in the future. The assessment showed that our questions had achieved high reliability, which confirms the validity of our findings in this research (see Chapter 4 and 6).
- **Questions validity assessment:** According to Ford and Scandura (2018); Messick (1990) poorly written survey questions can threaten the validity of the research findings.

For this reason, we evaluated the questionnaires to ensure the simplicity, clarity, integrity, and understandability of the questions, ensuring that the questions were well written and the language was not complicated or above language level of the reader. The latter point was particularly important because the respondents who participating were from different countries with different levels of English skills. This enabled our participants to understand the questions clearly and so provide more accurate results. This also proves the validity of our research findings.

3.3.2 Internal Validity

Internal validity relies largely on the accuracy of the study procedures and how rigorously they are performed (Arlin, 2020), particularly the extent to which every aspect of the experiment setting was controlled to ensure that the outcomes were valid and were not influenced by other factors or variables (Siegmund et al., 2015). In this research, we considered the following criteria to ensure internal validity:

- ***Exclude all influencing factors:*** We excluded all factors that can affect the participant answers during the execution of the studies so that we can prove whether the new approach indeed facilitates compatibility testing and improves the everyday work of different groups of developers and testers. This will ensure the high internal validity. Such excluded factors are:
 - ***Exclude time/ history effect:*** We did not intentionally or unintentionally influence the participants or place stress on them to perform the experiment at a specific time. We provided them with the freedom to complete the experiment and fill in the questionnaires at any time during the four months of the data collection period. This ensured that there would be no factors or pressure affecting their performance or answers and thus help to gather more accurate and valid findings.
 - ***Exclude testing effect:*** In this research, we excluded those who participated in the questionnaire assessment, pre-test, and pilot study from participating in the main evaluation studies. This decision was made as these participants have become more familiar with the questions formats and the use of the approach, which would in turn result in their feeling less stressed, and therefore they would be able to complete the experiment easily in a shorter time compared to other new participants, affecting the outcomes.
 - ***Avoid experimental manipulation:*** We avoided making any changes during the experiment or manipulating items with any independent variable in the study. This also ensured the high validity of the results collected.
- ***Unified study protocol:*** We followed the same procedures for all studies; we did not do things differently with one group of participants as oppose to other groups. Indeed,

we explained the purpose of the study and each process in detail to all participants from the beginning to make them familiar with and aware of each step in the experiment, which would help participants to perform better and produce high results' validity.

- **Randomization:** We randomly assigned participants to developer and tester groups and were not biased towards accepting participants who had specific characteristics or to select a specific number of participants for each group. This proves that this research does not have any systematic bias between groups of the selected sample, thus ensuring the validity of the research findings.

The successful fulfillment of these criteria is clear proof that this research and its findings have high internal validity (Arlin, 2020; McDermott, 2011; Pritha, 2021; Siegmund et al., 2015)

3.3.3 External Validity

External validity refers to whether the research findings resulting from the selected sample can be applied and generalised to the larger population and in a broader context (Pritha, 2021; Siegmund et al., 2015) In this research, we considered the following criteria to ensure external validity:

- **Probability sampling (Random selection):** In this research we used the "Random Selection" method for recruiting the participants of our studies, both developers and testers, which is a good indicator that the research achieved the external validity as confirmed by Acharya et al. (2013); McDermott (2011).
- **Sample characteristics:** Our samples have involved characteristics of all different groups of developers and testers around the world, such as different experience in mobile platforms (Android/iOS), levels of experience, backgrounds, countries, and work status (e.g., freelancers and employees in large or small organisations). This confirms that our sample is representative of the whole population and that it has a high population validity, as reported in (Pritha, 2021; Yin, 1994).
- **Use inclusion and exclusion criteria:** We inspected the characteristics of the participants with the identified selection criteria before they started the experiment. Those who did not satisfy the criteria were excluded from the evaluation process to ensure that the selected sample represented the population that we are studying in our research. Thus, this is clear evidence that this research and its findings achieved high external validity, as reported in (Arlin, 2020).

To summarise, the achievement of the above criteria is strong evidence that this research and its findings have high external validity, and consequently, our findings can be generalised to a larger group of developers and testers (a larger population).

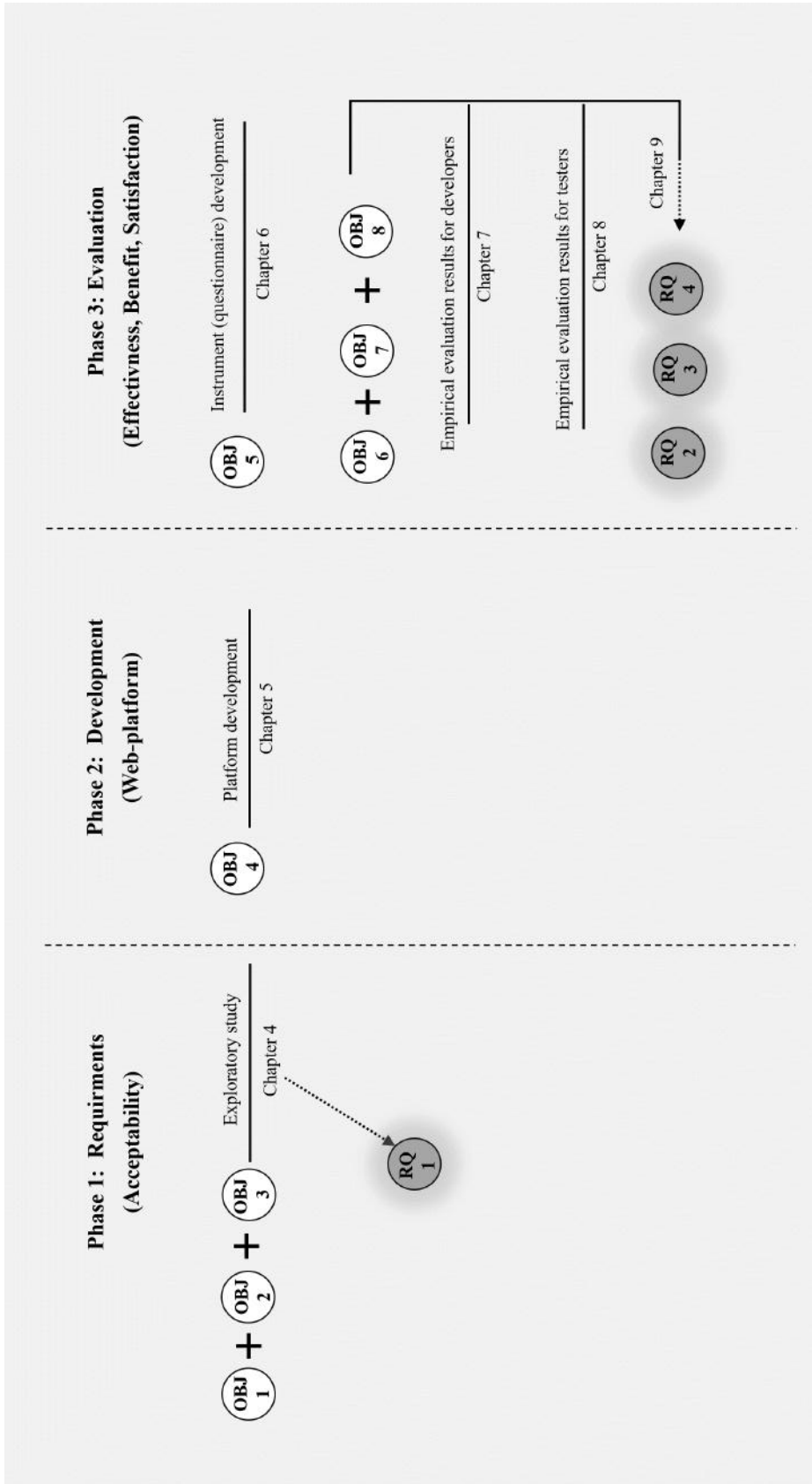


Fig. 3.2 Mapping of research objectives and research questions across the research methodological phase

3.4 Ethical considerations

As with much relevant HCI research, required ethical issues have been taken into account in this research to conduct the exploratory study and the two empirical evaluation studies. The ethical guidelines pointed out by [Myers and Newman \(2007\)](#) and [Lazar et al. \(2017\)](#) were adapted. These are as follows:

- **Ethical approval (permissions):** This research has undergone full ethical review by the Research Ethics Committee in the Department of Computer Sciences at the University of Sheffield. Ethical approval of this research was obtained (on 26-02-2019) prior to the commencement of the three main studies (see Appendix A, Part I).
- **Informed consent:** In this research, informed consent was obtained from the participants before conducting our studies. For the exploratory study, no written consent was required, the participants' responses to the online questionnaire were presumed as consent to their participation ([Karbwang et al., 2018](#)). While, for the two empirical evaluation studies, an online consent form was sent to participants so that they would provide their consent to participate (see Appendix C, Part I). The permission was obtained from the participants, who were given a clear description of the study and explanation of all information provided in the information sheet. A copy of the online consent form was given to each participant as a reference while retaining the online signed copies as documentation of the participants' consent.
- **Confidentiality and anonymity:** According to [Walsham \(2006\)](#), confidentiality and the participants' anonymity are considered critical factors in HCI research. The three main studies were considered 'minimal-risk' research since the questionnaires were strictly anonymous, no sensitive questions were asked, and no demographic or educational information regarding any participants was collected. Therefore, it was impossible to identify the details of participants or to link any answers to a specific person.

3.5 Chapter Summary

This chapter has briefly discussed the research design and its associated methodology. It was concluded that three linked phases would be completed to create an effective and practical solution. Through these three phases, three main studies would be carried out to prove how well the proposed approach tackles the main research problem. The first phase focused on gaining a holistic understanding of the developers' needs to extract the key requirements to build the proposed approach. This was achieved by a "mixed qualitative-quantitative exploratory study". The outcome of this study provides a basic understanding of the execution

of the crowdtesting design and development. The second phase focused on implementing the approach based on identified requirements as a "Web-based crowdtesting platform". Then, the implemented approach was assessed in terms of its effectiveness, benefit, and satisfaction to achieve phase three. This was completed through performing "two quantitative empirical evaluation studies", one for developers and the other for testers. The following chapters will discuss in detail each of these three phases, separately and in order.

Part I

Research Phase 1: Key Requirements for Crowdfunding Approach

4

Critical Requirements of Proposed Crowdtesting Approach

4.1 Introduction

Chapter 2 discussed the recent researches and contributions to reduce fragmentation-induced compatibility issues and obstacles behind performing effective mobile device compatibility testing. It also investigated the relevant literature and industrial practices on crowdsourcing to address these two challenges and identify the main requirements for building a practical crowdsourced compatibility testing approach. Many questions still need investigating on the actual needs of developers and testers. Also, more evidence is required to show whether the requirements reported in the literature are effectively applied to the real world. Therefore, this chapter aims to fill this knowledge gap in the literature and collect our core requirements from real practitioners. This chapter provides an exploratory survey study to understand the developers' perceptions and their needs to work with the public and unknown testers. The chapter describes the design and methods used to achieve this exploratory study. The chapter starts by giving an overview of the design and development of the data collection tool used. Then, it describes the way of distributing the survey questionnaire to the population and the sampling method. This is followed by a detailed explanation of the analysis methods applied to the gathered data. Next, the chapter presents the results and main findings from the exploratory study conducted. Then, an interpretation and discussion of the collected results will be given to clarify the fundamental requirements, followed by a list of all the functional and non-functional requirements that will be used for developing the proposed approach.

4.2 Data Collection Method

Both qualitative and quantitative data were gathered for this study over a temporal period from Jan 2018 to March 2018. Data collection was achieved through a structured online questionnaire survey to get a broad set of requirements from app developers scattered on various geographical locations. As spreadsheets are more suitable for summarising and analysing questionnaire responses to gain more insight (Mazumdar et al., 2017), the participants' responses were gathered and then stored in a Google Spreadsheets in a structured manner for later analysis.

4.2.1 Questionnaire Survey Design

This questionnaire was created by Google Form and includes a mixture of *close-ended* and *open-ended* questions (Denscombe, 2014). Most of the survey questions were designed as *closed-ended*, which contains a mixture of multiple-choice, YES/NO, and rating questions. However, there are also a few *open-ended* questions for capturing and summarizing the participant developers' opinions and provide an opportunity for them to answer the survey questions in their own words. Therefore, this may encourage them to express broader issues than those mentioned in the *close-ended* questions. From the study of relating literature and with the cooperation with the academic supervisor, it was discovered that there is limited knowledge regarding the trustworthiness, crowd motivation, and job evaluation in the crowdsourcing approach. These three areas were considered critical dimensions that must be considered for developing the proposed crowdtesting approach. Questions were constructed from the literature retrieved from searching these dimensions and based on identified requirements listed in Chapter 4. Thus, a structured questionnaire with fifteen unique questions was used (see Appendix B, Part I). The questionnaire was split into seven sections. Figure 4.1 shows the whole structure of the questionnaire, including the seven sections and the questions' dimensions within each section. The following detailed description of these sections.

1) Developers' experiences with crowdsourcing

This section started with a few questions related to developer's experiences with the use of crowdsourcing for mobile apps testing. The first question asked participants about their typical approach when testing their mobile apps. The options given included crowdtesting platform, testing companies, or automated/cloud testing tools. Participants who had previous experience in the use of crowdsourced testing were required to list the crowdsourcing platforms or companies they had used. Developers were then asked about the crowdsourced programming websites they mainly use to search for solutions to the programming issues they face. Three crowdsourced programming websites were listed as choices for this question

(Stack Overflow ¹, GitHub ², and Stack Exchange ³).

2) General expectations from the public crowdtesting process

In this section, the survey concluded with questions about developers' opinions on whether they will be able to obtain a critical mass of testers when they used a public crowdtesting and "why?". Finally, they were asked about the results they expect to obtain from using public crowdsourcing system.

3) Key requirements for the public crowdsourced testing

Questions in this section queried developers' broad views in respect of which testing requirements should be taken into consideration to aid public crowdtesting of mobile apps. The first question asked developers for the typical starting searching points that they use to search for a solution to any issues they may face during the app development process; four suggestions had been provided: Platform (e.g., iOS, Android, etc.), OS Version, Brand, and Model Number. Following that, they were asked if they would prefer to use a structured form that has multiple sections, or just title and general description similar to StackOverflow. Next, participants were asked about their point of views as to whether they think that direct interaction with testers is vital during the testing life-cycle, and what are their preferred ways to communicate with testers. Finally, the authors asked the participants if they think the ease or difficulty of the design of crowdtesting system interfaces affects the enthusiasm of the crowd.

4) Developers' confidence in public crowd testers.

The questions in this section sought information on the confidence of developers on unknown testers with varying levels of experience and information provided by them. The developers were asked whether they trusted public and unknown testers to perform their testing tasks; if they responded with "No" then they needed to justify why they did not trust them. Only developers who responded with "Yes" were asked another question in which how much they trusted information provided by public testers.

5) Preferred features and the motivations

This section focused on capturing information about the features that developers would prefer to be included in the design of a public crowdsourcing solution. The first question required the developers to identify the elements that will attract them to work with public testers to execute their tests, rather than dealing with crowd communities belong to specific testing organizations. They were then asked about the incentives that they would be willing to offer to public testers to encourage them to participate in executing testing tasks.

¹<https://stackoverflow.com/>

²<https://github.com/github>

³<https://stackexchange.com/>

6) Evaluation of the performance and quality of work

Participants were asked two question, in the case of dealing with unknown testers, to what extent developers know that testers have performed the test. Then, they were asked how they want testers to prove the validity and accuracy of results they have obtained.

7) Required information for effective public crowdtesting process

Questions concerned information that developers must clarify in defining the task to the testers, as well as the detailed information they need to collect from testers to recognize the reasons for the issues that would be reported.

4.2.2 Questionnaire Quality Evaluation

Researchers usually need to evaluate the quality of a data collection instrument to eliminate the issues of reliability and validity, which could impact the findings of the study. Several methods can be used to evaluate the reliability and validity, some of them are achieved by statistical techniques, and the others by the involvement of human factor (Heale and Twycross, 2015). In this research, as the survey has included *open-ended* questions, and since the evaluation methods that involved the human factor is the most appropriate in case of using qualitative questions (e.g., open-ended questions) as mentioned by McDonald et al. (2019). We decided to assess the quality of the survey questionnaire by considering the human factor. Thus, "Inter-rater Reliability" and "Face and Content Validity" was used for assessing the reliability and validity of the questionnaire. The results of these two tests provided evidence about the quality of the questionnaire and the ability to use it in this exploratory study to collect developers' requirements. Two PhD students in the department of computer science at the University of Sheffield were recruited to perform these two evaluations.

Inter-Rater Reliability (IRR):

The "Inter-rater Reliability (IRR)" test was used to measure the degree of agreement among raters on the consistency of the questionnaire in creating reproducible results (Venkitachalam, 2014). The focus was on measuring two main criteria: simplicity (understandability) and correctness of questions, in order to (1) Reduce bias and ambiguities of questions; (2) Provide a better quality of results (consistent results); and (3) Provide credible information. The two experts completed the process of evaluation in three main steps:

Step 1: The questionnaire was shared with the raters and asked them independently to give a ranking from 1 to 10 to each question for measuring simplicity and then correctness; The critical appraisal sheet that raters used to evaluate the reliability of the questionnaire is presented in Appendix B, Part II.

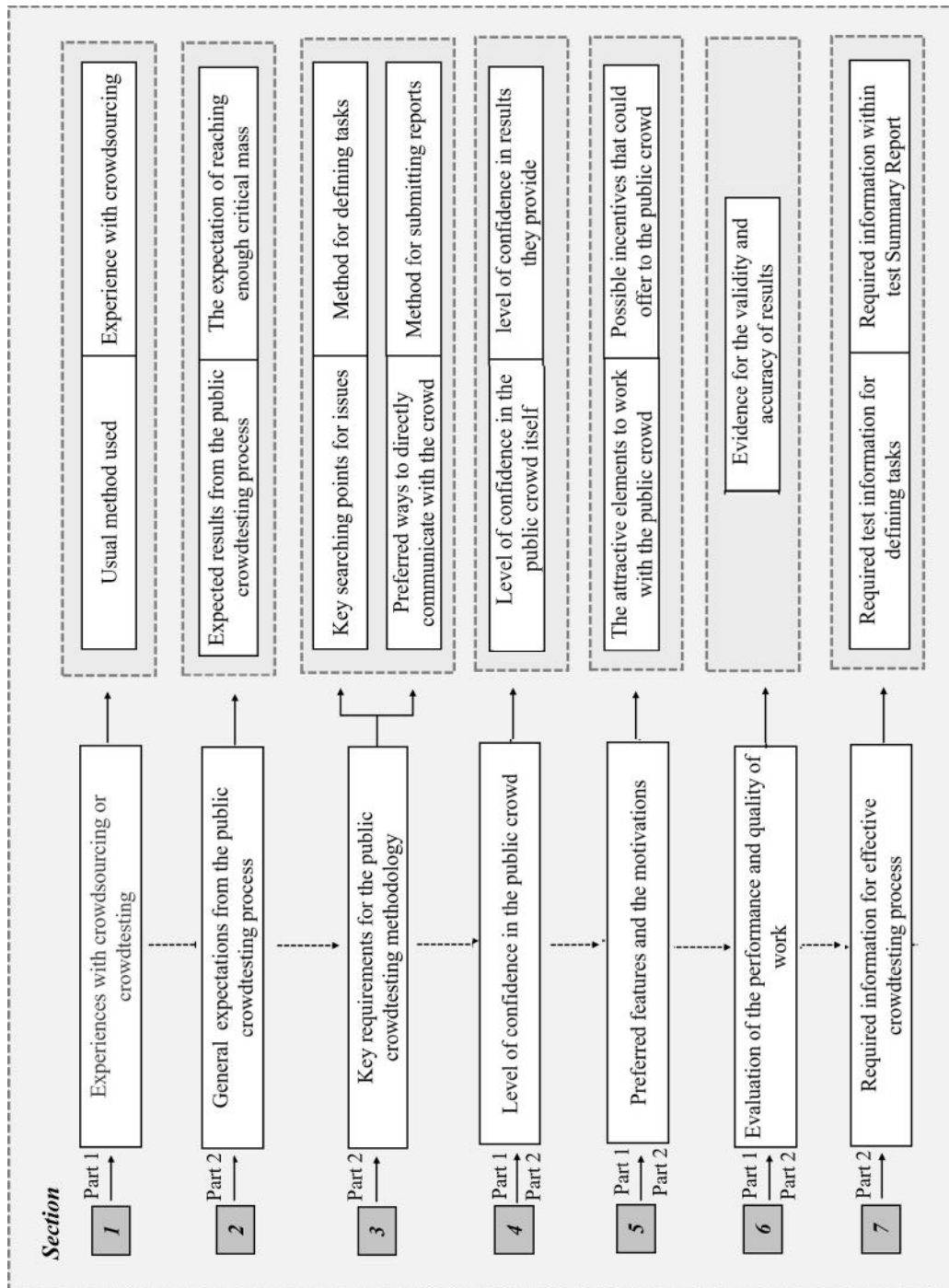


Fig. 4.1 Structural design of the questionnaire

Step 2: The results of the two raters were compared, and an agreement score was given, if the ranking of all raters is not matched for a specific question (score =0) and if matched (score=1), see Table 4.1;

Step 3: The level of agreement between the two raters for both simplicity and correctness was determined. The level of agreement is the measure of inter-rater reliability. This is done by dividing " the number of agreements (1) for all raters" on both " the number of agreements (1) plus the number of disagreements (0)". The results of this calculation show that IRR value for simplicity is 0.80 and for correctness is 0.86. These values indicate that the questionnaire is very reliable, as the minimum acceptable value for reliability is 0.70 as mentioned by [Guide \(2017\)](#); [Venkitachalam \(2014\)](#). Table 4.1 describes the results of inter-rater reliability/agreement test of the simplicity and correctness of the questionnaire between two raters.

Table 4.1 The overall outcomes of "Inter-rater Reliability" test

	<i>Simplicity</i>			<i>Correctness</i>		
	Rater 1	Rater 2	Agreement Score	Rater 1	Rater 2	Agreement Score
Q1	10	10	1	10	10	1
Q2	9	9	1	10	10	1
Q3	9	8	0	9	7	0
Q4	9	9	1	10	10	1
Q5	8	7	0	8	8	1
Q6	8	8	1	9	9	1
Q7	10	10	1	10	10	1
Q8	8	8	1	10	10	1
Q9	9	9	1	9	9	1
Q10	10	10	1	10	10	1
Q11	7	9	0	9	9	1
Q12	8	8	1	9	9	1
Q13	8	8	1	10	9	0
Q14	9	9	1	9	9	1
Q15	9	9	1	9	9	1
			Mach = 12			Mach = 13
			Total = 15			Total = 15
			IRR = 0.80%			IRR = 0.86%

Face and Content Validity:

The "Face and Content Validity" test is considered the most commonly used test by the researchers to ensure that the questionnaire truly measures what supposed to measure ([Heale and Twycross, 2015](#); [Venkitachalam, 2014](#)). Face and content validity test was used in this

study to evaluate the validity of the questionnaire by focusing on the following three criteria: (1) Relevance: How important each question is (is it relevant to crowdtesting, compatibility testing of mobile devices and fragmentation issues); (2) Feasibility: How feasible each question is (is it in the testable format); (3) Essentiality: How necessary each question is (is it helps in achieving the aim of the study). The same experts performed this evaluation. The suggestion resulted from this evaluation were: moving this question "what are the benefits you will obtain when dealing with public testers?" to be a subquestion to Q#9. Another suggestion is Changing the sub-question structure and the Q#7 to be 5 Likert-scale points rather than 4 points to provide more precise information. The other amendment was adding this part "the four essential parts of the information" to both Q#14 and 15 to allow participants to provide more options.

4.3 Population and Sampling Method

Since the target population for this study is mobile app developers, the population was sampled by applying a simple random sampling (Bryman and Burgess, 2002). The sampling was carried out through randomly distributing the questionnaire to a large pool of mobile apps development communities on Facebook, LinkedIn groups related to mobile development, and shared through our Twitter account. The random nature of sampling participants assists in the generalization of gathered information across different groups of developers and testers in the population without significant discrepancies (Bryman and Burgess, 2002; Mathers et al., 1998). This questionnaire was released online to the public for two months, starting from Dec 25, 2017. The questionnaire survey was fully completed by 50 app developers from different countries such as Saudi Arabia (SA), Egypt, United Kingdom, Canada, Germany, Singapore, Sweden, Romania, United Arab Emirates (UAE), and United States (US) who have diverse experiences in Android and/or iOS.

4.4 Procedures

The time needed to complete the survey was approximately 5 - 8 min. It was interesting to see how some participants gave encouraging comments; for example, "I hope this idea can facilitate the testing process of apps on a variety of mobile devices". Another sent a positive feedback on survey's excellent preparation in respect of its context, duration, and the variety of the survey's questions, as well as the whole concept of the survey "The survey is very good with a reasonable time and covers very interesting points". A group of participants had limited knowledge of crowdsourcing methods. They asked for a session specifically to provide brief information on the idea of crowdtesting before filling the survey because of their interest in the subject. Consequently, some sessions were set for some of them by textual explanation on LinkedIn, Facebook, or call over Facebook Messenger as well as online telephone calls

through WhatsApp, Viber, and Tango. Each explanation took approximately 10-15 minutes to explain the idea of crowdtesting and giving examples. In the end, some time was given to allow developers willing to participate in this study to complete the questionnaire.

4.5 Data Analysis Method

Due to a qualitative and quantitative data were collected by the questionnaire in this research, two data analysis methods were used in order to analyze the content of these two types of data, descriptive quantitative approach (using descriptive frequencies) and the descriptive qualitative approach (summative content analysis (Hsieh and Shannon, 2005)). The following is a detailed elaboration of the analysis steps accomplished in each method:

1) *Quantitative Data Analysis*

A descriptive quantitative analysis was conducted to analyze the categorical responses of *closed-ended* questions (from Q1-Q3, Q5-Q9) in this exploratory study. Usually, the quantitative data collected from online survey tools are ready to analyze (Lazar et al., 2017). In this quantitative analysis, the collected data from *closed-ended* questions were extracted from google form into the Google spreadsheet in an analyzable format and then analyzed using descriptive statistics for each question. The categorical responses were grouped based on the categories, then statistical analysis utilized to calculate the distribution count (frequency and percentage) for each combination of categories. A frequency of a response to each category was calculated using the COUNTIF function (which tells us the number of times each specific category is selected). Moreover, the percentage of response was calculated by dividing the frequency of each response category by the total number of responses for each combination of categories. In this analysis, descriptive statistical of the results (frequency and percentage) was adequate to summarising the gathered data and understanding developers' needs to provide a clear picture of the requirements needed for building the proposed crowdtesting approach. There was no need for further inferential statistics since there is no need to identify statistically significant differences between groups of data, understand the relationships among them, or studying how they impact each other (Fowler Jr, 2013; Lazar et al., 2017).

Result validation: The external validity of the results of this analysis is verified as it involves the participation of different subjects with a different experience of Android and iOS mobile, from different geographical regions. This random and different participation of developers is strong evidence for external validity (Acharya et al., 2013), which help in generalizing obtained results to the larger population (Bryman and Burgess, 2002; Mathers et al., 1998).

2) *Qualitative Data Analysis*

Content analysis is considered the standard and most appropriate method of analyzing responses to *open-ended* questions (Züll, 2016). A descriptive qualitative analysis, especially summative content analysis (Hsieh and Shannon, 2005) and code development methods (Boyatzis, 1998; Krippendorff, 2018), was conducted on the *open-ended* questions (Q4, Q9-Q15) to understand the subjective opinions of participant developers on the use of the public crowdtesting approach. In qualitative analysis, content analysis can be used with either an inductive coding (without a pre-defined code frame) or deductive coding (with a pre-defined code frame) and analysis approach (Krippendorff, 2018; Sgier, 2012). In this study, an inductive content analysis was used through a data-driven approach (Judger, 2016; Krippendorff, 2018; Popping, 2015), where categories are arising directly from the survey responses based on phrases or keywords in the response. This implementation of a summative approach to inductive qualitative content analysis involved a five-step process:

1. Extracting all the responses of the survey into one document;
2. Identifying the meaningful pieces of information within all responses of each question one-by-one;
3. Color coding schema is applied to facilitate the analysis process by shading the extracted pieces of information that have the same semantic meaning with the same color, see Figure 4.2 (a). This step is repeated twice to reconsider previously shaded/coded information.
4. Separately developing categories based on shaded colors (Each color is referring to a particular category). This step was re-conducted twice to ensure that the developed categories are correct. All pieces of information that have the same meaning and same color belonged to the same developed category.
5. Counting the frequency of each category using "hand-written tally marks" (see Figure 4.2 (b)) and then calculating the percentage as appropriate.

The responses that were presented by only one participant was quoted as they are without change.

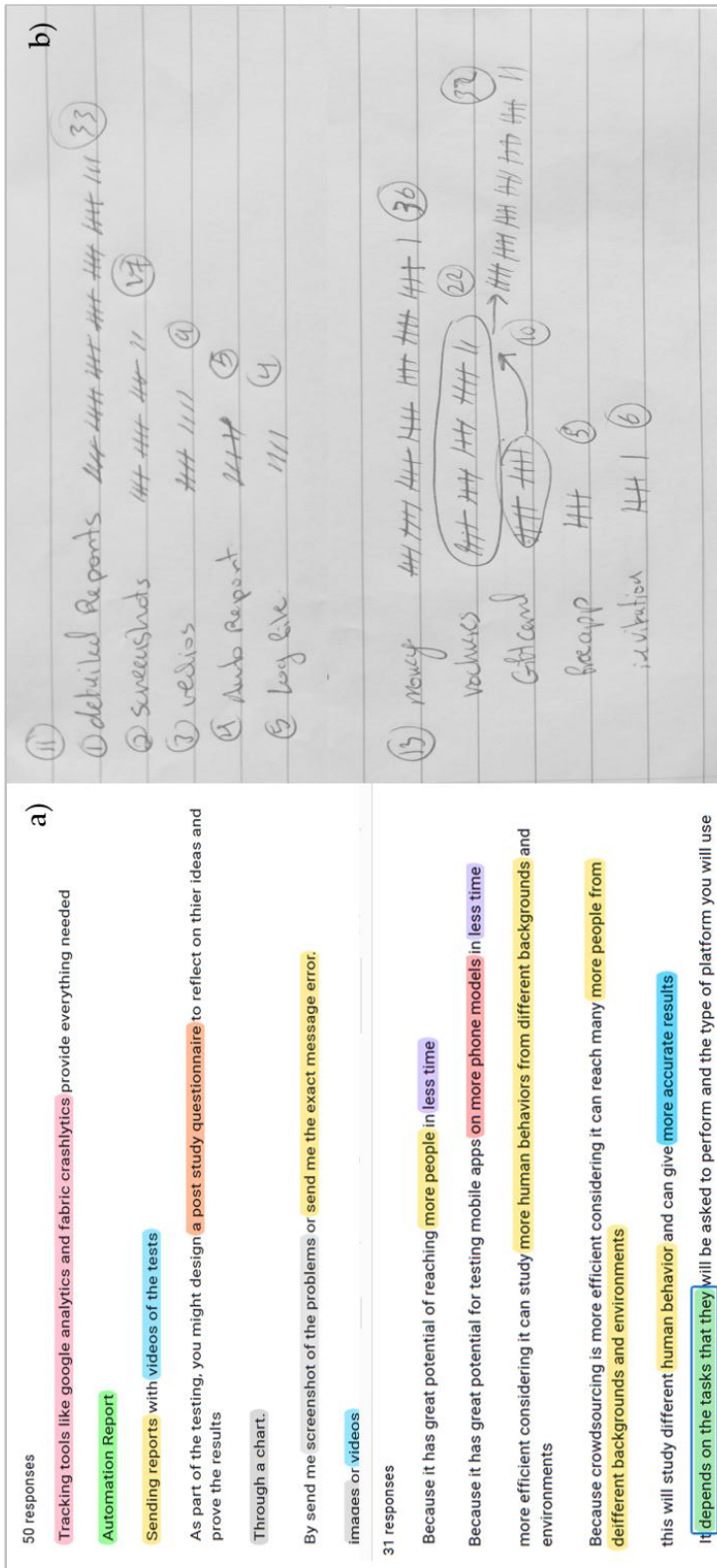


Fig. 4.2 Analysis of participant developers responses: (a) identification and encoding participants' feedback based on the color-coding of the relevant category; and b) hand-drawn tally marks helped to count and summarize the number of category repetitions based on survey responses.

Result validation: As mentioned before, the external validity of the results of this study is verified by the a large number (50) of random self-selected. Furthermore, in content analysis, as the last step, it is essential to scrutinize the credibility and trustworthiness of results, in particular, how well codes (categories) cover data (Elo et al., 2014; Woike, 2007). This examination process gives the reader a clear indication of the overall validity of the gathered results of the study (Hsieh and Shannon, 2005). Semantic validity is a method to estimate the credibility and trustworthiness of the codings and extracted categories (Krippendorff, 2018; Popping, 2015). It was used in this study to ensure the credibility and trustworthiness of the extracted categories from the participant responses to *open-ended* questions. An additional co-researcher from the same domain was invited to review the results. All responses were provided anonymously to co-researcher without accessing to the categories that we extracted, no relevant data were inadvertently excluded or irrelevant data included. The co-researcher was asked to identify some potential categories after reading some of the meaningful information extracts. We then evaluated whether the main results are matching the results extracted by co-researcher to check the credibility and trustworthiness of the results. The results showed that co-researcher extracted almost similar categories to that we have extracted in all open-ended questions (Q4, Q9-Q15). We calculated the number of agreed questions (7) on the overall number of questions (8) to obtain an agreement percentage. The percentage was almost 87% for the generated categories of all questions. This percentage indicates the credibility and trustworthiness of the results and extracted categories (Elo et al., 2014). Table 4.2 presents some of the meaningful information extracted from participant responses to *open-ended* questions and categories generated by us and co-researcher.

4.6 Study Results

The primary objective of this survey is to elicit the developers' opinions and measure their willingness to work with a member of the public and unknown testers for compatibility testing of their apps with mobile devices. This section summarizes the main findings of the conducted survey to illustrate the scale on which mobile apps developers agree to work with public and unknown testers directly without the need for a manager or leader, as in most of the crowdtesting methods used by testing organizations. Also, it provides the critical requirements needed to be included to achieve an effective process in terms of crowdtesting completion, crowd motivations, job evaluation, and to ensure the reliability of the crowd. The responses of participants were highly insightful, highlighting several guidelines that must be considered in the proposed crowdtesting solution to enhance the trust of crowd testers. The next section presents a description of the survey findings that were collected from the questionnaire survey.

Table 4.2 Examples of some generated categories from participant responses by the main researcher and co-researcher

Extracted meaningful information	Generated category by main researcher	Generated category by co-researcher
An idea could easily be stolen and published before finishing the app development process.	lack of identity and guarantees	lack of trust
There are countries which have a bad reputation in the software industry.	Level of technological development in different countries	Different experiences in software development
IT education in some countries is really not reliable.	level of education in different countries	Unreliable education in IT area
If it enables testers to work anywhere and at any time.	Flexible working	Flexible work environment
If there is more knowledge or experience provided by other developers.	Knowledge sharing	Knowledge sharing and collaboration
The different participant performance helpful to achieve a good quality product.	Diversity of users and testing experience	Variety of working experience
Use of others tools like google analytics and fabric crashlytics could provide evidence about the accuracy of results.	Integration of tracking tools	Use of automated tools

1) Developers' experiences with crowdsourcing

Question (Q.1.) in this survey was aimed at gaining more knowledge about the experiences of developers in the use of crowdsourcing platforms. Table 4.3 shows how frequently crowdtesting platforms and other testing methods such as private testing companies and automated/cloud testing tools were used by respondents. As observed, over half 58% of the developers never used crowdtesting platforms and 14% rarely used them. Meanwhile, 26% of the participants said they sometimes used platforms, and a small minority representing 2% of the developers said they often used them. Unfortunately, no one responded that they always used platforms. The following are some of the most frequently used crowdtesting platforms: uTest⁴, MyCrowd QA⁵, 99tests⁶, Mob4Hire⁷, BugFinders⁸, and TestIO⁹. The data in Table 4.3 also showed how often developers used automated tools or cloud testing services. It is clear that a

⁴<https://www.utest.com/>

⁵<https://https://mycrowd.com/>

⁶<https://99tests.com/>

⁷<http://www.mob4hire.com>

⁸<https://www.bugfinders.com/>

⁹<https://test.io/>

good proportion 34% of developers sometimes used them while only 6% always used them. Meanwhile, 24% responded to say they never used them while 22% rarely used. A small minority 14% of participants responded that they often use them. The data in the table also clarified that 34% of developers always used testing companies. A similar proportion of developers answered that they used them sometimes. Only 6% of developers replied they rarely used them while 4% said they never used.

Table 4.3 The proportional use of mobile apps testing methods from the participants' perspective

Testing Method	Never	Rarely	Sometimes	Often	Always
Crowdtesting Platforms	58%	14%	26%	2%	0%
Automated/Cloud Testing Tools	24%	22%	34%	14%	6%
Testing Company	4%	6%	34%	22%	34%

Similar to (Q.1.), participants were asked another question (Q.2.) whether they used crowdsourced programming websites such as Stack Overflow, GitHub, and Stack Exchange to search and solve their programming issues. As can be seen from Table 4.4, more than half of the participants indicated that they always use Stack Overflow for searching for programming issues and their solutions. It is interesting that none of them said they do not. While 26% indicated that they very much use it and 12% moderately. Only 8% of the participants said they use it somewhat. GitHub is another crowdsourced programming platform that used slightly less than Stack Overflow as indicated by participants. 34% of participants mentioned that they use GitHub all the time while around 24% use it very much. It is surprising that none of the participants said they do not use GitHub while 26% moderately used. 16% of participants said they use it somewhat. The least popular website among the developers was Stack Exchange with 18% of them using it all the time and only 8% indicated that they use it very much. 32% of participants mentioned that they have never used it.

Table 4.4 The proportional use of the three crowdsourced programming websites Stack Overflow, GitHub and Stack Exchange from the participants' perspective

Crowd-based Programming Platform	Never	Somewhat	Moderately	Very much	Always
Stack Overflow	0%	8%	12%	26%	56%
GitHub	0%	16%	26%	24%	34%
Stack Exchange	32%	22%	20%	8%	18%

2) General expectations and/or desired outcomes from the public crowdtesting process

A. The expectation of reaching enough critical mass:

For the development of new crowdtesting methods, reaching a critical mass of testers is a fundamental aim needs to be sure to achieve. Marwell et al. (1988) stated that most people know that crowdsourcing methods rely on voluntary participation, and there is no guarantee that the critical mass of tester contributions will be fulfilled. In response to the survey (Q.3.) of whether a critical mass of testing could be achieved, interestingly, none of the participants answered a definite 'No'. While 57% of the developers believed that this was possible, 43% answered 'Maybe'. For further exploration, the participants who answered with yes or maybe, were asked "what the expected benefits were that they could obtain when dealing with the public testers". Most of the participants agreed that the distribution of tests to the public and the involvement of public testers with different levels of experience and from different backgrounds/environments, would help to cover more mobile devices. Moreover, discover more issues faster than traditional crowdtesting methods. Some participants highlighted that this method would provide more results and present better feedback more rapidly. Consequently, reducing the time needed to finish the testing process. A few numbers of participants indicated that this way of testing would also enable the study of more human behaviors according to the particular pattern of behavior from each crowd tester. Other participants reported that testing by the public testers could give more useful results than individual testers, and it would lead to improving the developers' skills based on collected feedback. Two participants believed that public crowdtesting would help in performing testing many times in the early stages of the mobile app development life-cycle. One respondent stated that

"Because the test would be opened to whole testers in the world, this gives more variety in testing scenarios, techniques, and different tools in testing, which reduces the need for testing apps by companies."

B. Desired outcomes from the public crowdtesting process:

Majority of the participants 96% responded to question (Q.4.), which asked desired outcomes when using the public crowdtesting approach. The responses displayed a broad set of desires. The most common desire was the ability to execute more realistic tests involving a number of testers larger than possible with other means, rather than using artificial environments or designated testers. In addition, respondents expected finding more issues and in a short time. This shows that distributing the tests on a larger-scale and the time to response are crucial for most of the participants. Some participants mentioned that they hope to use public crowdtesting to enhance testers' and developers' skills by providing more testing information

and knowledge, storing the created testing scenarios and cases for later use. Furthermore, the importance of improving communication between developers and testers in industrial and academia domain was highlighted together with the possibility to exchange experience across environments. A few participants have indicated that the ability to distribute tests on a large scale to cover a variety of devices and OS versions is one of the critical and hoped for outcomes of public crowdtesting. Another group of participants showed their strong desire to obtain useful testing reports including all possible issues, exceptions or non-logic operations; as well as detailed information for each crowd tester who performed the test. One participant expressed the concerned about working with unknown testers and wrote:

"provide a secure way to test apps with protection for identity especially for app ideas" is a significant factor in using crowdtesting."

In our opinion, the participant here means that the desire to provide a secure way to be more confident about dealing with public testers. Another participant wrote:

"I hope that testers have a good technical /programming background as this may lead us to perform Gray-box Testing, which is better than Black-box testing."

We believe this participant's desire is the ability to provide a way that supports developers to perform the test in two different ways as a physical component on mobile or perform as a piece of code. This is what Gray-box testing means compared to the Black-box testing related to high-level testing (components). Surprisingly, in our view, one participant highlighted the importance and necessity that the use of public crowdtesting should be unbiased to a particular group of testers for any reason. The last desire extracted from the responses was the possibility of providing a dashboard to display a good sampling of data and metrics.

3) The essential requirements for public crowdtesting approach

A. Typical starting keywords of the search for issues:

The responses to the question (Q.5.) highlighted the way developers search for solutions when confronted with any issue. The responses show that Mobile device model represents the highest percentage 45%, followed by OS Version 29%, mobile platforms (e.g., iOS or Android) 16%, and brand/manufacturers 10%. It is clear that the model of devices is the first and most important element that the developers look for during development and testing processes of mobile apps. Whereas, the brand is the least important element to be searched for.

B. The preferred method for posting or defining issues:

Defining issues is simple and any misunderstandings that arise might be due to the unclear explanation. In fact, issues can be defined correctly in many ways. The responses to (Q.6.) show that: 74% of participants prefer to use title and general description similar to Stack

Overflow for defining their tasks and problems. While 20% of the participants found the structure form (divided into sections, e.g., payment method) more suitable for them. While 6% of participants prefer to use a simple forms as much as possible.

C. Bridging the gap between developers and crowd testers during software testing processes

The direct interaction between developers and testers is vital to perform an effective crowdtesting process on a large-scale. In this study, participants had were asked (Q.7.) whether they considered the direct interaction between developers and public testers important during the testing process. The responses show that none of the participant developers indicated that this is "Not important or Slightly important", while 70% agreed that this was very important. 10% they answered that this is important, and 20% fairly important. Overall, it can be said that all of the surveyed developers agreed on the importance of the direct interaction between testers and developers rather than the need for middleman crowd manager or leader during the testing process.

D. Issues' reporting method

The next question (Q.8.), asked whether the difficult reporting system will negatively affect public testers' contributions. The responses revealed that 86% of participants completely agreed that the difficulties of using test results reporting form will significantly affect the enthusiasm of the public testers to participate and use the crowdtesting approach. Likewise 55% of participants have strongly agreed and 31% agreed that this would significantly affect. On the contrary, 4% disagreed that this would have any effect. The remaining 10% responded neutrally that this might have negative impacts on the participation of the public testers.

4) Measuring level of trust in the public testers

A. level of confidence in the public crowd itself

In question (Q.9.) developers were asked about their confidence in testing their apps via public and international testers. The responses showed that the proportion of developers who trusted testers from any part of the world 68% is significantly higher than the ones who did not 32%. About 69% of the developers who replied negatively to this question justified "why?" Their responses covered a range of reasons. Some participants mentioned that their reason for the lack of trust is linked to the security of data (lack of identity and guarantees), as one of the developers said:

"An idea could easily be stolen and published before finishing the app development process."

Other participants' reasoning is linked to the level of education and technological development of some countries. While a few numbers of participants indicated that the main reason for not trusting is that the participation of public testers could only be for making money. Only one developer had a somewhat neutral response, which mentioned that trusting the public testers from different countries depends fundamentally on the specific region of the world that the mobile app targeted; in that case, the developer does not trust other testers from the particular region. The 68% of the participants who indicated that they would trust testers from any country in the world, were asked a further question about how much they would trust the information provided by these public and unknown testers. The participants' responses were equal 8% for too little and a little, 48% moderate, and 36% much while none of them answered with very much.

5) Evaluation of the performance and quality of work

A. Ensuring the correctness of the way the test was performed

The responses to the question (Q.10.) discussed how developers will know that the public testers have actually completed the test and hence produced several possible solutions. A review of the solutions indicated that most of the participants mentioned they could know that through the detailed description of the testing plans, test cases or testing scenarios that are reported by public testers. Additionally, almost half of the participants mentioned that repeating the testing steps implemented by testers to reproduce the same issues could also be a possible solution. A minority of the participants indicated that they must integrate a tracking tool to capture and record the testing results, processes, and activities carried out by testers. Three participants believed that the backgrounds or practical experiences of the developers might help in that situation. Two participants considered that making an issue or more intentionally in the app can be one of the best solutions for measuring if the testers actually executed the test. Interestingly, one developer pointed out that asking one or two precise questions at the last stage of the testing process is an accurate method of measuring if the testers actually conducting the testing process with integrity.

B. Evidence on the validity and accuracy of results

Question (Q.11.) asked participants how they would crowd testers to prove that the results are correct. Several different possible solutions were provided. The suggested solutions included the provision of images, video recordings, and textual reporting and automatic reports (e.g., log file. From the data presented in Table 3.11, most of the participants gave a preferred solution a textual explanation (A detailed report including full issues description, steps to rediscover the issue, test cases used) as evidence of accurate results. Other participants mentioned screenshots as evidence. Another group of participants required video recording

as a solution. The remaining possible solution as indicated by a few participants was to the importance of automatic reports (e.g., Google Analytics or Fabric Crashlytics) to prove that the results are correct.

Table 4.5 Important factors used as evidence on the accuracy of results

Evidence	Percentage (%)
A detailed report (including full issues description, steps to rediscover the issue, test cases used)	42%
Screenshots of issues	25%
Video recording of testing process	16%
An automated testing report for each test cases (similar to Google Analytics or Fabric Crashlytics reports)	11%
Automatic log files	8%

6) The incentives and motivation for testers and developers

A. The attractive elements to work with the public testers

90% of the participants responded to the open question (Q.12.) about the features that would attract and encourage them to work with the public testers to execute your tests and would make them leave working with testing companies. The responses covered a broad range of views that are organized into six categories:

- **Better quality:** Most of the participant agreed that obtaining fast and accurate testing results could be the main reason to deal with the public testers.
- **Lower cost:** Another group of the participants mentioned that the lower payment cost would be another reason for that;
- **Flexibility:** Only two participants referred to the flexibility for repeating the test more than once and any time during the development process as another reason that would motivate them to work with the public testers;
- **Diversity:** This is related to the need to cover a wide variety of environments, cultures, processes and steps for testing mobile apps. From participants' responses that belonged to this category, four sub-categories were identified:
 - *Test diversity:* The majority of participants mentioned the need to use a diverse set of real-world testing scenarios, test cases, techniques and steps for testing mobile apps;

- *Hardware resource diversity*: Other participants pointed out that the ability to cover a large variety of mobile devices models and OS versions are another reason to deal with the public testers;
- *Human knowledge diversity*: Two participants considered the accessibility of various levels of testers' experiences is also essential factor. Another participant said that

"the ability to find testers adapted to many different functions or activities is really important."
- *Human behavioural diversity*: Only one participant had considered the possibility of covering a large variety of end-users behaviors as an important feature;
- ***Organization and User-friendliness***: A small number of participants expressed that good organization of the testing processes, issues reporting mechanisms, and supporting free automated testing tools were also considered important features that may motivate them to leave working with testing companies and start working with the public testers. One participant mentioned that using free tools to list issues, in turn making developers aware of the complete problem is important. Three participants mentioned the importance of the ease of use of the crowdtesting platform. Two participants indicated the need for a good communication method between the public testers and developers;
- ***Other responses***: There were two interesting responses; the first response was:

"The patience in repeating questions and frequent communication without increasing service charges or feeling bored is considered one of the significant reasons to work with testers."

The second response was a neutral

"choosing the testing method between either companies or crowdsourcing depends on the type of the app itself whether it's allowed to be tested by the crowd."

B. Possible incentives that could offer to the public testers

To acquire further knowledge, the participants were asked another open question (Q.13.) about the incentives they would be willing to offer to public testers to motivate them and increase the participation rate. The responses covered a vast array of ideas to encourage testers to work sufficiently well in their testing role and in return provide adequate recognition for their good work. Some of these potential incentives was considered as a high significant incentives from the responses such as providing money, and gift cards and/or vouchers. We

think that the participants may considered the gift cards and/or vouchers is also small amounts of money, but in a forms gift cards and/or vouchers. While a less significant proportion of incentives included provision of free apps or allowing the use of a paid version of the app, invitations to training courses, providing certifications, and providing more knowledge related to testing scenarios and activities.

7) Required information for effective crowdtesting process

A. The required test information for defining testing tasks

The responses of the participants to the open question (Q.14.), in related to important information that developers must provide to testers for defining testing tasks clearly, which can assist them in achieving a correct crowdtesting process. Only 88% of the participants responded to this question, 5% indicating that the testing requirements are important without any explanation. While the remaining 83% of them provided interesting responses. From these responses, eight primary topics were identified:

- **Functional Behaviors:** Most of the participants mentioned the functional requirements of the apps, the components and the expected behaviors (output) as important information for announcing any testing task.
- **Mobile device information:** The need to provide the mobile platform, model, and OS versions details that need to be tested against the app as mentioned by most of the participants;
- **Timing information:** A small number of participants emphasized the importance of the estimated time needed for singular test cycle, alongside the deadline for submitting the complete test reports and obtaining fast results;
- **App information:** Other participants indicated that type of app and URL is necessary to provide, as recently many apps are launched with some even sharing names. and all may have the same name. Interestingly, only one participant said that "logo or image of the app is important for testers to know which app they need to test";
- **Test information:** They are related to the need to provide a full description of the apps and testing scenarios or test cases. Some participants specified the need for a complete description of the whole app's and the test instructions. other group of participants indicated that providing testing scenarios or test cases by developers rather than always created by testers. This might be useful for beginner testers to perform an accurate test;
- **App development information:** Only two participants mentioned the importance of providing the source code of the app in definitions of specific type of testing tasks (if

needed), and thus decide if the issues belong to the mobile device characteristics or in the code itself;

- ***Users characteristics:*** Interestingly, very few (3) participants indicated that target users and their characteristics (e.g. location, language, age, working domain, etc.) are also important information when defining the task.

B. The required test information within test Summary Report

The main information for submitting useful testing reports compared to open question (Q.14.), the same number of participants 88% also answered the open question (Q.15.) in relation to the information that testers must be considered in their reports to provide high-quality testing results. 7% participants did not provide clear information while 81% of them provided enlightening responses. Among provided responses, six primary pieces of information were identified:

- ***Testing environment:***The details of mobile devices used in testing (platform, model, OS version) and its characteristics are classified as vital and need inclusion when submitting reports as suggested by the majority of participants;
- ***Tester information:*** A small minority of participants mentioned that due to dealing with public and unknown testers, personal information (including name and contact information) and geographical information are deemed beneficial if included in submitted reports;
- ***Execution information:*** This relates to the need to submit information about the testing process that was performed test cases or scenarios used, a clear description of the steps that testers followed, error messages, which could enhance the quality of the report. Interestingly, only two participants mentioned the importance of providing the number of test repetitions and time taken for each test cycle;
- ***Issue information:*** Most of the participants highlighted the importance of receiving a clear description of issues within submitted reports, including issue id, issue name, category or type of issue, a priority of issues, severity, and actual results. Few participants them mentioned that videos or screenshots of the issues are key and must be included in the submitted testing reports;
- ***Supplementary information:*** Two participants stated that receiving additional information such as solutions or suggestions for solving issues, or expected causes of issues within submitted reports would be significant.

4.7 Main Discussion

This section provides a discussion of the results of the exploratory survey study. Based on the quantitative and qualitative data gathered from the participants, some initial conclusions have been drawn to address both parts of the second research questions (RQ1) as outlined in Section 1.3 separately.

P:1 To what extent are mobile app developers keen to work with public and unknown crowd testers with different levels of experience to perform their testing tasks?

The overall results of the exploratory study show that some app developers are not familiar with crowdtesting, or they do not have any knowledge about the testing methods used by testing organisations. This lack of understanding was particularly evident from the responses given by our participants and further evidence supporting this was provided by Illahi et al. (2019) who stated that crowdsourcing is still in the early stages, and too little developers presently know about it. This could be attributable to the gap in the literature in terms of limited publications on the discussion of crowdtesting services for mobile apps, and standards and operational guidelines on how it functions, particularly in respect of how it is used by testing organisations (Zhang et al., 2017b). In our view, the responses of our participants were surprising. 58% of the participants state that they never used crowdtesting, while almost the same percentage of 56% had used testing organisations. In the available literature, research has shown that most of the existing big and well-known testing organisations have used the crowdtesting method to test their apps (Alsayyari and Alyahya, 2018; Leicht, 2018; Zogaj et al., 2014), and some of them changed their name to become crowdtesting organisations (Tran, 2020). Such testing companies do not share specific information of their workflows with their clients, as they have their own crowd tester communities to perform testing. Therefore, their clients may not recognise that these companies use the crowdtesting approach to outsource the test quickly. It was espoused by (Guaiiani and Muccini, 2015) that most of the big and well-known testing organisations such as Clariter, uTest, Telcom Italia, Pass Brains, and Bug Finders follow the crowdtesting approach in their usual testing method. Therefore, we believe that all these developers have indeed used this crowdtesting method through these organisations, but they are not aware of the fact.

The results also revealed three main indicators of how developers are keen to accept the public crowdtesting approach and their willingness to engage with the public and unknown testers. **Firstly**, the evidently frequent use of public crowd-programming platforms, which explicitly shows inadequate knowledge regarding crowdsourcing. Almost 90% of the participants responded in the affirmative regarding the use of programming websites such as Stack Overflow, GitHub, and Stack Exchange. These websites are public crowdsourcing

platforms for programming/coding, which deal with the public and unknown crowd programmers (Vasilescu et al., 2013). This brings to light how much mobile app developers agree with the approach of working with unknown crowd workers, not only for programming but also perhaps for testing. **Secondly**, the participants' desires and expected outcomes from the use of such type of testing. The willingness of almost 90% of participants to move and work with public crowdtesting platforms is also evidence in their approving the use of this method if the fundamental testing features previously mentioned by participants in their responses to Q12 are applied. Furthermore, there is a positive outlook regarding the finding of sufficient critical mass when using public crowdtesting has also reported by our participants. Also, their future expectations and the outcomes they desire to obtain from the use of the crowdtesting method regarding the broader distribution of the test can be broken down into: reduction of time and cost, diversity in testing results, improvement of knowledge and experience, and enhancement of social networking and work cooperation between experts in industry and academia. These responses are preliminary indicators of the willingness of developers to accept the use of the public crowdtesting approach. **Finally**, the last indicator the level of confidence the developers who participated in our research had in public testers. 68% of the participants confirmed their trust in testers from any part of the world, thus showing strong approval to work with public testers. They expressed their happiness and enthusiasm to work with these testers and provided a list of possible solutions to certain problems (see Q.11). The implementation of such solutions could increase their trust in those testers. These indicators clearly show the willingness of developers to accept to dealing with the public and unknown testers in the future.

P:2 What are their critical requirements to use the crowdtesting approach and the key factors to consider to mitigate their concerns about dealing with these crowd testers?

Based on the related findings presented in Section 4.6, we are able to describe how the proposed crowdtesting approach can be practised more effectively by developers. We have identified a set of essential requirements that have to be implemented to represent real-world practices in developing and testing mobile apps from the developers' perspectives. We think that the need for these requirements is based on two main reasons; either these requirements have not been implemented in a manner suited to the way in which developers practise in their daily routine of work or have not been considered at all by the currently available crowdtesting approaches. Implementing these requirements in our approach could increase developers' confidence and mitigate their concerns about working and interacting with public testers in future practices. Some extracted requirements are relevant to the functionalities of the approach, which need to be achieved (e.g., adding a task, selecting testers, submitting results, evaluating results, etc.) and the others are related to the characteristics of the approach (e.g., the flexibility of the work, reliability, privacy, security of data, scalability,

availability, speed of the approach, and simple interfaces). Therefore, we have divided them into two categories: functional (see Table 4.7) and non-functional requirements (see Table 4.6). Some requirements are obvious and do not need more explanation; they have presented in Table 4.7 and 4.6 only. The distinct and key requirements which will enhance the knowledge gap in the literature have also been presented in both tables and are discussed as follows:

1. Allowing for large-scale test distribution

The need for the distribution of the test on a large scale is evident from a number of participants' responses to many questions. Relevant example of this is their desire to gain to gain more knowledge about human behaviours and interaction with the app, performing the test on all mobile device models and operating system versions, participation of testers with different experience levels, performing the test very quickly, and working with worldwide testers from different geographical regions. According to [Alyahya \(2020\)](#) specialist testers are always appropriate for discovering performance and security issues. This means that using testers with limited skills or inviting the participation of end-users who do not have any testing skills could be suitable for finding out normal functional issues. We believe that implementing this requirement will help obtain more useful compatibility testing results (discover more issues) from end-users and public testers rather than from those who work in testing organisations. This is because they have different levels of skills, background, and behaviour when interacting with the app, which can lead to the discovery of unexpected issues related to the app from different angles.

2. Finding an easy and flexible method for dividing and defining tasks

The need for a flexible method of defining and breaking the task down into smaller tasks was noticed in the responses from 74% of the participants. This is probably due to the fact that the current defining mechanisms of test requirements in existing crowdtesting approaches take more time and effort to break the app into a set of tasks ([Alyahya and Alsayyari, 2020](#)). Also, it does not provide developers with sufficient space to describe all the task's requirements with clear completion criteria. Consequently, this will require a longer time for testers to accurately understand the task, which will reduce their enthusiasm for participation as argued by ([Li et al., 2016](#)). The authors in ([Li et al., 2017](#); [Lykourantzou et al., 2019](#)) argue that breaking down the complex tasks into micro-tasks is very useful in the crowdsourcing context. However, this issue remains a big challenge in most of the crowdsourcing approaches. This is due to the difficulties involved in finding a specialist expert to divide the complex task into smaller tasks. Conversely, [Haas et al. \(2015\)](#) had a different perspective, and they argue that the decomposition of macro-tasks mainly depends on the type of tasks and the possibility of dividing them up. Sometimes it is challenging to split up the macro-tasks and allow them to be completed by multiple workers. Because

this could result in the context information being lost when the tasks are split up, and it is better to leave them as they are. Otherwise, it could lead to a misunderstanding of the task requirements, culminating in an undesirable or low-quality result. It seems to us that this reason applies to the workflows of current crowdtesting approaches, where it requires more QA members. These members in collaboration with the developer, often determine how the app or big tasks must be decomposed into smaller tasks. As a result of those members' involvement, the decomposition and definition method of tasks can be costly (Cheng et al., 2016). Implementing this requirement in our approach will reduce the risk of receiving valueless results and the cheat rate between testers (Hossfeld et al., 2013). It could also substantially impact the motivation of more testers (Qarout, 2019) and improve developers' confidence, and reduce their concern about public testers. A similar conclusion was reached by Wood et al. (2019) who indicated the need for further investigation, in a crowdsourcing context, to understand which crowdsourcing workflows can support such a requirement and how it can be designed to improve outcome quality. Recently, Alyahya (2020) argued that implementing such a requirement is vital and needs to be considered in future approaches.

3. Guarantee the tasks' privacy and security

Providing more constraints to guarantee the privacy of the published task is a critical requirement. The need for such a requirement is indicated by the participants, where they express their concern about the unauthorised use of the published apps or source codes by other public members. This is because some participant testers may be anonymous users, which are different from traditional testers (Daniel et al., 2018). Thus, if the tasks are defined and published online to the public, anyone will be able to access them, subsequently, important data could be leaked or stolen. The author cited in (Leicht et al., 2016b) concluded that controlling the privacy of tasks in crowdtesting might not always be beneficial; this is based on the results obtained from performing crowdtesting on two different apps: an industrial enterprise app and the other a mobile banking app. Similarly, the author in (Liu et al., 2019) agreed that this function is not suitable for some mobile apps with high density. In our opinion, controlling privacy of tasks not always unsuitable, it depend on the type of testing that will be achieved. Although the developers are able to publish anonymity data of tasks to the testers by leveraging data privacy methods (e.g., K-Anonymity), this problem is still challenging for most developers (Li et al., 2017). This is most likely because existing methods may lower the quality of testing results as the testers can not access the precise data. Therefore, we need to design a more practical and useful function with more constraints on the tasks in order to ensure the safety and privacy of confidential data of the published app. Such a function needs to effectively control unauthorised access by testers who are not granted access to browse the data during the execution and/or after the completion of the testing task. Further evidence on the need for a method to guarantee task security and

privacy was espoused by (Alyahya, 2020), who emphasised that this function is one of the frequently mentioned challenges in the majority of the previous literature studies.

4. Diversity in the knowledge and experience of testers to discover more issues

Allowing testers with different levels of background, knowledge, and testing experience to participate was another requirement mentioned by most of our participants. This is perhaps due to the fact that the distribution of the tests in current approaches is extremely limited. In such approaches, testers may execute test tasks when they have similar skills (selected by the testing organisations). However, in this case there would be a bias in the results of testing, as the issues discovered would be skewed (one-sided). This can lead to difficulty in finding other testing results (Liu et al., 2019) and a waste of time and resources (Cui et al., 2017a). One reason for this one-sidedness is that testers with similar skills may be prone to following the same testing guidelines and procedures and this way of testing may not be suitable (Liu et al., 2019). The developers would like to discover all unexpected issues to ensure their apps' compatibility with all mobile devices. Alyahya (2020); Habib et al. (2019) indicated that the variety of testers' skills and experience is essential to finding all issues but is not considered by most current crowdtesting approaches (Liu et al., 2019). Therefore, implementing such a requirement could be a better solution to find more issues from different sides and this could make up for the shortcomings of existing approaches. We believe that testers with different testing skills and experience (novice to expert) will effectively represent how real -users use the app. Human behaviour may lead us to conclude that, these testers may spend more effort on performing the test and providing better results when they get paid. Thus, such testers are more likely to identify more unexpected issues, which are very similar to those which would be discovered by target users after using the app. Such a way of testing could also help developers to gain more insight into end-users' behaviour and develop high-quality apps with better users' experiences in the future (Zanatta et al., 2017).

5. Selecting testers based on specified criteria

Another crucial requirement of the crowdtesting approach is that of accessing and selecting the most suitable testers based on specified criteria. The importance of this requirement was agreed upon by a small group of our participants. This was probably due to the long time they often spent when involved in such selection through most of the current crowdtesting platforms (Alyahya and Alrugebh, 2017). Since our approach focuses on dealing with public and unknown testers with different skills and experiences, the inclusion of such a requirement will increase developers' confidence and reduce their concerns about working with these testers. This would consequently help to collect more accurate testing results. The authors in (Liu et al., 2019; Yu et al., 2019) have also emphasised the importance of building this

function into any crowdtesting approach. Moreover, suggesting a list of eligible testers for each task will assist developers in the selection process, so it will be considered in our approach. Incorporating some of the ideas and recommendations in our approach can play a key role in the selection process of suitable testing task and testers and thus improving the quality of the test results (Wang et al., 2019b).

6. Cooperation and interaction between developers and testers.

The direct interaction and collaboration between developers and testers is considered a primary requirement for the proposed crowdtesting approach. All survey participants indicated the importance of this requirement. This is probably due to the long time taken to accomplish testing processes in existing approaches because of crowd managers and leaders (Alyahya and Alrugebh, 2017). Testers' responses in (Guaiani and Muccini, 2015) also emphasises this. Testers argued that one of the main issues they faced when working as a crowd in most current crowdtesting organisation is the delay that takes place because of managers and leaders who organise and lead the testing process. In our view, this length of time and the delay may affect continuity in terms of the testers performing more tests or not. The need for a short time for test completion was equally evident from participants in several responses. Another reason for the need for direct interaction during testing is that sometimes misunderstandings arise between the manager or leader and testers due to incomplete test information or delay in providing them, receiving feedback on work quality assessment, or rewarding-cost. This misunderstanding impedes job productivity and test cycle success. This according to Bano et al. (2019) who have identified as the major problem in teamwork usually arise because of the misunderstanding between the team members and team leaders, which will affect the quality of the crowdtesting process. Yu et al. (2019) also emphasised that the communication channel between developers and testers is an essential aspect to consider to improve the efficiency and quality of crowdsourced testing and development process. Therefore, building a cohesive and direct relationship between developers and testers will reduce the time taken and the potential misunderstanding which may occur due to the absence of some important information. Furthermore, it could reduce the budget earmarked for the manager and leader assigned to each test.

7. A simple and intelligent reporting mechanism

The easy-to-use and simple interfaces of the issue-reporting mechanism are considered critical requirements that need to be focused on more when developing a new crowdtesting approach. Evidence regarding the usefulness of this requirement is provided by 86% of our participants. They considered the difficulty of the issue-reporting mechanism to negatively affect the testers' enthusiasm to participate in the crowdtesting process. This is supported by (Rosson et al., 2002), who proved that the system's complicated interfaces could reduce the use of

the system by target users and, at the same time, lead them to express their dissatisfaction with continuing to use the system. This could result from the different levels of the testers' experience when the crowdtesting process recruits not only professional testers, but also testers with different experience (may have little experience) and ordinary end-users who are inexperienced in testing but are interested in performing some testing activities such as functional, compatibility, and usability testing (Chen et al., 2019). Each of them behaves and interacts differently with the same system. It may take some a long time to understand the system before submitting the report or lead to inaccurate results and low reward value. Such testers probably require support through the construction of a simple but intelligent reporting system that could quickly and automatically be inputting the data they would like to deliver. Therefore, there is a need to think of how to build an issue reporting mechanism that best supports ordinary testers with different experience and make up for the gaps in the reporting mechanisms of current crowdtesting approaches. The author in (Alyahya, 2020) implies that the contribution made by improving this requirement is limited, and further research is required in this area. As our approach supports the distribution of tests on a large scale, implementing this requirement in our approach will motivate more testers to participate and deliver accurate results quickly. We believe that incorporating some of the ideas, recommendations, and techniques (Dal Sasso et al., 2016; Zimmermann et al., 2010) in our issue report systems can play an essential role in facilitating the submission process of the test report and improving their quality.

8. Effective incentives method and payment schemes

Providing an effective incentivisation method with an appropriate reward schema to motivate testers is one of the critical requirements for the successful operation of crowdtesting approaches. The need for useful incentivisation schema has recently been indicated by many research studies (Alyahya, 2020; Gao et al., 2019; Knop and Blohm, 2018; Sari et al., 2019). Most participants provide further evidence for the vital need for a practical incentive method and fair rewarding schema. This may stem from the fact that rewarding schemes used by current crowdtesting approaches are low and that the value of the rewards is fixed regardless of the amount of effort that testers invest in order to find issues (Gao et al., 2019). Our results show that most of the participant developers believe that vouchers/gift cards, small amounts of money, a free app, job offers, etc. are other reward options that could be offered to the testers. Unfortunately, such reward options are often deemed inappropriate and perhaps do not fully consider the amount of effort invested by testers. This implies that a group of developers may think that testing is not a serious job; it is only a matter of simple cooperation. Thus, they would prefer to provide goods instead of payments to testers. Certainly, this is not be suitable and would reduce the motivation of testers or lead to slower task completion times (Lykourantzou et al., 2019). Testers who need to earn a living may

not show interest when rewarded with free access to the app, gift cards, ability to work in other apps etc. Developers should be mindful of the cost regarding the gig economy and globalization and pay testers according to the standard of living in their respective countries. In fact, testing is not easy and requires a great deal of experience and qualifications to attain a professional standard, similar to any other job, because of the different nature of mobile app functionalities and behaviour. This raises a question for future researchers in relation to the significance of studying the nature of the testing process, depending on what testers do. According to [Brabham \(2008\)](#); [Kaufmann et al. \(2011\)](#) monetary incentive (money) is the most dominant motivation for workers. Therefore, we strongly recommend considering money as the main reward for testers. We believe that the monetary value must be based on a set of identified criteria, and commensurate to the type of test performed, the kind of app, the test's complexity level, and testers' efforts and performance in finding more issues. These responses have focused our attention on the significance of providing additional incentives to ensure the reliability of good crowd testers and encouraging others who provide lower results quality to work harder to gain better results. Hence, the list of incentives mentioned by participants could be additional motivators (bonus) to motivate testers to provide more accurate results.

9. Clarity of task requirements and testing reports

Clarity of testing tasks and testing reports are former and primary requirements for the crowdtesting. [Liu et al. \(2012\)](#) have shown that these two requirements considerably affect the successful performance of testing. Clear and detailed explanation and presentation of the tasks' requirements will mitigate the ambiguity and complexity of executing tasks and help testers obtain more accurate results ([Guaiani and Muccini, 2015](#); [Knop and Blohm, 2018](#)). Furthermore, the detailed information from testing results will help developers in the evaluation stage and thus enable testers to gain better reward value ([Meier et al., 2013](#)). For academic researchers and industry practitioners, the lack of sufficient and detailed information still represent barriers to the successful practice of crowdtesting by testers and developers alike. However, little research has outlined solutions for the mitigation of such a problem ([Alyahya, 2020](#)). The need to address this problem and implement such requirements was echoed significantly from all our participants' responses. This is because most of the current crowdtesting approaches do not provide a detailed amount of information, leading to difficulties in the testing process. This is consistent with the findings of [Guaiani and Muccini \(2015\)](#), which demonstrated that a good number of the testers had indicated that the amount of information provided by the testing companies they work with is not sufficient to carry out the test to the necessary standards. Testers only receive information about the app itself, such as test scenarios, information about specific inputs, and occasionally information about the devices that need to be tested.

For the developers:

For the developers: To ensure that the task requirements are defined clearly and in a structured way, which could be understood by testers with different levels of experience, the following criteria must be included. Details of the mobile devices required for testing will help save time, effort, and avoid the time-consuming execution of test cases on devices which are not required for the test (Afzal, 2007). As a result of the fact that each app includes several functionalities, and a test case needs to be created for each functionality. In respect of this, providing and standardising the test cases or scenarios among testers with limited experience might be useful. This may enhance testers' knowledge and experience, and help obtain a broader background of generating better scenarios and/or test cases in the future which would, in turn, lead to performing more accurate testing processes (Alyahya, 2020). The standardised tests could also help developers to evaluate the testers' performance and find the gaps in the applied testing techniques to improve their experiences. The general description of the whole app could help obtain a broader background of the app's purposes and its functionalities; consequently, they can play with the app from different angles. This in turn, would lead to a higher probability of finding more testing issues. Moreover, providing clear information about the issues that have been solved will help testers perform a regression test on all changes made to ensure that they do not negatively affect other functions and help maintain the quality of the app (Afzal, 2007). We emphasise the importance of providing this information in order to complete the test quickly, accurately, and with less effort.

For the testers:

To ensure that the testers have indeed completed the test and to guarantee that all required details of a task's results included in the submitted report, will not be rejected, a textual description of issues with a screenshot and video (if possible) must be included (Alyahya, 2020; Liu et al., 2019). The textual descriptions should contain a clear description of the steps followed by testers. This will help developers to reproduce issues and therefore reward testers very quickly. Moreover, additional useful information (optional) can be provided by testers, including the expected causes of the issues they found in the test might help developers diagnose the issues and fix them quickly. Such additional information can enhance the quality of the report (Liu et al., 2019), and help developers understand the contents better. In our view, all the augmented pieces of information presented above might be considered as assistive factors for measuring the quality of testing reports gathered by testers and thus help provide appropriate incentives for them. Given that our approach allows for the participation of testers with different levels of experience, the aforementioned information and key features recommended by (Davies and Roper, 2014; Karim, 2019) is essential to ensure the correct execution of tests and to reduce the developers' concerns about the accuracy of the gathered

results. In addition, it will ensure effective delivery of testing reports, and fair evaluation by developers.

10. An effective method to control the overload work on testers

Providing an effective method to control the overload work on testers is one of the most critical requirements for our approach. A good number of our participants indicated the importance of this requirement. They suggested that there should be a task-announcement dashboard that displays all published tasks and provides worldwide testers with the freedom to access it and select the appropriate tasks based on their ability and free time. This is necessary because most of the current crowdtesting platforms are still unable to control work overload on testers, as demonstrated by [Alyahya and Alrugebh \(2017\)](#). Indeed, on these platforms many developers are selecting and inviting the same tester to perform different tasks at the same time, which will be exhausting for the tester while others are available and waiting for a new job. This causes a delay in performing the test, and in the development, and delivery of the app to the market ([Daniel et al., 2018](#)).

11. An effective method for evaluating testing reports

Providing an effective method for evaluating testing is an essential requirement for all crowdsourcing platforms ([Daniel and Farhad, 2014](#); [Jiang et al., 2018](#)). Our survey participants mentioned the need to build an automated method to evaluate testing reports, however, this automatic evaluation method is either impossible or may only guarantee a minimum quality ([Daniel et al., 2018](#); [Sánchez-Charles et al., 2014](#)). As our approach involves the participation of testers with different experience, it would probably result in the production of testing reports and the interpretation of results with different quality levels ([Jiang et al., 2018](#)). Therefore, we think that direct evaluation via developers would be the best solution as developers have a large amount of knowledge and resources to evaluate those testers. Another complementary requirement related to the effective report evaluation method has also been indicated by most of our participants' responses to other questions. The complementary requirement is the need for fair evaluation criteria to guarantee the different quality levels of testers' work and results. In our opinion, this is because some of the current crowdtesting approaches usually involve recruiting testers with a high-level of experience and evaluate them using unified high evaluation criteria. These assessment criteria may not be fair enough for evaluating the work of public testers with different levels of experience. For example, time is a critical factor that needs to be considered because people with limited experience may take more time to finish the test. Another factor that needs to be taken into account is the complexity level of the task due to the fact that humans have different levels of ability and different behaviour towards performing work ([Tavanapour and Bittner, 2018](#); [Yu et al., 2019](#)).

12. More insight into performance of testers work

Providing a summary report of the assessment results on each performed task to each tester is also a critical requirement to be implemented in our approach. This requirement would increase tester satisfaction and thus provide an incentive to work more and so improve work productivity (Chen et al., 2020). The results of the conducted study on the testers in the work of Guaiani and Muccini (2015) support our idea regarding the importance of sending an automated report with detailed information about evaluation criteria to testers; since 75% of their participants mentioned that most of the big companies they work with do not provide them with the results of quality evaluation criteria or an idea of how their work performance was rated. Such information could function as guidelines for testers, which would help them understand any weaknesses in their work and improve their performance in subsequent tasks (Jiang et al., 2018).

13. A collaborative learning and knowledge sharing environment

The need for large- scale communication and social collaboration among global developers and testers is evident from our participants' responses. We assume that the rapid improvement of mobile device models, changes in human work lives, and different ways of human interaction with apps have resulted in the need for such collaboration. Definitely, developers and testers would like to collaborate with each other and share knowledge in order to be sufficiently familiar with test information to build more effective apps services that are compatible with all device models (Zanatta et al., 2016). Therefore, we consider the provision of an online environment for large-scale communication, work cooperation, and collaborative learning and sharing of knowledge to be a critical requirement in our approach. Some of our participants' responses expressed the desire to implement such an environment. They suggested there being an online collaborative wiki rather than an online FAQ (frequently asked questions). This could be because they have little awareness of all incompatibility issues, which leads them to replicate similar issues previously discovered and solved by other developers. This practice would inevitable lead to more time-consumption in the development of new apps (Villanes et al., 2017). Therefore, developers want to have a public space that documents such information and which they have access to in order to search for solutions to the problems they may face. In line with previous studies (Illahi et al., 2019; Wnuk and Garrepalli, 2018; Yu et al., 2019) have emphasised the paramount need for online space where worldwide developers and testers could share all related knowledge, discuss issues and get support (Villanes et al., 2017). We argue that the implementation of this requirement will significantly encourage more global developers and testers to use our approach as well as provide a better understanding of the results of compatibility testing, including issues, their causes, possible solutions, and conducted testing scenarios (Wnuk and Garrepalli, 2018). This will help developers in delivering better-accepted apps of high quality for researchers in

the future. Illahi et al. (2019) emphasised that improving programming skills and learning new issues and solutions are more significant motivational factors than monetary rewards for software developers to use crowdsourcing technology.

14. An effective searching mechanism with different criteria

The importance of building a searching mechanism with various searching criteria was regarded as a critical requirement by our participants. This is because diversity criteria in the searching mechanism may reduce the time taken to search for particular issues or solutions at the testing and programming stage and provide a broader set of solutions that may not appear immediately. They emphasised that the device model is the most critical and precise searching element for developers seeking different solutions. A possible reason for this could be that the device model indicates the brand and platform simultaneously (e.g., Samsung S5, iPhone XR, Huawei P30 Pro, etc.), helping them to quickly reach relevant solutions. As far as we know the implementation of the wiki with the searching mechanism with diverse criteria will obviously require several sub-requirements such as: (1) Automated tagging to reduce human typing errors regarding sensitive information of a mobile device. This would be ensured by storing sets of similar issues and solutions under the relevant tag (categories e.g., mobile device name, model number, OS version, platform, as well as issue types) more accurately (Liu et al., 2018). This is necessary because the wrong tagging of such sensitive information would certainly lead to results which are very different from what developers or testers expect. (2) An easy navigation between the list of matched issues based on defined searching criteria and a score of bad and good post. (3) A secure method of storing and presenting the knowledge which could reduce developers' concern about their code or any other important piece of information being stolen by other members.

4.8 Chapter Summary

Although crowdsourcing has gained much attention among mobile app developers as shown in Chapter 2, much still needs to be done to change the perspectives of participants who still concerns about the use of public crowdtesting for mobile app testing. This chapter has presented an exploratory study investigating developers' points of view in agreeing to work with the public and unknown testers in crowdtesting processes for mobile apps. It has identified the desirable features or properties required by developers in order to effectively use the public crowdtesting approach for testing their apps. Furthermore, it has provided information on how they can ensure the reliability of the public crowd, motivate them, and evaluate their work from the developers' perspectives. In total, the responses from 50 mobile app developers with different experience in Android and iOS from various countries around the world have been analysed. The results show that app developers are willing to use the

public crowdtesting approach if some challenges can be addressed and the factors detailed in the study are fully met. Additionally, the study concludes that the direct interaction and development of trust between the public testers and the developers is key to performing an effective testing process and to establishing a long-term working relationship between these two groups. In addition, this chapter has discussed the results in detail to answer the second research question. More importantly, the study has helped to understand the essential requirements and issues of developers' concerns when using the public crowdtesting method. In particular, we have provided a set of requirements that need to be considered when developing a new crowdtesting approach, which would reduce their concerns regarding working with public testers. The next chapter aims to discuss the method needed to construct the proposed approach, considering all the requirements identified in this study to improve operational excellence among developers and testers of mobile apps to deliver a better crowdtesting approach and perform more effective compatibility testing.

Table 4.6 A list of non-functional requirements for the public crowdtesting approach

<i>Non-Functional Requirements</i>	
Req 1	The resulting approach shall be effective and seamlessly integrated into the everyday practical workplace.
Req 2	The resulting approach shall be available for worldwide developers and testers, e.g., not specified for a particular group of testers or those with a certain level of experience.
Req 3	Each request shall be processed as fast as possible.
Req 4	The resulting approach should be able to run on all browsers without creating problems.
Req 5	The resulting approach should be capable of handling many users without affecting its performance.
Req 6	The resulting approach shall only allow authorised testers to access and perform the privately published task.
Req 7	The resulting approach must ensure privacy while documenting and presenting data.
Req 8	The resulting approach shall maintain a high level of privacy among developers and testers.
Req 9	The resulting approach shall require the developer to complete the payment process before a test cycle can be closed.
Req 10	The resulting approach shall provide a measure to prevent each tester having more than three active tasks simultaneously.
Req 11	The resulting approach shall notify testers of the tasks that have been accepted and not performed every three days.
Req 12	The resulting approach shall be capable of being expanded geographically, e.g., collect results from different regions with different device characteristics to increase device coverage.
Req 13	The resulting approach shall have a user-friendly and simple interface.

Table 4.7 A list of functional requirements for defining and distributing tasks on a large-scale

<i>Set 1: Tasks Defining and Distribution:</i>	
Req 1	The approach shall enable an effective and easy task defining and announcing mechanism.
Req 2	The approach shall enable a large-scale test distribution to cover all mobile device models and versions.
Req 3	The approach shall enable both high-level testing (features of a device), and low-level testing (code sources).
Req 4	The approach will enable developers to control the privacy and confidentiality of published tasks.
Req 5	The approach shall enable the developer to delete or terminate any task within set of tasks.
Req 6	The approach shall notify testers about newly published tasks.
Req 7	The approach shall enable developers to upload their application (.ipa, .apk, .cc) to be ready for testing.
Req 8	The approach shall enable developers to define the app into small tasks as a package under one group or separately.
Req 9	The approach shall provide enough information about and clear presentation of the task's requirements.
<i>Set 2: Testers and Task Selection</i>	
Req 1	The approach shall enable easy access and selection of testers based on specialised knowledge to work on tasks.
Req 2	The approach shall provide testers with the freedom to search and select a task based on a specific platform type (Android/iOS).
Req 3	The approach shall inform developers of the number of testers who accepted to perform the task.
Req 4	The approach shall be able to provide a list of eligible testers for performing specific tasks based on the specified task criteria.
Req 5	The approach shall allow testers to confirm acceptance of the invitation and notify developers.
Req 6	The approach shall allow developers to grant permission to testers to perform the selected task and notify testers as well.
Req 7	The approach shall notify testers about any updates on the accepted tasks.
Req 8	The approach shall allow testers to access all tasks and select or accept more than one task.
<i>Set 3: Results Submission</i>	
Req 1	The approach shall provide an effective and easy report submission method.
Req 2	The approach shall enable automatic collection of all mobile data from the tested device.
Req 3	The approach shall ensure all required testing information is completed before submission of the testing reports.

<i>Set 4: Reports Evaluation</i>	
Req 1	The approach shall provide a fair evaluation schema for testing reports.
Req 2	The approach shall be able to inform testers about feedback or changes that occur in relation to submitted reports during the evaluation process.
Req 3	The approach shall provide testers with a detailed information about their performance for all performed tasks.
Req 4	The approach shall provide the developer with a summary report of the work quality of all testers for each task.
<i>Set 5: Tester Motivation</i>	
Req 1	The approach shall provide an effective and fair rewarding mechanism.
Req 2	The approach shall allow both testers and developers to confirm the completion of the payment process.
Req 3	The approach shall enable the developer to define the app into a set of subtasks with corresponding payments, with the approach showing the percentage of the payment allocated to each task.
Req 4	The approach shall be able to provide developers with a list of non-paid testers.
<i>Set 6: Trustworthiness Management</i>	
Req 1	The approach shall have an effective ranking (classification) system for testers.
Req 2	The approach shall provide reliable and effective communication mechanism between developers and public testers.
<i>Set 7: Repository/Wiki</i>	
Req 1	The approach shall provide a suitable and simple knowledge-sharing environment for professional and beginner developers and testers.
Req 2	The approach shall provide an effective documentation mechanism of testing results and testing scenarios.
Req 3	The approach shall provide an effective searching mechanism with different criteria based on entered keywords within the text. e.g., device models, OS version, platform type, HW component of the mobile device, and types of error.
Req 4	The approach shall enable the external developers and testers to add suggestions or solutions to any issue in the wiki.
Req 5	The approach shall display all the matching testing issues based on the main keywords in the searched text.
Req 6	The approach shall provide an effective tagging mechanism.
Req 7	The approach shall enable developers and testers to navigate between search results quickly.
Req 8	The approach shall notify both developers and testers about any new issues posted in the wiki.

Part II

Research Phase 2: Development of Crowdtesting Approach

Design and Development of the Proposed Crowdtesting Approach

5.1 Introduction

Based on the literature review and key challenges and limitations of current crowdtesting approaches presented in Chapter 2, and based on the outcome from the exploratory study conducted in the previous chapter to collect the key requirements, a new crowdtesting approach is developed, allowing developers to perform effective mobile device compatibility testing. The chapter begins with a clear description of the working mechanism of the proposed crowdtesting approach. Then it discusses how the proposed approach is designed and implemented to fill the gaps in the current state-of-the-art approaches and to fulfill the fundamental requirements and needs extracted from the exploratory study presented in the previous chapter in an effective manner. In particular, it provides detailed information regarding the design and implementation of the main processes of the entire approach that is relevant to both developers and testers. It starts by describing the method employed for defining and distributing the testing tasks mechanism, followed by the method of selecting the appropriate task and/or testers required to perform the tests. After that, the chapter explains how the results submission mechanism is designed and implemented to reduce testers' time and effort and help them to send more accurate results. Next, a clear description of the implementation of the results tracking system and method used to evaluate the collected results is provided, followed by a description of the method of motivation and rewarding the participant testers. Finally, there is a clear discussion of the documentation and searching mechanism regarding the collected results, including the issues reported, their causes, and solutions in the proposed knowledge sharing wiki.

5.2 Distinctive Features of the Proposed Approach

For the sake of clarity, this section specifies the main features which distinguish our crowdtesting approach from other approaches present in literature and industry practices, while bridging the existing gaps in the literature. The distinguishing features are listed as follows:

- According to the literature, most of the current crowdtesting approaches and industrial practices have not, in the main, mainly been developed for manual compatibility testing service; only a few of them are provided support for the compatibility testing services as detailed in Chapter 2. Our approach, however, is specifically developed to fulfill this need and enables developers to perform compatibility testing by testing in two different ways: testing the physical feature and/or a piece of code to ensure the compatibility of all mobile devices with the functionalities of a specific app.
- What is worthy of note is that our approach supports the participation of testers with different levels of experience and different backgrounds to represent the real users performing the test, unlike other approaches that only deal with testers who mostly have the same level of experience. Furthermore, through our approach, real users can also be involved and perform the test directly on their devices. This would extensively allow discovering issues resulting from different configurations of mobile devices and the unexpected compatibility issues produced from the different behaviors and interactions of real users with the app.
- Our approach uses a totally new crowdtesting workflow that relies on direct interaction between developers and testers without the middleman figure (manager or leader) as in most current approaches. This workflow bridges the gap between developers and testers and reduces the issues of costs and delays caused by involving middlemen, and helps to perform tests more effectively.
- Unlike other approaches, we provide a mechanism for monitoring and tracking the status of the published testing tasks is implemented to organise the test and ensure that the test really covers the requirements and whether everything is going well during the test cycle.
- Unlike other approaches, we provide a service that enables developers to perform testing for all three compatibility testing types: platform, a feature of mobile device, and API configuration simultaneously, which can facilitate the testing process.
- An internal reports/issues tracking system is implemented in our approach to automatically track and collect submitted reports and help developers review and evaluate the issues that have been reported quickly. This helps testers stay informed about all the statuses taken on their submitted reports. This eliminates the time-consuming

problems and limitations of gathering and organising the submitted test reports that the manager or leader encounter manually or through external tools (e.g., JIRA) as in most of the current crowdtesting approaches and platforms.

- Another asset worth considering is that our approach provides a public knowledge sharing environment (wiki) for public use. This wiki would document compatibility testing results, including discovered compatibility issues, their causes, possible solutions, and testing scenarios. This wiki will help to perform a more effective test and provide more insight into compatibility issues and different architectures of mobile devices and provide more knowledge for generating the best testing scenarios.

5.3 Proposed Crowdtesting Approach Working Mechanism

This section provides an overview of the working mechanism of the proposed approach, including our direct-interaction crowdtesting workflow, as outlined in Figure 5.1.

1) Developers defining test tasks: The concept of the crowdtesting process is to divide large projects into a set of small tasks that crowd testers could efficiently execute. In existing crowdtesting workflows, the defining of tasks is sophisticated and takes a longer time as it requires several steps to be undertaken by the crowd manager. Our direct interaction workflow eliminated these steps by allowing app developers to define the project themselves into a set of small tasks and determine the requirements for each task at the beginning of the test cycle; instead of defining the entire project and waiting for an available crowd manager to split the project, as in most of the current approaches. All these small tasks will directly be stored and displayed on the tasks page with previously defined tasks within the platform to be visible to all testers.

2) Announcing and distributing the test to testers: Once the developers complete the defining process of the testing tasks and their requirements, they will need to distribute the testing tasks to the testers, and a notification immediately will be sent to all testers registered within the platform who match the task requirements. The developer will also be able to select or invite specific testers inside the platform or invite external testers to execute the test based on matched requirements (more details in Section 5.4) as in most of the existing crowdtesting platforms.

3) Review tasks requirements by testers: Once the testers receive the announcement of the test, as displayed in 5.1, they can review the requirements and specification of the task and decide to accept or reject performing the test. If the testers are not interested in performing the test, they can review the requirements and specifications of other public tasks and select them accordingly.

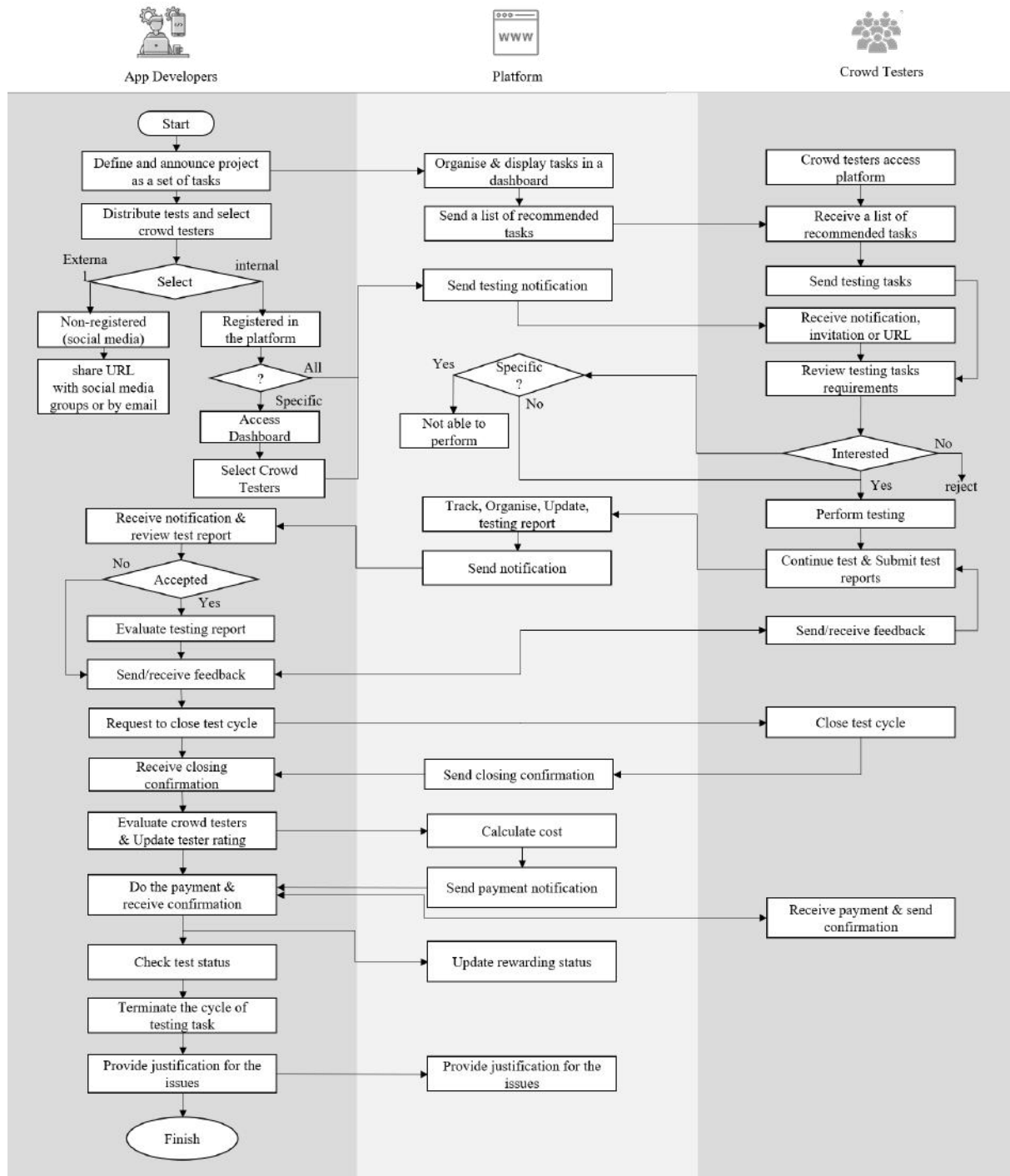


Fig. 5.1 Detailed direct-interaction workflow of proposed crowdtesting approach (DT-CT)

4) Selecting task by testers: Once testers access the platform, they can also select a suitable task from the list of recommended tasks displayed for them based on matching their properties. The testers can also choose suitable tasks from the list of public published tasks (not private) in the platform.

5) Execution of test and submission of reports by testers: Once developers accept the testers to perform the selected task, they will start executing the tasks based on the required types of mobile models and OS versions. After completing the test, testers must submit a single test report for each executed task using the submission form within our implemented platform.

6) Tracking testing reports: When testers submit tasks at different times, the platform will immediately track, collect, and organise the submitted test reports through the internal report-tracker system within the platform. Then, the platform will notify developers to review the submitted reports by themselves rather than waiting for the manager or leader to gather and organise the submitted reports manually or automatically using an external report tracker system (e.g., JIRA, Zoho, Bugzilla, etc.) as in most of the current crowdtesting approaches.

7) Review and validate test reports by developers: Once the developers receive notification to review the collected reports, they will directly review and validate every single report, including the reported issues. If the reported results are not accurate or something needs to be clarified, the developer will directly reject the report, and evaluation feedback will be sent to the tester to update the report. Once the testers update the report and send it back to the developer, the report will be rechecked; if the results are correct and the developer has approved it, a notification is sent to testers to inform them that the issues have been fixed.

8) Closure of testing cycle by testers : Once the tester receives feedback that the issue has been resolved, they will retest to check and then send a notification to the developers to inform them that the issue has been successfully fixed. Once the developers receive this notification, they will request the tester to close the test life cycle. Once the tester closes the test, our platform will immediately send a confirmation to the developer that the test cycle closed.

9) Reward testers by developers: Once the developers receive the termination confirmation notification from each tester who has finished and closed their test cycle, they have to evaluate the testers immediately and update their rating. Then, the platform will automatically calculate the reward value of each report submitted by testers based on their report quality and a set of criteria (as explained in Section 5.4). A payment notification will directly be sent to the developer with the exact price for each tester. In this case, the developer will reward the testers based on the identified reward value and notify the testers to accept the reward. This process will reduce the manager's time-consuming problem of

determining the list of testers that need to be rewarded, and then send it to the developers to complete the rewarding process.

10) Payment confirmation by testers: Once the tester receives the reward and accepts it, they will send a payment confirmation notification to the developers. Later, the platform will update the reward status of each task for each rewarded tester from a non-paid task to a paid one. This would facilitate the testing process and minimize the waiting time of testers to receive the reward, unlike other crowdtesting workflows that require testers to wait for the manager to prepare the final payment report and then send it to the developers to be rewarded.

11) Update or termination of the task by developers: In this step, the developers check the status of each task. If the task has not yet been tested on required devices or OS versions, the developer can go to the first stage, "defining and announcing task" and change the required information to clarify the devices not yet covered, while if the required devices and OS versions have been covered, then the developer will end (disable) the task so that no testers can test it later.

12) Justification of reasons of issues by developers: After the developer completes the validation of all submitted reports related to a particular task, they should justify reasons and/or solutions for all discovered issues resulting from the test of the task on different mobile devices and OS versions; if the developers find a reason or solution for the issues, they have to store it in the knowledge base. This process is not available in the existing crowdtesting workflows. Implementing this process would aid developers in understanding the compatibility issues for the particular functionality of apps with specific mobile devices and/or OS versions when developing new apps in the future.

5.4 Design and Implementation of the Proposed Approach

This section describes in detail how the processes of the proposed approach are designed and implemented to overcome the limitation of the existing crowdtesting approach listed in Chapter 2 (considered as requirements for our approach) and to meet the actual needs of the real practitioners as discussed in Chapter 4.

5.4.1 Tasks Defining and Distribution

Figure 5.2 describes how the defining and distributing process in our approach was designed to meet the developers' needs that are explained in Chapter 4, Table 4.7 and to eliminate the micro-task decomposition issue and allow developers to distribute the test on a large-scale. The implementation of this process included four linked stages that enable the entry of all the necessary information, as previously discussed in Chapter 4.

The first stage is related to the *"task information"*. As shown in Figure 5.2 (a), through this stage, the developers can define the entire app as a group of multiple small tasks by either putting them all under one project or defining them separately and then clarifying the requirements for each task. This stage includes information about (1) a detailed description of the apps; (2) the hardware components or features of the mobile device or OS versions they would like to test; (3) the ways of testing such as testing source code by providing the link of code or posting the code, or testing as a physical feature by providing the testing steps or scenarios that testers need to follow; (4) information on the complexity level of the testing task; (5) the submission deadline; (6) a list of expected results from each task. **The second stage** involves entering *"device information"* for each defined task in the previous stage, as shown in Figure 5.2 (b). At this stage, developers can specify the target mobile device models and OS versions of Android and/or iOS that need to be tested for all the tasks or each task individually. **The third stage** is related to the *"app information"*. As illustrated in Figure 5.2 (c), this stage contains the entry of the type of app and uploads the beta version and required files of the app to the platform to be able to be download and tested by testers. Developers can link the test to external resources (e.g., GitHub) by posting the link of the location of the source code. **The final stage** is related to the *"publishing of the testing tasks"*. To distribute the test and to control the task privacy before it is published, two options are proposed: to publish it as a private or public task, as illustrated in Figure 5.2 (d). If the developer chooses to define the task as a public task, this will enable any testers to access it and perform it (if the developer accepts that tester), while if they select a private task, in this case, only authorised testers selected by developers can see it. It should be noted that all new project and their relevant tasks (either private or public) will be stored on the "main tasks dashboard" with previously defined tasks visible to all testers. If the developers need to define new tasks to a particular project, they can easily add them under that project.

5.4.2 Testers and Task Selection

This section describes our design of the selection process of suitable testers for the test. After publishing the task privately or publicly, the task will immediately be added to the main task dashboard and link the developers directly to the invitation and selection page to select the testers. There are two options provided to select the suitable testers to perform the private task, either invite and share the task link privately by email to a specific tester or move to the testers selection dashboard (see Figure 5.3 (a and d)). As shown in Figure 5.3 (d), this dashboard includes different criteria represented in dynamic graphs to facilitate the selection process of suitable testers to work on specific tasks. Once the developers click on any of the criteria, a drop list will appear that includes testers' names and levels based on that criteria. The testers are displayed from higher to the lower level to facilitate the selection process. These criteria were defined and calculated as follow:

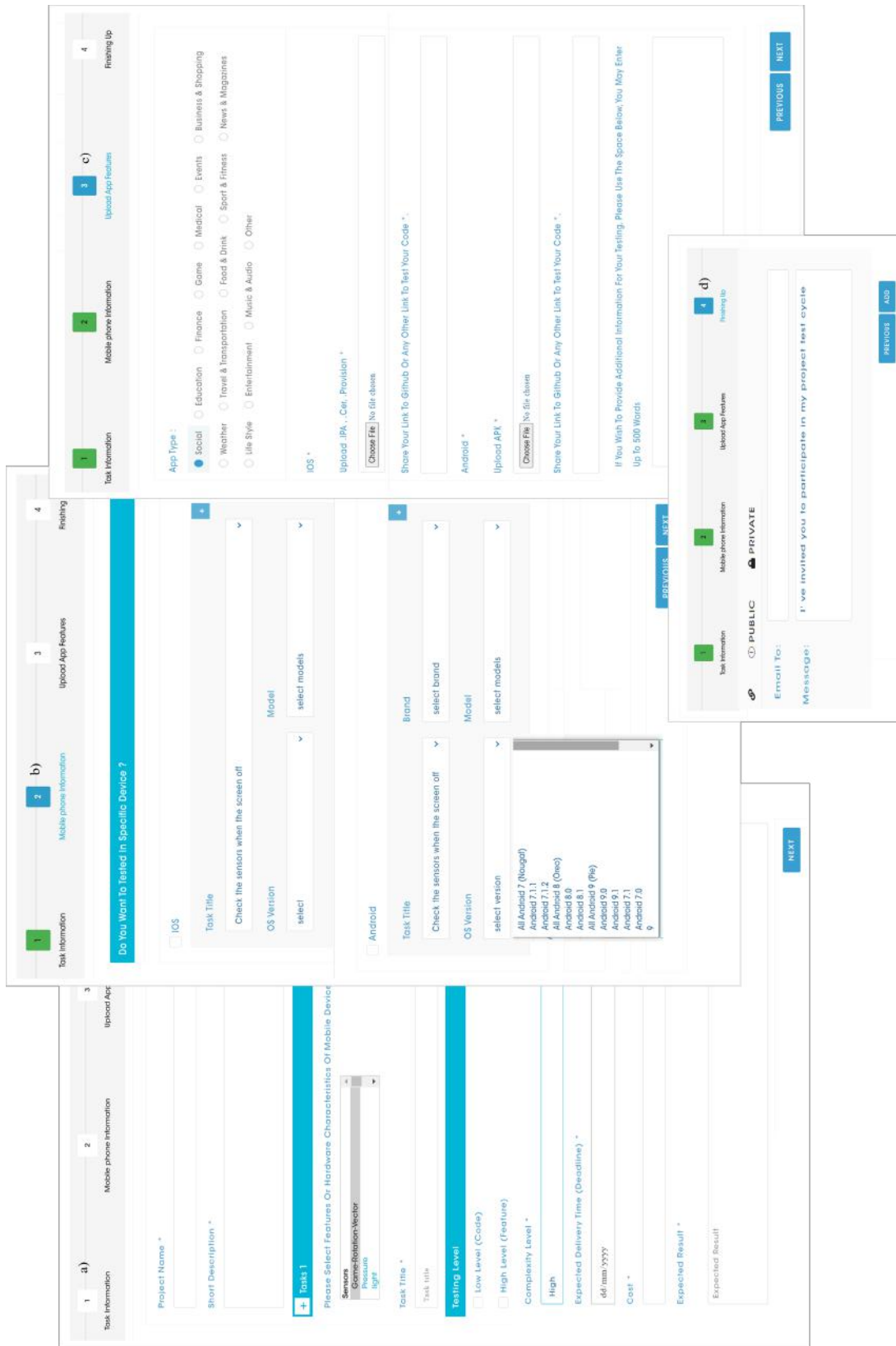


Fig. 5.2 Task defining and distributing mechanism

- **Complexity level:** It helps select testers who are suitable to perform tasks with different levels of complexity, low, medium, and high. Testers are classified as qualified for performing any tasks under these complexity levels based on the percentage of all previously performed tasks under each category through the formula presented below. If the percentage of performed tasks $> 33\%$, the testers will be added under that category.

$$CL = \left(\frac{\text{Number of performed tasks under each category}}{\text{Total number of all performed tasks}} \right) * 100 \quad (5.1)$$

- **Platform experience:** This facilitates the process of selecting process of appropriate testers to perform the tests for the Android and/or iOS mobile platform. Testers are considered suitable for performing the test on any of these two mobile platforms (MP) based on the number of tests performed under each using the following formula:

$$MP = \left(\frac{\text{Number of performed Android or iOS tasks}}{\text{Total number of all performed tasks}} \right) * 100 \quad (5.2)$$

If the proportion of the tester previous tests for Android or iOS exceeded 40%, the tester will be classified under that category. According to many works of literature, 40% is the least value in the average/adequate range in either five-rating and four-rating scale, as reported in (Dana et al., 2019; Hartati et al., 2020; Sagala and Andriani, 2019).

- **Work quality level:** It helps developers to distinguish between good and poor testers and select testers with good work quality. The levels of this feature were divided into five categories and the percentages of these levels was identified based on the percentage rating scale provided by Sagala and Andriani (2019), poor $< 20\%$, low 21% - 40%, average 41% - 60%, good 61% - 80%, very good $> 81\%$. The testers will be classified under the correct category based on the average quality percentage of all previous work.

$$Avg\ Quality = \left(\frac{\sum \text{Quality percentage of all submitted reports}}{\text{Number of submitted reports}} \right) * 100 \quad (5.3)$$

- **Completion time/delivery precision:** It helps choose testers who submit the work on-time. The testers were classified under the correct category based on the percentage of the on-time submitted tasks (OTS) for all previously performed tasks (Versa, 2021). The range of OTS percentage for each category was determined as follow: Not on time (OTS $< 34\%$), Somewhat (OTS = 34% - 67%), and On-time (OTS $> 67\%$).

$$OTS = \left(\frac{\text{Number of tasks submitted on time}}{\text{Total number of all the submitted tasks}} \right) * 100 \quad (5.4)$$



Fig. 5.3 The process of distributing the test and selecting the specialised testers

- **Reliability:** This helps developers to distinguish between reliable and unreliable testers when selecting testers to perform a specific task in a particular testing areas. Four levels were identified regarding the tester reliability, and the percentages of these levels are adapted from the percentage rating scale used by [Wulandari et al. \(2018\)](#), not reliable 25% – 43%, fairly 44% – 62%, reliable 63% – 81%, very reliable 82% – 100%. The tester is classified under the correct reliability level according to the reliability percentage resulting from their previous work in different testing areas and at different levels of task complexity.

On the other hand, two choices were implemented to publish the task publicly to all testers. First, sharing the link of the task with other testers groups on the internet to be visible to the public testers, as shown in Figure 5.3 (c). Second, adding the task to the "main task dashboard" where all testers can access and perform it, unlike the private task which only identified qualified testers can access. As shown in Figure 5.4 (a), the task dashboard lists all projects and the tasks that developers posted. This dashboard provides testers with the freedom to search and select the task they want to test for Android and/or iOS. To control the selection of the defined projects and their tasks, a validation check will be made. If the tester has selected any of the private tasks, a message will appear to them that they are not eligible to perform the task (see Figure 5.4 (b)), while if a public task is selected, then s/he would be able to review the task requirements. The task requirements page will include all information entered by developers with the beta version file or link of the app. After the review of the requirements, the tester will be able to ask to perform the task. A notification is then sent to the developers to inform them that someone has selected the task. If the developer accepts that tester to perform the test, the tester will be directly registered in the list as a new tester performing the task.

To facilitate the selection process of testers, a recommended list of eligible testers is sent to the developers. This list is determined based on four main elements; some of them had already been used by current recommendation systems in the literature ([Cui et al., 2017a,b](#); [Wang et al., 2019b](#); [Xie et al., 2017](#)). These elements are:

- **Testing Context:** This includes the device models and OS versions that testers had tested in their previous work.
- **Testers Capability:** This includes the number of tasks performed under accepted projects, task complexity level, fast completion time, and quality level.
- **Testers Experience:** This contains reliability level of testers in a specific testing area (e.g., sensing, audios, GUI, etc).
- **Testers Availability:** is related to the possibility of testers accepting new projects based on the number of active projects on their list (must be less than three projects).

Furthermore, to assist testers to choose the appropriate tasks, also a recommended list of tasks is sent to each tester based on the following three factors:

- **Task Specification:** This is related to the required mobile devices, OS versions, and hardware components that are required to be tested.
- **Task Complexity Level:** This includes the task level: difficult, medium, or simple.
- **App Type:** This is relates to previously tested apps (e.g., banking, health, education, activity tracking, etc.)

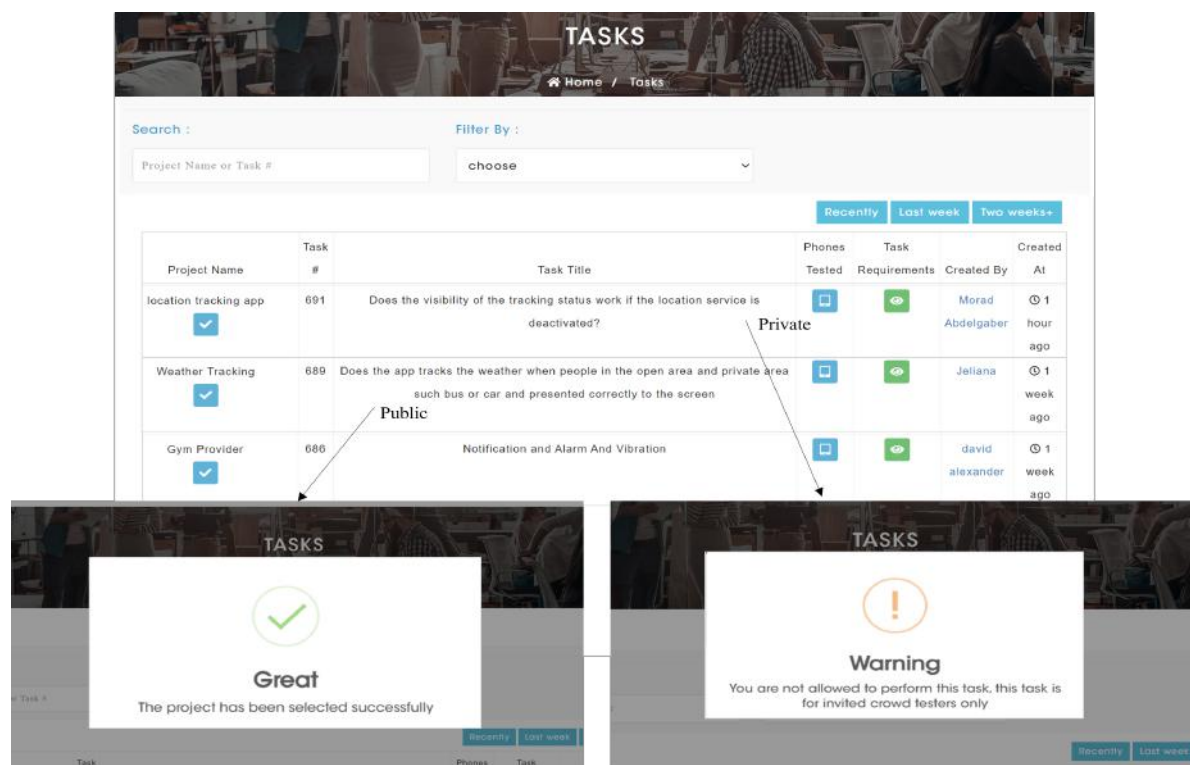


Fig. 5.4 The method of selecting tasks by testers

To control the testing environment, a notification will be sent to the developer after accepting each tester to perform their task to inform developers about the number of testers performing the task. If any update is made on any tasks, a notification will be sent to inform testers about the changes made in the accepted tasks. Testers can select and accept up to three projects to perform, including their tasks. To control the number of selected projects and overload work on the testers, a check is performed every time testers select a new task to ensure that they do not have more than three active testing projects simultaneously on their list. Moreover, to protect the testing environment from having non-active testers, a notification will be sent every three days to those who have accepted the test project and

have not performed for more than three days. Also, to control the testing environment and track published tasks' status, two features were implemented, as illustrated in Figure 5.5. The "#.of Tests Performed" describes how many times the test was performed by different testers and "Phones Tested" shows the types of mobile device and OS versions tested. These two features are shown for both, developers and testers, which helps to avoid repetition of testing the task multiple times on the same devices. The "Track Status" is another feature that was used to monitor the location of each tester who performed each task to ensure the test is covered in the target regions, as displayed in Figure 5.5.

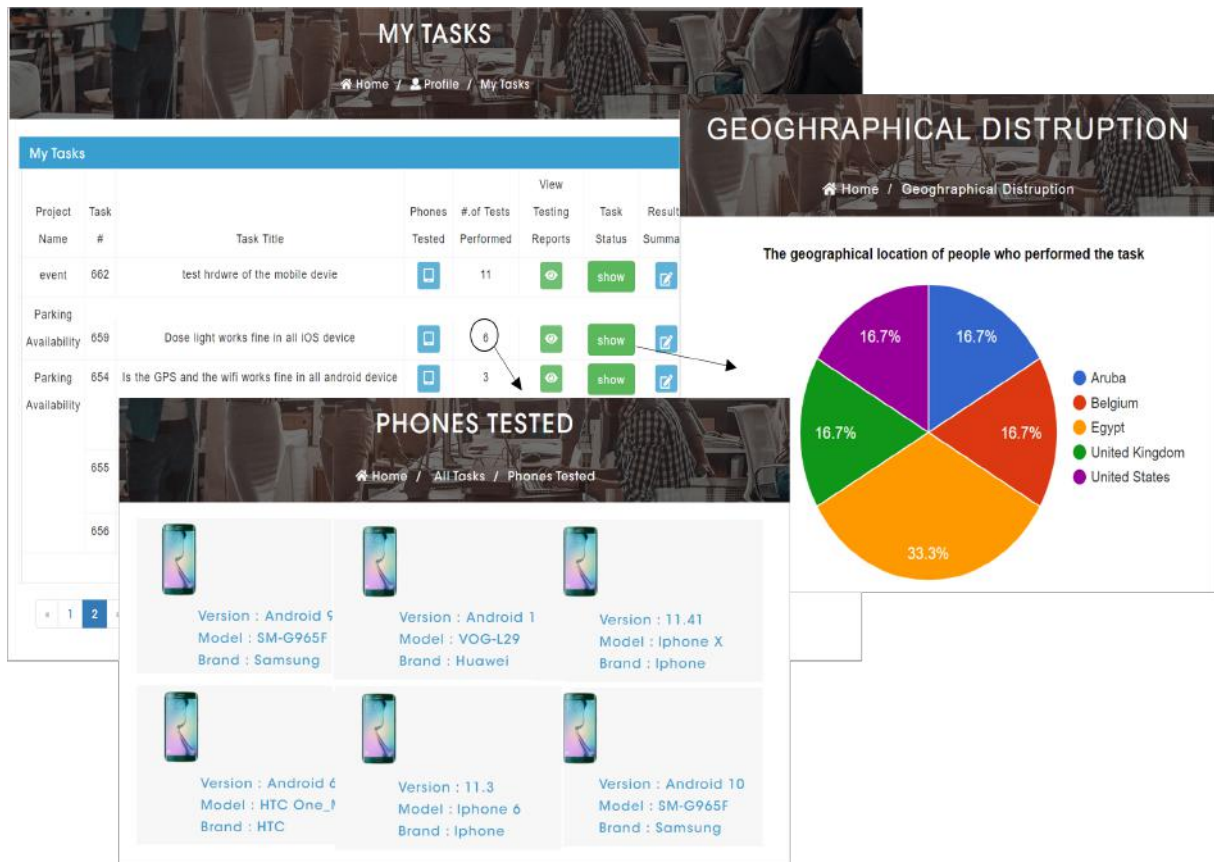


Fig. 5.5 The methods of tracking published tasks' status

5.4.3 Results Submission

Figure 5.6 explains how the results reporting process was designed in our approach to meet the testers' needs as discussed in Chapter 4, Table 4.7. As mentioned previously, each project can include more than one task. If testers accept the project, all the tasks under this project will be immediately added to their file. The testers have to perform each task separately and submit the result of each in a separate report. Similar to the defining and distributing

process of the testing project, the report submission process also involved four sequential stages that support the entering of the required testing information correctly, which was previously outlined in Chapter 4.

The first stage is related to the "*mobile information*". In this stage, the testers need to enter the details of tested mobile devices. Two options are implemented to provide accurate inputting of mobile phone details and avoid errors that may occur when the users enter the phone details manually in the submission form (see Figure 5.6 (a)). **Firstly**, If the mobile device used in the test is not registered; in other words, if the tester used the mobile for the first time, then they need to access the following link "<http://askcrowd2test.com/mobile/detect>" from the new mobile device. Once this link is accessed, the mobile device data are directly detected and displayed on the mobile screen. The detection is implemented through the use of two detection features of the JavaScript: the "User-Agent Sniffing" and "Screen Dimensions" of the mobile device (screen.width and screen.height) (Andrew, 2017). Using these two features together would ensure the correct detection for the model name and model number of the mobile devices that have the same screen size. After displaying the detected details on the screen, the tester needs to click register to upload the mobile data to the submission form which is filled in automatically in the correct fields during the reporting process. The device will directly be registered in the list of the tester for later use. Figure 5.7 describes all these processes of detection in detail. **Secondly**, if the mobile device use in the new testing task is already tested and registered before, the testers will need to select the register option and then choose from the list the mobile device and the other details to move to the next stage, as displayed in Figure 5.6 (a).

The second stage includes "*task information*", which shows the list of the expected results of the task and asks testers to clarify by providing the actual result (if each expected result has successfully been achieved or failed), as shown in Figure 5.6 (b). If some issues have been discovered, then testers need to enter the details of the issues in the next step. Note, our submission process requires testers to submit the discovered issues separately, each issue in an individual report.

The third stage is related to the "*issue information*". As illustrated in Figure 5.6 (c), this stage contains detailed information of the discovered issue, including the issue title, priority, a detailed description, screenshots, and the performed steps that led to this issue.

The final stage is related to the "*additional information*". In this stage, the testers need to clarify how sure they are about the results and provide any further information, suggestions, or ideas related to the improvement or solving the issue, as illustrated in Figure 5.6 (d).

a) Mobile Information

Mobile Details *
To Get Your Mobile Device Details Please Select One Of The Following Two Options

Not Registered Registered

To Detect Your Mobile Device Type Please Open This URL From Your Mobile Device That Used In Testing Process.
http://askcrowd2test.lk/mobile/decrypt

Not Registered Registered

Mobile Platform *
Select Platform

Mobile Brand *

Mobile Model *

Mobile OS Version *

2 Task Information

Issue Title

Issue Priority
low

Issue Description

Steps To Update The Issue

3 Bug Information

Screenshot Or Additional Files
Choose File No file chosen

4 Additional Information

Mobile Information

Task Information

Bug Information

Additional Information

How Confident You Are With These Results? *
 Unsatisfied Neutral Satisfied Very Satisfied

Do You Have Any Additional Information Or Suggestions You Wish To Share?

Expected Results *
bbhambmb

Actual Results *
 Success Fail

1 Mobile Information

2 Task Information

3 Bug Information

4 Additional Information

PREVIOUS NEXT SUBMIT

Fig. 5.6 Results reporting mechanism

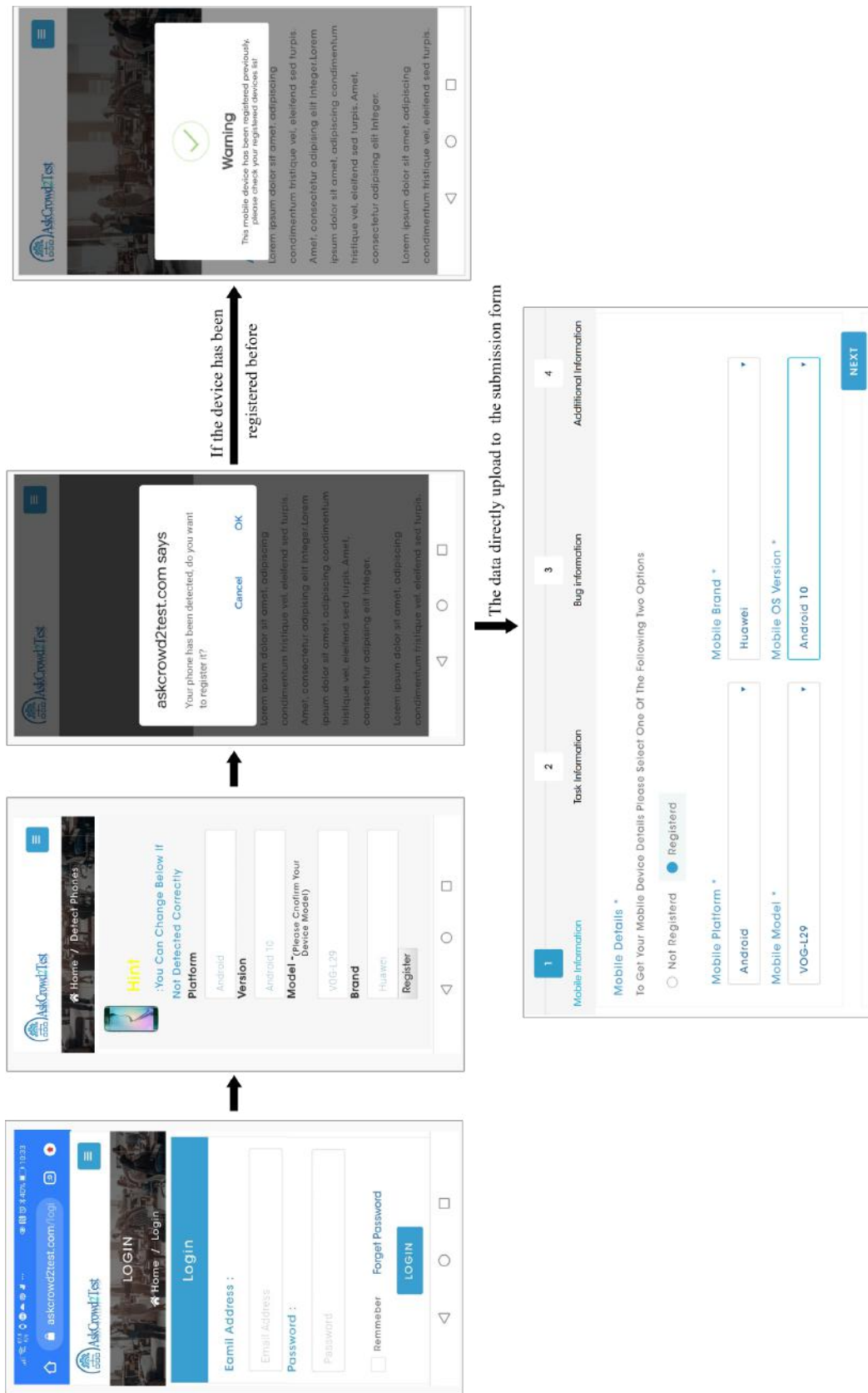


Fig. 5.7 Mobile device data detection method

5.4.4 Report Tracking and Evaluation

This section describes the design and implementation of our reporting mechanism and reports tracking system to demonstrate our idea and cover the requirements presented in the previous chapter. If different reports for different projects are submitted by testers at different times, our reports tracking system will automatically track, aggregate, and organise these reports according to the latest received report under their relevant tasks. Every time a new report is received, the tracking system will notify developers that a new report is ready for assessment. As shown in Figure 5.8, two options have been provided to help developers review and evaluate the reports quickly, based on sensitivity/priority of the reported issues and/or the report's evaluation status.

The figure illustrates two user interfaces for managing test reports. The top interface, labeled 'Developer', shows a task titled '#680 Check the sensors when the screen off'. It includes a table with columns: Submission #, Tester, Reported Date, Report Details, Tester Action, Issues Priority, and Reward. A dropdown menu for 'Priority Status' is open, showing options: Low, Medium, High. The bottom interface, labeled 'Tester', shows a task titled 'Re/Submitted Reports Of The Task'. It includes a table with columns: Submission #, Status, Report Submission Status, and Developer Feedback. A dropdown menu for 'Status' is open, showing options: All, New, Opened, Rejected, Defired, Duplicated, Resubmitted, Resolved, Finished, Closed. Both panels include 'Apply Filter By' sections with 'Issue Status' and 'Priority Status' dropdowns.

Fig. 5.8 The two ways of filtering and organising test reports

Figure 5.9 describes the complete process of tracking the report's status and how developers and testers are informed about the procedures that are taken by each. Once the tester submits the test report, the status will directly be changed to "submitted", and the status will be displayed for the developer "a new" submission. The developer then opens the report to fix the issue and take any of the following actions:

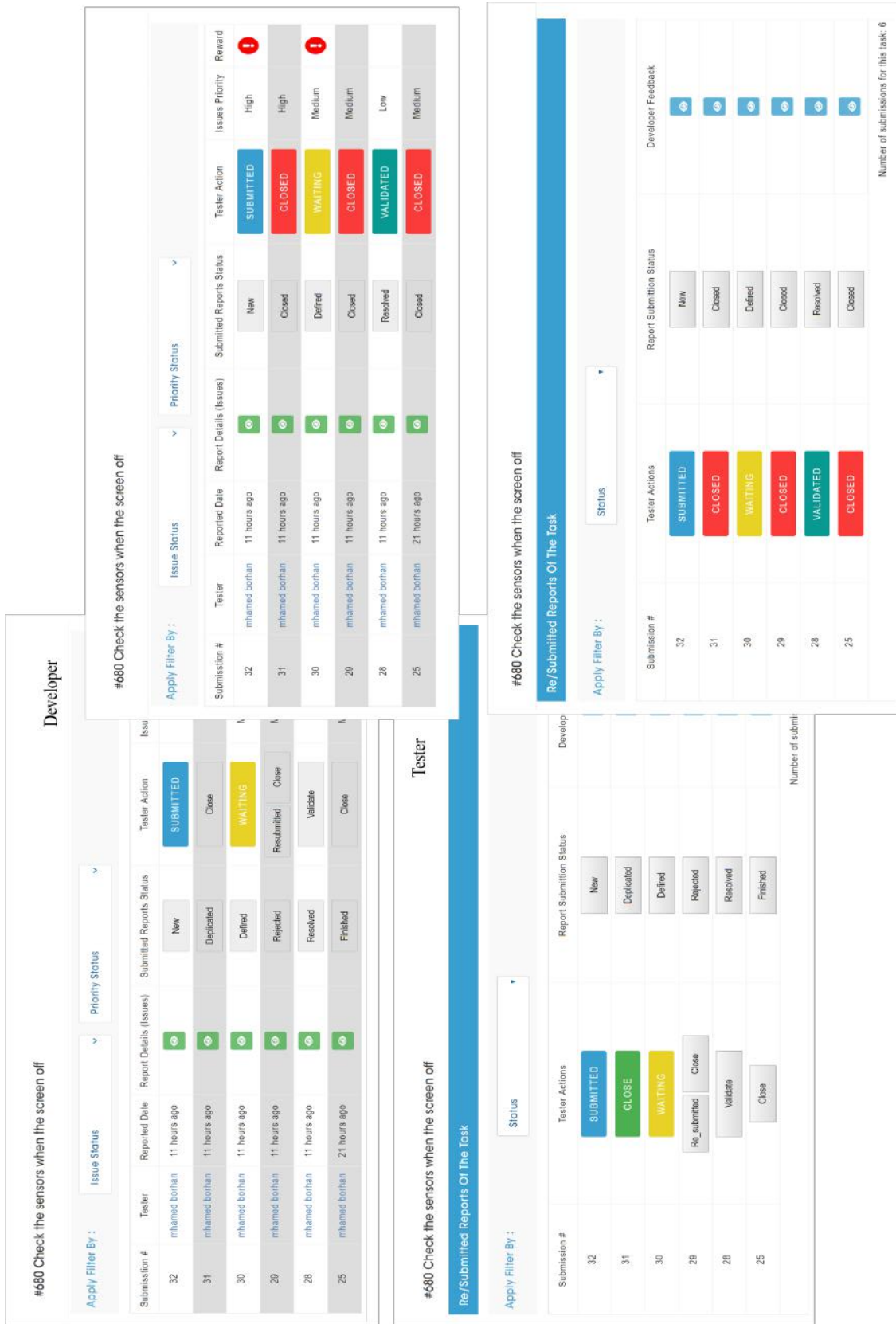


Fig. 5.9 The tracking method of reports/issues status from the developer and tester side

- **Deferred:** The developers may postpone the process of a specific issue because it does not have a higher priority and they are busy fixing another issue. The status will appear for the developer as "deferred" and for testers as "waiting".
- **Duplicate:** If the developer found the report or issue is repeated, the status will be "duplicated", the status will be changed for the tester to "close", this means that the tester needs to end the report life cycle.
- **Rejected:** If the developer does not accept the report for any reason (e.g., found the issue is not a genuine issue), the developer will change the status to "rejected". In this case, two options will be shown for the tester either fix the problem in the report and "resubmit" or just "close" it.
- **Fixed:** Once the developer checks and fixes the issue, they will send the report back to the tester to check whether the issue has been fully resolved or not. The status will appear for the developer as "solved" and for testers as "validate".
- **Validated/Retested:** After the tester "validate" the report and inform the developers that the issue has indeed been fixed and no new issues are found, the status is changed to validated to both sides. Then, the developer will be required to reward the tester.
- **Finished/closed:** If the issue no longer exists, and the tester received the reward, the status from the developer will change to "finished" and the tester is asked to close the test cycle. Once the tester closes the test, the status will appear for both developer and tester as "closed".

Figure 5.10 shows the different criteria used to evaluate report quality and testers' work. These criteria are:

- Report quality (RQ) (Correctness and completeness of results) totals 50%. Since it has 5 levels (5 max - 1 min) and according to the formula below, the percentage will be divided as follows: Very good 50%, good 38%, average 25%, low 12%, and poor 0%.

$$RQ = \left(\frac{\text{Obtained value} - \text{minimum value}}{\text{maximum value} - \text{minimum value}} \right) \quad (5.5)$$

- Task complexity level (TCL) 15%; the difficult task 15%, medium difficulty 10%, and simple task 5%.
- The number of rejection times (RT) with 10%. The tester will have two chances, 5% will be deducted for each rejection. The number of refusal times will be displayed in the report, as shown in Figure 5.10.

- Provision of additional information (ADD) represents 5%. Since it has 3 levels (3 max - 1 min), the percentage is divided based on the same formula 5.5 to the following percentages, if the information provided is very important, the tester will gain the full amount, fairly important 2.5%, not important the tester will receive nothing.
- On-time submission (ONS) represents 20%, as it has four levels (4 max and 1 min) and based on the formula 5.5, the percentage is divided as follows: if the report is submitted on time, the tester will obtain 20%, late up to three days 15%, up to a week 10%, more than a week tester will receive nothing.

The overall work quality (OWQ) for each tester is calculated based on the percentage obtained by all these criteria using the following formula:

$$OWQ = \sum[RQ + TCL + RT + ADD + ONS] \quad (5.6)$$

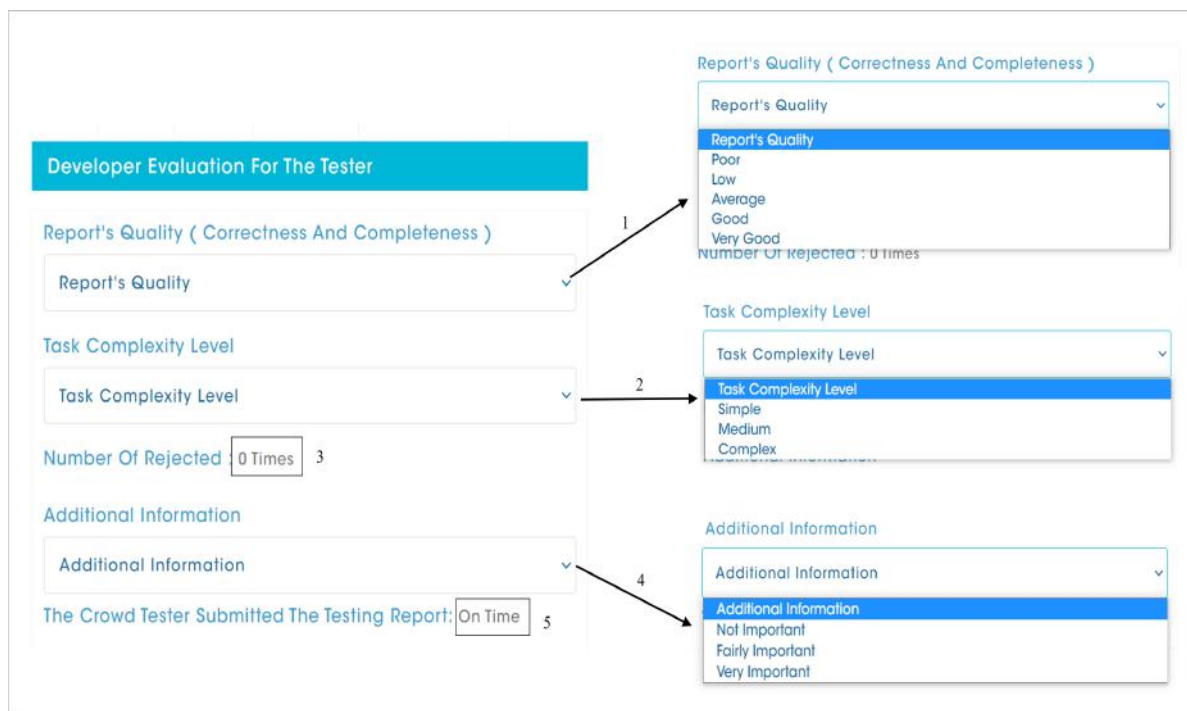


Fig. 5.10 Report quality evaluation criteria

To help developers to understand the quality level of the submitted results and the reason for low quality in the results of a specific task, a dashboard has been created that shows a dynamic graph about the work quality distribution of all testers for each task. Also, an activities summary dashboard was built for each tester to improve the work quality of testers, as shown in Figure 5.11. This dashboard presents information about work quality for each of the performed tasks, types of apps they tested, the complexity level of performed tasks,

and the consistency of work quality between completed tasks (if the tester performs the test always with similar performance quality or there is a difference in the work quality between all performed tasks). The consistency is measured through the calculation of error rate using the following formula:

$$Consistency = \left(\frac{\sum_{n=1}^n |Quality\ of\ specific\ task - Average\ quality\ of\ all\ tasks|}{Total\ number\ of\ all\ performed\ tasks} \right) \quad (5.7)$$

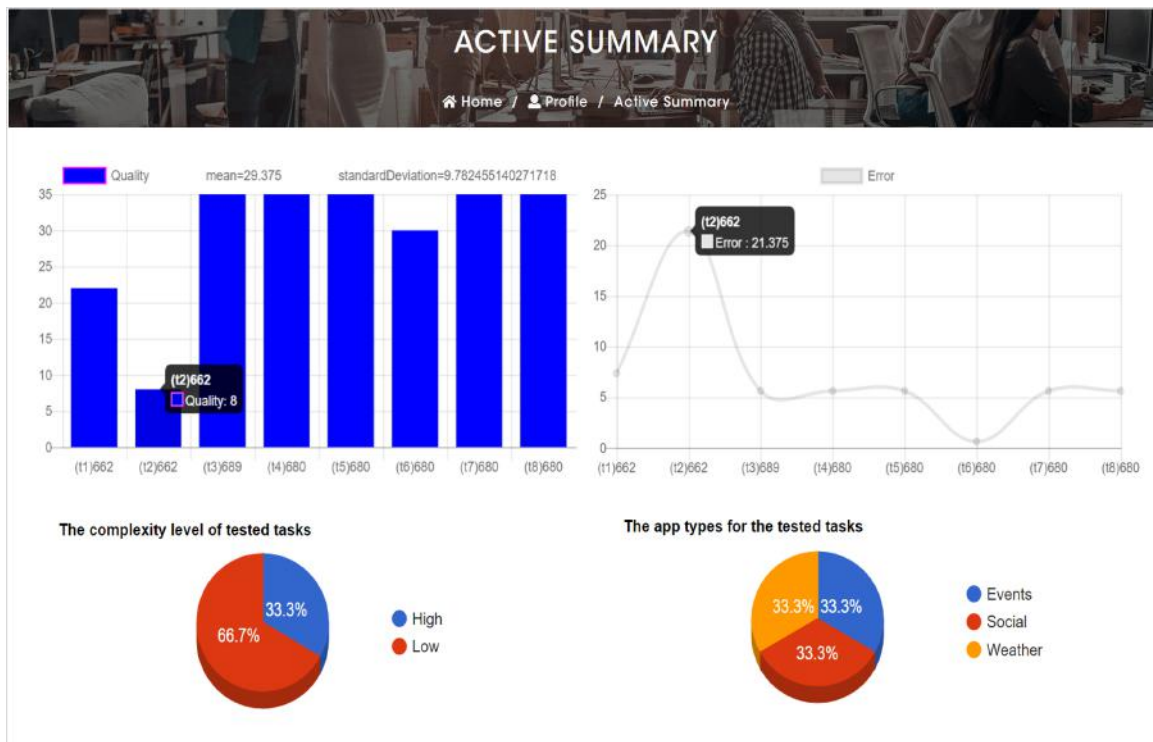


Fig. 5.11 Activities summary dashboard for the tester

5.4.5 Tester Motivation/Reward

This section describes how the rewarding mechanism has been designed in our approach. As shown in Figure 5.12 (a), the rewarding process commences after the tester has validated the resolved issue and report that it is successfully fixed. Meanwhile, the developers will reward testers based on the summary that appeared for each individual tester which shows the reward value and quality level obtained, as shown in Figure 5.12 (b). It should be noted that testers will be rewarded for each tested device. The overall work quality percentage for each evaluated report of the tester will help identify the work quality level and the fair reward value from the total payment cost for each task. The range of each quality level is assigned according to the percentage rating scale provided by [Sagala and Andriani \(2019\)](#). If

the work quality percentage is poor $\leq 20\%$, the testers will not get paid; if it is low 21% - 40%, they will obtain 25% of the total price; if it is average 41% - 60%, they will receive 50%; if it is good 61% - 80%, they will gain 75%; and if it is very good $\geq 81\%$, they will receive the full amount. Once the developer evaluates the tester, a notification will immediately be sent to the tester with detailed information about the reward value and their work quality level for the submitted report. Once the developer evaluates the tester, a notification will directly be sent to the tester with detailed information about the reward value and their work quality level for the submitted report, as shown in Figure 5.12 (c).

To make the testers aware of the amount that they may be paid for performing the tasks before they start executing the tests, the different work quality levels and the potential reward value allocated to each level are defined from the beginning within the requirement page, as illustrated in Figure 5.13. To control the motivation mechanism and ensure that all testers are rewarded, two methods have been implemented for developers to track payment status. First, through the reward state related to each task, which shows the list of non-paid testers for that task, as shown in Figure 5.14 (a). Second, through a notification that is sent to the developer, providing a list of all non-paid testers for the different tasks, as shown in Figure 5.14 (b).

5.4.6 Repository/Wiki

This section explains the design and implementation of our testing wiki. Our wiki was designed in a simple way that is slightly similar to the Stack Overflow forum in the GUI, but the documentation process of knowledge is different. Our documentation aims to provide valuable testing information that could be provided as online docs rather than just one-off answers around specific topics as in Stack Overflow and most of the current knowledge-sharing environments. The difference between these sites and our wiki is that these sites share the questions or the problems and short answers or solutions for that question only or linked the answers to other offsite links that are sometimes hard to find (Overflow, 2018). In comparison, our wiki will have more important information, examples, and guidance for solving the testing problems allocated and organised in one place under relevant topics.

To build our wiki, we have documented and categorized the testing task information automatically under specific categories based on the components of the mobile device (e.g., camera, GPS, sensors, etc.) that are related to the task that defined by the developers to be tested (see Figure 5.15 (a)). Within each category, for example, sensor, a set of related topics is listed based on the title of the task entered by developers when they announce the test requirements. Each topic includes several examples and essential information, including test scenarios and steps, discovered issues and the related mobile model and OS versions, causes of the issues, possible solutions, and other guidelines (see Figure 5.15 (b)). Accordingly, our

#689 Does the app tracks the weather when people in the open area and private area such bus or car and presented correctly to the screen

Apply Filter By : Issue Status Priority Status a)

Submission #	Tester	Reported Date	Report Details (Issues)	Submitted Reports Status	Tester Action	Issues Priority	Reward
27	mhamed borhan	3 days ago		Resolved	VALIDATED	Medium	\$
26	mhamed borhan	3 days ago		Closed	CLOSED	Medium	

REWARD
Home / All Submitted Tasks

Reward [Developer]

To: mhamed borhan b)

Your Work Quality Percentage : 35 %

Tester Quality Level: Low Quality

According To Your Quality Percentage And Quality Level: You Will Obtain 25% Of The Full Cost

Full Task Cost: 10 \$

Cost Payable (Reward Value): 2.5 \$

Account Number : 107010000000

Account Name : Grande

Bank Name : HSBC

Bank Address : Iskndri Branch

Swift Code : 971270y2

Additional Information :

Reference Number

Reference Number

CONFIRM

Reward Notification c)

Task #	Task Title	Created By	Full Payment	Quality level	Payment %	Reward Value	Reference Number	Actions
662	test hrdwre of the mobile devie	X-Dev	4\$	Low Quality	25%	1\$	6t1vcrrr	
689	Does the app tracks the weather when people in the open area and private area such bus or car and presented correctly to the screen	Jeliana	10\$	Low Quality	25%	2.5\$	AS34FT5	

Fig. 5.12 The rewarding process from the developer and tester side

Task Requirements

Project Name (App Name) : Weather Tracking

Project Short Description:
This app tracks the weather when people in the open area and private area such bus or car

Task Title : Does the app tracks the weather when people in the open area and private area such bus or car and presented correctly to the screen

App Type: Weather

Task Category : Temperature gps Humidity

Testing Level:
Low Level (Code)
weather.tracking.app/lkcs.com

Task Complexity Level: Medium

Expected Delivery Time (Deadline): 2021-06-10

Expected Results:
expected1: expected1: the weather must show the weather when the person in a public area with a blue color
expected2: expected2: the weather must show the weather when the person in a private area with a red color
expected3: The weather needs to be the same in private and public

Link of the project or code you required to test:


Link of the project or code you required to test: weather.tracking.app/lkcs.com

Additional Information:
If there is any solution or idea please feel free to provide

Cost: 10 \$

If your report quality:
Poor you will obtain: 0 \$
Low you will obtain: 2.5 \$
In average you will obtain: 5 \$
High you will obtain: 7.5 \$
Very high you will obtain: 10 \$

Please download app's files



Cost: 10 \$
If your report quality:
Poor you will obtain: 0 \$
Low you will obtain: 2.5 \$
In average you will obtain: 5 \$
High you will obtain: 7.5 \$
Very high you will obtain: 10 \$

Fig. 5.13 Display of potential reward value based on quality level within task requirement

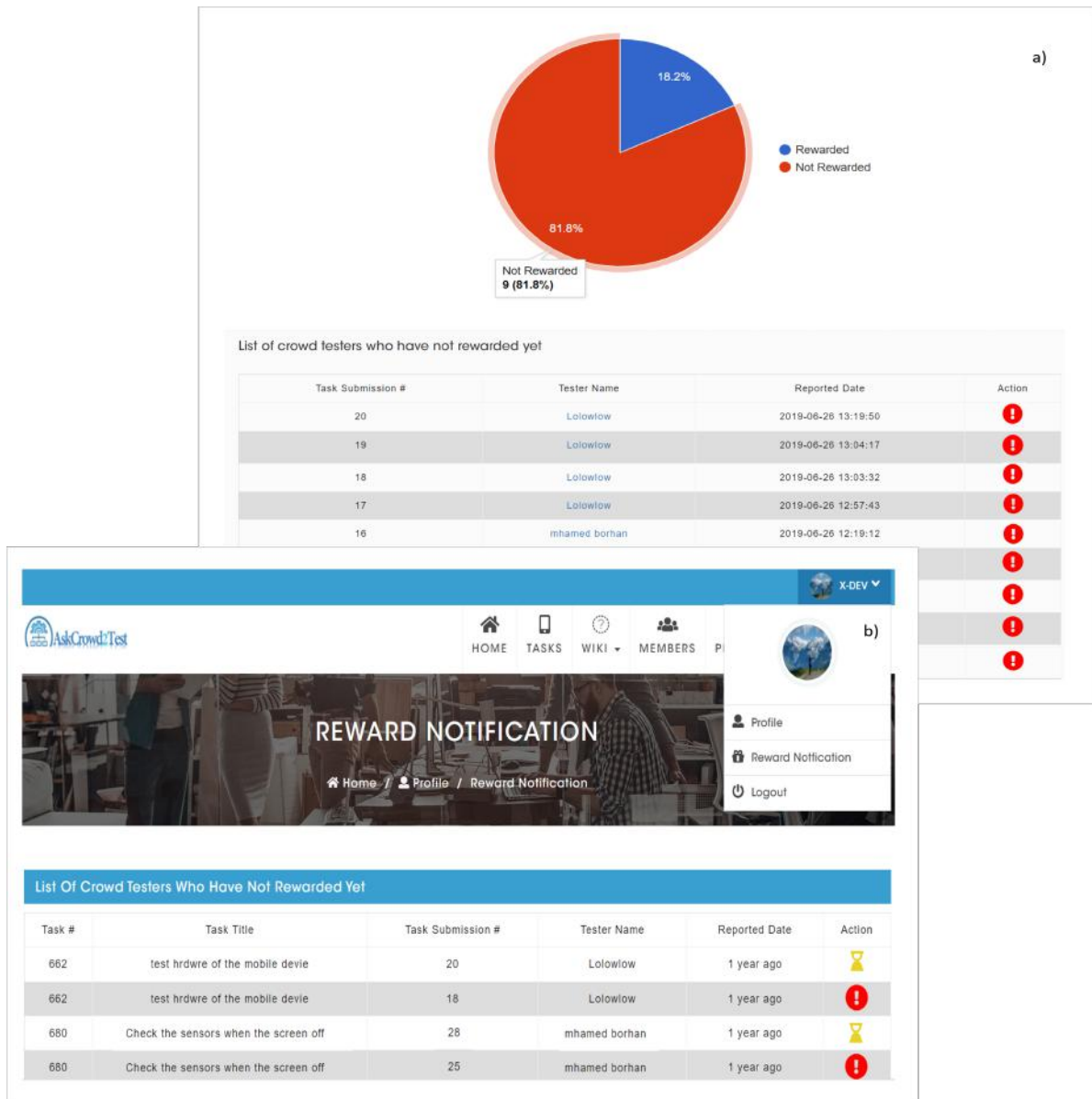


Fig. 5.14 The two implemented methods for tracking non-paid testers

wiki makes it easier to find the issues and their solutions in one place. This information is stored according to the received confirmation from the developer. Once the developer receives confirmation from the tester that the reported issue has been fully solved, the developer will be required to agree to publish the task’s information (testing scenarios, steps, issues, type of mobile devices, and so on); this will ensure that the best practice testing scenarios and accurate testing results are documented. Then, to complete the documentation process, the developers will be asked to provide the solution for that issue and some guidelines to help other developers solve similar issues or program similar app functionalities. Moreover, an

automated tagging system was implemented to help better categorisation of the compatibility issues and linking them with the correct tag. The tags are automatically identified based on the information entered when defining the test task and its results:

- **Task information:** The extracted tag is the mobile device components that need to be tested (sensors, screen, GPS, etc.).
- **Testing results:** The extracted tags are the type of issue, mobile platform, the model name/number of the mobile device that has the issue, and the OS version.

The figure illustrates a wiki interface for documenting mobile device compatibility testing knowledge. It is divided into two main sections: (a) and (b).

Section (a): A sidebar menu with icons representing different mobile device components and features:

- Topics related to mobile sensors
- Topics related to mobile screen
- Topics related to mobile GPS and location services
- Topics related to mobile device Wi-Fi, Bluetooth connectivity features
- Topics related to the mobile device camera
- Topics related to the mobile device audio, text language

Section (b): A detailed view of a task page.

- Topic (Task title):** "Does the app tracks the weather when people in the open area and private area such bus or car and presented correctly to the screen?"
- Number of test:** 2
- Task description:** "I'm developing an app for Weather purposes, I have tested the code itself ([weather.tracking.app/fkcs.com](#)) for the Temperature gps Humidity of the mobile device. For that, I need to check whether or not the Temperature , gps , Humidity working well on all Android mobile devices.
 - Expected Result:** expected1: the weather must show the weather when the person in a public area with a blue color
 - expected2: the weather must show the weather when the person in a private area with a red color
 - Additional Info or suggestions:** If there is any solution or idea please feel free to provide
- List of the examples:**
 - Example 1:**
 - Reporter:** mhamed borhan
 - Information of tested device:** Platform: Android, Mobile Brand/Manufacturer: Samsung, Mobile Device Model: SM-G965F, Operating System Version: Android 9
 - Information about the obtained issue:** Issue Title : (Does the app tracks the weather when people in the open area and private area such bus or car and presented correctly to the screen) the weather must show the weather when the person in a public area with a blue color : Success the weather must show the weather when the person in a private area with a red color : Success steps : (Does the app tracks the weather when people in the open area and private area such bus or car and presented correctly to the screen) Description: (Does the app tracks the weather when people in the open area and private area such bus or car and presented correctly to the screen) File : 831186397.jpg
 - Resolution:** Issue resolution (marked as resolved)
 - Guidelines & recommendations:** A section for providing additional information and a "Write a comment reply" field.

Fig. 5.15 The implemented wiki for documenting the mobile device compatibility testing knowledge

Each performed task and its results, including discovered issues, causes, and possible solutions, will be tagged based on these extracted tags (see Figure 5.16 (b)). This ensures that each

discovered issue is correctly stored under the correct mobile devices and components ensuring that the given tags can provide accurate search results. We adopted the tags mentioned above as important keywords relevant to the testing task and its results to implement our searching mechanism and assist developers when searching for specific issues so that our search engine would return the relevant topics (e.g., test issues or solutions), as shown in Figure 5.16 (a).

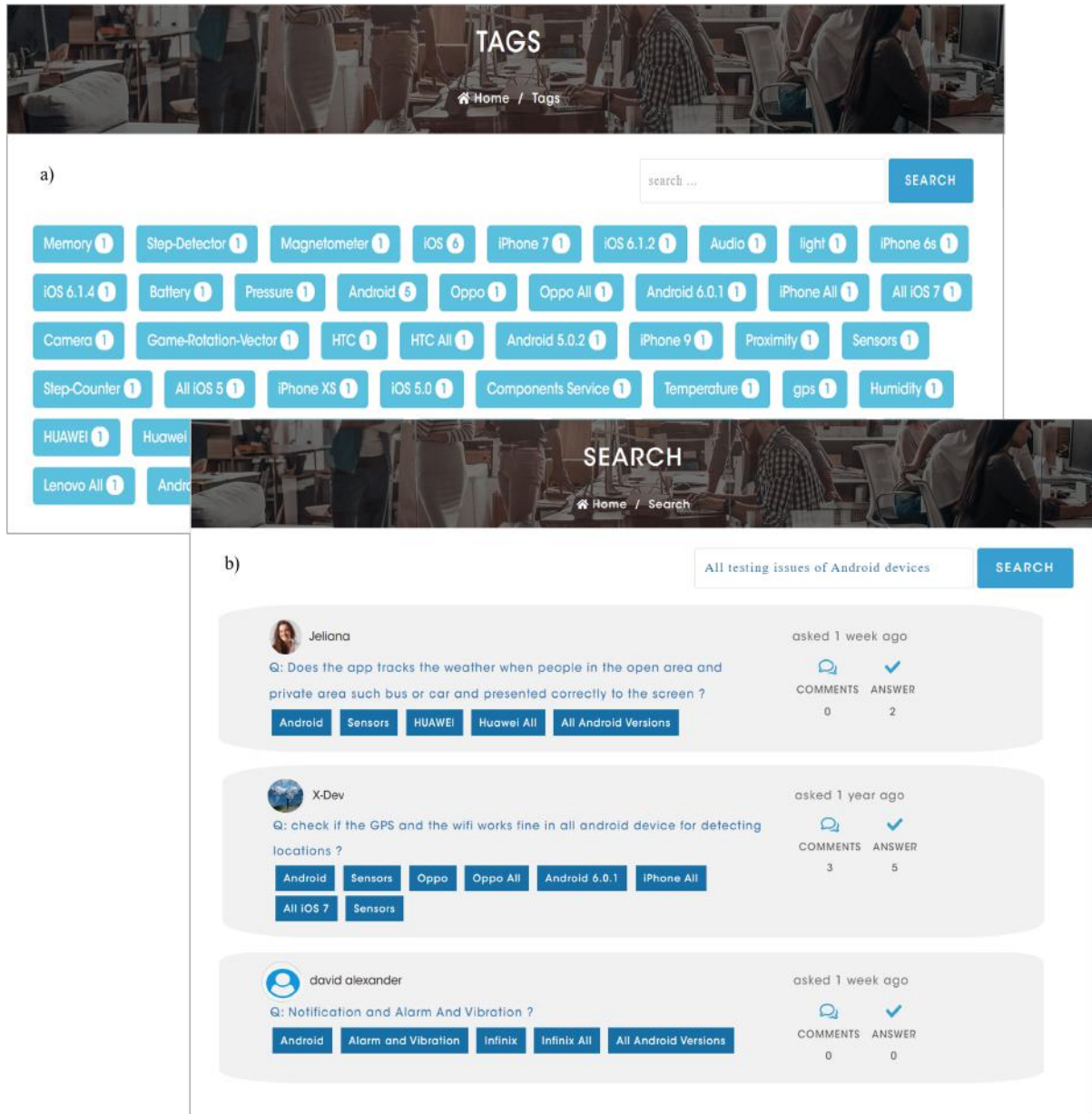


Fig. 5.16 The auto-tagging system and searching mechanism of the implemented wiki

5.5 Chapter Summary

This chapter provided an overview of the development of the proposed crowdsourced compatibility testing approach. It described the overall working mechanism through the use of the proposed direct interaction workflow. Specifically, it demonstrated the working mechanism of the whole processes of the approach and how they relate to the different involved roles (developers and testers). The chapter described how the defining and distributing process of the test task was designed to fulfill the needs of the developers that are unmet by current approaches. It also explained how the implemented features of task defining and distribution would help eliminate the task decomposition issue and assist developers in distributing the test on a large-scale. Also, it described how the testing environment would be controlled. In addition, there is a discussion of how the testers' selection process is implemented, explaining the method for distributing the task to the public testers and how the suitable testers will be chosen from the selection's dashboard to work on specific tasks. It also described how the testers are classified under these selection criteria. It also described how the main task dashboard is implemented in a way that aids the public testers in selecting the appropriate task from the list of public tasks.

There is also a discussion of the main criteria used for the testers' and tasks' recommendation system to facilitate the selection process. An explanation was provided in respect of how the results reporting process was designed to meet the testers' needs. In particular, the features implemented to facilitate the reporting mechanism and support the entering of the required testing information correctly into the submission form were discussed. Furthermore, the chapter provided a description of the design and implementation of the reports aggregation and overall evaluation process. More specifically, it showed how our reports tracking system would automatically aggregate and organise the different reports submitted by testers in an effective way that developers can quickly review and evaluate. The complete process of the report quality evaluation and the schema used for evaluating each tester was also discussed here. A description was provided regarding how the rewarding mechanism was designed in our approach and how the fair reward value was specified based on the overall work quality percentage of the evaluated report. It also explained how the motivation and rewarding mechanism is controlled to ensure that all testers were rewarded. The chapter concluded with a clear description of the design and implementation of our wiki, including the method of storing the testing information in the wiki; the implemented tagging system to categorise the compatibility issues, their causes, and solutions; and the implemented searching mechanism that help developers to search for the relevant test issues or solutions which they are looking for.

Part III

Research Phase 3: Empirical Assessment of the Developed Crowdtesting Approach

6

Empirical Evaluation Studies: Data Collection, & Data Analysis Methods

6.1 Introduction

The previous chapter outlined how the proposed crowdtesting approach was developed, taking into account the requirements presented in Chapter 4. In particular, the overall features and functionalities of the approach were discussed, as well as how it was implemented. In respect of the development of the approach, we believe it was important to assess its effectiveness in terms of performing compatibility testing on a large scale. It was also key to look at the benefits that arise from using the approach, as well as the level of satisfaction of both the developers and testers.

This chapter aims to provide details regarding the design of our empirical studies, which are used to evaluate our approach. The chapter begins with an explanation of the development process of the Crowdsourced Testing Experience Questionnaire (CSTE-Q), which is used as a tool for collecting evaluation data from the target population. Next, a detailed discussion of the methods used to assess the CSTE-Q is provided, including an overview of the iterative refinement process, and validity and reliability tests. This is followed by a discussion regarding the method employed to distribute both surveys to collect the data, sampling method, and both study procedures. Finally, the chapter elucidates the analysis methods that are applied to the data gathered from both studies, the data coding and entry to SPSS, and the statistical tests used.

6.2 Data Collection Method

Since the main dimensions that we wanted to measure in our approach, such as satisfaction, benefits (advantages), and effectiveness, rely on the developers' and testers' feelings, beliefs, and perceptions, the best way to measure that is through a questionnaire. Based on the literature, to date, no questionnaire had been designed specifically for evaluating full crowdtesting practice specifically or crowdsourcing in general. Therefore, in this research, the CSTE-Q questionnaire is designed as a tool to collect data from developers and testers to evaluate our crowdtesting approach and address our research questions (RQ2, RQ3, and RQ4). This questionnaire and its questions can be adopted for other crowdsourcing contexts.

6.2.1 Measurement Variables

The measurement variables of our evaluation are represented in three dimensions (dependent variables) and their sub-dimensions (independent variables) to measure our crowdtesting testing approach experience. For the evaluation of these variables, we adopted the 5- Point Likert Scale as the measurement method as in many prior studies (Leicht et al., 2016c; Mok et al., 2015; Wang and Wang, 2019; Xu et al., 2016; Ye and Kankanhalli, 2015, 2017). All our variables (dimensions and sub-dimensions) were measured using multiple item scales (Gardner et al., 1998), where each item is a single question that needs to be evaluated. These questions were adapted from previous studies and then revised with minor wording changes to make them more relevant to the context of crowdtesting and mobile device compatibility testing. Moreover, new questions were needed to evaluate features specific to our crowdtesting approach. These questions were derived and formulated based on the requirements listed in Chapter 4, alongside some relevant literature. To collect crucial information about the participants, eight demographic questions were created. Crucial information included the participants' countries, backgrounds, education, and work experience. The main measured variables (dimensions and their sub-dimensions) are described in detail in the following sub-section:

Effectiveness:

Effectiveness is an important and key measure of success for software. Our approach's effectiveness is primarily concerned with developers' and testers' ability to successfully perform large-scale compatibility testing. According to the literature, no prior research study has empirically assessed the overall effectiveness of an entire crowdsourcing process, including its sub-processes. As a matter of fact, most of these studies simply consider a specific aspects and features of crowdsourcing to evaluate their reciprocal influence in different contexts. The overall effectiveness of our crowdtesting approach was assessed based on the measures identified as key for the construction of a successful crowdsourcing platform in

the literature (Cullina et al., 2015; Daniel et al., 2018; Ho and Ting, 2016; Sakamoto et al., 2011). These measures were considered as sub-dimensions of the overall effectiveness of our approach. Some construction scale questions were adapted from relevant previous studies, while a set of new questions was also generated.

- ***Task Defining and Distribution:*** This refers to the extent to which the define and announce mechanism of tasks is easy and effective in distributing the test quickly on a large scale. A new five-question scale was created to measure the effectiveness of task defining and distribution in our crowdtesting approach.
- ***Privacy and Protection:*** We developed a three-question scale to measure the privacy and protection level in our approach from the developers' side and a two-questions scale from the testers' side. All developed questions were derived based on the three most important aspects of privacy and protection concerns in collaborative works; protection of the testing environment, improper access, and testers' sensitive information as stated by Malhotra et al. (2004).
- ***Accessing Specialized Tester Skills (Testers Selection):*** This refers to the extent to which the testers' selection mechanism with specific criteria to perform a particular task is easy and effective through our approach. Two questions were adapted from the scale of access to specialized skills used by Ye and Kankanhalli (2015) in addition to the two newly created questions.
- ***Task Selection:*** This refers to the extent to which the testers are able to browse and select multiple testing tasks (based on specific criteria) efficiently and effectively. A two-question scale was created to measure the effectiveness of selecting and accepting tasks method.
- ***Access and View Task's Requirements:*** This measures the degree of easy access to the task's requirements, the sufficiency of the information provided, and the clarity of its presentation for the testers. All three question of this sub-dimension were adopted from the information quality scale of PSSUQ (Lewis, 1995).
- ***Tracking Task Status:*** This measures how well tracking published task status helps developers stay informed or keep up to date in test distribution and coverage of mobile devices and OS versions. We developed a new two-question scale to measure the effectiveness of the task tracking method in our approach.
- ***Results' Submission:*** This refers to how clear, easy, and effective the results submission mechanism is for testers. A new three-question scale was developed to measure the effectiveness of the results submission mechanism of our approach.

- ***Outcomes Diversity:*** This refers to the developers' expectation of the range of different results (different issues and solutions) that could be received via various testers' experience. We adapted two questions from the scale of solution diversity used by (Ye and Kankanhalli, 2015) to measure this sub-dimension.
- ***Results' Aggregation and Tracking:*** This measures how well the internal aggregation and tracking mechanism quickly and effectively collect, organise, and track testing reports. A new three-questions scale was developed from Guzman et al. (2013) to measure the effectiveness of the results' aggregation and tracking mechanism.
- ***Results' Verification and Quality Control:*** This measures the extent to which developers can evaluate the submitted reports and control quality of testers' work accurately, effectively, and in a short time, through our results' verification and quality control mechanism. This sub-dimension was measured by a new seven-question scale, including four questions for developers and three for testers.
- ***Testers Reliability and Trustworthiness:*** This focuses on measuring how well the trustworthiness and reliability control mechanism can effectively and accurately classify testers and help developers to distinguish between testers (e.g., trusted/not trusted or good/weak testers). We generated a new five-question scale to measure this sub-dimension, including three questions for developers and two for testers.
- ***Testers Motivation and Incentivisation:*** Measures how well the rewarding method and its schema help developers reward their testers quickly and effectively. At the same time, it measures the level to which the rewarding method and its schema motivates testers to participate and perform more tests. A three-question scale gathered from different reward and motivation scales in literature was adopted to measure the effectiveness of our rewarding and motivation mechanism from the testers' side (Liang et al., 2018; Liu and Liu, 2019; Ta, 2018). Moreover, a four-question scale was created to evaluate this sub-dimension from the developers' side.
- ***Cost Effectiveness (Cost Reduction):*** This focuses on measuring the extent to which developers can effectively achieve compatibility testing on all mobile devices and OS versions at lower costs than other testing approaches. All questions of this sub-dimension were adopted from the cost reduction scale used by Ye and Kankanhalli (2015).
- ***Feedback Given:*** This refers to the testers' ability to understand the quality of their work completion through the information provided by developers about their work performance. This was measured by adapting two-questions from the feedback scale used in (Martinez, 2015) to assess the importance of feedback given within the approach.

- **Results' Documentation/ knowledge Sharing (Wiki):** This refers to how easy and effective the knowledge sharing (wiki) environment and associated searching mechanism are in sharing testing results and finding issues and solutions among worldwide developers' and testers' communities. A new three-question scale was developed to measure the effectiveness of the wiki environment. It is worth noting that the same questions were used for both developers and testers.
- **Interaction between Peers and Workflow:** This measures the effectiveness of the direct interaction workflow without a middleman for both developers and testers in performing a seamless, accurate, and easy compatibility testing process in less time. A new six-question scale was created to measure the effectiveness of direct interaction workflow; three for developers, and three for testers.
- **Ease of Use:** This measures the degree to which the proposed approach is easy to use in daily work routine and industry practices. A two-question scale was used adapted from the questions used to measure the ease of use in USE¹ Questionnaire for both developers and testers.

All the adapted and new developed questions are presented in Appendix C, Table C.1 for developers and C.2 for testers.

Benefits (Advantages):

The benefits (advantages) is the second dimension used for measuring the success of our approach. It measures the benefits the developers and testers can gain when using this approach. In order to develop the relevant questions, we reviewed the crowdsourcing literature. In literature, very few prior studies have empirically focused on evaluating the benefit of using crowdsourcing. Their assessment focused on factors such as skill enhancement, enjoyment, work autonomy, helping, and opportunities provided (Baldus et al., 2015; Liang et al., 2018; Wang and Wang, 2019; Ye and Kankanhalli, 2017). Some of our questions relevant to the testers were adapted from this literature and; we modified the questions to adjust them to our research context. A set of new questions were created to measure the benefit of our approach, which includes important aspects such as large-scale collaboration enhancement, skills development, productivity improvement, knowledge sharing, and the freedom of work for both developers and testers. The new questions allowed for a thorough assessment and a deeper understanding of the benefits of our approach compared to other approaches. All questions and their sources are listed in Appendix C, Table C.1 for developers and C.2 for testers.

¹<https://garyperlman.com/quest/quest.cgi?form=USE>

Satisfaction:

The level of satisfaction had been used for a long time to express the practitioners' feelings and attitudes towards the use of new technologies. The satisfaction in this research measures the degrees of developers' and testers' satisfaction with respect to using the proposed approach and services. According to the literature, the assessment of crowdsourcing practitioners' satisfaction had been determined in various contexts by three major constructs: the satisfaction about the product/platform experience, about the services provided, and about the benefits from their use. Questions measuring the satisfaction in both developers' and testers' questionnaire were adapted from Wang and Wang (2019), where it measure these three constructs as shown in Appendix C, Table C.1 for developers and C.2 for testers.

It should be noted that all the revised and newly generated questions (for effectiveness, benefit, and satisfaction) were discussed with the main supervisor and two PhD students from Computer Science at the University of Sheffield who have experience in the domain of mobile app testing. The collected feedback was used to generate concise and precise questions directly relevant to mobile device compatibility testing.

6.2.2 Scale of Measurement

Measurement scales represent the scale type and points levels that we intend to use to measure the variables. In our evaluation method, all items/questions in the CSTE-Q were designed and formulated as *close-ended* questions in an Interval/ratio data format. All the questions were measured using a five-point Likert scale (1= Strongly disagree to 5 = Strongly agree) and used for all questions except the demographic questions, which were a mix of multiple-check boxes and *open-ended* questions requiring short textual answers. *Close-ended* questions using the Likert scale help to collect more accurate data on the agreement or disagreement level for each question in terms of the satisfaction, benefits, and effectiveness of the approach (Marsden and Wright, 2010). Only one *open-ended* question was added as a necessary additional question: "Do you have any recommendations to help us improve the proposed approach?". The question allows participants to freely express their opinions, suggestions and unmet needs. All the questions were ordered in a positive manner (where 1 = "Strongly disagree", 5 = "Strongly agree"), only a few questions were formulated in a negative manner with the reverse order (where 1 = "Strongly agree", 5 = "Strongly disagree") for responses validation purpose. The aim of this was to force developers and testers to evaluate each question in its own right. The negative words were written in capital letters (e.g., NOT, DOES NOT, and DO NOT) to emphasize the actual meaning of the question to respondents. This is because when all questions are formulated in the same direction, responders seem to evaluate them equally (Rattray and Jones, 2007). These negative questions would further help determine the accuracy and reliability of answers given, thus detecting any respondents providing random answers.

6.2.3 Questionnaire Surveys Structure

The CSTE-Q questionnaire was then split into two separate questionnaire versions: the CSTE-Q/D and CSTE-Q/T in order to focus on the specific needs of developers and testers. Both questionnaires were created on the web using Google Form. The CSTE-Q/D covered the questions regarding the tasks performed by developers: 68 questions in total. In contrast, 55 questions, were devised for the testers in the CSTE-Q/T version. Questions containing mutual aspects common to both developers and testers were included in both versions. In both versions, all answers were required except for that of the optional *open-ended* question added at the end of each questionnaire. In order to avoid the results being influenced by the respondent's fatigue, we asked five PhD students from different countries, some of whom were not particularly fluent in English, to check the time required to fill in the questionnaire. They were asked to time how long it took to fill in the questionnaire. For non-fluent English speakers, the total time taken to complete the questionnaire was 18-25 minutes, while it was 8-15 minutes for those fluent in English. According to the relevant literature (Borst, 2010), this time range of 8-15 minutes is considered an acceptable time frame. For more information on both versions of the questionnaire see Appendix C, Part I Table C.1 for developers and Part II, Table C.2 for testers.

6.2.4 Evaluation of Questionnaires and Measurement Quality

To evaluate the CSTE-Q/T and CSTE-Q/D questionnaire, we completed two main procedures: iterative question refinements and testing the questionnaires' reliability and validity. The former is achieved by enhancing the questionnaire's accuracy and overall structure, while the latter is performed to verify the questions' content and the relationships between them.

6.2.4.1 Iterative Question Refinement

After creating the first draft of the CSTE-Q/T and CSTE-Q/D, the process of refining and assessing was implemented. The iterative refining process aimed to review the questionnaires, remove unnecessary and duplicated questions. For this, nine experts were invited to assess the structure and accuracy of each individual question within all dimensions of both CSTE-Q/T and CSTE-Q/D; this process was repeated twice. The experts were recruited from four different areas; three having theoretical and empirical experience in mobile app development and testing; and another three having experience in crowdsourcing, two having previous experience in carrying out quantitative studies; and one being an expert in the development of questionnaires. The experts were given access to the online versions of both CSTE-Q/T to facilitate the refinement process. Feedback was collected manually, with recommendations written on paper, or electronically via email. The refinement process focused on the following criteria:

- **Relevance:** Ensuring the questions covered all aspects required for assessing the success of the crowdtesting approach.
- **Structure:** (1) Reorganising questions to be in a meaningful order and format; (2) Checking the integrity of each question in terms of language and formulation.
- **Simplicity:** Reducing the ambiguity of the questions and improving clarity, as well as eliminating redundant questions.

Based on the experts' recommendations, the questions for both CSTE- Q/D and CSTE- Q/T were amended and improved. In the CSTE- Q/D, two additional questions were added to address other important aspects of crowdtesting not already covered. These were "Task decomposition" and "controlling of non-active testers". Also, five questions were deemed irrelevant to the aims of the research. As a result, three were amended to be more relevant, while the remaining two were deleted entirely. A further six questions were also deleted due to having similar meanings which would thus produce the same results. Another ten questions were revised, with five ambiguous questions being rephrased to be more precise and clearer, while the other five were rephrased to make them shorter and easier to comprehend. The assessment of the CSTE-Q/T also led to several amendments. The first was the addition of two questions regarding "Large-scale Collaboration" and "Work Flexibility" under the benefits dimension. Six questions were then eliminated, four of which were irrelevant to the job of testing, while the other two questions were deleted for repeating concepts already covered. Four ambiguous questions were revised and rephrased to make them more precise and clearer.

6.2.4.2 Reliability and Validity Assessment

Validity and reliability testing are important methods for assessing the ability of a research tool, such as a questionnaire, to achieve the target objectives (Taherdoost, 2016a). The objective of a validity test is to measure whether the instrument does in fact measure what the researcher intended (Bolarinwa et al., 2015; Taherdoost, 2016a). The objective of a reliability test is to measure whether the collected data is consistent and stable over time (Sarmah and Hazarika, 2012). In this research, after the initial revision of the first draft of both the CSTE-Q/D and CSTE-Q/T questionnaires, a second draft was produced based on the feedback gathered from the iterative question refinement process. Following this, it was necessary to test the second draft of each instrument to confirm its reliability and validity, alongside identifying any further updates required before distribution to a larger number of real participants. A validity test was performed to gauge whether the questions accurately measured the success of the crowdtesting experience, in order to meet our study objectives. A reliability test was also performed to ensure that the questions were clear, unambiguous, and understandable for both developers and testers, thus offering internally consistent results (Bolarinwa et al., 2015; Taherdoost, 2016a). The validity and reliability

test were performed through the pre-testing and pilot studies. The following sections explain the implementation of these two procedures in more detail.

A. Pre-testing: Face and Content Validity

The pre-test was performed in this research using a face and content validity test. This test aimed to assess whether the questions accurately covered the full range of all functionalities and requirements of our crowdtesting approach that need to be evaluated. This test was conducted by showing all questions of the CSTE-Q/D and questions of the CSTE-Q/T to five experts. The experts were selected from both academia and industry with different experience: two practitioners in the field of mobile app testing and development in industrial testing organisations; one PhD student in computer science from the University of Sheffield; two professors at the computer science department at the University of Jordan who are familiar with quantitative research. Accordingly, the experts were given access to the online versions of both questionnaires for validation purposes. Experts were asked to review the questions based on the highlighted criteria, alongside completing both questionnaires and providing their feedback regarding the experience. They were also asked to discuss what they had understood from each question, whether they had any recommendations and the reasons behind their selected answers to display their understanding of all questions to be confirmed. The analysis and assessment of each question focused on the following criteria:

- **Relevance:** The extent to which questions were salient to crowdtesting, compatibility testing of mobile devices, and fragmentation issues;
- **Essentiality:** The extent to which questions help to achieve the aim of the study;
- **Completeness:** The extent to which the questionnaire contained all necessary parts, meaning that no important questions faced omission;
- **Understandability (Clarity of Content):** The extent to which each question was clear enough to be interpreted and understood by different readers as intended;
- **Feasibility (Technical Quality/Format):** The extent to which questions were well-presented so that all response scales were in the correct order and direction.

Several insightful suggestions were received and implemented. The questions in the second draft of both questionnaires were amended based on those suggestions prior to the main field study. The suggestions for the CSTE-Q/D included adding questions regarding the payment schema and value under the "incentivisation" dimension; simplifying and rephrasing three questions for better understanding; rearranging the sequence of some questions to increase fluency and to enhance organisation; changing terms so that "internal complexity" was amended to "architecture", and "more information" was changed to "insight". Also, the

term "important information" was changed into "sensitive information", and "good or better" was changed into "effective", with the amendments deemed to be more indicative of the actual meaning intended. Further minor amendments were made, but no questions were deemed irrelevant or non-essential enough for deletion. On the other hand, the suggestions for the CSTE-Q/T included the deletion of two questions which were deemed only relevant to developers. The first question related to controlling the privacy of tasks, while the second question was about the announcement and publication of tasks. Another six questions were identified as not articulated clearly enough, thus, requiring to be rephrased for better understanding. A further three questions were rearranged to flow more smoothly and be more organised. Lastly, further minor amendments were made to a few questions, including the changing of repeated words. All other questions were considered essential and no further questions were added.

B. Pilot Studies

Both questionnaires underwent pilot testing, to assess their feasibility, consistency and accuracy. Two studies were carried out to ensure the construct validity and internal consistency reliability of both questionnaire versions. Participants in both studies were mobile app developers/testers experienced with Android, iOS, or both. Some of them were freelancers, while others were employees in various mobile app development and testing companies. A sample size of 20 was used in both pilot studies, with a response rate of 100%. Similarly, in both pilot studies, the number of participants who experienced with Android was higher than iOS. Likewise, the number of freelancer participants was also higher than those who were employees. Table 6.1 summarises the characteristics of the two pilot studies.

Table 6.1 Summary of the two pilot studies

Feature	Pilot Study1	Pilot Study2
Role	Developers	Testers
Sample size	n=20	n=20
Full Response Rate	100%	100%
Job Type	Android 62% iOS 48%	Android 68% iOS 54%
Job Status	Freelancer 57% Employee 43%	Freelancer 64% Employee 36%

Reliability Internal Consistency

It is very important to measure a questionnaire's ability to create reproducible results to ensure its reliability (Venkitachalam, 2014). In this research, internal consistency checks

were used as the most appropriate measure of reliability, when using Likert scales (Robinson, 2010; Taherdoost, 2016b). At the time of completion of the two pilot studies, the internal consistency was assessed by measuring the consistency of respondents' responses to all questions under all dimensions. This helps to gain more insight into how well the questions measured various aspects of crowdtesting to produce similar results. This was achieved by calculating the mean of the correlation among all developers' and testers' responses to all questions within each dimension in the CSTE-Q/D and CSTE-Q/T using the Cronbach's Alpha tests as applied in SPSS² statistical software. Tables 6.2 and 6.3 show the results of the internal consistency measurements across all questions within each dimension which were retained after the deletion of the weak questions which would affect the reliability of the questionnaires (see Appendix C, Part III). The tables show that the internal consistency coefficients (α) among all responses for the CSTE-Q/D ranged between 0.760 and 0.980. Similarly, for the CSTE-Q/T the range was between 0.732 and 0.960. The high α values indicate a high degree of reliability in both the CSTE-Q/D and CSTE-Q/T. This results from these values being greater than the minimum value of 0.70, which proves the reliability of a questionnaire, as mentioned in the literature of Ryu and Smith-Jackson (2006).

Construct Validity

A construct validity test was used to assess whether the different sets of questions in the CSTE-Q/D and CSTE-Q/T measure the different facets that we aim to measure (Venkitachalam, 2014). This was achieved by calculating the total correlation (Corrected Item (CI)) of all questions within both questionnaires using the Point biserial correlation coefficient (r_{bi}) to ensure that each set of relevant questions measures the dimension to which it belongs (Brown, 2001; Gupta, 1960). Tables 6.2 and 6.3 display the calculated CI value for each question with its associated dimension in both questionnaires. The point biserial correlation coefficient was used due to it being the most appropriate method for analysing interval and ratio data types in accordance with Ozer (1985). Table 6.2 and 6.3 show that CSTE-Q/D has good correlation (CI) between all questions and their dimensions, in the range from $r=0.574$ to $r=0.866$, and the CSTE-Q/T in the range from $r=0.526$ and $r=0.884$, as shown in Table 6.3. As all CI values presented in both tables are higher than $CI=0.30$ (the minimum accepted value) indicated in the literature of Unger-Saldaña et al. (2012), the validity of both CSTE-Q/D and CSTE-Q/T was proven. The questions presented in Tables 6.2 and 6.3. These are the final questions retained after the evaluation processes and those that used in both main studies after deleting the questions with low validity value see Appendix C, Part III).

²<https://www.ibm.com/analytics/spss-statistics-software>

Table 6.2 Internal Consistency Reliability (α) and Construct Validity of the CSTE-Q/D questions

Main dimensions and its sub-dimensions		Cronbach's alphas (α)	Corrected Item (CI)
Effectiveness			
<i>Task Defining and Distribution (TDD)</i>			
TDD 1	It enables me to define and publish testing tasks quickly with less effort.	(α) = 0.980	CI = 0.854
TDD 2	It enables me to distribute testing tasks on a large scale quickly.	(α) = 0.931	CI = 0.751
TDD 3	The task design method enables me to define the whole app as simple multiple tasks easily.		CI = 0.827
TDD 4	The two different levels of testing (feature and code) enable me to perform more effective compatibility tests.		CI = 0.789
TDD 5	It takes less time and effort to write task requirements.		CI = 0.809
<i>Assessing Specialised Testers Skills (ACC)</i>			
ACC 1	It provides an easy method of selecting testers in a short time.	(α) = 0.923	CI = 0.830
ACC 2	The suggestion of a list of eligible testers for performing a specific task helps facilitate the selection of testers.		CI = 0.843
ACC 3	Selection of testers with different skills and expertise areas to participate in and work on a specific task is helpful.		CI = 0.836
ACC 4	Insights provided into testers' performance helps select and invite several testers with specialised skills to work on tasks.		CI = 0.866
<i>Privacy and Protection (PP)</i>			
PP 1	It enables me to control the privacy of the task and prevent unauthorized testers from accessing it.	(α) = 0.878	CI = 0.760
PP 2	It has an effective method to protect sensitive information of testers (e.g., their work quality and reliability level) among each other.		CI = 0.773
PP 3	It has an effective strategy to protect the testing environment from having non-active and unreliable testers.		CI = 0.761
<i>Tracking Task Status (TS)</i>			
TS 1	Tracking the number of tests performed and the type of devices used for testing a specific task is helpful.	(α) = 0.821	CI = 0.702
TS 2	It is very useful to track the task status of the published task during the active test cycle		CI = 0.702
<i>Outcomes Diversity (OD)</i>			
OD 1	Different unexpected issues can be discovered in the early stages of an app's development process through this approach.	(α) = 0.772	CI = 0.629
OD 2	Results obtained by different testers' backgrounds represent different ways of thinking about implementing the app's functionalities.		CI = 0.629
<i>Results Aggregation and Tracking (RAT)</i>			
RAT 1	It has an effective and useful reports aggregation and tracking mechanism.	(α) = 0.865	CI = 0.774
RAT 2	The automated tracking mechanism helps reduce the time required for collecting and organizing reports.		CI = 0.748
RAT 3	It enables me to view and filter testing reports easily.		CI = 0.714
<i>Results Evaluation and Quality Control (RQC)</i>			
RQC 1	It requires less time and effort to assess the performance of testers on performed tasks.	(α) = 0.808	CI = 0.743
RQC 2	The evidence provided by public testers regarding testing results was NOT helpful to gauge their accuracy.		CI = 0.780
RQC 3	The insight provided into the results quality distribution helps understand the reason for low quality work.		CI = 0.644
RQC 4	The quality evaluation metrics enable me to perform a quick and fair evaluation of testers' works.		CI = 0.641
<i>Cost Effectiveness (COS)</i>			
COS 1	We can cover more devices to test our tasks at a lower price through this approach.	(α) = 0.864	CI = 0.760
COS 2	It is relatively cheaper than other approaches relying on a middleman.		CI = 0.760

Main dimensions and its sub-dimensions	Cronbach's alphas (α)	Corrected Item (CI)
Interaction between Peers and Workflow (INT)		
	(α) = 0.800	
INT 1		CI = 0.636
INT 2		CI = 0.747
INT 3		CI = 0.755
Testers Reliability and Trustworthiness (RT)		
	(α) = 0.891	
RT 1		CI = 0.755
RT 2		CI = 0.683
RT 3		CI = 0.776
Tester Motivation and Incentivisation (MI)		
	(α) = 0.878	
MI 1		CI = 0.772
MI 2		CI = 0.810
MI 3		CI = 0.783
MI 4		CI = 0.731
Knowledge Sharing (KS)/Wiki		
	(α) = 0.879	
KS 1		CI = 0.783
KS 2		CI = 0.764
KS 3		CI = 0.751
Ease of use (EU)		
	(α) = 0.760	
EU 1		CI = 0.615
EU 2		CI = 0.615
Benefits (BNF) / Advantages		
	(α) = 0.885	
BNF 1		CI = 0.729
BNF 2		CI = 0.637
BNF 3		CI = 0.751
BNF 4		CI = 0.746
BNF 5		CI = 0.746
BNF 6		CI = 0.771
BNF 7		CI = 0.700
BNF 8		CI = 0.612
Satisfaction		
	(α) = 0.896	
SAT 1		CI = 0.799
SAT 2		CI = 0.833
SAT 3		CI = 0.756

Table 6.3 Internal Consistency Reliability (α) and Construct Validity for the CSTE-Q/T questions

Main dimensions and its sub-dimensions		Cronbach's alphas (α)	Corrected Item (CI)
Effectiveness		(α) = 0.960	
Task Selection (TS)		(α) = 0.815	CI = 0.687 CI = 0.687
TS 1	It provides an easy and effective method of selecting tasks		
TS 2	The list of suggested tasks helps me to select and perform more tasks in a short time.		
Accessing Task Requirements (ACCR)		(α) = 0.838	CI = 0.746 CI = 0.608 CI = 0.768
ACCR 1	The information and instructions given by developers are sufficient to perform an effective testing process.		
ACCR 2	The presentation of tasks requirements is sufficiently clear and understandable.		
ACCR 3	Information on task complexity level helps me perform the most suitable test and obtain better quality results.		
Results Submission (SUB)		(α) = 0.859	CI = 0.782 CI = 0.788 CI = 0.689
SUB 1	The way of submitting results is simple and uncomplicated.		
SUB 2	Automatic detection service of mobile data helps me to collect and submit more accurate results in fewer steps.		
SUB 3	It enables me to perform that same task with different mobile devices quickly with less effort.		
Results Evaluation and Quality Control (RQC)		(α) = 0.919	CI = 0.810 CI = 0.825 CI = 0.876
RQC 1	It has an effective and useful mechanism for tracking the status of submitted reports.		
RQC 2	The quality evaluation metrics are fair and increase my commitment and desire to keep participating.		
RQC 3	The insight provided into the history of my work quality is helpful to know how well I am doing.		
Feedback Given (FED)		(α) = 0.811	CI = 0.682 CI = 0.682
FED 1	The direct and clear information about the effectiveness of my performance helps me to improve my work in the future.		
FED 2	The feedback about how well I am doing increases my satisfaction and helps me to continue to participate and work.		
Interaction between Peers and Workflow (INT)		(α) = 0.884	CI = 0.774 CI = 0.826 CI = 0.736
INT 1	Direct interaction between developers and testers, without middlemen, DOES NOT help me to understand more test requirements and effective completion of tasks.		
INT 2	Increasing the strength of the connection and interaction with developers and the other testers community makes me want to participate more.		
INT 3	Direct interaction reduces delays caused by the middlemen figures alike (managers or leaders) as compared to other approaches.		
Testers Reliability and Trustworthiness (RT)		(α) = 0.862	CI = 0.692 CI = 0.760
RT 1	The use of this approach helps improve my reputation among other testers in the global community.		
RT 2	I believe that the reliability level identified for each tester is fair and trustworthy.		
Privacy and Protection (PP)		(α) = 0.822	CI = 0.712 CI = 0.574
PP 1	It has an effective way to prevent ineligible testers from accessing and selecting the private task.		
PP 2	It provides an effective and secure method of linking device information with the submission form.		

Main dimensions and its sub-dimensions	Cronbach's alphas (α)	Corrected Item (CI)
<i>Tester Motivation and Incentivisation (MI)</i>		
MI 1 The reward value is fair and reflects the effort that I have put into crowdtesting work.	(α) = 0.732	CI = 0.526
MI 2 The pay-per-device increases the motivations of testers to do the test on more devices and get a higher quality of results.		CI = 0.638
MI 3 The reward information provided for each completed task seems fair to me.		CI = 0.613
<i>Knowledge Sharing (KS)/Wiki</i>		
KS 1 It provides an effective knowledge-sharing environment between worldwide developers' and testers' communities.	(α) = 0.883	CI = 0.867
KS 2 It has an effective documentation and knowledge searching mechanism.		CI = 0.768
KS 3 Documentation of testing steps and scenarios helps improve my testing strategy in different testing areas.		CI = 0.697
<i>Ease of use (EU)</i>		
EU 1 It is easy to use the proposed approach in daily work routine and industry practices.	(α) = NA	CI = 0.689
EU 2 Both Android and iOS specialists can easily use and interact with the proposed approach.		CI = 0.689
Benefits (BNF) / Advantages		
	(α) = 0.900	
BNF 1 It provides me with an opportunity to receive help from other communities.		CI = 0.564
BNF 2 It provides me with an opportunity to improve my testing skills in different areas.		CI = 0.550
BNF 3 It provides me with an opportunity to perform tests in the areas that I am good at.		CI = 0.673
BNF 4 It provides me with an opportunity to perform the test in different ways.		CI = 0.778
BNF 5 It provides a suitable environment for professional and beginner testers.		CI = 0.722
BNF 6 It supports large-scale collaboration and communication among testers and researchers in the domain.		CI = 0.721
BNF 7 This approach enables me to perform test anywhere and at any time I need to.		CI = 0.781
BNF 8 Participating and using this approach lets me feel a sense of personal achievement.		CI = 0.780
Satisfaction		
	(α) = 0.937	
SAT 1 Overall, I am satisfied with the use of the public crowdtesting approach to conduct the test in the future.		CI = 0.855
SAT 2 I am satisfied with the nature of the crowdtesting workflow and service provided.		CI = 0.878
SAT 3 I am satisfied with the benefits that will result from the use of this approach in the future.		CI = 0.884

6.3 Sampling Methods and Data Collection Procedure

The data of the two evaluation empirical studies was collected using the two survey questionnaires, the CSTE-Q/D for developers, and the CSTE-Q/T for testers. Since our target was to reach mobile app developers and testers worldwide, the population was sampled via a random sampling method (Bryman and Burgess, 2002). The sampling was performed by sending an open invitation to participate in the evaluation studies leveraging different mobile apps developers' and testers' communities across social media platforms such as, Facebook, LinkedIn, and our Twitter account. An invitation also was sent to different app development/testing organisations worldwide. The invitation included our contact details to enable confirmation of participation. Once the participants contacted us, we checked whether each participant met the required criteria (Chapter 3, Section 3.2.3), and provided them with an overview of the project in general. Further instructions were provided to these potential participants, and they were informed that they had the right to refuse to participate or answer the questionnaire if they did not want to. Next, we provided them with copies of the information sheets and encouraged them to read them carefully. We then offered them the opportunity to discuss any of their queries and give them clarification where necessary. Finally, to legally gain their final consent, we sent them an online consent form to sign, designed by a Google Form (see Appendix A, Part II).

Having received the consent form, we provided each participant with the URL of the crowdtesting platform and access to the online survey CSTE-Q/D for developers and CSTE-Q/T for testers. We asked both developers and testers to use our platform and perform several tasks before completing the questionnaire. We ended up with the selection of a set of tasks that developers (see Table 6.4) and testers (see Table 6.5) are required to perform, and this information was used for the final evaluation of our approach. Overall the data collection process took 4 months to complete (starting August 2019 and ending November 2019). This period provided the participants with the flexibility to use the approach and complete the questionnaire. Meanwhile, we sent them one reminder every two weeks. Once the participants submitted their responses, they were automatically stored on Google spreadsheets in a structured manner. It is important to know that the collection of participants' responses was conducted strictly anonymously. The fieldwork yielded 34 developers' and 31 testers' responses with different levels of experience in Android and/or iOS and from different countries. The detailed information about the results is presented in Chapter 7 for developers and Chapter 8 for testers. These numbers are considered sufficient to evaluate a new proposing approach (Hosseini et al., 2015). It is important to note that none of these participants was involved in the pilot testing of the questionnaires or the conducted exploratory study detailed in Chapter 4.

Table 6.4 List of tasks related to developers

Task 1:	Define a public task and distribute it to the public.
Task 2:	Define a private task and send an invitation to testers eligible for that task.
Task 3:	Define your project as a set of tasks or as one task.
Task 4:	Check the summary-dashboard to see testers' classification based on experience, quality level, reliability level, and working area.
Task 5:	View the list of suggested testers for each task.
Task 6:	Select the appropriate testers based on identified criteria.
Task 7:	Control the privacy of your task.
Task 8:	Check your task status (testers location, tested devices, #.of times performed).
Task 9:	Track submitted reports.
Task 10:	View the distribution of your testers results.
Task 11:	Validate testing reports.
Task 12:	Evaluate your testers' work.
Task 13:	Provide incentives - reward your testers.
Task 14:	Delete/stop one task within a set of task.
Task 15:	Search for an issue or solution in wiki.
Task 16:	Post issue in the wiki

Table 6.5 List of tasks related to testers

Task 1:	Search for different tasks with different criteria.
Task 2:	Check the list of suggested tasks.
Task 3:	Try to accept the received invitation from developers.
Task 4:	View the task requirements and then accept the task to perform.
Task 5:	Submit testing report
Task 6:	Submit another testing report for the same task with different mobile devices.
Task 7:	Update/edit/review your testing report status
Task 8:	Check your payment status (accept or send feedback to developers)
Task 9:	Verify mobile data detection services from different devices.
Task 10:	Search the wiki with different keywords
Task 11:	Add result or comment for a specific post in the wiki
Task 12:	Try to access private task and check if you are able to test it

6.4 Data Analysis Method

Descriptive statistical analysis was used to summarise, describe, and interpret the responses to each question, as well as present them in a meaningful way. Inference statistical analysis was used to draw conclusions regarding the differences between different groups in terms of the benefits, satisfaction, and effectiveness of our approach. Also, it was used to provide more insight into the correlation and factors that could affect the overall effectiveness of the approach. Inference statistical analysis would enable us to generalise the conclusions to a

larger population. Figure 6.1 summarizes the analysis approach and statistical methods used to analyze the data gathered from both studies.

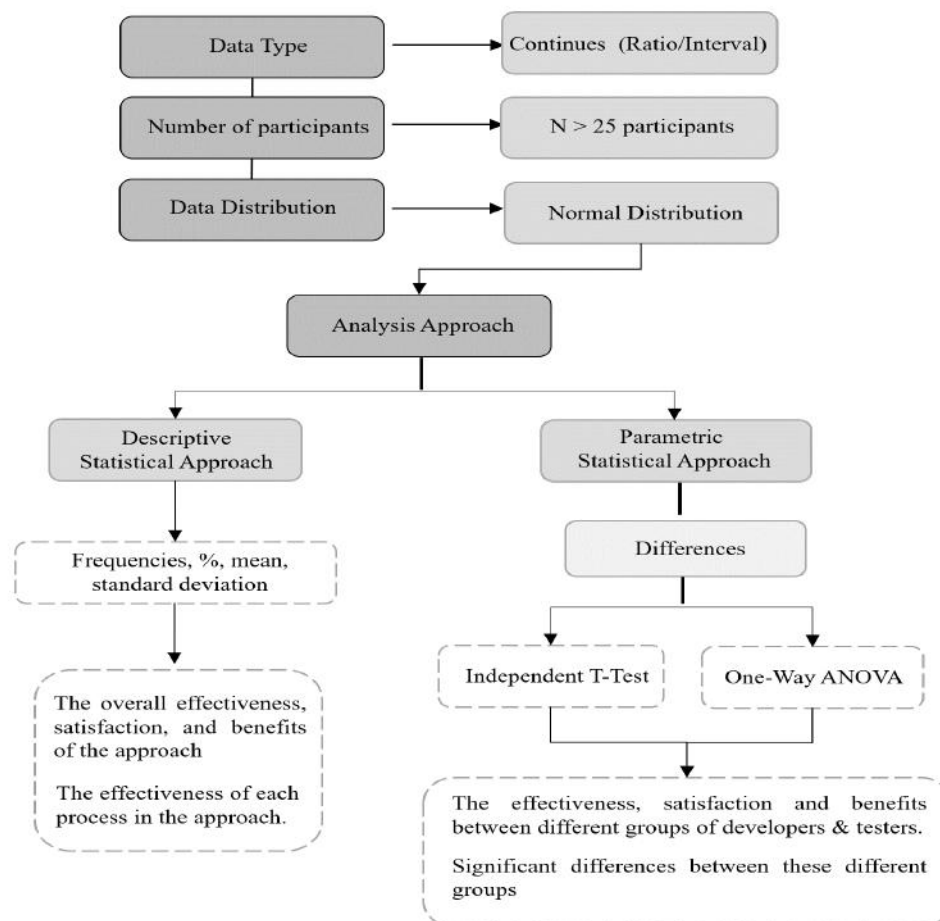


Fig. 6.1 The Descriptive and Inferential Statistics Data Analysis Methods

6.4.1 Data Coding

In order to analysing the collected data, descriptively and then statistically we used the SPSS software. First, we exported the data from Google Form to Excel. Then, we structured them (coding the questions and categorical responses) for use in SPSS. The questions were encoded according to their relevant dimensions. For example, the satisfaction questions SAT1, SAT2, and so on, and the benefit = BNF. The effectiveness questions were encoded based on their relevant sub-dimensions such as: Task Defining and Distribution = TDD, Tracking Task Status = TS, for all other codes see Table 6.2 and 6.3. The labels ranked on a Likert scale were coded as numbers; "Strongly Agree" was given 5 and "Agree"= 4, "Neutral"= 3, and "Disagree"= 2, and "Strongly Disagree" =1. The variables of demographic questions were

also coded as numbers, as shown in Table 6.6. For the years of experience, we grouped the results of participants into three levels; beginner, intermediate, and expert/senior and then coded them as a number (see Table 6.6).

Table 6.6 The variables of demographic questions and their coding

Variable Name	Coding	Variable Name	Coding
<i>Operating System Type:</i>		<i>Job Status:</i>	
Android	1	Freelancer	1
iOS	2	Employee	2
Both	3		
<i>Years of experience :</i>		<i>Company size (Number of employees) :</i>	
beginner 1- 2 years	1	Small Organisation ($n \leq 200$)	1
intermediate 3 - 4 years	2	Big Organisation ($n > 200$)	2
expert/senior > 4 years	3	Academia	3
		Government	4

6.4.2 Data Checking and Cleaning

After the coding process, we checked and cleaned the collected data, removing any noisy values (e.g., outlier values), inappropriate responses, repetitive submissions, and inconsistent responses that may negatively affect the quality of results. As all the survey questions were set as mandatory in both studies, incomplete survey issues were avoided. The process of data checking and cleaning included the following:

- **For repetitive submission:** we individually checked the responses from each submission of the online questionnaire to ensure that no respondents accidentally submitted more than once. No duplicated submissions were detected.
- **For outlier responses (value):** the outliers were looked at, meaning the values that differed significantly (much smaller or much larger) from all other collected data values. As Likert-scale questions were used for all measurement variables, it was impossible to have outlier values, which ensured the accuracy of our data (Treiblmaier and Filzmoser, 2011). As for responses to demographic questions, the results showed that question 6, (how many days they used the platform) produced three outlier answers. For the developers' data, just one answer of 30 days was found. For testers, two outliers of 12 and 16 days were found. As the average usage of the approach by both developers and testers was between 1-3 days, these answers were ignored as they would not add any value to the end results.
- **For inappropriate responses:** in question 2 (operating system type), one developer have selected both Android and iOS, and because it is lower than the minimum values

required to represent the group in SPSS, which need to be at least five values, thus this developer was considered with the iOS group as it had the lower percentage of participants. For question 3 (job status), one developer selected both a freelancer and a full-time employee, but since he is an employee this was counted as an employed developer. Question 5 relates to the participants' countries, with the answers demonstrating different expressions for the same answer. Here, respondents were naming the same country in alternative ways, such as Cairo rather than Egypt. Another example of this was the UK, United Kingdom, England, and British. To overcome this issue, answers indicating the same country were grouped so that the percentages could be recalculated.

- **For inconsistent responses:** we checked all answers of each respondent to detect whether any respondent had responded to survey questions with answer that are the opposite of answer they provided to a different survey question, which would lead to inconsistency (contradicting). No inconsistent answers were found. Thus, all the data collected was accepted as valid for the analysis.

After cleaning the data and removing inappropriate and inconsistent responses, the Excel file of structured data was loaded in SPSS to start the analysis.

6.4.3 Descriptive Statistical Analysis

All calculations and statistical analyses of the collected quantitative data were performed using SPSS (IBM Corp. Release 2017. IBM SPSS Statistics for Windows, Version 25.0. Armonk, NY: IBM Corp). As a first step, a descriptive statistical analysis was performed for all questions by calculating the frequencies and percentages to clarify the general data of both studies. For further analysis of the responses to Likert-scale questions, the mean analysis was applied. Subsequently, the results were organized and presented using graphs and tables. The *open-ended* question in both studies was analysed using Summative Content Analysis (SCA) (Hsieh and Shannon, 2005). The same steps used to analyse the open questions in Chapter 4 were followed by manually extracting the meaningful information from the responses, coding, and then summarising them in categories. Figure 6.2 illustrates the coding method applied in the analysis of the *open ended* question added in both studies. The descriptive statistical analysis results of both *open ended* questions and Likert-scale questions is described in detail in Chapters 7 and 8 for the developers and testers, respectively.

Mean Analysis

A mean analysis was performed to evaluate the suitability and effectiveness of the proposed approach and its sub-processes in addressing the difficulties of performing large-scale com-

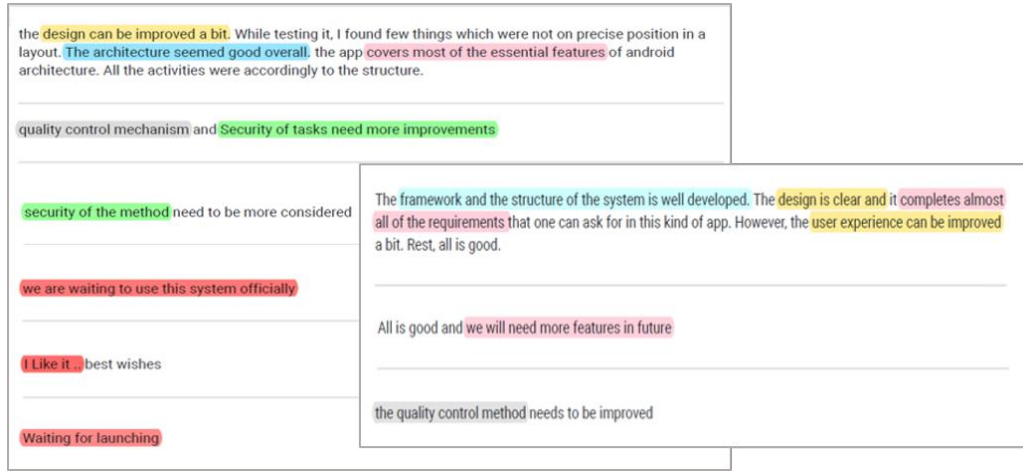


Fig. 6.2 Screenshots of the SCA method applied for analysing *open-ended* question in both studies.

patibility testing. The mean score of responses to each question, under each sub-dimension (each process), was calculated through the formula 6.1 to measure the effectiveness of each feature under that sub-dimension. In other words, to determine to the extent to which each feature of the sub-dimension contributed to the overall effectiveness of the sub-dimension.

$$\text{Mean of the responses} \\ \text{(Feature Effectiveness)} = \left(\frac{\sum_{n=1}^n \text{Responses to the question}}{\text{Number of all responses to the question}} \right) \quad (6.1)$$

The mean of each sub-dimension was calculated based on the effectiveness scores of the features (mean value of each question under the specific sub-dimension) that resulted from formula 6.1 to determine the effectiveness of each sub-dimension as follows:

$$\text{Sub-dimension} \\ \text{Effectiveness} = \left(\frac{\sum_{n=1}^n \text{Mean of each question under sub - dimension}}{\text{Number of all questions under sub - dimension}} \right) \quad (6.2)$$

The overall effectiveness of the whole approach was then quantified based on the effectiveness values of sub-dimensions (mean scores of sub-dimensions) that resulted from the formula 6.2, as in the following formula:

$$\text{Overall Effectiveness} = \left(\frac{\sum_{n=1}^n \text{Mean of each sub - dimension}}{\text{Number of all sub - dimensions}} \right) \quad (6.3)$$

A similar analysis method to the above was followed to determine the overall satisfaction and the benefits of the proposed approach. The mean of the responses to each question under the satisfaction or benefit dimension were quantified individually as in formula 6.4. Then the overall mean of the dimension was calculated based on these means as in formula 6.5:

$$\text{Mean of the Responses} = \left(\frac{\sum_{n=1}^n \text{Responses to the question}}{\text{Number of all responses to the question}} \right) \quad (6.4)$$

$$\begin{array}{l} \text{Overall Satisfaction} \\ \text{or Benefit} \end{array} = \left(\frac{\sum_{n=1}^n \text{Mean responses of each question under dimension}}{\text{Number of all questions under dimension}} \right) \quad (6.5)$$

This mean analysis method was applied to the collected data in both studies, the developers and the testers. This was done in order to determine the overall extent to which the proposed approach is effective, satisfying, and beneficial for both developers and testers. To specify the level of the benefit, overall satisfaction, and overall effectiveness of the approach and its sub-processes, the mean score's interpretation (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.68 – 5.00) was used (Mohamed et al., 2017; Yahaya et al., 2014), to give a detailed description of the data obtained and its the computed means. On top of that, the percentage of each mean (mean values produced from above formulas) was calculated to describe the effectiveness, benefits, and satisfaction of the approach in a clearer way and with more interpretation of the identified mean scale level. The percentage was computed by dividing the obtained mean score by 5.00 (the highest response Likert-scale) and multiplied by 100.0.

6.4.4 Inference Statistical Analysis

In order to select the most suitable inference statistical test to analyse our data, we first tested the normal distribution displayed within the participants' responses via calculations of the mean and variance of the three main dimensions using the Kolmogorov–Smirnov test (KS test) in SPSS (Drezner et al., 2010). The results showed that the sample set of responses had normal (or nearly normal) distribution for most of the dimensions in both studies with statistical test value higher than $p > 0.05$. Figure 6.3 and 6.4 shows the distribution of the sample data for all three dimensions. It can be seen that the responses of the effectiveness in Figure 6.3 and the responses of satisfaction in 6.4 may not represent the normal distribution curve. However, the results of Kolmogorov–Smirnov indicated the normality with $p\text{-value} > 0.05$, so it was considered to be a normal distribution. For more details about results from Kolmogorov–Smirnov tests regarding normal distribution see Appendix C, Part IV, Table C.4. As our data met the conditions of parametric analysis (e.g., the normality distribution of most of the data, usage of Likert-scale with Ratio/Interval, and the number of participants exceeded 25 participants), the parametric tests were used (Harpe, 2015; Kataike et al., 2017; Mircioiu and Atkinson, 2017).

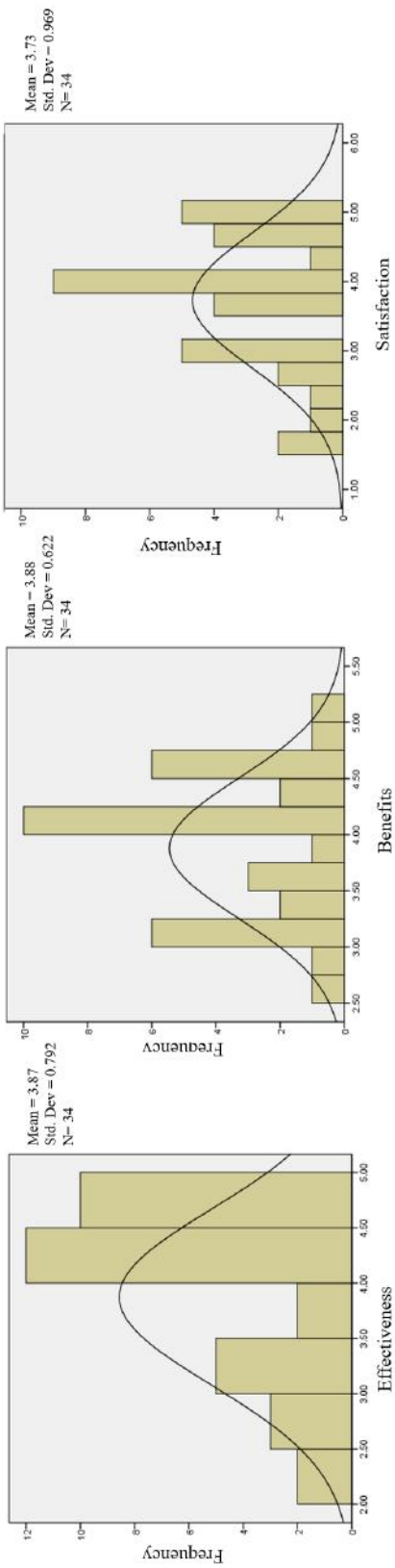


Fig. 6.3 A histogram of the normality distribution of the developers' data

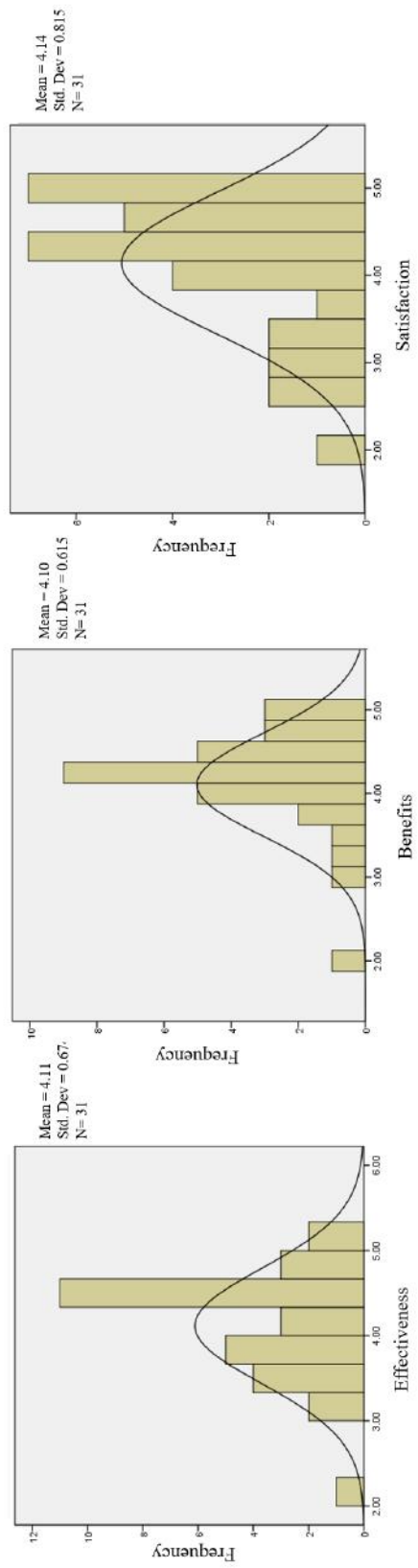


Fig. 6.4 A histogram of the normality distribution of the testers' data

Differences Analysis

To assess how successful our testing environment is, in supporting the industry's mainstream work practices and the actual needs of everyday work routine for the different groups of developers and testers (based on demographic variables), a differences analysis was used. The Independent T-test was used to assess and analyse differences between two groups and One-Way ANOVA to test the differences between three groups (Kataike et al., 2017). In this differences analysis, attention was drawn to the t-value/f-value to determine the difference that exists between independent groups. A p-value of less than 0.05 was considered to indicate the statistical significance of these differences. This analysis method was applied to the different developers and testers groups to evaluate the benefits from and satisfaction in the use of the crowdtesting approach based on their demographic characteristics as follows:

- **Operating System Type:** The T-test was used to test the differences between the two independent groups, Android and iOS developers and to specify whether there were significant differences between them. In respect of testers, some of them selected both mobile platforms and they formed a third group, one-Way ANOVA was therefore used to test the differences between the three groups, Android, iOS, and testers experienced in both operating systems.
- **Job Status:** The independent T-test was used to test the differences between freelance workers and employees for both developers and testers.
- **Years of experience:** The one-Way ANOVA test was employed to test the differences between the three different groups: beginner/novice (1-2), intermediate/mid-level (3-4), and expert/senior (more than 4) levels for both developers and testers alike.
- **Company size (Number of employees):** Workplace size variable involves four-levels: big/small organisation, academia, and government. Upon a review of all participants' responses, it was evident that all of the participants were from big or small organisations in both studies, except one developer from "Academia". This answer was considered as small organisations. This is because the minimum number of values accepted for each group in the SPSS must be at least 5. In this case, the t-test for both developers and testers was used to compare the differences between those who work in small and big organisations.

6.5 Chapter Summary

The existing questionnaires and tools used in the literature were considered insufficient for the purposes of our research as they were considered to be incapable of addressing the specific crowdtesting context of interest to be measured. In order to meet the requirements

needed for evaluating the satisfaction, benefits/advantages, and effectiveness of the proposed approach, the CSTE-Q was developed for use in this research. An explanation of the whole development process has been offered in this chapter, describing the steps taken in the designing, developing, and assessing the processes used to create the CSTE-Q. Firstly, an analysis of literature regarding crowdtesting and mobile app testing was conducted, thus allowing for the exploration of the main aspects that need to be measured as sub-dimensions within effectiveness. These sub-dimensions were deemed likely to have a significant impact on the evaluation of the success of the crowdtesting approach in terms of effectiveness, overall satisfaction, and benefits. These sub-dimensions were for instance, Task Defining and Distribution, Tracking of Task Status, Privacy Protection, Accessing Specialized Testers Skills, Outcomes Diversity, Task Selection, Accessing Task Requirements, Results Submission, Feedback Given, Results Evaluation and Quality Control, Cost Effectiveness, Interaction between Peers, Reliability and Trustworthiness Management, Motivations and Incentivisation, Knowledge sharing/WiKi, and Ease of use.

As the next step, an initial version of each of the CSTE-Q/D and CSTE-Q/T was created. All questions in both questionnaires were adapted from different resources in the literature. Other new questions were also generated, which were agreed upon by a consensus panel. A refining process was conducted with nine experts, thus, allowing for modifications and improvements of the first drafts of both questionnaires. Subsequently, this led to the formation of the second draft of questionnaires. After a validation test was conducted on the second draft of questionnaires by five experts to help validate the content of the questions. This was followed by two pilot tests which were conducted on CSTE-Q/D with 20 participant developers and on CSTE-Q/T with 20 testers. The experts' judgment and statistical tests supported the idea that both CSTE-Q/D and CSTE-Q/T had adequately relevant content and were simple yet sufficiently comprehensive with only minor adjustments. The results of testing showed that the CSTE-Q/D and CSTE-Q/T both have good content and face validity. The construct validity results demonstrated that both questionnaires offer good internal correlation, with CI values for all questions under each dimension in the range of 0.574 to 0.866 for CSTE-Q/D and between 0.526 and 0.884 for CSTE-Q/T. However, these values are all still higher than the published literature's threshold value of 0.30. Similarly, the Cronbach's alpha results for each dimension in both questionnaires demonstrated significant internal consistency reliability for all questions with values exceeding 0.70, which is the acceptable value for proving the reliability of the questionnaire. The values ranged from 0.760 and 0.980 in the CSTE-Q/D and from 0.732 and 0.960 within the CSTE-Q/T.

After proving the validity and reliability of both questionnaires, two empirical evaluation studies were performed by distributing survey questionnaires to different groups of developers and testers on social media websites. A random sampling method was applied to select the main samples for the two evaluation studies described. The participant developers and

testers used the platform to perform the required tasks and then completed the appropriate questionnaire. In total, responses were collected for analysis from 34 developers and 31 testers with varying experience in Android and iOS from countries worldwide. These participants had used the implemented approach from one day to one week. The descriptive and inference statistical analysis method was performed on all collected responses. Summative Content Analysis was used to effectively analyse the *open-ended* question. Likert scale responses were used for all questions in the questionnaires. The responses were coded, cleaned, and then analysed using parametric approaches. Statistical calculations like means, standard deviation or variance, and frequency were implemented to calculate the satisfaction, benefits/advantages, and effectiveness of our approach. Moreover, statistical tests independent T-test, one-way ANOVA were also implemented, as appropriate, to test the differences between collected data for both studies.

In the next chapters, the results of the analysis of the two empirical evaluation studies are described and descriptive analysis and the inferential statistical analysis of the results presented and discussed. For developers, see Chapter 7 and for testers, see Chapter 8. Chapter 9 provides a discussion of the results of both studies and its comparison with the literature available.

Empirical Study(I): Developers Experience With Crowdfunding Approach

7.1 Introduction

The previous chapter has described the methodology of conducting our empirical evaluation study for the developers, including measurement variables, data collection method, sampling method, and procedures that developers need to achieve. We evaluated our approach and data collected from 34 worldwide android and iOS developers with different levels of experience. This chapter will describe the main findings and results of the analysis of the collected data from the participant developers to evaluate the effectiveness, benefits, and satisfaction of the proposed crowd-based compatibility testing approach. In the beginning, we describe the results of the descriptive analysis, including the demographic information of the participants' developers and their actual answers to the CSTE-Q/D questionnaire. As a second step, we describe the results of the statistical analysis calculated using SPSS. First, we show how effective the overall approach, main processes, and features are in performing effective compatibility testing. Second, we explain the results of the benefit of the overall approach from the developers' perspective and then specific benefits that each group of developers will gain. Third, we describe the overall satisfaction of using the proposed approach in performing effective compatibility testing from different developers' groups' perspectives. Finally, we provide the results of the summative content analysis of the *open-ended* question to provide explicit knowledge about the future enhancement that developers recommend us to achieve.

7.2 Developers Demographic Information

As mentioned in Chapter 6, the empirical evaluation study related to the developers resulted in the participation of 34 worldwide developers. Table 7.1 shows the frequencies and percentages of the participants characteristics: operating system type, job status, company size, years of experience, usage period, and country. In terms of operating system type, 22 (64.7%) of the participants were Android developers, while 11 (32.4%) were iOS developers. Only 1 (2.9%) developer selected both operating systems. Job status (i.e., employed full-time or freelancer) was also considered, with 24 (70.6%) reporting themselves as in full-time employment, and 10 (29.4%) describing themselves as freelancers. For the organization size, small organisations were defined as those with less than 200 employees, while large organisations were defined as those employing more than 200 people. Whereas academia was for those developers who work in the academic sector. The distribution of these participants across small organisations, large organisations, and academia was 14 (58.3%), 9 (37.5%), and 1 (4.2%) respectively. The employed ones came from different well-known organizations, 3 (12.5%) came from Truffle, 2 (8.3%) from Trivago, and 2 (8.3%) from Marvel Wall, while only 1 (2.9%) developer came from each of the other mentioned organizations.

In addition, as for the years of experience of the participants, most were intermediates (with 3–4 years of experience) ($n = 16$, 47.0%), followed by beginners (less than 2 years of experience) ($n = 9$, 26.5%) and experts/seniors (with at least 4 years of experience) ($n = 9$, 26.5%). For the usage period of the approach, the common usage period was 1 day ($n = 17$, 50.0%), followed by 2 days ($n = 11$, 32.4%) and 3 days ($n = 6$, 17.6%). For the country, developers came from 9 different countries. 9 (26.4%) of the participants were from Egypt, 6 (17.6%) were from Saudi Arabia, and 3 (8.8%) were from the United Arab Emirates. Combined with 1 participant from Kuwait, accounting for 2.9% of the sample group, this meant that 18 of the participants were from countries in the Arab world, which was the largest geographic group represented in the sample. In terms of Western countries, 2 (5.8%) of the participants were from Canada, 4 (11.7%) were from the United Kingdom, 3 (8.8%) were from the United States, and 5 (14.7%) were from Germany, amounting to 14 participants in total. Finally, 2 (5.8%) of the participants were from Pakistan, who were the only participants outside the Western or Arab worlds.

Table 7.1 Developers demographic information (N = 34).

Characteristics	Range	Frequency (N)	Percentage (%)
Operating System Type	Android	n=22	64.7 %
	iOS	n=11	32.4 %
	Both	n=1	2.9 %
Job Status	Freelancer	n=10	29.4 %
	Employee	n=24	70.6%
Company size (Number of employees)	Small Organisation (n ≤ 200)	n=14	58.3 %
	Big Organisation (n > 200)	n=9	37.5%
	Academia	n=1	4.2%
Organization Name	App Square	n=1	4.1 %
	Crossover	n=1	4.1 %
	E-bakers	n=1	4.1 %
	Evince Development	n=1	4.1 %
	Ibtikar Technology	n=1	4.1 %
	Marvel Wall	n=2	8.3 %
	Mobile Doctors	n=1	4.1 %
	Mondia Media	n=1	4.1 %
	Samaat	n=1	4.1 %
	Trivago	n=2	8.3 %
	Trufla	n=3	12.5 %
Years of experience	Beginner ≤ 2 years	n=9	26.5 %
	Intermediate 3 - 4 years	n=16	47.0 %
	Expert/Senior > 4 years	n=9	26.5 %
Usage Period	1 Day	n=17	50.0 %
	2 Days	n=11	32.4 %
	3 Days	n=6	17.6 %
Country	Egypt	n=9	26.4 %
	Canada	n=2	5.8 %
	Kuwait	n=1	2.9 %
	Saudi Arabia	n=5	14.7 %
	United Arab Emirates	n=3	8.8 %
	United Kingdom	n=4	11.7 %
	United States	n=3	8.8 %
	Germany	n=5	14.7 %
Pakistan	n=2	5.8 %	

7.3 Descriptive Statistics Results

Table 7.2 provides an overview of the responses obtained from the participants to each of the questions in the CSTE/Q/D questionnaire. For the Task Defining and Distribution (TDD) dimension, which contained 5 questions in total, agreement or strong agreement were the main responses given by the participants. For example, for TDD, most of the participants 47.1% strongly agree about their ability to define and publish testing tasks quickly with less effort. The main response for TDD 2 was a strong agreement of 47.1% on the ability to distribute testing tasks on a large scale in a short time, while for TDD 3, 41.2% of responders agreed on the simplicity of the way to define the whole app as multiple simple tasks effectively.

Table 7.2 Developers' answers to all questions of the CSTE-Q/D questionnaire regarding Effectiveness

Code	Question Summary	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
Task Defining & Distribution						
TDD 1	Less time and effort to define and publish tasks.	(0) 0.0%	(4) 11.8%	(7) 20.6%	(7) 20.6%	(16) 47.1%
TDD 2	Easiness of distributing testing tasks on a large scale.	(0) 0.0%	(1) 2.9%	(6) 17.6%	(11) 32.4%	(16) 47.1%
TDD 3	Usefulness of the task designing method.	(0) 0.0%	(2) 5.9%	(7) 20.6%	(14) 41.2%	(11) 32.4%
TDD 4	Helpfulness of different ways of testing.	(1) 2.9%	(1) 2.9%	(8) 23.5%	(15) 44.1%	(9) 26.5%
TDD 5	Less time and effort to write task requirements.	(0) 0.0%	(0) 0.0%	(6) 17.6%	(13) 38.2%	(15) 44.1%
Testers' Selection						
ACC 1	Easiness and short time of selecting testers.	(0) 0.0%	(3) 8.8%	(8) 23.5%	(8) 23.5%	(15) 44.1%
ACC 2	Helpfulness of providing the list of suggested testers.	(0) 0.0%	(3) 8.8%	(9) 26.5%	(11) 32.4%	(11) 32.4%
ACC 3	Helpfulness of selecting testers with different experience.	(0) 0.0%	(3) 8.8%	(10) 29.4%	(9) 26.5%	(12) 35.3%
ACC 4	Quality of selection criteria to perform effective selection	(0) 0.0%	(4) 11.8%	(6) 17.6%	(12) 35.3%	(12) 35.3%
Privacy & Protection						
PP 1	Ability to control tasks' privacy and confidentiality.	(0) 0.0%	(4) 11.8%	(12) 35.3%	(12) 35.3%	(6) 17.6%
PP 2	Effectivity in protecting the information of testers.	(0) 0.0%	(3) 8.8%	(9) 26.5%	(16) 47.1%	(6) 17.6%
PP 3	Effectivity in controlling non-active/unreliable testers.	(0) 0.0%	(4) 11.8%	(10) 29.4%	(10) 29.4%	(10) 29.4%
Tracking Task Status						
TS 1	Usefulness of tracking the #.of tests and tested devices.	(1) 2.9%	(2) 5.9%	(9) 26.5%	(10) 29.4%	(12) 35.3%
TS 2	Usefulness of tracking the status of published tasks.	(1) 2.9%	(2) 5.9%	(5) 14.7%	(13) 38.2%	(13) 38.2%
Outcomes Diversity						
OD 1	Collect more unexpected issues in the early stages.	(0) 0.0%	(2) 5.9%	(7) 20.6%	(13) 38.2%	(12) 35.3%
OD 2	Different testers' skills help developing high apps quality.	(0) 0.0%	(3) 8.8%	(9) 26.5%	(8) 23.5%	(14) 41.2%
Results Aggregation & Tracking						
RAT 1	Usefulness of reports aggregation and tracking mechanism.	(1) 2.9%	(2) 5.9%	(7) 20.6%	(9) 26.5%	(15) 44.1%
RAT 2	Reducing the time required for collecting and organizing reports.	(1) 2.9%	(3) 8.8%	(9) 26.5%	(12) 35.3%	(9) 26.5%
RAT 3	Easiness of view and filter testg reports.	(0) 0.0%	(2) 5.9%	(10) 29.4%	(11) 32.4%	(11) 32.4%
Results Evaluation & Quality Control						
RQC 1	Less time and effort to assess testers' the performance.	(1) 2.9%	(1) 2.9%	(9) 26.5%	(12) 35.3%	(11) 32.4%
RQC 2	Sufficiency of the evidence provided by testers to gauge results' accuracy.	(0) 0.0%	(2) 5.9%	(11) 32.4%	(13) 38.2%	(8) 23.5%
RQC 3	Measuring results-quality distribution for each task.	(1) 2.9%	(1) 2.9%	(8) 23.5%	(11) 32.4%	(13) 38.2%
RQC 4	Quick and fair evaluation by the evaluation metrics.	(1) 2.9%	(2) 5.9%	(10) 29.4%	(15) 44.1%	(6) 17.6%
Cost Effectiveness (Cost Reduction)						
COS 1	Covering more devices to test at a lower price.	(1) 2.9%	(3) 8.8%	(6) 17.6%	(10) 29.4%	(14) 41.2%
COS 2	Relatively cheaper than other approaches relying on a middleman.	(1) 2.9%	(4) 11.8%	(6) 17.6%	(10) 29.4%	(13) 38.2%
Interaction between Peers (Workflow)						
INT 1	Direct interaction helps understanding testing results.	(2) 5.9%	(1) 2.9%	(9) 26.5%	(10) 29.4%	(12) 35.3%
INT 2	Effectivity of two-way communication mechanism.	(0) 0.0%	(5) 14.7%	(5) 14.7%	(13) 38.2%	(11) 32.4%
INT 3	Ability to reduces delays caused by the middlemen.	(0) 0.0%	(3) 8.8%	(8) 23.5%	(11) 32.4%	(12) 35.3%
Testers Reliability & Trustworthiness						
RT 1	Effectivity of the reliability tracking method.	(2) 5.9%	(3) 8.8%	(7) 20.6%	(16) 47.1%	(6) 17.6%
RT 2	Fairness of the reliability tracking method.	(3) 8.8%	(2) 5.9%	(7) 20.6%	(14) 41.2%	(8) 23.5%
RT 3	Reducing developers' concerns about working with public testers.	(2) 5.9%	(1) 2.9%	(9) 26.5%	(11) 32.4%	(11) 32.4%
Testers Motivations & Incentivisation						
MI 1	Effectivity of the rewarding mechanism.	(1) 2.9%	(4) 11.8%	(8) 23.5%	(15) 44.1%	(6) 17.6%
MI 2	Fairness of the rewarding mechanism.	(1) 2.9%	(4) 11.8%	(7) 20.6%	(18) 52.9%	(4) 11.8%
MI 3	Ability to motivate more testers.	(1) 2.9%	(3) 8.8%	(7) 20.6%	(16) 47.1%	(7) 20.6%
MI 4	Easiness of identified the list of non-paid testers.	(1) 2.9%	(2) 5.9%	(9) 26.5%	(9) 26.5%	(13) 38.2%
Knowledge Sharing (Wiki)						
KS 1	Effectivity of knowledge-sharing environment.	(1) 2.9%	(1) 2.9%	(11) 32.4%	(10) 29.4%	(11) 32.4%
KS 1	Effectivity of documentation mechanism.	(3) 8.8%	(1) 2.9%	(9) 26.5%	(13) 38.2%	(8) 23.5%
KS 1	Usefulness of documenting issues and its solutions.	(1) 2.9%	(3) 8.8%	(6) 17.6%	(13) 38.2%	(11) 32.4%
Ease of use						
EU 1	In daily work routine and industry practices.	(2) 5.9%	(1) 2.9%	(10) 29.4%	(10) 29.4%	(11) 32.4%
EU 2	For both Android and iOS specialists.	(0) 0.0%	(2) 5.9%	(6) 17.6%	(13) 38.2%	(13) 38.2%

For TDD 4, 44.1% agreed that testing the features and/or piece of code help to perform more effective compatibility testing. In the last question TDD 5, 44.1% of participants agreed that the task definition and announcement form enables them to write tasks' requirements with less time and effort. Low levels of strong disagreement are also identified across the 5 questions, with the highest levels of strong disagreement being associated with two questions, TDD 1 with 11.8% and TDD 4 with 8.8%. For the 4 questions involved in the Accessing and Selection of Testers (ACC) dimension, strong agreement or agreement was significantly more common than the disagreement, or strong disagreement. For example, for ACC 1, the greatest number of respondents strongly agreed 44.1% on that the method of accessing and selecting testers is easy and quick, while 32.4% strongly agreed or agreed that the suggestion of a list of eligible testers helps facilitate testers' selection (ACC 2). For ACC 3, 35.3% of respondents strongly agreed on the selecting testers with different skills and expertise areas to participate and work on a specific task is helpful. Regarding ACC 4, a high percentage of participants agreed or were neutral (35.3% in each case) that the importance of the selection criteria and the information provided on the history of testers' performance in selecting and inviting several testers with specialized skills to work on tasks. For the first three questions, the disagreement level was low at 8.8%, while it was 11.8% in the last question.

For the Privacy and Protection (PP) dimension, which consisted of 3 questions, the most popular response categories for all 3 questions, were agreement and then neutrality. For example, for PP 1, 35.3% of participants agreed or remained neutral about their ability to control tasks' privacy and confidentiality and prevent unauthorized testers from accessing it using the proposed approach. For PP 2, most of the participants 47.1% agreed that the proposed method can protect sensitive information of testers (e.g., their work quality and reliability level) among each other, compared to 11.8% who neutral. For PP 3, the participants' responses about effectiveness of the approach in protecting the testing environment from such having non-active and unreliable testers were similar 29.4% in terms of neutrality, agreement, and strong agreement. The participants' agreement and disagreement levels in total were comparable for the Tracking Tasks Status (TS) dimension, which contained only 2 questions. For example, for TS 1, more than half of participants 64.7% agreed that it was useful to track the number of tests and the type of tested devices for each task, compared to 8.8% who disagreed. In terms of TS 2, the majority of participants 76.4% agreed that tracking the status of published tasks during the active test cycle is useful, while 8.8% disagreed.

In terms of Outcomes Diversity (OD), which was the fifth dimension of interest in the CSTE-Q/D questionnaire, for OD 1, most of the participants agreed or strongly agreed (38.2% or 35.3%, respectively) on the ability to gather more unexpected issues in the early stages of the apps development process through our approach. Additionally, for OD 2, most of the participants 41.2% strongly agreed about the usefulness of participation of testers with different skills in discovering more issues and aid in developing high-quality apps. Regarding

the 3 questions for the results aggregation and tracking (RAT) dimension, the levels of strong disagreement and disagreement were low, similar to the previous dimensions. Specifically, for RAT 1, 44.1% strongly agreed with the effectively and usefulness of the reports aggregation and tracking mechanism in general (compared to 2.9% who strongly disagreed), while 35.3% in RAT 2 agreed about the ability of automated tracking mechanism in reducing the time required for collecting and organizing reports (compared to 26.5% who strongly agreed). Finally, for RAT 3, the same percentage of participants 32.4% agreed or strongly agreed on easiness of view, organize, filter collected reports. It is notable that the levels of disagreement with RAT 1, RAT 2, and RAT 3 were 8.8%, 11.8%, and 5.9%, respectively. For the Results Evaluation and Quality Control (RQC) dimension, which consisted of 4 questions, levels of disagreement are generally very low. Regarding the ability of developers to assess the performance of testers in less time and effort (RQC 1), the participants' responses were close in terms of agreement and strong agreement (35.3%, 32.4% respectively), while for neutrality was 26.5%. For RQC 2, the higher percentage 38.2% was for agreement in terms of that the evidence provided by testers sufficient to gauge the accuracy of the results, while was also high for neutrality 32.4%. More than half of the participants provided their agreement (32.4% agreed and 38.2% strongly agreed) that the insight provided into the results quality distribution helps understand the reason for low quality work (RQC 3). For RQC 4, the higher percentage of participants' responses about the ability to perform a quick and fair evaluation of testers' works by involved quality evaluation metrics, were the neutrality with 29.4% and agreement with 44.1%.

As for the 2 questions in Cost Effectiveness (COS) dimension, the percentage of all response categories were very similar. 70.6% of participants agreed with the possibility of covering more devices at a lower price through our approach (COS 1), while 67.6% agreed that the approach is relatively cheaper than other approaches relying on a middleman. On the dimension of Interaction between Peers (INT), which consisted of 3 questions, the most popular responses in general about the importance of direct interaction between developers and testers were strong agreement or agreement. For the statement on understanding more testing results (INT 1) was strong agreement at 35.3%, for the effectiveness of overall communication mechanism (INT 2) was agreement 38.2%, while regarding reducing delays caused by the middlemen (INT 3) were strong agreement 35.3%. For the dimension of Testers' Reliability and Trustworthiness (RT), which consisted of 3 questions, the most response category was agreement. For RT 1, Most of the participants 47.1% agreed with the that the reliability tracking method is effective, while only 5.9% and 8.8% disagreed or strongly disagreed, respectively. For RT 2, 41.2% of the participants agreed that the reliability tracking method was fair, and 32.4% strongly agreed or agreed that it reduced developers' concerns regarding working with public testers (RT 3). As for Testers' Motivations and Incentivisation (MI), most of the participants agreed that the reward mechanism was effective (MI 1), fair (MI 2), and motivated (MI 3) (44.1%, 52.9% and 47.1%, respectively).

For MI 4, 38.2% strongly agreed with the easiness of identifying the list of unpaid testers. As for Knowledge Sharing (KS), more participants agreed with than disagreed with the 3 questions, but the levels of neutrality were also substantial. For example, for KS 1, 32.4% of the participants strongly agreed about the effectiveness of the knowledge sharing environment between worldwide developers' and testers' communities, and 26.5% were neutral. For KS 2, 38.2% of the participants agreed about the effectiveness of the documentation mechanism. The same percentage 38.2% also agreed in the usefulness of documenting issues and their solutions in helping them to answer complicated questions and develop high-quality apps in the future (KS 3). In terms of Ease of Use (EU), which consisted of only 2 questions, the ease of use associated with daily working routines and industry practices (EU 1) found strong agreement in 32.4% of the sample. For EU 2, the majority of the participants were either in strong agreement or agreement (38.2% in both cases) about the ease of use of the approach for both android and iOS developers.

Table 7.3 shows that for the Benefits (Advantage) (BNF) dimension, which consisted of 8 questions, it was generally the case that more participants agreed with all BNF 1-8 with the exception of BNF 7, which was strongly agreed, compared to those who disagreed. For example, 52.9% agreed that it improves app development by incorporating target users early in the development process (BNF 1), as opposed to 8.8% who disagreed. Similarly, for BNF 2, 41.2% agreed on the opportunity to receive help from other worldwide developers communities, which was substantially greater than the 5.9% who disagreed. For the benefits related to gain more knowledge, 38.2% agreed that it helps developers to stay informed about new issues of different architectures of device models in the future (BNF 3). In comparison, 47.1% agreed that there are opportunities to cover all different versions of mobile devices and OS quickly through our approach (BNF 4). For BNF 5, 38.2% agreed that the approach increases work collaboration and communication among developers and researchers in the domain, while a higher percentage 47.1% agreed with the statement that the approach helps increase test coverage and get better results in a shorter time (BNF 6), compared to 8.8% who disagreed. For the benefits related to the development of mobile apps, 35.3% strongly agreed that the approach accelerates the development process and time-to-market delivery (BNF 7). For BNF 8, 38.2% agreed or strongly agreed that the approach helps obtain important information about the users' behaviours and interactions with the app, where none of the participants were disagreed with that.

Finally, for the Satisfaction (SAT) dimension, almost the same number of participants agreed with the 3 questions. For example, the agreement that the approach would be used to achieve a test in the future (SAT 1) was 67.6% in total. For SAT 2, 64.7% agreed with the statement that they are satisfied with crowdtesting workflow and service provided, while 55.8% agreed with the benefits gain from the approach (SAT 3). Similar to other dimensions, the level of disagreements for all 3 questions were low.

Table 7.3 Developers' answers to all questions of the CSTE-Q/D questionnaire regarding Benefits and Satisfaction

Code	Question Summary	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
Benefits (Advantage)						
BNF 1	Improves apps development by incorporating target users early.	(2) 5.9%	(1) 2.9%	(8) 23.5%	(18) 52.9%	(5) 14.7%
BNF 2	Opportunity to receive help from other developers.	(2) 5.9%	(0) 0.0%	(9) 26.5%	(14) 41.2%	(9) 26.5%
BNF 3	Staying informed about new issues and different architectures of devices.	(0) 0.0%	(1) 2.9%	(8) 23.5%	(13) 38.2%	(12) 35.3%
BNF 4	Cover all versions of mobile devices and OS quickly.	(0) 0.0%	(0) 0.0%	(9) 26.5%	(16) 47.1%	(9) 26.5%
BNF 5	Large-scale work collaboration and communication.	(1) 2.9%	(2) 5.9%	(10) 29.4%	(13) 38.2%	(8) 23.5%
BNF 6	Increase test coverage and get better results in a shorter time.	(0) 0.0%	(3) 8.8%	(7) 20.6%	(16) 47.1%	(8) 23.5%
BNF 7	Accelerates the development process and time-to-market delivery.	(1) 2.9%	(4) 11.8%	(7) 20.6%	(9) 26.5%	(12) 35.3%
BNF 8	Users' behaviours and interactions with apps.	(0) 0.0%	(0) 0.0%	(8) 23.5%	(13) 38.2%	(13) 38.2%
Satisfaction						
SAT 1	Use of the approach to achieve a test in the future.	(0) 0.0%	(4) 11.8%	(7) 20.6%	(13) 38.2%	(10) 29.4%
SAT 2	Nature of the crowdtesting workflow and service provided.	(0) 0.0%	(3) 8.8%	(9) 26.5%	(14) 41.2%	(8) 23.5%
SAT 3	Benefits received from this crowdtesting approach.	(3) 8.8%	(2) 5.9%	(10) 29.4%	(12) 35.3%	(7) 20.6%

7.4 Results I: Effectiveness

This section describes the main findings and results related to the evaluation of effectiveness of the approach. It begins with showing how effective the overall approach, main processes, and its features are in performing effective compatibility testing from developers' perspectives. Further, it shows whether the direct interaction between developers and testers, knowledge sharing, and diversity of testers' experiences affect the overall effectiveness of the approach or its processes.

7.4.1 Overall Effectiveness of the Approach

Table 7.4 demonstrates the mean score and the level for the overall effectiveness of the approach and its main processes (sub-dimensions), while Figure 7.1 presents the effectiveness percentages of each process according to detected mean scores. The data presented in Table 7.4 shows that the overall effectiveness of the approach has reached a high level at ($M=3.8735$, $SD=0.79176$) and percentage of 77.4%. It has also shown that all the approach processes have achieved a high effectiveness level of the mean score is above 3.67 (73.4%), which falls in the high category of effectiveness. Results gained have demonstrated that the method of publishing and distributing the test on a large-scale (TDD) recorded the highest overall mean score ($M=4.0824$, $SD=0.78797$) with an effectiveness percentage of 81.6%, followed by the ability to obtain different results and solutions to the complicated issues (OD) with a mean score ($M=4.0824$, $SD=0.84387$) and percentage of 80.0%. The privacy and protection (PP) and the process of classifying reliable testers (RT) scored the lowest mean

score ($M=3.6961$, $SD < 1.0$) with 73.8%. The results also show that the approach has proven its effectiveness in reducing costs (COS) with a high mean score ($M=3.9265$, $SD=1.08804$) and high effectiveness percentage at 78.4%. Further, the results also demonstrate the effectiveness of interaction and workflow between developers and testers (INT) at a high mean ($M=3.8922$, $SD=0.87068$) and effectiveness percentage at 77.8%.

Table 7.4 The mean score and standard deviation values of participant developers according to the "Overall Effectiveness".

Dimension/ Sub-dimension	Mean	Std. Deviation	Level
Overall Effectiveness	3.8735	.79176	High
Task Defining and Distribution (TDD)	4.0824	.78797	High
Accessing Specialized Testers Skills (ACC)	3.9338	.92791	High
Privacy and Protection (PP)	3.6961	.83431	High
Tracking Task Status (TS)	3.9559	1.00278	High
Outcomes Diversity (OD)	4.0000	.84387	High
Results Aggregation and Tracking (RAT)	3.8922	.92740	High
Results Evaluation and Quality Control (RQC)	3.8456	.87259	High
Cost Effectiveness (COS)	3.9265	1.08804	High
Interaction between Peers and Workflow (INT)	3.8922	.88276	High
Testers Reliability and Trustworthiness (RT)	3.6961	.97913	High
Testers Motivations and Incentivisation (MI)	3.7132	.87068	High
Knowledge Sharing (KS)	3.7941	1.01176	High
Ease of use (EU)	3.9412	.91092	High

Mean score's interpretation: (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.68 – 5.00).

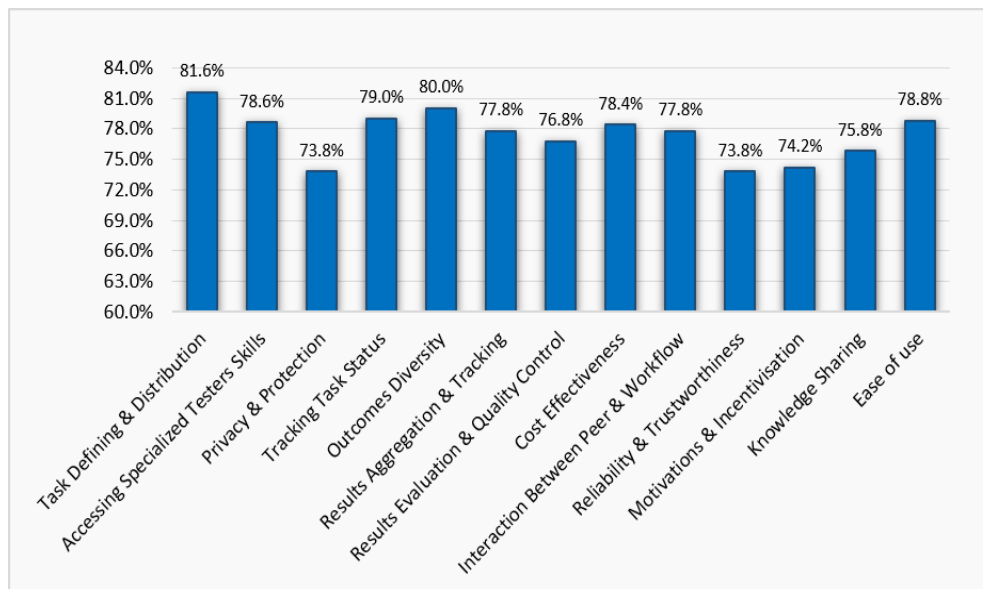


Fig. 7.1 The effectiveness percentage for each process in the approach, which achieves the overall effectiveness percentage (77.4%).

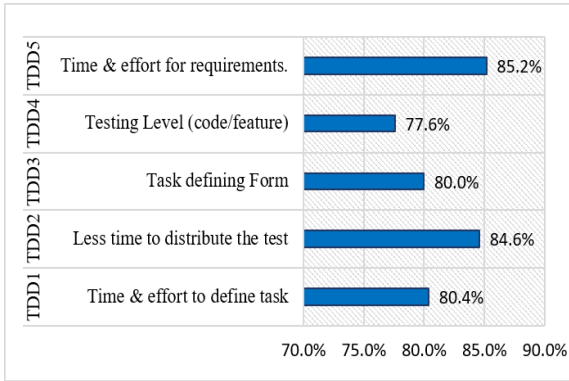
7.4.2 Effectiveness of the each processes in the Approach

Table 7.5 present the effectiveness of each question (feature) within sub-dimensions through the mean scores while Figure 7.2 present the effectiveness in percentages. Overall, the majority of the questions of all sub-dimension classified as a high level of effectiveness where only seven questions being classified with moderate effectiveness.

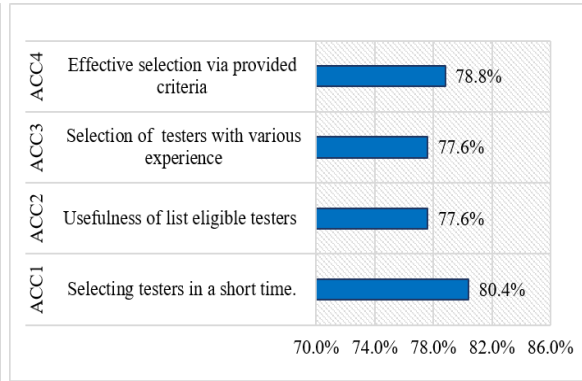
Table 7.5 Summary of the mean score analysis of each question (feature) within sub-dimensions of effectiveness.

Sub-dimension	Question Code	Mean (M)	Std. Deviation (SD.)	Level
Task Defining and Distribution	TDD1	4.0294	1.08670	High
	TDD2	4.2353	.85489	High
	TDD3	4.0000	.88763	High
	TDD4	3.8824	.94595	High
	TDD5	4.2647	.75111	High
Accessing Specialized Testers Skills	ACC1	4.0294	1.02942	High
	ACC2	3.8824	.97746	High
	ACC3	3.8824	1.00799	High
	ACC4	3.9412	1.01328	High
Privacy and Protection	PP1	3.5882	.92499	Moderate
	PP2	3.7353	.86371	High
	PP3	3.7647	1.01679	High
Tracking Task Status	TS1	3.8824	1.06642	High
	TS2	4.0294	1.02942	High
Outcomes Diversity	OD1	4.0294	.90404	High
	OD2	3.9706	1.02942	High
Results Aggregation & Tracking	RAT1	4.0294	1.08670	High
	RAT2	3.7353	1.05339	High
	RAT3	3.9118	.93315	High
Results Evaluation & Quality Control	RQC1	3.9118	.99598	High
	RQC2	3.7941	.88006	High
	RQC3	4.0000	1.01504	High
	RQC4	3.6765	.94454	Moderate
Cost Effectiveness	COS1	3.9706	1.11424	High
	COS2	3.8824	1.14851	High
Interaction between Peers & Workflow	INT1	3.8529	1.13170	High
	INT2	3.8824	1.03762	High
	INT3	3.9412	.98292	High
Reliability and Trustworthiness	RT1	3.6176	1.07350	Moderate
	RT2	3.6471	1.17763	Moderate
	RT3	3.8235	1.11384	High
Motivations and Incentivisation	MI1	3.6176	1.01548	Moderate
	MI2	3.5882	.95719	Moderate
	MI3	3.7353	.99419	High
	MI4	3.9118	1.08342	High
Knowledge Sharing	KS1	3.8529	1.01898	High
	KS2	3.6471	1.15161	Moderate
	KS3	3.8824	1.06642	High
Ease of use	EU1	3.7941	1.12221	High
	EU2	4.0882	.90009	High

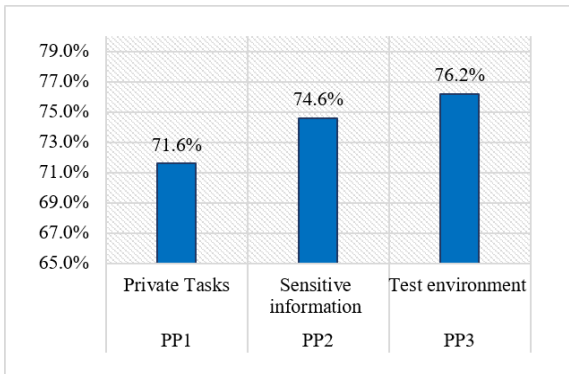
Mean score's interpretation: (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.68 – 5.00).



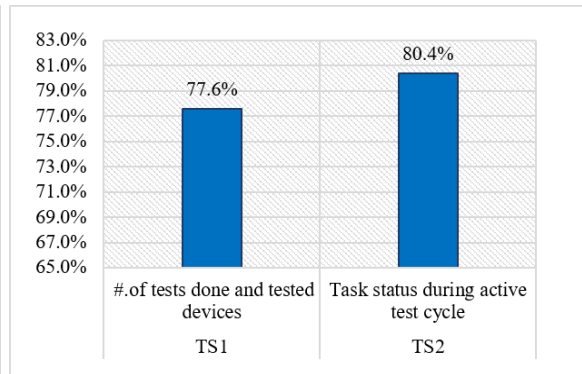
(a) Task Defining and Distribution (81.6%)



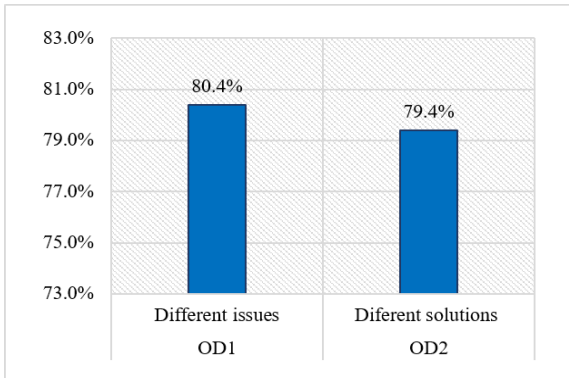
(b) Accessing Specialized Testers (Testers Selection)(78.6%)



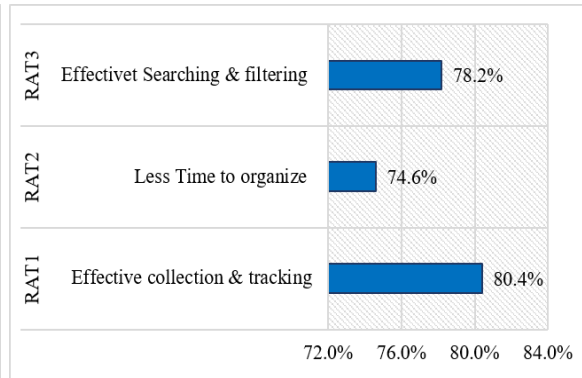
(c) Privacy and Protection (73.8%)



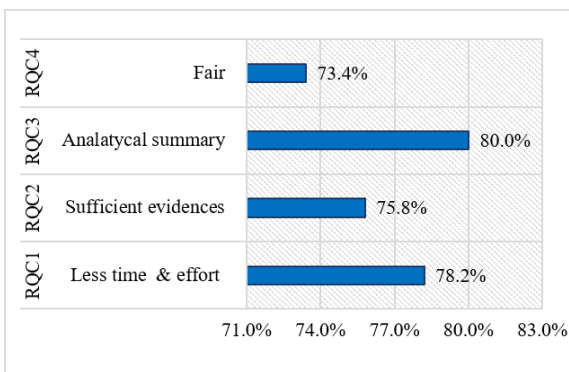
(d) Tracking Task Status (79%)



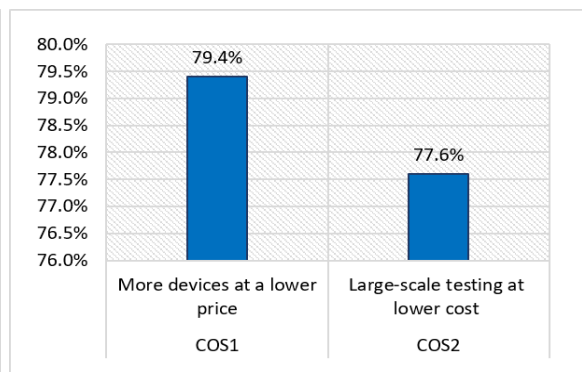
(e) Outcomes Diversity (80%)



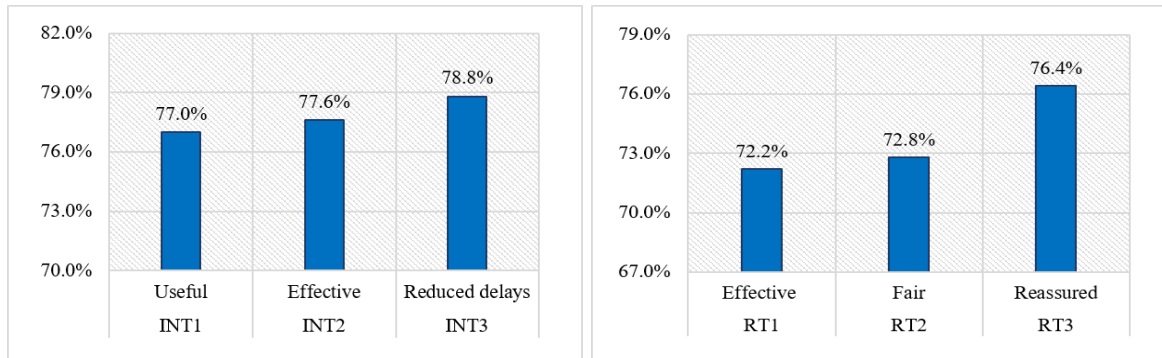
(f) Results Aggregation and Tracking (77.8%)



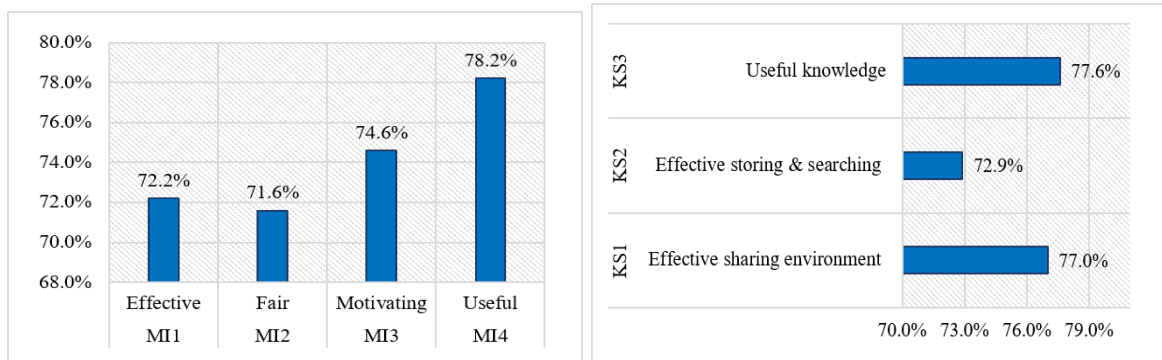
(g) Results Evaluation and Quality Control (76.8%)



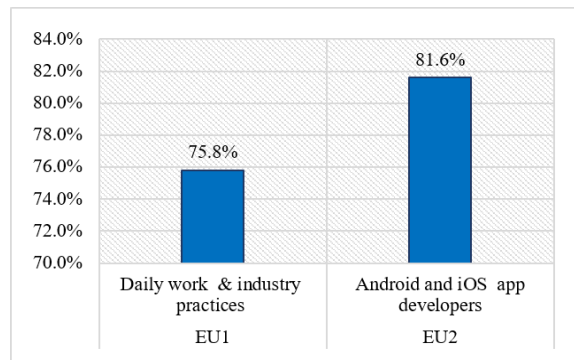
(h) Cost Effectiveness (78.4%)



(i) Interaction between Peers and Workflow (77.8%) (j) Testers Reliability and Trustworthiness (73.8%)



(k) Testers Motivations and Incentivisation (74.2%) (l) Knowledge Sharing (Wiki) (75.8%)



(m) Ease of use (78.8%)

Fig. 7.2 Effectiveness percentage for each characteristic within all processes (sub-dimensions) of the proposed approach

The first sub-dimension (process) contained five questions; all interpreted to have a high level of effectiveness. The ability to define and publish testing tasks quickly with less effort (TDD1) was effective with a high mean score ($M=4.02941$, $SD=0.08670$) and with a percentage of 80.4%. The method of distributing the testing on a large-scale (TDD2) was fast with a very high mean score ($M=4.2353$, $SD=0.85489$) at 84%. The task design method of defining the whole app as simple multiple tasks (TDD 3) was easy and effective with a

percentage of 80% and mean score ($M=4.0000$, $SD=0.88763$). The two different levels of testing (feature and code) (TDD 4) was also effective to perform more effective compatibility testing tests with a mean score ($M=3.8824$, $SD=0.94595$) and percentage of 77.6%. The method of defining the task and writing the requirements (TDD 5) was very effective in terms of time required and effort with a mean score ($M=4.2647$, $SD=0.75111$) and with a very high percentage of 85.2%.

The subsequent sub-dimension was Accessing Specialised Testers Skills. The method of selecting testers in a short time (ACC 1) was easy and effective, with a high mean score ($M=4.02941$, $SD=0.02942$) and at 80.4%. The method of suggesting a list of qualified testers for performing a specific task (ACC 2), and selection of testers with different skills and expertise areas to participate and work on specific tasks (ACC 3), was helpful with the same mean score ($M=3.8824$, $SD<1.000$) at 77.6%. The insights provided into testers' performance and defined selection criteria were helpful to select and invite several testers with specialized skills to work on tasks (ACC 4) with a mean score ($M=3.94121$, $SD=0.01328$) at 78.8%. For the Privacy and Protection sub-dimension, one question (PP 2) was in the moderate category. The ability to control the tasks' privacy and prevent unauthorized testers from accessing it (PP 1) was moderately effective with a mean score ($M=3.5882$, $SD=0.92499$) at 71.6%. The effectiveness regarding protecting the sensitive information of testers (e.g., their work quality and reliability level) among each other (PP 2) was high with a mean ($M=3.7353$, $SD=0.86371$) at 74.6%. Whereas, the effectiveness of protecting the testing environment from having non-active and unreliable testers (PP 2) was effective with a mean score ($M=3.76471$, $SD=0.01679$) and at 76.2%.

Subsequently, the Tracking Task Status sub-dimension has two questions ranked in the high-level. Tracking the number of tests performed and the type of devices used for testing a specific task (TS 1) was helpful with a percentage of 77.6% and mean score ($M=3.88241$, $SD=0.06642$). While, tracking the task status of the published task during the active test cycle (TS 2) was more useful with a higher percentage 80.4% and mean score ($M=4.02941$, $SD=0.02942$). Further, the Outcomes diversity sub-dimension contained two questions. The ability to discover different unexpected issues in the early stages of an app's development process through our approach (OD 1) was effective with a mean score ($M=4.0294$, $SD=0.90404$) at 80.4%. The results obtained by different testers' backgrounds was helpful in representing different ways of thinking about implementing the app's functionalities (OD 2) with 79.4% and a mean value ($M=3.97061$, $SD=0.02942$).

For the Results Aggregations and Tracking, the reports aggregation and tracking mechanism (RAT 1) was effective with a high mean score ($M=4.02941$, $SD=1.08670$) at 80.4%. The automated tracking mechanism was also helpful in reducing the time required to collect and organize test reports (RAT 2) with a mean score ($M=3.73531$, $SD=1.05339$) at 74.6%. The ability to view and filter reports easily (RAT 3) was effective with a mean score ($M=3.9118$, $SD=0.93315$) at 78.2%. For the Results Evaluation and Quality Control, three questions

are classified at the High level and just one question at the moderate level. The ability to assess testers' performance in less time and effort (RQC 1) was effective with 78.2% and mean score ($M=3.9118$, $SD=0.99598$). The evidence provided by public testers regarding testing results was sufficient at 75.8% and with mean score ($M=3.7941$, $SD=0.88006$) to gauge their accuracy (RQC 2). The insight provided into the results quality distribution was helpful to understand the reason for low quality work (RQC 3) with a high mean score ($M=4.00001$, $SD=0.01504$) and at 80.0%. The quality evaluation metrics' effectiveness to perform a quick and fair evaluation of testers' works (RQC 4) was on the border of high and moderate effectiveness with a mean score ($M=3.6765$, $SD=0.94454$) at 73.4%.

Both questions of the Cost Reduction (Cost Effectiveness) were at the high level. The ability to cover more devices to test our tasks at a lower price through this approach (COS 1) was more effective at 79.4% and a mean ($M=3.9706$, $SD=1.11424$). Regarding its cost reduction compared to other approaches relying on the middleman (COS 2) was effective at 77.6% with a mean score ($M=3.8824$, $SD=1.14851$). For Interaction between Peers and workflow, all three questions fall into the high category. The effectiveness of developers and testers' direct interaction without middlemen in understanding more the test results (INT 1) was high with a mean score ($M=3.8529$, $SD=1.13170$) and a percentage of 77%. The two-way communication provided by this approach's workflow (INT 2) was effective with a percentage of 77.6% and a mean ($M=3.8824$, $SD=1.03762$). The ability of the direct interaction in reducing the delays caused by the middlemen figures alike (managers or leaders) compared to other approaches (INT 3) was highly effective with a mean value ($M=3.9412$, $SD=0.98292$) at 78.8%.

For the Reliability and Trustworthiness, two questions were at a moderate level and one at the high level. The reliability tracking method was effective (RT 2) at 72.2% with a moderate mean ($M=3.6176$, $SD=1.07350$) and fair (RT 2) at 72.8% with also a moderate mean ($M=3.6471$, $SD=1.17763$). The classification method's effectiveness in reducing developers' concerns about working with public testers (RT 3) was high, with a mean score ($M=3.8235$, $SD=1.11384$) and a percentage of 76.4%. The Motivations and Incentivisation sub-dimension was evenly made up of two moderate effective questions and two highly effective questions. The rewarding mechanism was effective (MI 1) with a moderate mean value ($M=3.6176$, $SD=1.01548$) and percentage of 72.2%, was fair (MI 2) with also a moderate mean ($M=3.5882$, $SD=0.95719$) and percentage 71.6%, and was motivating (MI 3) with slightly high mean value ($M=3.7353$, $SD=0.99419$) at 74.6%. The effectiveness in efficiently controlling and tracking the non-paid testers' status (MI4) was high with 78.2% and at mean value ($M=3.9118$, $SD=1.08342$).

For Knowledge Sharing, the knowledge-sharing environment between worldwide developers' and testers' communities (KS 1) was effective with a mean score ($M=3.8529$, $SD=1.01898$) and a percentage of 77%. The documentation and knowledge searching mechanism (KS 2) effectiveness was moderate with a mean value ($M=3.6471$, $SD=1.15161$) at 72.9%. The

documentation of issues and their solutions was helpful in answering the developers' questions and developing high-quality apps (KS 3) with 77.6% and a mean ($M=3.8824$, $SD=1.06642$). For the final sub-dimension Ease of Use, both questions fell into the high level. The easiness of using the proposed approach in daily work routine and industry practices (EU 1) was high with 75.8% and a mean of ($M=3.7941$, $SD=1.12221$). The ease of use and interaction with the approach for both Android and iOS specialists with the (EU 2) was very high with 81.6% and mean value ($M=4.0882$, $SD=0.90009$).

7.5 Results II: Benefits/Advantages

This section describes the results and main findings related to the benefits of the approach in performing compatibility testing compared to other approaches, from developers' perspectives. Moreover, it shows whether the approach benefits a certain group of developers than others.

7.5.1 Benefits of the Approach

Overall, the mean analysis results demonstrate that the approach is 77.4% beneficial to all developers worldwide for compatibility testing with a high mean score ($M=3.8787$, $SD=0.62195$). The developers' responses presented in Table 7.6 highlight the three most significant benefits that can be obtained from using the approach: The ability to gain useful information regarding the users' behaviours or interactions with the app (BNF 8), the gain of knowledge regarding the new issues and different internal architectures of mobile device models (BNF 3), and the facilities to test all versions of mobile devices and OS quickly (BNF4). The obtaining of information regarding behaviours and interactions ranked first with a mean score ($M=4.1471$, $SD=0.78363$) and at 82.8%. The acquiring of knowledge regarding new issues and internal architectures of mobile followed with a mean score ($M=4.0588$, $SD=0.85071$) and at 81.0%. The facilities to test all versions of devices and OS was deemed least significant out of the three with ($M=4.0000$, $SD=0.73855$) and at 80.0%, which is also in the high level. In addition, receiving help from other developers' communities (BNF 2) and increasing test coverage in a shorter time (BNF 6) ranked second in terms of important benefits, with relatively close percentages at 77.0% and 76.4%, respectively, and a mean score above $M=3.7000$ and $SD < 1.0$ for BNF 6. Furthermore, the developers' responses also show that using the approach will aid in the communication and collaboration with other developers and researchers on a large-scale (BNF 5), and will accelerate the development process and time to deliver the app to the market in less cost (BNF 6), with the same mean value ($M=3.7353$, $SD < 1.0$) and the percentage of 74.6%. The approach's proficiency in improving and facilitating the development of mobile apps by incorporating target users early in the development process (BNF1), recorded the lowest mean score ($M=3.6765$, $SD=0.97610$) at 73.4%.

Table 7.6 The mean score and standard deviation values of participant developers according to the "Benefit//advantage"

Dimension	Question Code	Mean	Std. Deviation	Level	Percentage (%)
Benefits/Advantages	(BNF)	3.8787	.62195	High	77.4%
	BNF 1	3.6765	.97610	High	73.4%
	BNF 2	3.8235	1.02899	High	76.4%
	BNF 3	4.0588	.85071	High	81.0%
	BNF 4	4.0000	.73855	High	80.0%
	BNF 5	3.7353	.99419	High	74.6%
	BNF 6	3.8529	.89213	High	77.0%
	BNF 7	3.7353	1.23849	High	74.6%
	BNF 8	4.1471	.78363	High	82.8%

Mean score's interpretation: (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.68 – 5.00).

7.5.2 The Difference Between Developers Groups Regarding Benefits

Table 7.7 summarizes the results of the independent t-test analysis to assess the benefit of the approach for the different developers' groups, which presents the mean, standard deviation, degrees of freedom (df), difference value (t-value), and statistical significance value of the differences (sig.) between these groups. Figure 7.3 represents the converted percentages for the benefit mean score for each group. According to the results presented in Table 7.7, no difference in the mean scores between Android and iOS developers, as the percentages are very close. This means that both groups have found that the approach is beneficial with the same level. The independent t-test also demonstrated a big statistical difference between the mean value of the Employees (M=3.7011, SD=0.64814) and Freelancers (M=4.2750, SD=0.36705) at t-value = -3.222 and p-value = 0.003 < 0.01. This shows that the Freelancers developers have more benefited from the approach at 85.5% than Employees with 74.0%. The results also show a big difference in the mean scores between employed developers at Big Organisations (M=3.2917, SD=0.54127) and Small Organisations (M=3.9667, SD=0.56194). This difference is statistically significant with a t-value = -2.887 and p-value = 0.009 < 0.01. This shows that the developers employed in small organisations benefited more from the approach with a higher percentage of 79.3% than big organisations at 65.8%. In summary, we found that all developers groups have benefited from the proposed approach, with a high percentage of $\geq 74.0\%$. Only big organisations developers have moderately benefited with 65.8%.

7.5.3 Received Benefits by Each Group

The previous section explained the differences between the different developers' groups in terms of whether there is a statistically significant difference or not. This section will illustrate the most important benefits gained from the approach by each group separately by describing the percentages and mean scores of each group responses.

Table 7.7 Summary of the independent samples t-test results (regarding benefits)

Variable	Group	N	Mean	St. Dev	t-value	df	Sig.
Operating System	Android	22	3.8409	.65144	-.474	32	.639
	iOS	12	3.9479	.58499			
Job Status	Employee	24	3.7011	.64814	-3.222	32	.003
	Freelancer	10	4.2750	.36705			
Workplace Size	Big Organizations	9	3.2917	.54127	-2.887	22	.009
	Small Organizations	15	3.9667	.56194			

The difference is significant at the $\alpha \leq 0.01$ level (2-tailed)

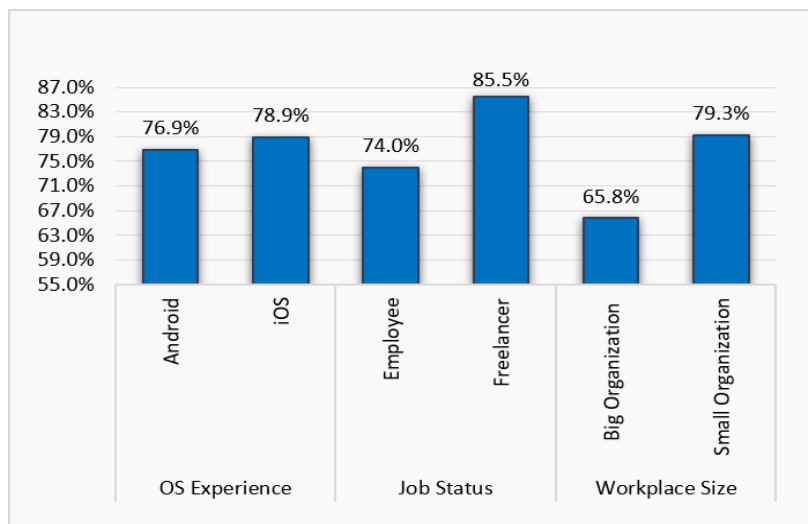


Fig. 7.3 Comparison between different developers' groups regarding benefit/advantage of the approach

For Android Developers: According to the responses collected from Android developers presented in Figure 7.4, the most useful can be seen to be the knowledge about the unexpected issues produced from users' behaviours and interactions with apps (BNF 8), which ranks first among android developers as the most crucial benefit gained from the approach with a mean score of ($M=4.1364$, $SD=0.8335$) at 82.6%. The following benefits, ability to cover all versions of mobile device models and OS quickly (BNF 4) ranked the second with a mean score of ($M=4.0000$, $SD=0.69007$) with 80.0%, and the ability to stay informed about new issues with the development of apps and different architectures of device models in the future (BNF 4) ranked the third with a mean value ($M=3.9545$, $SD=0.89853$) at 79.0%. On the contrary, the lowest scoring benefits were large-scale work collaborations and communication with other developers (BNF 5), receiving help from other developer communities (BNF2),

and facilitating apps development process (BN1). Developers considered BNF 2 and BNF 5 to have a similar level of benefit, with scores of (M=3.6818, SD=0.94548) at 73.6% and (M=3.6818, SD=1.04135) at 73.6%, respectively. The approach capability to facilitate the app development process (BN1) was considered least beneficial with a mean score of (M=3.5455, SD=0.85786) at 70.8%.

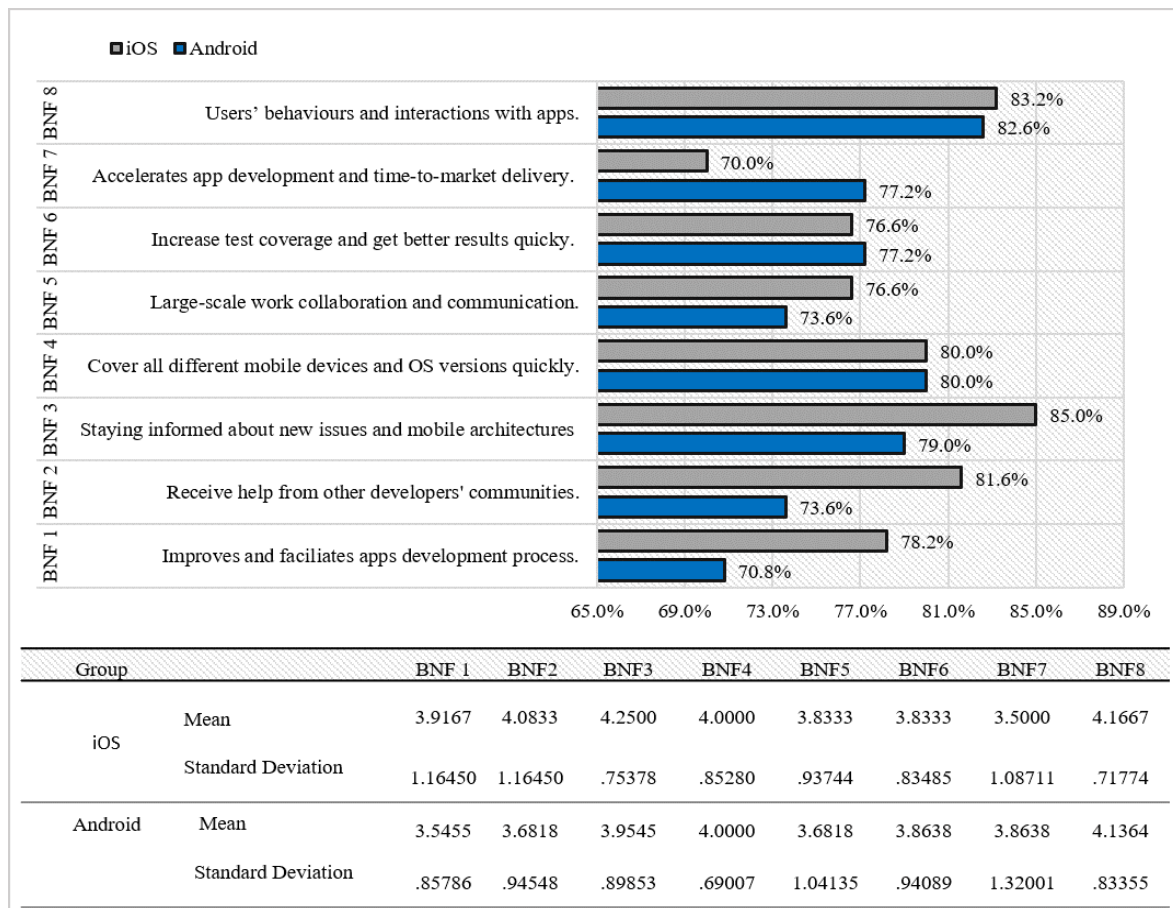


Fig. 7.4 The percentages and mean scores for the most important benefits received by Android and iOS developers.

For iOS Developers: Figure 7.4 also describes the responses gathered from iOS developers by mean scores and percentages. The most benefit obtained by the iOS developers was the possibility of the approach keeping iOS developers informed about new issues and mobile architecture of different device models or OS versions (BNF 3), with a mean score of (M=4.2500, SD=0.75378) at 85.0%. The developers ranked the ability to obtain more useful information about issues that occur by different user behaviour and interactions with the app or mobile devices (BNF 8) as the second gained benefit, holding a mean value (M=4.1667, SD=0.71774) at 83.2%. On the opposing side, the developers considered the approach capabilities to accelerate app development and time-to-market delivery (BNF 7) as

the last benefit they may gain, with the lowest mean score ($M=3.5000$, $SD=1.08711$) and percentage of 70.0%. The developers also considered the increase of the test coverage (BNF 5) and large-scale communication/collaboration with other developers (BNF 6) as one of the lowest gained benefits after (BNF 7) with succeeding mean scores of ($M=3.8333$) at 76.6%, with SDs of 0.93744 and 0.8333 respectively.

For Employees Developers: The employee developers' results are presented in Figure 7.5, highlighting the most important benefits gained from using the proposed approach. Here, employees developers believe the proposed approach is very beneficial for staying informed about new issues and different mobile architectures (BNF 3), with the highest mean score of ($M=3.9565$, $SD=0.87792$) and a percentage of 79.1%. Furthermore, the only benefits ranking close to BNF 3 were BNF 4 and BNF 8. The approach ability to cover all different mobile devices and OS versions quickly (BNF 4), as well as "gaining more information about issues related to user behaviour and interactions (BNF 8), scored the second highest mean values ($M=3.9130$, $SD=0.73318$) and ($M=3.9130$, $SD=0.79275$), respectively, with same percentage of 78.2%. From the remaining benefits, no responses yielded a percentage higher than 73.0% or mean value above $M=3.6522$. From these, the possibility of the approach accelerating app development and time-to-market delivery ranked lowest with a score of ($M=3.3913$, $SD=1.30530$) at 67.8%.

For Freelancers Developers: Figure 7.5 also presents the responses collected from freelance developers in terms of the received benefits. According to the results, no response yielded a scored lower than 78.0% or a mean value less than 3.9000 for all listed benefits. The least significant benefit gained from using the approach was the large-scale collaboration and communication with other developers and researchers in the field (BNF 5), with a mean score of ($M=3.9000$, $SD=0.56765$) at 78.0%. Subsequently, the facilities to cover all different mobile devices and OS versions quickly (BNF 4) was deemed relatively less beneficial with the next highest mean value ($M= 4.1000$, $SD=0.73786$) with 82.0%. Conversely, obtaining information about the unexpected issues produced from different users' behaviours and interactions with apps and devices (BNF 8) featured as the highest scoring benefit with a mean score of ($M=4.60000$, $SD=0.51640$) at 92.0%. The following benefits ranked after with mean score values of ($M=4.5000$, $SD=0.52705$) at 90.0% for the receiving of help from other developer communities (BNF 2), and ($M=4.4000$, $SD=0.69921$) at 88.0% for the acceleration of app development and time to deliver app t the market (BNF 7).

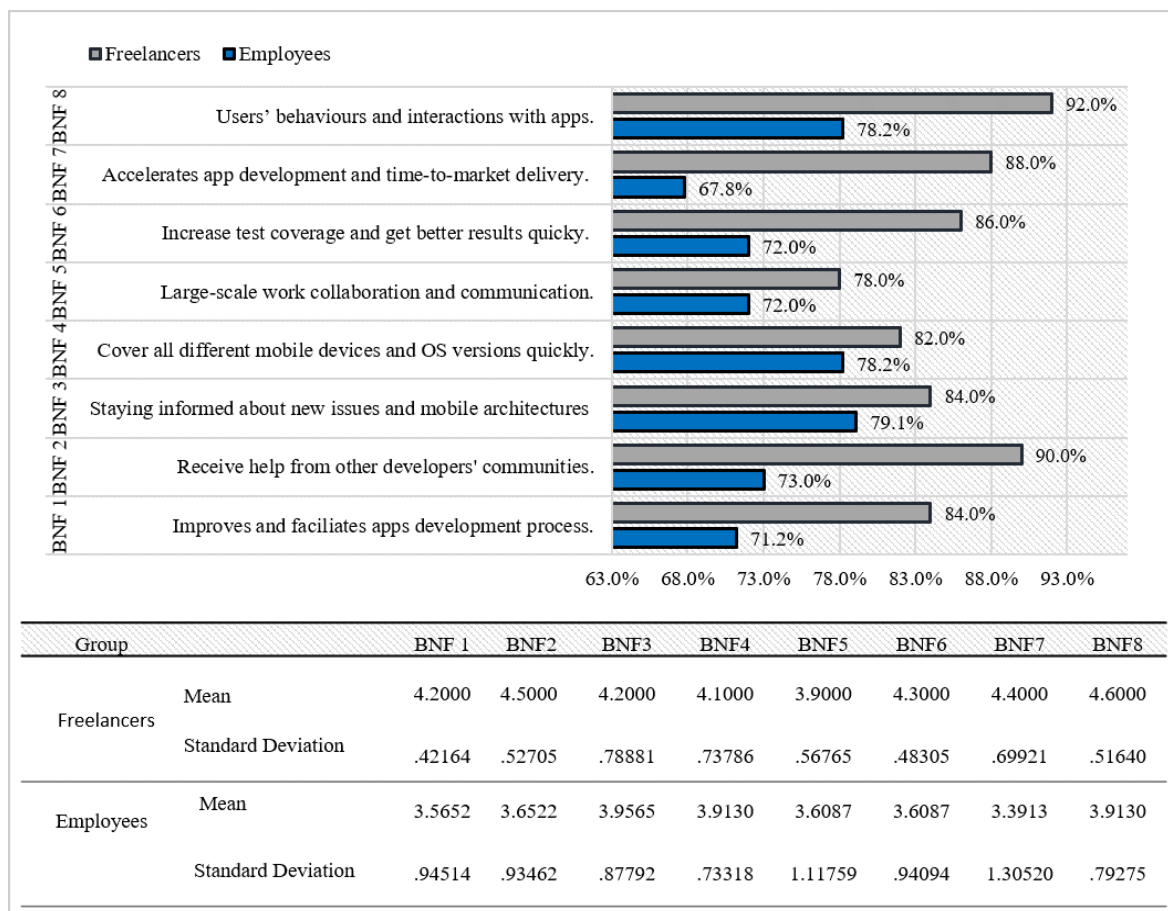


Fig. 7.5 The percentages and mean scores for the most important benefits received by Employees and Freelancers developers.

For Big Organisations Developers: The responses collected from Big Organisations' developers are presented in Figure 7.6. The developers scored the benefits of a range of scores from as low as 51.1% to 73.3%. In more detail, three benefits received the same amount of appreciation from the big organisations with the mode score of 73.3% ($M=3.6667$). This included the capability of the approach to assist developers in: staying informed about new issues discovered in different mobile device models architectures (BNF 3) ($SD=1.11803$); covering all different mobile devices and OS version quickly (BNF 4) ($SD=0.70711$); and gaining more knowledge about discovered issues related to different user behaviours and interactions with apps and mobile devices (BNF 8) ($SD=0.86603$). The next high scoring benefit received is (BNF 2) in terms of receiving help from other developer communities over the world, with a mean score of ($M=3.4444$, $SD=1.13039$) and percentage of 68.8%. On the contrary, the lowest scoring benefit received is (BNF 7) regarding the capability of the proposed approach to accelerate app development and time to deliver apps to the market, with a low mean score ($M=2.5556$, $SD=1.13039$) at 51.1%. Followed by the ability to enhance

large-scale work collaboration and communication (BNF 5), with a mean score ($M=2.8889$, $SD=1.1667$) at 57.0%.

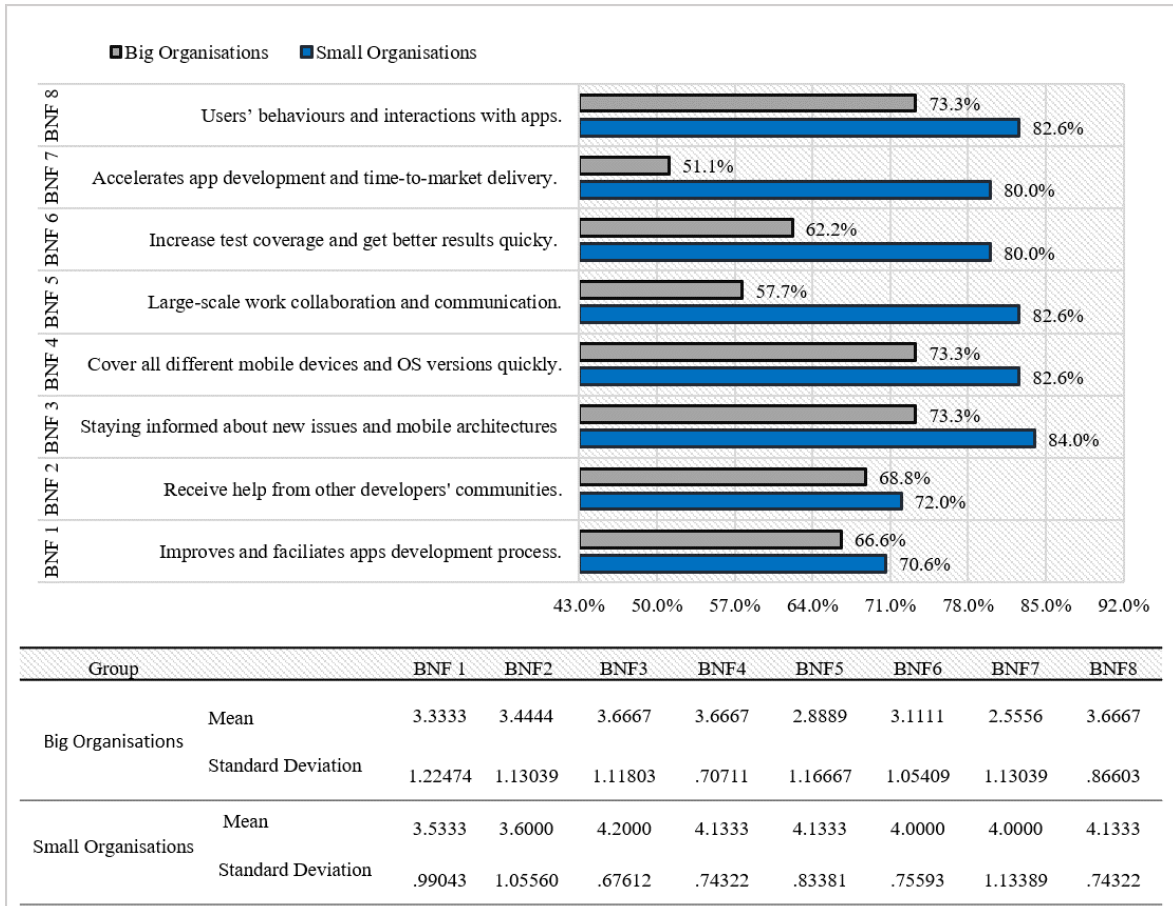


Fig. 7.6 The percentages and mean scores for the most important benefits received by Small Organisations and Big Organisations developers.

For Small Organisations Developers: According to Figure 7.6, within the responses collected from small organisation developers, two benefits have got lower percentages than 80.0% and mean scores less than ($M=4.0000$, $SD<1.0$) in comparison to the remaining other six benefits. When small organisation developers asked about the proposed approach to improve and facilitate the app development process (BNF 1), the developers did not rate this benefit relatively high. The benefit received a moderate mean score ($M=3.5333$, $SD=0.99043$) and 70.6%. The approach capacity to receive help from other developer communities (BNF 2) has recorded as the second lowest benefit with also a moderate mean score ($M=3.6000$, $SD=1.05560$) at 72.0%. All remaining benefits have scored very high mean values $\geq M=4.0000$, $SD<1.0$, and percentages $\geq 80.0\%$. The best scoring benefit record ($M=4.2000$, $SD=0.67612$) at 84.0% for (BNF 3), the capability to stay informed about new unexpected issues relevant to the internal complexity and different architectures of different mobile models or OS version.

7.6 Results III: Satisfaction

This section describes the main findings and results related to developers' satisfaction with the effectiveness and use of the proposed approach. In the beginning, it describes the overall satisfaction in using the proposed approach for performing effective compatibility testing from developers' perspectives. Then, it shows if a specific group of developers are more satisfied than other groups.

7.6.1 Overall Satisfaction of the Approach

Table 7.8 summarises the percentages and mean scores of developers' satisfaction with the proposed approach. The results show that the developers are satisfied with using the proposed approach for their future testing with 74.4% and mean score ($M=3.7255$, $SD=0.96912$) which falls in the high satisfaction level. According to Table 7.8, the responses of developers reflected their high satisfaction about the usage of the approach scored the highest percentage 77.0% and mean score ($M=3.8529$, $SD=0.98880$) followed by their satisfaction for services provided with mean value ($M=3.7941$, $SD=0.91385$) at 75.8%, while the satisfaction about new benefits provided by the approach record the lowest mean ($M=3.5294$, $SD=1.16086$) with a percentage of 70.4%. In general, all these three percentages and mean scores achieved a high satisfaction level according to mean score interpretation levels.

Table 7.8 The mean score and standard deviation values of participant developers according to the "Overall Satisfaction".

Dimension	Question Code	Mean	St. Dev	Level	Percentage (%)
Overall Satisfaction	(SAT)	3.7255	.96912	High	74.4%
	SAT 1	3.8529	.98880	High	77.0%
	SAT 2	3.7941	.91385	High	75.8%
	SAT 3	3.5294	1.16086	High	70.4%

Mean score's interpretation: (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.68 – 5.00).

7.6.2 Satisfaction for Different Developers Groups

This section describes the results obtained from an independent t-test analysis to assess the different developers' groups satisfaction with the use of the proposed approach for compatibility testing in the future. The results presented in Figure 7.7 show a minor difference in the mean scores and satisfaction percentages between Android and iOS developers, as shown in Figure 7.7 and Table 7.9. The t-test analysis results show that this difference is not statistically significant where $p\text{-value} > 0.05$ (0.639). According to the job status variable, the Employees and Freelancers groups, the results presented in Table 7.9 show a big difference in the mean scores of Freelancers developers ($M=4.1333$, $SD=0.47661$) and Employees

($M=3.49281$, $SD=1.05347$). The t-test analysis results show that this difference between these two groups is statistically significant at $p\text{-value} = 0.022 < 0.05$ and $t\text{-value} = -2.405$. This indicates that Freelancers are more satisfied with using the approach at 82.6% than Employees with 69.8%. For the Employees who work on Big Organisations or Small Organisations, the results also show a big difference in the mean scores between them. This difference is statistically significant at a $p\text{-value} < 0.05$ (0.015) and $t\text{-value} = -2.641$. This indicates that the developers employed in small companies were more satisfied with the approach with a high mean score ($M=3.9556$, $SD=0.88968$) and a percentage of 79.1% much more than those working in big companies with a moderate mean score ($M=2.8889$, $SD=1.06719$) with 57.7%. In summary, we found that the freelancer developers and small organisation employed developers are more satisfied with using the proposed approach to perform compatibility testing.

Table 7.9 Summary of the independent samples t-test results (regarding satisfaction)

Variable	Group	N	Mean	St. Dev	t-value	df	Sig.
Operating System	Android	22	3.7667	.94840	-.474	32	.639
	iOS	12	3.8333	1.03962			
Job Status	Employee	24	3.4928	1.05347	-2.405	32	.022
	Freelancer	10	4.1333	.47661			
Workplace Size	Big Organizations	9	2.8889	1.06719	-2.641	22	.015
	Small Organizations	15	3.9556	.88968			

The difference is significant at the $\alpha \leq 0.05$ level (2-tailed)

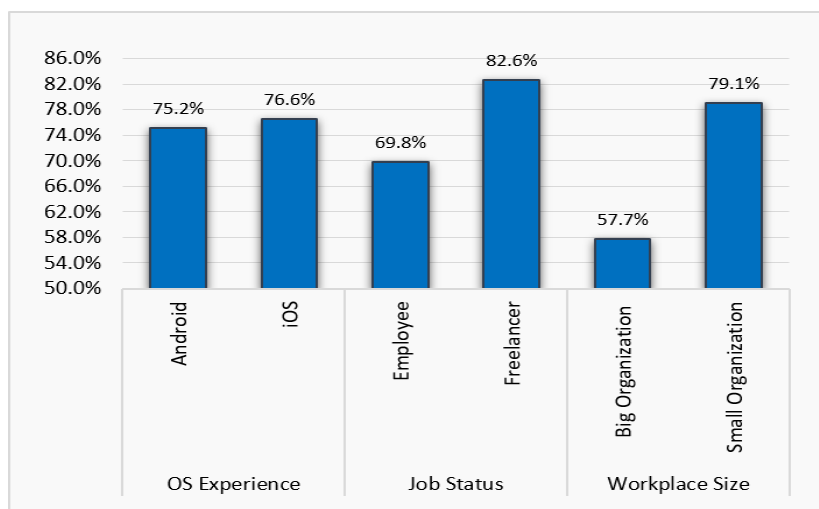


Fig. 7.7 Comparison between different developers' groups regarding overall satisfaction of the approach

7.7 Open Question Analysis Results

This section shows the developers' answers to the *open-ended* question regarding the recommendations they want to share with us to improve the approach's overall quality. A few participants answered this question; their responses involved some positive and negative comments in terms of the overall implementation and the services provided within the approach. For example, the negative comments provided were relevant to two issues that need to be improved: the security method and quality control method. On the other hand, the positive comments were included, such as the good architecture/structure, the seamless integration of the features, the appropriate development of the approach, the good design of the interfaces, and all required features and critical requirements of compatibility testing, especially for Androids mobile platform. It was interesting to us that there are few participants provided some motivational words, such as "I Like it", "We are waiting to use this system officially", and " Waiting for launching".

7.8 Chapter Summary

This chapter presented the main finding and results of the empirical study related to the developer's experience with our approach. We first described the characteristics of the participants' developers. Then, we showed that the approach's effectiveness in eleven defined processes ranges from 73.8% to 81.6%, which supports our approach as a promising approach from the developer's point of view. Task definition and distribution, received a high percentage of 81.6%. This proves the effectiveness of our approach in defining and distributing the test on a large scale successfully. Meanwhile, the privacy and protection process and reliability and trustworthiness scored the lowest effectiveness percentages 73.8%. It is worth mentioning that we also studied the benefits of our approach. As presented in this chapter, we can summarize that small organizations received much more benefit than big organizations. The chapter also described our approach's satisfaction from different developers' groups. It showed that all developers groups are delighted to use the approach. As expected, we find it out that big organizations are less delighted with our approach. Finally, we presented the developers' recommendation regarding the improvements of the approach. The developers agreed on that the security and privacy need to improve. The next chapter will present the results of the empirical study conducted on testers to provide clear evidence on the effectiveness, benefit, and satisfaction of our approach from both sides.

Empirical Study(II): Testers Experience With Crowdfunding Approach

8.1 Introduction

Chapter 6 has described the empirical evaluation study that was carried out on testers. It has explained the measurement variables, data collection method, and procedures that testers performed. The empirical study was completed by 31 mobile app testers familiar with android and iOS and with different levels of experience. Chapter 7 presented the main findings and results related to developers. This chapter will present the main findings and results related to the testers to demonstrate the effectiveness, benefits, and satisfaction of the proposed crowd-based compatibility testing approach. The chapter starts by describing the participants' demographic information and their actual answers to the CSTE-Q/T questionnaire. This is followed by statistical analysis of the results calculated via SPSS. First, we outline how effective the overall approach and the main processes of performing effective compatibility testing from the testers' side. Second, a description of results related to the benefit of the approach in general and then the benefits received by each tester group is provided. Third, we describe the overall satisfaction of the proposed approach for all different testers' groups. Finally, we explain the summative content analysis results of the *open-ended* question to gain more knowledge about the recommendations and advice that testers would like to share with us to improve our approach's experience.

8.2 Testers Demographic Information

This section displays the results related to the demographic information of the 31 participant testers. They had various backgrounds in terms of operating system type, job status, company size, years of experience, usage period, country, and qualifications of these testers. Table 8.1 shows the characteristics of participated testers in this study. In terms of operating systems type, 15 (48.4%) of the participants were solely Android testers, and 9 (29.0%) were iOS testers, while the remaining 7 (22.6%) were testers for both operating systems. It was important to consider the job status of testers. This characteristic shows that 15 (48.4%) were employed full-time, and 16 (51.6%) were freelancers. For the 15 employed testers, 8 (53.3%) of them were workers in large organisations, and 7 (46.7%) were workers in small organisations. The employed testers came from worldwide well-known organizations, such as Trufla Tech, Infusion Infotech, BackBase, FuGenX, Samrat Techno, Mobile doctors, NatWest, and Trivago.

For their level of experience, most of them were intermediates with experience between 3 to 4 years, accounting for 12 (38.7%) of the testers. This was followed by beginners with less than 2 years of experience, who accounted for 11 (35.5%), while expert (senior) tester who have more than 4 years of experience contributed were 8 (25.8%). For the usage period of the approach, the most common usage period was 2 days (n=13, 41.9%), followed by 1 day (n = 9, 29%) and 3 days (n = 7, 22.6%). The penultimate characteristic was the country. Testers have participated from 11 different countries (descending order): 6 (19.3%) from each United Kingdom and the United States, 5 (16.1%) from Germany, 4 (12.9%) from Saudi Arabia, 2 (6.4 %) from each ea Egypt and Pakistan, while only 1 (3.2%) testers came from each India, Netherlands, Romania, Singapore, United Arab Emirates.

The last characteristic was the participants' qualifications. Most of them 23 (74.1%) were certified testers. Some of them have multiple certificates in software testing. 20 (64.5%) of overall testers have the International Software Testing Qualifications Board (ISTQB®) certificate, 3 (9.6%) have certificates on such Agile Analysis Certificate (IIBA®-AAC); while 8 (25.8%) have other certificates on such American Software Testing Qualifications Board (ASTQB) (n=2, 6.4%), Certified Security Testing Professional (CSTP) (n=2, 6.4%), Certified Associate in Software Testing (CAST)(n=2, 6.4%), International Software Quality (iSQ) (n=1, 3.2%), BCS Intermediate Certificate in Software Testing (3.2%, n=1). Only 8 (25.8%) of the freelancer testers do not have any certificate yet.

Table 8.1 Testers demographic information (N = 31).

Characteristics	Range	Frequency (N)	Percentage (%)
Operating System Type	Android	n=15	48.4 %
	iOS	n=9	29.0 %
	Both OS	n=7	22.6 %
Job Status	Freelancer	n=16	51.6 %
	Employee	n=15	48.4 %
Company size (Number of employees)	Small Organisation (n ≤ 200)	n=7	46.7%
	Big Organisation (n > 200)	n=8	53.3 %
Organization Name	Trufla Tech	n=1	3.2 %
	Infusion Infotech	n=1	3.2 %
	BackBase	n=1	3.2 %
	FuGenX	n=1	3.2 %
	Samrat Techno	n=1	3.2 %
	Mobile doctors	n=1	3.2 %
	NatWest	n=1	3.2 %
	Trivago	n=1	3.2 %
Years of experience	Beginner ≤ 2 years	n=11	35.5 %
	Intermediate 3 - 4 years	n=12	38.7 %
	Expert/Senior > 4 years	n=8	25.8 %
Usage Period	1 Day	n=9	29 %
	2 Days	n=13	41.9 %
	3 Days	n=7	22.6 %
Country	United Kingdom	n=6	19.3 %
	Singapore	n=1	3.2 %
	Saudi Arabia	n=4	12.9 %
	Egypt	n=2	6.4 %
	Netherlands	n=1	3.2 %
	Germany	n=5	16.1 %
	India	n=1	3.2 %
	United States	n=6	19.3 %
	Pakistan	n=2	6.4 %
	Romania	n=1	3.2 %
United Arab Emirates	n=1	3.2 %	
Certifications	ISTQB®	n=20	64.5 %
	ASTQB	n=2	6.4 %
	CSTP	n=2	6.4 %
	CAST	n=2	6.4 %
	iSQI	n=1	3.2 %
	BCS	n=1	3.2 %
	Agile (IIBA®-AAC)	n=3	9.6 %
	None	n=8	25.8 %

8.3 Descriptive Statistics Results

Table 8.2 provides an overview of the responses acquired from the participants to each of the questions in the CSTE/Q/T questionnaire. For all 3 questions none of the participants opt for the strongly disagree with the exception of TS 2 and INT 1.

For the dimension of Task Selection (TS), which consisted of 2 questions, 38.7% of the participants agreed with the effectiveness and easiness of the method of selecting appropriate tasks (TS 1), while 48.4% strongly agreed that the list of suggested tasks facilitates the selection process of tasks and help them to perform more than one in a short time (TS 2). For the 3 questions of Access Task Requirements (ACCR) dimension, the agreement or strong agreement was significantly more common compared to the neutrality or the disagreement levels. For example, for ACCR 1, 51.6% of respondents strongly agreed on the sufficiency of the information and instructions given by developers within task requirements to perform an effective testing process. While for ACCR 2, 41.9% agreed that the way of presenting task requirements is sufficiently clear and understandable. Regarding ACCR 3, almost half of participants 48.4% agreed that the information about task complexity level helped them to select and perform the most suitable test, and obtain better results' quality. For Privacy and Protection (PP) dimension, which consisted of 2 questions, the most popular response categories were generally strong agreements. For example, for PP 1, 41.9% of participants strongly agreed that the approach effectively controls private tasks and prevents ineligible testers from accessing or performing it. For PP 2, most of participants 45.2% strongly agreed that the approach has an effective and secure method of linking device information with the submission form.

As for Results Submission (SUB) dimension, which included 3 questions, the most popular responses was strong agreement in all 3 questions. For example, for SUB 1, 41.9% strongly agree with the simplicity and easiness of the results submission method. For SUB 2, 45.2% strongly agreed that the automatic detection service of mobile data helped them submit more accurate results in fewer steps. For SUB 3, 41.9% strongly agreed with the statement of that the reporting mechanism enable them to submit multiple reports for the test of the same task on different mobile device models and OS versions quickly and with less effort. Regarding Results Evaluation and Quality Control (RQC) dimension, which included 3 questions, high agreement levels are identified across the 3 questions. For RQC 1, 32.3% agreed that the approach has a practical and useful mechanism for tracking the status of submitted reports, while for RQC 2, 45.2% agreed with the fairness of the quality evaluation metrics and their ability to increase their commitment and desire to keep participating. For RQC 3, the same number of participants 41.9% agreed or strongly agreed that the insight provided into the history of their work quality would help them to understand the progress and how well they are doing in their work. In terms of the Feedback Given (FED) dimension, which

contained only 2 questions, the most common response categories were agreement and strong agreement in FED 1 and FED 2. For example, for FED 1, 45.2% agreed that providing clear information on the effectiveness of testers' work performance helps improve their work in the future. Additionally, for FED 2, more than half of the participants 54.8% strongly agreed that the feedback about how well they are doing increases their satisfaction with continuous participating and work.

As for the dimension of Interaction between Peers (INT), which consisted of 3 questions, the most popular response categories was the strong agreement for all questions. For example INT 1, 38.7% of participants strongly agreed that direct interaction helps understand more test requirements and effective completion of tasks. For INT 2, most participants 45.2% strongly agreed that increasing the connection and interaction with developers and the other testers community motivates testers to participate more. Finally, for INT 3, regarding reducing the delays caused by the middlemen, the high percentage of responses were agreement or strong agreement with 29.4% each. In terms of Testers' Reliability and Trustworthiness (RT) dimension, which consisted of 3 questions, 35.5% of the participants agreed that the use of our approach with the reliability tracking method improves their reputation among the global testers community (RT 1), while 45.2% agreed about the fairness and trustworthiness of the information and the reliability level identified for each tester (RT 2). As with other dimensions, low levels of strong disagreement were identified for the 2 questions. As for Testers' Motivations and Incentivisation (MI), which consists of 3 questions, the agreement and strong agreement levels were high compared to the neutrality or disagreement levels. For example, for MI 1, 35.5% agreed or strongly agreed with the fairness of reward value compared to the effort they achieve during crowdtesting work. In terms of MI 2, the majority of participants 51.6% strongly agreed that the pay-per-device increases their enthusiasm to do the test on more devices and get better results. Also, most of the participants 48.4% strongly agreed that the evaluation results of each completed task is fair and motivating them to do more tests (MI 3).

As for Knowledge Sharing (KS) dimension, which consist of 3 questions, 41.9% of the participants strongly agreed that the approach provides an effective knowledge-sharing environment between worldwide developers' and testers' communities (KS 1). For KS 2, the number of participants who agreed or strongly agreed with the effectiveness of the documentation mechanism was the same 38.7%. Similarly, 38.7% of participants agreed that documenting testing steps and scenarios helps improve their testing strategies in different testing areas (KS 3). Very low percentage levels of strong disagreement, disagreement, and neutrality are also identified across the 3 questions, with the highest levels of disagreement being associated with KS 3 with 9.7%. In terms of Ease of Use (EU) dimension, which consisted of only 2 questions, most of the participants 45.2% agreed about the ease of use associated with daily working routines and industry practices (EU 1), while the same

percentage strongly agreed that the proposed approach is easy to used for both android and iOS testers (EU 2).

Table 8.2 Testers' Answers to all questions of the CSTE-Q/T questionnaire

Code	Question Summary	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
Task Selection						
TS 1	Easiness of selecting appropriate tasks.	(0) 0%	(1) 3.2%	(7) 22.6%	(12) 38.7%	(11) 35.5%
TS 2	Helpfulness of the suggested task list.	(1) 3.2%	(3) 9.7%	(2) 6.5%	(10) 32.3%	(15) 48.4%
Access Task Requirements						
ACCR 1	Sufficiency of the information given for task requirements.	(0) 0%	(1) 3.2%	(6) 19.4%	(8) 25.8%	(16) 51.6%
ACCR 2	Clarity of requirements presentation.	(0) 0%	(3) 9.7%	(5) 16.1%	(13) 41.9%	(10) 32.3%
ACCR 3	Helpfulness of the information of the task's complexity level.	(0) 0%	(1) 3.2%	(4) 12.9%	(15) 48.4%	(11) 35.5%
Privacy & Protection						
PP 1	Effectivity in controlling private task.	(0) 0%	(1) 3.2%	(7) 22.6%	(10) 32.3%	(13) 41.9%
PP 2	Effectivity and security of linking device information with the submission form.	(0) 0%	(1) 3.2%	(8) 25.8%	(8) 25.8%	(14) 45.2%
Results Submission						
SUB 1	The simplicity of the results submission method.	(0) 0%	(1) 3.2%	(7) 22.6%	(10) 32.3%	(13) 41.9%
SUB 2	Accurate and quick submission of mobile data and results.	(0) 0%	(1) 3.2%	(3) 9.7%	(13) 41.9%	(14) 45.2%
SUB 3	Less effort and quick submission for the same task on different devices.	(0) 0%	(1) 3.2%	(5) 16.1%	(12) 38.7%	(13) 41.9%
Results Evaluation & Quality Control						
RQC 1	Effectivity and usefulness of tracking the status of submitted reports.	(0) 0%	(3) 9.7%	(9) 29%	(10) 32.3%	(9) 29%
RQC 2	Fairness of the quality evaluation metrics.	(0) 0%	(2) 6.5%	(3) 9.7%	(14) 45.2%	(12) 38.7%
RQC 3	Helpfulness of my work quality history.	(0) 0%	(1) 3.2%	(4) 12.9%	(13) 41.9%	(13) 41.9%
Feedback Given						
FED 1	Helpfulness of the direct and clear information on my performance.	(0) 0%	(1) 3.2%	(4) 12.9%	(14) 45.2%	(12) 38.7%
FED 2	Feedback increases my satisfaction and participating.	(0) 0%	(1) 3.2%	(5) 16.1%	(8) 25.8%	(17) 54.8%
Interaction between Peers (Workflow)						
INT 1	Direct interaction help to understand requirements and perform an effective test.	(2) 6.5%	(3) 9.7%	(5) 16.1%	(9) 29%	(12) 38.7%
INT 2	Strength of interaction increases my desire to participate more.	(0) 0%	(1) 3.2%	(4) 12.9%	(12) 38.7%	(14) 45.2%
INT 3	Direct interaction reduces delays caused by the middlemen.	(0) 0%	(1) 3.2%	(8) 25.8%	(11) 35.5%	(11) 35.5%
Testers Reliability and Trustworthiness						
RT 1	Improving my reputation among the global testers community.	(0) 0%	(2) 6.5%	(9) 29%	(11) 35.5%	(9) 29%
RT 2	Fairness and trustworthiness of the testers' reliability level.	(0) 0%	(3) 9.7%	(6) 19.4%	(8) 25.8%	(14) 45.2%
Testers Motivations and Incentivisation						
MI 1	Fairness of the reward value	(0) 0%	(3) 9.7%	(6) 19.4%	(11) 35.5%	(11) 35.5%
MI 2	The pay-per-device increases testers' motivations.	(0) 0%	(1) 3.2%	(4) 12.9%	(10) 32.3%	(16) 51.6%
MI 3	Information provided is fair and motivating.	(0) 0%	(1) 3.2%	(8) 25.8%	(7) 22.6%	(15) 48.4%
Knowledge Sharing (Wiki)						
KS 1	Effectivity of knowledge-sharing environment.	(0) 0%	(2) 6.5%	(5) 16.1%	(11) 35.5%	(13) 41.9%
KS 2	Effectivity of documentation mechanism.	(0) 0%	(2) 6.5%	(5) 16.1%	(12) 38.7%	(12) 38.7%
KS 3	Usefulness of documenting testing steps and scenarios.	(0) 0%	(3) 9.7%	(5) 16.1%	(12) 38.7%	(11) 35.5%
Ease of use						
EU 1	Easiness of use for daily work routine and industry practices.	(0) 0%	(1) 3.2%	(6) 19.4%	(14) 45.2%	(10) 32.3%
EU 2	Easiness of use for both Android and iOS specialists.	(0) 0%	(1) 3.2%	(5) 16.1%	(11) 35.5%	(14) 45.2%
Benefits (Advantage)						
BNF 1	Opportunity to receive help from other testers.	(0) 0%	(1) 3.2%	(4) 12.9%	(18) 58%	(8) 25.8%
BNF 2	Opportunity to improve my testing skills in different areas.	(0) 0%	(1) 3.2%	(3) 9.7%	(22) 70.9%	(5) 16.1%
BNF 3	Opportunity to perform tests in the areas that I am good at.	(0) 0%	(1) 3.2%	(3) 9.7%	(18) 58%	(9) 29%
BNF 4	Opportunity to perform the test in different ways.	(0) 0%	(4) 12.9%	(3) 9.7%	(17) 54.8%	(7) 22.6%
BNF 5	Suitable environment for professional and beginner testers.	(0) 0%	(2) 6.5%	(3) 9.7%	(12) 38.7%	(14) 45.2%
BNF 6	Large-scale work collaboration and communication.	(0) 0%	(1) 3.2%	(5) 16.1%	(11) 35.5%	(14) 45.2%
BNF 7	Ability to perform test anywhere and at any time.	(0) 0%	(2) 6.5%	(5) 16.1%	(13) 41.9%	(11) 35.5%
BNF 8	Feeling a sense of personal achievement.	(0) 0%	(1) 3.2%	(3) 9.7%	(14) 45.2%	(13) 41.9%
Satisfaction						
SAT 1	Use of the approach to achieve a test in the future.	(0) 0%	(1) 3.2%	(5) 16.1%	(11) 35.5%	(14) 45.2%
SAT 2	Nature of the crowdfunding workflow and service provided.	(0) 0%	(2) 6.5%	(4) 12.9%	(12) 38.7%	(13) 41.9%
SAT 3	Benefits received from this crowdfunding approach.	(0) 0%	(3) 9.7%	(5) 16.1%	(11) 35.5%	(12) 38.7%

In the case of the Benefits (BNF) dimension, which consisted of 8 questions, it was generally the case that a very high number of participants agreed with all BNF 1-8 with the exception of BNF 5 and BNF 6, which was strongly agreed. For example, more than half of the participants agreed with the ability to receive help from other testers through our approach (BNF 1) at 58%, perform tests in the areas that I am good at (BNF 3) at 58%, and ability to perform the test in different ways (BNF 4) at 54.8%. In terms of BNF 2, a very significant number of participants 70.9% agreed that the approach could improve their testing skills in different areas. For BNF 5, 45.2% strongly agreed that the approach provides a suitable environment for professional and beginner testers, unlike other approaches. Similarly, the same percentage 45.2% strongly agreed that our approach supports large-scale collaboration and communication among testers and researchers in the domain (BNF 6). In comparison, almost 42% agreed with the ability to perform test anywhere and at any time (BNF 7), while 45.2% agreed that participating and using this approach lets them feel a sense of personal achievement (BNF 8). The same as other dimensions, low levels of disagreement and neutrality response categories are also identified across the 8 questions. Finally, for the Satisfaction (SAT) dimension, almost the same number of participants agreed or strongly agreed with the 3 questions compared to disagreement or neutrality levels. For example, in total, the agreement that the approach would be used to achieve a test in the future (SAT 1) was 80.7%, the agreement on satisfaction with crowdtesting workflow and service provided (SAT 2) was 80.6%, while the agreement on the benefits received from the approach (SAT 3) was 74.2%. As other dimensions, low levels of disagreement was identified for all 3 questions.

8.4 Results I: Effectiveness

Similarly to developers study in Chapter 7, this section describes the main results and findings in terms of the proposed testing approach's effectiveness from testers' perspectives. Moreover, it describes the results related to whether the knowledge sharing, diversity of testers' experiences, and direct interaction between developers and testers affect the overall effectiveness of the approach or its processes.

8.4.1 Overall Effectiveness of the Approach

Table 8.3 demonstrates the mean score and the level for the overall effectiveness of the approach and its main processes (sub-dimensions) from the testers' side. Figure 8.1 presents the effectiveness percentages (associated with each mean score) of each process. The data presented in Table 8.3 shows that the overall effectiveness of the approach for testers has reached a very high level at ($M=4.1141$, $SD=0.67432$) and percentage of 82.2%. A closer examination of Table 8.3 further reveals that all mean scores of the effectiveness sub-dimensions have recorded mean scores > 3.9500 and $SD < 0.09$, which represent a high effectiveness

level with a percentage $> 79.0\%$. The results show that of the 11 mean scores of effectiveness sub-dimensions, the highest mean score of ($M=4.2581$, $SD=0.64383$) at 85.0% resulted from the testers' responses to the effectiveness and importance of developers' feedback provided to the public testers (FED). The second highest effectiveness percentage and mean score have associated with the results submission mechanism with a mean ($M=4.2143$, $SD=0.67042$) and a percentage of 84.2% . The testers' responses also indicated they found that the implemented motivation and incentivisation method in our approach is the third most effective process with a mean score ($M=4.1505$, $SD=0.75933$) and a percentage at 83.0% . In addition, the results have demonstrated that the testers rated the method of classifying them based on their work quality and reliability (RT) as the lowest effectiveness processes, scoring a high mean of ($M=3.9677$, $SD=0.80556$) with 79.2% . Almost all the remaining processes (sub-dimensions) effectiveness was relatively convergent, with a percentage ranged from 80.8% to 82.8% and with fairly convergent mean scores, as outlined in Table 8.3.

Table 8.3 The mean score and standard deviation values of participant testers according to the "Overall Effectiveness".

Dimension/ Sub-dimension	Mean	St. Dev	Effectiveness Level
Overall Effectiveness	4.1141	.67432	High
Task Selection (TS)	4.0968	.89833	High
Access Task Requirements (ACCR)	4.1290	.74375	High
Privacy and Protection (PP)	4.1290	.78494	High
Results Submission (SUB)	4.2143	.67042	High
Results Evaluation and Quality Control (RQC)	4.0645	.79078	High
Feedback Given (FED)	4.2581	.64383	High
Interaction between Peers (INT)/Workflow	4.0430	.70837	High
Testers Reliability and Trustworthiness (RT)	3.9677	.80556	High
Testers Motivations and Incentivisation (MI)	4.1505	.75933	High
Knowledge Sharing (KS)/ WiKi	4.0753	.79680	High
Ease of use (EU)	4.1452	.74379	High

Mean score's interpretation: (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.67 – 5.00).

8.4.2 Effectiveness of the each processes in the approach

Table 8.4 describe the effectiveness of each question within all sub-dimensions via the mean scores. Figure 8.2 present the effectiveness in percentages. In general, all the questions of all sub-dimension was fall in the high level of effectiveness.

The Task Selection sub-dimension contained two questions that were interpreted to have a high level of effectiveness. The easiness and effectiveness of the tasks selection method (TS 1) was high with a mean value ($M=4.0645$, $SD=0.85383$) and a percentage of 81.2% . At the same time, the list of suggested tasks was helpful for testers to select and perform more tasks in a short time (TS 2) with a high mean value ($M=4.1290$, $SD=1.11779$) and a percentage of 82.4% .

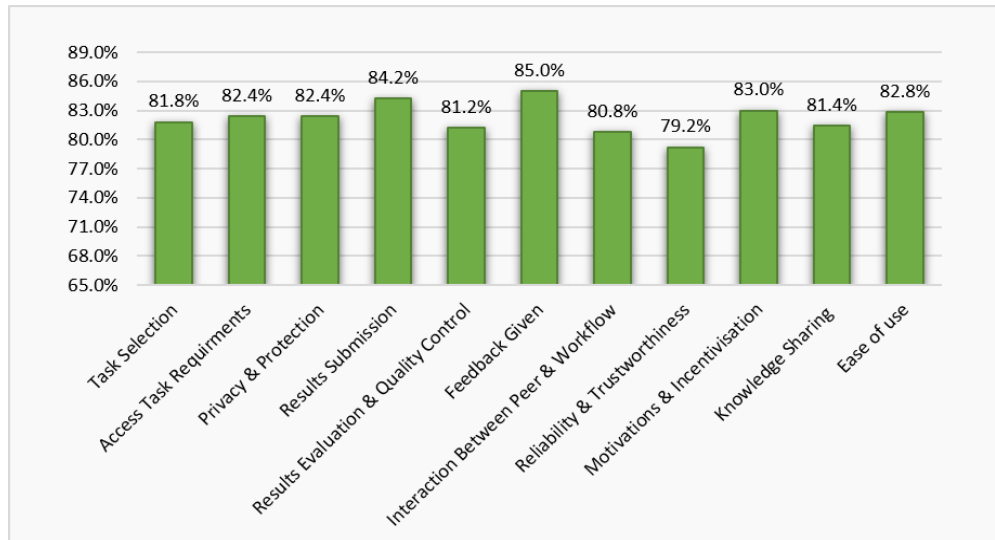
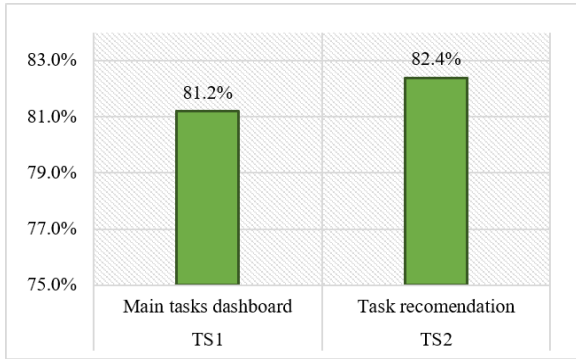


Fig. 8.1 The effectiveness percentage for each process in the approach, which achieves the overall effectiveness percentage (82.2%).

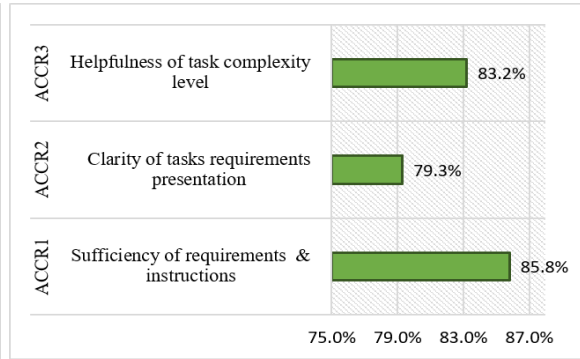
Table 8.4 Summary of the mean values of all questions within effectiveness sub-dimensions.

Sub-dimension	Question Code	Mean (M)	Std. Deviation (SD)	Level
Task Selection	TS 1	4.0645	.85383	High
	TS 2	4.1290	1.11779	High
Accessing Test Requirement	ACCR 1	4.2581	.89322	High
	ACCR 2	3.9677	.94812	High
	ACCR 3	4.1613	.77875	High
Privacy and Protection	PP 1	4.1290	.88476	High
	PP 2	4.1290	.92166	High
Results Submission	SUB 1	4.1290	.88476	High
	SUB 2	4.2903	.78288	High
	SUB 3	4.1935	.83344	High
Results Evaluation and Quality Control	RQC 1	3.8065	.98045	High
	RQC 2	4.1613	.86011	High
	RQC 3	4.2258	.80456	High
Feedback Given	FED 1	4.1935	.79244	High
	FED 2	4.3226	.87129	High
Interaction between Peers and Workflow	INT 1	3.8387	1.24088	High
	INT 2	4.2581	.81518	High
	INT 3	4.0323	.87498	High
Testers Reliability and Trustworthiness	RT 1	3.8710	.92166	High
	RT 2	4.0645	1.03071	High
Testers Motivations and Incentivisation	MI 1	3.9677	.98265	High
	MI 2	4.3226	.83215	High
	MI 3	4.1613	.93441	High
Knowledge Sharing	KS 1	4.1290	.92166	High
	KS 2	4.0968	.90755	High
	KS 3	4.0000	.96609	High
Ease of use	EU 1	4.0645	.81386	High
	EU 2	4.2258	.84497	High

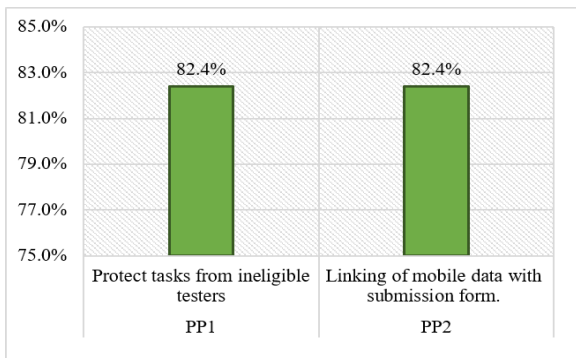
Mean score's interpretation: (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.67 – 5.00).



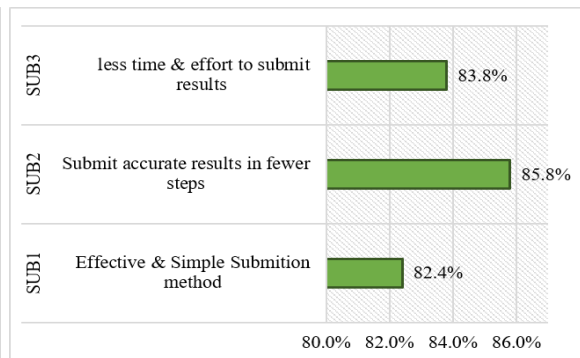
(a) Task Selection (81.8%)



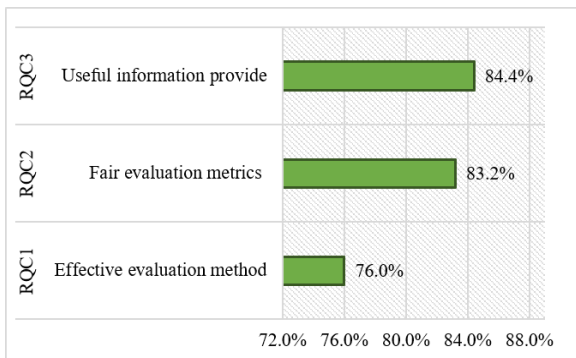
(b) Accessing Test Requirements (82.4%)



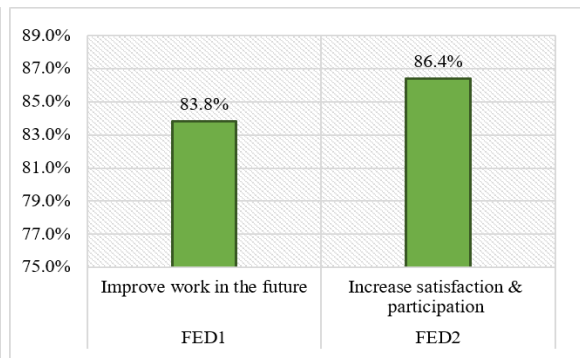
(c) Privacy and Protection (82.4%)



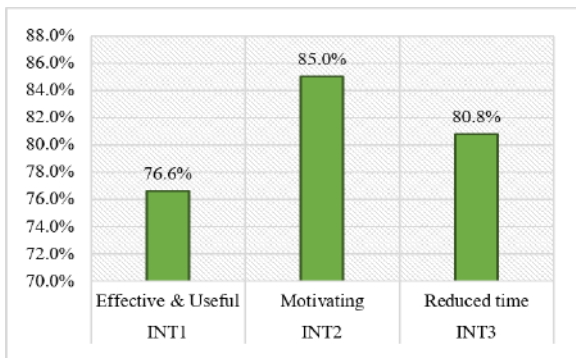
(d) Results Submission (84.2%)



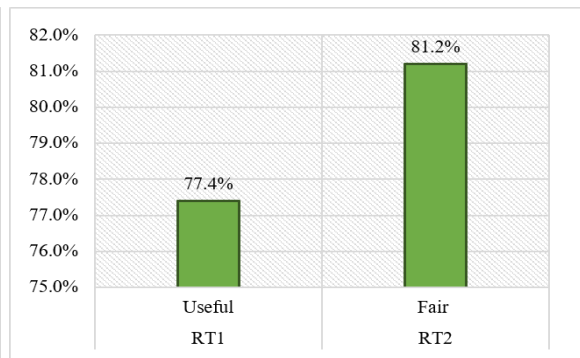
(e) Results Evaluation and Quality Control (81.2%)



(f) Feedback Given (85%)



(g) Interaction between Peers and Workflow (80.8%)



(h) Testers Reliability and Trustworthiness (79.2%)

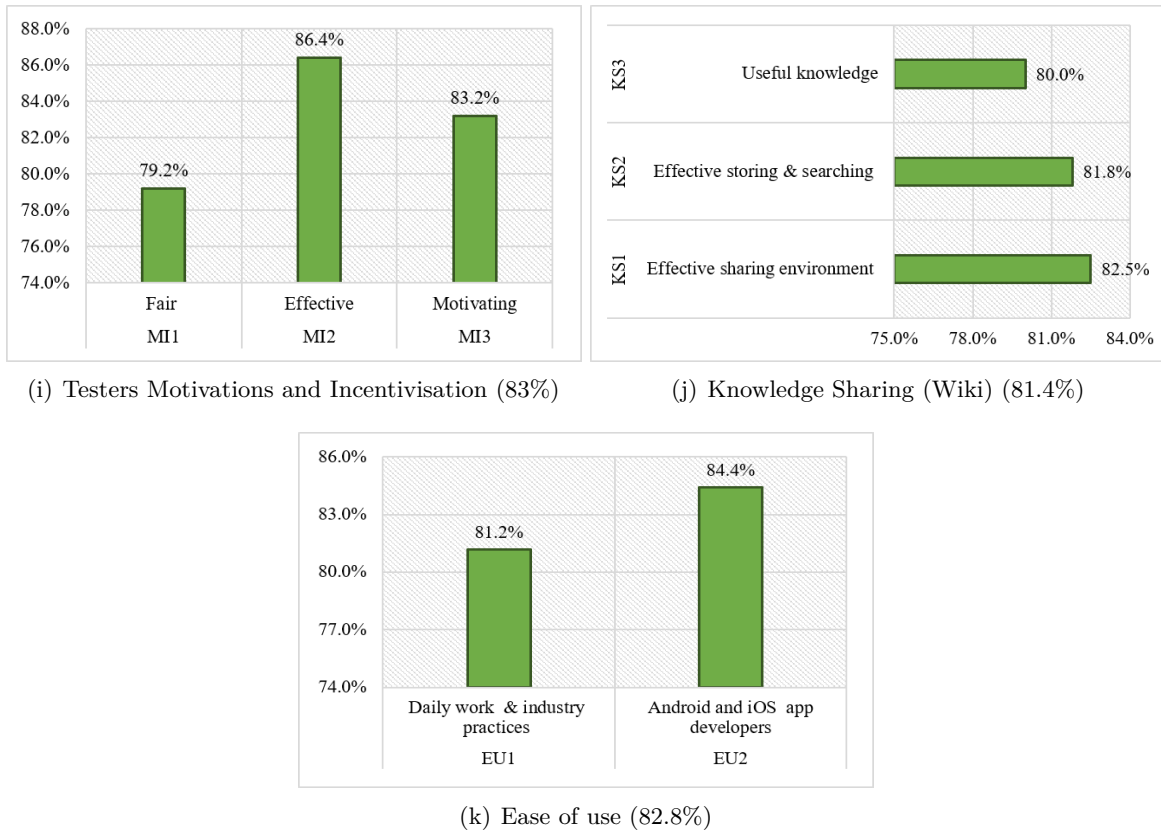


Fig. 8.2 Effectiveness percentage for each characteristic within all processes (sub-dimensions) of the proposed approach

The following sub-dimension was Accessing Testing Requirement, which contained three different questions. The information and instructions given by developers were significantly sufficient with a mean value ($M=4.2581$, $SD=0.89322$) and a percentage of 85.8% to perform an effective testing process (ACCR 1). The presentation of tasks' requirements was sufficiently clear and understandable (ACCR 2) with a high mean value ($M=3.9677$, $SD=0.94812$) at 79.3%. The information provided on task complexity level was helpful to perform the most suitable test and obtain better results' quality (ACCR 3) with a very high percentage of 83.2% and mean value ($M=4.1613$, $SD=0.77875$). For the Privacy and Protection sub-dimension, the effectiveness of the approach in preventing ineligible testers from accessing and selecting the private task (PP 1) and the effectiveness in securing the device information linked with the submission form (PP 2) was high with a percentage of 82.4% and with mean value ($M=4.1290$, $SD<1.0000$). The way of submitting results was simple and effective (SUB 1) with a mean value ($M=4.1290$, $SD=0.88476$) and a percentage of 82.4%. The automatic detection service of mobile data helped collect and submit more accurate results in fewer steps (SUB 2) with a significant percentage of 85.5% and at mean ($M=4.2903$, $SD=0.78288$). The submission mechanism's effectiveness in performing the same task by different mobile

devices quickly with less effort (SUB 3) was high at 83.8% and with a mean ($M=4.1935$, $SD=0.83344$).

For the Results Evaluation and Quality Control, the effectiveness of the mechanism for tracking the status of submitted reports (RQC 1) was 76% with a mean value ($M=3.8065$, $SD=0.98045$). The fairness of quality evaluation metrics and its helpfulness in increasing testers' commitment and desire to keep participating (RQC 2) was high with a mean value ($M=4.1613$, $SD=0.86011$) at 83.2%. The insight provided into the history of testers' work quality was helpful to know how well they are doing (RQC 2) with a high mean value ($M=4.2258$, $SD=0.80456$) at 84.4%. Between the two questions regarding Feedback Given sub-dimension, the direct and clear information about the effectiveness of testers' performance was helpful to improve testers' work (FED 1) with 83.8% and a mean value ($M=4.1935$, $SD=0.79244$). The feedback about how well testers are doing was effective in increasing their satisfaction and motivating them to continue participating and work (FED 2) with a significant mean value ($M=4.3226$, $SD=0.87129$) at 86.4%.

For the Interaction between Peers and Workflow, the effectiveness of developers' and testers' direct interaction without middlemen in understanding more test requirements and effective completion of tasks (INT 1) was high with a mean score ($m=3.8387$, $SD=1.24088$) at 76.6%. Increasing the strength of the connection and interaction between developers and the testers was effective in motivating testers more (INT 2) with a significant percentage of 85% and mean value ($M=4.2581$, $SD=0.81518$). The effectiveness of the direct interaction in reducing the delays caused by the middlemen figures alike (managers or leaders) compared to other approaches (INT 3) was highly effective with a mean value ($M=4.0323$, $SD=0.87498$) at 80.8%. For the two questions regarding Testers Reliability and Trust Worthiness, the helpfulness of the approach in improving testers' reputation among other testers in the global community (RT 1) was high with 77.4% and mean value ($M=3.8710$, $SD=0.92166$). The approach's effectiveness in identifying each tester's fair and reliable level (RT 2) was high with a mean value ($M=4.0645$, $SD=1.03071$) at 81.2%.

For the Testers Motivations and Incentivisation, the value of the rewards was fair and reflected the effort that testers achieve in crowdtesting work (MI 1) with a mean value ($M=3.9677$, $SD=0.98265$) and percentage of 79.2%. The pay-per-device was effective in increasing testers' motivations to do the test on more devices and get a higher quality of results (MI 2) with a significant percentage of 86.4% and mean value ($M=4.3226$, $SD=0.83215$). The reward information provided for each completed task was fair and motivating (MI 3) with a high mean value ($M=4.1613$, $SD=0.93441$) and a percentage of 83.2%. For Knowledge Sharing, the knowledge-sharing environment between worldwide developers' and testers' communities (KS 1) was effective with a mean score ($M=4.1290$, $SD=0.92166$) and a percentage of 82.5%. The documentation and the knowledge searching mechanism (KS 2) were highly effective with a mean value ($M=4.0968$, $SD=0.90755$) at 81.8%. The documentation of testing steps was helpful in improving testers testing strategy in different testing areas (KS 3) with a mean

($M=4.0000$, $SD=0.96609$) and a percentage of 80%. For the final sub-dimension, Ease of Use, the easiness of using the proposed approach in daily work routine and industry practices (EU 1) was high with 81.2% and a mean of ($M=4.0645$, $SD=0.81386$). The ease of use and interaction for Android and iOS specialists with the approach (EU 2) was very high with 84.4% and mean value ($M=4.2258$, $SD=0.84497$).

8.5 Results II: Benefits/Advantages

In this section, a description of the analysis results and main findings related to the benefits arising from the use of the approach in performing compatibility testing compared to other approaches, from testers' perspectives is presented. Also, it shows whether the approach benefits a particular group of developers than others.

8.5.1 Benefits of the Approach

The testers' responses show that the proposed approach is very beneficial with a high mean score ($M=4.1048$, $SD=0.61500$) and percentage of 82.0%. According to the results displayed in Table 8.5, no responses yielded a lower percentage than 77.0% or a mean value less than 3.800 for all asked questions. The approach ability to provide testers with a personal sense of achievement (BNF8) ranked the highest with a mean score ($M= 4.2581$, $SD=0.7732$) and percentages of 85.0%, compared to other benefits. The subsequent benefits ranked second with a very high mean score ($M=4.2258$, $SD=0.8$) with 84.4%. Namely, the suitability of the testing environment for professional and beginner testers (BNF5) and the large-scale collaboration/communication among testers and researchers in the domain (BNF6). Moreover, the opportunity to receive help from other testers communities (BNF1), improve testers skillets in different areas (BNF2), perform tests in the areas that they good at (BNF3), and the flexibility of work (BNF7) ranked as the third most beneficial category, with a fairly close mean scores ranging between $M=4.0000$ to $M=4.1290$, with $SD < 0.9$ and percentages from 80.0% to 82.4%. Finally, the data presented in Table 8.5 yielded the lowest mean score ($M=3.8710$, $SD=0.9216$) and percentage 77.4% (although still is also considered in the high level) for the BNF4, in terms of the ability of the approach in helping them to achieve the test in different ways.

8.5.2 The Difference Between Testers Groups Regarding Benefits

Table 8.6 summarizes the independent t-test results to assess the benefit of the approach for the different testers' groups, while Figure 8.3 illustrates the percentages for each group based on the calculated mean scores. The independent t-test analysis results presented in Table 8.6 show that there is no statistically significant difference ($t\text{-value} = 0.992$, $p\text{-value}=0.330 > 0.05$) between the mean score of Freelancers ($M=4.2109$, $SD=0.48244$), which represents

Table 8.5 The mean score and standard deviation values of participant testers according to the "Benefit//advantage"

Dimension	Question Code	Mean	St. Dev	Level	Percentage (%)
Benefits/Advantages	(BNF)	4.1048	.61500	High	82.0 %
	BNF 1	4.0645	.72735	High	81.2 %
	BNF 2	4.0000	.63246	High	80.0 %
	BNF 3	4.1290	.71842	High	82.4 %
	BNF 4	3.8710	.92166	High	77.4 %
	BNF 5	4.2258	.88354	High	84.4 %
	BNF 6	4.2258	.84497	High	84.4 %
	BNF 7	4.0645	.89202	High	81.2 %
	BNF 8	4.2581	.77321	High	85.0 %

Mean score's interpretation: (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.67 – 5.00).

84.2%, and Employees (M=3.9917 SD=0.73111), which represent 79.8%. This means that both of them have benefited at the same level. The results presented in Figure 8.3 show that there was a difference in the percentage of benefit for the two groups according to the "workplace size" variable. The Small Organisations testers have benefited at 86.4% and Big Organisations testers at 74.0%. Although this difference in the percentages the t-test results for the mean of testers who work in Small Organisations (M=4.3214, SD=0.27817) and those who work in Big Organisations (M=3.7031, SD=0.89377) demonstrated that this difference is not statistically significant, where t-value = 1.750 and p-value=0.104 > 0.05. As there are no statistically significant differences between the mean values of the two groups of the "Workplace Size" variable, this shows that both Small Organisations testers and Big Organisations have benefited from the approach with the same degree.

Table 8.6 Summary of the independent samples t-test results (regarding benefits)

Variable	Group	N	Mean	St. Dev	t-value	df	Sig.
Job Status	Freelancer	16	4.2109	.48244	.992	29	.330
	Employee	15	3.9917	.73111			
Workplace Size	Small Organizations	7	4.3214	.27817	1.750	13	.104
	Big Organizations	8	3.7031	.89377			

According to the One-way ANOVA analysis result presented in Table 8.7 no significant differences in the mean values of the three groups of testers, Android, iOS, and the experts in both OS were found regarding the benefit of the approach (F-value = 0.240, p-value = .788 > 0.05). This demonstrates that all these three groups have benefited with the approach with a percentage higher than 80.0%. Besides, regarding the three groups under the "Experience Level" variable, it was a difference in percentages of testers with a mid-level of experience

give the highest mean score ($M=4.2292$, $SD=0.34059$) and percentage 84.5%, followed by Beginner with a mean score ($M=4.1250$, $SD=0.51539$) at 82.5%, and the Experts testers who give the lowest mean score with ($M=3.8906$, $SD=0.98976$) with 77.9%. Despite this difference in the three groups, the One-way ANOVA analysis results showed no statistically significant differences were found ($F\text{-value}=0.723$, $p\text{-value}=0.494 > 0.05$). To summarize, the results show that all testers groups have benefited from the approach for large scale compatibility testing with a high percentage of $\geq 74.0\%$ in all testers groups.

Table 8.7 Summary of the one-way ANOVA test results (regarding benefits)

Variable	Group	N	Mean	St. Dev	Comparison	Sum of Squares	df	Mean Square	F	Sig.
Operating System	Android	15	4.1833	.51726	Between Groups	.191	2	.096	.240	.788
	iOS	9	4.0556	.82469	Within Groups	11.156	28	.398		
	Both OS	7	4.0000	.57282	Total	11.347	30			
Experience Level	Beginners	11	4.1250	.51539	Between Groups	.557	2	.279	.723	.494
	Mid-level	12	4.2292	.34059	Within Groups	10.790	28	.385		
	Experts	8	3.8906	.98976	Total	11.347	30			

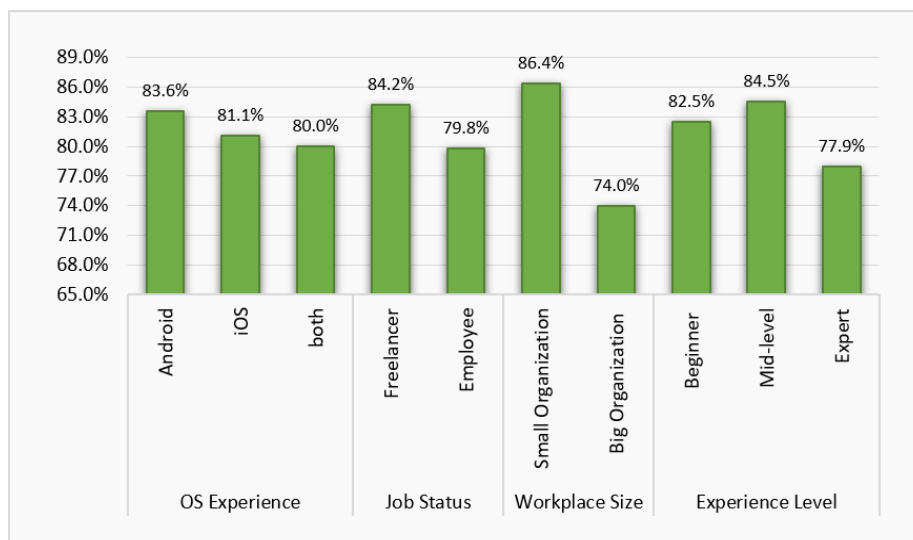


Fig. 8.3 Comparison between different contexts of testers regarding benefit/advantage of the approach

8.5.3 Received Benefits by Each Group

The previous section described the differences between the different testers' groups to discover if there is a statistically significant difference between them. This section will provide broader

knowledge about the most important benefits obtained from the approach for each group separately by describing the mean scores of each group’s responses and their percentages.

For Android Testers: Figure 8.4 demonstrates the responses collected from Android testers in terms of the received benefits. According to the results, no response yielded a scored lower than 77.0% or a mean value less than 3.8000 for all listed benefits. The least significant benefit gained from using the approach was performing testes in different ways (BNF 4), with a mean score of (M=3.8667, SD=0.92548) at 77.2%. All remaining benefits recorded very high mean scores above (M=4.0000) and percentages greater than 80.0%. The suitability of the testing environment for professional and beginner testers (BNF 5) featured the highest benefit/advantage with a very high mean score of (M=4.4000, SD=0.63256) at 88.0%. The following benefits regarding performing the tests in the areas that they are good at (BNF 3), large-scale collaboration and communication among testers and researchers in the domain (BNF 6), and a personal sense of achievement (BNF8) ranked the second with mean score values of (M=4.2667, SD < 0.900) at 85.2%.

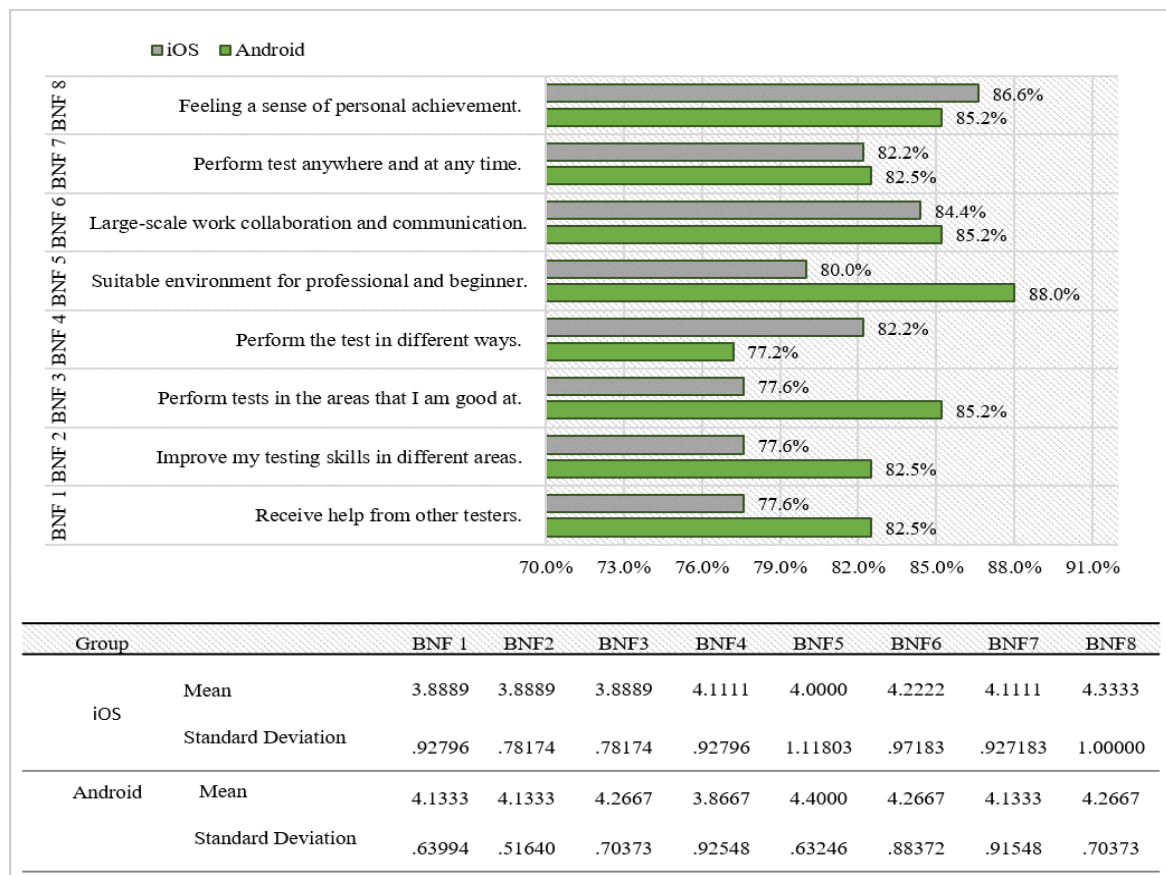


Fig. 8.4 The percentages and mean scores for the most important benefits received by Android and iOS testers.

For iOS Testers: The responses collected from iOS testers have also been presented in Figure 8.4. From the figure, the most beneficial that iOS testers gained from using the proposed approach are the personal sense of achievement (BNF8), with the highest mean score of (M=4.3333, SD=1.000) at 86.6%. The following benefit (BNF 6) in terms of large-scale collaboration and communication among testers and researchers in the domain ranked second with a mean score of (M=4.2222, SD=0.97183) with 84.4%. Testers ranked the freedom to perform the test in different ways (BNF 4) and workplace and time flexibility (BN F 7) as the third with a similar level of beneficial, with the same mean score (M=4.1111, SD=0.927) at 82.2%. On the contrary, the lowest scoring benefits were the recipient of help from other communities (BNF 1), improve their testing skills in different areas (BNF2), and the ability to select and perform tests in the areas that they good at (BNF 3), with the same mean score (M=3.8889, SD < 1.0) and with percentages of 77.6%.

For Employees Testers: Figure 8.5 presents the gained benefits of the approach for the employees' testers. From the results, the suitability and flexibility of the testing environment for professionals and beginners (BNF 5) emerged as a first gained benefit with the highest score of (M=4.2000, SD=0.86189) at 84.0%. The large-scale work collaboration and communication (BNF 6) ranked second with a mean value of (M=4.1333, SD=0.99043) at 82.6%. Receiving help from other communities (BNF 1), performing tests they prefer and in the areas that they are good at (BNF 3), and the personal sense of achievement (BNF8) placed the third with a mean score of (M=4.0667, SD < 0.9) and percentages of 81.2%. For the lower benefits, the employees' testers ranked the ease of using the approach to perform a test anywhere at any time (BNF 7) as the lowest, with a score of (M=3.7333, SD=0.88372) at 74.6%. The approach facilities to perform tests in different ways (BNF 4) was slightly more appreciated by testers with a mean score of (M=3.8000, SD=0.94112) and with 76.0%; followed by the ability to improve their testing skills in different areas (BNF 2) with a mean score of (M=3.8667, SD=0.74322) at 77.0%.

For Freelancers Testers: Figure 8.5 also describes the gathered responses from the freelance testers. The results show that all the benefits have scored a very high percentage above 80.0%, except BNF 4. According to the results, the approach's ability to provide the testers with a sense of personal achievement (BNF 8) ranked highest with a score of (M=4.4375, SD=0.62915) 88.6%. This was closely followed by the flexibility in performing tests anywhere and at any time (BNF 7) at (M=4.3750, SD=0.80623) with 87.4%. On the opposing side, the ability to perform tests in a different way (BNF 4) ranked the lowest with a very high mean score (M=3.9375, SD=0.92871) at 78.6%.

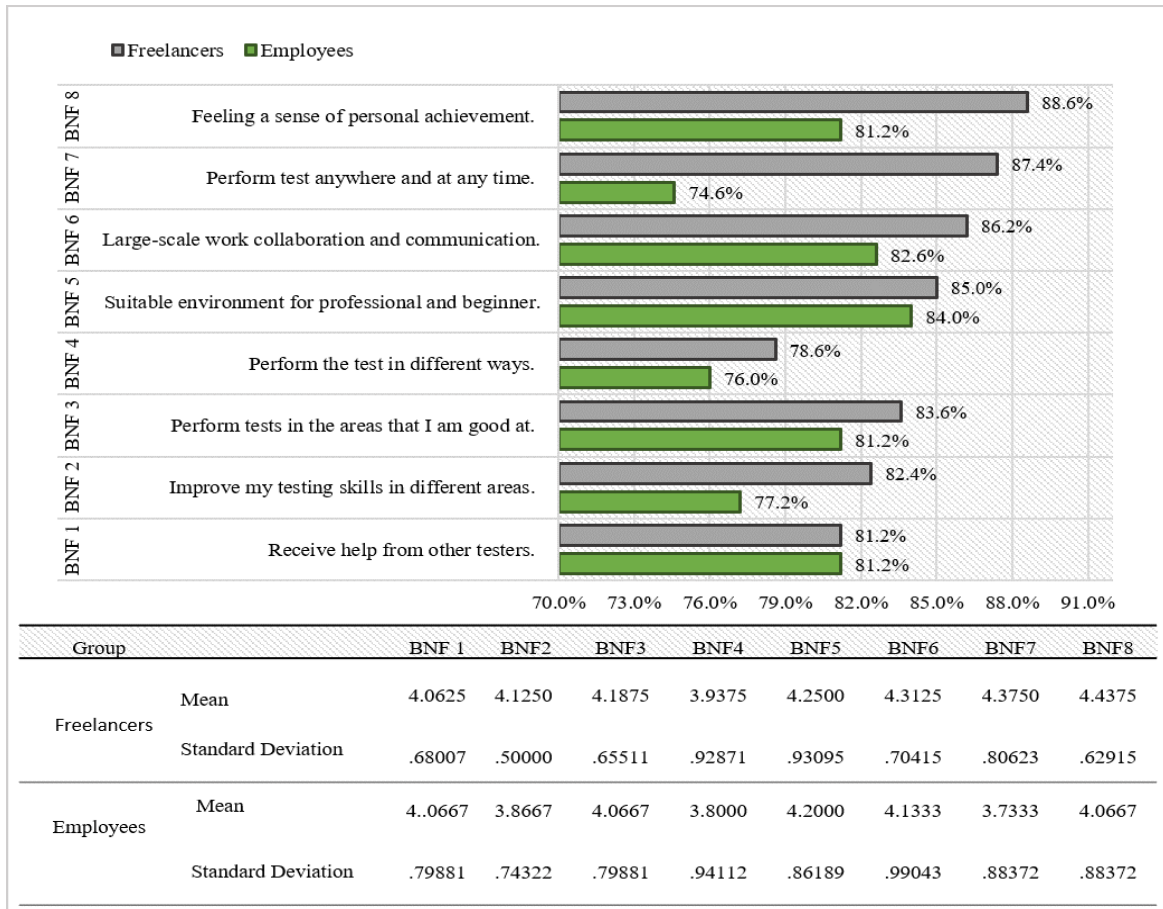


Fig. 8.5 The percentages and mean scores for the most important benefits received by Employees and Freelancers testers.

For Big Organisations Testers: The responses collected from Big Organisation testers were explained in Figure 8.6. The testers scored the benefits of a range of scores from 67.5% to 77.5%. In more detail, three benefits received the same amount of appreciation from the big organisations testers with a mean score of (M=3.8750, SD=0.99103) and percentage of 77.5%. This included receive help from other communities (BNF 1); perform tests in the areas that they good at (BNF 3); suitability of the testing environment for the different experience for professional and beginner testers (BNF 5). The next close high scoring benefits received a mean score of(M=3.7500) at 75.0% for ability of the approach to improve their testing skills in different areas (BNF 2) with (SD=88641) and a personal sense of achievement (BNF 8) with (SD=1.103510). On the contrary, the testers’ ability on performing the test at anywhere and at reasonable time (BNF 7) recorded the lowest benefit with a mean score of (M=3.3750, SD=1.06066) at 67.5%. Followed by the ability to perform the tests in different ways (BNF 4) with a mean score of (M=3.5000, SD=1.9523) at 70.0%.

For Small Organisations Testers: According to Figure 8.6 and presented responses from small organisations' testers, no responses yielded a lower percentage than 80.0% or a mean value less than $M=4.0000$ for all listed benefits. Only two benefits have scored significant percentages greater than 90.0% and mean scores above $M=4.5000$ compared to the remaining six benefits. Based on the testers' responses, the approach's ability to improve their testing skills in different areas (BNF 2) received the lowest scoring benefit with a very high mean value ($M=4.0000$, $SD=0.57735$) and 80.0%. The best benefit received by small organisations testers was large-scale collaboration and communication with other worldwide testers and developers (BNF 6) with a significant mean value ($M=4.7143$, $SD=0.48795$) at 94.2%. The suitability of the testing environment for professional and beginner testers with different skills to perform different tests accurately (BNF 5) recorded the second highest benefit with a significant mean score ($M=4.5714$, $SD=0.53452$) at 91.4%. All remaining benefits have scored very high mean values ranged from $M=4.1429$ to $M=4.4286$ with $SD < 0.6$, and percentages from 82.8 % to 88.5%.

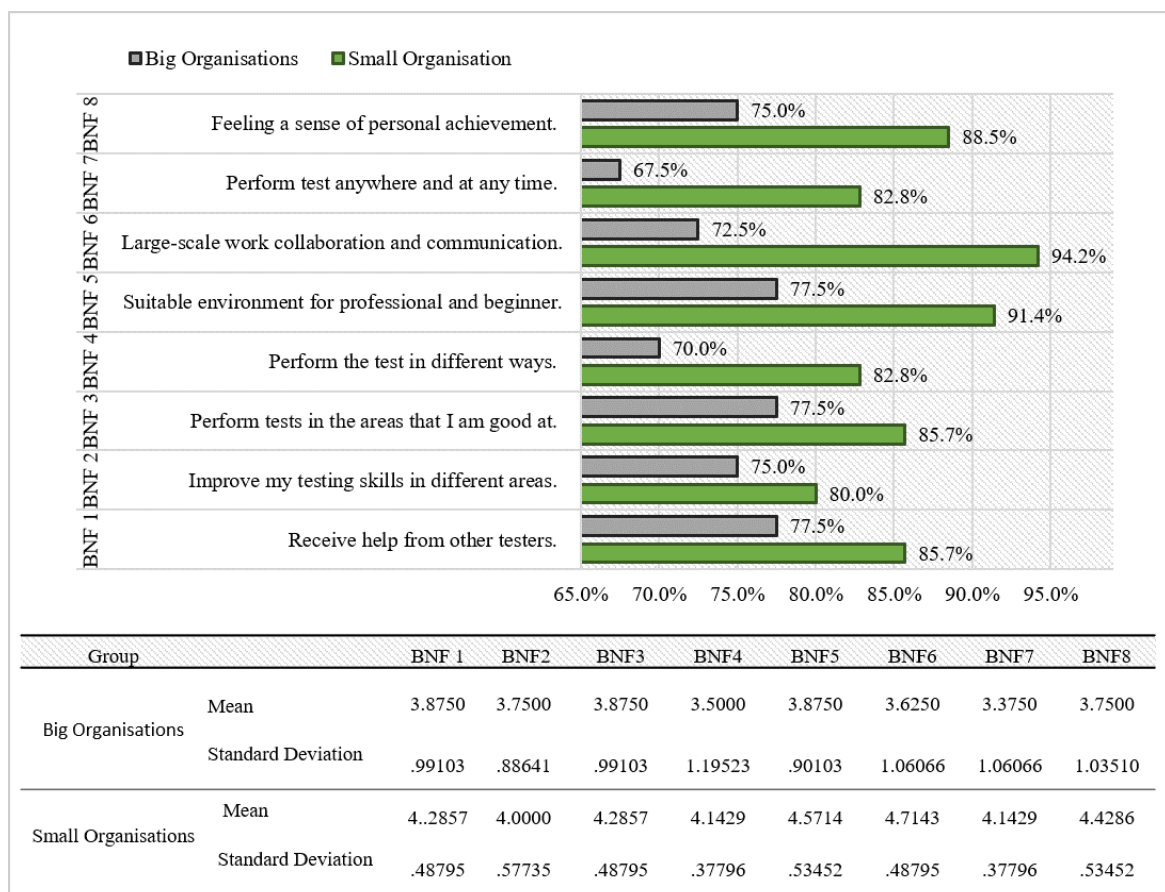


Fig. 8.6 The percentages and mean scores for the most important benefits received by Small Organisations and Big Organisations testers.

For Experts Testers: Figure 8.7 shows the responses of the experts, mid-level and beginners testers regarding the received benefits from the approach. In general, the experts were the most critical, providing the lowest mean score relative to others. Among expert testers' responses, two benefits ranked bottom with the same mean score of ($M=3.5000$, $SD=1.19523$) and percentages at 70.0%. This highlights that expert testers have less benefited from the approach in terms of flexibility in performing tests anywhere at any time (BNF 7) or perform the test in different ways (BNF 4). Above this, the testers ranked all remaining benefits with a score higher than ($M=3.8750$) at 77.4%. From the higher ranked benefits, two benefits again score the same, with the highest value of ($M=4.1250$) at 82.4%. The proposed approach gave testers confidence that they could perform tests in areas they are good at (BNF 3) ($SD=1.12599$) and enable large-scale collaboration and communication (BNF 6) ($SD=1.24642$).

For Mid-level Testers: Mid-level testers were more generous with their rating of the benefits, providing the highest mean value of ($M=4.5000$) at 90.0%. According to the results displayed in Figure 8.7, no responses yielded a lower percentage than 80.0% or a mean value less than 4.0000 for all benefits. The testers enjoyed the approach in which it made them feel a sense of personal achievement and job satisfaction, which ranked the best and first benefit ($M=4.5000$, $SD=0.67420$). The following benefits scored the same score of ($M=4.3333$) at 86.6%. Namely, the capability to communicate and collaborate with other testers and developers on a large-scale (BNF 6) ($SD=0.77850$) and the suitable environment provided by the approach for professionals and beginners (BNF 5) ($SD=0.65134$). The range between the highest and lowest ranked benefits was just 6.6%. According to mid-level testers, the benefit of improving testers' testing skills in different areas was the least beneficial aspect of the proposed approach, with a high mean value ($M=4.0000$, $SD=0.42640$) 80.0%, which represent a high beneficial percentage.

For Beginners Testers: In general, the beginners have ranked all benefits with percentages above 83.0% relative to all but one benefit, BNF 4. It can see from the results that beginner testers were impressed most by two out of the eight benefits with scores of ($M=4.2727$) at 85.4%. These were the success of the approach in providing a suitable environment for beginners (BNF 5) ($SD=1.00905$) and an easy way to perform tests anywhere at any time (BNF 7) ($SD=0.90453$). Following in close second, the ability to perform tests in the areas that they good at (BNF 3), large-scale work collaboration and communication (BNF 6) and give testers a personal sense of achievement (BNF 8). These three benefits scored the same mean score ($M=4.1818$, $SD=0.60302$) and percentage at 83.6%. From the remaining responses, the only benefit of a percentage of less than 81.8% was BNF 4 regarding the capability to achieve the tests in different ways compared to other benefits at 74.4% and with a mean score of ($M=3.7273$).

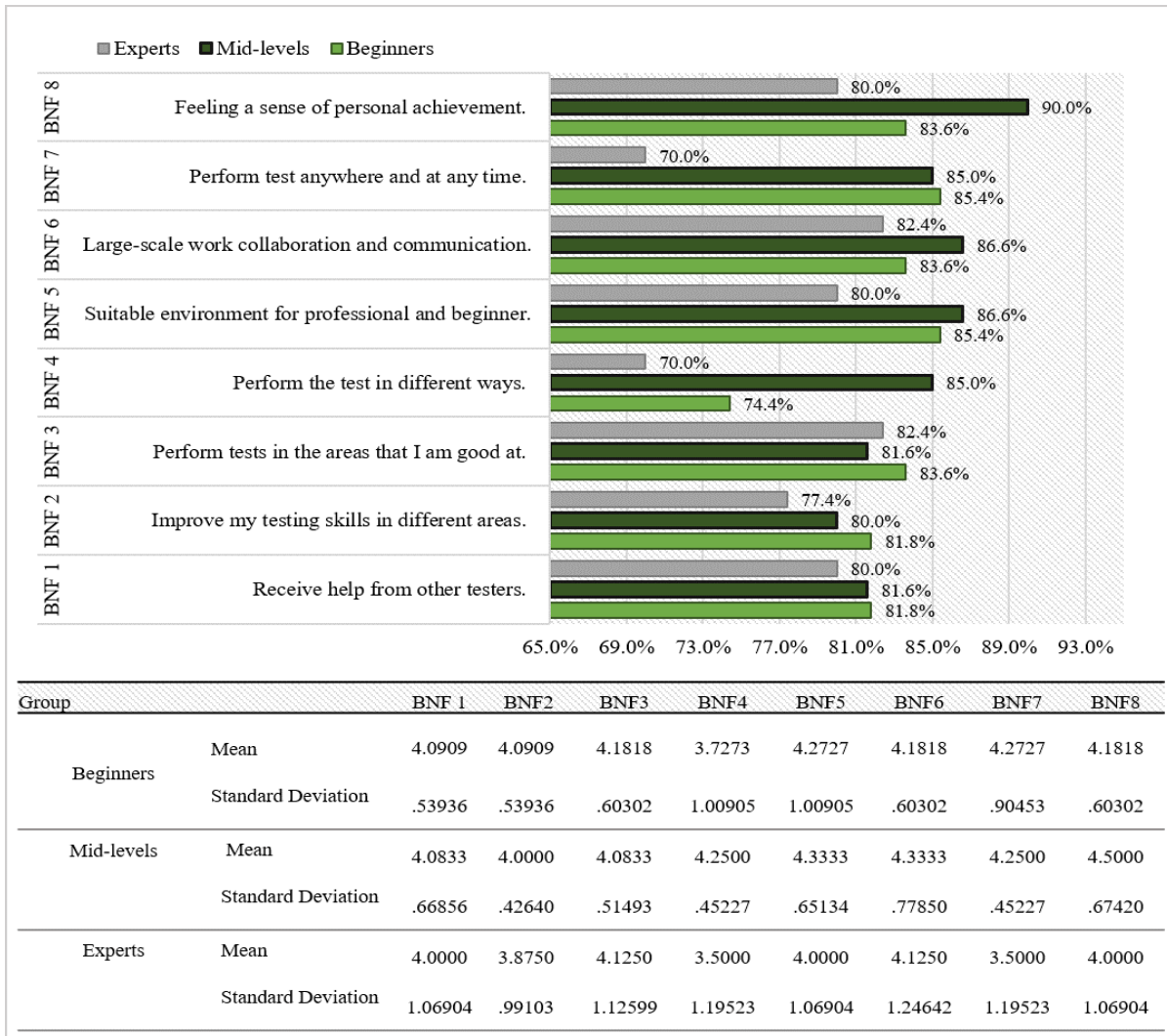


Fig. 8.7 The percentages and mean scores for the most important benefits received by Beginners, Mid-level, and Experts testers.

8.6 Results III: Satisfaction

This section describes the main results and findings in terms of the testers’ satisfaction with the use of the proposed testing approach. Firstly, it explains the overall approach satisfaction for all participated testers. Then, it explains if the proposed testing approach is more satisfactory for a particular group of testers than others.

8.6.1 Overall Satisfaction of the Approach

Table 8.8 summarises the percentages and mean scores of testers’ satisfaction with the proposed approach. The results show that overall satisfaction has reached a very high

percentage at 82.7% and mean score ($M=4.1398$, $SD=0.81547$). The results presented in the table demonstrate that among satisfaction about the usage, workflow and services provided, and benefits received from the approach, the satisfaction about the usage of the approach scored the highest percentage 84.5% with a mean value of ($M=4.2258$, $SD=0.84497$). The satisfaction about the workflow and services provided by the approach ranked the second highest mean value ($M=4.1613$, $SD=0.89803$) at 83.2%, while the satisfaction about benefits provided by the approach registered the lowest mean, but with a high mean value ($M=4.0323$, $SD=0.98265$) with a percentage of 80.6%. To summarize, the results show that testers are very happy and satisfied to use our approach in the future with a very high mean score of ($M > 4.000$, $SD < 1.000$).

Table 8.8 The mean score and standard deviation values of participant testers according to the "Overall Satisfaction".

Dimension	Question Code	Mean	St. Dev	Level	Percentage (%)
Overall Satisfaction	(SAT)	4.1398	.81547	High	82.7%
	SAT 1	4.2258	.84497	High	84.5%
	SAT 2	4.1613	.89803	High	83.2%
	SAT 3	4.0323	.98265	High	80.6%

Mean score's interpretation: (Low 1.00 – 2.33, Moderate 2.34 – 3.67, and High 3.67 – 5.00).

8.6.2 Satisfaction for Different Testers Groups

The independent t-test analysis results presented in Table 8.9 shows a very small difference between the mean score of Freelancers ($M=4.3125$, $SD=0.64943$), which represents 86.2%, and Employees ($M=3.9556$, $SD=0.95008$), which represent 79.1%. This difference is not statistically significant where $p\text{-value}=0.237 > 0.05$. For the "Workplace Size" variable, the results presented in Table 8.9 show a big difference in the mean of testers who work in Small Organisations ($M=4.4762$, $SD=0.57275$) and those who work in Big Organisations ($M=3.5000$, $SD=1.00791$). The t-test analysis indicates that this difference is statistically significant with $t\text{-value} = 2.257$ and $p\text{-value}=0.042 < 0.05$. This indicates that Small Organisations testers are much more satisfied with the approach with a very high percentage of almost 90% than Big Organisations with 70.0%. According to the One-way ANOVA analysis result presented in Table 8.10 no significant differences in the mean values of the three groups of testers, Android, iOS, and the experts in both OS were found in terms of the satisfaction ($F\text{-value} = 0.299$, $p\text{-value} = .744 > 0.05$). This demonstrates that all these three groups are satisfied with the approach with a percentage higher than 82.0%. Further, regarding the "Experience Level" variable, the percentages were different between the three groups where the testers with a mid-level of experience give the highest mean score ($M=4.5000$, $SD=0.54123$) and percentage 90.0%, followed by Beginners testers with ($M=4.1212$, $SD=0.68755$) at 82.4%, and the Experts

with the lowest mean value ($M=3.9250$, $SD=1.09018$) at 78.5%. Although there is a difference in the three groups' percentages and mean values, the One-way ANOVA analysis results showed no significant differences across Beginners, Mid-level, or Experts testers were found ($F\text{-value}=3.168$, $p\text{-value}=0.058 > 0.05$). To summarize, we can say that all testers groups are satisfied with using the proposed approach for large scale compatibility testing with a percentage higher than 78%, except the Big Organisations with 70.0% which also fall in the high level.

Table 8.9 Summary of the independent samples t-test results (regarding satisfactions)

Variable	Group	N	Mean	St. Dev	t-value	df	Sig.
Job Status	Freelancer	16	4.3125	.64943	1.228	29	.237
	Employee	15	3.9556	.95008			
Workplace Size	Small Organizations	7	4.4762	.57275	2.257	13	.042
	Big Organizations	8	3.5000	1.00791			

The difference is significant at the ≤ 0.05 level (2-tailed)

Table 8.10 Summary of the one-way ANOVA test results (regarding satisfactions)

Variable	Group	N	Mean	St. Dev	Comparison	Sum of Squares	df	Mean Square	F	Sig.
Operating System	Android	15	4.2444	.69541	Between Groups	.418	2	.209	.299	.744
	iOS	9	4.1111	.94281	Within Groups	19.532	28	.698		
	Both OS	7	3.9524	.97046	Total	19.590	30			
Experience Level	Beginners	11	4.1212	.68755	Between Groups	3.681	2	1.840	3.168	.058
	Mid-level	12	4.5000	.54123	Within Groups	16.269	28	.581		
	Experts	8	3.9250	1.09018	Total	19.950	30			

8.7 Open Question Analysis Results

This section presents testers' answers to the *open-ended* question regarding the other suggestions they would like to share to enhance the overall quality of the crowdtesting approach. This question was answered by a few participants; their responses included some positive and negative comments for the services provided and implementation of the approach in general. The positive comments were that the approach had covered all required features and essential requirements for compatibility testing; the simplicity and clarity of the interfaces designed; and the well-developed architecture and framework structure of the approach. In contrast, the negative comments also covered three small issues: improve user experience, adding some other features in the future, and improve the control quality method.

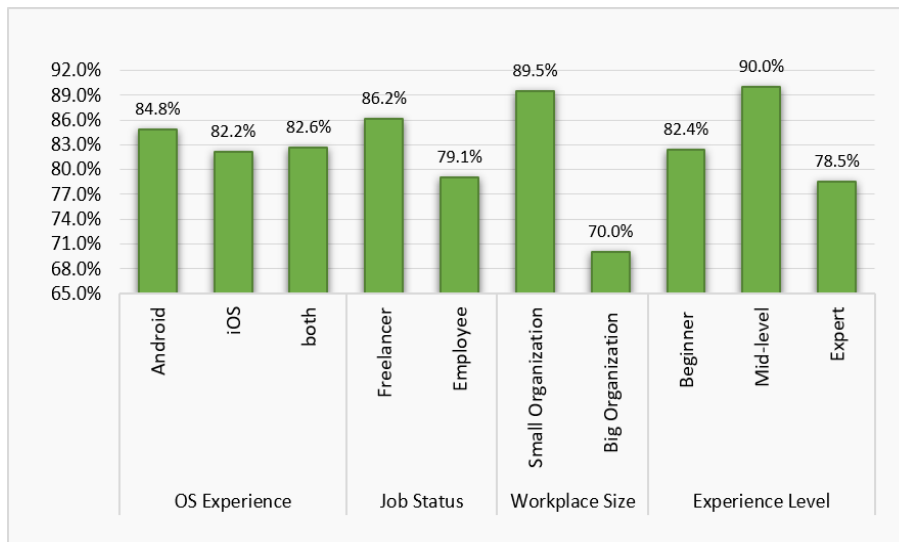


Fig. 8.8 Comparison between testers' group regarding overall satisfaction of the approach

8.8 Chapter Summary

This chapter presented the main finding and results of the empirical study related to the testers' experience with our crowdfunding approach. We described the demographic information of the participants' testers and their actual answers gathered by the questionnaire. Our results explicated that the approach success in performing the testing with a high effectiveness level. In particular, we determined that the approach's effectiveness in the ten defined processes ranges from 79.2% to 85%, which supports our approach as promising from the testers point of view. Similar to the developers in the previous chapter, the reliability and trustworthiness witnessed the lowest effectiveness percentage with 79.2%. Whereas, testers were pleased with the given feedback process with 85% and the results submission mechanism with 84.2%. This shows that our approach is helpful for testing and during the testing process. We extended our empirical evaluation study by measuring the benefits and satisfaction of our approach. Our results confirm that all testers groups received many benefits with a percentage above 79.5%, only big organizations with 74%. The results of the satisfaction provided high-level knowledgeable information about the satisfaction of different testers groups for using our approach for their testing in the future. We found that all testers group were satisfied with the approach, except the large organizations which less delighted with our approach. Finally, we presented the collected recommendations from the testers in respect of improving the approach. Similar to developers' advice in the previous chapter, the testers also agreed that the approach's security and privacy need to be improved. The next chapter will discuss in detail the main findings of the developers (presented in Chapter 7) and the main findings of the testers (presented in this chapter) to answers our research questions (outlined in Chapter 1).

Results Discussion and Related Implications

9.1 Introduction

This thesis has aimed to find and develop a new testing approach to overcome the mobile device and OS fragmentation issues and to perform effective compatibility testing. The previous chapters have described the methodology of collecting key requirements, implementation, and evaluation of the approach. In particular, the previous two chapters have explained the obtained results and main findings of the two empirical evaluation studies from both sides; those of the developers in Chapter 7 and the testers in Chapter 8. This chapter presents a discussion of the main findings of this dissertation divided into three main sections; each section answers one question from the main research questions presented in Chapter 1, Section 1.3, (RQ2, RQ3, RQ4, respectively). Section one discusses developers' and testers' main findings regarding the effectiveness of our approach by answering the set of sub-questions related to the effectiveness of the approach's internal processes to answer the general research question RQ2. Section two explores the perspectives of different developer and tester groups concerning the main advantages and the benefits of our approach that each group has received in particular, aiming to answer the research question RQ3. Section three explains the degree of satisfaction of all five different groups of developers and testers and their intention to use our approach in the future, answering the last research question, RQ4.

9.2 Effectiveness of the Proposed Crowdfunding Approach

In order to evaluate to what extent our public crowdfunding approach may be considered effective with respect to addressing the fragmentation issues of mobile devices and OS versions and in facilitating effective compatibility testing for both developers and testers. Based on the results gathered from developers (Chapter 7) and testers (Chapter 8), the analysis has been divided into ten sub-questions. Each of the sub-questions, A to J, provides evidence of how and to what extent our approach responds to previously unmet needs of developers and testers to perform effective compatibility testing. It also draws comparisons between the effectiveness of our approach and current state-of-the-art approaches. The answers of all these sub-questions will, in turn, help answer the second main research question (RQ2) regarding the effectiveness of our approach in addressing the fragmentation-induced compatibility issues and supporting effective compatibility testing. The sub-questions are organised as follows:

A: Is the proposed task defining and distribution mechanism effective?

Approach/Results Interpretation: Our findings demonstrate that our task defining and distribution mechanism is effective in saving developers time and effort. According to the developers' responses, the effectiveness of this mechanism is due to four main aspects: (1) effective task design; (2) providing information about the task complexity level; (3) sufficiency of the task requirements and clarity of the presentation; (4) and by making the test available to the public. Developers' responses in Chapter 7, Figure 7.2 (a), show that our task design method was effective with 80.0%. In our opinion, this might be because it allowed them to define their large apps as multiple small tasks from the beginning by themselves, by either putting them all under one project, clustering related tasks together, or defining them separately. **Comparison with the state-of-the-art:** This differs from existing approaches and industrial practices, which still face a great challenge to define the large projects into small tasks that can be easily achieved by testers (Alyahya and Alrugebh, 2017; Alyahya and Alsayyari, 2020; Lykourantzou et al., 2019). In such approaches, the micro-task decomposition activity involves many steps, which are often completed by crowd managers or leaders, which requires more time and effort, especially for large mobile app projects. **Lessons Learned:** Our findings imply that developers are more interested in defining the test by themselves rather than allowing them to be completed by intermediaries. One main reason can be the concern of missing testing some critical functionalities of the app and testing unnecessary tasks that do not affect the app's quality, which causes a delay in the testing process and poor test quality. Thus, this effective task design in our approach helps tackle this research gap and prevents taking a long time in testing unnecessary tasks (which is costly), and ensures that all parts of the app would be tested on all required mobile devices in a shorter time. This can also be another indicator of the improvements of our approach compared to other crowdfunding approaches.

Approach/Results Interpretation: Our task design strategy allowed developers to spend less time and effort in writing task specifications, this is evident by the very high percentage of developers' responses which stand at 85.2%. This is because our task design highlighted the necessary information identification that must be provided to the testers, and that which has been previously discussed in Chapter 4, to ensure there is no shortage in the information provided and that it is displayed clearly for the testers. This was evidenced by the high percentages of testers' responses on the clarity of the presentation and the sufficiency of the information and instructions given to perform an effective testing process, as outlined in Chapter 8, Figure 8.2 (b).

Comparison with the state-of-the-art: By comparing our approach to other approaches in the literature, we found that our task defining mechanism differs from the rest in terms of the quality of testing information provided and its clarity. This improved mechanism provides additional information for the testers, including a clear description of the app's overall objective, expected behaviour of the app functionalities under test, testing steps, expected results, and the required devices and OS versions, and complexity level of task, which would help testers select appropriate test and thus perform more effective testing. These improvements are recognised and supported from the evidence provided by testers in Chapter 8, Figure 8.2 (b), which received a high percentage regarding the sufficiency, clarity, and importance of the task requirements. As far as we know, most current state-of-the-art crowdtesting approaches do not provide all this information to their testers. This is supported by the conclusion reached by [Guaiani and Muccini \(2015\)](#) from the survey study conducted with crowd testers working in well-known industrial companies, where the interviewees pointed out that their companies do not provide them with enough information, and they indicated the need for receiving such information. Also, this is supported by the findings of our previous study (Chapter 4), which indicated the lack of enough information provided and a strong need for that. This is also consistent with the conclusion reached by [Alyahya \(2020\)](#) that the lack of sufficient information about the testing task in the existing crowdtesting approaches represents barriers to the successful practice of crowdtesting by testers and developers alike.

Lessons Learned: We found from our research, that the lack of providing testers with sufficient information makes the testers feel challenged to perform the test effectively, which mostly lead them to return to the leader or manager multiple times and inquire about the missing information that may impact their work performance and results, as well as delaying the testing process.

Approach/Results Interpretation: Regarding the issue of task complexity, [Kamangar et al. \(2019\)](#); [Leicht et al. \(2016c\)](#) stated that test tasks greatly differ in terms of complexity level and can impact performance in crowdsourced software testing. **Comparison with the state-of-the-art:** As far as we know, most of the current testing approaches do not provide any information on the complexity of the testing task, This may be why testers sometimes fail to conduct the tests in crowdtesting accurately. The conclusion supports this reported

in (Mejorado et al., 2020), which demonstrated that the task failure in crowdsourcing is mostly associated with the lack of information about the tasks' complexity. This is due to the fact that testers' experience is different, and some of them may feel during the execution that the test task is difficult, and they cannot complete it. **Lessons Learned:** This leads them to take one of the two actions: (1) leave the test and not participate anymore; (2) perform the test quickly and deliver poor results.

Approach/Results Interpretation: The findings demonstrated that our testing approach has succeeded in addressing this gap by providing additional information regarding task complexity level. The testers' responses in our study confirmed that determining task complexity level in our approach was very useful where it helped them select the appropriate tasks and supported them in carrying out the test easily and obtaining more accurate results with 83.2%, as illustrated in Chapter 8, Figure 8.2 (b). This might be because the task complexity level could provide them with an expectation of the effort and amount of time (task duration) that they will need to invest before accepting or selecting the task, which was proven to affect the success of performing crowdsourcing tasks in software development projects (Gefen et al., 2016); this also applied to tasks in software testing. According to Yang and Qi (2021), it was proven that the task duration affects crowd worker performance. **Lessons Learned:** To improve our approach, it would be better to explicitly provide the task duration time alongside the complexity level, thereby improving our task defining process. To advance the practice and theory of crowdtesting, we advocate developers to the necessity of identifying task complexity level and expected task execution duration during the task's definition. This will help testers select tasks compatible with their abilities to perform an effective testing process with more accurate results.

Approach/Results Interpretation: As for defining the test, the developers found that defining the test into two different testing levels (feature and code) and uploading the beta version, or apps' code was also helpful with a percentage of 77.6% to perform effective compatibility testing. However, the results (presented in Table 7.2) showed that a minority of participant developers are still concerned about this feature and sharing their source code. The reasons could be privacy and trustworthiness issues. **Lessons Learned:** Thus, by controlling the access and adding more restrictions for preventing the source code from being stolen by public testers, this feature will be improved, increasing the overall effectiveness of the defining method.

Summary: Overall, we can say that the effectiveness of task defining testing method has greatly helped distribute the tests on a large scale and execute them in a short time in comparison to other approaches. This is evident through the developers responses standing at 84.6% as shown in Chapter 7, Figure 7.2 (a). From a practical point of view, these findings will be a guideline for developers to develop effective task defining and distribution in the future and support current crowdtesting approaches and practices to improve their

crowdsourcing task design. It will also encourage other researchers and practitioners to continue exploring new features to improve task defining and distribution on a large scale.

B: Is the proposed task selection method effective?

Approach/ Results Interpretation: Overall, our findings presented in Chapter 8, Figure 8.1 showed that our task selection method is effective. This is due to the effectiveness of the task defining mechanism and its support in distributing the test on a large scale, which has facilitated testers to access and select tasks efficiently. **Comparison with the state-of-the-art:** Unlike task selection methods used by current crowdtesting approaches and practices that rely on assigning the tasks directly to testers through the platform or email such as platforms as pointed out by [Alyahya and Alrugebh \(2017\)](#); [Alyahya and Alsayyari \(2020\)](#); or suggesting a list of a few tasks based on matching tasks specification with testers profile (information registered by testers on their profile) such as device models, OS version, type of tasks performed (experience), and demographic locations as mentioned by [Gao et al. \(2019\)](#); [Liu et al. \(2019\)](#).

Approach/ Results Interpretation: Our approach has also used task recommendations based on similar criteria but based on matching with their previous work, not profile. Such a method will suggest a more relevant task to their work and performances due to that their profile information might change with time or may not update continuously. The testers' responses showed that this task recommendation method is effective in facilitating the task selection process and helping them to perform more tasks in a short time, with 82.4%, as shown in Chapter 8, Figure 8.2 (a). **Comparison with the state-of-the-art:** This is consistent with the conclusion in ([Kamangar et al., 2019](#); [Tunio et al., 2017](#)) that suggesting tasks based on the developers' (workers) experience is a good solution in the software development process because it may increase the task relevancy for developers. We agree that this may be a good solution. In our view, it may not always be suitable for testers because this may restrict them to a limited amount of tasks and types where hundreds of tasks posted daily and this allow them to access and perform few tasks only. **Lessons Learned:** Sometimes, the tester does not have enough time to perform the suggested task because the task may take more time than s/he has available, and thus s/he may search for a task that is compatible with their timeline ([Vaz et al., 2019](#)). This probably limits their choice reduces their motivation to participate or may lead to low quality if performed quickly, limiting test coverage. Such a method may not suitable for new testers because they do not have a record with enough data to match with.

Approach/ Results Interpretation: Our findings show that the proposed approach managed to successfully address this issue by allowing testers to access the main tasks' dashboard (including all posted tasks) and the freedom to choose the tasks they would like to work on among those available. The choice is based on the required skills and experience

that they have, their free time, and interest; regardless of recommended tasks based on their previous experience only. This is evident in testers' responses to the high effectiveness of this method with a score of 81.2%. **Comparison with the state-of-the-art:** Mao et al. (2017, 2015b) contradict this view claiming that the selection of tasks from a large set of tasks is very hard work and a time-consuming job. We agree, but at the same time if the selection process is well-controlled and advanced search features are implemented this would result in an efficient process.

Approach/ Results Interpretation: In our work, we controlled the selection of tasks by imposing some constraints, based on the requirements from Chapter 4, Section 4.7). Testers seeking prior permission of developers to perform the selected task and then informing developers of the number of testers who will perform the task. Our testers' responses are clear evidence of the effectiveness of this method which achieved almost the same degree of effectiveness with selecting from recommended tasks, despite it taking a little more time. **Comparison with the state-of-the-art:** Consequently, this shows the effective implementation of our task selection method, as well as providing further evidence of the capability of our approach to distributing the test on a larger scale, as compared to current approaches. This approach provides the opportunity for public testers to access and select tasks, and the degree of effectiveness is reflected in the results.

Summary: Based on all these results, we can conclude that testers are fine in selecting the tasks by themselves through either recommended tasks or searching the main tasks dashboard, rather than their being assigned and sent them directly by developers. This suggests that the developers and testing companies should provide testers access to different posted tasks in terms of time, money, types of tasks alongside the recommended tasks. This would help reduce the work complexity and encourage more testers, which in turn would accelerate the testing process and increases test coverage. This can also be applied in different crowdsourcing contexts to motivate more crowd workers.

C: Is the proposed testers selection method effective?

Approach/ Results Interpretation: In Chapter 7, Figure 7.2 (b), we showed that the selection process, in general, has achieved a high effectiveness percentage of 78.6%. This is due to the speed of accessing and selecting testers and the quality of the selection criteria, which is one of the important requirements highlighted by the developers in Chapter 4. According to the literature, the novelty was in thinking of a new approach to reduce the delays that occur in the testing process. As a result, our approach has relied on the direct interaction workflow where it has been dispensed with dealing with crowd leaders and managers, which are sometimes one of the causes of the delay in other approaches. Thus, this led to accelerating the selection process, reaching and inviting as many testers as possible to work on specific

tasks quickly with a percentage of 80.4%. **Lessons Learned:** This is consistent with what reported in (Zhang et al., 2018) in which direct interaction between developers and testers would lead to the fast selection of suitable testers.

Approach/ Results Interpretation: Regarding increasing the size of the group of testers, our method of suggesting qualified testers was based on four main criteria to characterize and identify the appropriate testers: testing context ¹ that helps to ensure coverage of all devices and OS versions), which was proven to be influential to the test outcomes (Mao et al., 2017; Xie et al., 2017). Testers capability ² and experience ³ (domain knowledge) that ensure the quality and diversity in the results, which were found to be critical to the test performance (Mao et al., 2017; Yang et al., 2016). The testers availability ⁴(tester absence or delay), which has been proven to affect the timeframe of the testing process (starting and completing the test) (Tung and Tseng, 2013) and the app development speed (Talby et al., 2006). The findings show that this testers suggestion method with all these four criteria were effective to a degree of 77.6%. On the other hand, providing developers with the information related to these criteria and enabling them to access it through the main dashboard to perform the selection process themselves was considered more effective, receiving an agreement of 79%, than the simple suggestion of a set of suitable testers. This is considered to be a critical requirement by developers as stated in Chapter 4. This is because when recommending testers, the app developers neither know who performs the test nor predict whether all the test contexts (device models and OS versions) can be covered for the tasks (Xie et al., 2017). Therefore, accessing such criteria makes them more confident about the testers selected and ensures coverage of the devices and diversity of the results. **Lessons Learned:** This implies that even developers are still concerned about working with the public and unknown suggested testers.

Comparison with the state-of-the-art: Compared to the other testers selection methods in the literature, we found that most of these methods are partially exploring the characteristics of testers, and they had not considered the four selection criteria together, especially testers availability. For example, Cocoon (Xie et al., 2017) had only considered the test context, MOOSE (Cui et al., 2017b) put more attention on the experience (domain knowledge), ExReDiv (Cui et al., 2017a) has focused on the experience and capability of testers, while MOCOM (Wang et al., 2019b) had considered the test context, experience, and capability of testers. On the other hand, by comparing our overall testers selection method of our crowdtesting approach with the other crowdtesting academic approaches and industrial practices, we found that the Baidu CrowdTest platform and other popular crowdtesting platforms ⁵ have mainly focused on selecting testers based on test context.

¹The device models and OS versions that tested in the previous works.

²The capability of testers extracted from their previous work.

³The knowledge and experience of testers in particular testing areas

⁴The time that testers can perform the test

⁵<https://www.softwaretestinghelp.com/crowdsourced-testing-companies/>

This is confirmed by the empirical study results conducted on crowd testers working in large testing organisations, which stated that they are selected for the test based only on their mobile device and the OS software version they own (Guaiani and Muccini, 2015). However, some of these platforms have considered the capabilities of testers. To the best of our knowledge, they still do not consider the availability and experience of testers (the testers are selected randomly). This is supported by (Wang et al., 2019b) who conclude that in the currently used selection methods most of the tester's experiences are not tightly related to the task's test requirements. **Lessons Learned:** Consequently, this lack of consideration on the testers' experiences and availability of testers will increase work pressure on the testers, which will lead them to quit or refuse the task (Yang et al., 2016). This would certainly reduce the number of selected testers and cause multiple delays while waiting for testers to be available (Alyahya and Alsayyari, 2020) and so affect the outcome and speed of performing the compatibility testing process. This is supported by Kuutila et al. (2020); Memar et al. (2020) whose work proved that performing the test under time restriction/pressure impacts the performance of testers and testing results quality.

Approach/ Results Interpretation: As our testing approach controls the overload work of testers and prevents recommending testers who have more than three active tests, we can say that our suggestion method has a very high probability of recommending the most appropriate testers for the performance of the test task and helps to minimize the total testing time, reduce time pressure on testers and provide good quality results within an acceptable timeframe. Our approach and testers' selection mechanism will also serve as a database that helps individual developers, small teams, or SME enterprises to access the testers and check their work quality and reliability level and so invite them to carry out their tests in their companies or anywhere else, even if they do not perform their test through our approach.

Summary: These observations provide clear evidence supporting the effectiveness of our testers selection method compared to state-of-the-art approaches and practices in both academia and industry.

D: Can the proposed approach reduce the time and effort for testers in submitting results and developers in aggregating test reports?

Approach/ Results Interpretation: The findings in Figure 8.1, Chapter 8, demonstrated the high effectiveness of our report submission mechanism in terms of simplicity, efficiency, and the ability to provide more accurate results, as well as its capability to motivate testers to perform more tests. This resulted from the implementation of the mobile device data detection service and automatic uploading of results, which facilitated the submission process by reducing the amount of text that the testers must enter in the submission form, more

specifically the precise information of tested mobile devices, which helped them submit more accurate results quickly in a short time receiving a score of 85.8%. **Comparison with the state-of-the-art:** This is different from the existing issue reporting mechanisms used by current crowdtesting approaches, in which testers manually enter all necessary information to test reports through the submission form within the online implemented platforms (Wang et al., 2018; Yan et al., 2015) or submitted as Excel files (Feng, 2019). **Lessons Learned:** This inevitably leads to more time and effort being required, as well as some human errors being evident when entering the information of mobile devices (e.g., model name, model numbers, or OS versions). The mobile device data is considered to be vital to developers when seeking for an issue or solution for a specific problem as mentioned in our previous study in Chapter 4.

Approach/ Results Interpretation: According to Figure 8.2 (d), Chapter 8, the findings also assured that this mobile detection service enabled testers to perform multiple tests and submit the results of the tests quickly with less effort at 83.88%. This emphasises the importance the testers attach to the issue of time and effort when considering contributing and performing more tests. **Lessons Learned:** This means that the more time and effort it takes to complete the process of submitting results, the more likely it is that the testers' enthusiasm to perform more tests decreases (Alyahya and Alsayyari, 2020). Based on this, we strongly assert that these two factors must be considered among the motivational factors in the crowdtesting process.

Approach/ Results Interpretation: Moreover, 82.4% of the responses from testers with different experience in our sample group indicated that it was easy and simple to use this submission mechanism, including the automatic detection service. This indicates that our submission mechanism is suitable for novices and professional testers alike. This can show our approach's success in overcoming the major challenge faced by testers in current crowdtesting approaches regarding the need for "A simple and intelligent reporting mechanism" that is suitable for different testers skills as highlighted by Alyahya (2020). The above findings also provide evidence that our submission mechanism has addressed the main requirements mentioned by the majority of our participants (86%) in our prior study in Chapter 4.

Summary: Based on these findings, it can be concluded that our issue reporting mechanism can be more effective in terms of providing more accurate results and reducing the effort and time involved in submitting results as compared with other existing submission mechanisms. Therefore, we suggest that researchers and industry practitioners think of more features that facilitate submission mechanisms and help testers submit testing reports easily with less effort and better results.

Approach/ Results Interpretation: To aggregate and track the submitted reports, in most of the existing crowdtesting approaches, the developers or middleman (leader or

manager) collect and organise reports manually using external issue-tracking systems, such as JIRA (Atlassian) ⁶, Bugzilla ⁷, Mantis ⁸, and Zoho ⁹ (Feng, 2019). Obviously, such processes may prove to be a much more time-consuming and challenging job (Hao et al., 2019; Huang et al., 2020), which could reduce the intended push to use such crowdtesting platforms. Our aggregation mechanism addressed this challenge by designing an internal report tracker system to automatically collect and organises testing reports based on submission time and sensitivity/priority of the issues without human effort; in order to standardise the access and avoid the use of multiple external systems, which in turn would be more efficient. According to Figure 7 (f), Chapter 8.2, the findings clarified that this aggregation mechanism was able to collect and track the testing reports effectively with 80.4% in agreement. As for filtering and viewing the content of the received reports, the developers were able to view, update, and filter testing reports easily with 78.2%. This high score may well be because of the quality of the set of filtering and searching criteria used, which relies on reports' status, such as new reports, opened, rejected, duplicated, waiting, and deferred, which helped developers reach and view the reports quickly. Regarding the efficiency, the results indicate that our issue tracker system can collect, track, and organise the testing reports in less time and with less effort at 74.6%. Although this percentage indicates a high effectiveness level, it does imply that the collection method needs to be improved slightly. The reason behind this might be the time and effort that developers or intermediaries spend manually triaging reports to identify the duplicate ones (Donepudi, 2020). **Comparison with the state-of-the-art:** Recent researchers dating from 2019 (Hao et al., 2019; Wang et al., 2019a, 2020; Yu et al., 2021) have argued that duplicated reports detection services are essential to facilitate the report aggregation process. **Lessons Learned:** Implementing this service would help reduce the time and effort required to identify duplicated testing reports and improve the effectiveness of our aggregation and tracking mechanism.

Summary: In general, the overall findings demonstrate that our aggregation mechanism and report tracking have improved existing aggregation mechanism in terms of: (1) eliminating the human effort needed for entering results manually; (2) reducing the time required for aggregating and organising reports; (3) avoiding the delay or problems resulting from missing some reports that probably were caused by manual collection and organising of testing reports by the developer or middleman (Alyahya and Alsayyari, 2020). We suggest crowdtesting practitioners put more effort into finding more features that can enhance the automatic aggregation mechanism and identifying the higher-quality reports from the submitted reports set. Moreover, all the above findings can prove that both the reports submission and aggregation mechanisms in our crowdtesting approach are superior to other mechanisms of

⁶<https://www.atlassian.com/software/jira>

⁷<https://bugzilla.mozilla.org/home>

⁸<https://www.mantisbt.org/>

⁹<https://www.zoho.com/bugtracker/>

current crowdtesting approaches in terms of the submission and aggregation of testing reports in a shorter timeframe with less effort. As most of the current crowdtesting approaches have not implemented the report duplicated detection method, as reported by [Hao et al. \(2019\)](#), implementing this service will significantly improve our methods compared to other approaches.

E: Will the proposed approach help in finding a variety of compatibility issues and reducing unnecessary cost?

Approach/ Results Interpretation: The evaluation findings have demonstrated that our approach is significantly effective at 80% in meeting developers' needs regarding discovering more useful results related to unexpected issues of the different architectures of mobile device models and OS versions and relevant solutions in the early stages of the apps development process (see Chapter 7, Figure 7.2 (e)). This is due to our testers selection method's effectiveness, which enables developers to choose suitable testers with experience relevant to the task and with different capacities (testing skills). This is because different testers who perform the same given testing task follow different testing strategies and can bring in very different test results, which is consistent with the conclusion made by [Chi et al. \(1981\)](#). Another reason might be that experts sometimes have a tendency to overlook simple and surface features and focus more on finer details ([Foong et al., 2017](#)). We argue that both experts and novices should engage in the crowdtesting process. Experts would do better in discovering compatibility issues related to source code, deep configuration of the hardware component, APIs, or OS version. Simultaneously, the novices would outperform them in discovering concrete issues related to specific physical mobile device components. This would ensure that the apps work seamlessly across all devices.

Comparison with the state-of-the-art: A prior research study conducted by [Donepudi \(2017\)](#) indicates that professionals/experts do not necessarily outperform beginners in performing tasks relevant to expertise. **Lessons Learned:** For instance, because testers' experience and their skills level vary considerably, some testers may detect issues, while others may not. This finding corresponds to the finding in [Cui et al. \(2017a\)](#) who stated that balancing the testers' experiences and their diversity in expertise with the test task could make a great deal of difference in the results and reduce the chance of discovering duplicated issues. This is because novice testers can detect simple issues while experts focus more on the deeper issues ([Donepudi, 2020](#); [Leicht et al., 2016c](#)). **Comparison with the state-of-the-art:** This finding is also consistent with what has been stated by [Liu et al. \(2019\)](#) that the performance of the test by traditional testers with similar skills and experience will lead to a bias in the results of testing, and thus the issues discovered would be one-sided. This is most likely because they usually follow the same testing steps, leading to difficulties in finding other testing results. This finding is also in line with a finding from 1988 ([McDaniel](#)

et al., 1988), which revealed that once a certain level of experience is reached, the additional experience may not make much difference in job performance. This refers to the importance of including testers with different testing skills (novices to experts) to perform the same task in order to discover more results from different aspects. **Lessons Learned:** Thus, increasing the participation of testers with different experiences and skills in our approach would lead to discovering more issues early and reducing their occurrence in the future or after delivering the app to the market. In current practices, the tasks are performed by a random set of testers who are good at testing and cannot guarantee the differences in the experiences or testing skills, resulting in a low issue detection rate.

Approach/ Results Interpretation: With reference to Figure 7.2 (d), it is evident that the method of tracking tasks' status and tested devices and OS version during the active test cycle also helps to obtain various test results, which has proved to be effective with a high percentage of 79%. Developers were able to track tasks and tested devices, which helped them to know whether the test has covered the requirements and thus adjust the requirements based on the devices that have not yet been tested. Obviously, this enables them to achieve better testing coverage of devices and reduces the chance of obtaining duplicated issues as well as reducing risks of wasting time and efforts. **Comparison with the state-of-the-art:** In current crowdtesting practices, monitoring test status is still regarded as a 'sophisticated' process because of the high cost and effort required. This is because their testing approaches rely on the presence of a middleman (leader or manager). Assigning the middleman to the tracking process will cost them more money (Alyahya and Alsayyari, 2020); they are just waiting for the middleman to collect the results, as testers are already assigned to the task previously based on their available devices where each tester test a specific device. **Lessons Learned:** This will limit the testing space and testers' range of experiences, which is the main reason for not gaining a variety of results.

Summary: Thus, we can demonstrate that our approach could be a powerful method for tracking tasks status and covering important test contexts, which help provide more useful results during the testing process, compared to other approaches.

Approach/ Results Interpretation: Regarding cost reduction, most of the existing approaches focused on minimizing the cost by reducing the number of testers through the testers' recommendation methods (Cui et al., 2017a), which affects the test outcomes and remains costly for many developers. Our findings, however, show that our approach has been able to reduce the cost without limiting the number of testers, or affecting the results, with a high percentage of agreement standing at 78.4% (see Figure 7.2 (h)); this is one of the critical requirements for developers (as discussed in Chapter 4). This figure is a result of our testing workflow's effectiveness in reducing the unnecessary costs that are paid to the leader or manager. The developers' responses indicate that our overall approach is cheaper

than other approaches with a percentage of 77.6% (Chapter 7, Figure 7.2 (h)). Another factor that reduces the cost in our approach is the easy access of developers to many devices and the ability to test them at a low cost, which achieved a high percentage of rating 79.4%. **Comparison with the state-of-the-art:** Moreover, according to [Ali and Dominic \(2016\)](#) who proved that knowledge sharing practice positively correlates with cost reduction in organizations. The knowledge sharing wiki could also play a significant role in reducing the cost. This is due to the fact that the use of the wiki in our approach leads to the ability to store a considerable amount of issues relevant to different architectures of mobile models, causes, and possible solutions that are discovered and solved by other developers. Thus, this helps developers avoid previous mistakes and overcome programming and testing challenges, leading to faster problem-solving and the opportunity to quickly finding solutions for the complicated issues they face. Furthermore, this allows them to take into consideration these issues early during the app development process and therefore prevent them from repeating the test multiple times on the same devices and thus reduce wasting time and the unnecessary cost in terms of the redundant tests, devices cost or expense for hiring testers to perform the test. A similar conclusion was reached by [Ali et al. \(2019\)](#); [Parekh \(2009\)](#); [Wnuk and Garrepalli \(2018\)](#).

Summary: To conclude, we can say that our approach most likely improves on other crowdtesting approaches regarding covering more devices and discovering more unexpected issues and cost reduction.

F: Is the proposed test workflow effective and easy to use?

Approach/ Results Interpretation: Our findings show that both developers and testers found the direct interaction workflow highly effective (Chapter 7, Figure 7.1 and Chapter 8, Figure 8.1). However, it has been more effective for testers at 80.8% than the developers at 77.8%. This is natural, due to the fact that testers sometimes may not have in-depth information about all functionalities and small details of the app, making it necessary for them to continuously communicate with developers to have a clear idea about functionalities and new features of the app, which would make their job faster and more effective ([Cruzes et al., 2016](#)). Regarding testers, the results clarified that they were able to perform much more effective testing through the direct interaction workflow. This is because of the sufficiency of the information provided with the defined task and ability to contact developers directly during the test to understand more about the test requirements if they are not clearly defined, as confirmed by their responses standing at 76.6%. This might be due to the developers who divided the tasks and defined the project immediately to testers because they have in-depth information about the project's objectives; whereas the intermediaries (middlemen) may omit the provision of some essential information about the new features to test or separate the related app functionalities in different test tasks, which may result in misunderstanding the

overall idea of the app and result in poor test results. The testers' results also showed that this workflow and increasing the strength of the connection and interaction with developers and the other testers community highly motivated them to participate in perform more tests with 85%; this is a good indicator of the ability of our approach to increase test coverage.

Comparison with the state-of-the-art: This finding is consistent with what has been found in a previous study ([Guaiani and Muccini, 2015](#)) in which the crowd testers surveyed pointed out their desire for more interaction and communication with developers to improve their work and have mentioned that this is one of the issues they face in their crowdsourced testing companies. **Lessons Learned:** Overall, these findings elucidate that testers preferred to communicate directly with developers rather than a middleman to perform more effective tests.

Approach/ Results Interpretation: At the same time, the developers were able to clearly understand testing results at 77.0%. This is perhaps due to the fact that the developers were able to see all the relevant information and explanations of the reported issues. **Comparison with the state-of-the-art:** This is often not the case in other approaches in which the intermediaries who manage the crowdtesting process inspect and extract all reported issues and summarise them before submitting them to the developers ([Alyahya and Alsayyari, 2020](#); [Donepudi, 2020](#)). Although this will save developers time, it may lead to intermediaries with less knowledge not reporting some issues or essential information that can help understand the results and consequently leads to an unfair evaluation process. Statements from intermediaries who manage crowdtesting stated in [Leicht et al. \(2016a\)](#) supported this by stating that sometimes they face problems in the management process of submitted issues due to the fact that they do not know enough information about the objective and the test scope, which may often require them to receive further support from the companies' test manager during the tests. Furthermore, interviews with the main test managers revealed that they also did not know how to frame the tests to use crowdtesting, and they were not sure about the specific tasks they had to perform in crowdtesting. This is also supported by [Cruzes et al. \(2016\)](#) where the interviewed companies clarified that the use of JIRA/user stories were not enough to avoid misunderstandings of issues. **Lessons Learned:** They found that developers strongly preferred direct communication with testers over indirect communication; these findings are also confirmed by our two evaluation studies and our previous study in Chapter 4 where most of our surveyed developers (80%) have strongly emphasised the importance of directly interacting with testers. These findings lead to the conclusion that the testing companies should: (1) make the developers responsible for dividing the project into simple tasks and defining the requirements to the testers directly; (2) find ways to keep the direct communication channel open between developers and testers to gain more support from the developers during the testing process; (3) know more about the competences and test management skills of the intermediaries so that they can

employ someone who has enough knowledge about the objective of the app and testing process.

Approach/ Results Interpretation: According to Chapter 7, Figure 7.2 (i) and Chapter 8, Figure 8.2 (g), our evaluation results emphasised the effectiveness of our testing workflow in reducing the delays caused by the middlemen figures (managers or leaders) from both sides, the developers at 78.8% and testers at 80.8%. **Comparison with the state-of-the-art:** Unlike our approach here, many current crowdtesting approaches involve the overall crowdtesting process mainly being managed by an intermediary (Leicht et al., 2016a), and as a result the testing process takes more time than needed (Alyahya and Alsayyari, 2020). This is directly in line with previous findings of Cruzes et al. (2016) who stated that the usage of a broker between developers and testers creates distances between them and more waiting time to answer their questions and queries. This shows that our approach is capable of accelerating the testing process compared to other crowdtesting approaches that rely on an intermediary. **Lessons Learned:** According to the previous discussion in task defining (sub-question A) and results submission (sub-question D), our findings have shown that both developers and testers prefer to do their work independently. For example, developers prefer to select testers and define tasks in a small set of tasks, and the testers would like to search and select the appropriate tasks on their own. This indicates that there is not much need for intermediaries and that dispensing with intermediary's presence is unlikely to cause any problem in the testing process; it would in fact speed up the testing life-cycle. Therefore, we argue that testing organisations need to decrease the communication gap between app developers and testers in order to build a working environment in which testers and developers trust each other.

Approach/ Results Interpretation: Concerning the ease of use of the overall approach and ease of interaction with this workflow the findings of Chapter 7, Figure 7.2 (m) and Chapter 8, Figure 8.2 (k) showed the endorsement of both developers and testers with regard to the ease of use of our approach and its workflow in their daily work routine and industry practices. However, it was regarded as more practical for testers with 81.2% than developers at 75.8%. **Lessons Learned:** The dispensation of intermediaries could be the reason for this discrepancy as the developers or companies are accustomed to dealing with the intermediaries to assist them in distributing tests, selecting potential testers, and collecting results from (Alyahya and Alsayyari, 2020; Donepudi, 2020; Leicht et al., 2016c) and therefore, the absence of these intermediaries may somewhat increase their responsibilities. This is perhaps a fact of human nature and a new approach is not easy as it is simply unfamiliar when it is implemented for the first time.

Approach/ Results Interpretation: The results also showed the high percentage regarding the ease of use of our approach ease of use by Android and iOS developers and testers with over 81%. This percentage includes responses from developers and testers who have different backgrounds, levels of experience, work status, and different mobile platforms

experience (Android and iOS), as shown in our samples' characteristics Chapter 7, Table 7.1 and Chapter 8, Table 8.1. This implies that our crowdtesting approach is highly suitable and effective for the different groups of developers' and testers' work environments.

Summary: all the findings discussed above provide clear evidence of the effectiveness of our approach in filling the literature gap (Chapter 1, Section 1.1 (G5)) and meeting the desires of developers (discussed in Chapter 4) in providing an easy and effective testing workflow that supports direct communication and collaboration between developers and testers, accelerates the testing process, and is suitable for the daily work routines of different developers and testers contexts. Consequently, this strongly demonstrates the outperformance of the proposed approach compared to other state-of-the-art crowdtesting approaches.

G: Is the proposed test reports quality evaluation method effective?

Approach/ Results Interpretation: Similar to most of the current crowdtesting approaches listed in (Alyahya and Alrugebh, 2017; Alyahya and Alsayyari, 2020) and that use the manual method of test reports quality evaluation with the help of the crowd leader, our approach also used a manual inspection and evaluation method, but this was performed by the developers themselves. The reason for developers performing this is the fact that they have more resources and knowledge in order to provide a good explanation of issues and steps taken than a leader or manager, and that would probably guarantee a fair and efficient evaluation process. According to Figure 7.2 (g), Chapter 7, our findings demonstrated that our results' evaluation and quality control method were effective in terms of time, effort, and ease of use of the evaluation process. The developers were able to complete the evaluation process of testers for all performed tasks very fast and with less time and effort at 78.2%. This might be because of the sufficiency of the evidence provided by testers within the submitted results, as the developers have confirmed that such pieces of evidence helped them gauge the accuracy of the results and accomplish the evaluation process quickly with 75.8%. **Comparison with the state-of-the-art:** The is directly in line with what was reported in (Chen et al., 2020), in which it was reported that some submitted reports by current crowdtesting practices lack the complete and necessary information of the revealed issues that could help developers better understand and fix issues, thus adversely affecting the efficiency of report inspection and evaluation process.

Approach/ Results Interpretation: Although less time and effort is required, the developers found that our quality evaluation metrics can achieve a moderate fair evaluation 73.4%. In our opinion, the possible reason for this could be the additional evaluation metrics used in our approach (correctness and completeness of results, on-time submission, numbers of rejection time, task complexity level, and additional information provide). **Comparison with the state-of-the-art:** Such metrics have probably not yet been considered currently

by most of the existing evaluation methods, according to recent summarised evaluation metrics provided by [Chen et al. \(2020\)](#). Based on that, developers have felt doubt whether these new evaluation metrics will achieve a fair evaluation process and the testers will agree with it.

Approach/ Results Interpretation: In this aspect, the testers confirmed that our evaluation method and its quality metrics could achieve a very high rate of 83.2% in terms of fairness. In our opinion, this might be because of two reasons: (1) developers have a large amount of knowledge and resources to evaluate testers with different experience levels and understand their interpretation correctly, which would help to solve the problem of trustworthiness in evaluation ([Sánchez-Charles et al., 2014](#)); (2) providing them with the obtained percentage for each of the quality evaluation metrics, which will be as guidelines that would help them understand any weaknesses in their work and improve their performance in subsequent tasks. Consequently, since the testers found that these metrics are effective, this is a strong indicator of our evaluation method's fairness and its metrics; because they are the ones who receive the monetary reward based on these evaluation metrics. **Comparison with the state-of-the-art:** Recent research has focused on finding automatic methods to analyse contained contents of test reports generally (quality assessment of textual documents) to help developers reduce the cost and time of test reports evaluation, according to the recent research of ([Chen et al., 2020](#)). Our findings will help researchers to consider that automatic evaluation can only guarantee a minimum quality ([Sánchez-Charles et al., 2014](#)). **Lessons Learned:** From this we discover that it is important to use both combination of the automatic and manual evaluation by developers; this will save developers time, give high quality and fair evaluation process, and make testers more confident about the evaluation feedback, as confirmed in our results.

Approach/ Results Interpretation: The testers' responses also revealed that the implemented report's evaluation status tracking service enabled them to know the evaluation's results early with 76% in agreement with this. **Comparison with the state-of-the-art:** Unlike our approach, other approaches require testers to wait a long time until they receive feedback of the evaluation by the crowd leader ([Alyahya and Alsayyari, 2020](#); [Donepudi, 2020](#)), which leads them to repeat the same mistakes in other tasks. Other evidence on the effectiveness of our method is the dashboard of testers' results quality distribution submitted for each task which helped developers to understand the reason for low quality work with 80%, as clear from their responses in [Figure 7.2 \(g\)](#). As far as we know, this is the first crowdtesting approach that implemented such a feature. In our point of view, providing such information leads to the evaluation and quality control process improvement and will also add value to the current crowdtesting platform.

Approach/ Results Interpretation: As for the feedback provided to the testers after the evaluation process that related to their work quality, the findings in [Chapter 8, Figure 8.2 \(e\)](#) showed that testers took advantage of the feedback provided regarding the analytical

summary of their activities and work quality in enhancing their performance in crowdtesting tasks with 84.4%. This may be attributed to their uncertainty, as in cases where the testers are unsure whether they have performed a particular test correctly and provided a clear, sufficient, and an accurate explanation of the testing results. Therefore, providing feedback in our approach helped them to avoid repetition of the same mistakes and improve performance and complete the testing tasks more accurately and faster at 83.8%, as shown in Chapter 8, Figure 8.2 (f). **Comparison with the state-of-the-art:** This is consistent with an early study from 1990 by Earley et al. (1990) in which feedback is usually used to encourage individuals to improve their performance. This is also consistent with the recent study by Lim et al. (2021) which proved that feedback improves the quality of crowdsourcing tasks' outcomes. **Lessons Learned:** Based on these findings, we can say that feedback is a reference or source that enables the testers to have continuous knowledge of their work quality, enhancing their work performance in the future. This is directly in line with the conclusion drawn by Kam et al. (2017). Here they state that when the crowd workers receive direct feedback about the effectiveness of their work, they would obtain knowledge about the correctness of their results.

Approach/ Results Interpretation: Our findings presented in Figure 8.2 (f) demonstrated that providing evaluation feedback in visual form (which consists of dynamic bars and graphs with different colours) and textual form (which contained words such as "Very Poor" to "Very Good"), helped testers to gain more knowledge about the accuracy of their tests and the quality of their work performance. Thus, this, in turn, increased their competence and satisfaction in participating and working at 86.4%. This is also supported by the correlation analysis results that show that feedback given strongly and positively relates to the testers' motivation. **Comparison with the state-of-the-art:** This is consistent with what has been found in previous research by Riccardi et al. (2013) in which the textual and visual feedback has a significant positive motivational effect on the improvements of work quality and performance. A similar conclusion to our results was also reached by Barashev and Li (2019); Zhou et al. (2020) in which the multiple direct feedback provides a sense of competence, recognition, and achievement in workers.

Approach/ Results Interpretation: The findings showed our evaluation and quality control method has increased testers commitment and desire to keep participating not only because of the feedback given but also because of the information provided about quality control and evaluation metrics that increased their confidence about the fairness of the evaluation, as confirmed by their responses with 83.2% in Chapter 8, Figure 8.2 (e). **Lessons Learned:** This indicates the importance of providing information about quality evaluation metrics used to testers in increasing enthusiasm and commitment to work, increasing test coverage and productivity. **Comparison with the state-of-the-art:** Based on our review of the available literature, we found that to date, most of the current crowdtesting approaches and well-known industrial practices do not provide information about metrics used for the

evaluation and quality control for their testers. This is also confirmed by the surveyed testers' responses in a previous study conducted by [Guaiani and Muccini \(2015\)](#) who express that the crowdtesting companies they work with (e.g., Clariter, Utest, Telecom Italia, Pass Brains, and Bug Finders) do not provide them with the metrics used for the evaluation and quality control. From this, we can conclude that our evaluation and quality control method can enhance current practices in motivating more testers, increasing their job satisfaction and sense of achievement, providing them with more information about how good the results of their test are and performance, which helps them to perform a better test. We appeal to the crowdtesting companies and developers to consider that one way to motivate testers to improve their work performance is by giving them more knowledge about their task accuracy and information about the reports evaluation and quality control metrics they use.

Summary: The findings of this research should bring to light the need for practitioners and researchers to further investigate how to provide an in-depth understanding of the importance of evaluation and quality control metrics and their influences to improve testers' performance in the crowdtesting process. Based on our overall findings, we can argue that assessing test report quality by developers is practical and would lead to a more efficient and fair evaluation process guaranteeing the different quality levels of testers' work and results, as well as improving job satisfaction and testers' confidence in the evaluation results. Implementing the automatic textual quality assessment service mentioned in ([Chen et al., 2020](#)) would save developers' time and effort; thus, improving the effectiveness of our evaluation method and the overall effectiveness of our approach.

H: Are the proposed trustworthiness/reliability management and privacy control methods effective?

Approach/ Results Interpretation: One of the difficulties in current crowdtesting lies in reducing developers' concerns in terms of distinguishing between trusted and not trusted testers ([Alyahya, 2020](#); [Liu et al., 2019](#)). The overall findings in Chapter 7, Figure 7.2 (j) reveal that our trustworthiness management was able to reduce developers' concerns about working with public testers at 76.4%. This could be because of the effectiveness of our testers reliability classification method, where we defined the reputation and reliability level for each tester according to the quality of all works in each testing area. **Comparison with the state-of-the-art:** This is unlike most of the current crowdtesting approaches that generate a reputation score for each tester based on the quality of all works submitted generically ([Alyahya and Alsayyari, 2020](#); [Liu et al., 2019](#); [mycrowd, 2021](#)). Using the dynamic graph and the colour-coding specified to each reliability level to distinguish between trustworthy testers (as explained previously in Chapter 5) can also be another reason for reducing their concerns. Thus, our method would enable developers to easily distinguish

between trusted and not trusted testers in particular areas compared to existing methods. This reliability classification method has also proven its effectiveness from the side of the testers, in which they found that it can provide them with an acceptable reliability level up to 81.2%. This refers to the quality of our criteria that were considered for calculating a fair trust value, which would improve their reputation among developers and testers in the global community at 77.4%, as shown in Figure 8.2 (h).

Comparison with the state-of-the-art: When we compared our trustworthiness management method with other methods used by current approaches, it has been found that if a general reputation is assigned for each tester, they will automatically be trusted, as s/he may have a low trust value for a specific type of test (e.g., complex tasks, sensing areas, or any other testing areas). This method may not always be the best; it depends on the context of use (Hromic, 2018). Therefore, our method might be better for such a case. However, although developers found that our classification method can reduce their concerns, they found that in terms of fairness, our method can achieve a moderate level up to 72.8%. The possible reason might be that sometimes there are some developers with a very high reputation, but they frequently like to work in a few particular areas for some reasons. Thus, classifying them as reliable testers according to only that areas could exclude them from working on other areas, which would be unjust. **Lessons Learned:** Based on this, we found that it would be much better if trustworthiness is managed in two levels: (1) generic reputation as in current crowdtesting approaches and industrial practices (2) then classifying each tester's reliability level according to the areas they are professional with. This would improve the fairness level of testers classification and the overall effectiveness of our trustworthiness management.

Approach/ Results Interpretation: As for privacy and protection control, our findings in Figure 7.1 and Figure 8.1 showed that our approach is effective in protecting the test environment in terms of task privacy, reporting mechanism, and controlling non-active/productive testers. However, it was more secure from the testers' side with around an 8% difference evident. This is natural because developers are always more careful about app programming privacy and its code, making them discover more privacy and protection gaps. The findings also showed that developers still have concerns about the capability of our approach in protecting the privacy of their published tasks and prevent the arrival of unauthorized testers from reaching them, as evident from their responses in Chapter 7, Figure 7.2 (c). One reason for this can be that they sometimes develop apps for specific sectors or clients, and therefore, sharing sensitive information with anonymous testers can pose risks and may lead to data security and intellectual property issues (Abhinav et al., 2015), which is also supported by the responses of participant developers in our previous study (in Chapter 4).

Approach/ Results Interpretation: The testers' responses achieved a high rate of 82.4% and eliminated this concern by confirming that they could not access the full test requirements description of the tasks that developers defined as private tasks. Even in

respect of the detected issues and results' submission mechanism, the testers clarified that our approach was able to provide a highly secure linking of detected mobile data with the submission form, as shown in Chapter 8 Figure 8.2 (c). This means that our approach protects the information of the task and information related to results submissions. Thus, this demonstrates the effectiveness of our protection method. Moreover, results in Chapter 7, Figure 7.2 (c), showed that our approach could improve on other approaches in protecting the testing environment and detecting non-active testers to around 76.2%, as compared to most of the current big testing companies mentioned in (Alyahya and Alsayyari, 2020) and that still do not consider this to be such an important feature. Only a few platforms such as Test IO and Rainforest took into consideration non-active testers and such a lack in most of the current approaches will lead to delays or reduce the test's quality. For improving practical use, our findings will help developers and crowdtesting companies generate different privacy protection scenarios to ensure that the testing task's intellectual property is not stolen and utilised by anonymous testers and for controlling the non-active testers.

I: Is the proposed incentivisation (compensation) method and its schema reasonable?

Approach/ Results Interpretation: Incentivisation and rewarding mechanisms are critical factors that could affect the effectiveness of the crowdtesting approach. Usually, crowd testers work for financial rewards. Unfair reward value may lead to low tester motivation, minimal tasks are testing (insufficient testing), and overlooking the detection of further issues (Deak et al., 2016; Gao et al., 2019; Yan et al., 2014; Yang and Qi, 2021). Overall, our findings in Chapter 7, Figure 7.2 (k) and Chapter 8, Figure 8.2 (i) demonstrate the effectiveness of our incentivisation and rewarding mechanism in tackling this gap and motivating testers in performing the test from the sides of both, developers and testers. Our approach meets all testers and developers requirements presented in Chapter 4, in respect of providing an effective and fair rewarding mechanism, letting testers know more information about their reward, and facilitating the rewarding process via controlling the payment status and providing developers with a list of non-paid testers. However, it is seen to be much more effective for the testers at 83.0% than developers at 74.2%. We believe this is due to the fact that the developers do not consider the testing as a serious job, and they are unaware of the optimal reward models or financial reward value to motivate testers appropriately. There appears to be a deficiency in the literature regarding the selection of the better reward (incentive) model that provides a fair reward value, and this represents a major concern for the developers and even crowd testers until now (Alyahya, 2020; Gao et al., 2019; Knop and Blohm, 2018; Sari et al., 2019). This is also supported by the results of our prior exploratory study presented in Chapter 4 which demonstrated that a large group of developers believe the crowdtesting process is simply a matter of cooperation, and

providing non-monetary rewards is appropriate to motivate testers rather than paying money. Since our incentivisation mechanism mainly focused on monetary reward, they found the incentivisation mechanism is moderately effective and fair with almost 72%. This is most likely why the overall incentivisation mechanism was approximately 10% less effective from the developers' side than that from that of the testers. On the other hand, our results show that our incentivisation mechanism enabled developers to effectively follow up all testers' reward status, especially those who have still not been paid after completing the tasks, which has achieved a high effectiveness level of 78.2%.

Comparison with the state-of-the-art: Compared to the existing rewarding models used by current crowdtesting approaches and industrial practices, our findings indicate that our model is much more effective than these models. This is because most of these approaches and practices, if not all, seek to achieve profit goals and tend to reward testers economically, which has constraints in terms of rewarding and motivating the testers. In such approaches and practices, the rewarding models are particularly based on: (1) the competition model of *Best-gets-paid*, which may be unfair because most of the testers perform the test, and only the best will be rewarded, and the rest are not rewarded (Muntés-Mulero et al., 2013), and this could reduce the motivation of testers to participate; (2) *Pay-per-hour* or *pay per-day*: these two models could also be unfair as they do not take the amount of testers' work and quality of work into account, thus reducing testers motivation; (3) *Pay-per-amount of work* done which also does not take the quality of testers work into account (Muntés-Mulero et al., 2013), which possibly lead to low result quality; (4) *Pay-per-quality of work*, in our view, this is probably the best reward model compared to other available models as it incorporates the quality of work and guarantees all testers being paid, thus motivating testers and ensuring the quality of results. However, because our aim is to motivate the testers more to perform more testing on a larger number of devices this might not be the optimal model in our context due to not considering the number of tested devices.

Approach/ Results Interpretation: Our rewarding model differs from these rewarding models, as it focuses on pay-per-device that has been tested for each task, taking into account the quality of testers' work, where each tester will be rewarded for each tested device based on the work quality. This will significantly motivate testers to perform more tests on more devices and perform more effort in order to get a higher quality results, compared to other models. Evidence of that is the very high percentage rating of 86.4% from testers, as shown in Chapter 8, Figure 8.2 (i). To the best of our knowledge, pricing the testing tasks and identifying a fair monetary reward value is still a great challenge in the current crowdtesting platforms and industrial practices (Alyahya, 2020). This is because most of them determine the reward value based on the reputation of testers and/or the number of issues detected only even large companies like UTest, without considering the effort of testers (Gao et al., 2019; Zogaj et al., 2014). **Comparison with the state-of-the-art:** Given the low variance in the reward values of the testing tasks in current crowdtesting approaches and industrial

practices, our testing approach focuses more on identifying reward values based on the effort of testers' work, including several essential criteria needed to identify a fair reward value to be paid for the experienced, unexperienced, or testers with lower skills in general. These criteria include: on-time completion of the task; the complexity level of the task; testers' experience; the importance (sensitivity) of the issues; additional results or suggestions; and the quality and accuracy of the results. In this respect, according to the testers' perspectives, our results show that these criteria are highly effective in identifying fair financial reward values that reflect the testers' effort in crowdtesting work with 79.2%. **Lessons Learned:** This will motivate testers to make more effort to discover more hard-to-detect issues although this takes more time to discover because there is a guarantee that they will get paid for their efforts invested in detecting the issues.

Comparison with the state-of-the-art: The issues above are aligned with the results of the large study that was recently conducted by [Gao et al. \(2019\)](#) on crowd testers who are working on current big crowdtesting platforms, which indicated that 40.3% of the crowd testers were dissatisfied with the reward value they receive. This is because this value is not commensurate with their effort exerted since they only get paid a fixed value for the issues they found and not the work and effort they put into finding the issues. Unfortunately, this would lead them to concentrate more on discovering easy-to-detect issues instead of investing tremendous effort and time finding hard-to-detect issues, potentially valuable and impacting the app's functionalities and test quality. This is because they do not receive adequate reward value for the effort they put in to detect such issues. **Lessons Learned:** Our findings strongly indicated that including simple information about these criteria with the payment value within the rewarding form is important to increase the testers' confidence about the reward value and evaluation process and will consequently lead to better job satisfaction and more motivation. This is evident in the testers' responses, which achieved a very high percentage of 83.2%.

Summary: Based on all above findings and comparisons with state-of-the-art motivation mechanisms, it could be argued that in respect of the motivation of testers, our incentivisation and rewarding mechanism are practical and much better than those used by other crowdtesting approaches. In practice, crowdtesting should allow testers to understand the rewarding process easily and adequately to perform the tasks with high quality. Without a doubt, if the efforts of testers are not appropriately valued based on the criteria mentioned above, and included in the rewarding schema (since these are the most critical work-related factors in this context), the fair reward value can not be accurately identified. Thus the motivation of testers to work harder will undoubtedly be diminished. Besides, if such intrinsic incentives are not included, testers' enthusiasm to perform more tests in the future might also decrease.

J: Is the knowledge sharing environment (wiki) effective?

Approach/ Results Interpretation: The findings of this research demonstrate that both developers and testers found the knowledge sharing wiki generally effective, as shown in Chapter 7, Figure 7.2 (l) and Chapter 8, Figure 8.2 (j). However, it has been more effective for testers at 81.4% than the developers at 75.8%. This might be due to the auto-documentation mechanism (as discussed in Chapter 5). Our goal for the auto-documentation was to reduce the manual effort and facilitate the storage process as much as possible. Furthermore, we sought to prevent errors of storing issues related to a specific device model under another category or topic by mistake. This because manual tagging sometimes leads developers or testers to assign incorrect tags to the issues, which can affect the searching process and, in turn, confuses the developer when searching for solutions for their discovered issues (Sonam et al., 2019). We believe that this method was not well suited to the developers, having achieved a moderate effectiveness percentage of 72.9% compared to the testers at 81.8%. This may be because the developers are more concerned about the details of the test, including the precise details of the documentation of issues, causes, and solutions, compared to testers. To enhance the developers' experience, it may be better if we suggest a set of categories based on the content of the testing task information. Once the developers define the tasks, they can choose the categories where they would like to store their tasks and their relevant information under these suggested categories. Consequently, this will avoid having duplicated categories and increase the accuracy levels of the searching mechanism. We believe that if the documentation process were improved, the effectiveness of the searching mechanism and information retrieval process would also be increased, which depends entirely on the method of storage. Thus, the overall effectiveness of the knowledge sharing wiki would also be increased.

Approach/ Results Interpretation: As for the importance of the stored information, the developers found the stored issues and their causes, and possible solutions are essential and useful in helping them answer their questions and develop high-quality apps rating these aspects at 77.6%. However, the testers found the stored testing steps and scenarios much more effective as a means of improving their testing strategy in different testing areas returning a very high percentage of 82.5%. We expected to obtain such results because the wiki is still new, and there is not a massive amount of information documented that helps developers or testers when searching with different searching criteria. Thus, with frequent use, more testing results and scenarios will be stored, and there could be a greater possibility of getting more results when they search. Thus, the overall effectiveness of the wiki and the usefulness of the documented information will also increase.

Comparison with the state-of-the-art: Compared to the current testing approaches and practices, none of them has a public place (wiki) in which to share and store the results of

compatibility testing which global developers can access. Their results are internally stored in a private space of a company, only community members within the company will be able to access these. This is one of the reasons for the existence of the fragmentation_induced compatibility issues so far, which has been represented in the motivations (G3 and G4) and aims (AM1, AM5) of our research (see Chapter 1, Section 1.2 and 1.1). According to the literature, there are some well-known knowledge sharing websites or repositories that, to some extent, addresses the testing issues and these include: Stack Overflow¹⁰, Stack Exchange¹¹, and GitHub¹². However, these sites or repositories do not collect and document the information in a structured manner, where each issue related to specific mobile devices or OS versions, their causes, and how can solve them, as well as the testing scenarios performed to discover them are stored in one place. It is just a questions and answers forum; the questions and their answers are distributed under different tags. Stack Overflow has attempted to develop a Wiki to store and organize all programming issues and their solutions in one place. Although the results were encouraging, and many developers found the beta version of this Wiki is useful, they failed to continue developing the wiki and decided to shut it down on August 8, 2017 (Johnson, 2017; Stack Overflow, 2017). This closure is because of the tremendous effort required from the developers and team members to carry out all suggested edits and re-organize and moved the stored information, which is cumbersome especially because Stack Overflow has a massive amount of stored information that has been collected over the years (Stack Overflow, 2017).

Approach/ Results Interpretation: Our approach has tried to tackle this issue by using automated documentation, where the task is automatically stored at the beginning with the attached results. In this case, all tasks, issues related to different mobile devices, and scenarios performed to discover these issues and their solutions will be documented in one place with less effort and in a short time. This can be strong evidence of the effectiveness of our approach compared to other approaches. This was pursued because it would meet one of the critical requirements that developers need for in their actual work environment (see Chapter 4), in respect of providing a useful wiki that allows them to share their relevant knowledge and experiences and helps them to seek a solution for issues they are looking for quickly and bring supporting evidence to enable them to understand the reasons for specific issues which appear.

Summary: The implemented wiki in this work will have a practical and positive impact in the future, as it will help developers and testers in academia and industry share their testing knowledge and experiences, which will help them to share different testing scenarios and figure out more issues that they have not yet discovered. In this way, neither developers nor testers perspectives should be neglected. If the documentation strategy is not well

¹⁰<https://stackoverflow.com/>

¹¹<https://stackexchange.com/>

¹²<https://github.com/>

developed based on their requirements, then the knowledge sharing process would not be very efficient and thus the quality of their contributions to exchange and share their testing knowledge and experience would decrease. Moreover, this wiki will also be a beacon for future research, where it highlights more issues about compatibility issues which enables researchers to focus on and find solutions for a specific issue. The implemented wiki can also be a reference model for mobile device manufacturers, which can help them detect the reasons or issues in a specific mobile device component or OS version configurations. Thus, it would enable them to improve it in the upcoming versions.

RQ2: How effective is the proposed public crowdtesting approach in addressing the fragmentation-induced compatibility issues and allowing effective compatibility testing in comparison to the state-of-the-art approaches?

The proposed public crowdtesting approach has demonstrated its high effectiveness in addressing the fragmentation issues of mobile devices and OS versions and facilitating effective compatibility testing for both developers and testers compared to other state-of-the-art approaches. Based on the sub-questions A to J discussed above, it is clear that our approach has succeeded in meeting the unmet needs of developers and testers in existing approaches, as well as covering the five main causes behind the continuing difficulty in performing compatibility testing and those that have hampered the success of previous approaches, as outlined in Chapter 1, Section 1.1.

The discussion of ***sub-question "A"*** proves that our task defining and distribution mechanism meets the needs of testers in terms of providing sufficient information regarding the test as well as an indication to the degree of complexity of the task and the amount of time and effort needed to perform the task. It is evident from sub-question "A" that this mechanism greatly helped developers to define large apps in terms of small testing tasks and subsequently allowed them to deliver the tests on a large scale as public testers worldwide could access and select tasks using the dashboard and were able to select tasks themselves rather than having the tasks assigned to them by developers, as proved in ***sub-question "B"***. When considering how to select testers efficiently, our system, as opposed to others, highlighted the need to consider four criteria: test context, tester capability, tester experience, and tester availability. The discussion of ***sub-question "C"*** proves that our tester selection method is effective and would serve well as a database that would help individual developers, small teams, or SMEs to gain access to testers and perform their tests successfully. The discussion related to ***sub-question "D"*** hinges on the capability of our approach to meet the need for a simple and efficient report submission and aggregation mechanism. The discussion shows that our approach succeeded in providing an effective and easy report submission mechanism that reduces the effort of testers and assists them in providing accurate results quickly compared to existing methods. It also demonstrates that our internal aggregation mechanism is effective

and allows developers to track, collect, and organise test reports in a shorter time and with less effort compared to other approaches that use external reports-tracker systems such as JIRA or other systems.

The discussion in *sub-question "E"* relating to how our approach helps to find more compatibility issues and reduce costs, proves that the method we use enables developers to achieve better testing coverage of devices and helps them to discover more valuable results related to unexpected issues of the different architectures of mobile device models and OS versions in the early stages of the app development process. The discussion in sub-question "E" also clarifies that many aspects of our approach overlap and together help speed up the testing process and reduce unnecessary costs compared to existing approaches. The implemented wiki helps developers consider the documented issues and their possible solutions early, which speeds up the development process and reduces the unnecessary cost of redundant tests. In addition, the proposed direct interaction workflow also proves its effectiveness in reducing the cost and accelerating the testing process. The workflow dispenses with the need for crowd leaders and managers who are often the cause of delays in other approaches and eliminates the consequent cost that must be paid for them. The discussion in *sub-question "F"* shows that our workflow, which supports direct interaction between developers and testers, allows for the organisation of the test on a large scale and performing a more effective compatibility testing process. It helps testers to understand the test requirements more fully and provide more accurate results. At the same time, it assists developers in understanding the results of tests and, in turn, increases their confidence in the testers they have selected to perform their tests. The second part of the discussion of sub-question "F" also proves that our workflow is easy to use and meets the requirements of the daily work routines in the various contexts of different developers and testers.

Based on the finding discussed in *sub-question "G"*, it is evident that our report quality evaluation method and its metrics provide developers with easy access to a fair evaluation process that can be quickly and accurately completed with less effort or cost compared to the current evaluation methods. The discussion also proves that the evaluation status tracking service enables testers to avoid repeating the same mistakes in other tasks by allowing them to know the evaluation results earlier without having to wait a long time for the feedback as in other approaches where they have to wait a long time to receive the feedback from crowd leaders. The integrated feature on the dashboard displaying testers' results quality distribution submitted for each task also helped developers to identify the reason for the low quality of testing results. The discussion presented in answer to *sub-question "H"* revolves around the capability of our approach to meet developers' needs in terms of reducing their concerns related to distinguishing between testers who are trusted and those who are not. The discussion proves that trustworthiness/reliability management reduces developer concerns and enables them to quickly identify areas within which testers are specifically reliable. It also shows that our trustworthiness/reliability management can outperform other

approaches if we manage the trustworthiness in two levels based on generic reputation as in current approaches and then classifying each tester's reliability level according to the areas they are professional within as applied in our approach. The second part of the discussion in sub-question "H" shows that the privacy and protection mechanism of our test environment enables developers to protect the privacy of their published tasks and prevent unauthorised testers from accessing the information. The testers' responses confirmed this as they were unable to access details of the tasks that developers defined as 'private' tasks. The discussion also indicates that our approach provides successful protection and protects the testing environment by controlling non-active/productive testers and providing a secure link for detecting mobile data with the submission form.

Moreover, it was necessary to meet testers' needs for the provision of an effective and fair rewarding mechanism. The discussion in *sub-question "I"* shows that our motivation method and the reward schema are capable of providing fair payment value that reflects testers' effort and motivates them to perform more tests accurately compared to existing methods. Also, it shows that our motivation method enables developers to stay informed about all testers' payment status, which notifies them directly of testers who have not been paid for each task. The discussion in *sub-question "J"* demonstrates that our wiki can effectively improve the collaboration and sharing of knowledge and experience among professionals in academia and industry. More specifically, it meets the needs of testers in terms of gaining more knowledge about best practice testing cases or scenarios and developers to stay informed of the latest advancements in mobile app testing, programming issues, and unexpected testing issues and their possible solutions. The discussion of sub-question "J" also shows that with more frequent use, the wiki will be of more benefit to both developers and testers as it will store more accessible data; at present, it is still new, and therefore, its usefulness is limited.

In conclusion, it is evident from the points selected above that our crowdtesting approach does improve on the current approaches to some degree in respect of addressing the fragmentation issues and performing effective large-scale compatibility testing. This improvement results from considering more aspects in order to meet the actual needs of the everyday work routine of both developers and testers which is unmet by other approaches. That which has been discussed in sub-questions A to J, clearly proves that our approach makes considerable improvements to many of the current approaches by identifying failings in other approaches and exploring how our approach addresses these failings.

9.3 Benefit of the Proposed Crowdtesting Approach

This research aims to assess the benefit of our crowdtesting approach and the benefits obtained by each group of developers and testers particularly, to know how our approach can outperform other current approaches in terms of benefit and being able to effectively

meet the needs of those working with large companies, SMEs, or as freelancers. Based on the results gathered from developers (Chapter 7) and testers (Chapter 8), we were able to draw substantial conclusions to achieve this goal and address the following sub-question, and then answer the fourth main research question (RQ3).

A: To what extent does our approach meet the need of the different developer and tester groups in their everyday work environment?

The overall findings of this research in relation to the benefits gained by developers and testers, offered deep insights into the actual needs they had in the context of compatibility testing and that our approach has been able to address. Issues that have been considered regarding the needs of developers include: (1) the support to complete the test and deliver apps on time; (2) increase coverage of the test; (3) cover more human and hardware resources; and (4) gain more information about compatibility issues. The information required includes: (1) unexpected issues resulted from different mobile device models and OS versions; (2) unexpected issues produced from the different usage of target users; (3) information about causes of the issues and possible solutions, which was confirmed by developers' responses in Chapter 7, Table 7.6. On the other hand, it was important to consider the various needs of testers such as: (1) diversity in tasks and testing areas; (2) freedom in the selection of testing task; (3) suitable and flexible testing environment; and (4) feeling a sense of personal achievement and job satisfaction which was confirmed by testers' responses in Chapter 8, Table 8.5. Some needs were common between developers and testers, such as communication and work collaboration with other developers' and testers' communities. Some of these needs have been mentioned in the literature. However, no research so far has combined and identified the level of these needs for each of the different developers' and testers' groups, so that can be adequately addressed. Our findings contribute to advancing the knowledge and filling the literature gap in terms of this context, proving the proposed approach succeeds in achieving all the actual needs of the everyday work routine of these different groups. The following is a detailed discussion of these needs and gained benefits for each group. The high percentages associated with each gained benefit by each group (as shown in Chapter 7, Figure 7.4, 7.5, 7.6 and Chapter 8, Figure 8.4, 8.5, 8.6, 8.7) prove how successful our approach is in addressing each need.

A. Android and iOS developers and testers:

Based on Figure 7.4, Chapter 7 and Figure 8.4, Chapter 8, our findings showed that "Android and iOS" developers and testers alike benefited from our approach and all services provided, without exception. This is because we considered all their needs, and we were not biased towards providing for any one of them more than the other. Despite this, there were some

slight differences in the gained benefits perceived by these two groups for developers and testers, which would indicate the level of importance of that need to each group.

In respect of the developers, the findings clearly showed that both of them still suffer from the fragmentation issue and difficulty of performing compatibility testing on all devices to ensure the quality of the apps. Therefore, they found that our approach was highly beneficial in helping them test more devices and discover all unexpected issues in a short time, as shown in Figure 7.4. Furthermore, the iOS developers benefited more than Android developers in respect of gaining help from other developers' communities and learning more about mobile app development issues. This is probably because Apple does not provide more learning resources for beginners, interactive materials, and training programs or courses for different levels of iOS developers, compared to the extensive knowledge that Google offers (Rami, 2020). Therefore, they need to contact more developers and get help regarding programming issues and important guidance that may not exist on the Apple Developer site to avoid rejection of the app by the apple store, which is one of the challenges for iOS developers (Lin, 2021; Pierce and Wooldridge, 2014) Since 2019, Apple started to overcome this shortcoming by introducing a new learning resource (Sarah, 2019), but we think it may still be necessary to document more programming information and issues. For Android developers, they benefited slightly more than iOS regarding the acceleration of the development process and delivering the apps on time to the market. This means that our approach helped them reach more testers and quickly perform the test on all devices. It was also indicated that iOS developers do not face delays in submitting the app to the market as oppose to Android developers. The reason for this lies in the fact that iOS developers do not face difficulty in performing compatibility testing to the same degree as Android developers who have more device models, and they need more human resource to overcome the issue of the delay to test the app on these devices in a short time.

In respect of the testers, both groups have benefited the same equally but one notable difference between them which is that Android testers had benefited slightly more than iOS regarding the suitability of the testing environment that enabled them to receive help and work opportunities from different communities and perform more tests in different areas which they good at, which will improve their testing skills. This may well be the result of the existence of many versions of Android OS and mobile models, the high amount of functionalities created compared to iOS that is restricted to specific functionalities, and the necessity to perform the test quickly to enter the market (Ki et al., 2019), which increases the demand for Android testers to perform the tests on many devices and allows them to collaborate with other members and get several invitations to perform more tests in different areas. Another reason for the high benefits from collaborating and gaining help from other communities is the difficulty of dealing with multiple platforms and transferring knowledge from one platform to another, which sometimes required them to gain help from other testers

to design more suitable and effective testing scenarios and/or test cases to perform an accurate testing process (Vasquez et al., 2018).

Based on the above discussion, it seems clear that testing Android devices is more difficult; the Android developers and testers are keen on the support provided by our approach. Despite these simple differences, the findings show that the level of benefits for the Android and iOS, in general, was the same for both developers and also for testers, none of them has benefited more than other, as outlined in Chapter 7, Table 7.7, and Chapter 8, Table 8.7. This strongly confirms that the proposed approach is successful in meeting the actual needs of the daily work of Android and iOS developers and testers.

B. Freelancers (independent developers and testers)

Based on our findings in (Chapter 7, Table 7.7) it is evident that *freelancer developers* had benefited more than employees in general. The very high percentages given for the benefits gained indicates that they are the developer group suffering most in performing tests. This emphasises their intense need for the services provided by our approach in order to enable them to improve their work. They are inclined to take advantage of this approach due to the fact that freelancer developers usually work on their own or as small teams working to a limited budget. The lack of human resources and sufficient funding hinders their ability to perform the test on the largest possible number of mobile devices (Linares-Vásquez et al., 2017; Vilkomir, 2018). This means that current solutions may not serve them adequately and our approach was found to be capable of meeting this need as it allowed them to perform the test on all devices. Another reason could be that current solutions lack the capability of providing the freelancer with a suitable and practical testing environment and platforms for managing their works and supporting them to communicate and gain help from other developers' communities (Akhmetshin et al., 2018). Indeed, such a group of developers has a small testing environment at the offices or even at homes (Hamza, 2020), which makes the testing process very hard and exhausting and thus reduces their productivity and work. Therefore, in order to help overcome these barriers and help them to manage and track all their works, they found our approach to be an effective testing environment. In particular, they found that our approach helps them to increase their human and hardware resources and complete the test smoothly with less effort and involving lower costs. Also, it gives them a chance to communicate with public developers and testers communities which would most likely increase their test community size and enhance their productivity. This is obvious from the very high percentages in the feedback related to the related benefits to these points that they received from our approach. This is clear evidence of the success of our approach in addressing the freelancer developers' needs.

As for *freelancer testers*, the findings presented in Chapter 8 Figure 8.5 indicate that freelance testers sorely need all the services provided by our approach to improve their work;

therefore, they have highly benefited from our approach. This is because some of the current approaches and testing companies do not take freelancing seriously (Asha, 2018; Gupta et al., 2020b), as also shown by participant developers in our previous study (Chapter 4). Thus, they do not offer them a suitable testing environment to manage their business work or provide them with enough support to perform more tests effectively and discover all compatibility issues. Freelance testers often face obstacles such as a lack of good suitable tests to perform them or deal with challenging tasks by themselves (that companies are provided to them). Therefore, they highly benefited from our approach, as it provided them with a suitable testing environment that enables them to perform more tests that are compatible with their experience and in the areas they are very familiar with it. Moreover, one of their challenges is to need to learn different testing strategies and gain a wide professional experience to perform several tests (Zhang et al., 2017b). Therefore, they found that our approach benefits them in communicating and collaborating with different developers and testing companies and completing more testing tasks in different ways, which would improve their testing knowledge, skills, and professional experience. The high percentage rating in respect of the benefits gained by freelance testers when adopting our approach, as outlines in Figure 8.5 is clear evidence of the success of our approach in meeting the needs of freelance testers.

C. Small/Large Company Developers and Testers:

Likewise, as is the case with freelancer developers, *small organisation developers* also found the approach very beneficial as it provided them with a means of addressing all their needs. This is most likely due to the same reasons mentioned for freelancers regarding the small internal community size, limited hardware resources, and lack of work collaboration, and limited knowledge about all possible issues, which causes a delay in delivering apps on time. Therefore, they found that our approach addresses all these issues effectively. The only difference was that employed developers in small organisations felt that they had benefited less in terms of receiving help from other developers' communities and facilitating mobile app development through our approach. This can be because they sometimes get an offer of temporary work with large companies, which aids them to benefit from the community members inside these companies. Indeed the small companies' developers seek to increase their required knowledge, skills, and experience to develop apps for various platforms. One reason can be that "a large portion of the app developer community is considered to be novices" and so they may lack the level of knowledge and experience (Asfour et al., 2019); especially, small company developers and freelancers, they may come from traditional web or desktop development backgrounds (Guo et al., 2014). Therefore, they found our approach to be, overall, beneficial in improving their testing knowledge and facilitating the app development process to deliver their app on time; this is evident from the high percentages associated with these benefits as shown in Chapter 7, Figure 7.6.

With regard to *large organisation developers*, although they have a large number of employees, sometimes they may not have the immediate availability of hardware or human resources (Gupta et al., 2020a). This is due to the high demand for these companies and work pressure, which most likely would lead to employees with niche skills not being available to undertake specific tasks or unavailability of specific devices because they might be in use. This in turn could lead to delivery delays, which is not acceptable for large companies, but our approach has successfully helped them overcome such challenges by inviting more testers from public members and covering all versions of mobile devices and OS quickly at 73.3%. The results also showed that large company developers can benefit from our approach and the developed wiki by allowing them to improve their testing knowledge and skills. This is evident from the high developers' benefit rating for the knowledge provided about issues that result from different users' behaviour and their interaction with apps, and new issues related to the different architectures of mobile devices and OS versions, as shown in Chapter 7, Figure 7.6. Regardless of these benefits, they found our approach to be only moderately beneficial for facilitating app developments, delivering the apps on time, increasing test coverage, improving work collaboration, or gaining help from other developers' communities. This may well be due to their vast communities, which enable them to overcome collaboration issues or seek external assistance from other communities, as well as helping them to cover the test much faster, and thus resolving the discovered issues quickly before delivering the app to the market (Tran, 2020). This is clear evidence of why the large organisation developers have benefited less than small companies from our approach, as has been confirmed by the results of the t-test analysis in Chapter 7, Table 7.7. Consequently, this strongly confirms the success of our approach in meeting the actual needs of the daily work of developers who work in small organisations.

Similar to the case with freelancers, *small company testers* have benefited highly from our approach. Our findings show that small or startup companies seek to improve their testing skills and use different testing techniques to quickly adapt to changes in the underlying market (compared to larger companies). Therefore, they highly benefited from our approach as they were able to perform the tests in different ways and in different areas (types of apps) that are compatible with market requirements. From Figure 8.6, it also seems that small testing organisations still have a big problem with the actual time required to perform the test. This can be because of their work pressure as they have testing communities that involve a small number of individual testers and need to work fast and to work under pressure and work overtime if necessary in order to meet the developers' needs (Giardino et al., 2014). They found our approach to be highly beneficial with 82.8% in addressing this challenge as it helped them to hire and leverage more testers to carry out the test at any time and anywhere successfully under short time constraints. Moreover, one of the greatest benefits for them compared to freelancers was the suitability of our testing environment for their different testers' skills and enabling them to communicate and collaborate with

external testing communities and thus receive help from them 8, Figure 8.6. This is probably because small companies usually lack a space in which to share testing knowledge between their testers where not all of them have the same level of high experience, and such lack of sharing information about the testing structure, steps, scenarios might hinder their work in the company. This large-scale collaboration with other testing communities might be the reason behind helping them to counterbalance their lack of human resources and addressing their time constraints issue as discussed above.

In respect of *large testing company testers*, our findings point out that the benefits gained by testers working in large organisations did not differ much from other groups. That is probably because testing is pretty similar everywhere for all groups where the support that developers receive in the large organisation allows them to solve issues more quickly. On the other hand, small organisation testers and freelancers miss that support, so they benefited more from our approach and are keen to use it. One of the differences that is evident is that large company testers have benefited less from our approach than small company testers in terms of communicating and collaborating with other testing communities in order to perform the tests in different ways. This is normal because of the large size of their testing communities and the high level of the testers' experience, which allows them to collaborate and perform the tests in different ways. Moreover, our findings have indicated that large company testers have more flexibility in performing the test. This would also explain the high percentage rating of the benefit our approach by small company testers in which our approach addresses the time constraint issue they have, as we discussed previously. Therefore, they moderately benefited less regarding this benefit than testers working in small companies who benefited highly. Regardless of this, our findings clarified that the overall perceived benefit level of both testers who work in large and small companies is the same as shown in Chapter 8, Table 8.6. This proves the effectiveness of our approach in meeting the actual needs of these two groups especially testers who work in small companies.

D. Different Levels of Testers Experience:

Overall, our findings show that there is not much difference between the needs of beginner (novice), mid-level, and expert/professional testers. All of them have greatly benefited from almost all of the benefits offered by our approach (see Chapter 8, Figure 8.7). This probably indicates that the current solutions and the industry do not cover all the needs of the testers whom they are seeking, and this lack can affect their work negatively. Indeed, testers need to have a channel that allows them to communicate and share testing knowledge with other testers who are experts in different areas and have different testing skills (Florea and Stray, 2019; Vasanthapriyan et al., 2017), which would help them to improve their testing strategies and detect more errors (Salman et al., 2020). In this respect, they found that our testing environment and its wiki are very beneficial in covering this need. This is evident from

the very high benefit percentages related to these points as shown in Figure 8.7. This implies that regardless of the testers' levels of experience, they always need to broaden their communication and testing knowledge and skills. We strongly recommend that the testing companies foster a knowledge sharing culture between their members for all software testing activities as this would consequently improve their work quality. Moreover, testers need to always be familiar with the different methods, test cases, or scenarios relating to performing the tests (Vasanthapriyan et al., 2017). In terms of this need, all of these three groups found that our approach is beneficial to achieve this need; especially the novices and mid-level testers who found that our approach was highly beneficial in helping them to perform more tests in different ways and in different areas that they are good at, compared to experts. The reason could be that they plan to enhance their testing skills according to the current industrial needs to find a better job whereas the experienced testers are always hired by the companies, and thus they have more chance to perform more tests in different areas and in different ways. Regardless of this slight difference, the overall benefit level for all novice, mid-level, and professional testers from the approach is the same, as confirmed by the differences test in Table 8.7. This proves the success of our approach in covering the needs of all different levels of testers' experience.

RQ3: Does the proposed public crowdtesting approach benefit both developers and testers? If so, is there a specific context where the public crowdtesting approach is particularly beneficial for developers and testers?

The findings of this research demonstrate that our crowdsourced compatibility testing approach is highly beneficial for developers and testers alike. However, the overall benefit for developers was 77.4%, relatively lower than testers at 82%. This difference is perhaps because some developer groups benefited slightly less than other groups leading to a decrease in the overall benefit percentage. Freelancers benefited more than employee developers in general, and small organisations received much more benefit than large organisation developers, while both iOS and Android benefited to the same level. Regarding testers, our findings showed that, in general, there was no difference in the overall level of gained benefit among the testers groups. All of them have benefited highly from the approach with a slight difference in favour for the freelancers and those who work in small companies. The high benefit level of all groups of both developers' and testers' is due to the ability of our approach to properly capture all their particular needs and provide support in several areas relevant to the context of compatibility testing and mobile apps development (as proved in sub-question (A) related to the benefit).

Indeed, the findings provided by this study go beyond previous research studies and find a preliminary answer to the hot question in the field "Why do fragmentation-induced compatibility issues and difficulties of performing effective compatibility testing still exist to

date, although many solutions were developed during the recent five years?" The answer is that most of the researchers did not consider all the requirements of the different groups of developers and testers when they developed their solutions. Therefore, current approaches and practices in academia and industry are unable and insufficient to cover the actual needs of all the different groups, as we discussed previously. Another piece of evidence to prove that is the literature, in which that most of the previous research studies have focused on helping Android developers and testers only because they face the fragmentation issue more often than iOS developers and testers. Without a doubt, all these existing solutions have somehow reduced the challenges of Android developers and testers regarding fragmentation issues and compatibility testing. However, this has left other groups, such as freelancers and those who work in small organisations, still suffering from this issue and waiting for a solution. The findings of this research study are sufficient evidence to definitively prove that the requirements reported in the literature and used by other researchers do not represent the actual needs of the developers and do not apply to their real-work environment. This is the main reason why all different groups of developers and testers have benefited from the approach to perform effective large-scale compatibility testing.

9.4 Satisfaction with the Proposed Crowdfunding Approach

This section aims to achieve the goal of our research by answering our main research question (RQ4) regarding the satisfaction of the different groups of developers and testers and their intention to use the proposed approach in the future.

RQ4: To what extent are developers and testers satisfied and keen to use the proposed crowdfunding approach?

According to our findings presented in Chapter 7, Table 7.8 and Chapter 8, Table 8.8, it was interesting to see that testers are more satisfied than developers regarding the use of this approach. This is perhaps due to the fact that some companies have their own QA teams and testing methods, and they may not need to conduct additional testing with non-permanent human resources, which leads to having many good testers available for work, but they do not have a job or the opportunities to perform tests through the current approaches. Since our approach has opened the test for the public, they found the opportunity to work, improve their knowledge and skills, and demonstrate their ability to work. This higher percentage will increase developers' confidence in the fact that if they use our approach, they will find many testers available to perform their tests quickly. Regarding different groups for each, our findings also indicate that all developers' and testers' groups are very satisfied and delighted to use our approach and the services provided as a priority for their tests in the future, especially Android, iOS, freelancer, and those who are working in small organisations, as well as testers who have different experiences. This is because it has successfully met their actual

work needs (as it has proven previously in RQ2 (A)), which is becoming more complicated day by day because of the exponential growth of mobile technologies. This means that by using our approach, these groups would be able to scale up their work according to the needs of the business. As expected, the developers and testers who work in large organisations were less satisfied using the proposed method than those in small organisations, as clarified in Table 7.9 and Table 8.9. Although the testers who work in large companies feel they benefited to the same degree as those who work in small companies, they are not very keen to use it. This is undoubtedly because they have support from the companies they are working with. Other groups are still suffering from difficulties in performing compatibility testing so far and because they do not have sufficient support that addresses the fragmentation and compatibility issues, they are highly keen to use our approach for their work.

Based on the discussion above, we can confirm that our approach is superior to other approaches in tackling one of the crucial gaps in the literature regarding the development of a solution that helps both Android and iOS developers and supports the individual developers and small/medium-sized enterprises (SMEs), which represents the motivation of our research (Chapter 1, Section 1.1). Thus, we argue that our approach would significantly reduce this issue if not eliminate it entirely.

9.5 Chapter Summary

This chapter has provided an in-depth discussion based on the findings of the empirical evaluation studies conducted on developers (Chapter 7) and testers (Chapter 8) in order to answer the research questions (RQ2, RQ3, and RQ4). The discussion points are supported with reference to the specific details of the outcomes of the empirical evaluation studies to help ascertain to what extent the approach and its sub-processes are effective in facilitating the performing of large-scale compatibility testing. In particular, the methods employed in this approach are effective for developers and testers in terms of cost, time, effort, and ease of use. Particular attention has been paid to the benefits offered by this approach as opposed to those offered by other state-of-the-art approaches in respect of freelancers and those working within small and medium-sized companies. Specific intrinsic limitations have been identified in certain approaches that overlook requirements that in our view need to be met. The discussion presented here has examined to what extent these limitations were overcome through the development of this approach and evaluated whether these requirements were adequately met or if they need to be improved in the future. The chapter also clarifies to what extent the different groups of developers are satisfied to use the approach for their work in the future. Based on our in-depth discussion provided in this chapter, the following chapter offers guidance to researchers and practitioners seeking to design effective crowdtesting solutions in the future. The next chapter also concludes the research and discusses its limitation, and provides recommendations for future research in this field.

10

Conclusion and Future Work

10.1 Introduction

This chapter concludes the research thesis. It begins with providing a summary of the entire research, which recapped all the thesis chapters. This is followed by a list of guiding principles for researchers and practitioners interested in crowdsourcing contexts. The chapter then concludes with a discussion of the research limitations and suggestions for future research directions.

10.2 Conclusion of the Thesis

In this thesis, we researched fragmentation-induced compatibility issues and the reasons behind their existence to date, and how they can be tackled. Based on this, we developed a new and generic crowdsourced testing approach that supports the effective performing of large-scale compatibility testing to prevent the rise of new compatibility issues for Android and iOS apps with the different mobile devices and OS versions in the future. In particular, we focused on the following challenges: (1) distributing the test to the global testing communities to cover a broad set of mobile devices and OS versions with minimal cost, time, and effort; (2) improving communication and work collaboration between developers and testers during the app development and testing process; (3) facilitating collaboration and sharing of test knowledge between academia and industry; (4) providing more knowledge regarding best practices of test scenarios as well as the compatibility issues, their causes, and possible solutions. The main motivation behind this research is to find a means of resolving the issues involved having identified that testing mobile app remains a major barrier to the large-scale deployment of apps by individual developers (freelancers) and SMEs. The problems identified are: (1) they do not have the ability to test the full set of mobile device models or OS versions

in the markets due to its high cost; (2) they do not have knowledge resources that can be used or referenced.

To facilitate the discussion, we reviewed the relevant literature and industrial practices in the field of crowdsourced testing for mobile apps and its contributions to reductions in fragmentation-induced compatibility issues (Chapter 2). Hence, we provided an insight into how crowdtesting works and how testers and developers make use of it. Emphasis was given to existing approaches used for performing compatibility testing. Throughout that chapter, we were able to contribute to the field and provide (1) a state-of-the-art comparison of crowdtesting approaches and industrial platforms, including their achievements, as well as the salient challenges of each; (2) an in-depth comparison of current crowdtesting workflows used by these approaches and platforms and gaps in them; (3) a list of current challenges faced by developers and testers which have not been addressed through current research so far. Through the literature review and analysis of the three points above, it has been shown that the existing testing approaches and practices do not receive enough attention to enter mainstream work practices in the industry because it has focused on the literature regarding prior studies and the challenges involved in order to identify their basic requirements. Also, we found that the existing requirements reported in the literature do not cover the actual needs of real practitioners. This is likely the main reason why developers and testers did not appear to benefit much from the existing testing approaches.

Based on that, we conducted an in-depth exploratory survey study on practitioner developers in both academia and industry to (1) gauge their perspectives regarding working with public testers, with varied experience; (2) to obtain a broader view about the actual needs in respect of their everyday work routine in their testing environments. The findings showed that app developers are willing to work with public testers if certain crucial challenges can be addressed and the needs detailed in Chapter 4 are fully met. Additionally, the results clarified that the basis for establishing an effective testing process requires flexibility in respect of defining the task; clarity and sufficiency of the task requirements and testing results; simplicity of the reporting mechanism. The results also showed that the direct interaction and cooperation between developer and tester, knowledge sharing, and the development of trust between them, are key to establishing a good and long-term working relationship between testers and developers in the working environment. From the findings of this survey, it was revealed that some developers have little knowledge of crowdtesting and how it works, even those who work within companies where it is used. Also, it became clear that the work of testers is not held in high esteem by some developers, and the work is regarded as purely a matter of cooperation, not a truly professional job deserving payment; therefore, they prefer to offer alternative non-financial rewards. Through this study, we were able to fill the knowledge gap in the literature regarding two clear aims: (1) the extent to which developers are willing to work with members of the public as testers; (2) the identification

of the developers' needs and factors that would reduce their concerns when working with unknown testers and help increase their confidence in them.

Based on the identified developers' needs, we formulated our core requirements to develop our approach as an online web-based crowdtesting platform in an effective way that supports the practice of compatibility testing applied in reality. This approach is designed based on a direct crowdtesting workflow to bridge the communication gap between developers and testers and efficiently organise the distribution of test on a large scale. This approach enables developers to perform the test by public testers and real users on a larger number of devices quickly. The approach supports performing the three main dimensions of compatibility testing: platform compatibility, mobile device feature compatibility, and native API compatibility. This approach helps explore different behaviours of the app and the users with the app to identify all compatibility issues. The platform also included a public wiki to document the compatibility testing results, including discovered issues, their causes, and possible solutions (Chapter 5).

To evaluate the success of the implemented approach in terms of performing the compatibility testing effectively and efficiently, its benefit and satisfaction, we conducted two empirical evaluation studies (Chapter 6). The outcomes of the evaluation of developers (Chapter 7) and testers (Chapter 8) and its comprehensive discussion (Chapter 9) showed the following: **Firstly**, our approach is highly effective and has greatly helped developers to distribute the tests on a large scale and execute them by public testers on larger number of devices in a short time. As it also clarifies that our approach outperforms the state-of-the-art approaches by making considerable improvements to their internal sub-processes, which address their failings and hence facilitate the compatibility testing. More specifically, the evaluation showed that our approach and its direct testing workflow as well as the improvements made on the sub-processes have assisted both developers and testers to carry out the test with minimal cost, time, and effort, compared to other approaches. **Secondly**, our approach is beneficial for both developers and testers as it successfully meets the actual needs of the real work environment of the following groups: Android, iOS, freelancers, and those who work in small and large companies as either developers or testers. In particular, the proposed approach responded to the unmet needs of freelance developers and testers and those who work in SMEs who do not have enough support as opposed to those who work in large organisations, which was the main motivation behind conducting this research. **Thirdly**, the evaluation proved that all the different groups of developers and testers are highly satisfied with the approach and intend to use it to conduct their tests in the future, although the level of satisfaction was especially high in small-medium enterprises.

Through the evaluation findings, it became apparent that testers need more guidance on the requirements and clearer instructions of the tasks to carry out more accurate and effective tests efficiently. It also became apparent that taking the complexity level of the task into account is one of the main factors that can lead to obtaining more accurate results,

and regarded as crucial by all groups of testers, without exception, in order to perform more suitable tests that are compatible with their testing skills. It became evident that communication and work collaboration is regarded as extremely important for both testers and developers, regardless of their level of experience, which would help make their business more successful in the long term. Furthermore, it is evident that evaluating testers' work and rewarding their work fairly will help greatly increase the trust between developers and testers, which could serve as the key for effective cooperation. One further insight was that providing feedback of testers' work evaluation early is important to foster their work and avoid continued repetition of mistakes. Without a doubt, we can say that our approach has succeeded in showing the way to provide a solution meeting the several unmet needs of both developers and testers in SMEs and in handling the reasons behind the difficulty of performing compatibility testing and reducing fragmentation issues if not eliminating them. Based on that, we were able to identify a set of principal guidelines that hopefully will assist practitioners, researchers, and companies in developing a new generation of crowdtesting platforms that better support their users (as presented in this chapter). In the end, we detailed the research limitations and discussed the possible future research directions of crowdsourced mobile app testing.

10.3 Guiding Principles for Researchers Interested in Crowdsourcing Contexts

Drawing from the outcomes of this research which demonstrate that it has implications on practice as shown in Chapter 9. From a practical perspective, our research allows us to offer in-depth guidance to researchers and practitioners interested in using crowdsourcing in their work that should be considered in order to design effective crowdtesting solutions. These guidelines can also apply to different contexts of crowdsourcing to facilitate future research.

- **Task Defining** The developers (crowdsourcer) need to be careful about the appropriate design of the task defining method and provide sufficiency and clarity of the task description when defining the task. This will help to perform an effective testing process and increase the task execution results to reach a high quality
- **Task Complexity Level:** It is necessary to include the task complexity level and the expected execution duration along with the task specifications when defining the task. This will facilitate the task selection process for testers and help them decide whether they can perform specific tasks early before they are executed, achieve a high level of performance, and gain the desired results.
- **Task Selection:** Receiving an improper task may decrease the testers' motivation and quality of the results. Therefore, it is important to consider better sorting criteria so as

to implement an effective task recommendation and provide testers with the ability to search for their tasks "with some control constraints". This will support and facilitate the process of task selection and execution, and so increase test coverage.

- **Testers Selection:** The process of selecting testers should not be limited to the suggested list only. It is preferable to allow developers to reach testers' information and choose the appropriate testers by themselves. This will help developers be more comfortable and aware of the testers' experience and activities relevant to working on their apps, which will definitely increase the developers' trust in these testers, lead to better collaboration work and a more effective testing process.
- **Task-monitoring Mechanism:** It is key to build an automated monitoring mechanism in order to track the coverage of published tasks after they are distributed to the testers on a large scale. This should include, number of times task execution, tested devices and OS versions, coverage of target geographical locations, coverage of target users backgrounds. This will help them stay informed about the progress of tasks' status and update them about the remaining requirements. In respect of that, this will facilitate the testing process by eliminating the delay, reducing the chance of getting repetitive results, and giving a clear idea about the actual time needed to complete the work. The provision of this information may improve the test outcomes, work productivity, and time-to-the-market delivery.
- **Issue Reporting Mechanism:** It is necessary to consider the ease of use of the reporting mechanism used and the time and effort required to finalize the submission process. The fact that this involves a longer time and more effort will inevitably affect testers' motivation with regard to performing more tests. It is suggested that the automated service are also applied to support testers to enter data or screenshots directly from the tested devices. This will ensure the accuracy of the presented data, facilitate the testing process, and increase developers' confidence in submitted data.
- **Aggregation and Tracking Mechanism:** Due to the large number of submitted reports in crowdtesting, an automatic aggregation and tracking mechanism with duplicated reports detection service must be applied. This will save developers time and effort to collect, organise, and identify duplicated or uncompleted reports, which would probably accelerate the testing process.
- **Diversity of knowledge, skills, and experience:** The testing process should not be limited to experts only. The diversity of knowledge, skills, and experience must be considered when selecting testers to represent a larger segment of society with different backgrounds and behaviours of app users. This will help them broaden their horizons in respect of their work, solve problems from different aspects, and ensure seamless work and high-quality apps.

- **Availability of Testers** It is important to consider the testers' availability during the selection process, as it is one of the crucial factors that can affect the completion time and quality of the work. If the testers have only a short time to perform the task, this will create time pressure and psychological stress that inevitably will affect the test execution and lead to the poor quality results.
- **Providing Feedback:** Developers, practitioners, and testing companies must provide feedback and activity summaries to the crowd testers in textual and visual form. This will help them obtain more insight into how effectively they are performing, which enhances their performance, provides them with a sense of achievement, and motivates them to perform more crowdtesting tasks.
- **Incentives and Motivation:** It is key to understand the criteria related to the work context to achieve a useful reward schema that can identify a persuasive and fair reward value. It is also important to design a crowdtesting approach with the testers in mind, allowing them to understand the rewarding process (providing information on how the reward is calculated). This will increase their confidence in the payment value and motivate them more to perform better testing tasks. Different non-monetary incentives also need to be used to motivate testers; this is useful for achieving high-quality results for a long time.
- **Direct Communication and Interaction:** Crowdtesting companies should always be keen to increase direct communication and interaction between the developers (crowdsourcers) and testers (crowd). This will help testers gain more understanding of the tests' requirements and perform more effective tests quickly. It will also help developers understand the testing results more fully and consequently accelerate the app development process.
- **Test Reports Quality Evaluation:** It is necessary to assess test reports provided by developers, taking into account other features such as: using evaluation metrics that ensure all different testers work quality; providing testers with percentage received for each metric; providing developers with a results quality distribution dashboard; and keeping testers informed of the evaluation's status of their submitted reports or the action taken. This will help to perform a more effective, efficient, fair, and motivational evaluation process.
- **Trustworthiness/Reliability Management:** Developers and crowdtesting organisations should engage testers in the development of trust mechanisms. This will help build effective worker reliability classification that supports a high degree of trust among developers and testers and improves the satisfaction of participating and using the crowdtesting approach from both sides.

- **Knowledge Sharing Environment:** Suppose that the developers' or practitioners' objective is to create a knowledge sharing environment to store key results, including issues, causes, and solutions, or link it with other Q&A sites or knowledge repositories. In this case, it is necessary to consider the documentation process from the beginning. Also, a suitable automatic storage method must be found to reduce human effort. This is because when dealing with a considerable amount of stored information, the documentation process (especially manual documentation) in respect of this information will be exhausting and extremely challenging. A clear example of a similar occurrence of this happened in the case of the Stack Overflow. Such a problem can lead to a poor documentation process and undesirable usage from users.

10.4 Research Limitations

Although this research has succeeded in objectives, and produced several contributions, it has some intrinsic limitations.

- Due to time constraints and the platform's usage scope, the number of respondents (testers and developers) was limited and hence not fully representative of the worldwide developers and testers community. Despite having collected data from different countries and different companies, the level of representativeness of the results is to be proven on a wider scale, and that is only possible with the implementation of a fully-fledged industrial solution. This was not in the scope of the thesis, but it is enabled by it. Although this research sample was randomly selected, a larger sample size would have helped us obtain more precise results that could be generalized to a larger population.
- Another challenge in our studies was the scarcity of literature providing more detailed knowledge about how effective current crowdtesting approaches in practice and their challenges. This limitation is supported by the conclusion of (Gao et al., 2019). As most of them are related to industrial testing organisations; the overall workflows and some parts of their testing processes had only been published in the literature. This reduced the possibility of more in-depth comparisons with our approach. Furthermore, the lack of participants' knowledge about the crowdtesting concept, which the findings of this research have confirmed, may have skewed some of the results, especially could have caused some of the lower numbers of respondents. My works have shown how this gap in the literature needs further study.
- We understand that each testing organisation may have different methods and ideas for performing the test. The big testing organisations might have strong testing backgrounds and cultures, while small organisations or freelancers may not. It was important to gain access to more details about the rewarding mechanism, workers

classification, budget of the test, or systematic ways of testing that developers and testers use and the challenges they face inside their organisations that are not published yet in the literature. Due to the privacy and confidentiality of the organisations' internal information, it is not easy to obtain such information from employed developers for confidentiality reasons. This did not allow us to perform more in-depth comparisons and provide more evidence on our approach's ability to outperform.

10.5 Future Research Directions

The limitations mentioned above did not affect the achievement of the research main goals; it paves the way for future research developments. The following are a set of future research directions extracted from limitations and those that emerged from the discussion in Chapter 9.

- The findings of this research show that providing more information about task complexity (its difficulty and duration) is crucial in order to achieve high work performance. However, the influence of the duration time and difficulty level of the testing task itself on testers' behaviour was not easy to assess because these two factors are strongly subjective and hard to measure using quantitative indicators. New measures could be developed in the future to address this issue, maybe starting from a psychological background rather than a computer science one.
- The results of this research are promising and further research should be performed to confirm and generalize the results of this research more by conducting further evaluation studies with more participants (developers and testers) from different regions and over a longer period of time. Future research with more participants will help determine whether some countries or developers in specific countries or regions have difficulty performing compatibility testing and fragmentation-induced compatibility issues more than other counties or regions.
- This research focused on collecting only quantitative data and was able to achieve the research goals. However, the research results indicate that collecting qualitative data may have added more value to this study. Future qualitative research (e.g., in-depth interviews) should be performed to gain more insight into our approach's experience and the features or processes that need to be improved so that more improvements and comparisons can be carried out that provide further evidence that confirms our approach's effectiveness compared to the state-of-the-art approaches. Future qualitative research will be significantly helped by interviewing the managers or members of the organisations who may have the authority to provide more information about the testing methods that developers and testers (freelancers and those working in small and big organizations) use and the challenges they face in their organizations and can

be considered in our approach. As the findings of this research concluded that there are differences in the gained benefits among the different developers and testers groups, future qualitative research should aim to obtain a broader view of these differences and better understand their causes.

- The findings of this research show the importance of knowledge sharing wiki and direct interaction for both developers and testers. Studying the effect of knowledge sharing wiki and direct interaction on compatibility testing was not the goal of this research, but including this will add value to this research. Future research can examine the effect of knowledge sharing wiki and exploring the influential factors on compatibility testing processes. Also, it can widely investigate the influence of direct interaction on the testing process. Furthermore, since our findings have also demonstrated that the diversity of experience and feedback given could affect the testers' performance, the future work will also target the effect of these factors on the testers' performance for different types of testing tasks, depending on the task nature. Finally, future research could study the impact of task design, task complexity level, and task duration on testers' participation performance.
- Albeit the privacy and protection process and testers reliability (trustworthiness) tracking mechanism showed to be effective, the level of effectiveness is lower than that of other processes; both developers and testers have recommended further enhancement. To further our research, we intend to continue towards the goal of improving these two processes and identifying better patterns of ensuring security, privacy, and protection method and classification schemes. Also, we will continue improving the aggregation mechanism by implementing report duplicated detection service that would lead to better performance and effectiveness.
- Several research studies have investigated the effect of complexity level for crowdsourcing tasks on crowd workers' performance in different contexts. It is still unclear how the task complexity and task duration impact testers' performance in crowdtesting tasks. Subsequently, more research needs to be carried out to examine the influence of these two factors on testers' behaviours, execution performance, and outcomes quality and what contributes to task complexity.

Bibliography

- Abhinav, K., Dwarakanath, A., and Singh, P. (2015). *Trustworthiness in crowdsourcing*. PhD thesis.
- Abras, C., Maloney-Krichmar, D., Preece, J., et al. (2004). User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction*. Thousand Oaks: Sage Publications, 37(4):445–456.
- Acharya, A. S., Prakash, A., Saxena, P., and Nigam, A. (2013). Sampling: Why and how of it. *Indian Journal of Medical Specialties*, 4(2):330–333.
- Afzal, W. (2007). Metrics in software test planning and test design processes.
- Akhmetshin, E. M., Kovalenko, K. E., Mueller, J. E., Khakimov, A., Yumashev, A., and Khairullina, A. (2018). Freelancing as a type of entrepreneurship: advantages, disadvantages and development prospects. *Journal of Entrepreneurship Education*, 21(S2):1.
- Ali, A. A. and Dominic, P. (2016). Organizational and individual factors impact on knowledge sharing practice: The association with cost reduction. In *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, pages 536–541. IEEE.
- Ali, A. A., Paris, L., Gunasekaran, A., et al. (2019). Key factors influencing knowledge sharing practices and its relationship with organizational performance within the oil and gas industry. *Journal of Knowledge Management*.
- Almeida, M., Bilal, M., Finamore, A., Leontiadis, I., Grunenberger, Y., Varvello, M., and Blackburn, J. (2018). Chimp: Crowdsourcing human inputs for mobile phones. In *Proceedings of the 2018 World Wide Web Conference*, pages 45–54.
- Alnawas, I. and Aburub, F. (2016). The effect of benefits generated from interacting with branded mobile apps on consumer satisfaction and purchase intentions. *Journal of Retailing and Consumer Services*, 31:313–322.
- Alsayyari, M. and Alyahya, S. (2018). Supporting coordination in crowdsourced software testing services. In *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 69–75. IEEE.
- Alyahya, S. (2020). Crowdsourced software testing: A systematic literature review. *Information and Software Technology*, page 106363.
- Alyahya, S. and Alrugebh, D. (2017). Process improvements for crowdsourced software testing. *International Journal of Advanced Computer Science and Applications*.
- Alyahya, S. and Alsayyari, M. (2020). Towards better crowdsourced software testing process. *International Journal of Cooperative Information Systems*, 29(01n02):2040009.

- Amen, B., Mahmood, S., and Lu, J. (2015). Mobile application testing matrix and challenges. *Computer Science & Information Technology*, pages 27–40.
- Andrew, A. (2017). Detecting mobile browsers with one line of javascript. [online] <https://medium.com/@rchr/detecting-mobile-browsers-with-one-line-of-javascript-109713d5869c>.
- Arlin, C. (2020). Understanding internal and external validity and how these concepts are applied in research. [online] <https://www.verywellmind.com/internal-and-external-validity-4584479>.
- Asfour, A., Zain, S., Salleh, N., and Grundy, J. (2019). Exploring agile mobile app development in industrial contexts: A qualitative study. *International Journal of Technology in Education and Science*, 3(1):29–46.
- Asha, B. (2018). How to become a freelance software tester? [online] <https://blog.qatestlab.com/2018/08/01/become-freelance-tester/>.
- Baba, Y. and Kashima, H. (2013). Statistical quality estimation for general crowdsourcing tasks. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–562.
- Baldus, B. J., Voorhees, C., and Calantone, R. (2015). Online brand community engagement: Scale development and validation. *Journal of business research*, 68(5):978–985.
- Bano, G., Ali, Q., Khuwaja, S. S., Farah, I., Lal, P., Memon, I., and Zubedi, A. (2019). Comparative analysis of mobile application testing and crowd source software testing. In *2019 8th International Conference on Information and Communication Technologies (ICICT)*, pages 129–134. IEEE.
- Barashev, A. and Li, G. (2019). Content factor analysis of crowd workers’ satisfaction. In *Proceedings of the 2019 the 5th International Conference on e-Society, e-Learning and e-Technologies*, pages 34–38.
- Binh, N. T., Allagui, M., Parissis, I., et al. (2020). Experience report on developing a crowdsourcing test platform for mobile applications. In *International Conference on Computational Collective Intelligence*, pages 651–661. Springer.
- Bolarinwa, O. A. et al. (2015). Principles and methods of validity and reliability testing of questionnaires used in social and health science researches. *Nigerian Postgraduate Medical Journal*, 22(4):195.
- Borst, I. W. (2010). *Understanding Crowdsourcing: Effects of motivation and rewards on participation and performance in voluntary online activities*. Number EPS-2010-221-LIS.
- Boyatzis, R. E. (1998). *Transforming qualitative information: Thematic analysis and code development*. sage.
- Brabham, D. C. (2008). Moving the crowd at istockphoto: The composition of the crowd and motivations for participation in a crowdsourcing application. *First monday*, 13(6).
- Brown, J. D. (2001). Point-biserial correlation coefficients. *Statistics*, 5(3).
- Bryman, A. and Burgess, B. (2002). *Analyzing qualitative data*. Routledge.

- Chen, W.-K., Liu, C.-H., Liang, W. W.-Y., and Tsai, M.-Y. (2018a). Icat: An iot device compatibility testing tool. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pages 668–672. IEEE.
- Chen, X., Jiang, H., Chen, Z., He, T., and Nie, L. (2019). Automatic test report augmentation to assist crowdsourced testing. *Frontiers of Computer Science*, 13(5):943–959.
- Chen, X., Jiang, H., Li, X., He, T., and Chen, Z. (2018b). Automated quality assessment for crowdsourced test reports of mobile applications. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 368–379. IEEE.
- Chen, X., Jiang, H., Li, X., Nie, L., Yu, D., He, T., and Chen, Z. (2020). A systemic framework for crowdsourced test report quality assessment. *Empirical Software Engineering*, 25(2):1382–1418.
- Cheng, J., Teevan, J., Iqbal, S. T., and Bernstein, M. S. (2016). Break it down: A comparison of macro-and microtasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4061–4064.
- Cheng, J., Zhu, Y., Zhang, T., Zhu, C., and Zhou, W. (2015). Mobile compatibility testing using multi-objective genetic algorithm. In *2015 IEEE Symposium on Service-Oriented System Engineering*, pages 302–307. IEEE.
- Chi, M. T., Feltovich, P. J., and Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive science*, 5(2):121–152.
- Crowdtesting platforms (2021). 10 most popular crowdsourced testing companies in 2021. [online] <https://www.softwaretestinghelp.com/crowdsourced-testing-companies/>.
- Crowe, M. and Sheppard, L. (2012). Mind mapping research methods. *Quality & Quantity*, 46(5):1493–1504.
- Cruzes, D. S., Moe, N. B., and Dybå, T. (2016). Communication between developers and testers in distributed continuous agile testing. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, pages 59–68. IEEE.
- Cui, Q., Wang, J., Yang, G., Xie, M., Wang, Q., and Li, M. (2017a). Who should be selected to perform a task in crowdsourced testing? In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 75–84. IEEE.
- Cui, Q., Wang, S., Wang, J., Hu, Y., Wang, Q., and Li, M. (2017b). Multi-objective crowd worker selection in crowdsourced testing. In *SEKE*, volume 17, pages 218–223.
- Cullina, E., Conboy, K., and Morgan, L. (2015). Measuring the crowd: a preliminary taxonomy of crowdsourcing metrics. In *Proceedings of the 11th International Symposium on Open Collaboration*, pages 1–10.
- Dal Sasso, T., Mocci, A., and Lanza, M. (2016). What makes a satisficing bug report? In *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 164–174. IEEE.
- Dalati, S. and Gómez, J. M. (2018). Surveys and questionnaires. In *Modernizing the Academic Teaching and Research Environment*, pages 175–186. Springer.

- Dana, N. et al. (2019). The application of cooperative learning model type problem base learning (pbl) to increase the learning activities of students of class xii mia 3 in sma negeri 1 padang. In *Journal of Physics: Conference Series*, volume 1317, page 012195. IOP Publishing.
- Daniel, F., Kucherbaev, P., Cappiello, C., Benatallah, B., and Allahbakhsh, M. (2018). Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR)*, 51(1):1–40.
- Daniel, S. and Farhad, D. (2014). User requirements of a crowdsourcing platform for researchers: findings from a series of focus groups. In *Proceedings of the 2014 Pacific Asia Conference on Information Systems*, pages 1–11.
- Davies, S. and Roper, M. (2014). What’s in a bug report? In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10.
- Deak, A., Stålhane, T., and Sindre, G. (2016). Challenges and strategies for motivating software testing personnel. *Information and software Technology*, 73:1–15.
- Denscombe, M. (2014). *The good research guide: for small-scale social research projects*. McGraw-Hill Education (UK).
- Donepudi, P. K. (2017). Machine learning and artificial intelligence in banking. *Engineering International*, 5(2):83–86.
- Donepudi, P. K. (2020). Crowdsourced software testing: A timely opportunity. *Engineering International*, 8(1):25–30.
- Drezner, Z., Turel, O., and Zerom, D. (2010). A modified kolmogorov–smirnov test for normality. *Communications in Statistics—Simulation and Computation*, 39(4):693–704.
- Drisko, J. W. and Maschi, T. (2016). *Content analysis*. Pocket Guides to Social Work R.
- Dubey, A., Singi, K., and Kaulgud, V. (2017). Personas and redundancies in crowdsourced testing. In *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, pages 76–80. IEEE.
- Dumas, J. S. (2002). User-based evaluations. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 1093–1117.
- Earley, P. C., Northcraft, G. B., Lee, C., and Lituchy, T. R. (1990). Impact of process and outcome feedback on the relation of goal setting to task performance. *Academy of management journal*, 33(1):87–105.
- Elliott, V. (2018). Thinking about the coding process in qualitative data analysis. *The Qualitative Report*, 23(11):2850–2861.
- Elo, S., Kääriäinen, M., Kanste, O., Pölkki, T., Utriainen, K., and Kyngäs, H. (2014). Qualitative content analysis: A focus on trustworthiness. *SAGE open*, 4(1):2158244014522633.
- Eraslan, S. and Bailey, C. (2019). End-user evaluations. In *Web Accessibility*, pages 185–210. Springer.
- Erickson, T. (2011). Some thoughts on a framework for crowdsourcing. In *Workshop on crowdsourcing and human computation*, pages 1–4.

- Fazzini, M. and Orso, A. (2017). Automated cross-platform inconsistency detection for mobile apps. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 308–318. IEEE.
- Feng, Y. (2019). Leveraging the power of crowds: Automated test report processing for the maintenance of mobile applications.
- Florea, R. and Stray, V. (2019). The skills that employers look for in software testers. *Software Quality Journal*, 27(4):1449–1479.
- Foong, E., Gergle, D., and Gerber, E. M. (2017). Novice and expert sensemaking of crowdsourced design feedback. *Proceedings of the ACM on Human-Computer Interaction*, 1(CSCW):1–18.
- Ford, L. R. and Scandura, T. A. (2018). A typology of threats to construct validity in item generation. *American Journal of Management*, 18(2):132–142.
- Fowler Jr, F. J. (2013). *Survey research methods*. Sage publications.
- Gao, R., Wang, Y., Feng, Y., Chen, Z., and Wong, W. E. (2019). Successes, challenges, and rethinking—an industrial investigation on crowdsourced mobile application testing. *Empirical Software Engineering*, 24(2):537–561.
- Gardner, D. G., Cummings, L. L., Dunham, R. B., and Pierce, J. L. (1998). Single-item versus multiple-item measurement scales: An empirical comparison. *Educational and psychological measurement*, 58(6):898–915.
- Gefen, D., Gefen, G., and Carmel, E. (2016). How project description length and expected duration affect bidding and project success in crowdsourcing software development. *Journal of Systems and Software*, 116:75–84.
- Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., and Abrahamsson, P. (2014). What do we know about software development in startups? *IEEE software*, 31(5):28–32.
- Guaiani, F. and Muccini, H. (2015). Crowd and laboratory testing, can they co-exist? an exploratory study. In *2015 IEEE/ACM 2nd International Workshop on CrowdSourcing in Software Engineering*, pages 32–37. IEEE.
- Guide, H. (2017). Chapter 3: Understanding test quality-concepts of reliability and validity.
- Guo, C., Xu, J., Yang, H., Zeng, Y., and Xing, S. (2014). An automated testing approach for inter-application security in android. In *Proceedings of the 9th international workshop on automation of software test*, pages 8–14.
- Gupta, S. D. (1960). Point biserial correlation coefficient and its generalization. *Psychometrika*, 25(4):393–408.
- Gupta, V., Fernandez-Crehuet, J. M., and Hanne, T. (2020a). Freelancers in the software development process: A systematic mapping study. *Processes*, 8(10):1215.
- Gupta, V., Fernandez-Crehuet, J. M., Hanne, T., and Telesko, R. (2020b). Fostering product innovations in software startups through freelancer supported requirement engineering. *Results in Engineering*, 8:100175.
- Guzman, J. C. et al. (2013). Evaluation of issue tracking and project management tools for use across all csiro radio telescope facilities. In *Proceedings of ICALEPCS*.

- Haas, D., Ansel, J., Gu, L., and Marcus, A. (2015). Argonaut: macrotask crowdsourcing for complex data processing. *Proceedings of the VLDB Endowment*, 8(12):1642–1653.
- Habib, A., Hussain, S., Khan, A. A., Sohail, M. K., Ilahi, M., Mufti, M. R., and Faisal, M. I. (2019). Knowledge based quality analysis of crowdsourced software development platforms. *Computational and Mathematical Organization Theory*, 25(2):122–131.
- Ham, H. K. and Park, Y. B. (2011). Mobile application compatibility test system design for android fragmentation. In *International Conference on Advanced Software Engineering and Its Applications*, pages 314–320. Springer.
- Ham, H. K. and Park, Y. B. (2014). Designing knowledge base mobile application compatibility test system for android fragmentation. *IJSEIA*, 8(1):303–314.
- Hamza, S. A. (2020). A proposed model for an organized remote-work environment.
- Hao, R., Feng, Y., Jones, J. A., Li, Y., and Chen, Z. (2019). Ctras: Crowdsourced test report aggregation and summarization. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 900–911. IEEE.
- Harpe, S. E. (2015). How to analyze likert and other rating scale data. *Currents in Pharmacy Teaching and Learning*, 7(6):836–850.
- Hartati, E. et al. (2020). User satisfaction level on implementation of siskeudes application. In *Journal of Physics: Conference Series*, volume 1500, page 012102. IOP Publishing.
- Heale, R. and Twycross, A. (2015). Validity and reliability in quantitative studies. *Evidence-based nursing*, 18(3):66–67.
- Ho, C. C. and Ting, C.-Y. (2016). Measuring crowd sourced analytics: A review. *International Information Institute (Tokyo). Information*, 19(10B):4891.
- Hosseini, M. (2014). Crowdsourcing definitions and its features: An academic technical report. RCIS 2014 conference.
- Hosseini, M., Shahri, A., Phalp, K., Taylor, J., Ali, R., and Dalpiaz, F. (2015). Configuring crowdsourcing for requirements elicitation. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pages 133–138. IEEE.
- Hossfeld, T., Keimel, C., Hirth, M., Gardlo, B., Habigt, J., Diepold, K., and Tran-Gia, P. (2013). Best practices for qoe crowdtesting: Qoe assessment with crowdsourcing. *IEEE Transactions on Multimedia*, 16(2):541–558.
- Howe, J. (2006). The rise of crowdsourcing. *Wired magazine*, 14(6):1–4.
- Hromic, E. (2018). A study of how trust can be assigned in crowdsourced information retrieval. Master’s thesis.
- Hsieh, H.-F. and Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative health research*, 15(9):1277–1288.
- Huang, J.-f. (2014). Appacts: Mobile app automated compatibility testing service. In *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pages 85–90. IEEE.

- Huang, J.-f. and Gong, Y.-z. (2012). Remote mobile test system: a mobile phone cloud for application testing. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 1–4. IEEE.
- Huang, Y., Wang, J., Wang, S., Liu, Z., Hu, Y., and Wang, Q. (2020). Quest for the golden approach: An experimental evaluation of duplicate crowdtesting reports detection. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–12.
- Illahi, I., Liu, H., Umer, Q., and Zaidi, S. A. H. (2019). An empirical study on competitive crowdsourcing software development: Motivating and inhibiting factors. *IEEE Access*, 7:62042–62057.
- Jay, C., Lunn, D., and Michailidou, E. (2008). End user evaluations. In *Web accessibility*, pages 107–126. Springer.
- Jiang, H., Chen, X., He, T., Chen, Z., and Li, X. (2018). Fuzzy clustering of crowdsourced test reports for apps. *ACM Transactions on Internet Technology (TOIT)*, 18(2):1–28.
- Johnson, T. (2017). Why stack overflow’s documentation effort failed. [online] <https://idratherbewriting.com/2017/08/05/why-stack-overflow-documentation-effort-failed/>.
- Judger, N. (2016). The thematic analysis of interview data: An approach used to examine the influence of the market on curricular provision in mongolian higher education institutions. *Hillary Place Papers (3rd ed.)*, University of Leeds.
- Kam, H.-J., Shah, V., and Ho, S. M. (2017). Bridging the security gap between software developers and penetration testers: A job characteristic theory perspective.
- Kamangar, Z. U., Kamangar, U. A., Ali, Q., Farah, I., Nizamani, S., and Ali, T. H. (2019). To enhance effectiveness of crowdsourcing software testing by applying personality types. In *Proceedings of the 2019 8th International Conference on Software and Information Engineering*, pages 15–19.
- Kamran, M., Rashid, J., and Nisar, M. W. (2016). Android fragmentation classification, causes, problems and solutions. *International Journal of Computer Science and Information Security*, 14(9):992.
- Karbwang, J., Koonrungsesomboon, N., Torres, C. E., Jimenez, E. B., Kaur, G., Mathur, R., Sholikhah, E. N., Wanigatunge, C., Wong, C.-S., Yimtae, K., et al. (2018). What information and the extent of information research participants need in informed consent forms: a multi-country survey. *BMC medical ethics*, 19(1):79.
- Karim, M. R. (2019). Key features recommendation to improve bug reporting. In *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, pages 1–4. IEEE.
- Karlsson, S., Čaušević, A., Sundmark, D., and Larsson, M. (2021). Model-based automated testing of mobile applications: An industrial case study. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 130–137. IEEE.
- Kataike, J., Kulaba, J., and Gellynck, X. (2017). What junior researchers must know before and after data collection: difference between parametric and nonparametric statistics. *Int. J. Sci. Res.*, 6(6):653–8.

- Kaufmann, N., Schulze, T., and Veit, D. (2011). More than fun and money. worker motivation in crowdsourcing—a study on mechanical turk. In *AMCIS*, volume 11, pages 1–11. Detroit, Michigan, USA.
- Khalid, H., Nagappan, M., Shihab, E., and Hassan, A. E. (2014). Prioritizing the devices to test your app on: A case study of android game apps. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 610–620.
- Ki, T., Park, C. M., Dantu, K., Ko, S. Y., and Ziarek, L. (2019). Mimic: Ui compatibility testing system for android apps. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 246–256. IEEE.
- Knop, N. and Blohm, I. (2018). Leveraging the internal work force through crowdtesting “crowdsourcing in banking.
- Knott, D. (2015a). Hands-on mobile app testing. *Indiana: Pearson education Inc.*
- Knott, D. (2015b). *Hands-on mobile app testing: a guide for mobile testers and anyone involved in the mobile app business.* Addison-Wesley Professional.
- Kong, P., Li, L., Gao, J., Liu, K., Bissyandé, T. F., and Klein, J. (2018). Automated testing of android apps: A systematic literature review. *IEEE Transactions on Reliability*, 68(1):45–66.
- Kowalczyk, E., Cohen, M. B., and Memon, A. M. (2018). Configurations in android testing: they matter. In *Proceedings of the 1st International Workshop on Advances in Mobile App Analysis*, pages 1–6.
- Krippendorff, K. (2018). *Content analysis: An introduction to its methodology.* Sage publications.
- Kuutila, M., Mäntylä, M., Farooq, U., and Claes, M. (2020). Time pressure in software engineering: A systematic review. *Information and Software Technology*, 121:106257.
- Lanui, A. and Chiew, T. K. (2019). A cloud-based solution for testing applications’ compatibility and portability on fragmented android platform. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 158–164. IEEE.
- Lazar, J., Feng, J. H., and Hochheiser, H. (2017). *Research methods in human-computer interaction.* Morgan Kaufmann.
- Leedy, P. D., Ormrod, J. E., and Johnson, L. R. (2014). *Practical research: Planning and design.* Pearson Education.
- Leicht, N. (2018). Given enough eyeballs, all bugs are shallow—a literature review for the use of crowdsourcing in software testing. In *Proceedings of the 51st Hawaii International Conference on System Sciences.*
- Leicht, N., Blohm, I., and Leimeister, J. M. (2016a). How to systematically conduct crowdsourced software testing? insights from an action research project.
- Leicht, N., Blohm, I., and Leimeister, J. M. (2017). Leveraging the power of the crowd for software testing. *IEEE Software*, 34(2):62–69.
- Leicht, N., Knop, N., Blohm, I., Müller-Bloch, C., and Leimeister, J. M. (2016b). When is crowdsourcing advantageous? the case of crowdsourced software testing.

- Leicht, N., Rhyn, M., and Hansbauer, G. (2016c). Can laymen outperform experts? the effects of user expertise and task design in crowdsourced software testing.
- Leotta, M., Petito, V., Gelati, L., Delzanno, G., Guerrini, G., and Mascardi, V. (2019). Orchestrated crowdsourced testing of a mobile web application: a case study. In *Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming*, pages 1–6.
- Lewis, J. R. (1995). Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1):57–78.
- Li, G., Wang, J., Zheng, Y., and Franklin, M. J. (2016). Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2296–2319.
- Li, G., Zheng, Y., Fan, J., Wang, J., and Cheng, R. (2017). Crowdsourced data management: Overview and challenges. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1711–1716.
- Li, H., Fang, C., Wei, Z., and Chen, Z. (2019). Cocotest: collaborative crowdsourced testing for android applications. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 390–393.
- Liang, H., Wang, M.-M., Wang, J.-J., and Xue, Y. (2018). How intrinsic motivation and extrinsic incentives affect task effort in crowdsourcing contests: A mediated moderation model. *Computers in Human behavior*, 81:168–176.
- Lim, J.-E., Lee, J., and Kim, D. (2021). The effects of feedback and goal on the quality of crowdsourcing tasks. *International Journal of Human-Computer Interaction*, pages 1–13.
- Lin, F. (2021). Demystifying removed apps in ios app store. *arXiv preprint arXiv:2101.05100*.
- Linares-Vásquez, M., Moran, K., and Poshyvanyk, D. (2017). Continuous, evolutionary and large-scale: A new perspective for automated mobile app testing. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 399–410. IEEE.
- Liu, C.-H. (2018). A cloud platform for compatibility testing of android multimedia applications. In *International Conference on Frontier Computing*, pages 169–178. Springer.
- Liu, C.-H. (2019). A compatibility testing platform for android multimedia applications. *Multimedia Tools and Applications*, 78(4):4885–4904.
- Liu, C.-H., Chen, W.-K., and Chen, S.-L. (2016). A concurrent approach for improving the efficiency of android cts testing. In *2016 International Computer Symposium (ICS)*, pages 611–615. IEEE.
- LIU, C.-H., CHEN, W.-K., and CHEN, S.-L. (2018). A concurrent approach for improving the efficiency of android cts testing. *Journal of Information Science & Engineering*, 34(5).
- Liu, D., Bias, R. G., Lease, M., and Kuipers, R. (2012). Crowdsourcing for usability testing. *Proceedings of the American Society for Information Science and Technology*, 49(1):1–10.
- Liu, M., Peng, X., Jiang, Q., Marcus, A., Yang, J., and Zhao, W. (2018). Searching stackoverflow questions with multi-faceted categorization. In *Proceedings of the Tenth Asia-Pacific Symposium on Internetware*, pages 1–10.

- Liu, Y. and Liu, Y. (2019). The effect of workers' justice perception on continuance participation intention in the crowdsourcing market. *Internet Research*.
- Liu, Y., Zhang, T., and Cheng, J. (2019). Survey on crowd-based mobile app testing. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, pages 521–527.
- Lu, X., Liu, X., Li, H., Xie, T., Mei, Q., Hao, D., Huang, G., and Feng, F. (2016). Prada: Prioritizing android devices for apps by mining large-scale usage data. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 3–13. IEEE.
- Lucko, G. and Rojas, E. M. (2010). Research validation: Challenges and opportunities in the construction domain. *Journal of construction engineering and management*, 136(1):127–135.
- Lykourantzou, I., Khan, V.-J., Papangelis, K., and Markopoulos, P. (2019). Macrotask crowdsourcing: An integrated definition. In *Macrotask Crowdsourcing*, pages 1–13. Springer.
- Machado, L., Kroll, J., Marczak, S., and Prikladnicki, R. (2016). Breaking collaboration barriers through communication practices in software crowdsourcing. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, pages 44–48. IEEE.
- Malhotra, N. K., Kim, S. S., and Agarwal, J. (2004). Internet users' information privacy concerns (iuipe): The construct, the scale, and a causal model. *Information systems research*, 15(4):336–355.
- Mao, K., Capra, L., Harman, M., and Jia, Y. (2015a). A survey of the use of crowdsourcing in software engineering. *Rn*, 15(01).
- Mao, K., Capra, L., Harman, M., and Jia, Y. (2017). A survey of the use of crowdsourcing in software engineering. *Journal of Systems and Software*, 126:57–84.
- Mao, K., Yang, Y., Wang, Q., Jia, Y., and Harman, M. (2015b). Developer recommendation for crowdsourced software development tasks. In *2015 IEEE Symposium on Service-Oriented System Engineering*, pages 347–356. IEEE.
- Marsden, P. V. and Wright, J. D. (2010). *Handbook of survey research*. Emerald Group Publishing.
- Martinez, M. G. (2015). Solver engagement in knowledge sharing in crowdsourcing communities: Exploring the link to creativity. *Research Policy*, 44(8):1419–1430.
- Marwell, G., Oliver, P. E., and Pahl, R. (1988). Social networks and collective action: A theory of the critical mass. iii. *American Journal of Sociology*, 94(3):502–534.
- Mathers, N. J., Fox, N. J., and Hunn, A. (1998). *Surveys and questionnaires*. NHS Executive, Trent.
- Mazumdar, S., Wrigley, S., and Ciravegna, F. (2017). Citizen science and crowdsourcing for earth observations: An analysis of stakeholder opinions on the present and future. *Remote Sensing*, 9(1):87.
- McDaniel, M. A., Schmidt, F. L., and Hunter, J. E. (1988). Job experience correlates of job performance. *Journal of applied psychology*, 73(2):327.
- McDermott, R. (2011). Internal and external validity. *Cambridge handbook of experimental political science*, pages 27–40.

- McDonald, N., Schoenebeck, S., and Forte, A. (2019). Reliability and inter-rater reliability in qualitative research: Norms and guidelines for cscw and hci practice. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–23.
- Meier, F., Bazo, A., Burghardt, M., and Wolff, C. (2013). Evaluating a web-based tool for crowdsourced navigation stress tests. In *International Conference of Design, User Experience, and Usability*, pages 248–256. Springer.
- Mejorado, D. M., Saremi, R., Yang, Y., and Ramirez-Marquez, J. E. (2020). Study on patterns and effect of task diversity in software crowdsourcing. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–10.
- Memar, N., Krishna, A., McMeekin, D., Tan, T., et al. (2020). Investigating information system testing gamification with time restrictions on testers' performance. *Australasian Journal of Information Systems*, 24.
- Messick, S. (1990). Validity of test interpretation and use.
- Mircioiu, C. and Atkinson, J. (2017). A comparison of parametric and non-parametric methods applied to a likert scale. *Pharmacy*, 5(2):26.
- Mohamed, N. H., Hamzah, S. R., Samah, I., et al. (2017). Parental and peer attachment and its relationship with positive youth development. *International Journal of Academic Research in Business and Social Sciences*, 7(9):352–62.
- Mok, R. K., Li, W., and Chang, R. K. (2015). Detecting low-quality crowdtesting workers. In *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, pages 201–206. IEEE.
- Moran, K., Linares-Vásquez, M., Bernal-Cárdenas, C., Vendome, C., and Poshyvanyk, D. (2018). Crashescope: A practical tool for automated testing of android applications. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 15–18. IEEE.
- Müller, H. and Sedley, A. (2015). Designing surveys for hci research. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2485–2486.
- Muntés-Mulero, V., Paladini, P., Manzoor, J., Gritti, A., Larriba-Pey, J.-L., and Mijnhardt, F. (2013). Crowdsourcing for industrial problems. In *International Workshop on Citizen in Sensor Networks*, pages 6–18. Springer.
- mycrowd (2021). How to determine the size of a bug on mycrowd qa. [online] <https://mycrowd.com/how-to-determine-what-size-a-bug-should-be/>.
- Myers, M. D. and Newman, M. (2007). The qualitative interview in is research: Examining the craft. *Information and organization*, 17(1):2–26.
- Nguyen, T. B., Mariem, A., Oum-El-Kheir, A., Ioannis, P., and Le, T. T. B. (2020). Experience report on developing a crowdsourcing test platform for mobile applications.
- Otolo, R. A. (2016). *An information sharing system for crowd-sourced software testers*. PhD thesis, Strathmore University.

- Overflow, S. (2018). Warlords of documentation: A proposed expansion of stack overflow. [online] <https://meta.stackoverflow.com/questions/303865/warlords-of-documentation-a-proposed-expansion-of-stack-overflow>.
- Ozer, D. J. (1985). Correlation and the coefficient of determination. *Psychological Bulletin*, 97(2):307.
- Parekh, R. A. (2009). Knowledge sharing: collaboration between universities and industrial organisations. In *International Conference on Academic Libraries (ICAL-2009)*, pages 146–151. University of Delhi.
- Park, J.-H., Park, Y. B., and Ham, H. K. (2013). Fragmentation problem in android. In *2013 International Conference on Information Science and Applications (ICISA)*, pages 1–2. IEEE.
- Pierce, T. and Wooldridge, D. (2014). Keys to the kingdom: The app store submission process. In *The Business of iOS App Development*, pages 333–376. Springer.
- Popping, R. (2015). Analyzing open-ended questions by means of text analysis procedures. *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, 128(1):23–39.
- Pratama, F. (2020). Designing an online-based questionnaire application for mobile devices. In *Journal of Physics: Conference Series*, volume 1469, page 012042. IOP Publishing.
- Pritha, B. (2021). Internal vs external validity. [online] <https://www.scribbr.com/methodology/internal-vs-external-validity/>.
- Professional QA Group,2020 (2020). Backward compatibility testing. [online] <https://www.professionalqa.com/backward-compatibility-testing>.
- Qarout, R. (2019). *Novel Methods for Designing Tasks in Crowdsourcing*. PhD thesis, University of Sheffield.
- Rami, A. (2020). Android vs ios development: Which platform should i develop for first? [online] <https://easternpeak.com/blog/android-vs-ios-development-which-platform-first/>.
- Rattray, J. and Jones, M. C. (2007). Essential elements of questionnaire design and development. *Journal of clinical nursing*, 16(2):234–243.
- Riccardi, G., Ghosh, A., Chowdhury, S., and Bayer, A. O. (2013). Motivational feedback in crowdsourcing: a case study in speech transcription. In *INTERSPEECH*, pages 1111–1115.
- Robinson, J. (2010). *Triandis' theory of interpersonal behaviour in understanding software piracy behaviour in the South African context*. PhD thesis, University of the Witwatersrand Johannesburg.
- Rosson, M. B., Carroll, J. M., and Hill, N. (2002). *Usability engineering: scenario-based development of human-computer interaction*. Morgan Kaufmann.
- Ryu, Y. S. and Smith-Jackson, T. L. (2006). Reliability and validity of the mobile phone usability questionnaire (mpuq). *Journal of usability studies*, 2(1):39–53.
- Sagala, P. and Andriani, A. (2019). Development of higher-order thinking skills (hots) questions of probability theory subject based on bloom's taxonomy. In *Journal of Physics: Conference Series*, volume 1188, page 012025. IOP Publishing.

- Sakamoto, Y., Tanaka, Y., Yu, L., and Nickerson, J. V. (2011). The crowdsourcing design space. In *International Conference on Foundations of Augmented Cognition*, pages 346–355. Springer.
- Salman, I., Rodriguez, P., Turhan, B., Tosun, A., and Gureller, A. (2020). What leads to a confirmatory or disconfirmatory behaviour of software testers? *IEEE Transactions on Software Engineering*.
- Samuel, T. and Pfahl, D. (2016). Problems and solutions in mobile application testing. In *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17*, pages 249–267. Springer.
- Sánchez-Charles, D., Nin, J., Solé, M., and Muntés-Mulero, V. (2014). Worker ranking determination in crowdsourcing platforms using aggregation functions. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1801–1808. IEEE.
- Sarah, P. (2019). Apple launches a dedicated mobile app for its developer community. [online] <https://techcrunch.com/2019/11/18/apple-launches-a-dedicated-mobile-app-for-its-developer-community/>.
- Sari, A. and Alptekin, G. I. (2017). An overview of crowdsourcing concepts in software engineering. *International Journal of Computers*, 2.
- Sarı, A., Tosun, A., and Alptekin, G. I. (2019). A systematic literature review on crowdsourcing in software engineering. *Journal of Systems and Software*, 153:200–219.
- Sarmah, H. and Hazarika, B. B. (2012). Determination of reliability and validity measures of a questionnaire. *Indian Journal of Education and Information Management*, 5(11):508–517.
- Spier, L. (2012). Qualitative data analysis. *An Initiat. Gebert Ruf Stift*, pages 19–21.
- Siegmund, J., Siegmund, N., and Apel, S. (2015). Views on internal and external validity in empirical software engineering. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 9–19. IEEE.
- Sonam, S., Verma, A., Lal, S., and Sardana, N. (2019). Tagstack: Automated system for predicting tags in stackoverflow. In *2019 International Conference on Signal Processing and Communication (ICSC)*, pages 223–228. IEEE.
- Source, A. (2021). Compatibility test suite. [online] <https://source.android.com/compatibility/cts>.
- Spink, A. (2002). A user-centered approach to evaluating human interaction with web search engines: an exploratory study. *Information processing & management*, 38(3):401–426.
- Stack Overflow (2017). We will stop accepting contributions to documentation on august 8 2017. [online] <https://meta.stackoverflow.com/questions/354217/sunsetting-documentation>.
- Starov, O. (2013). *Cloud platform for research crowdsourcing in mobile testing*. East Carolina University.
- Starov, O. and Vilkomir, S. (2013). Integrated taas platform for mobile development: Architecture solutions. In *2013 8th International Workshop on Automation of Software Test (AST)*, pages 1–7. IEEE.

- Stolee, K. T. and Elbaum, S. (2010). Exploring the use of crowdsourcing to support empirical studies in software engineering. In *Proceedings of the 2010 ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 1–4.
- Ta, H. H. (2018). Assessing the impacts of crowdsourcing in logistics and supply chain operations.
- Taherdoost, H. (2016a). Validity and reliability of the research instrument; how to test the validation of a questionnaire/survey in a research.
- Taherdoost, H. (2016b). Validity and reliability of the research instrument; how to test the validation of a questionnaire/survey in a research. *How to Test the Validation of a Questionnaire/Survey in a Research (August 10, 2016)*.
- Talby, D., Keren, A., Hazzan, O., and Dubinsky, Y. (2006). Agile software testing in a large-scale project. *IEEE software*, 23(4):30–37.
- Tavanapour, N. and Bittner, E. A. C. (2018). The collaboration of crowd workers. In *ECIS*, page 65.
- Tran, T. H. (2020). New product development of applause.
- Treiblmaier, H. and Filzmoser, P. (2011). Benefits from using continuous rating scales in online survey research.
- Tung, Y.-H. and Tseng, S.-S. (2013). A novel approach to collaborative testing in a crowdsourcing environment. *Journal of Systems and Software*, 86(8):2143–2153.
- Tunio, M. Z., Luo, H., Cong, W., Fang, Z., Gilal, A. R., Abro, A., and Wenhua, S. (2017). Impact of personality on task selection in crowdsourcing software development: A sorting approach. *IEEE Access*, 5:18287–18294.
- Unger-Saldaña, K., Peláez-Ballestas, I., and Infante-Castañeda, C. (2012). Development and validation of a questionnaire to assess delay in treatment for breast cancer. *BMC cancer*, 12(1):626.
- us Saqib, N. and Shahzad, S. (2018). Functionality, performance, and compatibility testing: A model based approach. In *2018 International Conference on Frontiers of Information Technology (FIT)*, pages 170–175. IEEE.
- Vasanthapriyan, S., Tian, J., and Xiang, J. (2017). An ontology-based knowledge framework for software testing. In *International Symposium on Knowledge and Systems Sciences*, pages 212–226. Springer.
- Vasilescu, B., Filkov, V., and Serebrenik, A. (2013). Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *2013 International Conference on Social Computing*, pages 188–195. IEEE.
- Vasquez, M. L., Bernal-Cárdenas, C., Moran, K., and Poshyvanyk, D. (2018). How do developers test android applications? *arXiv preprint arXiv:1801.06268*.
- Vaz, L., Steinmacher, I., and Marczak, S. (2019). An empirical study on task documentation in software crowdsourcing on topcoder. In *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, pages 48–57. IEEE.
- Venkitachalam, R. (2014). Validity and reliability of questionnaires.

- Versa (2021). On-time delivery definition and measurement. [online] <https://versa.com/on-time-delivery/>.
- Vilkomir, S. (2018). Multi-device coverage testing of mobile applications. *Software quality journal*, 26(2):197–215.
- Vilkomir, S. and Amstutz, B. (2014). Using combinatorial approaches for testing mobile applications. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, pages 78–83. IEEE.
- Villanes, I. K., Ascate, S. M., Gomes, J., and Dias-Neto, A. C. (2017). What are software engineers asking about android testing on stack overflow? In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, pages 104–113.
- Villanes, I. K., Costa, E. A. B., and Dias-Neto, A. C. (2015). Automated mobile testing as a service (am-taas). In *2015 IEEE World Congress on Services*, pages 79–86. IEEE.
- Vishwakarma, A. R., Lohar, A. A., Desai, B. S., and Mate, P. (2020). Crowdsourcing platform for website testing.
- Vuolle, M., Tiainen, M., Kallio, T., Vainio, T., Kulju, M., and Wigelius, H. (2008). Developing a questionnaire for measuring mobile business service experience. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 53–62. ACM.
- Walsham, G. (2006). Doing interpretive research. *European journal of information systems*, 15(3):320–330.
- Wang, J., Cui, Q., Wang, Q., and Wang, S. (2016). Towards effectively test report classification to assist crowdsourced testing. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10.
- Wang, J., Li, M., Wang, S., Menzies, T., and Wang, Q. (2019a). Images don’t lie: Duplicate crowdtesting reports detection with screenshot information. *Information and Software Technology*, 110:139–155.
- Wang, J., Wang, S., Chen, J., Menzies, T., Cui, Q., Xie, M., and Wang, Q. (2019b). Characterizing crowds to better optimize worker recommendation in crowdsourced testing. *IEEE Transactions on Software Engineering*.
- Wang, J., Yang, Y., Menzies, T., and Wang, Q. (2020). isense2. 0: Improving completion-aware crowdtesting management with duplicate tagger and sanity checker. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 29(4):1–27.
- Wang, J., Yang, Y., Yu, Z., Menzies, T., and Wang, Q. (2018). Crowdtesting: When is the party over? *arXiv preprint arXiv:1805.03218*.
- Wang, M.-M. and Wang, J.-J. (2019). Understanding solvers’ continuance intention in crowdsourcing contest platform: An extension of expectation-confirmation model. *Journal of theoretical and applied electronic commerce research*, 14(3):0–0.
- Wei, L., Liu, Y., and Cheung, S.-C. (2016). Taming android fragmentation: Characterizing and detecting compatibility issues for android apps. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 226–237.

- Wei, L., Liu, Y., and Cheung, S.-C. (2019). Pivot: learning api-device correlations to facilitate android compatibility issue detection. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 878–888. IEEE.
- Wei, L., Liu, Y., Cheung, S.-C., Huang, H., Lu, X., and Liu, X. (2018). Understanding and detecting fragmentation-induced compatibility issues for android apps. *IEEE Transactions on Software Engineering*.
- Westwater, M. G. and Johnson, G. I. (1995). Comparing heuristic, user-centred and checklist-based evaluation approaches. In *Annual Conference of the Ergonomics Society*, pages 538–543. Taylor & Francis.
- Wnuk, K. and Garrepalli, T. (2018). Knowledge management in software testing: a systematic snowball literature review. *e-Informatica Software Engineering Journal*, 12(1).
- Woike, B. A. (2007). Content coding of open-ended responses. *Handbook of research methods in personality psychology*, pages 292–307.
- Wood, A. J., Graham, M., Lehdonvirta, V., and Hjorth, I. (2019). Networked but commodified: The (dis) embeddedness of digital labour in the gig economy. *Sociology*, 53(5):931–950.
- Wu, G., Cao, Y., Chen, W., Wei, J., Zhong, H., and Huang, T. (2017). Appcheck: a crowd-sourced testing service for android applications. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 253–260. IEEE.
- Wulandari, G. S. et al. (2018). The development of learning management system using edmodo. In *IOP Conference Series: Materials Science and Engineering*, volume 336, page 012046. IOP Publishing.
- Xie, M., Wang, Q., Yang, G., and Li, M. (2017). Cocoon: Crowdsourced testing quality maximization under context coverage constraint. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 316–327. IEEE.
- Xu, B., Zheng, H., Xu, Y., and Wang, T. (2016). Configurational paths to sponsor satisfaction in crowdfunding. *Journal of Business Research*, 69(2):915–927.
- Yahaya, A., Lee, G. M., Ma'alip, H., HjTuah, D. K. Z. P., Ali, Z. M., Kasah, K. H., and Sis, Z. B. M. (2014). The impact of knowledge, skill, attitude and confidence in information communication and technology in teaching and learning among teachers in technical school. In *Computational Intelligence in Information Systems: Proceedings of the Fourth INNS Symposia Series on Computational Intelligence in Information Systems (INNS-CIIS 2014)*, volume 331, page 181. Springer.
- Yan, M., Sun, H., and Liu, X. (2014). itest: testing software with mobile crowdsourcing. In *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies*, pages 19–24.
- Yan, M., Sun, H., and Liu, X. (2015). Efficient testing of web services with mobile crowdsourcing. In *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, pages 157–165.
- Yang, K. and Qi, H. (2021). The nonlinear impact of task rewards and duration on solvers' participation behavior: A study on online crowdsourcing platform. *Journal of Theoretical and Applied Electronic Commerce Research*, 16(4):709–726.

- Yang, Y., Karim, M. R., Saremi, R., and Ruhe, G. (2016). Who should take this task? dynamic decision support for crowd workers. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10.
- Ye, B. and Wang, Y. (2016). Crowdrec: Trust-aware worker recommendation in crowdsourcing environments. In *2016 IEEE international conference on web services (ICWS)*, pages 1–8. IEEE.
- Ye, H. J. and Kankanhalli, A. (2015). Investigating the antecedents of organizational task crowdsourcing. *Information & Management*, 52(1):98–110.
- Ye, H. J. and Kankanhalli, A. (2017). Solvers’ participation in crowdsourcing platforms: Examining the impacts of trust, and benefit and cost factors. *The Journal of Strategic Information Systems*, 26(2):101–117.
- Yin, R. K. (1994). Case study research: Design and methods, applied social research. *Methods series*, 5.
- Yu, D., Zhou, Z., and Wang, Y. (2019). Crowdsourcing software task assignment method for collaborative development. *IEEE Access*, 7:35743–35754.
- Yu, S., Fang, C., Cao, Z., Wang, X., Li, T., and Chen, Z. (2021). Prioritize crowdsourced test reports via deep screenshot understanding. *arXiv preprint arXiv:2102.09747*.
- Zanatta, A. L., Machado, L. S., Pereira, G. B., Prikladnicki, R., and Carmel, E. (2016). Software crowdsourcing platforms. *IEEE Software*, 33(6):112–116.
- Zanatta, A. L., Steinmacher, I., Machado, L. S., de Souza, C. R., and Prikladnicki, R. (2017). Barriers faced by newcomers to software-crowdsourcing projects. *IEEE Software*, 34(2):37–43.
- Zhang, T., Chen, J., Luo, X., and Li, T. (2017a). Bug reports for desktop software and mobile apps in github: What’s the difference? *IEEE Software*, 36(1):63–71.
- Zhang, T., Gao, J., and Cheng, J. (2017b). Crowdsourced testing services for mobile apps. In *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 75–80. IEEE.
- Zhang, T., Gao, J., Cheng, J., and Uehara, T. (2015). Compatibility testing service for mobile applications. In *2015 IEEE Symposium on Service-Oriented System Engineering*, pages 179–186. IEEE.
- Zhang, X., Nickels, D., Poston, R., and Dhaliwal, J. (2018). One world, two realities: Perception differences between software developers and testers. *Journal of Computer Information Systems*, 58(4):385–394.
- Zhang, Z. and Cai, H. (2019). A look into developer intentions for app compatibility in android. In *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 40–44. IEEE.
- Zhou, X., Tang, J., Zhao, Y. C., and Wang, T. (2020). Effects of feedback design and dispositional goal orientations on volunteer performance in citizen science projects. *Computers in Human Behavior*, 107:106266.

- Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schroter, A., and Weiss, C. (2010). What makes a good bug report? *IEEE Transactions on Software Engineering*, 36(5):618–643.
- Zogaj, S., Bretschneider, U., and Leimeister, J. M. (2014). Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary. *Journal of Business Economics*, 84(3):375–405.
- Züll, C. (2016). Open-ended questions. *GESIS Survey Guidelines*, 3.

A

Ethical Considerations

Part I: Ethical Approval From The University of Sheffield



Downloaded: 11/04/2020
Approved: 26/02/2019

Qamar Naith
Registration number: 150264248
Computer Science
Programme: Computer Science - Organisations, Information and Knowledge research group

Dear Qamar

PROJECT TITLE: Evaluation of Crowdsourced Testing Platform for Compatibility Testing of Mobile Devices.
APPLICATION: Reference Number 024269

On behalf of the University ethics reviewers who reviewed your project, I am pleased to inform you that on 26/02/2019 the above-named project was **approved** on ethics grounds, on the basis that you will adhere to the following documentation that you submitted for ethics review:

- University research ethics application form 024269 (form submission date: 13/02/2019); (expected project end date: 10/04/2020).
- Participant information sheet 1055329 version 2 (13/02/2019).
- Participant consent form 1055382 version 2 (13/02/2019).

If during the course of the project you need to [deviate significantly from the above-approved documentation](#) please inform me since written approval will be required.

Your responsibilities in delivering this research project are set out at the end of this letter.

Yours sincerely

Alice Tucker
Ethics Administrator
Computer Science

Please note the following responsibilities of the researcher in delivering the research project:

- The project must abide by the University's Research Ethics Policy: <https://www.sheffield.ac.uk/rs/ethicsandintegrity/ethicspolicy/approval-procedure>
- The project must abide by the University's Good Research & Innovation Practices Policy: https://www.sheffield.ac.uk/polopoly_fs/1.671066!/file/GRIPPolicy.pdf
- The researcher must inform their supervisor (in the case of a student) or Ethics Administrator (in the case of a member of staff) of any significant changes to the project or the approved documentation.
- The researcher must comply with the requirements of the law and relevant guidelines relating to security and confidentiality of personal data.
- The researcher is responsible for effectively managing the data collected both during and after the end of the project in line with best practice, and any relevant legislative, regulatory or contractual requirements.

Part II: Consent Form

Informed Consent Form

Title of the project: Mobile Device Compatibility Services Using Crowdsourced Testing Strategy

Name of researcher: Qamar Hamid Naith

Name of participant: *

Email address *

Short-answer text

Valid email address

Date of sign: *

Day, month, year



Please provide your consent by ticking the appropriate boxes below *

	Yes	No
I confirm that I have read and und...	<input type="radio"/>	<input type="radio"/>
I had the opportunity to ask questi...	<input type="radio"/>	<input type="radio"/>
I agree to take part in the project. I...	<input type="radio"/>	<input type="radio"/>
I understand and agree to perform...	<input type="radio"/>	<input type="radio"/>
I understand that my taking part is...	<input type="radio"/>	<input type="radio"/>
I understand that there is no agree...	<input type="radio"/>	<input type="radio"/>

How my information will be used during and after the project.

Please provide your consent by ticking the appropriate boxes below *

	Yes	No
I understand that my responses wi...	<input type="radio"/>	<input type="radio"/>
I understand that my responses wi...	<input type="radio"/>	<input type="radio"/>
I give permission to members of t...	<input type="radio"/>	<input type="radio"/>
I agree that quotes of my respons...	<input type="radio"/>	<input type="radio"/>
I give permission to use all the an...	<input type="radio"/>	<input type="radio"/>

Using the information you provide legally by the researchers in presentations or conferences.

Please provide your consent by ticking the appropriate boxes below *

	Yes	No
I agree to assign the copyright I h...	<input type="radio"/>	<input type="radio"/>
I understand and agree that my w...	<input type="radio"/>	<input type="radio"/>
I understand and agree that other ...	<input type="radio"/>	<input type="radio"/>
I agree that anonymized data can ...	<input type="radio"/>	<input type="radio"/>

B

Requirements Gathering Questionnaire

Part I: Questionnaire

The use of public crowdsourcing method and anonymous crowd testers for testing the compatibility of mobile device and app.

Thanks for agreeing to participate in this survey. The purpose of this study is to understand peoples' impressions about the use of crowdsourcing for software testing and their needs. Their feedback will be used in the development of our project. In this evaluation, we will be focusing on testing various aspects of crowdsourcing.

Section 1: The experiences with Crowdsourcing

1. How do you usually do your mobile app testing?

	Never	Rarely	Sometimes	Often	Always
Crowdsourcing Platforms	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Testing Company	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Automated/Cloud testing Tools	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Which crowdsourcing platforms do you use?

2. Do you use any of the following crowdsourced programming websites to look for solutions to programming problems that you face?

	Never	Rarely	Sometimes	Often	Always
Stack Overflow	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
GitHub	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stack Exchange	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Section 2: The general expectation from the public crowdtesting process

3. Do you think you will be able to find enough a critical mass of testers if you used public crowdtesting?

Yes

No

Maybe

If "Yes" or "maybe", what are the benefits you could gain when dealing with public crowd testers?

4. What are your desires from a crowdsourcing system?

Section 3: Key requirements for the public crowdtesting methodology

5. What are the first typical starting elements to search for a solution to any issue you may face during mobile app development?"

	1	2	3	4
Model Number	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Manufactory (Brand)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Platform (e.g., iOS, Android etc)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
OS Version	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

6. How would you prefer to post or define your problem?

- Title and Structured Form (Divided into sections)
- Title and General Description (e.g., like StackOverflow)

7. The direct interaction between developers and crowd testers is important to perform an effective crowdtesting process on a large-scale

Strongly Disagree 1 2 3 4 5 Strongly Agree

8. How much you think the easy/difficulty design of the results reporting template will affect the encouragement and participation of the crowd testers?

Strongly Disagree 1 2 3 4 5 Strongly Agree

Section 4: Level of confidence in the public crowd

9. Would you trust public and anonymous crowd testers from any country in the world to perform your testing tasks?

Yes No

If No (Why?) -----

If Yes, how much you trust the information provided by crowd testers?

Just a little 1 2 3 4 5 Very much

Section 5: Preferred features and incentives

10. In your opinion, what are the elements that will attract you to work with public crowd testers to execute your tests and would make you leave working with testing companies?

.....

11. In your opinion, what are the incentives that you would be willing to offer to public crowd testers to motivate them to participate more? (please list more than one idea)

.....

Section 6: Evaluation of the performance and quality of work

12. Consider you are working with public and anonymous crowd testers; how do you will know that they have really performed the test?

.....

13. How do you want crowd testers to prove that the results they provided are correct?

.....

Section 7: Required information for effective crowdtesting process

14. What are the four essential parts of the information that developers must provide it to crowd testers as part of task defining stage?

15. What are the four essential parts of the information that testers should provide it as part of their feedback in the test reports?

Part II: Critical Appraisal Sheet of Reliability and Validity Test



The
University
Of
Sheffield.

Critical Appraisal Sheet

Researcher: Qamar Naith, Ph.D student in Computer Sciences - The University of Sheffield,

Tel: (+44) -784 – 630 – 0066

E-mail: qhnaith1@Sheffield.ac.uk

The purpose of this invitation:

You are being invited to participate in this research study to test the reliability and validate of the developed survey questionnaire.

Instructions for reliability of the questionnaire :

This questionnaire has 15 questions. Kindly review this it and provide your feedback on the following:

1. Simplicity (Understandability) of each question (how clear and easy each question to be understood?);
2. Correctness of each question (how correct each question in terms of language and formulation?);

Instructions for the validation of the questionnaire :

This questionnaire has 15 questions. Kindly review this it and provide your feedback on the following:

1. Relevance of each question (how important and relevant the question to crowdtesting, compatibility testing of mobile devices and fragmentation issues?);
2. Feasibility of each question (is it in the testable format?);
3. Essentiality of each question (how necessary is the question to achieve the aim of the study?).

Sheet (A): The Critical Appraisal Sheet for Measuring Reliability of Questionnaire

	Simplicity										Integrity									
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Q1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q6	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q8	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q9	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q10	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q11	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q12	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q13	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q14	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q15	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Name: _____

Signature: _____

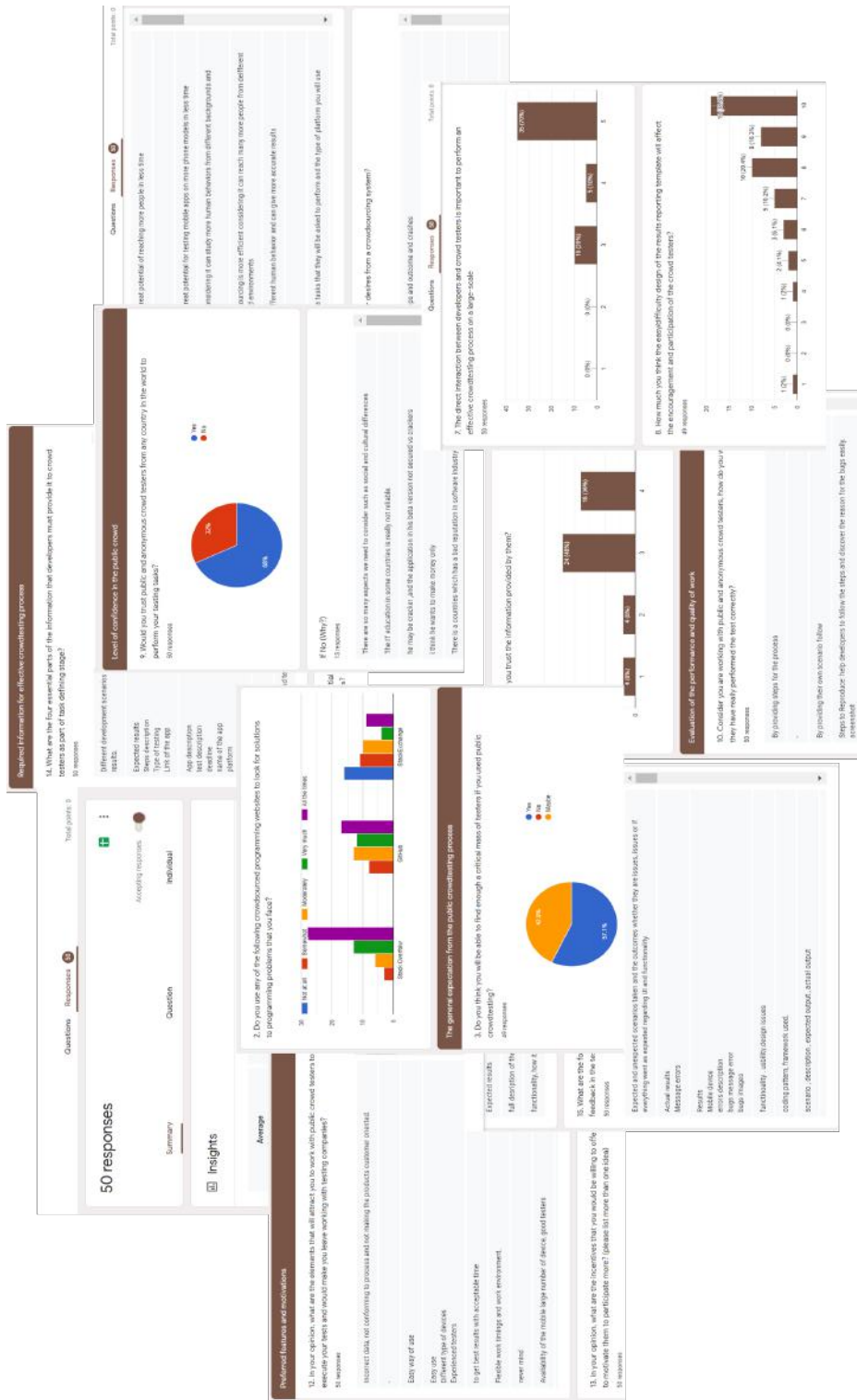


Fig. B.1 Screenshots of the developers' answers collected by Google Form.

C

Empirical Evaluation Questionnaires

Part I: Online Questionnaire For Developers

Demographic Information

1- Work Status: *

 Developer

2- Operating System Type: *

 Android iOS

3- Job Type: *

 Freelancer Employee

4-If you are an employee please select from the following options:

 Industry - Big Organization (up to 2000 employees) Industry - Small Organization (approximately 50-200 employees) Academia Government

Please provide the name of the organization (optional)

Your answer _____

5- Years of experience: *

Your answer _____

6- I have used the platform for (specify for how many days) *

Your answer _____

7- Country you live in *

Your answer _____

Table C.1 Questionnaire Questions for Developer (CSTE-Q/D)

Questions (structured to solicit Likert-style " Strongly Disagree to Strongly Agree responses)		Source of Questions
Effectiveness		
Tasks Defining and Distribution		
8	It enables me to define and publish testing tasks quickly with less effort.	New developed
9	It enables me to distribute testing tasks on a large scale quickly.	New developed
10	The task design method enables me to define the whole app as simple multiple tasks easily.	New developed
11	The two different levels of testing (feature and code) enable me to perform more effective compatibility tests	New developed
12	It takes less time and effort to write task requirements.	New developed
Accessing Specialised Testers Skills		
13	It provides an easy method of selecting testers in a short time.	New developed
14	The suggestion of a list of eligible testers for performing a specific task helps facilitate the selection of testers.	New developed
15	Selection of testers with different skills and expertise areas to participate in and work on a specific task is helpful.	Adapted (Ye and Kankanhalli, 2015)
16	Insights provided into testers' performance helps select and invite several testers with specialised knowledge to work on tasks.	Adapted (Ye and Kankanhalli, 2015)
Privacy and Protection		
17	It enables me to control the privacy of the task and prevent unauthorized testers from accessing it.	Developed (Malhotra et al., 2004)
18	It has an effective method to protect testers' sensitive information (their work quality and reliability level) from other testers.	Developed (Malhotra et al., 2004)
19	It has an effective strategy to protect the testing environment from having non-active and unreliable testers.	Developed (Malhotra et al., 2004)
Tracking Task Status		
20	Tracking the number of tests performed and the type of devices used for testing a specific task is helpful.	New developed
21	It is very useful to track the task status of the published task during the active test cycle	New developed
Outcomes Diversity		
22	Different unexpected issues can be discovered in the early stages of an app's development process through this crowdtesting approach.	Adapted (Ye and Kankanhalli, 2015)
23	Results obtained by different testers' backgrounds represent different ways of thinking about implementing the app's functionalities.	Adapted (Ye and Kankanhalli, 2015)
Results Aggregation and Tracking		
24	It has an effective and useful reports aggregation and tracking mechanism.	Developed (Guzman et al., 2013)
25	The automated tracking mechanism helps reduce the time required for collecting and organizing reports.	Developed (Guzman et al., 2013)
26	It enables me to view and filter testing reports easily.	Developed (Guzman et al., 2013)
Results Evaluation and Quality Control		
27	It requires less time and effort to assess the performance of testers on performed tasks.	New developed
28	The evidence provided by public testers regarding testing results was NOT helpful to gauge their accuracy.	New developed
29	The insight provided into the results quality distribution helps understand the reason for low quality work.	New developed
30	The quality evaluation metrics enable me to perform a quick and fair evaluation of testers' works.	New developed
Cost Effectiveness (Cost Reduction)		
31	We can cover more devices to test our tasks at a lower price through this approach.	Adapted (Ye and Kankanhalli, 2015)
32	It is relatively cheaper than other approaches relying on a middleman.	Adapted (Ye and Kankanhalli, 2015)

Questions (structured to solicit Likert-style " Strongly Disagree to Strongly Agree responses)		Source of Questions
Interaction between Peer and Workflow		
33	Direct interaction between developers and testers, without middlemen, DOES NOT help me understand the test results more.	New developed
34	The two-way communication mechanism provided by this approach's workflow is better than what I expected.	Adapted (Wang and Wang, 2019)
35	Direct interaction reduces delays caused by the middlemen figures alike (managers or leaders) as compared to other approaches.	New developed
Tester Reliability and Trustworthiness		
36	The reliability tracking method is effective.	New developed
37	The reliability tracking method is fair	New developed
38	It provides an effective classification method to reduce developers' concerns about working with public testers.	New developed
Tester Motivation and Incentivisation		
39	The rewarding mechanism is effective.	New developed
40	The rewarding mechanism is fair.	New developed
41	The rewarding mechanism is motivating.	New developed
42	It enables me to easily control and track the non-paid testers status.	New developed
Knowledge Sharing / Wiki		
43	It provides an effective knowledge-sharing environment between worldwide developers' and testers' communities.	New developed
44	It has an effective documentation and knowledge sharing mechanism.	New developed
45	Documentation of issues and their solutions helps me to answer my questions and develop high-quality apps.	New developed
Ease of use		
46	It is easy to use the proposed approach in daily work routine and industry practices.	Adapted (USE)
47	Both Android and iOS specialists can easily use and interact with the proposed approach.	Adapted (USE)
Benefits / Advantages		
48	It improves the development of mobile apps by incorporating target users early in the development process.	New developed
49	It provides me with an opportunity to receive help from other developers' communities.	adapted (Baldus et al., 2015)
50	It is an effective solution to stay informed about new issues with the development of apps and different architectures of mobile devices	New developed
51	It provides me with an opportunity to cover all versions of mobile devices and OS quickly.	New developed
52	It supports large-scale collaboration and communication among developers and researchers in the domain.	New developed
53	It helps increase test coverage and obtain better results in a shorter time.	New developed
54	It accelerates the development process and time-to-market delivery of an app with less cost.	New developed
55	It helps obtain important information about the users' behaviours or interactions with the app, unlike other approaches.	New developed
Satisfaction		
56	Overall, I am satisfied with the use of this approach to conduct my test in the future.	Adapted (Wang and Wang, 2019)
57	I am satisfied with the nature of the crowdtesting workflow and service provided.	Adapted (Wang and Wang, 2019)
58	I am satisfied with the benefits that will result from the use of this approach in the future.	Adapted (Wang and Wang, 2019)

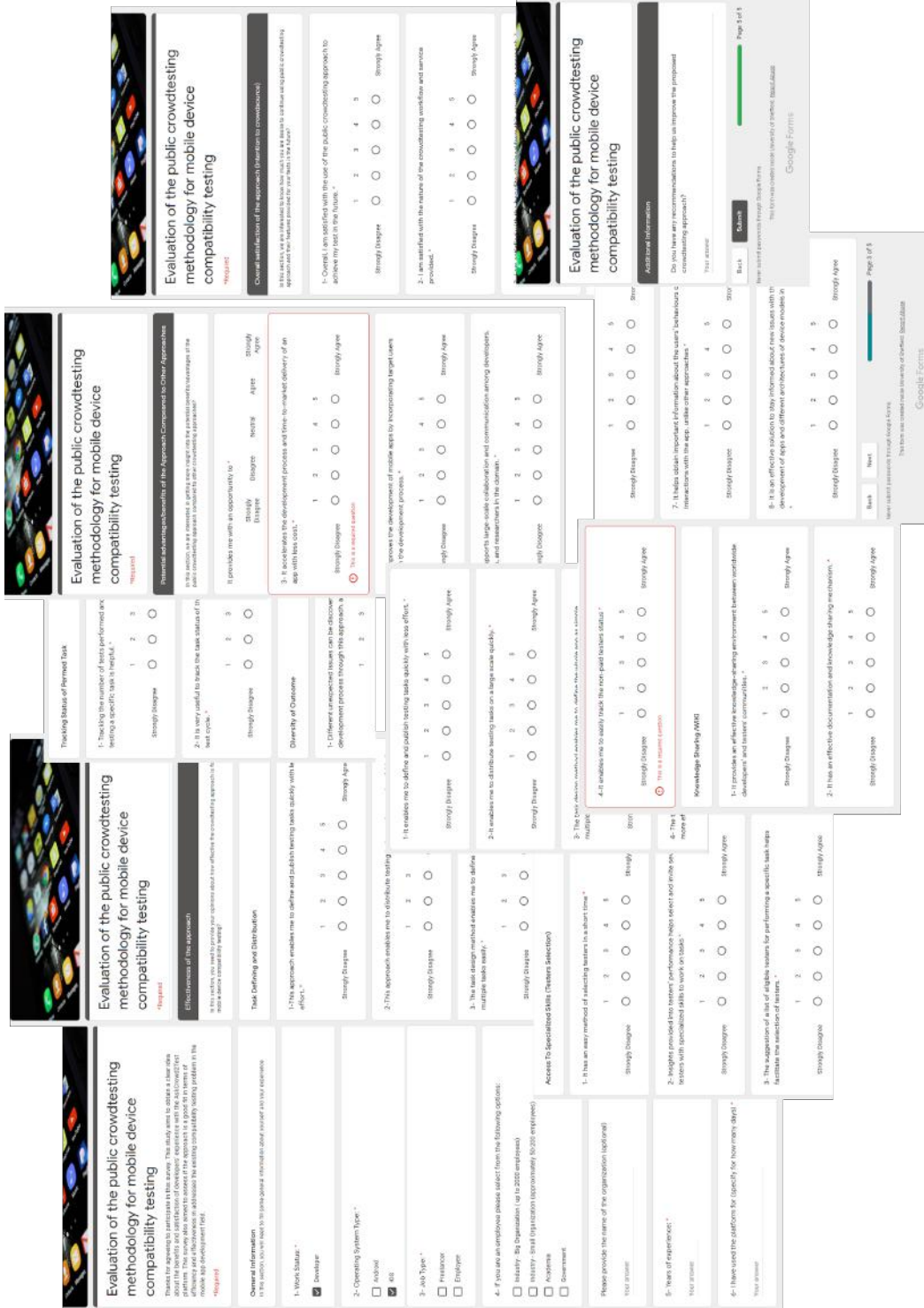


Fig. C.1 Screenshots of the online questionnaire for developers

Part II: Online Questionnaire For Testers

Demographic Information

1- Work Status: *

 Tester

2- Operating System Type: *

 Android iOS

3- Job Type: *

 Freelancer Employee

4- If you are an employee please select from the following options:

 Industry - Big Organization (up to 2000 employees) Industry - Small Organization (approximately 50-200 employees) Academia Government

Please provide the name of the organization (optional)

Your answer _____

5- Years of experience: *

Your answer _____

6- I have used the platform for (specify for how many days) *

Your answer _____

7- Country you live in *

Your answer _____

8- List of software testing certifications (name) *

Your answer _____

Table C.2 Questionnaire Questions for Tester (CSTE-Q/T)

Questions (structured to solicit Likert-style " Strongly Disagree to Strongly Agree responses)		Source of Questions
Effectiveness		
<i>Task Selection</i>		
9	It provides an easy and effective method of selecting tasks	New developed
10	The list of suggested tasks helps me to select and perform more tasks in a short time.	New developed
<i>Accessing Task Requirements</i>		
11	The information and instructions given by developers are sufficient to perform an effective testing process.	Adapted (Lewis, 1995)
12	The presentation of tasks requirements is clear and understandable.	Adapted (Lewis, 1995)
13	Information on task complexity level helps me perform the most suitable test and obtain better quality results.	Adapted (Lewis, 1995)
<i>Privacy and Protection</i>		
14	It has an effective way to prevent ineligible testers from accessing and selecting the private task.	Developed (Malhotra et al., 2004)
15	It provides an effective and secure method of linking device information with the submission form.	Developed (Malhotra et al., 2004)
<i>Results Submission</i>		
16	The way of submitting results is simple and uncomplicated.	New developed
17	Automatic detection service of mobile data helps me to collect and submit more accurate results in fewer steps.	New developed
18	It enables me to perform that same task with different mobile devices quickly with less effort.	New developed
<i>Interaction between Peer and Workflow</i>		
19	Direct interaction between developers and testers, without middlemen, DOES NOT help to understand more test requirements and effective completion of tasks.	New developed
20	Increasing the strength of the connection and interaction with developers and the other testers community makes me want to participate more.	New developed
21	It reduces delays caused by the middlemen figures alike (managers or leaders) as compared to other approaches.	New developed
<i>Feedback Given</i>		
22	The direct and clear information about the effectiveness of my performance helps me to improve my work in the future.	Adapted (Martinez, 2015)
23	The feedback about how well I am doing increases my satisfaction and helps me to continue to participate and work.	Adapted (Martinez, 2015)
<i>Results Evaluation and Quality Control</i>		
24	It has an effective and useful mechanism for tracking the status of submitted reports.	New developed
25	The quality evaluation metrics are fair and increase my commitment and desire to keep participating.	New developed
26	The insight provided into the history of my work quality is helpful to know how well I am doing.	New developed
<i>Testers Reliability and Trustworthiness</i>		
27	The use of this approach helps improve my reputation among other testers in the global community.	New developed
28	I believe that the reliability level identified for each tester is fair and trustworthy.	New developed
<i>Tester Motivation and Incentimisation</i>		
29	The reward value is fair and reflects the effort that I have put into crowdtesting work.	Adapted (Jiu and Liu, 2019);
30	The pay-per-device increases the motivations of testers to do the test on more devices and get a higher quality of results.	New developed
31	The reward information provided for each completed task seems fair to me.	Adapted (Ta, 2018)

Questions (structured to solicit Likert-style " Strongly Disagree to Strongly Agree responses)		Source of Questions
Knowledge Sharing / Wiki		
32	It provides an effective knowledge-sharing environment between worldwide developers' and testers' communities.	New developed
33	It has an effective documentation and knowledge sharing mechanism.	New developed
34	Documentation of testing steps and scenarios helps to improve my testing strategy in different testing areas.	New developed
Ease of use		
35	It is easy to use the proposed approach in daily work routine and industry practices.	Adapted (USE)
36	Both Android and iOS specialists can easily use and interact with the proposed approach.	Adapted (USE)
Benefits / Advantages		
37	It provides me with an opportunity to receive help from other communities.	Adapted (Baldus et al., 2015)
38	It provides me with opportunities to improve my testing knowledge and skills in different areas.	Adapted (Wang and Wang, 2019; Ye and Kaikanhalli, 2017)
39	It provides me with an opportunity to perform tests in the areas that I am good at.	Adapted (Wang and Wang, 2019; Liang et al., 2018)
40	It provides me with an opportunity to perform the test in different ways.	New developed
41	It provides a suitable environment for professional and beginner testers.	New developed
42	It supports large-scale collaboration and communication among testers and researchers in the domain.	New developed
43	It enables me to perform test anywhere and at any time I need.	New developed
44	Participating and using this approach lets me feel a sense of personal achievement.	Adapted (Wang and Wang, 2019; Liang et al., 2018)
Satisfaction		
45	Overall, I am satisfied with the use of the public crowdtesting approach to conduct the test in the future.	Adapted (Wang and Wang, 2019)
46	I am satisfied with the nature of the crowdtesting workflow and service provided.	Adapted (Wang and Wang, 2019)
47	I am satisfied with the benefits that will result from the use of this approach in the future.	Adapted (Wang and Wang, 2019)

Privacy and Protection
1- It has an effective way to prevent navigable factors to access and select the private task. *

Effectiveness of the approach
1- It has an effective way to prevent navigable factors to access and select the private task. *

Task Selection
1- It has an easy and effective method of selecting tasks. *

Demographic Information
1- Work Status *

5- Years of experience

6- I have used the platform for (speed) for how many days? *

7- Country you live in *

B- List of software testing certifications (name)

Efficiency of the approach
1- It has an effective way to prevent navigable factors to access and select the private task. *

Task Selection
1- It has an easy and effective method of selecting tasks. *

2- The list of suggested tasks helps me to select and perform more tasks in a short time. *

1- Direct interaction of developers and testers, without moderators, DOES NOT help to understand more test requirements and effective completion of tasks. *

1- It provides an effective knowledge sharing environment between mobile developers and testers' communities. *

2- It has an effective documentation and knowledge sharing mechanism *

3- The documentation of testing steps and execution helps to improve my testing strategy in different testing areas. *

1- The reason value is fair and reflects the effort that you put into crowdtesting work. *

2- The pay per device increases the motivation of testers to do the field on more devices and get a higher quality of results. *

1- It provides me with a considerable opportunity to *

1- I can help from other communities.

2- Perform tests in the areas that I am good at.

4- Improve my testing skills in different areas

1- The reason value is fair and reflects the effort that you put into crowdtesting work. *

2- The pay per device increases the motivation of testers to do the field on more devices and get a higher quality of results. *

1- The reason value is fair and reflects the effort that you put into crowdtesting work. *

4- It supports large-scale collaboration and communication among testers and researchers in the domain.

7- The approach enables me to perform test anywhere and at any time I need to.

1- Overall, I am satisfied with the use of the public crowdtesting approach to address the test in the future. *

2- I am satisfied with the nature of provided. *

Overall advantages/benefits of the Approach Compared to Other Approaches
1- It provides me with a considerable opportunity to *

Overall satisfaction of the approach (intention to participate)
1- Overall, I am satisfied with the use of the public crowdtesting approach to address the test in the future. *

Additional Information
Do you have any recommendations to help us improve the proposed crowdtesting platform? *

Next

Progress Bar: 100% of 5

Fig. C.2 Screenshots of the online questionnaire for Testers

Part III: Questions deleted after reliability and validity tests

Table C.3 Questions deleted after reliability and validity tests

Code	Question	CI	Category
TDD	It has an effective method to share the app's beta version or the code link in external sources with public testers.	0.258	Developer
TS	It has the potential to avoid testing unnecessary tasks	0.510	Testers
MI	Participating and performing tasks with this crowdtesting approach help testers to earn more money.	0.480	Developer
BNF	The tasks' dashboard accelerates the process of performing multiple tests.	0.272	Developer
SUB	The submission mechanism helps prevent human errors when typing the details of the mobile device.	0.302	Testers

Part IV: Normality test using One-Sample Kolmogorov-Smirnov Test.

Table C.4 Testing the normality distribution of the developers' and testers' responses using "One-Sample Kolmogorov-Smirnov Test".

Developers		Effectiveness	Benefits	Satisfaction
N		34	34	34
Normal Parameters ^{a,b}	Mean	3.8735	3.8787	3.7255
	Std. Deviation	.79176	.62195	.96912
Most Extreme Differences	Absolute	.215	.166	.170
	Positive	.121	.122	.096
	Negative	-.215	-.166	-.170
Kolmogorov-Smirnov Z		1.253	.965	.993
Asymp. Sig. (2-tailed)		.086	.309	.277

Testers		Effectiveness	Benefits	Satisfaction
N		31	31	31
Normal Parameters ^{a,b}	Mean	4.1141	4.1048	4.1398
	Std. Deviation	.67432	.61500	.81547
Most Extreme Differences	Absolute	.157	.161	.207
	Positive	.094	.102	.146
	Negative	-.157	-.161	-.207
Kolmogorov-Smirnov Z		.874	.895	1.151
Asymp. Sig. (2-tailed)		.430	.400	.141

D

Additional Results of Both Studies

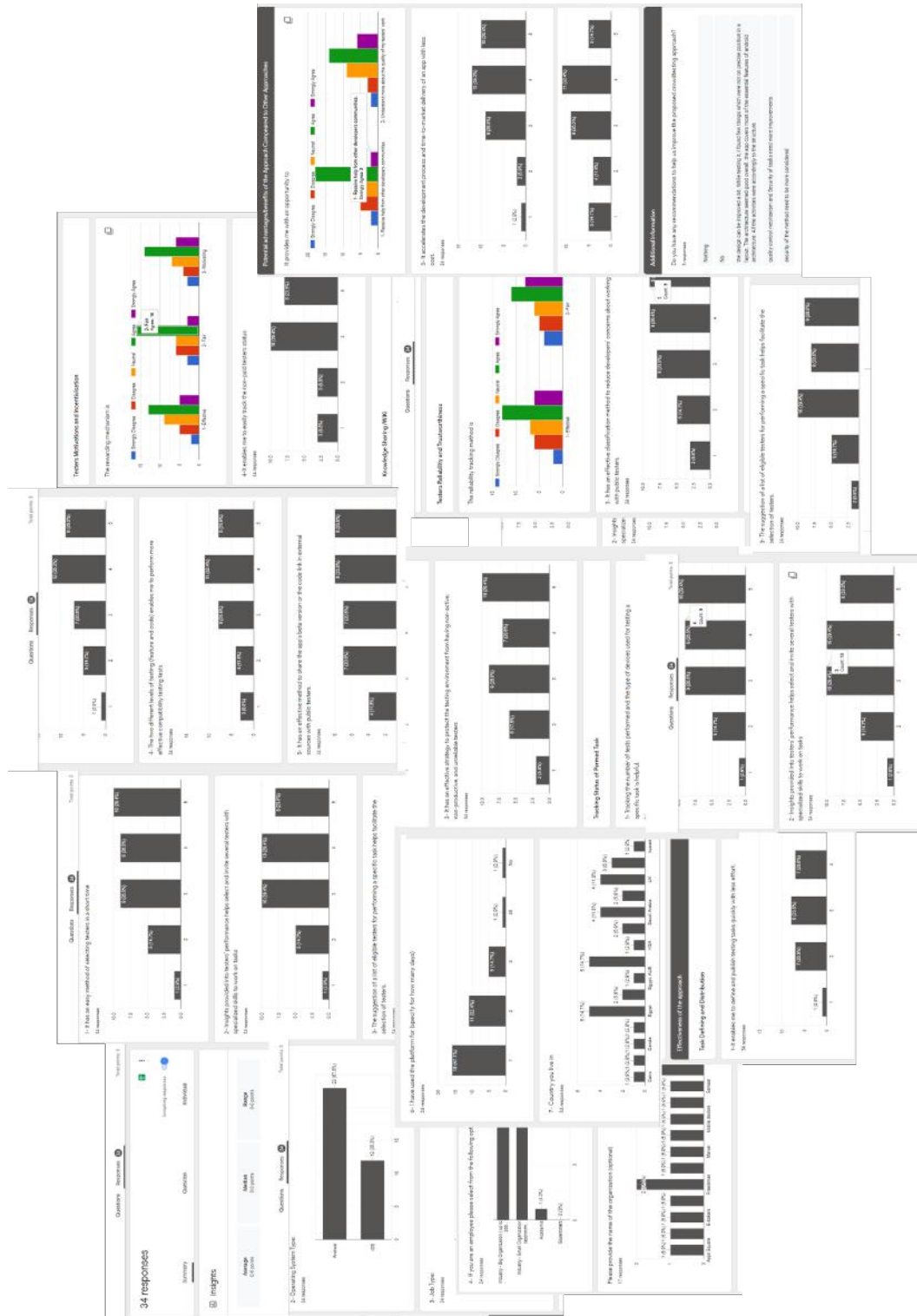


Fig. D.1 Screenshots of the developers' answers' collected by Google Form.

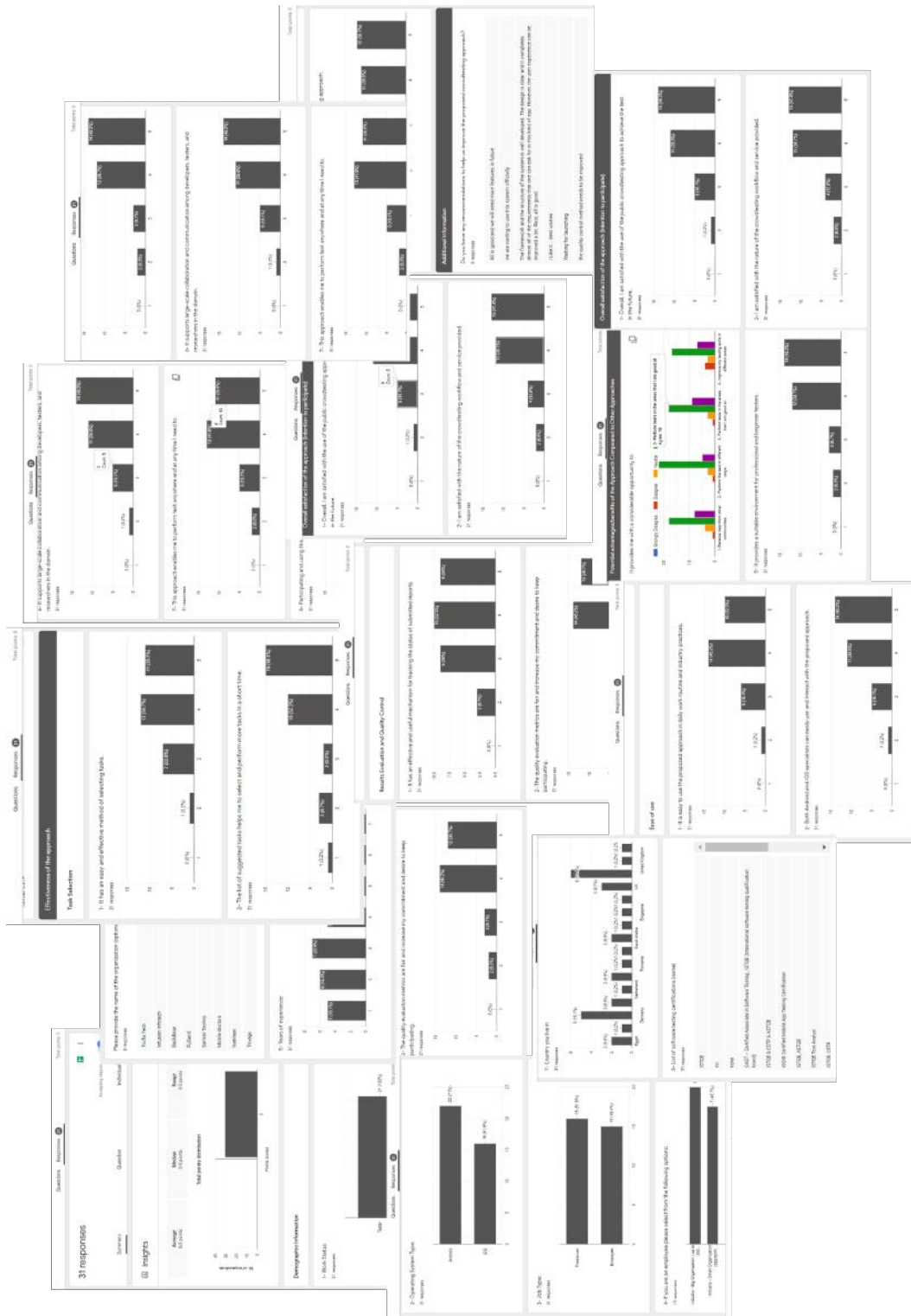


Fig. D.2 Screenshots of the testers' answers collected by Google Form.

