



INTERNATIONAL  
HELLENIC  
UNIVERSITY

# **Affective system monitoring personal expenses, helping the user to stay on budget**

**Tziala Aikaterini**

SID: 3306160009

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

*Master of Science (MSc) in Mobile and Web Computing*

DECEMBER 2018

THESSALONIKI – GREECE



INTERNATIONAL  
HELLENIC  
UNIVERSITY

# **Affective system monitoring personal expenses, helping the user to stay on budget**

**Tziala Aikaterini**

SID: 3306160009

**Supervisor: Dr. Christos Moridis**

**SCHOOL OF SCIENCE & TECHNOLOGY**

A thesis submitted for the degree of

*Master of Science (MSc) in Mobile and Web Computing*

**DECEMBER 2018**

**THESSALONIKI – GREECE**

# Acknowledgements

I gladly take this opportunity to express my gratitude to the people who contributed in this project.

First and foremost, I would like to thank my thesis supervisor Dr. Christos Moridis for the precious advice and guidance he provided throughout this time, and for motivating and challenging me to reach beyond my expectations.

Moreover, I would like to thank my mentor Antonis Korosidis for initiating me in the fascinating world of programming, which allowed me to discover my truly passion. I would also like to thank him for providing the hosting server for the developed app for this project, which helped to facilitate this research.

I would also like to thank those who participated in my survey by using the developed app, for their valuable contribution to my research.

A big thank you to my friends Yiannis Karamitsos, Penelope Amarantidou, Maria Mantziari, Yiannis Tsiknas, and Anna Progrevnoi for being there for me, even in the toughest moments of the past year. Each one supported me in a unique way, whether this was endless discussions, encouragement, care, laughter or understanding.

Finally, I would like to dedicate this thesis to my dearest parents, Vasilis and Eleni, and my siblings, Dimitris and Angela, who always believe and keep full faith in me. I would also like to express my profound gratefulness to them for all the sacrifices they made, their kind and valuable support, to make my dreams come true, so far in my lifetime.

Thank you for everything!

# Abstract

This thesis was written as a part of the MSc in Mobile and Web Computing at the International Hellenic University. This study aimed in developing an affective system that promotes budget adherence. The focus was twofold. The first part aimed at developing a state-of-the-art expense monitoring system that helps users stay on budget. Innovative and modern web technologies were employed in order to create a fully functional, mobile-friendly, progressive web app. The second part focused on the study of affective feedback techniques. Specifically, the study aimed to answer how the implementation of animated GIFs in the system as a mean to provide affective feedback can influence budget adherence in populations. One hundred thirty one (131) volunteers used the developed app for a three-week period. Data was gathered through the web app during this period and a statistical analysis was conducted. Key findings suggested that the developed system aids in and promotes budget adherence. The animated GIFs as a mean to provide affective feedback was expected to influence the budget adherence. However, this hypothesis was not confirmed. The developed prototype system has the potential to be expanded and become more compound facilitating thus future research both on web app development technologies and on study of affective feedback techniques.

Keywords: affective feedback, animated GIFs, budget management, expense monitoring system, progressive web application.

Tziala Aikaterini

9 December 2018

# Contents

<b>ACKNOWLEDGEMENTS</b> .....	<b>III</b>
<b>ABSTRACT</b> .....	<b>IV</b>
<b>CONTENTS</b> .....	<b>V</b>
<b>1 INTRODUCTION</b> .....	<b>1</b>
1.1 THESIS OUTLINE .....	3
<b>2 THEORETICAL AND CONCEPTUAL FRAMEWORK</b> .....	<b>4</b>
2.1 WHAT IS FEEDBACK .....	4
2.2 TYPES OF FEEDBACK.....	5
2.3 POSITIVE AND NEGATIVE FEEDBACK.....	9
2.4 EFFECTIVE AND CONSTRUCTIVE FEEDBACK.....	10
2.5 AFFECTIVE FEEDBACK .....	12
2.6 AFFECTIVE FEEDBACK TECHNIQUES.....	13
2.6.1 <i>Textual Information, Colors and Avatars</i> .....	14
2.6.2 <i>Emoticons and emojis</i> .....	15
2.6.3 <i>Images and GIFs</i> .....	18
2.6.4 <i>Audio and Sounds</i> .....	21
2.6.5 <i>Videos</i> .....	23
<b>3 THE PROTOTYPE SYSTEM</b> .....	<b>26</b>
3.1 THE <i>BUDGET MANAGER</i> SYSTEM .....	26
3.2 FUNCTIONAL REQUIREMENTS.....	26
3.3 NON-FUNCTIONAL REQUIREMENTS.....	28
3.4 PROVIDING FEEDBACK TO USERS .....	30
<b>4 SYSTEM ARCHITECTURE &amp; DESIGN</b> .....	<b>36</b>
4.1 SYSTEM ARCHITECTURE.....	36
4.2 DATABASE DESIGN .....	37
4.2.1 <i>Description of Entities</i> .....	38
4.2.2 <i>Table Specifications</i> .....	39
4.3 BACK-END DESIGN .....	50
4.4 USER CHARACTERISTICS AND INTERFACES .....	52

4.4.1	<i>Unauthorized users</i> .....	52
4.4.2	<i>Authorized users</i> .....	55
<b>5</b>	<b>IMPLEMENTATION OF THE SYSTEM .....</b>	<b>61</b>
5.1	IMPLEMENTATION OVERVIEW .....	61
5.2	BACK-END DEVELOPMENT .....	63
5.3	FRONT-END DEVELOPMENT.....	70
5.3.1	<i>Web Development practices</i> .....	70
5.3.2	<i>Development Orientation</i> .....	73
5.3.3	<i>App Shell Implementation</i> .....	75
5.3.4	<i>Dynamic Content, UI features and libraries</i> .....	80
5.3.5	<i>Fonts, Libraries and Plugins</i> .....	83
5.3.6	<i>The Web App Manifest</i> .....	87
5.3.7	<i>Service Worker implementation</i> .....	90
5.3.8	<i>Use of the Gulp Build Tool</i> .....	93
<b>6</b>	<b>EVALUATION AND RESULTS .....</b>	<b>97</b>
6.1	RESEARCH OBJECTIVE.....	97
6.2	DATA COLLECTION AND PARTICIPANTS .....	98
6.3	ANALYSIS.....	98
6.4	DESCRIPTIVE STATISTICS RESULTS .....	99
6.5	INFERENTIAL STATISTICS RESULTS .....	104
<b>7</b>	<b>DISCUSSION.....</b>	<b>107</b>
7.1	CONCLUSIONS AND CONTRIBUTIONS.....	107
7.2	LIMITATIONS.....	108
7.3	THEORETICAL AND PRACTICAL IMPLICATIONS .....	109
	<b>BIBLIOGRAPHY .....</b>	<b>113</b>
	<b>APPENDIXES .....</b>	<b>124</b>
	ABBREVIATIONS AND DEFINITIONS.....	124
	RESOURCES FOR DEVELOPMENT .....	129
	<i>APIs, Plugins and Libraries</i> .....	129
	<i>Software and Tools</i> .....	129
	DESCRIPTIVE STATISTICS .....	132
	<i>Tables</i> .....	132
	<i>Graphs</i> .....	143

LIST OF TABLES .....	147
LIST OF FIGURES.....	149





# 1 Introduction

Modern life offers a plethora of services and goods for consumers to choose from. With too many options available, people often indulge in temptations and spend much more than they really need. Leading a specific lifestyle can significantly increase the cost of living and financial discrepancies may arise. As a result, the need to monitor the expenses and maintain a healthy financial balance becomes more urgent.

Therefore, the need to create a spending plan, a budget, emerges. A budget is an estimation of revenue and expenses over a specified future period of time, that is compiled and re-evaluated on a periodic basis (“What is a Budget? Budgeting Terms and Tips”, n.d.; “What is a Budget?” n.d.). According to Fontinelle (2017), creating a budget plan is a proactive approach, rather than a reactive one, in maintaining a balance between income and expenses. It includes making long-term and short-term projections about personal finances and allows determining in advance and prioritizing the spending focus. A budget aids in making intentional and conscious decisions on spending choices and money allocation (Fontinelle, 2017). Even though budgeting is an entirely optional process, it constitutes a vital component in financial success of a person, a family, a group of people, a business, a government, a country, a multinational organization or just about anything else that makes and spends money (“What is Budgeting? What is a Budget?”, n.d.).

Currently, there is a lot of discussion going on regarding the best options and approaches on setting and sticking on a budget. Many articles have been written on steps, tips and tools that promote effective and efficient budgeting (Berger, 2015a; Berger, 2015b; Williams, 2015; Colston, 2018, Frank, 2018, Vohwinkle, 2018; Rosenberg, 2018; Schneider & Auten, 2018; O'Shea & Schwahn, 2018). Another aspect is concerned with the necessity and the benefits of making the process of budgeting and tracking of expenses a habit (Phil, 2013; Brooke, 2014; Veksler, 2016; Caldwell, 2018; Penzo, n.d.). This is a healthy habit with sole purpose the provision of a forecast of revenues and expenditures, and the assessment of the actual financial operation against the forecast.

The advent and rapid evolution of technology has been beneficial to the mankind making life easier. According to Funk, Kennedy& Podrebarac-Sciupac (2016) the positive effects of technology on society can be observed in social sector, educational sector, enterprise sector, industrial sector, health sector, and many more. Funk, Kennedy& Podrebarac-Sciupac also stated that “the possibilities for human enhancement stem from new scientific and technological innovations”, since technology increases knowledge and understanding and promotes sharing of information. Nowadays, there is an application available and easily accessible pretty much for everything. Web and mobile applications comprise various different categories encompassing among others Entertainment, Games, Lifestyle, Food and Drink, Education, Sports, Health and Fitness, Finance, Behavioral Change. At present, there is a vast amount of applications that fall in the finance category and promote self-monitoring and self-regulation of budget and expenses. Such applications serve people in managing their lifestyle by managing their finances accordingly.

The aim of this thesis was to develop a system that will help users track their personal expenses and stay on budget. But what is the added value of yet another app for tracking expenses? The motivation of this endeavor was to develop an affective system with state-of-the-art web technologies, that will contribute in affective computing and behavioral sciences. The term affect comprises emotions, feelings, and moods and is a fundamental aspect of human behavior and user experience, since it influences reflex, perception, cognition, and behavior (Norman, 2002; Russell, 2003). Affective systems exhibit human-like behavior and increase engagement through affective feedback techniques, such as emoticons, images, sounds, videos, and avatars (Muntean, 2011; Sallnäs & Sundblad, 2012; Moridis & Economides, 2012b; Wang, Zhao, Qiu & Zhu, 2014; Henderson & Phillips, 2015; Kanai, 2015; Hall, Tracy & Lamey, 2016; Kaye, Wall & Malone, 2016; Tolins & Samermit, 2016; Miltner & Highfield, 2017). The goal of this thesis was to study affective feedback techniques, and identify how the implementation of animated GIFs, as a mean to provide affective feedback, can influence and shape users’ behavior regarding their financial attitudes and performance.

## 1.1 Thesis Outline

Chapter 1 presents the objective, scope, motivation and goal of this research, providing a clear and complete overview of the whole thesis.

Chapter 2 contains the theoretical and conceptual framework related to feedback and affective feedback. This chapter begins with the definition of feedback and its types, and continues with the description regarding its core concepts of positive, negative, effective, and constructive feedback. The last part dives into existing research on affective feedback and the techniques that enable it and model users' behaviors.

In chapter 3, the purpose and the requirements of the developed system is discussed. Moreover, the affective technique that was selected for implementation and study is presented. Finally, in this chapter it is illustrated how feedback was served to users.

Chapter 4 is concerned with the design and architecture of the system. This chapter focuses on how the requirements of the system were modeled in order to proceed with its successful implementation. The first part is concerned with the design of the database and the back end development of the system. The second part concentrates on user characteristics and user interfaces and provides the use cases of the system in a pictorial way.

In chapter 5, the implementation of the system is discussed. This chapter emphasizes on pioneer web practices and technologies, and underlines the orientation of the system as a state-of-the-art progressive web application. Additionally, the most significant tools, libraries and plugins that were utilized are presented.

Chapter 6 is concerned with the evaluation of the prototype system based on descriptive and inferential statistical analysis that was conducted. Additionally, in this chapter it is described to what extent animated GIFs, as a mean to deliver affective feedback, influence budget adherence, if any. This chapter also explains the characteristics of the participants, and the data collection and data analysis procedures. Moreover, the results of the statistical analysis are presented followed by the summary of the key findings.

Finally, in Chapter 7, the contribution and the conclusions are summarized. Last but not least, limitations are discussed, and related future work is proposed.

# 2 Theoretical and Conceptual Framework

This section deals with the concepts that are the central focus of this research. Firstly, it is discussed how the definition of the term “*feedback*” has been shaped and interpreted over the years and in different contexts. Secondly, the key different types of feedback are presented, followed by the notions of positive, negative, effective, and constructive feedback. Then the key aspect of this research, “*affective feedback*”, is discussed in detail, focusing on the various techniques that were employed to implement and convey affective feedback.

## 2.1 What is Feedback

To begin with, feedback has been, and still is, the focal point of many research studies from various scientific fields, with the initial researches dating way back. Studies and researches on feedback do not only focus on its meaning but also on its types, its formats, its derivations and limitations, as well as the ways to efficiently convey it. Currently, there is a plethora of definitions of feedback based on the general context that it has been studied. According to Merkel (1973) feedback is “*the interchange of information on the part of human beings in a communication or problem-solving situation*”. Page, Thomas and Marshall (1978) defined feedback as “*information on progress of teaching and learning provided through various methods of assessment*”. Meyer (1995) claimed that feedback is “*information provided to the learner concerning the correctness, appropriateness or accuracy. In short feedback is information about a learner’s performance*”. In the dictionary of psychology Corsini (1999) describes feedback as “*a direct response by an individual or group to another person’s behavior, such as the reactions of an audience to a speaker’s remarks*”. As stated in the [businessdictionary.com](http://businessdictionary.com), feedback is the “*process in which the effect or output of an action is 'returned' (fed-back) to modify the next action... As a two-way flow, feedback is inherent to all interactions, whether human-to-human, human-to-machine, or machine-to-machine*”. A more specialized term regarding feedback in electronic

devices, circuits or mechanical systems was given by *technopedia.com*, according to which feedback is an event where the return of the output, either as a whole or part of it, is used as input back into the system itself. This consequently modifies the variables of the system and results in a different output and in a different feedback. The aforementioned continuing chain of cause and effect is used either to oppose (negative feedback) or to aid (positive feedback) the input or an action. In both cases, it assists in producing the desired output, adjusting and driving further a system, achieving thus the preferable performance.

Van De Ridder, Stokking, McGaghie & Ten Cate (2008) conducted a research in order to propose a consensual and operational definition of feedback in clinical education. Based on existing literature they found out that there are three dominant concepts regarding feedback defining it as information, as reaction including information and as a cycle, encompassing both information and reaction. Comparing numerous definitions, feedback features and limitations from various sources they defined feedback in clinical education as *“specific information about the comparison between a trainee’s observed performance and a standard, given with the intent to improve the trainee’s performance”*. Another definition of feedback was given by Hamid and Mahmood (2010) in their research regarding constructive feedback in a clinical-educational context. They defined feedback as *“a process which involves a two-way, non-judgmental communication with the purpose providing information about quality of work to enhance one’s ability”* and stated that the two key objectives of feedback are the appreciation of the good/right things with logical explanations and the identification of the bad/wrong things, including the provision of options to change them.

Based on above definitions, it is safe to assume that feedback is, in general, any reaction or response to a specific process, behavior or activity that brings about a change in the recipient’s behavior/performance.

## **2.2 Types of Feedback**

Pursuant to the existing theoretical framework, feedback can be categorized in numerous ways since researchers have not yet agreed on a universal classification. On an intentionality basis, feedback can be grouped in unintentional and intentional. Unintentional feedback is the aftereffect of incidental natural interactions with the social

and physical environment. In educational environments, unintentional feedback often occurs in the form of unguided simulations or unstructured peer interactions and can be a powerful incentive for learning and self-development. On the contrary, intentional feedback occurs mostly in instructional settings and is a deliberate action designed to make students aware of the quality, correctness, and general appropriateness of their performance (Bangert-Drowns, Kulik, & Morgan, 1991).

Another axis that can be used to categorize feedback is the way in which it is provided. Direct feedback is delivered in the act of interpersonal communication among people, whilst, indirect, or mediated, feedback is delivered through a range of artifacts, with computers being the most prominent mean. Kluger and Adler (1993) studied the effects of feedback provided by a person versus that provided by a computer on performance, motivation, and feedback seeking. Key evidence indicated that people are more prone to seeking feedback from a computer rather than from another person, whilst direct feedback can lower performance. Both direct and mediated feedback can be further discriminated on the vectors of load and type of information in regard to their content. Load refers to the amount of information delivered through feedback and can be from a single character (letter or number) representing a grade to a thorough narrative report of students' performance (Moreno, 2004; Kulhavy & Stock, 1989).

Based on the type of information, feedback can be distinguished into outcome or process related feedback. Outcome related feedback, alternatively evaluative or summative feedback, provides information about the correctness of responses or about the level of performance on reaching an end point, such as assessment, at the conclusion of the module. Evaluative feedback represents a judgment that often carries a connotation of social comparison that can be conveyed in the form of grades, percentile scores, number of solved items, brief general comments, etc. The disadvantage of this type of feedback is that the information conveyed do not offer the necessary indications and guidance required for improvement. On the other hand, process related feedback, also referred in the literature as descriptive or formative, is significantly more effective than the previously discussed type. Formative feedback provides information about how one performs the task (not necessarily how well) and details possible ways to overcome difficulties, develop the work on progress and improve performance while there is still time. This type of feedback has three key aspects: it is linked to the expected outcome (Where am I going?), addresses faulty interpretations and lack of understanding (How

am I going?), provides clarified and manageable “*next steps*”, based on an assessment of the work at hand, as well as an example of what “*good work looks like*” allowing recipients to take on the responsibility of self-assessing and self-correcting (What do I need to do to improve and how do I do it?). Formative feedback is usually conducted regularly in the classrooms by various forms, encompassing written, oral, formal and informal feedback. Teachers consider it as a great strategy to engage students and to constantly reflect on how they can approach, orient, and evaluate learning. Thus, the learning process becomes motivational, informative and corrective and leads successfully to the desired outcomes (Issa, 2016; Earl, 2012; Buczynski, 2009; Shute, 2008; Hattie & Timperley, 2007; Linn & Miller, 2005).

More classifications regarding the feedback types were proposed over the years. Bangert-Drowns et al. (1991) suggested the discrimination of feedback into error correction, presentation of prototypic responses, display of the consequences of responses, and explanation of the appropriateness of responses. Another, more complex, alternative was recommended by Tunstall and Gipps (1996) who dichotomized feedback into two broad categories: feedback as socialization and feedback as assessment. The aforementioned categories were further classified depending on the specific function that a feedback message served. Such functions included but were not limited to, rewarding/punishing, approving/disapproving, specifying improvements, constructing achievement, and constructing the way forward. On the other hand, Narciss (2008) argued that feedback can be either “*knowledge of performance*” or “*knowledge of result*”, illustrating the percentage of correctly solved tasks and indicating correct or incorrect answers respectively. Narciss (2008) also discussed that feedback, in other cases, encompasses, either separately or blended, explanations for error correction and detailed information strategically effective for task completion.

According to Tsutsui (2004) feedback can be categorized based on the time of intervention into interactive feedback, intrusive feedback, and delayed feedback. Interactive feedback is delivered during the performance via student-teacher interaction. The essential advantage of this type of feedback is that any problems which are caused by interrupting the students’ performance or by giving feedback at a later point in time (after an activity is completed) can be avoided, since it can be conveyed along the natural flow of oral activities. Nonetheless, interactive feedback is not free of disadvantages. First of all, this type of feedback is limited to those activities in which

instructor-student interaction constitutes an integrated part. Additionally, interactive feedback arises the “*conflict of interests*” problem between individual students and the entire class. Because of students’ diversity, in terms of capabilities, strengths, needs, difficulties, etc., interactive feedback on a student’s utterance can be beneficial for one student but not for the whole class at the same time. Moreover, Tsutsui (2004) cautions that interactive feedback can be inconsistent because it is given on the spot with no preparation, and ambiguous since in some situations students may not perceive it. As far as intrusive feedback is concerned, this type of feedback is conveyed during the student’s performance by interrupting it. Intrusive feedback is applicable in more cases than interactive feedback, is direct and allows student correction timely and immediately. Similar to interactive feedback, intrusive feedback also comes along with issues of inconsistency and conflict of interests. However, its fundamental disadvantage is that students’ interruption obstructs the flow of the performance and negatively affects the confidence and the mood of the class by redirecting the latter in a context where rules and forms are the prevailed features. All problems that are surfaced in both interactive and intrusive feedback can be prevented with delayed feedback which is delivered after the performance by providing comprehensive comments on the whole performance. The disadvantages of this type of feedback are inseparably related to the time. Instructors do not have the required time to keep accurate and detailed notes on students’ performance and responses, whilst they may confuse one student’s performance with another’s when trying to complete their notes at a later point. From the students’ point of view, connecting comments to the performance can be problematic since students may not remember the utterance on which the instructor comments or may confuse utterance-feedback pairs when there are many. Lastly, in delayed feedback there are no chances for reinforcement unless the instructor gives students that opportunity explicitly.

A completely different approach was followed by Hattie and Timperley (2007) who developed a model, focusing on formative feedback, that differentiated feedback into four levels: the task level, the process level, the self-regulation level, and the self-level. To begin with, the task level focuses on how well a task was performed, with corrective feedback, references and other aspects regarding task accomplishment being the most prominent types of feedback. It is noticeable the fact that feedback comments at this level do not necessarily generalize to other tasks. The process level involves feedback about the processes used in the completion of tasks. Focusing on students’ strategies for



error detection and on assisting learners create meaning and relate to the connections between concepts, feedback at this level enhances deeper understanding and can be more effective than feedback at the task level. The self-regulation level, as implicated by its name, aims in promoting students' self-monitoring, directing, and regulating of actions. Self-regulation feedback is conveyed through posing questions rather than providing information. The effectiveness of the feedback at this level is mediated, among other, by six key factors: the capacity to create "*internal*" feedback; the ability to self-assess; the willingness to invest effort into seeking and dealing with feedback information; the degree of confidence or certainty in the correctness of the response; the attributions about success or failure; the level of proficiency at seeking help. Finally, the self-level is person centered and includes personal evaluations and affective reactions directed at personal attributes, such as understanding, intelligence and ability. Usually delivered as praise, feedback at this level is the least effective in this model. Although students highly value praise, since it can boost their ego and self-esteem, it does not translate into more engagement (including or not commitment to learning goals), does not promote self-efficacy, nor leads to a greater understanding about learning tasks. Yet, if person-focused feedback is employed appropriately, both regarding frequency and content of comments, then strong trust bonds between a learner and a supervising professional can be built.

## **2.3 Positive and Negative Feedback**

According to Losada (1999), positive feedback is defined as feedback that shows support, encouragement, or appreciation, whilst negative feedback as feedback that shows disapproval, or even sarcasm. It is indisputable affirmed that both negative and positive feedback have aftereffects, however, they do not have equal impact on learning and skill acquisition processes nor influence the same way performance and behavior. When feedback is by and large positive, people tend to let their defenses down, accept it easier and act on it in a constructive way. On the other hand, negative feedback evokes mostly defensive attitudes, extending from evasion in elaborating on the feedback to negative thoughts. Nonetheless, evidence showed that negative feedback can be effective too, whilst blending both positive and negative feedback seems to be the most efficient practice. The overuse of negative feedback is highly discouraged by researchers, although it is highly encouraged in the form of brief occurrences in a

greater context of positivity, since small bits of negative feedback can infiltrate extraordinarily well. (Van Beuningen, De Jong and Kuiken, 2012; Van Beuningen, 2011; Shute, 2008; Hattie & Timperley, 2007; Fredrickson & Losada 2005; Goodman, Wood & Hendrickx 2004; Losada & Heaphy 2004; Baumeister, Bratslavsky, Finkenauer & Vohs 2001).

## **2.4 Effective and Constructive Feedback**

Feedback is essential in any kind of development and learning process, since it provides both crucial information regarding the current progress and opportunities for future improvement. No matter what the type of feedback is it has to be effective and constructive. Bee & Bee (1998) supported that feedback can be either positive (reinforcing good performance and behaviors) or negative (improving poor performance and behaviors), but in any case, it can and must be constructive. But what are the principles for effective and constructive feedback?

According to Taras (2002) the three general conditions for effective feedback are: a) knowledge of appropriate standards by all parties involved, b) comparison of one's work with these standards, and c) taking appropriate action to close the gap between the two. Juwah et al. (2004) drawn from their review of existing research literature seven principles of good feedback practice in an educational context. As stated by Juwah et al. (2004) good and effective feedback:

1. Facilitates the development of self-assessment (reflection) in learning.
2. Encourages teacher and peer dialogue around learning.
3. Helps in clarifying what good performance is (goals, criteria, standards expected).
4. Provides opportunities to close the gap between current and desired performance.
5. Delivers high quality information to students about their learning.
6. Encourages positive motivational beliefs and self-esteem.
7. Provides information to teachers that can be used to help shape the teaching.

Archer (2010) stated that in order to achieve truly effective feedback, an integrated approach that supports a feedback culture must be developed. A culture where feedback is conceptualized as a supported sequential process, rather than a series of unrelated events, and builds upon self-monitoring while nurtures recipient reflection-in-action at the same time.

Hamid & Mahmood (2010) after a thorough investigation on the principles and guidelines of constructive feedback concluded that there are fourteen standards that define good, efficient and constructive feedback. These standards focus on medical students, can be generalized, and indicate that feedback should be:

- 1) well timed and expected, i.e. as early as possible and agreed between participants for their common goals;
- 2) based on first hand data without any intermediate source and through direct observation;
- 3) confidential in order to maintain trust and respect;
- 4) quantity regulated, i.e. reasonable amount of information;
- 5) balanced in terms of appreciation for good things and suggestions for improvement;
- 6) clear as far as goals, criterion and standards are concerned;
- 7) encouraging for time, effort, positive believes i.e. encouragement for whatever is right or good, interaction and dialogues with peer and teacher;
- 8) helpful in improving teaching and for achieving common academic goals;
- 9) opportunistic, i.e. providing opportunities for raising current performance to meet standard performance;
- 10) purposeful, e.g. to plan a strategy, to improve results, to clarify standards, etc.;
- 11) relevant and tailored according to needs and interest of an individual;
- 12) factual, i.e. based on actual performance rather than assumptions or interpretations;
- 13) descriptive rather than evaluative, and
- 14) specific, focusing the observed and changeable behavior.

Henderson & Phillips (2015) asserted that constructive feedback must be timely, unambiguous, educative rather than evaluative, proportionate to criteria and goals, phrased as an ongoing dialogue rather than an endpoint and emphasizing on task performance, while instructors have to be sensitive to the addressed individual's feelings. Sultan & Khan (2017) in their review regarding feedback in clinical-educational settings highlighted the essential components of constructive feedback. Pursuant to their article, constructive feedback requires the establishment of an appropriate interpersonal climate and is well planned, timed, expected and agreed

upon a common goal. Additionally, it is descriptive rather than evaluative and based on direct observation of performance. Moreover, constructive feedback is confidential and delivers accurate and reasonable amount of information (i.e. is regulated in quantity). Furthermore, it deals with decisions, actions and specific performance rather than assumed intentions and generalized behaviors and is limited to those behaviors that are remediable.

## **2.5 Affective Feedback**

Picard in 1997 defined affective computing as “*computing that relates to, arises from or deliberately influences emotions*”. Through affect recognition computer systems can nowadays detect and recognize users’ affective states as humans do in face-to-face interaction. However, according to Picard (2003) one of the major challenges in affective computing is the improvement of the accuracy of recognizing people’s emotions, a view that has been supported by several researchers in the field of human computer interaction (Oviatt, 2003; Pantic & Rothkrantz, 2003).

Pantic & Rothkrantz (2003) additionally claimed that the human sensory system is able to recognize another party’s affective states using a multimodal analysis of multiple communication channels. Humans are capable of detecting recognizing and interpreting, apart from verbal signals, non-verbal communicative signals. Facial expressions, gesture and posture of body and vocal intonations are such signals which are easily detected with little or no effort, whilst psycho-physiological correlates of emotion such as pulse, respiration rate or skin temperature demand much more effort.

As far as computer systems are concerned, in order to improve the validity of the estimations regarding users’ emotions and imitate human to human interaction evidence of many modes of interaction should be combined throughout emotion detection technologies. By achieving accurate affect recognition, computer systems are enabled to respond appropriately to user’s affective state, rather than simply respond to user’s commands, and provide affective feedback.

Moreover, according to Pantic & Rothkrantz (2003) computer systems that are able to recognize users’ affective states and provide feedback accordingly tend to be more human-like, more effective, and more efficient. Modern applications, either web-based or mobile-based, adhere to user experience and design principles so as to engage users and provide optimum user experience. With the advent of affective feedback techniques

and as research on this field evolves, more and more applications utilize affective feedback. Implementation of this feature seems to be more common nowadays in e-learning platforms, education-oriented applications, videogames and sensing applications. Sensing applications in particular are a dominant element in every mobile device, since the latter encompass by definition a wide variety of sensors (e.g. accelerometer, digital compass, gyroscope, GPS, microphone, and camera) and sensing algorithms, applications, and systems easing that way affect recognition (Lane, Miluzzo, Peebles, Choudhury and Campbell, 2010).

Concluding, affective feedback is driven by affective states and emotions play a key role. McDarby, Condrón, Hughes and Augenblick (2004) stated that *“Affective feedback is the process of using technology to help people achieve and maintain specific internal states. Essentially, we are trying to create immersive systems that encourage people to reach a specific state, such as relaxation or concentration, and 'teach' them how to control it.”*

According to Moridis and Economides (2008a, 2008b, 2009) and based on their research regarding affective and emotional feedback on learning and training activities, affective feedback can be applied by using constructively positive emotions, while preventing, controlling and managing negative emotions. Furthermore, they also argued that emotional feedback can be also implemented utilizing negative emotions which can also result in increasing users' engagement. Moridis and Economides also suggested that affective feedback should correspond to user's personalized needs and should be integrated into computer systems through suitable emotional strategies.

## **2.6 Affective Feedback Techniques**

Currently, there is a variety of affective feedback techniques. Each technique is unique, in the way it conveys information and affects individuals, and comes along with both advantages and disadvantages. Furthermore, affective feedback techniques can be applied, either individually or combined, in various settings and are directed toward achieving a specific purpose. In the following section, the most prominent and widely used means to provide affective feedback are presented. Additionally, the role and the way that these means (i.e. textual information, colors, avatars and animated agents, emoticons, still and animated images, sounds, and videos) affect information delivery and users' emotions is discussed.

### **2.6.1 Textual Information, Colors and Avatars**

An empirical review regarding interaction between users and animated agents was conducted by Dehn and Van Mulken (2000), who studied avatars as an affective feedback method in comparison to textual information. They concluded that the use of avatars is a great feedback technique to communicate more delicate information through eye contact and/or gestures, whilst textual information as a mean to provide feedback is more direct. Summarizing their findings, they highlighted that a greater level of feedback could be provided to users through multimodal interaction.

When it comes to feedback provision, the effectiveness of a multimodal approach was advocated by McDarby et al. (2004). They presented the games “*Relax to win*” and “*Brainchild*” which were both designed to provide motivation, with the first one focusing on teaching users how to relax, whilst the latter one focused on helping users relax in a stress-inducing situation. The key findings were that the combination of sounds, colors, dialogues and animations blended with game elements and storytelling engenders the desired outcome. This point of view was also supported by Kelley et al. (2012) in their research regarding the effect of strength meters on password creation. They noted that textual information containing specific words as a feedback technique is capable of altering users’ state and behavior. The same effect can be evoked through the use of colors, with red indicating a negative outcome and green or blue a positive occurrence. From their research, they inferred that affective feedback is better achieved when combining colors, textual information and visual elements such as progress bars.

Avatars as a mean to provide affective feedback were mostly studied on educational oriented environments and proved to be mostly beneficial. Hall, Woods, Aylett, Newall and Paiva (2005) conducted a research regarding affective interactions with synthetic characters in virtual learning environments. By deploying avatars in a personal social and health education environment this research aimed in achieving empathic engagement while educating children about bullying. Results showed that the avatars not only provoked the desired empathetic effect on children but also indicated that a similar result could be achieved in adults if the same type of feedback was applied. The use of avatars as an affective feedback technique was reinforced by Robinson, McQuiggan and Lester (2009) throughout their research regarding affective feedback in intelligent tutoring systems. In their experimental research, support to users was provided by utilizing avatars which were able to decide whether or not to intervene

while a user was working. The outcome of this research highlighted that the time of the agent's intervention plays an important role in user's experience and engagement since evidence showed that the intervention of the agent at the wrong time has a negative impact on users.

The delivery of affective feedback through avatars was also advocated by Moridis and Economides (2012a) in their research regarding embodied conversational agents (ECAs). This research focused on examining the impact of ECAs' emotional, facial and tone of voice expressions combined with empathetic verbal behavior when displayed as feedback to user's fear, sad, and happy emotions in the context of a self-assessment test. In this research three identical female agents that were performing parallel empathy were tested. The first agent combined parallel empathy with neutral emotional expressions, the second agent was performing parallel empathy displaying emotional expressions that were relevant to the emotional state of the user, whilst the third one was performing parallel empathy by displaying relevant emotional expressions followed by emotional expressions of reactive empathy with the goal of altering the student's emotional state. Key findings of this research indicated that the second agent caused the persistence of an emotion whilst the third one succeeded in changing an emotional state of fear to a neutral one.

## **2.6.2 Emoticons and emojis**

It is an indisputable fact that the internet and the advent of communication applications changed the way we communicate. Having any kind of portable device, such as smartphone, tablet, or laptop, and access to internet is just enough to allow us to communicate with other people directly and instantly. Over the past years, a significant growth of online communication and interaction was observed. As Dresner & Herring (2010), Kaye, Malone, and Wall (2017), and Walther and D'Addario (2001) highlighted, this rapid increase of interactions among people emerged the need to augment electronic communication by enhancing non-verbal communication. That is because when it comes to text messages or emails, digital communication lacks in nuanced meaning since body language and tone of voice cannot be conveyed. However, the mankind being highly resourceful managed to enhance written communication with the addition of two new-age hieroglyphic languages: emoticons and emojis, that are capable of communicating extra-linguistic information to users.

Starting with the predecessor of the two, an emoticon is a typographic display, i.e. a sequence of a set of characters (punctuation marks, letters, and numbers) used to create pictorial icons that imitate a facial expression and can convey an emotion or a sentiment. Indeed, the words emotion and icon are the roots of the portmanteau word “*emoticon*”. The emoticons owe their genesis in a joke posted online on a message board on 1982 at Carnegie Mellon University. The joke actually went wrong and because of the confusion it created, the computer scientist Dr. Scott E. Fahlman suggested that messages should be marked with the characters :- ) and :-( in order to distinguish jokes from serious statements respectively. Unlike emoticons, emojis are pictographs of faces, objects, and symbols, or, even simpler, emojis are small pictures of anything we can imagine, that can be embedded into text. The word “*emoji*” is a contraction of the Japanese words “*e*” (i.e. picture) and “*moji*” (i.e. character). The emojis were created in 1999 by the Japanese communications company NTT DoCoMo and originally were addressed to Japanese customers only. Now, emoji are used worldwide and can be translated across different platforms. However, their translation is not unified, meaning that the same emoji can be displayed differently in an iPhone from a Samsung Galaxy smartphone (Khalid, 2017; Hern, 2015; Grannan, n.d.).

From all the above, it is clarified that emoticons and emojis are two completely different things. From now on and in the context of this research with the term emoticons we will refer to both emoticons and emojis that display facial expressions only. After all, most of the times we try to type an emoticon in its symbolic form it usually renders to its graphical one.

Research regarding emoticons was mostly carried out in communication and education-oriented environments. Wang, Zhao, Qiu & Zhu (2014) studied the effects of emoticons on the acceptance of negative feedback in computer-mediated communication focusing on two types of emoticons, liking and disliking ones. Evidence affirmed that liking emoticons increased perceived good intention of the feedback provider and decreased perceived feedback negativity when the feedback is specific, but they did not have a significant effect for unspecific feedback. On the other hand, the use of disliking emoticons decreased perceived good intention of the feedback provider and increased perceived feedback negativity when the feedback is unspecific, but they did not have a significant effect for specific feedback. The key finding of this research was



that the acceptance of negative feedback is highly affected by the perceived good intention of the feedback provider and the perceived feedback negativity.

Kaye, Wall, and Malone (2016) studied the use of emoticons on different virtual platforms and concluded that emoticons boost communication and can serve both personal and interpersonal functions of users with high level of consistency across platforms. The basic reasons of emoticon usage revealed to be their ability to facilitate and ease personal expression (encompassing their use to establish emotional tone and lighten the mood), reduce ambiguity of discourse and enhance appropriateness of context. Wall, Kaye, and Malone (2016) conducted an exploration research of psychological factors on emoticon usage and implications for judgement accuracy and found out that emoticon use on Facebook and in online chat was related to perceptions of the emoticon user's personality when profiles were independently rated for personality. Specifically, the user's agreeableness, conscientiousness, and openness were not only positively but also significantly interrelated with the use of happy emoticons.

Building upon previous research regarding the use of emoticons on social media and online communication Grieve, Moffitt, & Padgett (2018) studied the role of emoticons in an educational context. Grieve et al. (2018) focused in studying whether and to what extent the use of emoticons in assignment feedback presented online (in e-learning environments) influences perceptions of the educator's personality and intelligence. By carrying out an experimental paradigm they concluded that educators were considered to be more agreeable, extraverted and open when they used emoticons in assignment feedback.

As discussed above, emoticons have both positive and negative effect on people, and can be used as an affective feedback technique. However, emoticons are open to interpretations. Evidence showed that there is a variance regarding the interpretation of emoticons not only because of the human factor but also because of technology. As already mentioned, different platforms and software do not render emoticons in the same way. The sentiment that emoticons evoke is not perceived in the same way by all individuals since their cultural background plays a key role in expressing and perceiving facial expressions. Of course, there are more factors affecting the perceived emotions by such interpretations. Factors that derive from each person separately and include among others the individual's EQ (emotional quotient or emotional intelligence or emotional

intelligence quotient), mental state, and current mood. Because of all the aforementioned, emoticons can be misinterpreted, leading to miscommunication and even to disputes (Miller, Thebault-Spieker, Chang, Johnson, Terveen and Hecht, 2016; Park, Barash, Fink and Cha 2013).

### **2.6.3 Images and GIFs**

Images are widely used in mobile and web applications since they constitute a means to communicate information and convey messages smoothly. Indeed, many applications are based on sharing images and photographs. A great example is Instagram, where users create and “consume” content. Utilizing photographs, images, animated images and often videos, users are able to share emotions and moods. Obviously, images, photographs and any kind of visual representation, such as infographics, are an inseparable part of the web as we know it today.

Burri (2012) conducted a research regarding the sociology of images and outlined a framework that aids in exploring how visuals are used within public communication, how they are produced and how they are read and interpreted by different audiences. More specifically, this article points out that the social context within images are created and consumed is a key aspect in understanding images as meaningful. As a consequence, images become communication images and have various effects. In order to describe the intersection between social practices and images Burri introduced the concept of “*visual logics*” mentioning that a visual logic is composed by three dimensions: the visual value, the visual performance and the image persuasiveness. The visual value refers to the characteristics of images that make them stand out over discursive communication. A glance is often enough for someone to gain a sense of the entire image allowing in that way a simultaneous perception of visual information. The dimension of the visual performance refers to the ways that images, encompassing natural images like photographs, are composed, framed and interpreted. In short, this dimension refers to the visual signs that are represented and conveyed through images. Finally, the image persuasiveness highlights the importance of visual information in communication and the persuasive power of images, pointing out that images are not only used to make and reinforce specific arguments but also to convince audiences of particular things.

Previous work (Machajdik and Hanbury, 2010; Datta, Li & Wang, 2008; Yanulevskaya et al., 2008; Datta et al., 2006) has proven that emotions are aroused and affected by images and that different types of visual features are associated with the emotional contents of images. Features representing the color, texture, composition and content of images were extracted by Machajdik et al. (2010) whilst Wang, Jia, Yin, & Cai (2013) focused on figure-ground relationship, color pattern, shape and composition, visual features that affect human emotional perception directly. Even abstract images play a key role in provoking and affecting emotions. Bartoszek and Cervone (2016) studied how abstract images could evoke and reveal emotional states. Through a series of experiments where emotions of sadness, anger and fear were evoked, elevated levels of the target emotion were observed while there were no indications of non-target negative emotions.

From all the above, we can infer that images significantly affect audiences and can be used as a mean to provide affective feedback since they combine colors, shapes, texture and content, even in an abstract form, in order to deliver information and enhance messages. In this section we dive in the use of a specialized type of images, the Graphics Interchange Format images or GIFs in short.

Animated GIFs have been around for more than two decades and constitute an exceptional media form between videos and still images, since they can add visual content and movement to a website or application even when technological and bandwidth limitations exist (Suhr, 2014). Having numerous unique characteristics such as briefness, looping, silence and emotional expressiveness, and due to their high compatibility and portability animated GIFs established their presence and use on the web. Currently, animated GIFs are massively found on social media, message boards, digital forums, instant messaging applications, and generally in any kind of websites, and applications, both web and mobile, of every genre.

According to Bourlai, & Herring (2014) animated GIFs as a form of image-based communication, conveys more emotion and greater intensity of emotion than text-based communication with positive emotions being the most predominant among the expressed ones. As underlined by The New York Times (Isaac, 2015), animated GIFs are now *“a way to relay complex feelings and thoughts in ways beyond words and even photographs”*. In the same vein, Kanai (2015) pointed out that animated GIFs allow users to respond to and portray lived experiences via affective reactions to ordinary

situations with loops from popular television, film, and other media sources. Tolins & Samermit (2016) studied GIFs as embodied enactments in text-mediated conversations and highlighted that animated GIFs are used as co-speech gestures that enhance communication and can evoke or/and convey affective responses. Key findings in their research also indicated that animated GIFs are a powerful means for visually depicting and enacting emotions, actions and affective states. Except from being effective and affective, animated GIFs have also proven to be highly engaging, especially in social media (Bakhshi et al., 2016). Animated GIFs that capture facial expressions, body language and gestures of humans increase engagement and induce higher motion energy (Bakhshi et al., 2016; Tolins & Samermit, 2016). Through such GIFs users are enabled to express specific behaviors and actions by mirroring and projecting their own emotions and affective states. However, GIFs are not restrained in facilitating the performance of affect at a specific point in time. They can also augment and model affective performances by shaping and manipulating users' capacities on an affective level without being limited to the encapsulated moment (Ash, 2015; Miltner & Highfield, 2017). After all, the act of selecting and using a GIF image is a performance in and of itself, with a specific meaning.

Apparently, animated GIFs play an increasingly important role in conveying information and messages, delivering news, telling stories (through photo-journalism) and expressing emotions. In educational settings, an efficient series of GIFs can function as a storyboarding technique reinforcing tutorials by allowing users to view and review steps as needed (Suhr, 2014). Individuals process and “consume” animated GIFs, however the meanings of the latter are created within the context of a community (Newman, 2016; Eppink, 2014). As Newman (2016) stated “*GIFs are examples of vernacular creativity among groups of users with shared interests and reference points*”. Like other forms of communication, GIFs can be used to establish in-group and out-group boundaries and different groups may utilize any GIF with their own conventions and meanings (Newman, 2016; Eppink, 2014).

Even though GIFs are community-oriented their interpretation is neither easy nor universal. As a polysemic and intertextual form, animated GIFs act as quotation or reference inducing individual commentary and/or reaction (Tolins & Samermit, 2016) while promoting the demonstration of cultural knowledge (Miltner & Highfield, 2017). According to Sha (2016) GIFs are “*a visual language unto themselves, an emotive*

*vocabulary made out of culture*". Similar to emoticons, GIFs may be misinterpreted and can often lead to miscommunication (Jiang, Brubaker & Fiesler, 2017; Bourlai & Herring, 2014). Positive GIFs tend to have more diverse interpretations than negative ones, whilst embedded text only serves to reinforce the existing visual content (Jiang, Brubaker & Fiesler, 2017). Concluding, the meaning of a GIF is extracted based on the content of the GIF itself (multiple and repetitive frames of images) as well as the surrounding determinants (text captions, messages, and "likes") which provide additional layers for interpretation, while it highly depends on who is using it and in what context (Miltner & Highfield, 2017; Jiang, Brubaker & Fiesler, 2017; Bourlai, & Herring, 2014). Other factors that also affect GIF interpretation are the length and the duration of GIFs, the load of information they carry and deliver, and of course the human factor in terms of linguistic abilities, emotional state and sentiment polarity (Jiang, Brubaker & Fiesler, 2017).

#### **2.6.4 Audio and Sounds**

Sound cues are usually employed to increase accessibility in applications and systems making them available to users with visual impairments. Additionally, audible stimuli have significant effects on users' emotions and affective states and the provoked impressions are highly influenced by the context in which audible feedback is applied (Seebode, Schleicher & Möller, 2012).

Moll, Huang, & Sallnäs (2010) studied the impact of sound cues in haptic collaborative virtual environments. They used a haptic 3D environment where participants were asked to build composed objects out of building blocks. A haptic feedback pointing device was utilized to help users pick up and move around the building blocks. Participants worked in pairs (one sighted and one blindfolded) and audio feedback experienced by half of the teams. In the context of their research, Moll et al. implemented a set of event oriented sound cues: a grip sound that was heard every time one lifted an object, a kind of touch down sound, which was heard every time an object touched the floor, a collision sound, that was heard every time an object landed on top of another, and a contact sound which was used to help the blindfolded participant locate the position of his/her teammate relatively to his/her own position. Results demonstrated that blindfolded participants, who were receiving audio feedback, were able to feel space details such as height, position and direction understanding thus better the workspace. Moreover, they were able to get feedback and guidance on their

own without addressing questions to the sighted teammate regarding the work on progress and the changes in the workflow. On the other hand, blindfolded participants with access to the sound cues experienced increased levels of anxiety when long intervals with no audio feedback occurred (in cases where the other participant did not interact with the system for a while). Summarizing, Moll et al. pointed out that audio feedback increased efficiency in the use of the app and made a difference in the interaction between the collaborators.

These results were advocated by Huang, Moll, Sallnäs & Sundblad (2012) in their experimental study regarding auditory feedback in haptic collaborative interfaces. Building upon the previously mentioned research, they highlighted that adding audio cues increased the efficiency of the collaboration while performance time decreased essentially. Furthermore, they underlined throughout a qualitative analysis that the combination of haptic and audio feedback was used in numerous ways in the participants' grounding process and in supporting the group members' action awareness.

Merry & Orsmond (2008) and Lunt & Curran (2010) studied the advantages of electronic audio feedback (via audio files) in educational environments. Both researches advocated in favor of the use of audio feedback since it proved to be efficient and effective. This type of feedback was perceived by students as being of good quality, easier to understand, more personal and having more depth. Students enjoyed audio feedback and responded in a positive way, since they were able to utilize it better (annotating their work while listening). A more specialized research was carried out by Moridis & Economides (2012b), who explored the effect of the applause sound as an achievement-based reward during a computerized self-assessment test. They concluded that providing auditory feedback via applause after a correct answer is indeed an affective feedback technique. This kind of affective feedback contributes to the personalization of the learning experience and correlates to the levels of anxiety experienced before and after a self-assessment test. They highlighted that the effect of the applause was not equal for both genders (higher-state of anxiety observed in males who were not receiving an applause rather than in females who were not receiving an applause or males who were receiving an applause) and that affective systems must take under consideration gender-centered approaches.

Auditory feedback was studied in other contexts and environments too and as an added element to other types of feedback. Maculewicz, Erkut, & Serafin (2016) investigated the impact of auditory and haptic feedback on rhythmic walking interactions. Auditory and haptic feedback signals were either ecological physically-based synthetic walking signals or simple sinusoidal beeps. Results indicated that participants were able to synchronize equally well with the tempo with either audio or haptic cues, however, the combination of the two made synchronization even easier while provided a more natural feeling at the same time. Bringoux et al. (2017) carried out an experimental research regarding the effect of speed-related auditory feedback on braking in a 3D-driving simulator. They conducted two kinds of experiments, testing naturalistic auditory feedback that mimic sounds issued from electric cars and synthesized auditory feedback with adjustments on the mapping between pitch variations and visual speed changes. The first experiment showed that sound stimuli from electric cars in its naturalistic form did not influenced braking kinematics as in combustion vehicles. On the other hand, synthesized auditory feedback can significantly influence braking modulations (initiation and regulation) by providing better information about vehicle speed changes and car dynamics. They also pointed out that synthesized auditory feedback improved speed control in emergency braking situations speeding up braking reaction and most of the times resulting in an earlier time-to-peak deceleration and car immobilization.

### **2.6.5 Videos**

Videos can also be used as an affective feedback technique, since as recent researches showed, video-based feedback assists in establishing a connection between the unconscious mind and the emotions (Hung, 2016), and removes both cognitive and affective barriers in the learning and skill acquisition process (Hall, Tracy & Lamey, 2016). Additionally, videos as a mean of feedback strengthen the bond between instructors and students (Borup et al. 2014; Parton, Crain-Dorough & Hancock, 2010) while also enhance the bond between peers (Hung, 2016). What is more, videos add additional, valuable layers with regards to the quantity, quality, and clarity of feedback (Hall, Tracy & Lamey, 2016).

To begin with, Crook et al. (2012) explored the use of videos in the feedback process in educational settings. They investigated the effect of videos on both teaching staff and students. For the needs of their research, they enabled teaching staff to produce

brief feedback videos via an online resource while students were able to remotely-access them. Through their study, Crook et al. (2012) inferred that the use of video technology is an advantageous feedback provision technique that reinforces active engagement and enhances the feedback experience for both teaching staff and students. They also highlighted that this feedback process evoked positive changes regarding the strategies developed and applied by the teaching staff in creating the feedback videos. In more detail, evidence of this study showed that videos increase time efficiency for staff, timeliness, and quality of feedback received by students. Additionally, this type of feedback improves students' potential to benefit from comments and observations, especially when addressed to part-time, overseas and distance learners. Moreover, videos can be used for both generic and individual feedback, with the latter being applicable only when resources allow and ratios between teaching staff and students are relatively low.

Additionally, Walker & White (2013) affirmed that video-based feedback enables language learners to perceive body language and facial expressions in addition to seeing artifacts. In that way, learners are able to comprehend the wider context of the learning process and make sense of their learning progress. As stated by Walker & White (2013) all the aforementioned factors result in learners' diligent engagement while learning becomes a more personalized experience. Henderson and Phillips (2015) also advocated that video-based feedback was perceived by students as being individualized and personalized. Key findings of their research demonstrated that students perceived this type of feedback as constructive with clear, detailed and unambiguous information while reflecting on their task performance and thinking.

The impact of video feedback on instructor social presence in blended courses was examined by Borup et al. (2014). Even though they did not reach in definitive results with regards to the focal point of their research, evidence indicated that from the students' point of view, feedback via video was rich in emotions, eased the feeling of connection with the instructor and provided a sense of conversation and interaction. As students reported they were able to elicit emotions through facial expressions, tone of voice, body language and gestures while they were also able to perceive their instructor's personality, mannerisms and demeanor. In addition, Borup et al. (2014) draw out that from instructors' viewpoint video-based feedback eased the expression of emotions, facilitated a more open and natural communication and aided in building a



sense of closeness with their students. These results were also supported by Hall, Tracy & Lamey (2016), who explored the benefits of video-based feedback for both instructors and students in the subject of philosophy. Furthermore, they pointed out that more information is delivered in a few minutes of face-to-face conversation than in a few written paragraphs since the conveyed nonverbal communication is an added value of the former. By the same token, video-based feedback is more clarified, more detailed, and easier to understand than traditional, written feedback since instructors do not just say more but actually elaborate more on explaining what they mean. What is more, video-based feedback is directed towards improvement, motivates students, and increases the engagement of and the sense of intimacy for both instructors and students. Last but not least, according to Hall, Tracy & Lamey (2016) video-based feedback is applied relatively easy, however issues of accessibility may arise.

# 3 The Prototype System

In this chapter the prototype system is introduced. At first, The functionality of the system and the flow of use is briefly described. Then, the requirements that need to be met in order to implement the system are presented, followed by the affective feedback technique that was selected for implementation and study.

## 3.1 The *Budget Manager* System

The aim of the *Budget Manager* system was to help users track their personal expenses and stay on budget. The system gathered information/data added by the users. This data/information was used to enable the research that was conducted in the context of this thesis. Based on the needs of this research the system focused on allowing users to track their expenses daily and weekly in terms of summary, bar graphs, and pie-charts.

The overview of the system's flow of use is described briefly below: The new user creates an account by completing the required information and consenting to the terms of use of the system. Then, the user has to activate the created account by following the link in the "Account Activation" email that the system sends. After that, the user can log in the system. When a user logs in the system for the first time, the system indicates that the weekly budget has to be set, i.e. the amount of money available to the user to spend over the week. Sequentially, all authorized users are able to add and monitor their expenses when convenient and at their own pace. At the end of the week, all users receives feedback regarding their performance and get notified if they managed to stay on budget. In order to start monitoring the expenses of the next week, the users are required to set again the available budget.

## 3.2 Functional Requirements

To achieve the desired functionality, the system was designed to satisfy a set of specific functional requirements. Table 1 explains briefly these requirements.

Table 1: Functional requirements of the *Budget Manager* prototype system

FR id	Reference	Description
1	Sign Up	The system should provide an interface that facilitates the creation of an account.
2	Account Activation	The system should activate each created account after a user's request in order to ensure that the created account belongs to a real person.
3	Sign In	The system should provide an interface where the user has to enter his/her unique credentials in order to log into the system.
4	Forgot Password	The system should provide an interface where a user can ask to reset the password of the account he/she created.
5	Reset Password	The system should provide an interface where a user can change the password of his/her account.
6	Contact	The system should provide an interface in order to ease communication between a user and the system's administrator.
7	Display Terms	The system should provide an interface where the user can access and read the terms of use any time.
8	Show Profile	The system should provide an interface where the users will be able to see the information of their accounts.
9	Manage Profile	The system should allow users to make changes to their profiles by updating their personal information.
10	Show Categories	The system should provide an interface where its users can see all the available categories for the expenses.
11	Add Categories	The system should allow its users to add categories for their expenses.
12	Update Categories	The system should allow users to update the categories they added.
13	Delete Categories	The system should allow users to delete the categories they added.
14	Set Budget	The system should provide an interface where its users will set their weekly budget.
15	Show Budget	The system should provide an interface where its user will be able to see their weekly budget.
16	Edit Budget	The system should allow its users to update the amount and the period of their weekly budget.
17	Add Goals	The system should facilitate setting goals for a week, i.e. the maximum amount that a user wants to spend for a category of expenses.
18	Display Goals	The system should display all goals that were added by a user for each week.
19	Sort Goals	The system should allow its users to sort their goals by category or by amount.
20	Edit Goals	The system should allow users to update the amount that

		was added for a goal.
21	Delete Goals	The system should allow users delete their own goals if they wish to.
22	Add Expenses	The system should provide an interface where its users can add their expenses.
23	Show Expenses	The system should provide an interface where its users will be able to see all the expenses that they added on the system, along with the critical information of each one of them, i.e. amount, category, date, time and payment method.
24	Filter Expenses	The system should allow its users to filter and display the expenses they desire.
25	Sort Expenses	The system should allow its users to sort their expenses by numerous ways in order to give them the opportunity to get a better understanding of their expenses.
26	Delete Expenses	The system should allow its users to delete their expenses at their will.
27	Display Details of Expenses	The system should provide an interface where the users can see all the details of the added expenses (one at a time) .
28	Update Expenses	The system should allow its users to update the information of each expense if they want to.
29	Show Statistics	The system should display statistics of the ongoing week, in order to provide an overview of a user's budget state. The system should also allow its users to see overviews of previous weeks, along with a summary of all weeks if they wish to. Moreover, for each week the system should allow users to see charts regarding their expenses in order to be able to monitor and comprehend better their expenses.
30	Provide Feedback	The system should display an overview of the budget and expenses to each user at the end of each week providing thus feedback regarding the user's performance.
31	Sign out	The system should allow users to log out of the system.
32	Inform Users About System's State	The system should inform its users about its state. The system should also inform its users for the result of any of their action.

### 3.3 Non-Functional Requirements

The system should also satisfy a set of specific non-functional requirements. Table 2 below presents these requirements.

Table 2: Non-functional requirements of the *Budget Manager* prototype system

<b>NFR id</b>	<b>Reference</b>	<b>Description</b>
1	Responsive Design	The system should make use of the available screen real estate and should display correctly at all screen sizes.
2	Accessibility	The system should be accessible and usable for everyone. Polished accessible experiences should be provided to all users taking care their diversity while focusing especially on those with motor, hearing and visibility impairments.
3	Precision	The system should display precise data that correspond to each user individually. The system should also store data that reference correctly to each user.
4	Personalization	The system should display the content dynamically, based on each user's data providing thus a personalized experience
5	Simplicity	The system should provide simple and neat interfaces that are both understandable and easy to use.
6	Zestful	The system should be interesting, joyful, and capable of engaging users and capture their attention.
7	Fast	The system should be fast both in terms of displaying the content and respond to users' actions.

The prototype system is also compliant with [ISO 9126](#) quality characteristics, as depicted in Table 3 below.

Table 3: ISO 9126 quality characteristics

<b>Category</b>	<b>Subcategory</b>	<b>Description</b>
Functionality	Compliance	The system should satisfy all the essential functional requirements as defined above, minimize intrusiveness and be compliant with privacy laws.
	Suitability	All system's functions should be appropriate and serve the system's scope.
	Accurateness	The system's functions should work correctly. In cases were users' data is involved the system should apply appropriate validation and transformation mechanisms.
	Interoperability	The interfaces of the system should render appropriately for authorized and unauthorized users. Each interface should appropriately be connected to other interfaces and, when necessary, able to successfully connect to the system's database.

	Security	Unauthorized access to the system should be prohibited. Each user should be only capable of managing and view only his/her account.
Reliability	Maturity	The system should rarely face failures.
	Fault Tolerance	The system should be able to withstand and recover from failures. Each interface should be able to operate independently in order to prevent the failing of one to affect the others.
	Recoverability	Ability to bring back a failed system to full operation, including data and network connections.
Usability	Understandability	The system's functionality should be easily understood.
	Learnability	Learning effort to use and comprehend the system should be minimized catering for novice users.
	Operability	The system should be easily and smoothly operated by a given user in a given environment.
Efficiency	Time behavior	Response time for a given thru put should be minimized, aiming at less than 5 sec.
	Resource Behavior	The system should rely only on network usage and depend only on the browser that a user has chosen to use.
Maintainability	Analyzability	The system should be able to identify the root cause of any failure.
	Changeability	The system should be developed in a way that it favors implementation of new future features. New features should be easily implemented in the system without losing its operational dynamic.
	Testability	The system should facilitate its verification and testing.
Portability	Adaptability	The system should be portable with smart phones, tablets, laptops and PCs. The adaptable system should operate correctly on any of these devices.

### 3.4 Providing feedback to users

As stated above, the system provided feedback to each user (functional requirement 30) at the end of the user's week (seven day period starting on the day the user selects). The feedback provided by the system is basically an overview of the user's expenses in regards to the budget that the user has set. That way users are

informed whether or not they managed to stay on budget. For all users this feedback includes:

- 1) the period that the system monitors,
- 2) the budget of the user,
- 3) the total amount of money that was spend on expenses, and the total amount of money saved in this period

However, the aim of this thesis was to examine the influence of affective feedback on users' performance regarding their expenses and budget. Therefore, the users were separated into two groups:

- 1) the control group, i.e. users that do not receive affective feedback, and
- 2) the experimental group, i.e. users that do receive affective feedback

The technique that was used to provide affective feedback to the experimental group was animated GIFs. This choice was based on the numerous advantages of the animated GIFs as already discussed in chapter 2. Specifically, this type of feedback was not only selected due to its ability to deliver complex feelings and convey emotions, but also because of its capability to augment and model affective performances by shaping and manipulating users' capacities. On the technological side, the advantages of the animated GIFs encompass wide and cross platform compatibility and portability, whilst they can add visual content and movement to a website or web application even when technological and bandwidth limitations exist.

Animated GIFs as a mean to provide feedback were implemented through the concept of an affective feedback card. In particular, two cards were implemented: one for users that managed to stay on budget and one for users that did not. Each card consists of four images: one basic and three complementary. All images used were retrieved from [giphy.com](http://giphy.com). To begin with, for the affective feedback card that was delivered to the users that managed to stay on budget an extended search on the "[giphy.com](http://giphy.com)" was conducted with the terms "*money*", "*cash*", "*make it rain*", "*excited*", "*exciting*", "*success*", and "*cheers*", whilst for the for the users that did not manage to stay on budget the search was conducted with the terms "*no money*", "*empty pockets*", "*empty wallet*", "*fail*", "*budget*", "*broke*", "*sad*", "*disappointed*", and "*frustrated*". From the vast amount of the results for each term, four animated GIFs

were selected for each card, focusing on those that capture facial expressions, body language and gestures of humans.

Figure 1 and Figure 2 depict eight frames of each one of the affective feedback cards that were implemented.

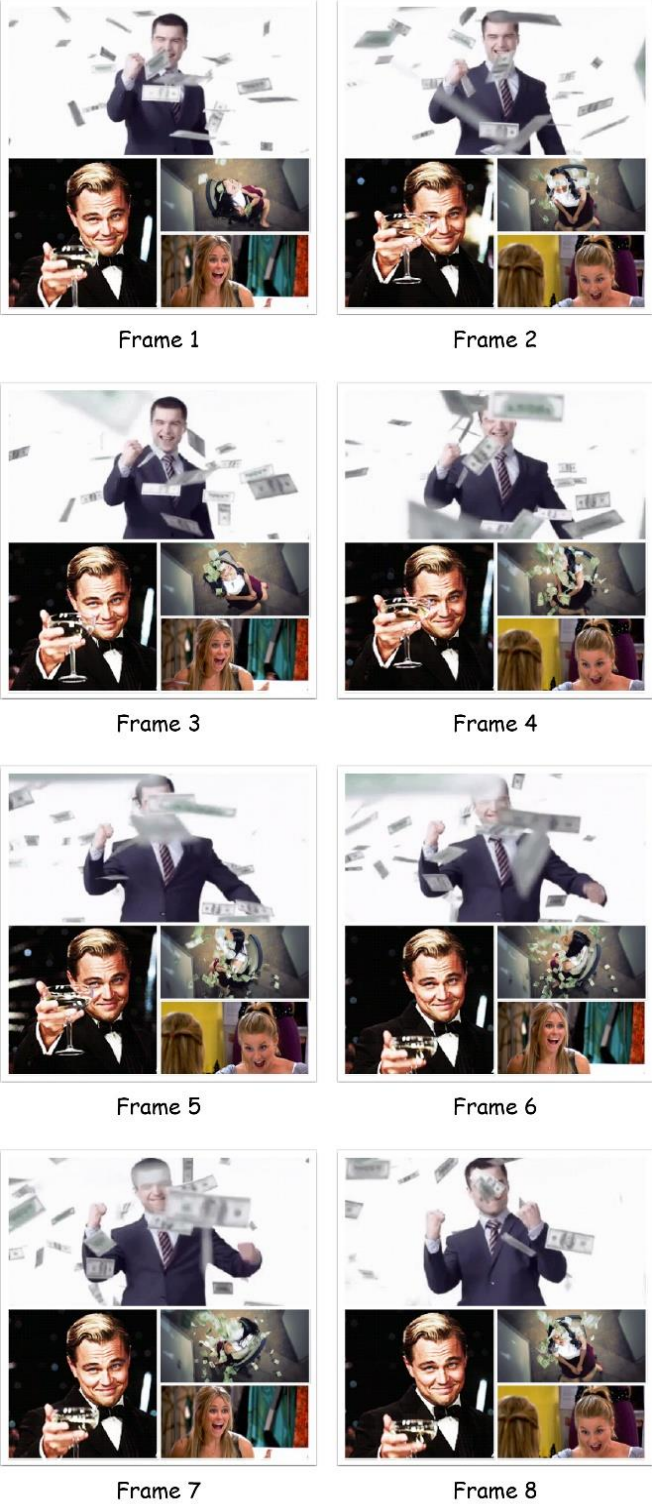


Figure 1: Eight frames of the affective feedback card that was delivered to users who managed to stay on budget





Figure 2: Eight frames of the affective feedback card that was delivered to users who did not manage to stay on budget

Noticeable is the fact that each one of the animated GIFs that were used has its own unique number of frames and is repeated independently of the others. Figures 1 - 4 are a sample of the final way that feedback was served to all users.

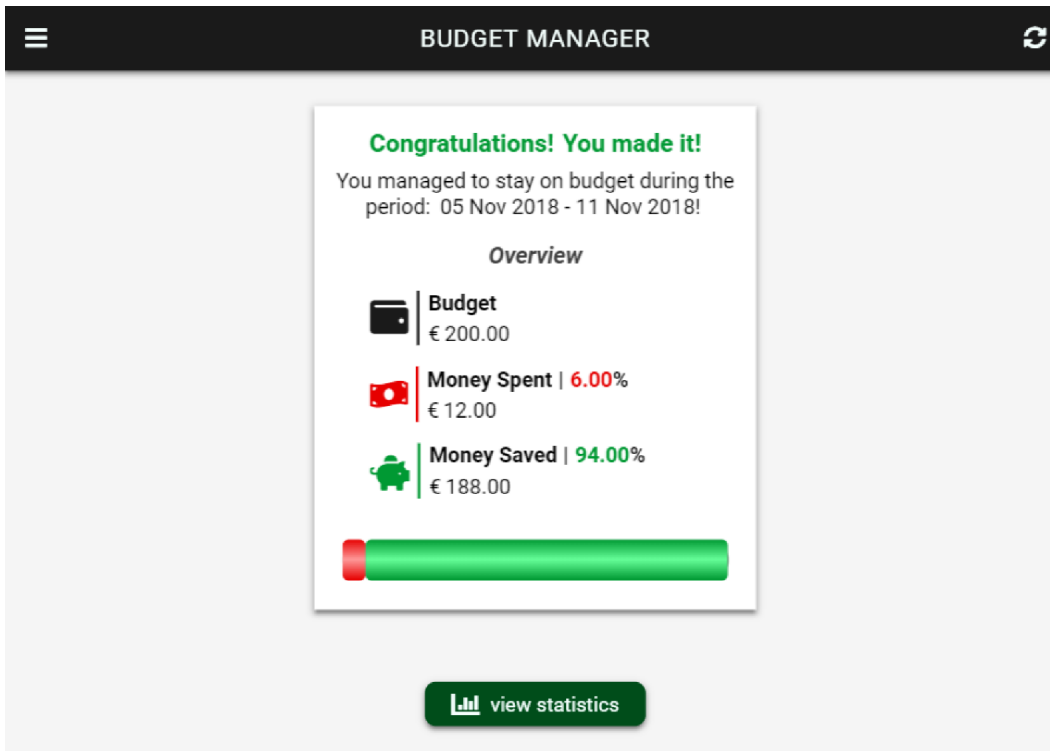


Figure 3: Feedback on budget for users in the control group who successfully stayed on budget

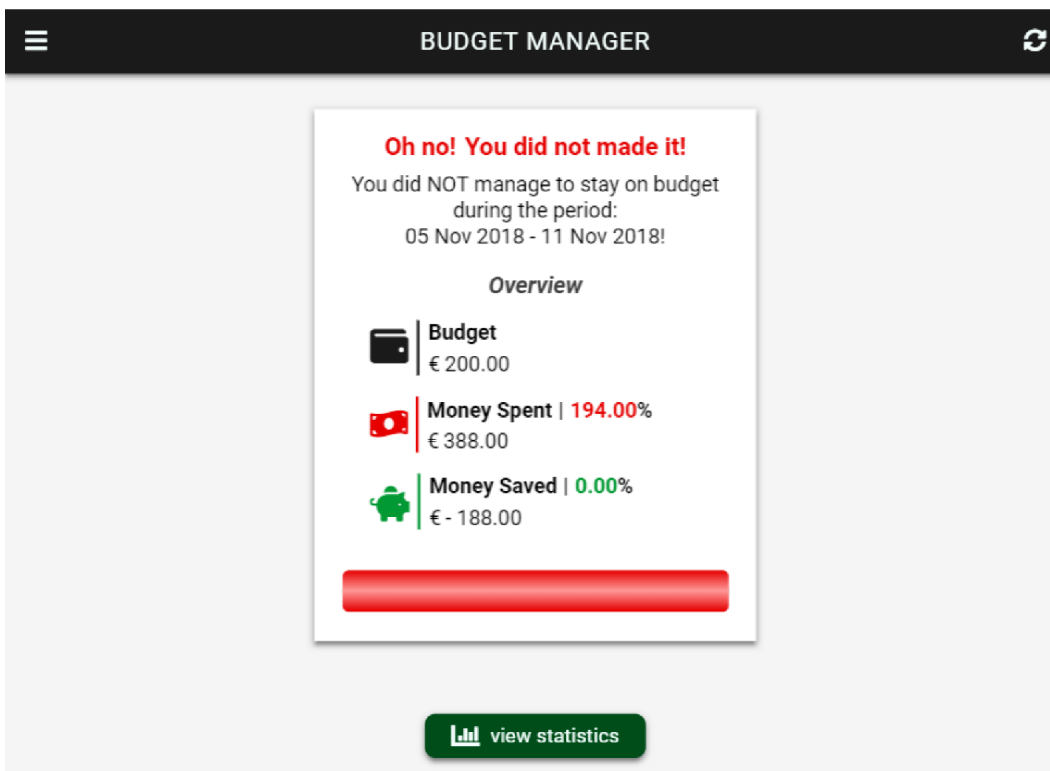


Figure 4: Feedback on budget for users in the control group who failed to stay on budget

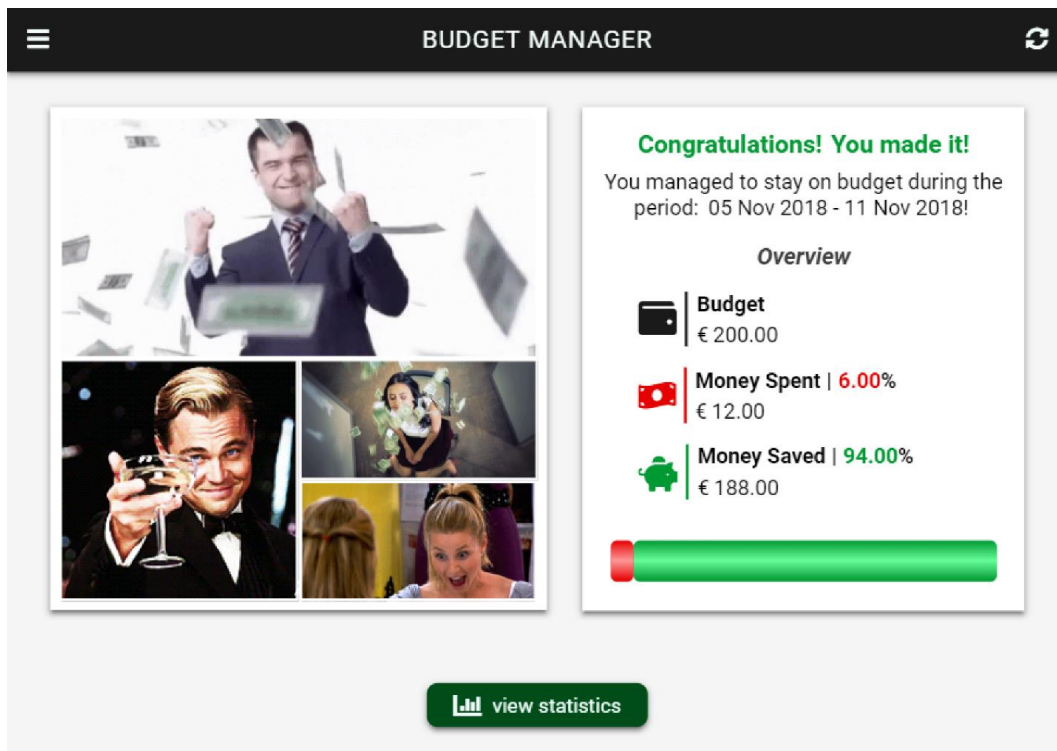


Figure 5: Feedback on budget for users in the experimental group who successfully stayed on budget

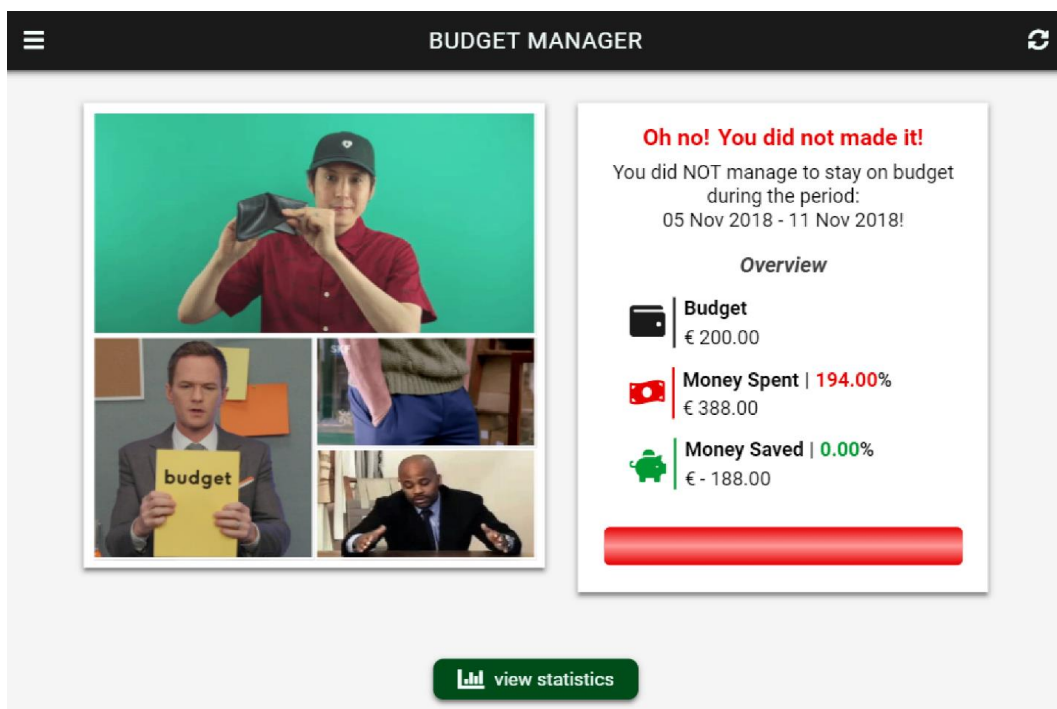


Figure 6: Feedback on budget for users in the experimental group who failed to stay on budget

# 4 System Architecture & Design

The current chapter presents the architecture of the prototype system and how it is designed to satisfy the requirements that were presented in the previous chapter. In the first section the architecture of the system is presented. The next two sections deal with the design of the database and the class diagram that refers to the back-end development of the system. The last section, describes the users' characteristics, the interfaces of the system and the use cases that depict the interaction between the users and the system.

## 4.1 System Architecture

The prototype system is basically a web application. Due to scalability and security concerns, the system employs the 3-tier deployment architecture as shown in Figure 7.

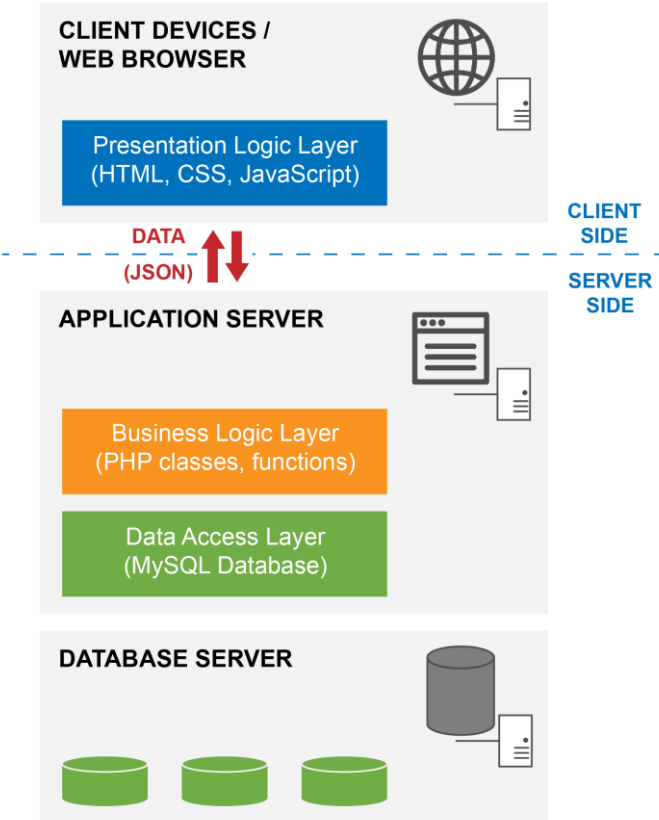


Figure 7: Architecture of the *Budget Manager* prototype system

The deployed three tier architecture dig consists of the client side, the application side and the database side. This approach allows a 3-way split of functions and clear separation of responsibilities and each component in the system plays a specific key role.

The client side performs the presentation logic and is responsible for handling the user interfaces (UIs). This layer displays data to users in a meaningful way and also accepts input from users. It is composed of three sublayers which operate in conjunction to apply the required logic:

- 1) The structure/content layer, which is basically the mark-up language defining what a certain text or media is (HTML).
- 2) The presentation layer, which is responsible for the look and feel of the interfaces (CSS, Cascading Style Sheets, images).
- 3) The behavior layer, which defines how different elements render and behave, how user actions are executed, initiates AJAX calls to retrieve the data, and modifies the content and the interface when necessary (JavaScript).

The application side handles the business/application logic and the data access logic. The first layer handles data validation, business rules and task-specific behavior, whilst the later communicates with the database by constructing and executing SQL queries. This layer is the mediator between the web browser and the database server, and returns the data stored in the database in JSON format to the web browser.

Finally, the database side is a separate component that uses the underlying database management system and is responsible for storing, updating, deleting and retrieving the data provided by the users to the system.

## **4.2 Database Design**

As mentioned above, the system gathers data from users and stores them in a database. This database consists of twelve tables in total and its collation is utf8\_unicode\_ci. The picture below illustrates the entity relationship diagram of the developed database, in its abstract form, whilst a detailed description of each table follows afterwards.

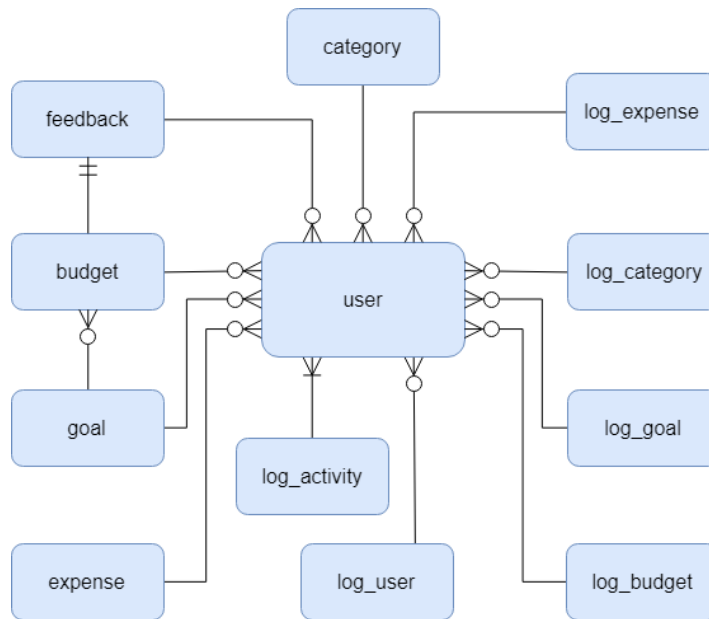


Figure 8: ERD (Entity Relationship Diagram) of the system's database

#### 4.2.1 Description of Entities

Each entity of the diagram above represents a table in the database and is described in Table 4 below.

Table 4: Description of the entities in the ERD

Entity	Description
user	The table where the system stores all necessary information for the system's users.
budget	The table where the system stores the budget of each user.
category	The table where the system stores all available categories of expenses. Users can also add their own preferable categories.
expense	The table where the system stores all expenses of every user.
feedback	The table where the system stores the feedback served to each user.
goal	The table where the system stores the goals of each user.
log_activity	The table where the system stores when users log into the system and when they are logged out.
log_user	The table where the system stores any changes that users make in their personal information.
log_category	The table where the system stores any changes that users make in their categories.
log_budget	The table where the system stores any changes that users make in their budget.
log_expense	The table where the system stores any changes that users make in their goals.
log_goal	The table where the system stores any changes that users make in their budget.

The structure of each table is presented in the following section along with a sample dataset of each one of them.

## 4.2.2 Table Specifications

The designed tables store the necessary data for both the system's functionality and the conducted research. Each table is presented analytically below.

### The *user* table

The *user* table stores all necessary information for the system's users. The primary key of this table is the field *id*, which is automatically incremented, while the fields *username* and *email* are unique for each user. The format and the type of the data are illustrated in Figure 9.

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>username</b> 🔑	varchar(255)	utf8_unicode_ci		No	None
3	<b>email</b> 🔑	varchar(40)	utf8_unicode_ci		No	None
4	<b>password</b>	varchar(255)	utf8_unicode_ci		No	None
5	<b>gender</b>	enum('male', 'female')	utf8_unicode_ci		No	None
6	<b>birthdate</b>	date			No	None
7	<b>feedback</b>	int(1)			No	None
8	<b>registered_at</b>	timestamp			No	CURRENT_TIMESTAMP
9	<b>activationcode</b>	varchar(255)	utf8_unicode_ci		No	0
10	<b>verified</b>	int(1)			No	0
11	<b>signed_in</b>	int(1)			No	0

Figure 9: Structure of table *user*

A sample containing five random rows of the *user* table was selected from the final dataset by executing the command:

```
SELECT * FROM `user` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 10.

id	username	email	password	gender	birthdate	feedback	registered_at	activationcode	verified	signed_in
71	diosls	dsoulis1@gmail.com	\$2y\$10\$EKH	male	1990-01-01	1	2018-10-17 20:18:56	\$2y\$10\$sk2JTU	1	1
119	sofibab	sofibab@gmail.com	\$2y\$10\$SRL8	female	1989-10-14	0	2018-10-20 21:44:50	\$2y\$10\$E.dUW	1	1
121	stark	evf.trev@gmail.com	\$2y\$10\$reilZ	female	1993-06-23	0	2018-10-21 00:11:27	\$2y\$10\$Uft.pkiC	1	1
13	chris_teg	tegoschris@yahoo.gr	\$2y\$10\$gGC	male	1996-08-23	0	2018-10-16 19:24:30	\$2y\$10\$1.wvzF	0	0
29	kiki_kok	kyriakikokkinidou@gmail.co	\$2y\$10\$IIOH	female	1989-05-31	0	2018-10-16 23:31:53	\$2y\$10\$im7.SV	1	1

Figure 10: A sample containing five random rows of the *user* table

## The *budget* table

The budget table stores information gathered regarding the budget of each user. The format and the type of this table's data are presented in Figure 11 below.



#	Name	Type	Collation	Attributes	Null	Default
1	id 	int(11)			No	None
2	user_id 	int(11)			No	None
3	amount	float			No	None
4	budget_from	date			No	None
5	budget_to	date			No	None
6	created	timestamp			No	CURRENT_TIMESTAMP

Figure 11: Structure of table *budget*

The primary key of this table is the field *id*, which is automatically incremented, while the *user\_id* field is the foreign key that references to the *id* of the *user* table. A sample containing five random rows of the *budget* table was selected from the final dataset by executing the command:

```
SELECT * FROM `budget` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 12 below.

id	user_id	amount	budget_from	budget_to	created
397	9	120	2018-11-05	2018-11-11	2018-11-05 14:32:40
137	97	150	2018-10-22	2018-10-28	2018-10-22 15:57:32
244	61	60	2018-10-24	2018-10-30	2018-10-25 19:57:23
318	14	120	2018-10-30	2018-11-05	2018-10-30 22:09:06
230	75	100	2018-10-24	2018-10-30	2018-10-25 00:25:34

Figure 12: A sample containing five random rows of the *budget* table

## The *category* table

This table contains the categories of the expenses that are available for each user. The format and the type of the data that are stored in this table are depicted in Figure 13.



#	Name	Type	Collation	Attributes	Null	Default
1	id 	int(11)			No	None
2	category_name	varchar(255)	utf8_unicode_ci		No	None
3	added_by 	int(11)			Yes	NULL

Figure 13: Structure of table *category*



The primary key of this table is the field `id`, which is automatically incremented. The field `added_by` stores the `id` of the user that adds a category and is a foreign key that references to the `id` of the user table. This field was implemented to accept null values, since the system provides twenty categories of expenses to users. The categories that are provided by the system for each user are illustrated in Figure 14.

<code>id</code>	<code>category_name</code>	<code>added_by</code>
1	bar & café	<i>NULL</i>
2	bills & fees	<i>NULL</i>
3	clothing	<i>NULL</i>
4	communication	<i>NULL</i>
5	cosmetics & beauty	<i>NULL</i>
6	donations & charity	<i>NULL</i>
7	education	<i>NULL</i>
8	entertainment	<i>NULL</i>
9	gifts	<i>NULL</i>
10	health	<i>NULL</i>
11	housing	<i>NULL</i>
12	investments	<i>NULL</i>
13	restaurant & delivery	<i>NULL</i>
14	sports & fitness	<i>NULL</i>
15	supermarket	<i>NULL</i>
16	technology	<i>NULL</i>
17	transportation	<i>NULL</i>
18	traveling & vacation	<i>NULL</i>
19	vehicle	<i>NULL</i>
20	miscellaneous	<i>NULL</i>

Figure 14: Basic categories of expenses

## The *expense* table

All expenses of each user are stored in the *expense* table. The format and the type of the data of this table are illustrated in Figure 15.

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>user_id</b> 🔑	int(11)			No	None
3	<b>amount</b>	float			No	None
4	<b>category</b>	varchar(255)	utf8_unicode_ci		No	None
5	<b>payment</b>	enum('cash', 'credit_card', 'debit_card', 'prepaid')	utf8_unicode_ci		No	cash
6	<b>expense_date</b>	date			No	None
7	<b>expense_time</b>	varchar(5)	utf8_unicode_ci		No	None
8	<b>location</b>	varchar(255)	utf8_unicode_ci		Yes	NULL
9	<b>store</b>	varchar(255)	utf8_unicode_ci		Yes	NULL
10	<b>comments</b>	varchar(255)	utf8_unicode_ci		Yes	NULL
11	<b>created_at</b>	timestamp			No	CURRENT_TIMESTAMP

Figure 15: Structure of table *expense*

In the figure above we see the information gathered for each one of the expenses for every user. The primary key of this table is the field *id*, which is automatically incremented, while the *user\_id* field is the foreign key that references to the *id* of the user table. Fields *location*, *store* and *comments* are optional, whilst the field *expense\_time* will be also an optional choice for users, but if it is not specified the time that will be inserted will be the same as the time in the *created\_at* field. A sample containing five random rows of the *expense* table was selected from the final dataset by executing the command:

```
SELECT * FROM `expense` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 16.

id	user_id	amount	category	payment	expense_date	expense_time	location	store	comments	created_at
291	71	4	supermarket	debit card	2018-10-17	17:09	ah campus			2018-10-18 18:10:20
3213	24	2	supermarket	debit card	2018-11-04	09:07				2018-11-04 09:07:35
269	73	5	bar & café	cash	2018-10-18	11:19	thessaloniki	4all	coffee , sandwich	2018-10-18 13:36:59
814	4	5.85	miscellaneous	cash	2018-10-20	16:50				2018-10-21 05:21:21
2411	10	13	bar & café	cash	2018-10-26	22:30				2018-10-30 22:30:57

Figure 16: A sample containing five random rows of the *expense* table

## The *feedback* table

The feedback table stores all information regarding the feedback that is delivered to the users. The format and the type of the data of this table are depicted in Figure 17 below.

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>user_id</b> 🔑	int(11)			No	None
3	<b>budget_id</b> 🔑	int(11)			No	None
4	<b>type</b>	enum('regular', 'affective gifs')	utf8_unicode_ci		No	regular
5	<b>user_performance</b>	enum('fail', 'success')	utf8_unicode_ci		No	None
6	<b>served_at</b>	timestamp			No	CURRENT_TIMESTAMP

Figure 17: Structure of table *feedback*

The primary key of this table is the field `id`, which is automatically incremented, while the `user_id` field is the foreign key that references to the `id` of the user table, and the `budget_id` field is the foreign key that references to the `id` of the budget table. For users that belong in the experimental group the `type` field holds the value 'affective gifs' whilst for users that belong in the control group the value 'regular'. The `type` field was implemented in that way in order to be more flexible in testing more feedback techniques in the future. This could be easily implemented by just adding one more value in the enumeration list. A sample containing five random rows of the *feedback* table was selected from the final dataset by executing the command:

```
SELECT * FROM `feedback` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 18.

id	user_id	budget_id	type	user_performance	served_at
102	87	78	affective gifs	fail	2018-10-25 08:43:39
240	76	249	regular	success	2018-11-01 21:04:01
28	101	91	affective gifs	fail	2018-10-22 21:54:22
388	137	388	affective gifs	success	2018-11-09 22:14:14
223	77	222	regular	fail	2018-10-31 21:14:11

Figure 18: A sample containing five random rows of the *feedback* table

## The *goal* table

This table contains the goals of each user for a specific budget. The format and the type of the data that are stored in this table are illustrated in the Figure 19 below.

#	Name	Type	Collation	Attributes	Null	Default
1	id 🔑	int(11)			No	None
2	budget_id 🔑	int(11)			No	None
3	user_id 🔑	int(11)			No	None
4	amount	float			No	None
5	category	varchar(255)	utf8_unicode_ci		No	None
6	created	timestamp			No	CURRENT_TIMESTAMP

Figure 19: Structure of table *goal*

The primary key of this table is the field *id*, which is automatically incremented, while the *user\_id* field is the foreign key that references to the *id* of the *user* table, and the *budget\_id* field is the foreign key that references to the *id* of the *budget* table. The *goal* refers to the maximum amount that the user is willing to spent on a category of expenses. The goals were implemented as a technique to help users monitor better their expenses and are an optional feature for each user. A sample containing five random rows of the *goal* table was selected from the final dataset by executing the command:

```
SELECT * FROM `goal` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 20.

id	budget_id	user_id	amount	category	created
86	127	51	15	bar & café	2018-10-22 11:55:43
32	49	57	150	bar & café	2018-10-17 14:30:29
99	246	94	20	clothing	2018-10-25 21:42:26
6	4	10	20	bar & café	2018-10-16 20:11:36
10	6	12	10	restaurant & delivery	2018-10-16 20:29:12

Figure 20: A sample containing five random rows of the *goal* table

## The *log\_activity* table

In this table the activity of each user is stored and specifically the time that a user signs in or signs out of the system. The format and the type of the data of this table are depicted in Figure 21.

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>user_id</b> 🔑	int(11)			No	None
3	<b>log_time</b>	datetime			No	None
4	<b>activity_type</b>	enum('sign_in', 'sign_out')	utf8_unicode_ci		No	None

Figure 21: Structure of table *log\_activity*

The primary key of this table is the field *id*, which is automatically incremented, while the *user\_id* field is the foreign key that references to the *id* of the *user* table. A sample containing five random rows of the *log\_activity* table was selected from the final dataset by executing the command:

```
SELECT * FROM `log_activity` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 22.

id	user_id	log_time	activity_type
84	49	2018-10-17 08:13:03	sign_in
515	87	2018-10-20 22:02:33	sign_out
2984	128	2018-11-05 01:29:55	sign_in
326	32	2018-10-19 03:14:54	sign_out
3107	64	2018-11-05 21:13:45	sign_in

Figure 22: A sample containing five random rows of the *log\_activity* table

## The *log\_user* table

The *log\_user* table stores the modifications that the users make to their profile. The *updated\_field* stores the name of the field which is updated, the *prev\_value* stores the previous value of the field that is updated, whilst the *new\_value* stores the new value. The primary key of this table is the field *id*, which is automatically incremented, while the *user\_id* field is the foreign key that references to the *id* of the *user* table. The structure of this table are illustrated in Figure 23.

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>user_id</b> 🔑	int(11)			No	None
3	<b>updated_field</b>	varchar(20)	utf8_unicode_ci		No	None
4	<b>prev_value</b>	varchar(255)	utf8_unicode_ci		No	None
5	<b>new_value</b>	varchar(255)	utf8_unicode_ci		No	None
6	<b>log_time</b>	timestamp			No	CURRENT_TIMESTAMP

Figure 23: Structure of table *log\_user*

A sample containing five random rows of the *log\_user* table was selected from the final dataset by executing the command:

```
SELECT * FROM `log_user` ORDER BY RAND() LIMIT 5
```

The result is depicted in the following Figure 24.

id	user_id	updated_field	prev_value	new_value	log_time
2	4	birthdate	1990-01-01	1990-08-06	2018-10-16 04:51:13
4	96	gender	male	female	2018-10-19 01:01:04
5	41	birthdate	1990-01-01	1990-03-28	2018-10-21 02:24:10
1	4	gender	male	female	2018-10-16 04:51:01
3	85	birthdate	1970-01-01	1987-03-17	2018-10-18 07:16:27

Figure 24: A sample containing five random rows of the *log\_user* table

### The *log\_category* table

The *log\_category* table stores all records that are added in the category table by the users. The structure of this table are presented in the Figure 25.

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>user_id</b> 🔑	int(11)			No	None
3	<b>category_id</b>	int(11)			No	None
4	<b>category_name</b>	varchar(255)	utf8_unicode_ci		No	None
5	<b>log_type</b>	enum('inserted', 'deleted', 'updated')	utf8_unicode_ci		No	None
6	<b>log_time</b>	timestamp			No	CURRENT_TIMESTAMP

Figure 25: Structure of table *log\_category*

The primary key of this table is the field *id*, which is automatically incremented, while the *user\_id* field is the foreign key that references to the *id* of the user table.

A sample containing five random rows of the *log\_category* table was selected from the final dataset by executing the command:

```
SELECT * FROM `log_category` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 26.

id	user_id	category_id	category_name	log_type	log_time
6	56	26	bakery	inserted	2018-10-22 23:58:16
5	8	25	opap	inserted	2018-10-20 21:33:05
2	71	22	tickets	inserted	2018-10-17 20:23:33
4	98	24	betting	inserted	2018-10-19 00:47:31
3	98	23	smoking	inserted	2018-10-19 00:47:24

Figure 26: A sample containing five random rows of the *log\_category* table

### The tables *log\_budget*, *log\_expense* and *log\_goal*

These tables were implemented in order to monitor how users modify their budget, their expenses and their goals respectively. These tables are the same as the original they reference to. The difference is that the field where the timestamp of the record is stored was renamed to *log\_time*. Furthermore, two more fields were added to these tables:

- 1) The *log\_type* field, which is an enumeration of the values “inserted”, “deleted” and “updated”. This field stores the type of the modification that a user makes and is common for all tables.
- 2) The *budget\_id*, *expense\_id* and *goal\_id* respectively, which hold the id of the original record.

For each of these tables the primary key of the table is the field *id*, which is automatically incremented, while the *user\_id* field is the foreign key that references to the id of the user table. Note that the *budget\_id* field, which is the foreign key that references to the id of the budget table, makes actually sense to be stored in the table since for each user it is not allowed to delete a budget once it is inserted in the system.

The specification of each table is illustrated in the following Figures 27- 29.

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>user_id</b> 🔑	int(11)			No	None
3	<b>budget_id</b>	int(11)			No	None
4	<b>amount</b>	float			No	None
5	<b>budget_from</b>	date			No	None
6	<b>budget_to</b>	date			No	None
7	<b>log_type</b>	enum('inserted', 'deleted', 'updated')	utf8_unicode_ci		No	None
8	<b>log_time</b>	timestamp			No	CURRENT_TIMESTAMP

Figure 27: Structure of table *log\_budget*

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>user_id</b> 🔑	int(11)			No	None
3	<b>expense_id</b>	int(11)			No	None
4	<b>amount</b>	float			No	None
5	<b>category</b>	varchar(255)	utf8_unicode_ci		No	None
6	<b>payment</b>	enum('cash', 'credit_card', 'debit_card', 'prepaid...')	utf8_unicode_ci		No	cash
7	<b>expense_date</b>	date			No	None
8	<b>expense_time</b>	varchar(5)	utf8_unicode_ci		No	None
9	<b>location</b>	varchar(255)	utf8_unicode_ci		Yes	NULL
10	<b>store</b>	varchar(255)	utf8_unicode_ci		Yes	NULL
11	<b>comments</b>	varchar(255)	utf8_unicode_ci		Yes	NULL
12	<b>log_type</b>	enum('inserted', 'deleted', 'updated')	utf8_unicode_ci		No	inserted
13	<b>log_time</b>	timestamp			No	CURRENT_TIMESTAMP

Figure 28: Structure of table *log\_expense*

#	Name	Type	Collation	Attributes	Null	Default
1	<b>id</b> 🔑	int(11)			No	None
2	<b>goal_id</b>	int(11)			No	None
3	<b>budget_id</b> 🔑	int(11)			No	None
4	<b>user_id</b> 🔑	int(11)			No	None
5	<b>amount</b>	float			No	None
6	<b>category</b>	varchar(255)	utf8_unicode_ci		No	None
7	<b>log_type</b>	enum('inserted', 'updated', 'deleted')	utf8_unicode_ci		No	None
8	<b>log_time</b>	timestamp			No	CURRENT_TIMESTAMP

Figure 29: Structure of table *log\_goal*

A sample containing five random rows of the *log\_budget* table was selected from the final dataset by executing the command:

```
SELECT * FROM `log_budget` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 30.



id	user_id	budget_id	amount	budget_from	budget_to	log_type	log_time
304	128	120	50	2018-10-22	2018-10-28	updated	2018-10-28 22:30:28
154	129	130	200	2018-10-17	2018-10-23	inserted	2018-10-22 11:45:05
227	4	109	200	2018-10-22	2018-10-28	updated	2018-10-24 10:46:29
411	86	369	150	2018-11-01	2018-11-06	inserted	2018-11-01 19:39:41
256	79	223	70	2018-10-24	2018-10-30	inserted	2018-10-24 22:33:44

Figure 30: A sample containing five random rows of the *log\_budget* table

A sample containing five random rows of the *log\_expense* table was selected from the final dataset by executing the command:

```
SELECT * FROM `log_expense` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 31.

id	user_id	expense_id	amount	category	payment	expense_date	expense_time	location	store	comments	log_type	log_time
374	80	354	10	technology	credit card	2018-10-18	22:47	eshop stavroupoli			inserted	2018-10-18 22:47:42
585	100	552	45.8	education	cash	2018-10-19	23:05			book for my master	updated	2018-10-19 23:10:23
1325	33	1257	75	clothing	credit card	2018-10-23	12:18		cosmos		deleted	2018-10-23 12:40:18
2894	96	2783	10	bills & fees	cash	2018-11-01	22:34	thessaloniki	4all	mobile card	inserted	2018-11-01 22:34:46
3688	44	3564	3.6	bar & café	cash	2018-11-06	20:45				inserted	2018-11-06 20:45:56

Figure 31: A sample containing five random rows of the *log\_expense* table

A sample containing five random rows of the *log\_goal* table was selected from the final dataset by executing the command:

```
SELECT * FROM `log_goal` ORDER BY RAND() LIMIT 5
```

The result is depicted in Figure 32.

id	goal_id	budget_id	user_id	amount	category	log_type	log_time
69	48	65	75	50	bar & café	inserted	2018-10-17 21:06:43
34	25	37	47	10	entertainment	inserted	2018-10-17 01:35:34
64	43	56	66	50	restaurant & delivery	updated	2018-10-17 18:32:53
6	2	1	4	20	gifts	deleted	2018-10-16 04:44:19
94	72	88	98	20	smoking	inserted	2018-10-19 00:50:51

Figure 32: A sample containing five random rows of the *log\_goal* table

The developed database constitutes the resource of the dataset that is used for analysis. The final dataset encompasses explicit information given by actual users who use the system for a certain period. Note that the system gathers data by authorized users only for all tables except the *user* table. The latter table keeps records of all users that were signed up whether or not they verified their account and used the system.

## 4.3 Back-End Design

Having identified the structure of the system's database, the design of the code that handles the data was defined. This code is written in object oriented programming PHP and consists of the following seven models/classes:

- 1) The *Database* class, which is responsible for handling both the connection and the data between the system and the database. This class is the core of the back-end development, since all other classes are connected to it, in order to facilitate their functionality. Its basic functionality encompasses: running queries to the database, updating fields in the tables and deleting records.
- 2) The *User* class, which implements the required functionality for every user that accesses the system. This class is responsible for creating an account for a user (sign up a user), activating an account, allowing a user to request and change his/her password, logging a user into the system, and sending emails from the system to a user and vice versa.
- 3) The *LoggedUser* class, which implements the required functionality for every authorized user that accesses the system. This class is a subclass of the *User* class, i.e. inherits everything from the parent class, and implements the additional functionality required for authorized users. Specifically, the *LoggedUser* class handles all data that reference to a user: personal information, categories of expenses, budgets, expenses, goals, and controls all other classes.
- 4) The *BudgetList* class, which is responsible for managing all data regarding the budget of a user. This class retrieves and returns all budgets that a user has set, and facilitates the addition and update of a budget.
- 5) The *CategoryList* class, which is responsible for managing all data regarding the categories of expenses of a user. This class retrieves and returns all categories of expenses that are available for a user, and facilitates the addition, deletion and update of a category.
- 6) The *GoalList* class, which is responsible for managing all data regarding the goals of a user. This class retrieves and returns all goals that a user has set, and facilitates the addition, deletion and update of a goal.

7) The *ExpenseList* class, which is responsible for managing all data regarding the expenses of a user. This class retrieves and returns all expenses that a user added, and facilitates the addition, deletion and update of expenses.

The following class diagram (Figure 33) illustrates the functionality and structure of each class and also, indicates how these models connect to and interact with each other. Note that all classes and models are accessed through the *dataManager.php* file. This file is written in procedural PHP and is responsible for calling the appropriate model and returning the data in JSON format.

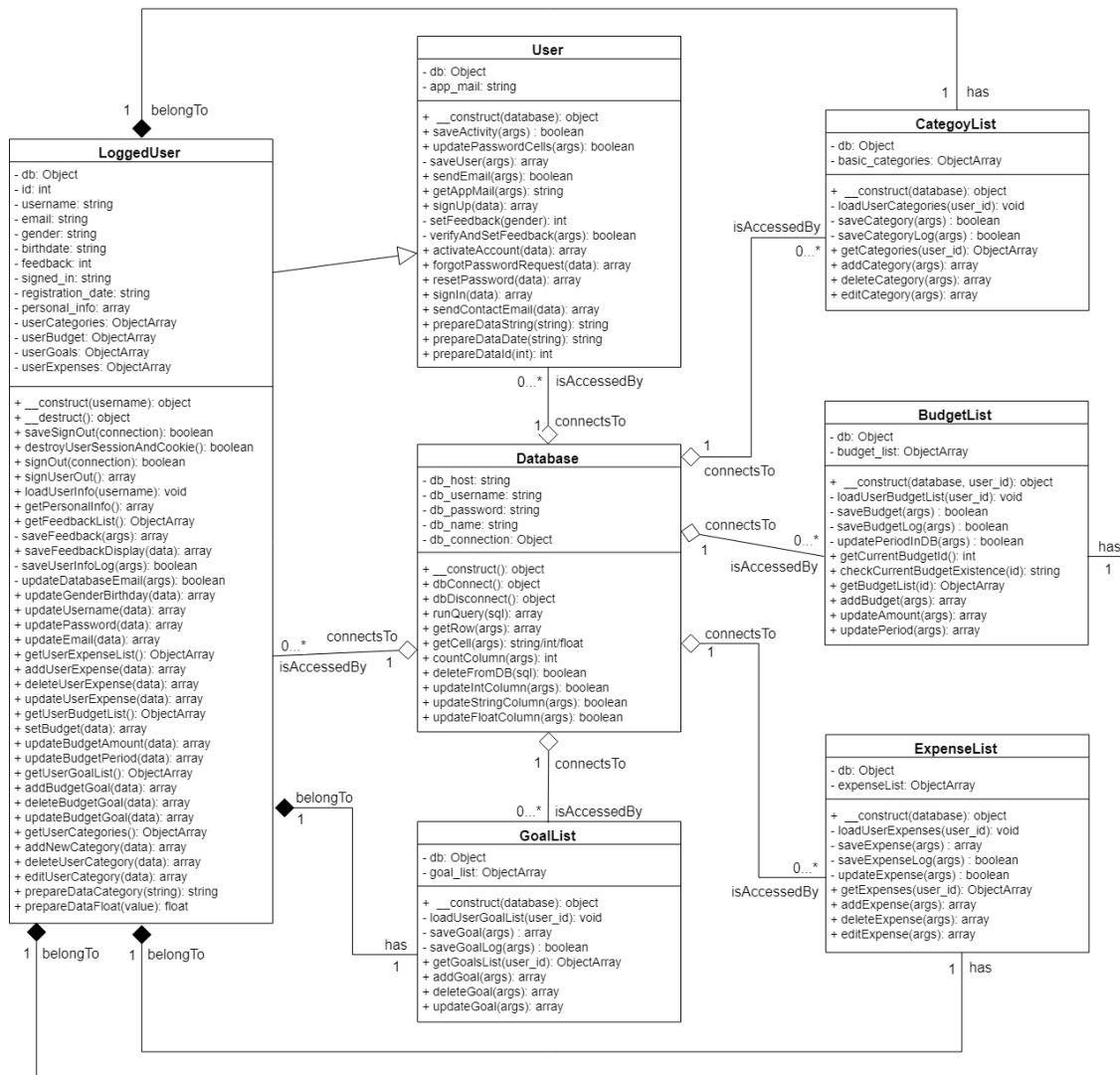


Figure 33: Class diagram of the php models

### 4.4 User Characteristics and Interfaces

All users interact with the system through the developed interfaces. However, there are two different types of users: authorized and unauthorized users. Each user has a specific role when accessing the system. This means that the user can be either authorized or unauthorized at a specific point in time when using the system but cannot be both. For each user type there are specific privileges, unique actions that can be performed, and sole interfaces that are allowed to be accessed. In other words, there are responsibilities and requirements characterizing each type of user individually. Nevertheless, there are common interfaces and actions for both types of user.

Considering the above, it is crystal clear that the system should cater for both types of users, performing the appropriate actions to provide the needed interfaces and respond to users' actions. Based on the system's requirements described above, the first step in designing the system was to identify the necessary interfaces and how users interact with the system through them. The next sections illustrate and describe this interaction.

#### 4.4.1 Unauthorized users

Figure 34 depicts the interfaces for unauthorized users.

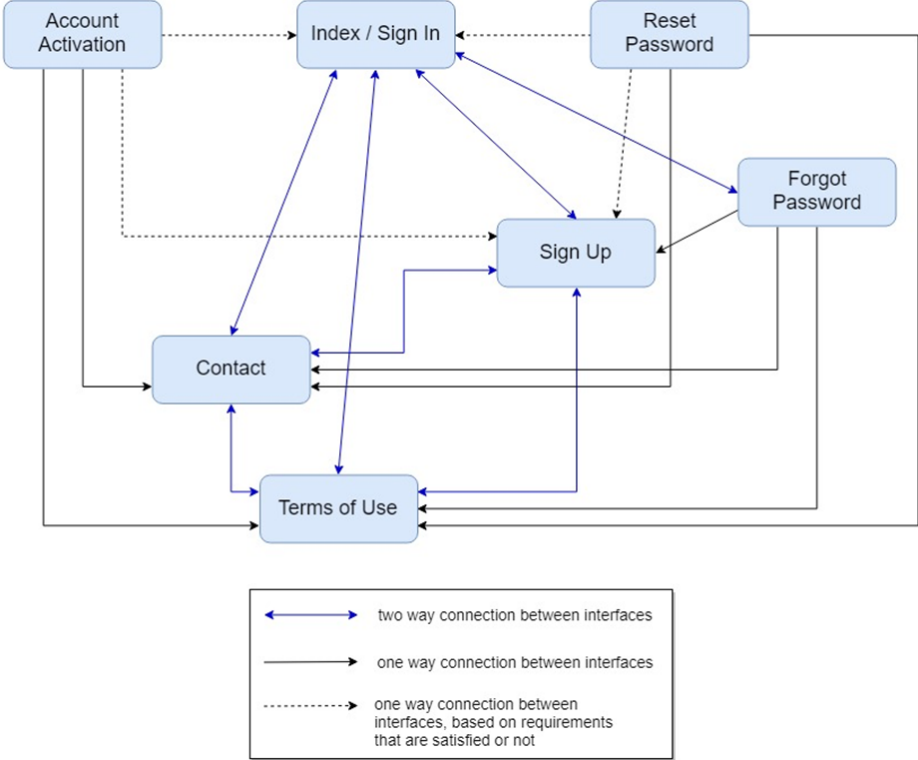


Figure 34: Interfaces for unauthorized users

As illustrated in the picture above, unauthorized users are able to access and use the system through seven (7) interfaces. These interfaces facilitate the implementation of the functional requirements FR1 to FR7. Table 5 below describes briefly the use case of each interface, its connections to the other interfaces, and the functional requirements that have to be implemented.

Table 5: Description of interfaces for unauthorized users

<b>Interface</b>	<b>Use Case</b>	<b>Navigation To</b>	<b>FR id</b>
Index/Sign In	This is the starting interface of the system for unauthorized users. Though this interface unauthorized users are allowed to access their accounts.	<ul style="list-style-type: none"> <li>• Index/Sign In</li> <li>• Sign Up</li> <li>• Contact</li> <li>• Terms of Use</li> <li>• Forgot Password</li> </ul>	3
Sign Up	Though this interface unauthorized users are allowed to create an account.	<ul style="list-style-type: none"> <li>• Index/Sign In</li> <li>• Sign Up</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	1
Contact	Though this interface unauthorized users are allowed to send contact emails to the administrator of the app, to make suggestions and give feedback, ask questions or request help.	<ul style="list-style-type: none"> <li>• Index/Sign In</li> <li>• Sign Up</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	6
Terms of Use	Though this interface unauthorized users are allowed to display and read the terms of use of this system.	<ul style="list-style-type: none"> <li>• Index/Sign In</li> <li>• Sign Up</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	7
Account Activation	Though this interface unauthorized users who created an account are allowed to activate their account. During sign up the system sends an email, to the email address that the users provide, with a link to this interface. Activation is performed automatically and users can see the result. In case that an authorized user accesses this interface the system redirects the user to the “Home” interface (described below).	<ul style="list-style-type: none"> <li>• Index/Sign In</li> <li>• Sign Up</li> <li>• Contact</li> <li>• Terms of Use</li> <li>• Account Activation</li> </ul>	2
Forgot Password	Though this interface unauthorized users are allowed to make a request to reset their password in case that they have forgotten it.	<ul style="list-style-type: none"> <li>• Index/Sign In</li> <li>• Sign Up</li> <li>• Contact</li> <li>• Terms of Use</li> <li>• Forgot Password</li> </ul>	5

Reset Password	Though this interface unauthorized users who are allowed to change their password after their request. When users request to change their password the system sends an email, to the email address that the users provide, with a link to this interface. In case that an authorized user accesses this interface the system redirects the user to the “Home” interface (described below).	<ul style="list-style-type: none"> <li>• Index/Sign In</li> <li>• Sign Up</li> <li>• Contact</li> <li>• Terms of Use</li> <li>• Reset Password</li> </ul>	4
----------------	--	---	---

In the diagram below (Figure 35) the use cases that are described in table 11 are depicted.

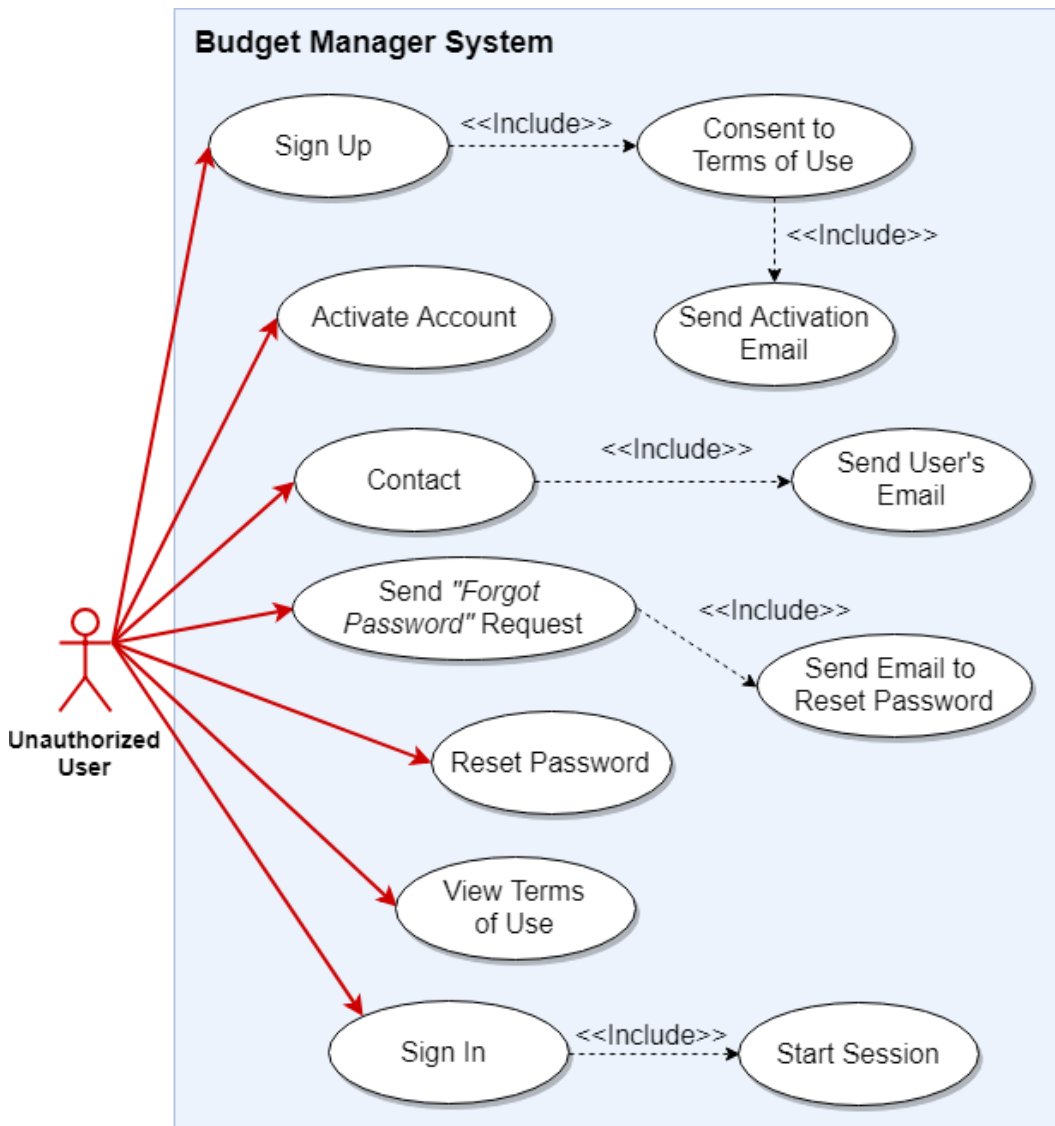


Figure 35: Use cases of interaction between the system and unauthorized users

#### 4.4.2 Authorized users

The following figure (Figure 36) depicts the interfaces for authorized users.

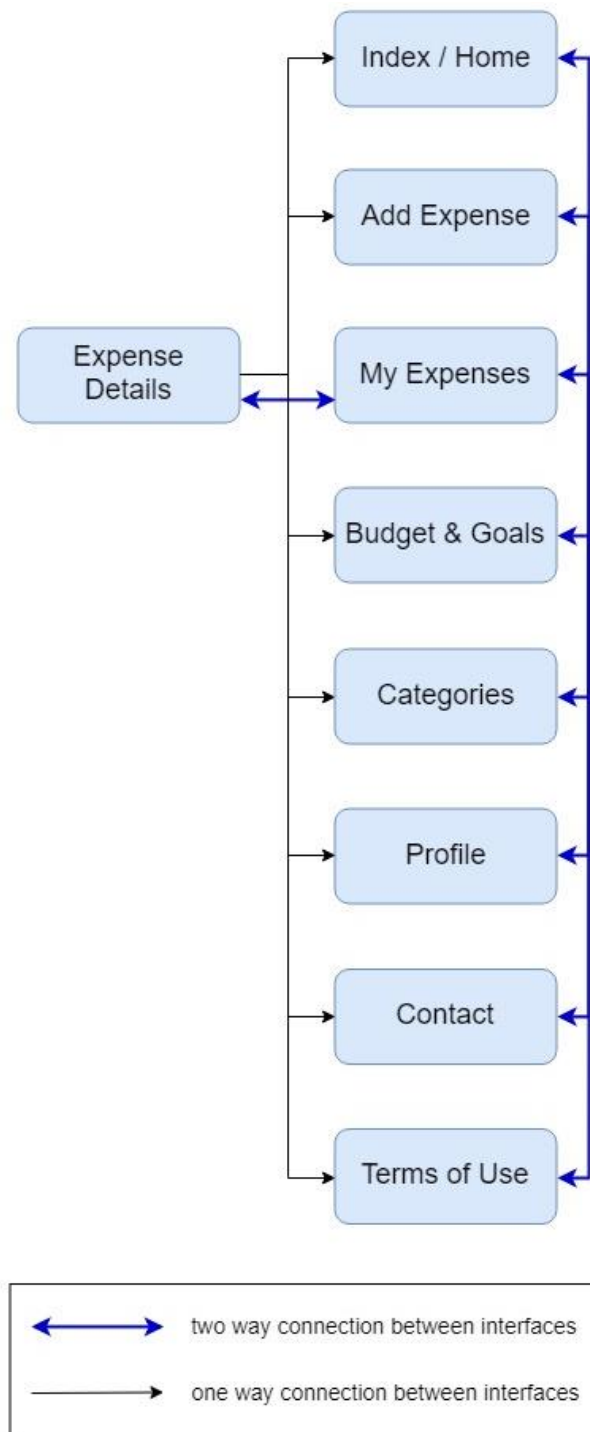


Figure 36: Interfaces for authorized users

Table 6 below describes briefly the use case of each interface and its connections to the other interfaces as long as the functional requirement that has to be implemented. Noticeable is the fact that an authorized user is able to log out of the system any time

from all interfaces. This means that the functional requirement F31 is common for each one of these interfaces.

Table 6: Description of interfaces for authorized users

<b>Interface</b>	<b>Use Case</b>	<b>Navigation To</b>	<b>FR id</b>
Index/Home	This is the starting interface of the system for authorized users. Though this interface authorized users are allowed to monitor their expenses. Users can see the overview of each week and display statistics for all weeks. For each week users can see a summary of their expenses, while they are also able to see their expenses per day, per goal (if they have defined a goal), per category and per payment method. Moreover, users can display and view charts regarding their expenses. What is more, though this interface users get feedback regarding their budget at the end of each week.	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	29, 30, 31
Add Expense	Though this interface authorized users are allowed to add their expenses. Amount, category, payment method and date of expense are required. Users can also keep record of more details for their expenses. This is achieved by adding and filling sections dynamically. Sections that are allowed to be added include location, store, time and comments of an expense.	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	22, 31
My Expenses	Though this interface authorized users are allowed to view the basic details of each one of their expenses (date, time, category, amount, and payment method). For each one of the displayed expenses a link is provided in order to be able users to view more details and update their expenses. Additionally, users are able to filter and sort their expenses and also view a summary of the displayed expenses. Users are also	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Expense Details</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	23, 24, 25, 26, 31



	able to delete expenses at their will.		
Expense Details	Though this interface authorized users are allowed to view details and update their expenses, one at a time.	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Expense Details</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	27, 28, 31
Budget & Goals	Though this interface authorized users are allowed to set and update their weekly budget. They are also allowed to add, update and delete weekly goals if they wish to. Additionally, the can view their goals and sort them based on the category, or the amount.	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	14, 15, 16, 17, 18, 19, 20, 21, 31
Categories	Though this interface authorized users are view the categories of the expenses. Additionally, users can add their own categories, update them of delete them	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	10, 11, 12, 13, 31
Profile	Though this interface authorized users are allowed to view and update the information of their account	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	8, 9, 31
Contact	Though this interface authorized users are allowed to send contact emails to the administrator of the app, to make suggestions and give feedback, ask questions or request help	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	6, 31

Terms of Use	Though this interface authorized users are allowed to display and read the terms of use of this system	<ul style="list-style-type: none"> <li>• Index/Home</li> <li>• Add Expense</li> <li>• My Expenses</li> <li>• Budget &amp; Goals</li> <li>• Categories</li> <li>• Profile</li> <li>• Contact</li> <li>• Terms of Use</li> </ul>	7, 31
--------------	--	--	----------

In the diagrams below the use cases that are described in Table 12.

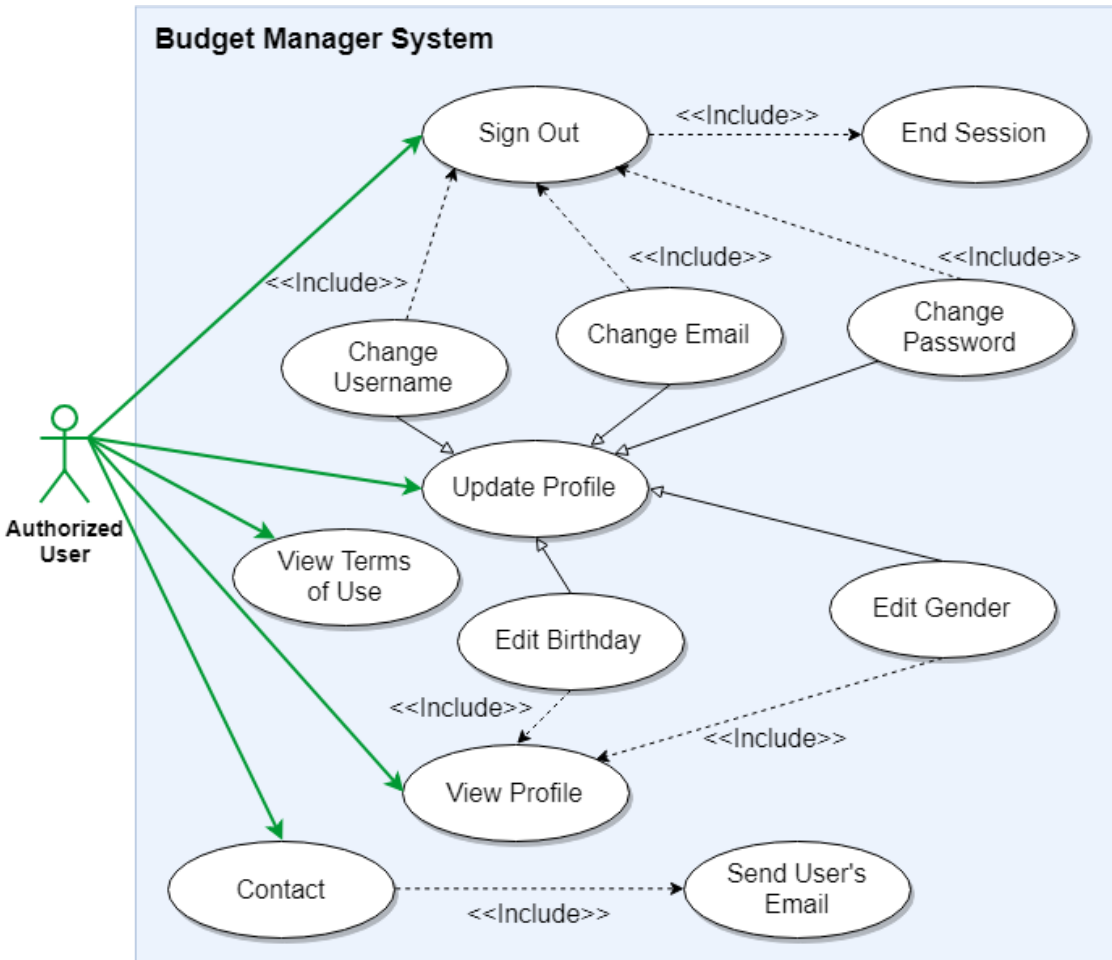


Figure 37: Use cases of interaction between the system and authorized users, part 1

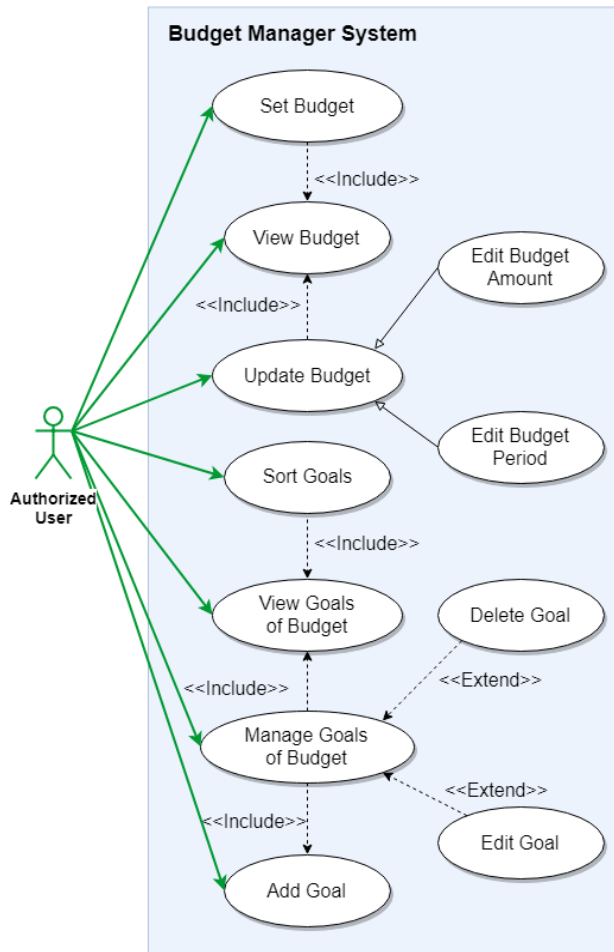


Figure 38: Use cases of interaction between the system and authorized users, part 2

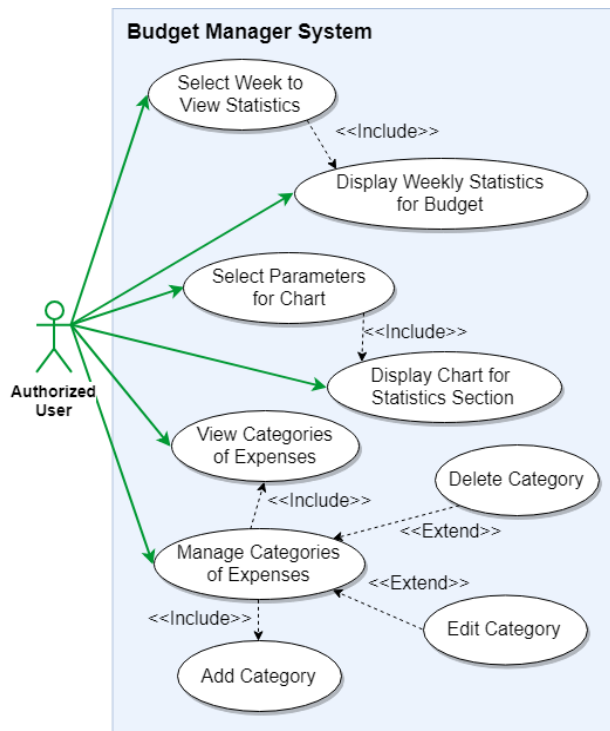


Figure 39: Use cases of interaction between the system and authorized users, part 3

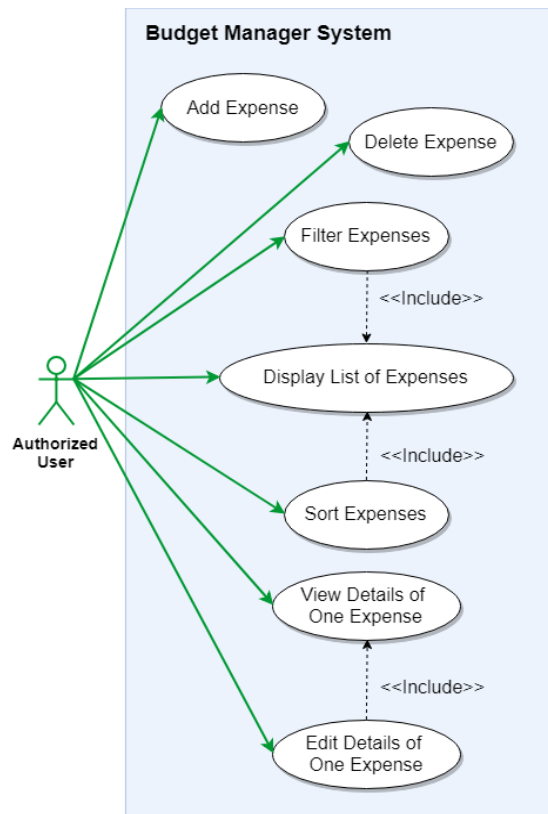


Figure 40: Use cases of interaction between the system and authorized users, part 4

Concluding from all the above, each interface had to be implemented separately, since there are different concerns and distinct functionality for each one of them. However, there are three interfaces that have to implement the required functionality for both authorized and unauthorized users:

- 1) "Index": As described above, interfaces Index/Home and Index/Sign In constitute the starting page for authorized and unauthorized users respectively. Aiming in having one universal starting interface the required functionality for both interfaces was implemented in the "Index" interface, which now facilitates the functional requirements FR id 3, 29, 30, 31.
- 2) "Contact": This interface has to provide the appropriate navigation for each type of user, and must also have the user's email predefined in the form when a user is authorized.
- 3) "Terms of Use": This interface has to provide the appropriate navigation for both authorized and unauthorized users.

Of course, all interfaces have to correctly satisfy the functional requirement FR32 ("Inform User About System's state"), as long as all non-functional requirements.

# 5 Implementation of the System

This chapter describes how the prototype system was developed. The key concepts and all web technologies and tools that were employed are presented below.

## 5.1 Implementation Overview

The prototype system is a mobile-friendly web application that was developed especially for the needs of this research. It was developed from scratch by writing thousands of lines of code and therefore this section focuses on the key concepts, practices, and functions.

A variety of modern web languages and technologies, tools, libraries, and frameworks was utilized. The front-end (client-side) layer of the software was developed in HTML5, CSS3, and JavaScript, whilst the back-end (server-side) layer of the software was developed in SQL and PHP programming languages. Figure 41 illustrates the percentage of the code written in each programming language in order to successfully implement the system.

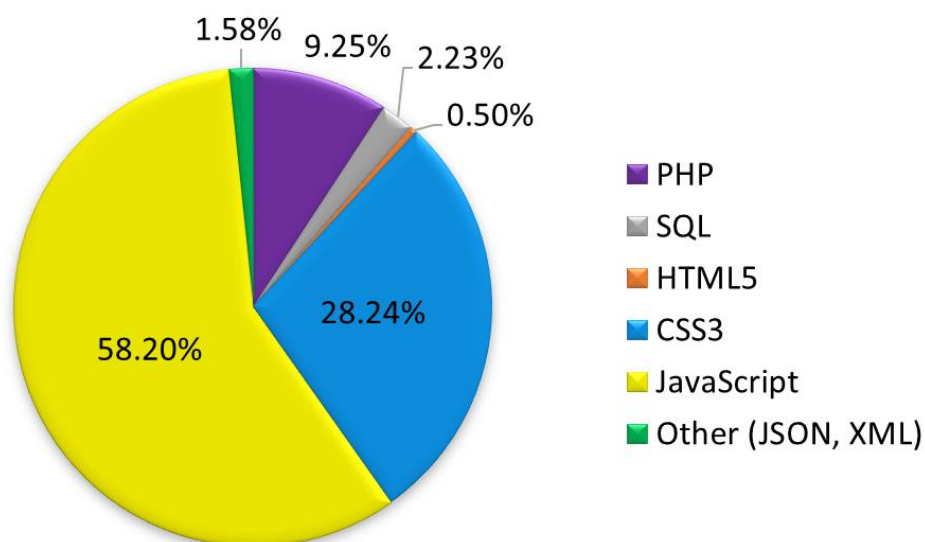


Figure 41: Percentage of programming languages used

The goal was to create a code that would be compliant to the software development cycle without sacrificing the quality of the system. Therefore, the need to create a clean,

readable and maintainable code emerged. To achieve the desired goal, the code was organized in multiple folders and files with allowed maximum length of 500 lines for each file. That way, separation of functionality and concerns was achieved, implementation was carried out step by step, debugging became a much easier task, and finally the ultimate goal of readable and maintainable code was accomplished. The source code of the “Budget Manager” prototype system is available online on github.com: [https://github.com/katerina-tziala/budget\\_manager\\_ems\\_app](https://github.com/katerina-tziala/budget_manager_ems_app)

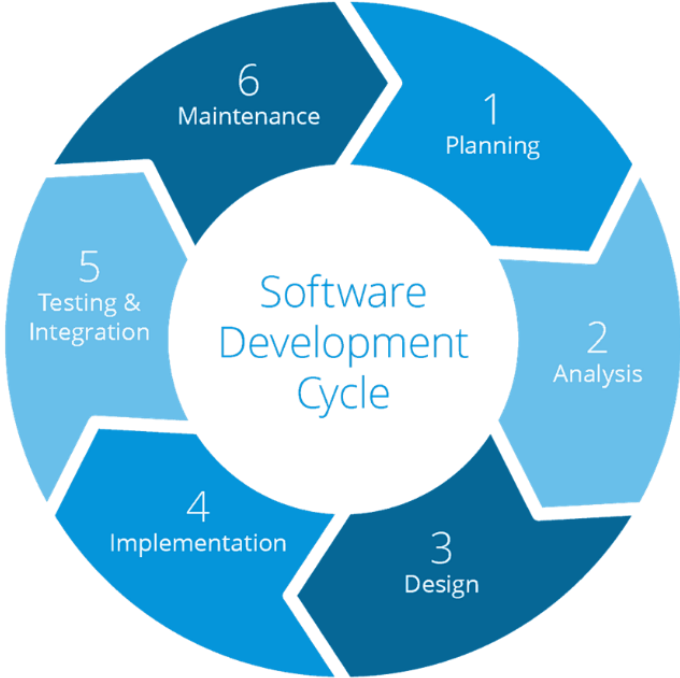


Figure 42: The Software Development Cycle  
(Source: <https://images.mendix.com/wp-content/uploads/Artboard-1@2x-701x700.png>)

The system was developed using the Atom IDE (Integrated Development Environment) and the XAMPP stack. The Atom editor was selected mainly due to its applicability across all operating systems and the customized features it offers. Additionally, this editor includes smart auto-completion, outline view of the code, reference finder and allows the user/programmer to split the interface into multiple panes in order compare and edit code across files. All the above enabled a fast code-writing and code-monitoring.

The XAMPP open source package is one of the most popular PHP development environments, since it is completely free, easy to install and use. It is a powerful tool that allows website designers and programmers to test their work on their own computers without any access to the Internet.

The XAMPP abbreviation is a recursive acronym that derives from the development stack it describes:

X an ideographic letter referring to cross-platform

A Apache HTTP Server

M MariaDB, formerly MySQL

P PHP

P PERL

This stack provided both the required application and data servers, simulating how the system would operate when online.

## 5.2 Back-end development

The back-end (server-side) of the system includes the business logic and the data access layers as depicted in Figure 7. The required database and its tables were developed with SQL. The following figures, (Figure 43, Figure 44, and Figure 45) are some examples that illustrate how SQL was utilized for the development of the system.

```
76 --
77 -- Table structure for table `expense`
78 --
79 CREATE TABLE `expense` (
80   `id` int(11) NOT NULL,
81   `user_id` int(11) NOT NULL,
82   `amount` float NOT NULL,
83   `category` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
84   `payment` enum('cash','credit card','debit card','prepaid card','gift card',
85     'bank transfer','check','mobile payment','web payment')
86     COLLATE utf8_unicode_ci NOT NULL DEFAULT 'cash',
87   `expense_date` date NOT NULL,
88   `expense_time` varchar(5) COLLATE utf8_unicode_ci NOT NULL,
89   `location` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
90   `store` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
91   `comments` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
92   `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
93 ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
94 -- -----
```

Figure 43: Creating table *expense* with SQL

```

357 --
358 -- Constraints for table `expense`
359 --
360 ALTER TABLE `expense`
361     ADD CONSTRAINT `fk_expenseUserId` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`);
362 --
363 -- Constraints for table `feedback`
364 --
365 ALTER TABLE `feedback`
366     ADD CONSTRAINT `fk_feedbackBudgetId` FOREIGN KEY (`budget_id`) REFERENCES `budget` (`id`),
367     ADD CONSTRAINT `fk_feedbackUserId` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`);

```

Figure 44: Adding constraints for tables *expense* and *feedback*

```

51 --
52 -- Dumping data for table `category`
53 --
54 INSERT INTO `category` (`id`, `category_name`, `added_by`) VALUES
55 (1, 'bar & café', NULL),
56 (2, 'bills & fees', NULL),
57 (3, 'clothing', NULL),
58 (4, 'communication', NULL),
59 (5, 'cosmetics & beauty', NULL),
60 (6, 'donations & charity', NULL),
61 (7, 'education', NULL),
62 (8, 'entertainment', NULL),
63 (9, 'gifts', NULL),
64 (10, 'health', NULL),
65 (11, 'housing', NULL),
66 (12, 'investments', NULL),
67 (13, 'restaurant & delivery', NULL),
68 (14, 'sports & fitness', NULL),
69 (15, 'supermarket', NULL),
70 (16, 'technology', NULL),
71 (17, 'transportation', NULL),
72 (18, 'traveling & vacation', NULL),
73 (19, 'vehicle', NULL),
74 (20, 'miscellaneous', NULL);
75 -- -----

```

Figure 45: Inserting predefined categories of expenses

The MySQL open source relational database management system (RDBMS) was used to develop the database for the current research. The interface that was required to interact with this database was developed by combining PHP and MySQLi. Through this interface, the system executes a series of queries against the database, inserts, updates, deletes and retrieves data.

The PHP programming language was used to ensure that the data that was returned and displayed to a user, corresponded to and was related to that user only. Moreover, data was sanitized before using it in queries or storing them in the database, ensuring in that way that users' input does not contain illegal characters, and data is valid and in the



appropriate form. This added an additional layer of security to the application. For every request that was initiated by a user the system had to return a response, even in cases of failure. Hence, users were informed about the system's state, the results of their actions and the implementation of the functional requirement 32 (“Inform Users About System's State”) was achieved. All responses from the server were returned in JSON format, allowing that way to parse the data appropriately with JavaScript and apply the presentation logic.

As already mentioned, PHP was written both in object oriented programming and procedural programming. The functions used to sanitize data are presented below, along with the function that was deployed to allocate users into the experimental and control groups required for the research that was carried out in the context of this thesis.

```
398  /*
399  * FUNCTIONS TO PREPARE DATA BEFORE SENDING THEM TO THE DATABASE
400  */
401  public function prepareDataString($string){
402      $ready_string = filter_var(trim($string), FILTER_SANITIZE_STRING);
403      return $ready_string;
404  }
405  public function prepareDataDate($string){
406      $ready_date = filter_var(trim(date('Y-m-d',strtotime($string))), FILTER_SANITIZE_STRING);
407      return $ready_date;
408  }
409  public function prepareDataId($incoming_id){
410      $ready_id = (int)str_replace(array(' ', ','), '', $incoming_id);
411      return $ready_id;
412  }
```

Figure 46: Functions to sanitize string, date, and id values

Figure 46 presents the functions that were used to sanitize data that refer to string and date data types. The function to sanitize data that refer to the id of a record of a table is also included in Figure 46. That way, it was ensured that all strings are valid, all date values have the format YYYY-mm-dd (year, month, and date numbers with a dash “-” between them), and the accessed id of a record is always an integer. These functions are located in the User class and are inherited and used by the LoggedUser class too.

```
478  /*
479  * FUNCTIONS TO PREPARE DATA BEFORE SENDING THEM TO THE DATABASE
480  */
481  public function prepareDataCategory($string){
482      $ready_category = strtolower(filter_var(trim(utf8_decode($string)), FILTER_SANITIZE_STRING));
483      return $ready_category;
484  }
485  public function prepareDataFloat($value){
486      $ready_float = floatval(filter_var($value, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_FRACTION));
487      return $ready_float;
488  }
```

Figure 47: Functions to sanitize float and category values

Figure 47 depicts the implemented functions to sanitize float data types and the category of expenses. The category is a string but since the `utf8_unicode_ci` collation was used for the database, the category strings had to be decoded. This ensured that strings that contain special characters such as the “é” character of the category “bar & café” are stored appropriately in the database. These functions are located in `LoggedUser` class.

In Figure 48 the function that was deployed to allocate users into the experimental and control groups is presented.

```
159 //function to set feedback for users:
160 private function setFeedback($gender){
161     $query_a_params = array('table' => 'user',
162         'column' => 'feedback',
163         'where' => "feedback=1 AND verified=1 AND gender='".$gender."'");
164     $query_b_params = array('table' => 'user',
165         'column' => 'feedback',
166         'where' => "feedback=0 AND verified=1 AND gender='".$gender."'");
167     $feedback = $this->db->countColumn($query_a_params);
168     $nofeedback = $this->db->countColumn($query_b_params);
169     $addfeedback;
170     if($feedback===$nofeedback){//same number of users with feedback and without feedback:
171         $addfeedback=(rand(0,1));
172     } elseif ($feedback > $nofeedback) {//if we have more users with feedback then we add a user without feedback:
173         $addfeedback=0;
174     } else {//if we have more users without feedback then we add a user with feedback:
175         $addfeedback=1;
176     }
177     return $addfeedback;
178 }
```

Figure 48: Functions to allocate users into the experimental and control groups

The *setFeedback* function returns the appropriate value that has to be stored in the feedback field of the user table for each user. For users that belong in the control group the value 0 is returned, whilst for the users in the experimental group the value 1. This function is executed when the account of a user is activated/verified and updates the feedback value. Based on the gender of a user, the *setFeedback* function executes two queries against the database and retrieves the number of users that belong in a group. For example, if the user is a man, this function counts how many men are in the control and in the experimental group respectively. If the men in the experimental group are more than the men in the control group then the user is allocated in the control group, otherwise the user is allocated in the experimental group. In case both groups have the same number of men users, the new user is allocated randomly in one of these groups. The same applies for women.

A brief description of the way the developed classes were accessed, and how data were returned from the server-side to the client-side layer of the system follows. As

already mentioned, all classes are accessed via the dataManager.php file. When a request is send to this file, a new session is initiated. At first, the code checks if a cookie exists. This means that a user has already accessed his/her account and has requested from the system to “remember” him/her the next time the system is accessed. If a cookie exists, then the session is set to refer to the user specified in the cookie as illustrated in Figure 49.

```

1  <?php
2  session_start();
3  $return_data = [];
4  if(isset($_COOKIE['bm_ems_user']) && !empty($_COOKIE['bm_ems_user']))){
5      $_SESSION['bm_ems_user'] = $_COOKIE['bm_ems_user'];
6  }

```

Figure 49: Setting session parameters based on cookie existence

Then, it is checked if a user was signed into the system. If so, an instance of the LoggedUser class is initiated. Then, the system checks if an action was requested. If this action belongs in the allowed actions of the user then it checks the type of the request. If the request is send to update or store data in the database, the incoming input is decoded, since it is transferred in JSON format. Then, the appropriate action/function of the LoggedUser class is executed and the results of the action are returned. In case the requested action does not belong in the allowed actions, the system returns a message of an invalid request. All the aforementioned are presented in Figure 50 below.

```

7  if(isset($_SESSION['bm_ems_user']) && !empty($_SESSION['bm_ems_user']))){
8      $allowed_actions = ["getPersonalInfo",
9          "updateGenderBirthday","updateUsername","updatePassword", "updateEmail",
10         "getUserBudgetList","setBudget","updateBudgetAmount", "updateBudgetPeriod",
11         "getUserGoalList","addBudgetGoal","deleteBudgetGoal", "updateBudgetGoal",
12         "getUserCategories","addNewCategory", "deleteUserCategory", "editUserCategory",
13         "getUserExpenseList", "addUserExpense", "deleteUserExpense","updateUserExpense",
14         "getFeedbackList", "saveFeedbackDisplay","signUserOut", "sendContactEmail"];
15     require_once("classes/LoggedUser.php");
16     $loggedUser = new LoggedUser($_SESSION['bm_ems_user']);
17     if (isset($_GET['action']) && !empty($_GET['action'])) {
18         $required_action = $_GET['action'];
19         if($_SERVER['REQUEST_METHOD'] === 'POST' && in_array($required_action, $allowed_actions)){
20             $input_data = json_decode(file_get_contents('php://input'), true);
21             $return_data = $loggedUser->{$required_action}($input_data);
22         }elseif ($_SERVER['REQUEST_METHOD'] === 'GET' && in_array($required_action, $allowed_actions)){
23             $return_data = $loggedUser->{$required_action}();
24         }elseif (!in_array($required_action, $allowed_actions)) {
25             $results = array("message" => "invalid_request", 'target' => "no_target");
26             $return_data = $results;
27         }
28     }
29 }else {

```

Figure 50: Accessing the LoggedUser class and executing its functions

In case that a user is not logged into the system, an instance of the User class is initialized. The system checks if an action was requested. If this action belongs in the allowed actions of the user then the incoming input is decoded, since it is transferred in JSON format. Afterwards, the appropriate action/function of the User class is executed and the results of the action are returned. In case were the requested action does not belong in the allowed actions, the system returns a message of an invalid request. An exception is the request to get the personal information of a user. In that case the system returns some dummy data that indicate that the user is not logged into the system. Finally, the returned data are encoded and echoed in JSON format. All the aforementioned are illustrated in Figure 51 below.

```
29 } else {
30     require_once("classes/User.php");
31     $allowed_actions = ["signUp", "activateAccount", "signIn", "sendContactEmail",
32     "forgotPasswordRequest", "resetPassword"];
33     $basicUser = new User();
34     if (isset($_GET['action']) && !empty($_GET['action'])) {
35         $required_action = $_GET['action'];
36         if (in_array($required_action, $allowed_actions)) {
37             $input_data = json_decode(file_get_contents('php://input'), true);
38             $return_data = $basicUser->{$required_action}($input_data);
39         } else {
40             if ($required_action === 'getPersonalInfo') {//override this action when user is not signed in
41                 $return_data = array('id' => "no_id",
42                 'username' => "no_user",
43                 'email' => "no_email",
44                 'gender' => "no_gender",
45                 'birthdate' => "no_birthdate",
46                 'feedback' => 0,
47                 'signed_in' => 0,
48                 'registration_date' => "no_date",
49                 'has_current_budget' => false);
50             }else{
51                 $results = array("message" => "invalid_request", 'target' => "no_target");
52                 $return_data = $results;
53             }
54         }
55     }
56 }
57 echo json_encode($return_data);
58 ?>
```

Figure 51: Accessing the User class and executing its functions

Figures 52 - 55 are some examples of the responses of the server for some of the requests.

```

▼[{id: 1787, amount: 5, category: "bar & café", payment: "cash", expense_date: "2018-11-13",...},...]
  ▼0: {id: 1787, amount: 5, category: "bar & café", payment: "cash", expense_date: "2018-11-13",...}
      amount: 5
      category: "bar & café"
      comments: ""
      expense_date: "2018-11-13"
      expense_time: "09:38"
      id: 1787
      location: ""
      payment: "cash"
      store: ""
  ▶1: {id: 1313, amount: 12, category: "transportation", payment: "cash", expense_date: "2018-11-07",...}
  ▶2: {id: 1248, amount: 18, category: "transportation", payment: "cash", expense_date: "2018-11-04",...}
  ▶3: {id: 1247, amount: 16.5, category: "restaurant & delivery", payment: "prepaid card",...}
  ▶4: {id: 1246, amount: 15, category: "transportation", payment: "cash", expense_date: "2018-11-03",...}
  ▶5: {id: 1245, amount: 29, category: "bar & café", payment: "prepaid card", expense_date: "2018-11-03",...}
  ▶6: {id: 1233, amount: 4.8, category: "bar & café", payment: "cash", expense_date: "2018-11-03",...}
  ▶7: {id: 1232, amount: 5.8, category: "bar & café", payment: "cash", expense_date: "2018-11-03",...}
  ▶8: {id: 1231, amount: 20, category: "transportation", payment: "cash", expense_date: "2018-11-03",...}
  ▶9: {id: 1230, amount: 12, category: "bar & café", payment: "prepaid card", expense_date: "2018-11-03",...}
  ▶10: {id: 1094, amount: 8.6, category: "bar & café", payment: "prepaid card", expense_date: "2018-11-02",...}
  ▶11: {id: 1076, amount: 15.6, category: "traveling & vacation", payment: "cash", expense_date: "2018-11-01",...}
  ▶12: {id: 1059, amount: 10, category: "cosmetics & beauty", payment: "cash", expense_date: "2018-11-01",...}
  ▶13: {id: 1031, amount: 2.5, category: "bar & café", payment: "cash", expense_date: "2018-10-31",...}
  ▶14: {id: 1030, amount: 4, category: "miscellaneous", payment: "cash", expense_date: "2018-10-31",...}
  ▶15: {id: 1029, amount: 7.2, category: "cigars", payment: "cash", expense_date: "2018-10-31",...}
  ▶16: {id: 978, amount: 2, category: "miscellaneous", payment: "cash", expense_date: "2018-10-30",...}
  ▶17: {id: 948, amount: 10, category: "cigars", payment: "cash", expense_date: "2018-10-29",...}
  ▶18: {id: 824, amount: 5.6, category: "supermarket", payment: "cash", expense_date: "2018-10-27",...}
  ▶19: {id: 823, amount: 8.3, category: "restaurant & delivery", payment: "cash", expense_date: "2018-10-26",...}
  ▶20: {id: 740, amount: 13.6, category: "traveling & vacation", payment: "cash", expense_date: "2018-10-25",...}

```

Figure 52: Expenses of a user returned in JSON format from the server

```

▼{message: "success", target: "expense"}
  message: "success"
  target: "expense"

```

Figure 53: Response returned in JSON format from the server after successfully adding an expense

```

▼{id: 4, username: "kate_tzi", email: "katerina.tziala@gmail.com", gender: "female",...}
  birthdate: "1990-08-06"
  email: "katerina.tziala@gmail.com"
  feedback: 0
  gender: "female"
  has_current_budget: true
  id: 4
  registration_date: "2018-10-16 02:26:00"
  signed_in: 1
  username: "kate_tzi"

```

Figure 54: Personal information returned in JSON format from the server for a user that is signed into the system

```
▼{id: "no_id", username: "no_user", email: "no_email", gender: "no_gender", birthdate: "no_birthdate",...}
  birthdate: "no_birthdate"
  email: "no_email"
  feedback: 0
  gender: "no_gender"
  has_current_budget: false
  id: "no_id"
  registration_date: "no_date"
  signed_in: 0
  username: "no_user"
```

Figure 55: Personal information returned in JSON format from the server for a user that is not signed into the system

## 5.3 Front-end development

The front-end (client-side) of the system includes the presentation logic layer as depicted in Figure 7. It was developed with HTML5 (the fifth major version of HTML), CSS3 (the third major version of CSS), and JavaScript programming languages. These languages constitute the standard web technologies that are used to specify the content (HTML), the presentation (CSS), and the behavior (JavaScript) of web pages and applications (Flanagan, 2011).

### 5.3.1 Web Development practices

In order to develop and implement the front-end layer of the system the following web practices were followed ("Best Practices for Speeding Up Your Web Site - Yahoo Developer Network", n.d; "Front-end performance for web designers and front-end developers", 2013 ; "12 Techniques of Website Speed Optimization: Performance Testing and Improvement Practices", 2018):

#### Minimization of HTTP Requests

HTTP (Hypertext Transfer Protocol) requests are counted whenever a browser fetches a file, a page, a picture or data from a web server. An HTTP request is a request/response between a client and a host. This means that the more HTTP requests a website (or web application) needs to load, the longer it takes for the web page to go and retrieve them all, increasing thus web page's load time. According to Yahoo ("Best Practices for Speeding Up Your Web Site - Yahoo Developer Network", n.d.), 80% of the loading time of a web page is spent in downloading the different files that compose the page. One of the most significant web practices is to bundle and combine all JavaScript files that are required for a web page into a larger file (same for CSS).

## **Minification of Resources**

Minification refers to the process of removing unnecessary or redundant code lines from HTML, JavaScript, and CSS files (unused code, white space characters, new line characters, comments and block delimiters), elimination of the code formatting, and use of shorter variable and function names. Minification of a file does not affect the functionality of the file nor the way that the resource is processed by the browser but rather simply optimizes it for downloading purposes. This speeds up load times as it reduces the amount of code, and as a result the size of a file, that has to be requested from the server.

## **Image Size Optimization**

Even though photos and images improve engagement and convey more information, their disadvantage is that they are usually large files that slow down a website. Images often account for most of the downloaded bytes on a page. Compressing image sizes without compromising their quality while ensuring that images are in a correct size and format and correspond to their usage is a vital aspect in reducing load time. Image optimization can often yield some of the largest byte savings and performance improvements: the fewer bytes the browser has to download, the less competition there is for the client's bandwidth and the faster the browser can download and render content on the screen.

## **Use of external files**

Inline JavaScript and CSS in HTML documents are considered as a bad practice in web development. This reduces the number of HTTP requests that are needed, but increases the size of the HTML document that is downloaded every time the HTML document is requested. On the other hand, the use of external JavaScript and CSS files generally produces faster pages because the size of the HTML document is reduced without increasing the number of HTTP requests, since JavaScript and CSS files are cached by the browser.

## **Positioning Appropriately Links to External Files**

Links to external JavaScript and CSS files should be appropriately placed inside HTML files. Specifically, CSS code must be loaded inside the <head> tags. Adding links of stylesheets in the HEAD section of an HTML file allows the page to render progressively. When CSS is loaded at the top, the page renders with the correct layout, colors, structure etc. as desired. Users see this nice structure progressively, since

elements such as the header, the navigation bar, the logo at the top, etc. are rendered appropriately, as the page is loading. The rendered elements serve as visual feedback for the user who is waiting for the rest of the HTML to be rendered, images to be downloaded etc. This improves the overall user experience.

On the contrary, loading CSS code outside of the HEAD section prevents Web browsers from displaying CSS content immediately after downloading it. This happens because browsers block rendering to avoid having to redraw elements of the page if their styles change. Therefore, the user is stuck viewing a blank white page, which degrades user experience.

On the other hand, links to external JavaScript code must be placed at the bottom of an HTML document inside the <body> tags. That way it is ensured that JavaScript code is loaded after the whole content of the web page has been displayed. Hence, users are able to view and assess a web page before they make a decision about what to do next. It is better to stare at a static page than a blank white screen (“Front-end performance for web designers and front-end developers”, 2013). By the time they have made a decision about what they want to do next, or while they are reading/viewing some content on the page, the JavaScript will have been loaded. This is a crucial aspect in optimizing performance of a website (or web app) since JavaScript scripts block parallel downloads; that is, when a script is downloading, the browser will not start any other downloads.

### **Post-load Components**

Another good web practice is to ensure that a page initially loads what is absolutely required. This means that the page initially loads the crucial structure, presentation and behavior of a page (CSS, HTML, JavaScript), whilst additional files are loaded afterwards, enhancing progressively user experience. When a page works properly, it is enhanced with some post-loaded scripts that provide the additional interaction and user experience. Such scripts are loaded dynamically through JavaScript.

### **Use of prefetching techniques**

Prefetching entails reading and executing instructions before a user initiates them. Prefetching is rather common and works well if/when a system anticipates user’s actions. From the plethora of the existing prefetching techniques, “dns-prefetch” and “pre-connect” were selected for the “Budget Manager” prototype system. These



techniques provide browsers with some “hints” in order to start and execute prefetching work.

Specifically, “dns-prefetch” notifies the client (i.e. the browser) that there are assets and resources that will be needed later on from a particular URL and the browser can start the DNS resolution as quickly as possible. Once the browser completes parsing the document, it starts with the DNS resolution. When further requests for resources are initiated by the system, the requested resources load slightly faster, as there is no need to wait for the DNS lookup. Using a “dns-prefetch” can save a lot of time with redirects and on mobile devices where internet speed might be slower. As Roberts argued (2013) “dns-prefetch” dictates supportive browsers to start prefetching the DNS for the specified domain a fraction before it's actually needed. This means that the DNS lookup process will already be underway by the time the browser requests a resource.

For a browser, actions like DNS resolution, TCP (Transmission Control Protocol) handshake and TLS (Transport layer security) negotiations are costly actions. Modern browsers attempt their best to envision what associations a site will require before the initial request is made. However, they cannot foresee dependably all the future connections for each and every website.

Fortunately, with the “pre-connect” technique developers can enable the browser to anticipate what connections and which sockets will be needed before initiating the actual requests. Much like the “dns-prefetch” method, not only does “pre-connect” resolve the DNS, but also makes the TCP handshake, and optional TLS negotiation. “Pre-connect” allows the browser to set up early connections before an HTTP request is actually sent to the server, thus initiating beforehand all connections required and setting up the necessary sockets ahead of time . This saves time for users by eliminating round-trip latency for the aforementioned connections (Grigorik, 2015).

### **5.3.2 Development Orientation**

As already stated, one of the goals of this thesis was to develop a mobile friendly web-based interface that will help users track their personal expenses and stay on budget. In order to define the orientation of the system, the terms “mobile friendly” and “web-based” from the sentence above were the key In order to create simultaneously a mobile friendly and web based system, the “budget manager” was developed as a Progressive Web App (PWA).



Figure 56: PWA logo

(Source: <https://responsivedesign.is/wp-content/uploads/2018/08/PWA-Progressive-Web-App-Logo-800x460.png>)

The term “Progressive Web Apps”, PWAs, was coined by the designer Frances Berriman and the Google Chrome engineer Alex Russell in 2015 (Russell, 2015), in order to describe web applications that take advantage of new features that are supported by most modern browsers while offering the benefits of native mobile apps. As denoted by Google Developers (“Progressive Web Apps”, n.d.), PWAs are user experiences that have the reach of the web. To explain further, PWAs deliver an app-like experience via a mobile phone’s browser but have the same flexibility and gestures as a native application. Because PWAs are powered by mobile browsers, they cross-platform compatibility rather than being targeted to a specific operation system such as Android or iOS. Berriman and Russel also state that PWAs are reliable (i.e. load instantly, regardless of the network state), fast (i.e. respond quickly to user interactions with silky smooth animations and no janky scrolling), and engaging (i.e. feel like a natural app on the device offering an immersive full screen experience, while they are also installable and live on the user's home screen).

The aforementioned are the most significant advantages of PWAs, providing a new level of quality for the developed app. Gazdecki (2017), advocated also in favor of PWAs stating that they are the future of apps by utilizing the latest technology and creating a balance between mobile apps and web pages. What is more, Google developer LePage (2018) stated that a PWA combines the best of the web and the best of apps since it is:

- Progressive: works for every user, regardless of browser choice,
- Responsive: displays seamlessly and identically on all devices, including desktop, mobile, tablet, or whatever comes next,

- Connectivity independent: operates online, offline, and even in low-quality or uncertain network conditions,
- App-like: it feels like a native app and provides app-style navigations and gestures,
- Fresh: is always up-to-date since it utilizes an update process that allows to launch updates and fixes immediately,
- Safe: is launched in a secure environment, since it is served via a secure connection (HTTPS), preventing thus snooping and ensuring that content hasn't been tampered with, and guaranteeing users' information protection,
- Discoverable: allows search engines to find it, since it is identifiable as an "application",
- Re-engageable: facilitates re-engagement through state-of-the-art features like push notifications,
- Installable: it can be added in the home screen of a mobile phone without having to download it from an app store, and
- Linkable: it can be shared and launched immediately, straight from the application's URL and does not require complex installation.

Focusing on the development of the prototype system as a PWA, the “Baseline Progressive Web App Checklist”, provided by Google Developers (2018c), and the code lab titled “Your First Progressive Web App”, provided by LePage (2018), were utilized. In the following sections, the implementation of the “Budget Manager” PWA is presented.

### **5.3.3 App Shell Implementation**

The “App Shell”, or application shell, concept refers to the crucial HTML, CSS, JavaScript and images required to power the minimal user interface of a PWA. The app shell is loaded from the network on users' first visit, and is immediately cached (i.e. saved to the local device) ensuring instant, reliably good performance to users on repeat visits. This means that the shell files are not loaded from the network in every subsequent time that the user opens the app, but are loaded from the local device's cache instead. This results in blazing-fast startup times, since the basic UI is loaded from the cache instantly and only new content is requested from the server (if it isn't available in

the cache already). That way, a web application feels like a native app with instant interaction and solid, improved performance while keeping all the benefits of the web.

The app shell is developed in HTML5. The <head> section contains the basic information of the “Budget Manager” app (title and description), the link to the basic stylesheet that specifies the presentation of the app, and the manifest of the app (described later). Additionally, it contains all necessary supporting resources and meta-tags required for a PWA app: links to the necessary formats of the app’s logo, the right configurations to increase browser support, and the links to apply the aforementioned prefetching techniques.

The logo of the application was developed with the Adobe Illustrator software, which provided the ability to design and extract all the required images of the logo. Additionally, two open source online tools were utilized, the [realfavicongenerator.net](http://realfavicongenerator.net) and the [iconifier.net](http://iconifier.net), in order to generate all appropriate Android, Apple and Favicon icons.



Figure 57: The logo of the budget manager PWA

The implementation of the head section is illustrated in Figure 58 below.

```
1 <!DOCTYPE html>
2 <html lang='en'>
3 <head>
4 <meta charset='utf-8'>
5 <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6 <meta name='viewport' content='width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=5.0'>
7 <meta name='wap-font-scale' content='yes'>
8 <meta name='description' content='Budget Manager - Expense Monitor System'/>
9 <title>Budget Manager</title>
10 <meta name='application-name' content='Budget Manager'>
11 <meta name='apple-mobile-web-app-title' content='Budget Manager'>
12 <meta name='msapplication-tooltip' content='Budget Manager'>
13 <link rel='manifest' href='manifest.json'>
14 <!--build:css-->
15 <link href='' rel='stylesheet'>
16 <!--endbuild-->
17 <link rel='shortcut icon' type='image/x-icon' href='favicon.ico'>
18 <link rel='icon' type='image/png' sizes='32x32' href='favicon-32x32.png'>
19 <link rel='icon' type='image/png' sizes='192x192' href='assets/img/logo/android-chrome-192x192.png'>
20 <link rel='icon' type='image/png' sizes='16x16' href='favicon-16x16.png'>
21 <link rel='icon' type='image/png' sizes='48x48' href='assets/img/logo/android-chrome-48x48.png'>
22 <link rel='icon' type='image/png' sizes='128x128' href='assets/img/logo/logo128.png'>
23 <link rel='mask-icon' size='any' href='assets/img/logo/safari-pinned-tab.svg' color='#009933'>
24 <link rel='apple-touch-startup-image' sizes='256x256' href='assets/img/logo/logo256.png'>
25 <link rel='apple-touch-icon' sizes='180x180' href='assets/img/logo/apple-touch-icon.png'>
26 <link rel='apple-touch-icon' sizes='57x57' href='assets/img/logo/apple-touch-icon-57x57.png'>
27 <link rel='apple-touch-icon' sizes='72x72' href='assets/img/logo/apple-touch-icon-72x72.png'>
28 <link rel='apple-touch-icon' sizes='76x76' href='assets/img/logo/apple-touch-icon-76x76.png'>
29 <link rel='apple-touch-icon' sizes='120x120' href='assets/img/logo/apple-touch-icon-120x120.png'>
30 <link rel='apple-touch-icon' sizes='152x152' href='assets/img/logo/apple-touch-icon-152x152.png'>
31 <meta name='theme-color' content='#0d0d0d'>
32 <meta name='mobile-web-app-capable' content='yes'>
33 <meta name='apple-mobile-web-app-capable' content='yes'>
34 <meta name='apple-mobile-web-app-status-bar-style' content='#0d0d0d'>
35 <meta name='msapplication-navbutton-color' content='#0d0d0d'>
36 <meta name='msapplication-tap-highlight' content='no'>
37 <meta name='msapplication-TileColor' content='#f5f5f5'>
38 <meta name='msapplication-TileImage' content='mstile-144x144.png'>
39 <meta name='msapplication-config' content='browserconfig.xml'>
40 <meta name='full-screen' content='yes'>
41 <meta name='imagemode' content='force'>
42 <meta name='browsermode' content='application'>
43 <meta name='nightmode' content='disable'>
44 <meta name='layoutmode' content='fitscreen'>
45 <meta name='x5-fullscreen' content='true'>
46 <meta name='x5-page-mode' content='app'>
47 <meta name='renderer' content='webkit|ie-comp|ie-stand'>
48 <!--build:dns-->
49 <link rel='dns-prefetch' href=''>
50 <!--endbuild-->
51 <!--build:preconnect-->
52 <link rel='preconnect' href=''>
53 <!--endbuild-->
54 <!--build:base-->
55 <base href=''>
56 <!--endbuild-->
57 <!--build:starturl-->
58 <meta name='msapplication-starturl' content=''>
59 <!--endbuild-->
60 </head>
```

Figure 58: Implementation of the head section

As depicted in Figure 58 above, there are tags inside the `<head>` section that are located inside the comments `<!--build:[name of build]-->` and `<!--endbuild-->`. These comments are employed for development purposes in order to dynamically add the appropriate links for the development and distribution of the app utilizing the Gulp workflow. The `<body>` section contains the basic HTML elements of all interfaces, is illustrated in the Figure 59.

```

61 <body>
62 <div id='opacity_layer' class='layer hidden'></div>
63 <header id='header'>
64 <h1 id='app_title'>budget manager</h1>
65 <button type='button' id='btn_refresh' class='round_btn header_btn fas fa-sync-alt' aria-label='refresh app' onclick='refreshApp(event)'>
66 </header>
67 <div id='content_wrapper'>
68 <main id='main'></main>
69 <div id='loader_container' class='centeredFlexbox'>
70 <div id='loader' class='loader'>
71 <div id='loader_image' class='loader_image hidden'></div>
72 <svg viewBox='0 0 32 32' width='32' height='32'>
73 <circle id='spinner' class='spinner spin' cx='16' cy='16' r='14' fill='none'></circle>
74 </svg>
75 </div>
76 </div>
77 <footer id='footer'>
78 <span>Copyright &copy; 2018</span><a href='index.html'>Budget Manager</a><span>All Rights Reserved</span>
79 <i>powered by A.Tzialis</i><a href='terms_of_use.html'>Terms of Use</a>
80 </footer>
81 </div>
82 <div id='toaster' class='toaster'>
83 <i id='notification_bell' class='fas fa-bell'></i><p id='toast_msg'></p>
84 <div id='notification_btns'></div>
85 </div>
86 <script src='https://cdnjs.cloudflare.com/ajax/libs/babel-polyfill/7.0.0/polyfill.min.js' type='application/javascript'></script>
87 <!--build:js -->
88 <script src=''></script>
89 <!-- endbuild -->
90 </body>
91 </html>

```

Figure 59: Implementation of the body section

In Figure 59 above, the key components of the body section are presented:

- the header of the app with the title and the refresh button
- the main section of the interface, where the content is added dynamically
- a loading indicator
- the footer of the app
- a template for notifications
- and the ‘opacity\_layer’ element which is added only for display purposes

Note that the `<main>` tag, the loader and the footer are wrapped inside a div element (i.e. the “content\_wrapper”), in order to properly display the interface. Finally, the links that load the basic JavaScript files are placed at the bottom of the `<body>` tag. The first link loads a JavaScript polyfill (i.e. a piece of code used to provide modern functionality on older browsers that do not natively support it) from the CDNJS library repository

(CDNJS is a Content Delivery Network as the initials CDN indicate) which is hosted on cloudflare.com. The second link loads the crucial JavaScript of the app and is located inside the comments `<!--build:js-->` and `<!--endbuild-->`. These comments are employed in order to add the appropriate link of the bundled and minified files that compose the required functionality utilizing the Gulp workflow. The final result of the app shell is depicted in Figure 60.

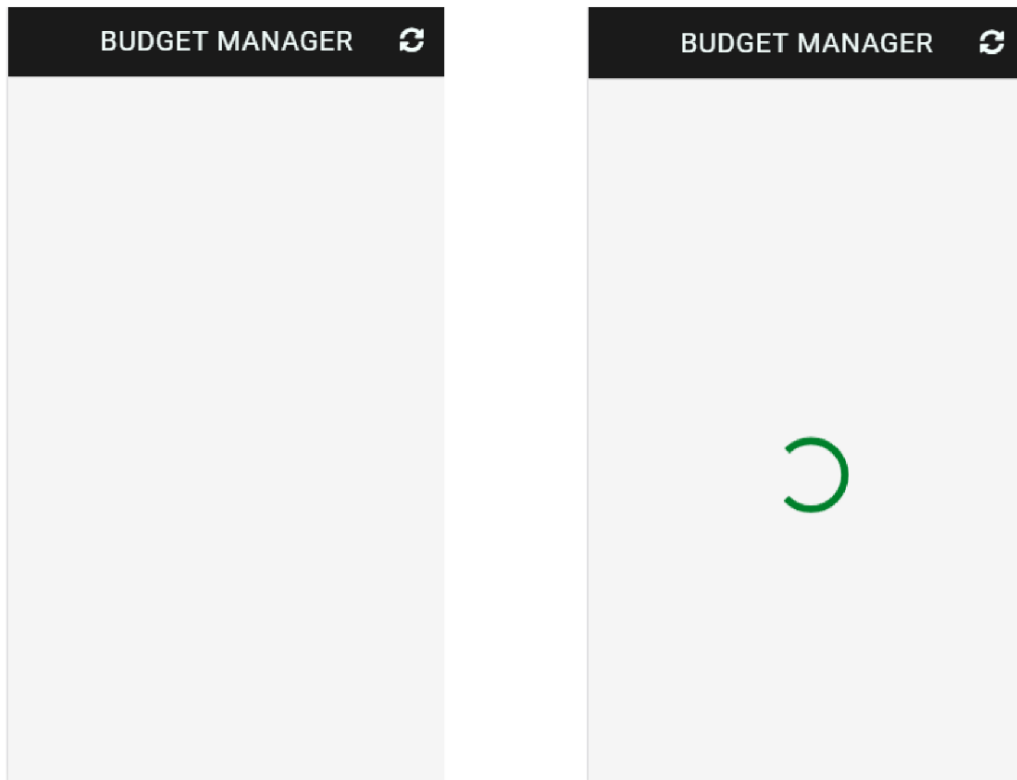


Figure 60: The app shell of the system

The app shell is similar to the bundle of code that is published to an app store when building a native app. It contains the core and necessary components to get an app off the ground, but does not contain any data or meaningful content. The app shell is not doing anything interesting. It is just the minimal skeleton of the app with a spinner that verifies the web server functionality.

Noteworthy is that this is the structure of all interfaces. In comply with the “Baseline Progressive Web App Checklist” each individual page/interface has to be deep linkable via unique URLs for the purpose of shareability and accessibility (i.e. it can be opened and directly accessed via new browser windows). Rather than copying and renaming the app shell HTML file, all HTML files were automatically compiled utilizing the Gulp workflow. Each one of them provides an interface with unique

content, functionality and UI features that are added dynamically with JavaScript as described in the following section.

### 5.3.4 Dynamic Content, UI features and libraries

As already mentioned above, the content, the functionality and the UI features are added dynamically in each interface with JavaScript. Soon after the app shell is loaded the app retrieves the required data and the content of each view is populated and rendered. That way, the UIs are compiled appropriately for both authorized and unauthorized users. Until all required data are fetched and UIs are populated the user sees the app shell with a loading spinner

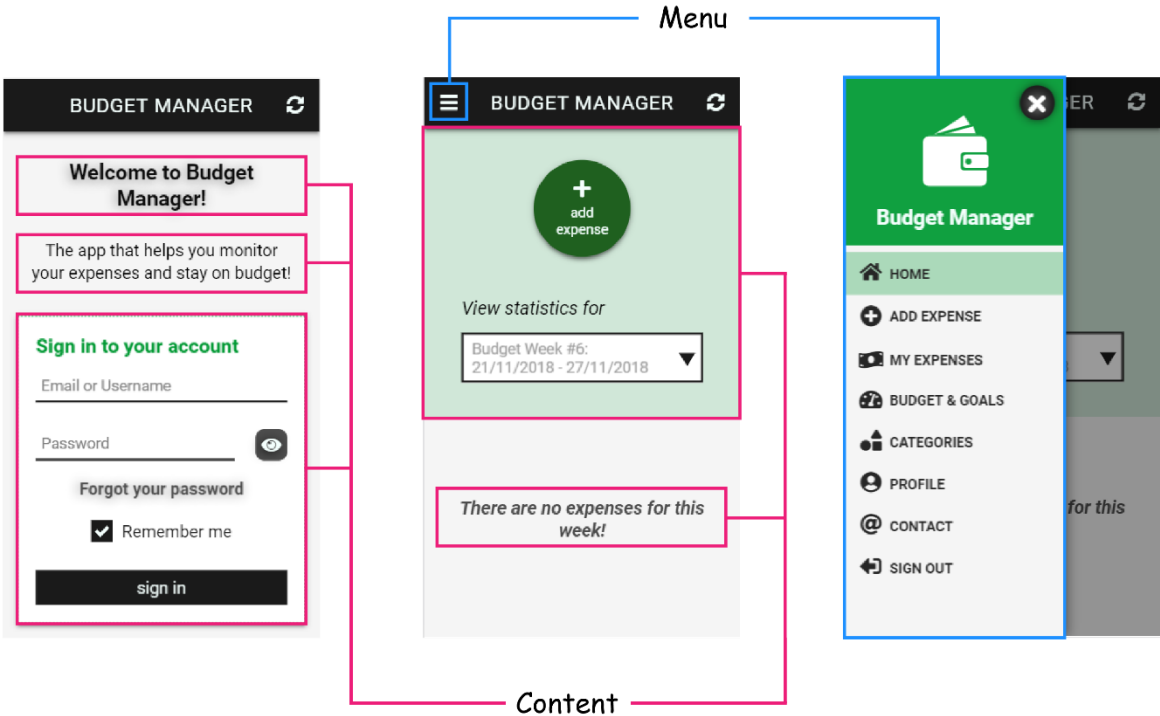


Figure 61: Populating dynamically the content and menu in interfaces

Figure 61 above illustrates how the content of the home page for both authorized and unauthorized users is added. Building upon the app shell, new elements and UI features are added along with all the required functionality. Additional content, functionality and UI features dictated additional CSS and JavaScript code for each interface. This code is loaded dynamically through the basic JavaScript code, which is loaded through the app shell. To achieve this, two functions were implemented: the loadStyle and the loadScript function. As illustrated in the following Figures 62-63, both functions accept two parameters: the path of the file that is required to be loaded



and the callback function that triggers the execution of the next needed function. Adhering to the web practices that were mentioned above, the stylesheets that are loaded are added in the <head> section of the interface, whilst the JavaScript files in the bottom of the <body> section.

```
274 //Load css file
275 const loadStyle = (path, callback)=>{
276   const css = document.createElement("link")
277   css.type = "text/css";
278   css.rel = "stylesheet";
279   css.href = path;
280   if (css.readyState){ //IE
281     css.onreadystatechange = () => {
282       if (css.readyState == "loaded" || css.readyState == "complete"){
283         css.onreadystatechange = null;
284         callback();
285       }
286     };
287   } else { //Others
288     css.onload = () => {
289       callback();
290     };
291   }
292   document.head.appendChild(css);
293 };
```

Figure 62: The loadStyle function – loading dynamically CSS code for interfaces

```
255 //Load javascript file
256 const loadScript = (path, callback)=>{
257   const script = document.createElement("script")
258   script.type = "application/javascript";
259   script.src = path;
260   if (script.readyState){ //IE
261     script.onreadystatechange = () => {
262       if (script.readyState == "loaded" || script.readyState == "complete"){
263         script.onreadystatechange = null;
264         callback();
265       }
266     };
267   } else { //Others
268     script.onload = () => {
269       callback();
270     };
271   }
272   document.body.appendChild(script);
273 };
```

Figure 63: The loadScript function – loading dynamically JavaScript code for interfaces

Aiming in fulfilling the requirements of a PWA it had to be ensured that the additional content and UI features would perfectly fit any form factor: desktop, laptop, mobile, and tablet. This means that all interfaces have to be responsive, enhancing hence user experience by optimizing the display of the app across multiple devices, screen sizes and screen orientations (Figure 64). This goal was achieved with the use of CSS3 media queries and the concept of the Flexbox Layout Module as described in w3schools.com (CSS Media Queries, n.d.; CSS Flexbox, n.d.).

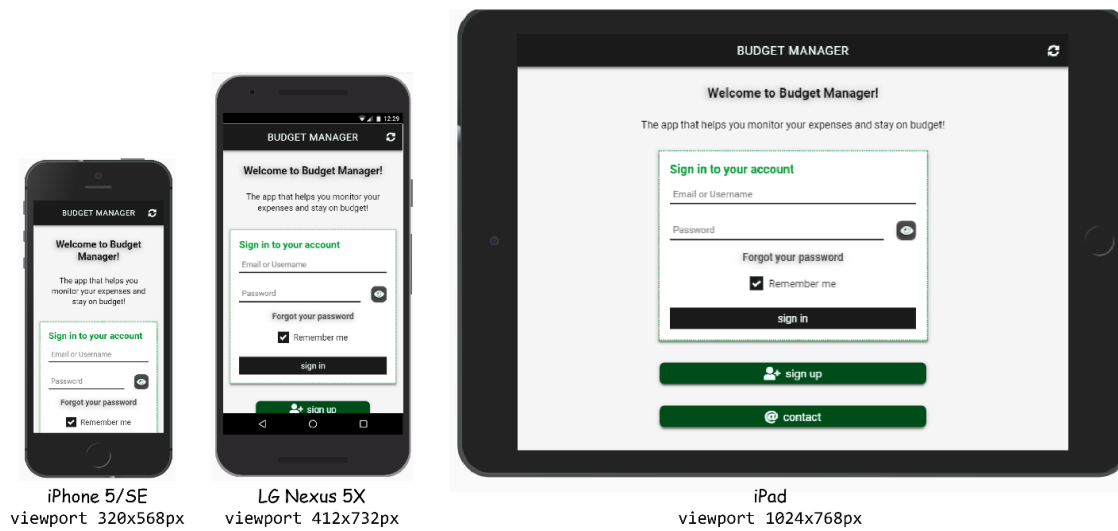


Figure 64: The application on mobile phone and tablet with different viewports and orientations

In order to provide a playful, zestful and accessible user experience, a plethora of graphical elements were implemented: input and text area fields, a variety of buttons encompassing radio buttons and switch buttons, select boxes, pickers and much more. All elements were also defined semantically with the appropriate attributes. For those elements in which a semantic element is not available, appropriate ARIA (Accessible Rich Internet Applications) roles were defined. Additionally, the colors of the elements were selected based on the colors of the logo while focusing on providing the right contrast and color combinations so as to make the app friendly to users with visibility impairments. Furthermore, many functions were implemented in JavaScript, in order to facilitate the appropriate tab navigation, catering thus for users with motor impairments.

### 5.3.5 Fonts, Libraries and Plugins

The user experience was also enhanced with the implementation of specific plugins and libraries.

Starting with the typography, the Google Fonts API was utilized. With the implementation of the open source web fonts the interfaces of the “Budget Manager” PWA became more beautiful and intuitive. Specifically, the “*Roboto*” Google fonts were employed (under the Apache License, Version 2.0), which were developed by Christian Robertson (year). Through the [fonts.google.com](https://fonts.google.com) website the aforementioned font family was selected, customized, downloaded and finally embedded into the app.

Furthermore, the fifth version of the “*Font Awesome*” toolkit was implemented (under the MIT, SIL OLF, and CC BY licenses). “*Font Awesome*” is a robust icon set that contains scalable vector icons (“Basic Use”, n.d.). Icons are important in websites and web apps because they are a visual way to add meaning to elements (Kay, n.d.). Their infographic nature enriches UIs and user experience (Fitzgerald, 2015). The “*Font Awesome*” icon fonts are just fonts, that instead of containing letters or numbers, they contain symbols and glyphs. The momentous advantages of the “*Font Awesome*” icon set, except being free to use, are scalability (i.e. because they are vector graphics, scale up or down without losing their quality), customization (i.e. due to the fact that they are a web font CSS effects can be applied to them, such as changing colors and sizes), fast (i.e. because of their small size they load fast), performance (i.e. instead of multiple HTTP requests that bitmap images require, icon fonts require one single or few HTTP request(s) to load, compatibility (i.e. “*Font Awesome*” icons are supported in all browsers).

Figure 65 below illustrates how the “*Font Awesome*” icon set was utilized to enhance the meaning of the content of the interfaces.

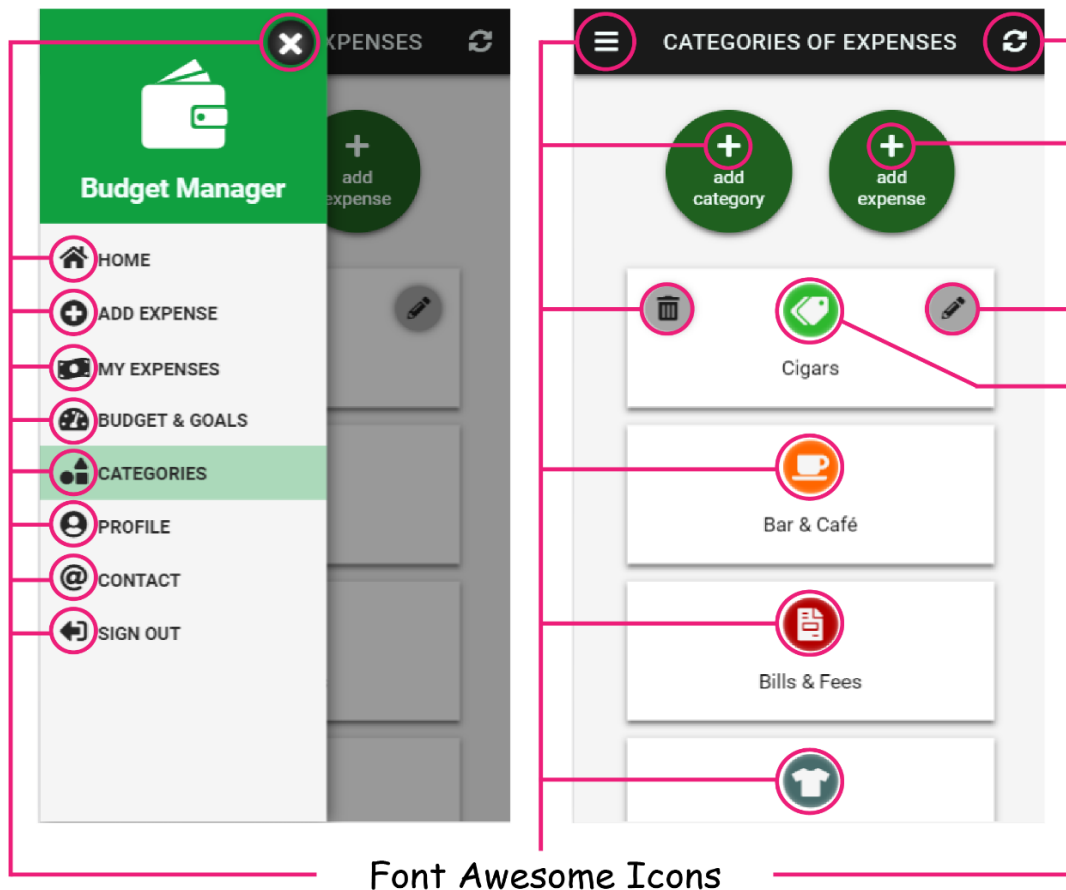


Figure 65: Enhancing the meaning of the content with “Font Awesome” icons

Three more plugins/libraries were utilized to boost user experience:

- blob-select: a dependency-free JavaScript plugin for styling <select> elements with an emphasis on markup simplicity and performance. (under public license, version 2.0).
- md-date-time-picker: a material design picker developed in vanilla CSS, JavaScript, and HTML (licensed under an MIT License). From this plugin only the date picker was utilized.
- grudus timepicker: a material design time-picker written in JavaScript, HTML and CSS, without any external dependencies (licensed under the Apache License 2.0)

These plugins were customized to fit better in the “Budget Manger” PWA. In the following figure is depicted how the aforementioned plugins enriched the user interfaces.

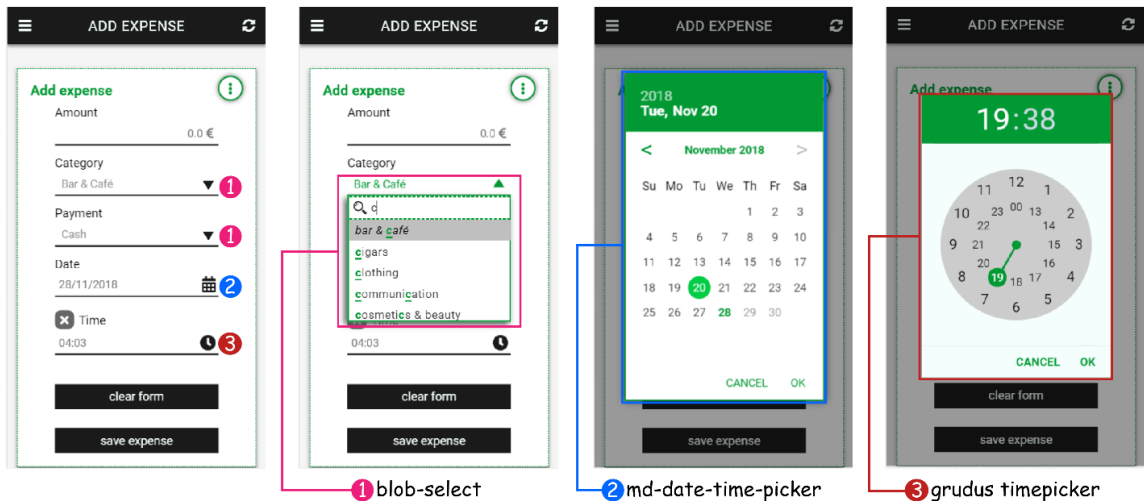


Figure 66: Implementation of blob-select, md-date-time-picker, and grudus timepicker plugins/libraries

Furthermore, the open source community maintained project Chart.js (available under the MIT license) was integrated into the app to allow users visualize the data regarding their expenses (Figure 67). Chart.js utilize the HTML5 <canvas> element and provide great rendering performance across all modern browsers.

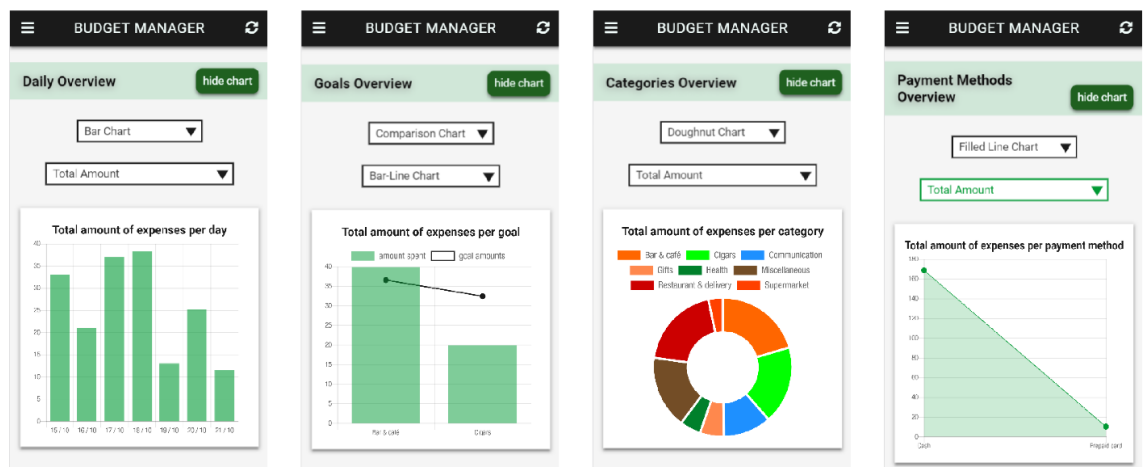


Figure 67: Charts regarding the expenses of a week with the use of Chart.js

Finally, all the desired interaction with the database was achieved with the implementation of the Fetch API (licensed under the Creative Commons Attribution 4.0 International). The Fetch API provides an interface for accessing, fetching and sending resources and data asynchronously across the network (Gaunt, 2018a). In the below figures, Figure 68 and Figure 69, it is presented how the Fetch API was embedded in the app.

```

2 //send data to server
3 const sendData = (action, data) => {
4   const url = data_url+action;
5   const send_data = JSON.stringify(data);
6   const send_promise = sendHandler(url, send_data);
7   return new Promise((resolve, reject) => {
8     send_promise.then(send_data => {
9       resolve(send_data);
10    }).catch((error) => {
11      const retundata = {
12        "message":"app_error",
13        "error":error
14      }
15      return retundata;
16      //reject(error);
17    });
18  });
19 };
20 //function to handle how data are send to the server
21 async function sendHandler(url, data) {
22   const headers = new Headers();
23   headers.append('Content-Type', 'application/json; charset=utf-8');
24   const request = new Request(url, {
25     method: 'POST',
26     mode: 'no-cors',
27     credentials: 'same-origin',
28     headers: headers,
29     body:data});
30   try {
31     const fetchResult = fetch(request);
32     const response = await fetchResult;
33     const jsonData = await response.json();
34     return jsonData;
35   } catch(error){
36     const retundata = {
37       "message":"app_error",
38       "error":error
39     }
40     return retundata;
41     //throw Error(error);
42   }
43 }

```

Figure 68: Implementation of the Fetch API to send data to the server

```

44 //retrieve data from server
45 const fetchData = (action, callback) => {
46     const url = data_url+action;
47     fetchHandler(url).then((response)=>{
48         callback(null, response);
49     }).catch((error) => callback(error, null));
50 };
51 //function to handle how data are fetched from the server
52 async function fetchHandler(URL) {
53     const headers = new Headers();
54     headers.append('Accept', 'application/json; charset=utf-8');
55     const request = new Request(URL, {
56         method: 'GET',
57         mode: 'no-cors',
58         cache: 'reload',
59         headers: headers});
60     try {
61         const fetchResult = fetch(request);
62         const response = await fetchResult;
63         const jsonData = await response.json();
64         return Promise.resolve(jsonData);
65     } catch(error){
66         throw Error(error);
67     }
68 }

```

Figure 69: Implementation of the Fetch API to retrieve data from the server

All libraries and plugins that were used are listed in appendix APIs, Plugins and Libraries.

### 5.3.6 The Web App Manifest

The Web App Manifest is a crucial part of a PWA. It is a simple JSON text file that provides information about a web app (such as its name and description icons it should use, the start\_url it should start at when launched, etc.) (LePage, 2018). The manifest informs the browser about the web app and how it should behave when “installed” on the user's mobile device or desktop. It also controls the screen orientation for optimal viewing. The figures below, Figure 70 and Figure 71, present the manifest that was deployed in the “Budget Manager” app and how it is “read” by the browser.

```
1 {
2   "name": "Budget Manager",
3   "orientation": "portrait-primary",
4   "short_name": "Budget Mngr",
5   "description": "Budget Manager is an app that helps you monitor your expenses and stay on budget!",
6   "display": "standalone",
7   "background_color": "#f5f5f5",
8   "scope": "/",
9   "serviceworker": {
10    "src": "ServiceWorker.js",
11    "scope": "/"
12  },
13  "start_url": "start_url_of_app",
14  "theme_color": "#0d0d0d",
15  "lang": "en-US",
16  "icons": [{
17    "src": "favicon.ico",
18    "sizes": "32x32",
19    "type": "image/png"
20  }],
21  "background_color": "#f5f5f5",
22  "display": "standalone",
23  "orientation": "portrait-primary",
24  "short_name": "Budget Mngr",
25  "start_url": "start_url_of_app",
26  "theme_color": "#0d0d0d"
27 }
```

Figure 70: The manifest.json file

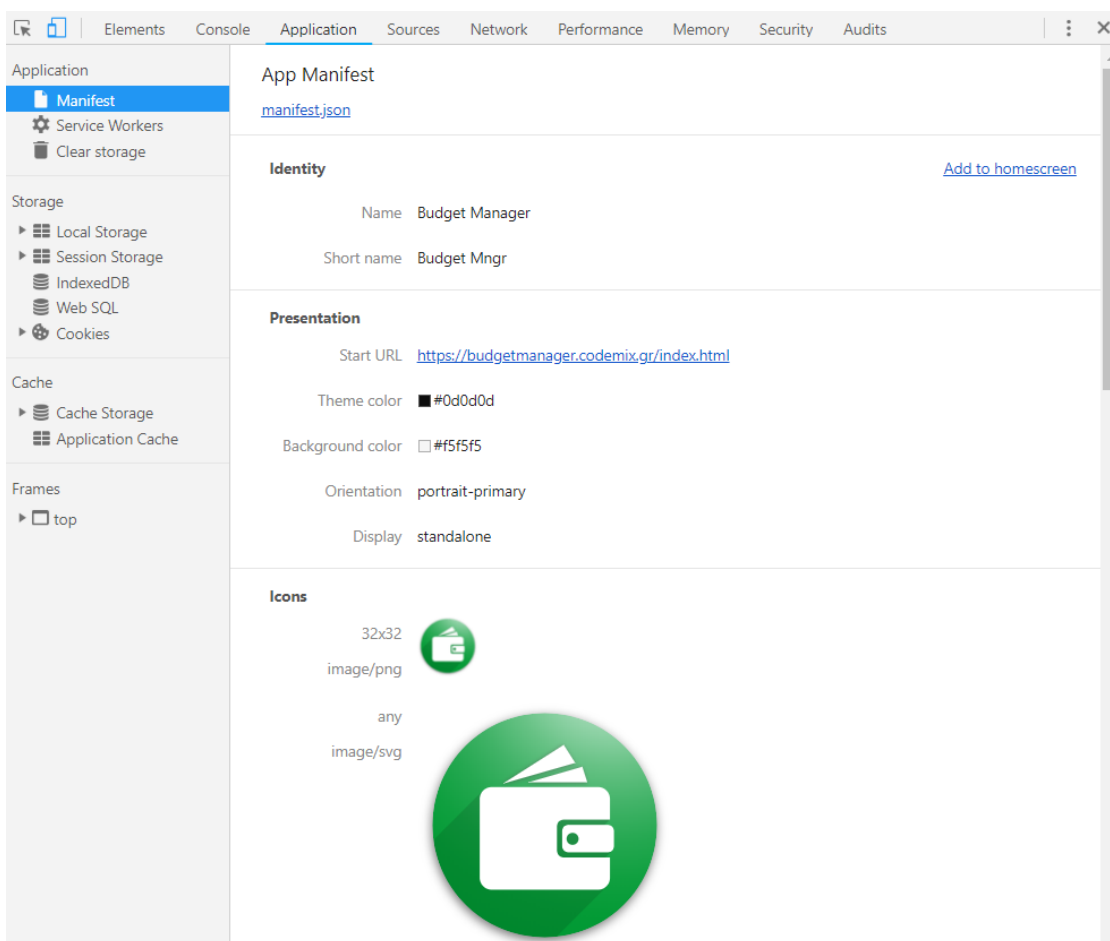


Figure 71: Manifest tab of Chrome DevTools



The Web App Manifest supports native integration with the “Add to Home Screen” feature. With this feature, users can choose to add a shortcut link to their device just as they would install a native app from a store, but with a lot less friction. Thus, users can launch the app from the home screen of their smartphone by simply touching the icon of the app instead of launching a web browser and typing the URL on the keyboard (Gaunt & Kinlan, 2018).

Browsers handle most of the heavy lifting of the app’s installation with the help of web app install banners (meta tags placed on the <head> section of the HTML) and the manifest file that contains all the details about the app. The installation process of the “Budget Manager” PWA is illustrated in Figure 72 below.

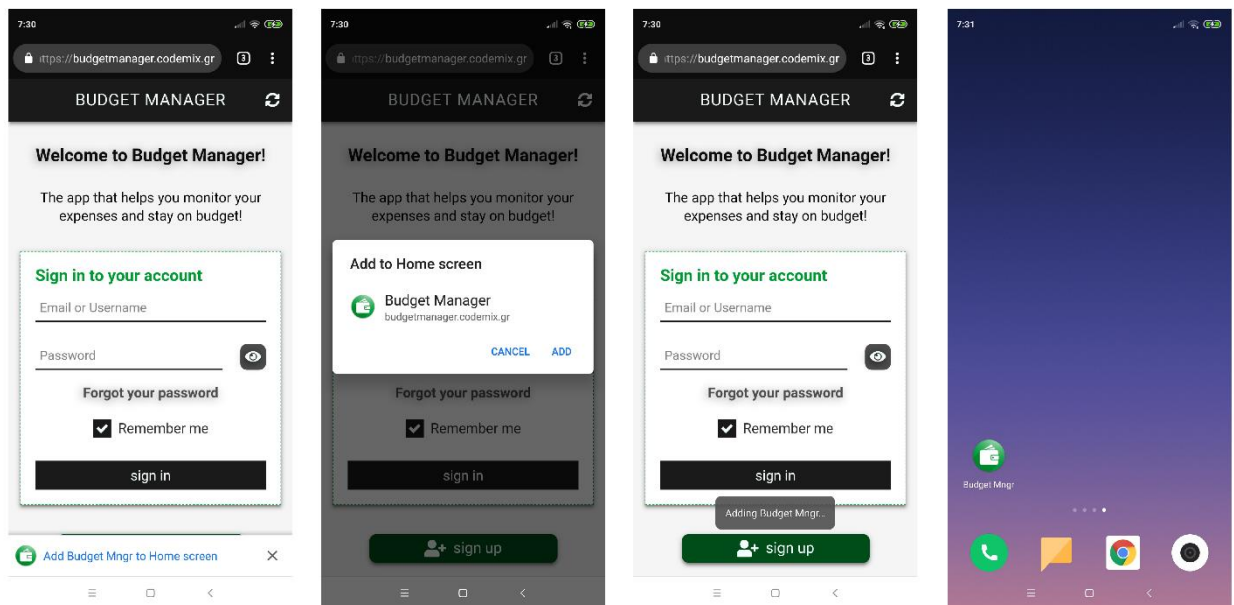


Figure 72: The “Add to Home Screen” feature of the “Budget Manager” PWA

When the app first launches from the home screen of a mobile phone, it can take a moment for the browser to spin up, and the initial content to begin rendering. Instead of displaying to the user a white screen that may look like the app is stall, the browser shows a splash screen, until the first meaningful paint (i.e. the paint after which the biggest above-the-fold layout change has happened, and web fonts have been loaded). The splash screen is created by the browser automatically based on the properties of the app, as specified in the manifest. In the following Figure 73 is depicted the splash screen that is displayed to users (both authorized and unauthorized) when the app is launched from the home screen.

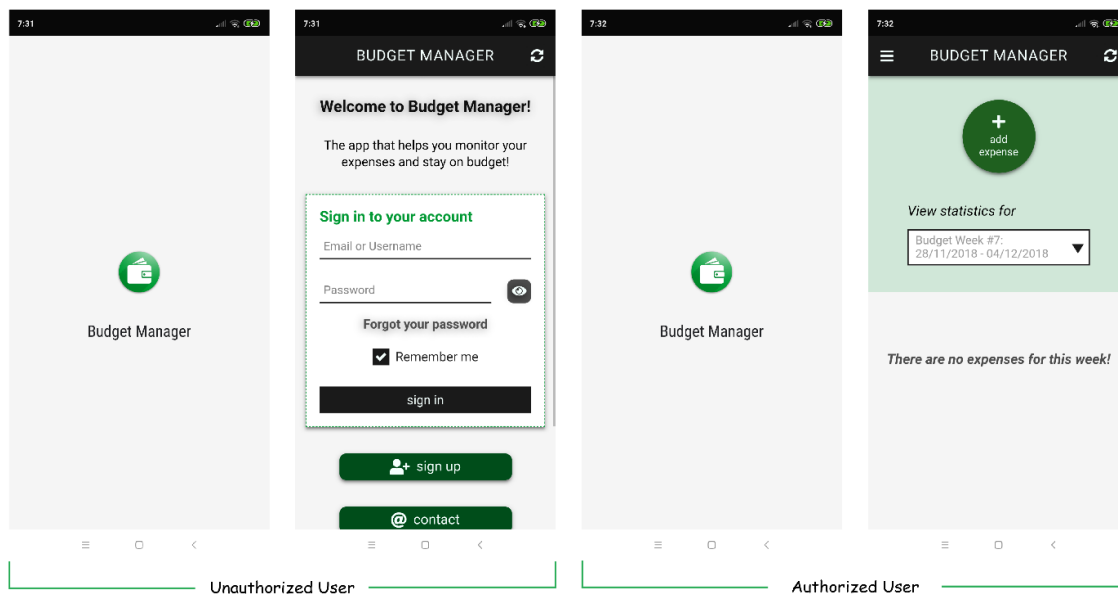


Figure 73: Launching the “Budget Manager” PWA from the home screen

The implementation of the manifest increases engagement since a web app can have in a rich presence on the user's smartphones. It also optimizes user experience since it can be launched in full-screen mode on a smartphone with no URL bar and also provides a “splash screen” launch experience (Gaunt & Kinlan, 2018).

### 5.3.7 Service Worker implementation

Besides being installable, PWAs have to be fast and work online, offline, and on intermittent, slow connections. To ensure that the app is always available quickly and reliably, a *Service Worker* has to be implemented (Figure 74). Features provided via *Service Workers* (such as periodic background syncs and push notifications) are considered a progressive enhancement, and added only if supported by the browser. When *Service Workers* are not supported, the code is not executed, and users get a basic experience. The *Service Worker* runs separately from the main browser thread, intercepting network requests, caching or retrieving resources from the cache, and delivering messages. Caching the resources of the app, makes the content to load faster under most network conditions. The *Service Worker* improves the performance of the app since it controls the network requests and provide offline access to cached content (Archibald, 2018, Gaunt, 2018b; Google Developers, 2018b; Posnick, 2018).

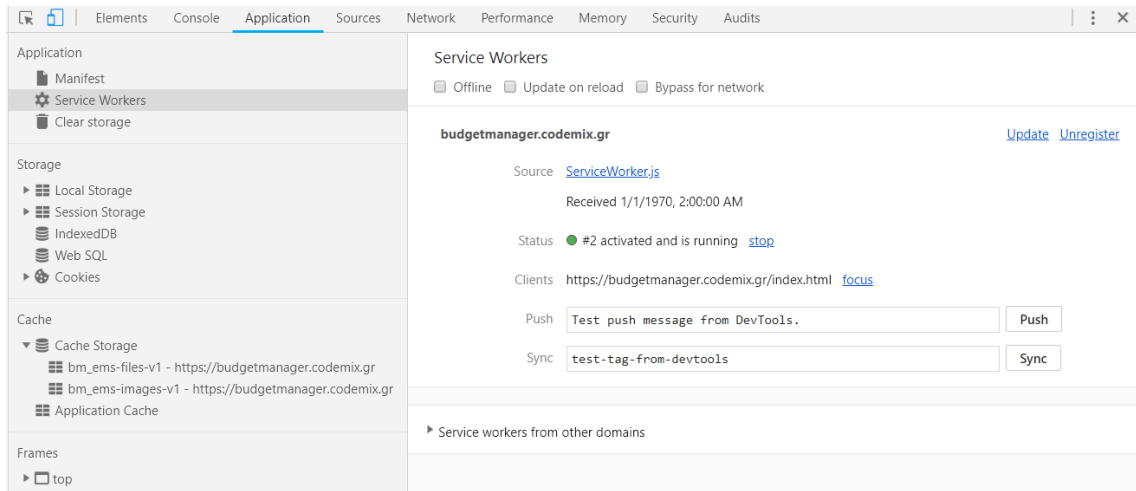


Figure 74: The registered Service Worker as displayed in Chrome DevTools

The workflow of the implemented Service Worker is described briefly below:

At first, the Service Worker caches all assets of the app: the app shell of each interface, all CSS and JavaScript files and all the images of the app (Figures 75-76).

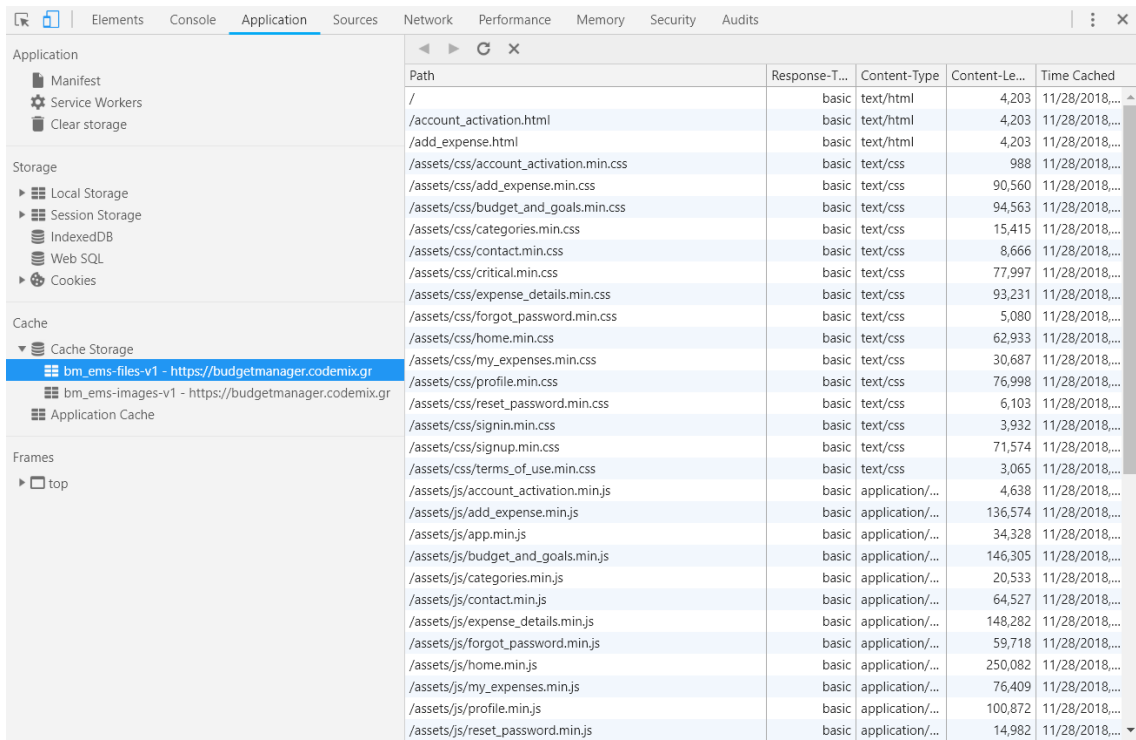


Figure 75: Static files saved in the cache storage of the browser by the Service Worker

Path	Response-T...	Content-Type	Content-Le...	Time Cached
/assets/img/beauty.svg	basic	image/svg+...	1,129	11/28/2018,...
/assets/img/feedback/budget_folder.gif	basic	image/gif	10,718,606	11/28/2018,...
/assets/img/feedback/empty_pockets.gif	basic	image/gif	1,463,404	11/28/2018,...
/assets/img/feedback/empty_wallet.gif	basic	image/gif	501,150	11/28/2018,...
/assets/img/feedback/excited.gif	basic	image/gif	1,013,625	11/28/2018,...
/assets/img/feedback/money_girl.gif	basic	image/gif	4,119,610	11/28/2018,...
/assets/img/feedback/money_rain_one.gif	basic	image/gif	1,697,135	11/28/2018,...
/assets/img/feedback/sad.gif	basic	image/gif	464,662	11/28/2018,...
/assets/img/feedback/toast.gif	basic	image/gif	935,819	11/28/2018,...
/assets/img/female.svg	basic	image/svg+...	3,244	11/28/2018,...
/assets/img/lock_refresh.svg	basic	image/svg+...	1,024	11/28/2018,...
/assets/img/logo/android-chrome-144x144.png	basic	image/png	5,640	11/28/2018,...
/assets/img/logo/android-chrome-192x192.png	basic	image/png	7,615	11/28/2018,...
/assets/img/logo/android-chrome-256x256.png	basic	image/png	10,347	11/28/2018,...
/assets/img/logo/android-chrome-36x36.png	basic	image/png	1,813	11/28/2018,...
/assets/img/logo/android-chrome-384x384.png	basic	image/png	16,249	11/28/2018,...
/assets/img/logo/android-chrome-48x48.png	basic	image/png	2,177	11/28/2018,...
/assets/img/logo/android-chrome-512x512.png	basic	image/png	23,489	11/28/2018,...
/assets/img/logo/android-chrome-72x72.png	basic	image/png	2,985	11/28/2018,...
/assets/img/logo/android-chrome-96x96.png	basic	image/png	3,858	11/28/2018,...
/assets/img/logo/apple-touch-icon-120x120.png	basic	image/png	4,705	11/28/2018,...
/assets/img/logo/apple-touch-icon-152x152.png	basic	image/png	6,119	11/28/2018,...
/assets/img/logo/apple-touch-icon-57x57.png	basic	image/png	2,410	11/28/2018,...
/assets/img/logo/apple-touch-icon-72x72.png	basic	image/png	2,936	11/28/2018,...
/assets/img/logo/apple-touch-icon-76x76.png	basic	image/png	3,121	11/28/2018,...
/assets/img/logo/apple-touch-icon.png	basic	image/png	7,378	11/28/2018,...
/assets/img/logo/logo.svg	basic	image/svg+...	117,877	11/28/2018,...
/assets/img/logo/logo128.png	basic	image/png	11,936	11/28/2018,...
/assets/img/logo/logo256.png	basic	image/png	30,373	11/28/2018,...
/assets/img/logo/logo333333.svg	basic	image/svg+...	819,772	11/28/2018,...

Figure 76: Images saved in the cache storage of the browser by the Service Worker

When a user initiates a request, the *Service Worker* checks if the requested resource is already cached. If so, it returns immediately a response from the cache. Otherwise, the request is sent to the server over the network, its response is served to the user, and is saved in the cache memory of the browser for future use.

In cases where the user is offline the *Service Worker* displays properly the user interface. However, because of time limitations, the functionality of the app was not implemented for offline use. Instead, users get notified when connection is lost and when connection is re-established. Nonetheless, the implementation of the *Service Worker* provides a robust and steady ground to build upon and provide offline functionality of the app in the future.

Last but not least, when an update is released, the *Service Worker* informs the user that a new version of the app is available. The user can choose either to migrate to the new version immediately or keep using the last accessed one.

The difference between a web app that does not register a *Service Worker* and a web app that does, is depicted in Figure 77.

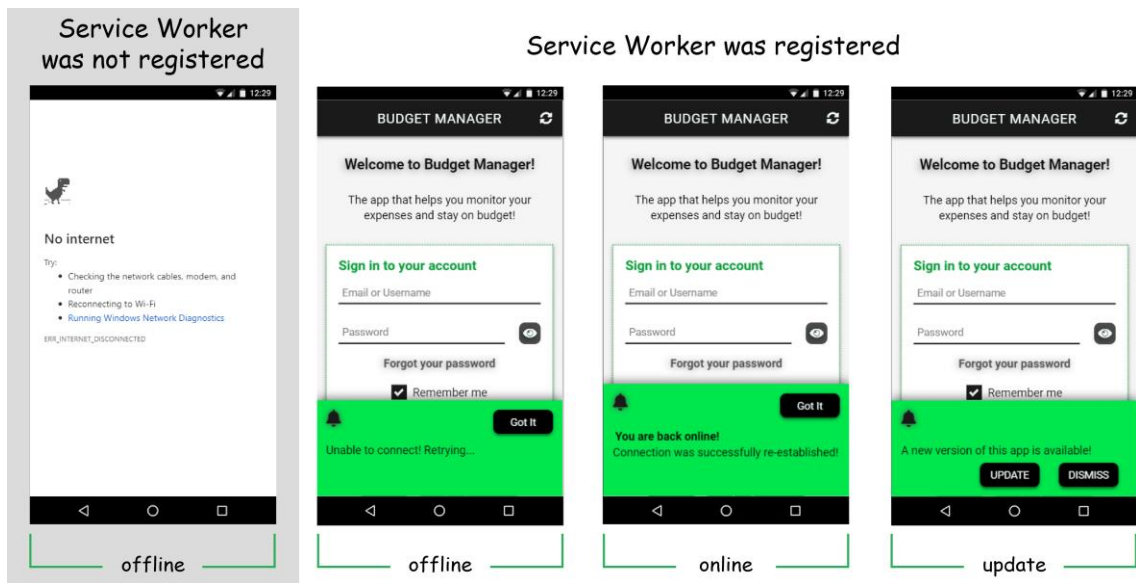


Figure 77: User experience with and without Service Workers

### 5.3.8 Use of the Gulp Build Tool

In modern web development, there are many repetitive tasks such as bundling and minifying code, optimizing assets (CSS, JavaScript, and images), running code analysis and unit tests, copying modified files to an output directory and much more. This problem of repetition is solved with the use of Gulp, a cross-platform build tool. In more detail, Gulp is a JavaScript streaming task runner, that reads files as streams and pipes the streams to different tasks (Google Developers, 2018a). These tasks are code-based and use plugins in order to modify and build the source files into production files. Gulp is extremely powerful, since it allows each developer to create his/her own customized build processes. By simplifying and automating repetitive tasks, Gulp provides web developers with more time to do non repetitive tasks. This, sequentially, leads to increased productivity.

Gulp was utilized in the development of the prototype system (under the MIT License) in order to:

- optimize the resources of the system (JavaScript, CSS and images)
- bundle and minify the resources of the system (JavaScript and CSS)
- build the system for development
- test the system during development

- build the system for distribution

Note that Gulp requires [Node.js](#) and its package manager, [npm](#), which installs the required gulp plugins.

### **Creating the new project**

Before installing gulp plugins and developing the gulp workflow, the application had to be initialized. Within the project's working directory the following command line command was executed:

```
npm init
```

This command initiated the generation of a package.json file, prompting with questions about the application. This file contains all necessary information in order to build the app. All packages that were installed later, were also saved in this file. The package.json file was used to track the project's packages. Tracking packages allows a quick re-installation of all the packages and their dependencies in future builds. By executing the npm install command in the command prompt within the directory of the project all the required packages are automatically installed (the npm install command will read package.json and automatically install everything listed).

All packages installed are listed in Table A1, in appendix Software and Tools.

### **The Gulp workflow**

All gulp tasks are located in the `gulp_tasks` folder and are executed through the `gulpfile.js`. This folder contains also a JSON file (`config.json`) where all information required for both development and production were placed.

When executing the command *gulp* in the command prompt, within the directory of the project, the app is built for development and testing purposes. This command removes all folders that were created previously as a result of the same *gulp* command. After that, the necessary variables in all files are replaced automatically with the appropriate values that are retrieved from the `config.json` file. Then, the appropriate links are added in the app shell, and all required HTML files are compiled. Additionally, the HTML, CSS, and JavaScript code is optimized, bundled and minified. That way, for each interface, there is one HTML, one CSS and one JavaScript file

required to appropriately define the content, the presentation and the functionality of an interface. Finally, all required images were optimized in order to reduce their size.

For example, the home interface requires four CSS files to properly render the interface.

The paths of these files are specified in the config.json file as depicted in Figure 78.

```
35   "css" : {
36     "account_activation" : ["/src/assets/css/account_activation.css"],
37 >   "add_expense" : [
43 >     "budget_and_goals" : [
48 >     "categories" : ["/src/assets/css/common/*.css"],
50 >     "critical" : ["/src/assets/libs/fontawesome/css/all.min.css"],
53 >     "contact" : ["/src/assets/css/common/menu.css"],
56 >     "expense_details" : [
62 >     "forgot_password" : ["/src/assets/css/common/form.css"],
64     "home" : [
65       "/src/assets/libs/blobselect/blobselect.css",
66       "/src/assets/css/common/loggeduser.css",
67       "/src/assets/css/common/menu.css",
68       "/src/assets/css/home/*.css"
69   ],
```

Figure 78: Array of CSS resources of the “Home” interface

These files are/were automatically optimized to provide cross browser compatibility (all necessary prefixes are added with the use of a Gulp plugin), comments and unnecessary code lines are removed, and then they are combined in one minified file. The file contains only the necessary information, its size is 57KB and is retrieved within 4ms approximately. In Figure 79 is illustrated how the browser retrieves the CSS file for the home interface.

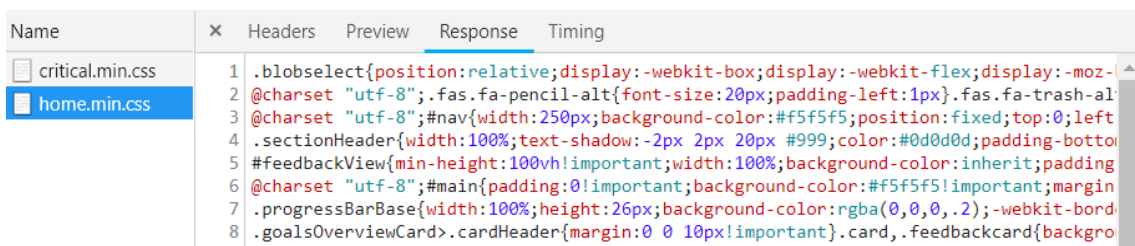


Figure 79: The CSS file required from the browser for the presentation of the “Home” interface

Instead of sending four requests to retrieve all the required stylesheets for the home page, only one request is needed. Similarly, one request is required for the JavaScript file of the home interface.

In order to build the app for distribution, the appropriate variables were specified in the config.json file. Specifically, the “*app\_params*” illustrated in the figure below were updated with the values that were provided by the administrator of the hosting server.

```
1  {
2    "app_params" : {
3 >   "dev" : {
10   "dist" : {
11     "path" : "the_link_of_the_uploaded_app",
12     "db_username" : "your_username_to_access_database_on_server",
13     "db_password" : "your_password_to_access_database_on_server",
14     "db_name" : "your_database_name_on_server",
15     "app_scope" : "scope_of_the_uploaded_app"
16   }
17 },
```

Figure 80: Parameters of the application required for distribution

That way, it was not necessary to replace all hardcoded values manually, nor the need to check them across multiple files. The command that builds the app for production is the *gulp dist* command that creates the “dist” folder. Inside that folder, can be found all files that are required to be uploaded on a server in order to release the app.

### **Building and Releasing the “Budget Manager” PWA**

After executing the *gulp dist* command, the contents of the “dist” folder were uploaded to the hosting server with the use of the FileZilla Client (an open source software which is distributed free of charge under the terms of the GNU General Public License).



# 6 Evaluation and Results

This section presents the evaluation of the developed system with regards to the objectives of this research. In order to evaluate the developed system, statistical analysis was conducted on the gathered data that the users of the system provided. First, the objective of this research is presented, followed by the data collection process and the characteristics of the participants. Then, the statistical analysis that was conducted is presented and the results that emerged are discussed.

## 6.1 Research Objective

The aim of this study was to develop an affective system for monitoring personal expenses, helping the user to stay on budget. Specifically, the focus of this research was twofold. The first part focused on the development of a system that would successfully promote budget adherence. The second part aimed at shedding light into how the use of animated GIFs as a means to provide affective feedback, influenced users' behavior towards their budget. Additionally, the influence of animated GIFs on budget adherence was also studied in relation to gender and in relation to age. Affective feedback was provided to users after the completion of each week of monitoring their expenses.

The main research question that this study aimed to answer can be seen below:  
RQ1: To what extent do animated GIFs as an affective feedback technique influence budget adherence?

It was hypothesized that participants in the group that received affective feedback in the form of GIFs tend to adhere to their budget more successfully compared to the group that did not receive affective feedback through GIFs. The study also focused on examining the influence of animated GIFs on budget adherence in relation to the participants' gender and age. Therefore two more research questions were defined:

RQ2: To what extent do animated GIFs, as an affective feedback technique, in relation to gender influence budget adherence?

RQ3: To what extent do animated GIFs, as an affective feedback technique, in relation to age influence budget adherence?

## **6.2 Data Collection and Participants**

The required data was collected from actual users who used the “Budget Manager” PWA for a three-week period. In order to imitate real life conditions, men and women from different age groups were invited to participate . Inclusion criteria encompassed users with internet access and at least one digital device (e.g. desktop, laptop, tablet, or smartphone) at their possession. Furthermore, users had to be willing to use the system for the requested period. Exclusion criteria encompassed users who were not capable of using a digital device, did not have internet access, and/or did not have the will to participate in the survey.

Potential users were informed via personal emails for the goal of this research and were provided with guidelines regarding the use of the system. Additionally, they were requested to use the system for a period of three weeks. That way, insights regarding the progress of users’ budget adherence in the long-run were obtained. The experimental phase began on 16 October 2018 and completed on 11 November 2018 providing a time margin for late starters. Worth mentioning is the fact that the start and end dates of each user’s weeks can be different, therefore the whole week (seven-day period) was accounted as a reference point.

In total, 141 responders out of the possible 250 created an account in the system with their consent. However, 10 participants were excluded from the analysis; 9 because they did not verified/activated their accounts and 1 due to the fact that the required period was not completed. The final sample, used for the analysis, consisted of 131 participants who were actively involved in the survey.

## **6.3 Analysis**

Gathered data was both qualitative and quantitative, and was analyzed accordingly on SPSS version 23. At first, a preliminary analysis was conducted in order to prepare the data. Descriptive statistics were obtained to provide an overview of the variables and characteristics of the sample. Inferential statistics were used to further analyze the data. Chi-Square and Kruskal-Wallis were utilized to answer the research questions of the current study.

## 6.4 Descriptive Statistics Results

This section provides an overview of the characteristics of the sample and the studied variables. The respective tables (Table A2 – Table A5) and graphs (Figure A1 – Figure A7) can be found in the appendix.

The sample consisted of 70 females (53.44%) and 61 males (46.56%), ranging in age from 16 to 65, with a mean of 31.28 and a standard deviation of 8.24. With regard to age, the 27-37 age group accounted for a majority of 63.36% (N = 83, 40 males and 43 females), followed by 16-26 with 21.37% (N = 28, 10 males and 18 females), 38-48 with 11.45% (N = 15, 8 males and 7 females) and finally 49 years and older with 3.82% (N = 5, 3 males and 2 females).

For the needs of the study, the participants were randomly distributed into two equal groups; 66 (50.38 %) subjects who received affective feedback through animated GIFs (31 males and 35 females), and 65 (49.62%) subjects that did not (30 males and 35 females). Within the 16-26 age group 16 out of 28 participants received affective feedback, and within the 27-37 age group 38 out of 83 participants received affective feedback. As far as elder age groups are concerned, within the 38-48 age group 8 participants received affective feedback and 7 did not, whilst within the  $\geq 49$  age group 3 participants received affective feedback and 2 did not.

Additionally, only 29.55% (N = 39) of the sample had set at least one budget goal over the 3-week period as opposed to 70.45% (N = 93), only one participant defined at least one budget goal in each week, 5 participants in the first two weeks and 37 participants in the first week. Moreover, there were 2 participants who defined a budget goal only for the second week. Thus, the budget goal variable was not studied further. Concluding, it is implied that the participants did not utilize the budget goal feature of the “Budget Manager” PWA.

Table 7: Descriptive statistics of participants who defined at least one budget goal in each week

<b>Descriptive statistics of participants who defined at least one budget goal in each week</b>				
<b>Variables</b>	<b>Categories</b>	<b>N</b>	<b>Percent of participants (N=131)</b>	<b>Percent of participants that had a budget goal (N=39)</b>
Budget goal 1st week	No	94	70.23%	-
	Yes	37	28.24%	94.87%
Budget goal 2nd week	No	124	94.66%	-
	Yes	5	3.82%	12.82%
Budget goal 3rd week	No	130	99.24%	-
	Yes	1	0.76%	2.56%

N: Frequency

f%: Relative frequency

As far as budget adherence is concerned, it seems that there is an improvement over the period of the three weeks. Specifically, in the first week, the rate for the “Success” category was 31.3% (N = 41) and "Fail" 68.7% (N = 90), while in the second week the percentages for "Success" and "Fail" respectively were 52.67% (N = 69) and 47.33% (N = 62). In the third week, the improvement was even greater with the “Success” category accumulating 77.10% (N = 101) and "Fail 22.9%" (N = 30). The results are presented in Figure 81.

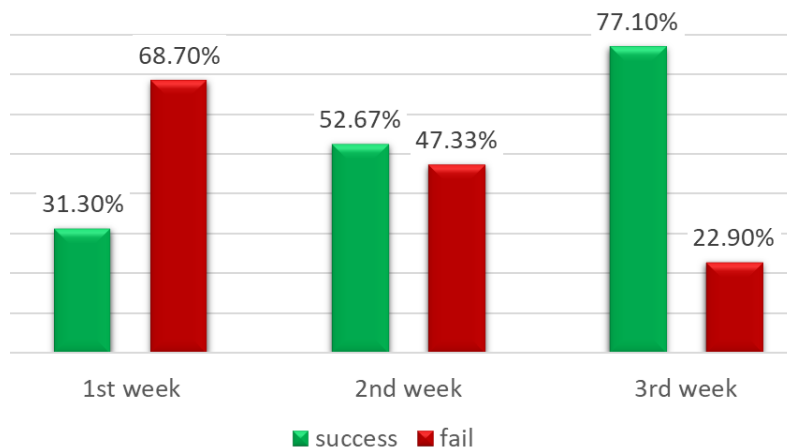


Figure 81: Graph regarding the budget adherence of participants

As depicted in Figure 82 below, budget adherence was improved in both feedback groups over the three week period. As also implied, the participants who received affective feedback were slightly more adherent to their budget than the participants who did not receive affective feedback. The respective statistical tables, tables A6 – A8, can be found in appendix.

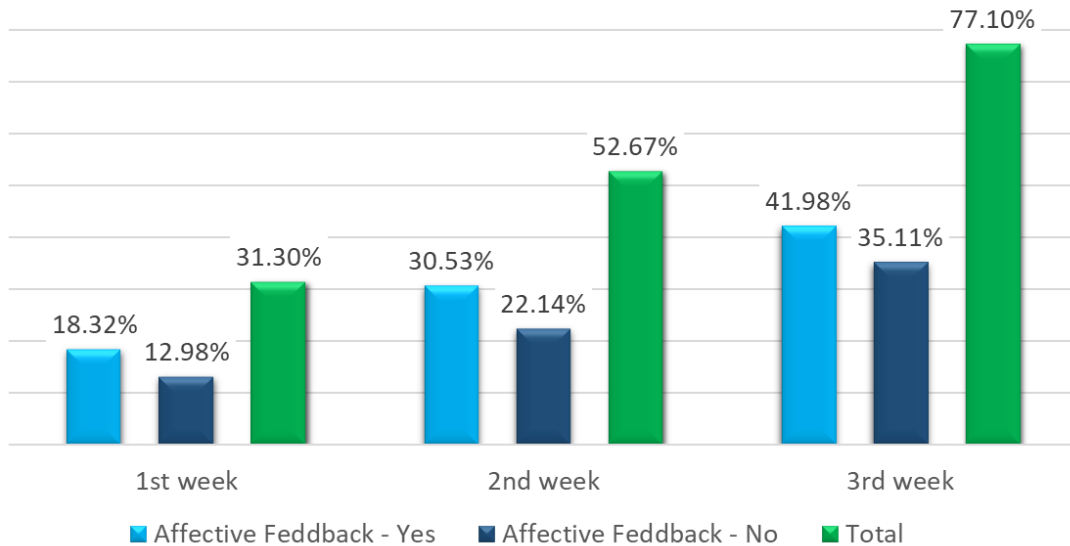


Figure 82: Graph regarding the budget adherence in correlation to affective feedback

The following graphs, Figure 83 – Figure 85, present how the population of the participants, who were successfully adherent to budget, was shaped based on affective feedback – gender and affective feedback - age. All percentages reference to the population who succeeded in budget adherence. The respective statistical tables, tables A9 – A14, can be found in appendix.

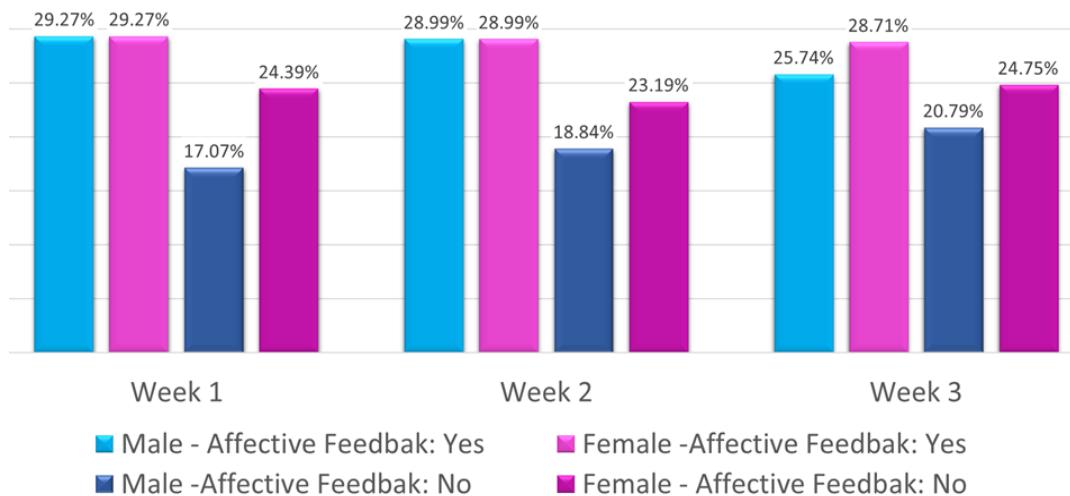


Figure 83: Graph regarding budget adherence in correlation to the feedback groups based on gender

As observed in Figure 84 above, budget adherence among men and women that received affective feedback in comparison with those who did not receive affective feedback, seems to only slightly differ across the weeks. However, since budget adherence was improved for the whole population week after week (31.30%, 52.67%, and 77.10% respectively) the hypothesis that animated GIFs influence budget adherence was not clear and therefore it was further studied with inferential statistics.

Figure 85 and Figure 86, illustrate the age groups of users who managed to stay on budget in correlation to whether they received affective feedback or not, respectively. It seems that in age group 27-37, budget adherence increased week after week in both groups. The participants in age group 16-26 seem to improve their budget adherence when affective feedback was not provided in contrast to the participants in the same age group who received affective feedback. Budget adherence for the participants in the age group 38-48 who received affective feedback increased the second week in comparison to the first, and decreased in the third week in comparison to the second. For the participants in the same age group who did not receive affective feedback, budget adherence decreased the second week in comparison to the first and increased in the third week in comparison to the second. Finally, budget adherence for the participants in the age group  $\geq 49$  decreased the second week in comparison to the first and increased in the third week in comparison to the second irrespective of receiving or not affective feedback.

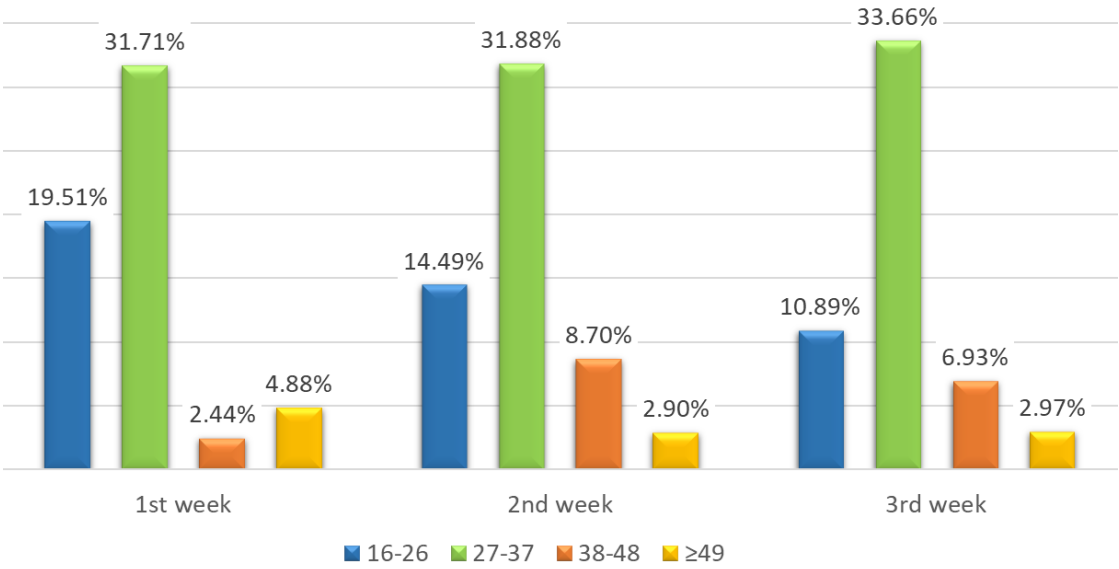


Figure 85: Graph regarding budget adherence in correlation to users who received affective feedback based on age

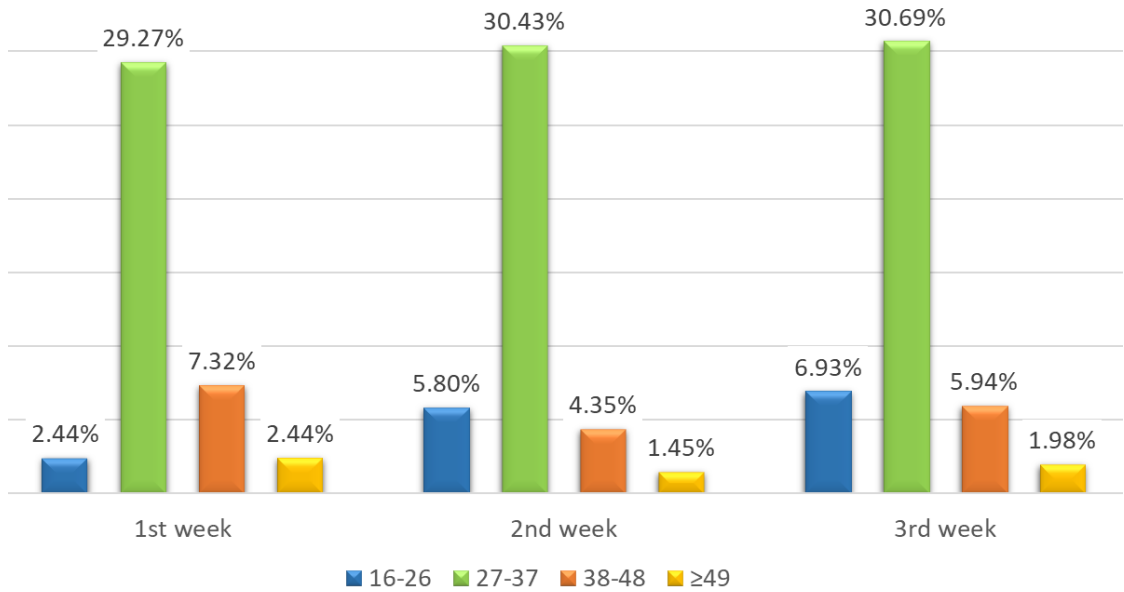


Figure 86: Graph regarding budget adherence in correlation to users who did not receive affective feedback based on age

Below, in Figure 87 and Table 8, the descriptive statistics of quantitate variables are presented.

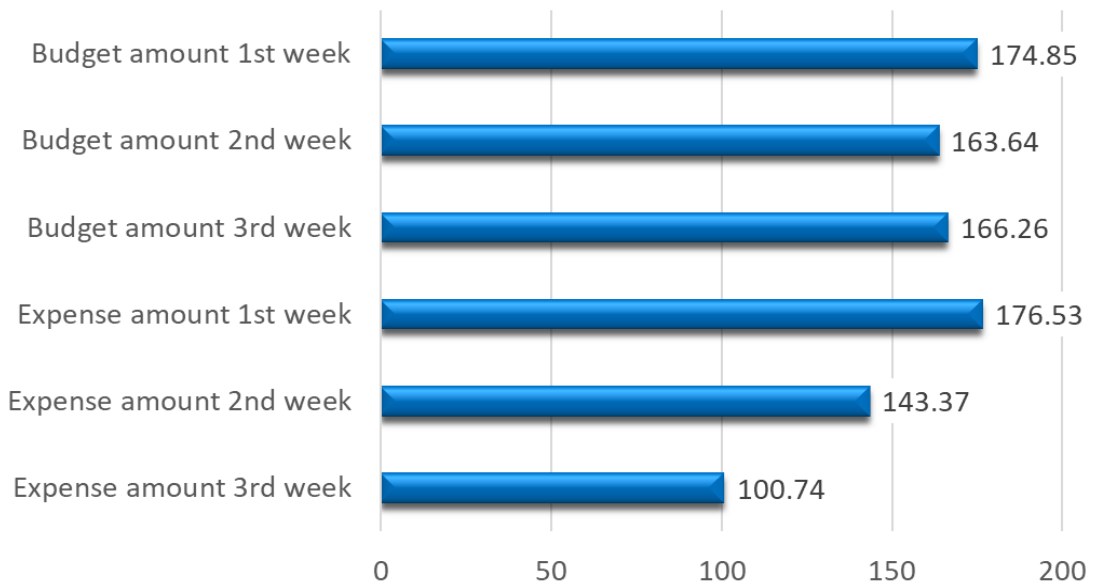


Figure 87: Graph of descriptive statistics of quantitate variables

Apparently, there was no significant difference in the “Budget Amount” variable in the three weeks period, as the average values were respectively 174.85, 163.64 and 166.26 for the first, second and third week, respectively. On the contrary, in the “Expense Amount” variable there was a decrease over time as the average values were

176.53, 143.37 and 100.74 respectively for the first, second and third week, respectively. The standard deviations were quite large and varied between [145.83-194.23] for “Budget Amount” and [97.51-140.29] for “Expense Amount”. Finally, the average values yield that in the first week the participants spent more than their budget indicating that they were likely drifted along by their consuming habits.

Table 8: Descriptive statistics of quantitate variables

<b>Descriptive statistics of quantitate variables</b>		
<b>Variables</b>	<b>Mean</b>	<b>Std. Deviation</b>
Budget amount 1st week	174.85	194.23
Budget amount 2nd week	163.64	145.83
Budget amount 3rd week	166.26	167.71
Expense amount 1st week	176.53	140.29
Expense amount 2nd week	143.37	97.51
Expense amount 3rd week	100.74	110.21

## 6.5 Inferential Statistics Results

Inferential statistics were obtained through parametric and non-parametric tests in order to answer the formulated research questions.

In order to answer the main research question regarding the influence of animated GIFs as an affective feedback technique on budget adherence (RQ1) the chi-square (X2) test was conducted. The results in Table 9 confirm that there were no statistically significant correlations between budget adherence animated GIFs ( $p > 0.05$ ). Chi-square was also used to study the influence of animated GIFs on budget adherence in relation to gender (RQ2). The results in Table 9 indicate that budget adherence was not influenced by animated GIFs in relation to gender ( $p > 0.05$ ). Finally, the Kruskal Wallis test was conducted in order to study the influence of animated GIFs on budget adherence in relation to age (RQ3). As observed in Table 9 there was not a statistically significant correlation between budget adherence and animated GIFs in relation to age ( $p > 0.05$ ).



Table 9: Correlation of “Budget Adherence” with the variables “Affective Feedback”, “Gender - Affective Feedback”, and “Age - Affective Feedback”

<b>Correlation of “Budget Adherence” with the variables “Affective Feedback”, “Gender - Affective Feedback”, and “Age - Affective Feedback”</b>			
<b>Variables</b>	<b>Affective Feedback (Chi-Square)</b>	<b>Gender – Affective Feedback (Chi-Square)</b>	<b>Age - Affective Feedback (Kruskal-Wallis)</b>
Budget adherence 1st week	0.208	0.584	0.147
Budget adherence 2nd week	0.067	0.289	0.730
Budget adherence 3rd week	0.137	0.517	0.206

Since the results did not implied a reliance between budget adherence and affective feedback, affective feedback – age, and affective feedback – gender respectively, a further analysis was conducted to investigate if the developed system promoted budget adherence in the whole population. In Table 10 below, the 95% confidence intervals for the “Budget Adherence” variable are displayed for the 3 weeks. As observed, the confidence intervals do not intersect under any circumstances, so differences can be considered statistically significant. It seems that overall, participants adhered to their budget more successfully week by week, with the first week being the worst and the third week being the best with regards to budget adherence.

Table 10: Confidence Intervals 95% for percentages of successful budget adherence

<b>Confidence Intervals 95% for percentages of successful budget adherence</b>			
<b>Variables</b>	<b>Percent of success</b>	<b>95%Lower Bound of success</b>	<b>95%Upper bound of success</b>
Budget adherence 1st week	31.30%	23.36%	39.24%
Budget adherence 2nd week	52.70%	44.15%	61.25%
Budget adherence 3rd week	76.30%	69.02%	83.58%

The above result was confirmed with the analysis that was conducted on the expense amounts. The mean difference between the values in respondents' expenses over the 3 weeks period was tested with the ANOVA Repeated Measures parameter test with the use of Bonferroni for multiple comparisons. The initial hypothesis was that

there was no difference in mean values (accepted when  $p \geq 0.05$ ), whilst the alternative hypothesis suggested that the difference existed (accepted when  $p < 0.05$ ).

Table 11 presents the results from the Bonferonni test for the “Expense Amount” variables. In all cases the differences were considered statistically significant ( $p < 0.05$ ). Participants spent most money during the first week, whilst they reduced their expenses during the 2nd and 3rd week. During the 3rd week the participants’ expenses were reduced even more in comparison to the 2nd week.

Table 11: Multiple comparisons for expense amount via Bonferonni

<b>Multiple comparisons for Expense Amount via Bonferonni</b>			
<b>Variable I</b>	<b>Variable J</b>	<b>Mean difference I-J</b>	<b>p-value</b>
Expense amount 1st week	Expense amount 2nd week	33.15	0.047
	Expense amount 3rd week	75.79	0.000
Expense amount 2nd week	Expense amount 1st week	-33.15	0.047
	Expense amount 3rd week	42.63	0.000
Expense amount 3rd week	Expense amount 1st week	-75.79	0.000
	Expense amount 2nd week	-42.63	0.000

Further analysis on the correlation (t-test, ANOVA, Kruskal-Wallis) of the expense amounts of the participants with the variables “Affective Feedback”, “Gender - Affective Feedback”, and “Age - Affective Feedback” (Table A15, Appendix). However, the results were not significant. This result indicates that the participants’ expenses were not influenced by animated GIFs, nor by animated GIFs in relation to their age, nor by animated GIFs in relation to their gender.

# 7 Discussion

This section starts with a brief discussion on the developed web app and the web technologies that were selected. Then, the key findings of the conducted research on affective feedback are summarized and discussed. Following, the limitations of the development of the system and the evolved experimental paradigm are analyzed. Finally, the possible theoretical and practical implications of the study are discussed, while proposals and directions for future research are provided.

## 7.1 Conclusions and Contributions

In this thesis, the need for developing an expense monitoring system with human-like behavior through affective feedback was addressed. Pertaining to highest level of development at time the “Budget Manager” prototype system was created based on the “Progressive Web App” standards. Engagement, accessibility, responsiveness, and offline availability are just some of the key features of the developed budgeting system, besides the required functionality. Moreover, based on the existing research on affective feedback techniques, an experimental paradigm was deployed to examine the influence of animated GIFs on budget adherence. Two groups of participants were studied, one that received affective feedback through GIFs and one that did not. Key findings entailed that the “Budget Manager” PWA actively aids in and promotes budget adherence. It was profound on the basis of the performed statistical analysis that the developed system achieved the primary goal of helping the users to stay on budget. The worst successful budget adherence of the participants was during the first week, and the best during the 3rd week, whilst successful budget adherence during the second week was better than the first week and worse than the third week. A closer look to participants’ expenses through this time confirmed the above result. Participants spent most money during the first week, whilst they reduced their expenses during the second and third week progressively.

However, there were not enough statistical evidence to support that animated GIFs as a means to provide affective influence budget adherence. In general, budget

adherence was improved for the whole population week after week, meaning that all participants reduced their expenses in each week successively, whether they were receiving affective feedback through GIFs or the default feedback (i.e. overview of their expenses). A possible explanation of this result is that the users were able to monitor their expenses every day and make strategical decisions regarding their budget and expenses. This means that the users had the flexibility of adjusting their expenses based on the remaining budget or adjusting their weekly budget based on their expenses. This resulted in better budget adherence for the majority of the participants. Additionally, users received feedback and affective feedback (depending on the week they belonged to) once per week. The duration of the feedback view was depended only on the time that the users dedicated to read and comprehend their weekly progress. This could mean that the time that users spent every week to comprehend the provided feedback was not enough to ensure the indulged emotion would endure throughout the week. The above offers another possible explanation of the non-significance of the findings.

## **7.2 Limitations**

Even though this study was carefully designed and conducted, some limitations should be considered. With regards to the developed app, time limitations did not allow the implementation of offline functionality. However, its design and implementation could support offline functionality, since the implementation of the Service Worker provides a robust and steady ground to build upon and provide the desired functionality of the app in the future.

As far as the experiment is concerned, one of the limitations lies in the fact that participants used the app only for a limited period of time. Repeating the experiment allowing the participants to use the app for a longer period of time, would allow for more accurate conclusions on the budget adherence and the influence of affective feedback. Additionally, the participants' population was not equally diverted with regards to age. The majority of the volunteers belonged in the early adulthood group (27 – 37 years old) as opposed to the other age groups. Different age groups have different characteristics and behaviors. For example, younger people tend to be more enthusiastic and intimate with new technologies than older people and therefore, more likely to use digital devices and applications. What is more, this diversity in age did not support statistical analysis by cross-referencing all the combinations of the “Affective

Feedback”, “Age” and “Gender” variables with the “Budget Amount”, “Expense Amount”, and “Budget Adherence” variables. A bigger sample is suggested to further study the influence of affective feedback in relation to demographic characteristics on budget adherence.

Worth of mentioning is also the fact that the conclusions of this study target only the specific animated GIFs that were deployed and the messages they convey. Different GIFs could possibly result in greater/different budget adherence. This lies on the fact that animated GIFs can be misinterpreted and often lead to miscommunication, with more diverse interpretations for positive GIFs than negative ones (Jiang, Brubaker & Fiesler, 2017; Bourlai & Herring, 2014). Finally, another limitation is that real time reactions and emotional states of the participants before and after receiving affective feedback were not monitored. Emotions are dynamic processes with meaningful differences in duration influenced by psychological and neural mechanisms (Verduyn & Lavrijsen, 2015; Verduyn, Delaveau, Rotgé, et al. 2015). This makes emotions and their duration a personal matter, since they are affected by the perception of an individual for the importance of an encountered event and the regulation strategy selected by the same individual to deal with that emotion (Verduyn & Lavrijsen, 2015; Verduyn, Delaveau, Rotgé, et al. 2015). The indulged emotion of the affective feedback that was received from users was not identified instantly and on time therefore, it was not affirmed that users could remember the provoked emotion through animated Gifs and act accordingly during the whole week while monitoring their expenses. Monitoring real time reactions and emotional states would allow for a more accurate recognition of users’ emotions and affective states instantly, i.e. the moment that animated GIFs were viewed.

### **7.3 Theoretical and Practical Implications**

Notwithstanding these limitations, the findings of this study have both theoretical and practical implications. Regarding theoretical implications, this study builds on the existing research surrounding the main elements that were in the central focus. The current study adds value in the sense that that animated GIFs were studied as a means of affective feedback in the context of budget adherence. Further research on the possible affective feedback techniques that shape user behavior is needed of course, to broaden our horizons on the underlying factors that can determine the successful

budget adherence. It is suggested for future research to implement another collection of animated GIFs that can provoke positive emotions while monitoring users' reactions in real time. Other suggestions encompass the use of other affective feedback techniques in relation to users' educational background, IQ and EQ.

On the practical implications of this study, it is important to highlight that PWAs constitute the future of web development. Such applications, PWAs, provide the same feeling as native mobile apps do, and can work offline or on low-quality networks (Gazdecki, 2017; LePage 2018, Google Developers, n.d.). Additionally, they are cross-platform applicable and allow users to add the apps they find most useful to their mobile home screen without the intervention of an app store (Russell, 2015; Gazdecki, 2017; LePage 2018). Furthermore, PWAs depend only on the browser used to be launched rather than the operational system of the device that access it. However, since these apps are still in its infancy many browsers do not fully support them. According to Gazdecki, (2017) and LePage (2018) these apps are developed with innovative and modern web technologies and further study and research is needed. Supplementary research in this emerging field could aid in understanding better the limitations of the current browser programs, and improve the compatibility of PWAs. Last but not least, the user experience of a PWA in contrast to a native mobile app and/or a simple web app could be investigated. This could be further prove the need for the PWAs and their importance by studying their advantages, disadvantages and users' attitudes against them.

With regards to the functionality of the system, there are opportunities for extending the existing scope of the application. A great feature to be implemented is the ability to connect a bank account to the app allowing to retrieve expenses automatically without the need of users' interference, while respecting privacy issues that may arise. Another key feature would be the integration of APIs that display sales on categories selected by the users. Additionally, the implementation of scanning receipts via a camera when the app is accessed via mobile devices could add an augmented reality component in the app. Moreover, it could be very useful if users were able to partition their budget in accounts, defining a specific purpose in each one. Last but not least, another feature that could increase the usefulness of the app is the ability to display the savings per week, and aggregated based on selected filters. All the aforementioned are

just some examples that could broaden the functionality of the system catering for the most demanding users.

Concluding, the insights of this study are a valuable addition to the existing knowledge both on affective computing and web development while offering guidelines and directions for future research.





# Bibliography

- Archer, J. C. (2010). State of the science in health professional education: effective feedback. *Medical education*, 44(1), 101-108.
- Archibald, J. (2018). The Offline Cookbook. Retrieved from <https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/>
- Ash, J. (2015). Sensation, networks, and the GIF: Toward an allotropic account of affect. In K. Hillis, S. Paasonen, & M. Petit (Eds.), *Networked affect*, 119 - 133. Cambridge: The MIT Press.
- Bakhshi, S., Shamma, D. A., Kennedy, L., Song, Y., de Juan, P., & Kaye, J. J. (2016). Fast, cheap, and good: Why animated GIFs engage us. In *Proceedings of the 2016 chi conference on human factors in computing systems*, 575-586, ACM.
- Bangert-Drowns, R. L., Kulik, C. L. C., Kulik, J. A., & Morgan, M. (1991). The instructional effect of feedback in test-like events. *Review of educational research*, 61(2), 213-238.
- Bartoszek, G., & Cervone, D. (2016). Toward an implicit measure of emotions: Ratings of abstract images reveal distinct emotional states. *Cognition and emotion*, 31(7), 1377-1391.
- Baumeister, R. F., Bratslavsky, E., Finkenauer, C., & Vohs, K. D. (2001). Bad is stronger than good. *Review of general psychology*, 5(4), 323.
- Bee, R., & Bee, F. (1998). *Constructive feedback*. CIPD Publishing.
- Berger, R. (2015a). Budgeting Tools To Better Manage Your Money. Retrieved from <https://www.forbes.com/sites/robertberger/2015/11/19/7-budgeting-tools-to-better-manage-your-money/#f3186d442746>
- Berger, R. (2015b). 7 Tips For Effective And Stress-Free Budgeting. Retrieved from <https://www.forbes.com/sites/robertberger/2015/07/26/7-tips-for-effective-and-stress-free-budgeting/#981e53b26872>
- Borup, J., West, R. E., Thomas, R., & Graham, C. R. (2014). Examining the impact of video feedback on instructor social presence in blended courses. *The International Review of Research in Open and Distributed Learning*, 15(3).

- Bourlai, E., & Herring, S. C. (2014). Multimodal communication on tumblr: i have so many feels!. In Proceedings of the 2014 ACM conference on Web science, 171-175, ACM.
- Bringoux, L., Monnoyer, J., Besson, P., Bourdin, C., Denjean, S., Dousset, E., ... & Martha, C. (2017). Influence of speed-related auditory feedback on braking in a 3D-driving simulator. *Transportation research part F: traffic psychology and behaviour*, 44, 76-89.
- Brooke, K. (2014). 5 habits of budget-savvy people. Retrieved from <https://www.livingwellspendingless.com/2014/05/09/habits-budget-savvy-people/>
- Buczynski, S. (2009). Formative feedback stimulates students' thinking and provides teachers with information to guide future instruction. *Tips for Providing Formative Feedback*, 10, 1-2.
- Burri, R. V. (2012). Visual rationalities: Towards a sociology of images. *Current Sociology*, 60(1), 45-60.
- Caldwell, M. (2018). How to Budget Successfully. Retrieved from <https://www.thebalance.com/how-to-budget-successfully-2385709>
- Colston, K. (2018). How To Create Your Event Budget. Retrieved from <https://helloendless.com/how-to-create-your-event-budget/>
- Corsini, R. J. (1999). *The dictionary of psychology*. Psychology Press.
- Crook, A., Mauchline, A., Maw, S., Lawson, C., Drinkwater, R., Lundqvist, K., Orsmond, P., Gomez, S., & Park, J. (2012). The use of video technology for providing feedback to students: Can it enhance the feedback experience for staff and students?. *Computers & Education*, 58(1), 386-396.
- Datta, R., Joshi, D., Li, J., & Wang, J. Z. (2006). Studying aesthetics in photographic images using a computational approach. In *European Conference on Computer Vision*, 288-301, Springer, Berlin, Heidelberg.
- Datta, R., Li, J., & Wang, J. Z. (2008). Algorithmic inferencing of aesthetics and emotion in natural images: An exposition. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, 105-108, IEEE.
- Dehn, D. M., & Van Mulken, S. (2000). The impact of animated interface agents: a review of empirical research. *International journal of human-computer studies*, 52(1), 1-22.
- Earl, L. M. (2012). *Assessment as learning: Using classroom assessment to maximize student learning*. (2nd edition) Corwin Press.

- Eppink, J. (2014). A brief history of the GIF (so far). *Journal of Visual Culture*, 13(3), 298-306.
- Fitzgerald, A. (2015). Why you should start using Font Awesome. Retrieved from <https://getflywheel.com/layout/why-you-should-start-using-font-awesome/>
- Flanagan, D. (2011). *JavaScript: The definitive guide: Activate your web pages.* " O'Reilly Media, Inc."
- Frank, T. (2018). How to Budget and Save Money as a College Student. Retrieved from <https://collegeinfo geek.com/budgeting-for-college-students/>
- Fredrickson, B. L., & Losada, M. F. (2005). Positive affect and the complex dynamics of human flourishing. *American psychologist*, 60(7), 678.
- Fontinelle, A. (2017). Budgeting Basics. Retrieved from <https://www.investopedia.com/university/budgeting/>
- Funk, C., Kennedy, B. & Podrebarac-Sciupac, E. (2016). Public sees science and technology as net positives for society. Retrieved from <http://www.pewinternet.org/2016/07/26/public-sees-science-and-technology-as-net-positives-for-society/>
- Gaunt, M. (2018a). Introduction to fetch(). Retrieved from <https://developers.google.com/web/updates/2015/03/introduction-to-fetch>
- Gaunt, M. (2018b). Service Workers: an Introduction. Retrieved from <https://developers.google.com/web/fundamentals/primers/service-workers/>
- Gaunt, M. & Kinlan, P. (2018). The Web App Manifest. Retrieved from <https://developers.google.com/web/fundamentals/web-app-manifest/>
- Gazdecki, A. (2017). 10 reasons progressive web apps will be the future of apps. Retrieved from <https://torquemag.io/2017/11/10-reasons-pwas-will-future-apps/>
- Goodman, J. S., Wood, R. E., & Hendrickx, M. (2004). Feedback Specificity, Exploration, and Learning. *Journal of Applied Psychology*, 89(2), 248-262.
- Google Developers. (2018a). Introduction to Gulp. Retrieved from <https://developers.google.com/web/ilt/pwa/introduction-to-gulp>
- Google Developers, (2018b). Introduction to Service Worker. Retrieved from <https://developers.google.com/web/ilt/pwa/introduction-to-service-worker>
- Google Developers, (n.d.). Progressive Web Apps. Retrieved from <https://developers.google.com/web/progressive-web-apps/>
- Google Developers, (2018c). Progressive Web App Checklist. Retrieved from <https://developers.google.com/web/progressive-web-apps/checklist>

- Grannan, C. (n.d.). What's the Difference Between Emoji and Emoticons? Retrieved from <https://www.britannica.com/story/whats-the-difference-between-emoji-and-emoticons>
- Grieve, R., Moffitt, R. L., & Padgett, C. R. (2018). Student perceptions of marker personality and intelligence: The effect of emoticons in online assignment feedback. *Learning and Individual Differences*. doi:10.1016/j.lindif.2018.02.008
- Grigorik, I. (2015). Eliminating Roundtrips with Preconnect, Retrieved from <https://www.igvita.com/2015/08/17/eliminating-roundtrips-with-preconnect/>
- Hall, T., Tracy, D., & Lamey, A. (2016). Exploring Video Feedback in Philosophy: Benefits for Instructors and Students. *Teaching Philosophy*.
- Hall, L., Woods, S., Aylett, R., Newall, L., & Paiva, A. (2005). Achieving empathic engagement through affective interaction with synthetic characters. In *International Conference on Affective Computing and Intelligent Interaction*, 731-738, Springer, Berlin, Heidelberg.
- Hamid, Y., & Mahmood, S. (2010). Understanding constructive feedback: a commitment between teachers and students for academic and professional development. *J Pak Med Assoc*, 60(3), 224-7.
- Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of educational research*, 77(1), 81-112.
- Henderson, M., & Phillips, M. (2015). Video-based feedback on student assessment: scarily personal. *Australasian Journal of Educational Technology*, 31(1).
- Hern, A. (2015). Don't know the difference between emoji and emoticons? Let me explain. Retrieved from <https://www.theguardian.com/technology/2015/feb/06/difference-between-emoji-and-emoticons-explained>
- Huang, Y. Y., Moll, J., Sallnäs, E. L., & Sundblad, Y. (2012). Auditory feedback in haptic collaborative interfaces. *International journal of human-computer studies*, 70(4), 257-270.
- Hung, S. T. A. (2016). Enhancing feedback provision through multimodal video technology. *Computers & Education*, 98, 90-101.
- Issa, T. (2016). Teamwork Assessment and Self/Peer Evaluation in Higher Education. In *Leadership and Personnel Management: Concepts, Methodologies, Tools, and Applications*, 1713-1729, IGI Global.
- Isaac, M. (2015) For Mobile Messaging, GIFs Prove to Be Worth at Least a Thousand Words. Retrieved from <https://www.nytimes.com/2015/08/04/technology/gifs-go-beyond-emoji-to-express-thoughts-without-words.html>

- Jiang, J. A., Brubaker, J. R., & Fiesler, C. (2017). Understanding Diverse Interpretations of Animated GIFs. In Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, 1726-1732, ACM.
- Juwah, C., Macfarlane-Dick, D., Matthew, B., Nicol, D., Ross, D., & Smith, B. (2004). Enhancing student learning through effective formative feedback. *The Higher Education Academy*, 140.
- Kanai, A. (2015). Jennifer Lawrence, remixed: approaching celebrity through DIY digital culture. *Celebrity Studies*, 6(3), 322-340.
- Kay, M. (n.d.). Better Font Awesome. Retrieved from <https://el.wordpress.org/plugins/better-font-awesome/>
- Kaye, L. K., Wall, H. J., & Malone, S. A. (2016). "Turn that frown upside-down": A contextual account of emoticon usage on different virtual platforms. *Computers in Human Behavior*, 60, 463-467. doi: <http://dx.doi.org/10.1016/j.chb.2016.02.088>.
- Khalid, A. (2017). Emoji vs. emoticon: What's the difference? Retrieved from <https://www.dailydot.com/debug/emoji-vs-emoticon-differences-explained/>
- Kluger, A. N., & Adler, S. (1993). Person-versus computer-mediated feedback. *Computers in human behavior*, 9(1), 1-16.
- Kulhavy, R. W., & Stock, W. A. (1989). Feedback in written instruction: The place of response certitude. *Educational Psychology Review*, 1(4), 279-308.
- Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., & Campbell, A. T. (2010). A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9).
- LePage, P. (2018). Your First Progressive Web App. Retrieved from <https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/>
- Linn, R. L., & Miller, M. D. (2005). *Measurement and assessment in teaching* (9th ed.). Upper Saddle River, NJ: Pearson Prentice Hall.
- Losada, M. (1999). The complex dynamics of high performance teams. *Mathematical and computer modelling*, 30(9-10), 179-192.
- Losada, M., & Heaphy, E. (2004). The role of positivity and connectivity in the performance of business teams: A nonlinear dynamics model. *American Behavioral Scientist*, 47(6), 740-765.
- Lunt, T., & Curran, J. (2010). 'Are you listening please?' The advantages of electronic audio feedback compared to written feedback. *Assessment & evaluation in higher education*, 35(7), 759-769.

- Machajdik, J., & Hanbury, A. (2010, October). Affective image classification using features inspired by psychology and art theory. In Proceedings of the 18th ACM international conference on Multimedia, 83-92, ACM.
- Maculewicz, J., Erkut, C., & Serafin, S. (2016). An investigation on the impact of auditory and haptic feedback on rhythmic walking interactions. *International Journal of Human-Computer Studies*, 85, 40-46.
- McDarby, G., Condrón, J., Hughes, D., Augenblick, N., & Sharry, J. (2004). Affective feedback. In *Enabling Technologies: Body Image and Body Function*, 115 - 130, Churchill Livingstone
- Merkel, W. R. (1973). *Dictionary of education*, 3rd edition New York: McGraw-Hill Publishing;227–8;
- Merry, S., & Orsmond, P. (2008). Students' attitudes to and usage of academic feedback provided via audio files. *Bioscience Education*, 11(1), 1-11.
- Meyer, R. E. (1995). Feedback. In: Anderson LW, ed. *International Encyclopaedia of Teaching and Teacher Education*, 2nd edition. Oxford: Pergamon Press; 249–51.
- Miller, H., Thebault-Spieker, J., Chang, S., Johnson, I., Terveen, L., & Hecht, B. (2016). "Blissfully happy" or "ready to fight": Varying Interpretations of Emoji. *Proceedings of Seventh International AAAI Conference on Web and Social Media (ICWSM)*.
- Miltner, K. M., & Highfield, T. (2017). Never gonna GIF you up: Analyzing the cultural significance of the animated GIF. *Social Media+ Society*, 3(3), 2056305117725223.
- Moll, J., Huang, Y., & Sallnäs, E. L. (2010). Audio makes a difference in haptic collaborative virtual environments. *Interacting with Computers*, 22(6), 544-555.
- Moreno, R. (2004). Decreasing cognitive load for novice students: Effects of explanatory versus corrective feedback in discovery-based multimedia. *Instructional science*, 32(1-2), 99-113.
- Moridis, C. N., & Economides, A. A. (2008). Toward computer-aided affective learning systems: a literature review. *Journal of Educational Computing Research*, 39(4), 313-337.
- Moridis, C. N., & Economides, A. A. (2009). Mood recognition during online self-assessment tests. *IEEE Transactions on Learning technologies*, 2(1), 50-61.
- Moridis, C. N., & Economides, A. A. (2012a). Affective learning: Empathetic agents with emotional facial and tone of voice expressions. *IEEE Transactions on Affective Computing*, 3(3), 260-272.

- Moridis, C. N., & Economides, A. A. (2012b). Applause as an achievement-based reward during a computerised self-assessment test. *British Journal of Educational Technology*, 43(3), 489-504.
- Muntean, C. I. (2011). Raising engagement in e-learning through gamification. In *Proc. 6th International Conference on Virtual Learning ICVL (Vol. 1)*.
- Narciss, S. (2008). Feedback strategies for interactive learning tasks. *Handbook of research on educational communications and technology*, 3, 125-144.
- Newman, M. Z. (2016). *GIFs: The attainable text*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library.
- Norman, D. (2002). Emotion & design: Attractive things work better. *Interactions: New Visions of Human-Computer Interaction*, 9(4), 36-42.
- Oviatt, S., Coulston, R., Tomko, S., Xiao, B., Lunsford, R., Wesson, M., & Carmichael, L. (2003, November). Toward a theory of organized multimodal integration patterns during human-computer interaction. In *Proceedings of the 5th international conference on Multimodal interfaces*, 44-51, ACM.
- O'Shea, A., Schwahn, L. (2018). Best Budget Apps and Personal Finance Tools for 2019. Retrieved from <https://www.nerdwallet.com/blog/finance/budgeting-saving-tools/>
- Page, G. T., Thomas, J. B., & Marshall, A. R. (1978). *International dictionary of education*.
- Park, J., Barash, V., Fink, C., & Cha, M. (2013). Emoticon Style: Interpreting Differences in Emoticons Across Cultures. *Proceedings of Tenth International AAI Conference on Web and Social Media (ICWSM)*.
- Pantic, M., & Rothkrantz, L. J. (2003). Toward an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 91(9), 1370-1390.
- Parton, B. S., Crain-Dorough, M., & Hancock, R. (2010). Using flip camcorders to create video feedback: Is it realistic for professors and beneficial to students. *International Journal of Instructional Technology & Distance Learning*, 7(1), 15-23. Retrieved from [http://www.itdl.org/Journal/Jan\\_10/article02.htm](http://www.itdl.org/Journal/Jan_10/article02.htm)
- Penzo, L. (n.d.). 50 Personal Finance Habits Everyone Should Follow. Retrieved from <http://time.com/money/collection-post/4023439/personal-finance-habits/>
- Phil. (2013). How to Make Budgeting a Habit. Retrieved from <https://www.budgetsimple.com/blog/how-to-make-budgeting-a-habit/>
- Picard, R. W. (1997). *Affective computing*. The MIT Press, Cambridge (MA)



- Picard, R. W. (2003). Affective computing: challenges. *International Journal of Human-Computer Studies*, 59(1-2), 55-64.
- Posnick, J., (2018). Service Worker Registration. Retrieved from <https://developers.google.com/web/fundamentals/primers/service-workers/registration>
- Roberts, H. (2013) . Front-end performance for web designers and front-end developers. Retrieved from <https://csswizardry.com/2013/01/front-end-performance-for-web-designers-and-front-end-developers/#section:dns-prefetching>
- Robison, J., McQuiggan, S., & Lester, J. (2009). Evaluating the consequences of affective feedback in intelligent tutoring systems. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, 1-6, IEEE.
- Rosenberg, E. (2018). The 8 Best Budgeting Apps to Download in 2018. Retrieved from <https://www.thebalance.com/best-budgeting-apps-4159414>
- Russell, A. (2015). Progressive Web Apps: Escaping Tabs Without Losing Our Soul. Retrieved from <https://medium.com/@slightlylate/progressive-apps-escaping-tabs-without-losing-our-soul-3b93a8561955>
- Russell, J. A. (2003). Core affect and the psychological construction of emotion. *Psychological review*, 110(1), 145.
- Schneider, J., Auten, D. (2018). 5 Awesome Budgeting Apps For 2018 And Beyond. Retrieved from <https://www.forbes.com/sites/debtfreeguys/2018/09/09/5-awesome-budgeting-apps-for-2018-and-beyond/#5e1e38c55561>
- Seebode, J., Schleicher, R., & Möller, S. (2012, December). Affective quality of audio feedback in different contexts. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia* (p. 32). ACM.
- Sha, H. (2016). The digital materiality of GIFs. Retrieved from <https://a-million-neon-rainbows.tumblr.com/post/140270434771/essay-the-digital-materiality-of-gifs>
- Shute, V. J. (2008). Focus on formative feedback. *Review of educational research*, 78(1), 153-189.
- Suhr, K. (2014) Using animated GIF images for library instruction. Retrieved from <http://www.inthelibrarywiththeleadpipe.org/2014/using-animated-gif-images-for-library-instruction/>
- Sultan, A. S., & Khan, M. A. M. (2017). Feedback in a clinical setting: A way forward to enhance student's learning through constructive feedback. *J Pak Med Assoc*, 67(7), 1078-84.



- Taras, M. (2002). Using assessment for learning and learning from assessment. *Assessment & Evaluation in Higher Education*, 27(6), 501-510.
- Tsutsui, M. (2004). Multimedia as a means to enhance feedback. *Computer Assisted Language Learning*, 17(3-4), 377-402.
- Tolins, J., & Samermit, P. (2016). GIFs as embodied enactments in text-mediated conversation. *Research on Language and Social Interaction*, 49(2), 75-91.
- Ur, B., Kelley, P. G., Komanduri, S., Lee, J., Maass, M., Mazurek, M. L., ... & Christin, N. (2012). How does your password measure up? The effect of strength meters on password creation. In *USENIX Security Symposium*, 65-80.
- Van Beuningen, C. (2011). The effectiveness of comprehensive corrective feedback in second language writing. (Doctoral dissertation). Retrieved from Universiteit van Amsterdam Digital Academic Repository.
- Van Beuningen, C. G., De Jong, N. H., & Kuiken, F. (2012). Evidence on the effectiveness of comprehensive error correction in second language writing. *Language learning*, 62(1), 1-41.
- Van De Ridder, J. M., Stokking, K. M., McGaghie, W. C., & Ten Cate, O. T. J. (2008). What is feedback in clinical education?. *Medical education*, 42(2), 189-197.
- Veksler, D. L. (2016). Financial Responsibility is About Habits, Not Budgeting. Retrieved from <https://fee.org/articles/financial-responsibility-is-about-habits-not-budgeting/>
- Verduyn, P., & Lavrijsen, S. (2015). Which emotions last longest and why: The role of event importance and rumination. *Motivation and Emotion*, 39(1), 119-127.
- Verduyn, P., Delaveau, P., Rotgé, J. Y., Fossati, P., & Van Mechelen, I. (2015). Determinants of emotion duration and underlying psychological and neural mechanisms. *Emotion Review*, 7(4), 330-335.
- Vohwinkle, J. (2018). Basic Budgeting Tips Everyone Should Know. Retrieved from <https://www.thebalance.com/budgeting-101-1289589>
- Wall, H. J., Kaye, L. K., & Malone, S. A. (2016). An exploration of psychological factors on emoticon usage and implications for judgement accuracy. *Computers in Human Behavior*, 62, 70-78.
- Walker, A., & White, G. (2013). *Technology Enhanced Language Learning: connecting theory and practice-Oxford Handbooks for Language Teachers*. Oxford University Press.
- Wang, X., Jia, J., Yin, J., & Cai, L. (2013). Interpretable aesthetic features for affective image classification. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, 3230-3234, IEEE.

- Wang, W., Zhao, Y., Qiu, L., & Zhu, Y. (2014). Effects of emoticons on the acceptance of negative feedback in computer-mediated communication. *Journal of the Association for Information Systems*, 15(8), 454.
- Williams, G. (2015). 5 Ways to Break Your Bad Money Habits. Retrieved from <https://money.usnews.com/money/personal-finance/articles/2015/04/16/5-ways-to-break-your-bad-money-habits>
- Yanulevskaya, V., van Gemert, J. C., Roth, K., Herbold, A. K., Sebe, N., & Geusebroek, J. M. (2008). Emotional valence categorization using holistic image features, 101-104, in ICIP.
- “12 Techniques of Website Speed Optimization: Performance Testing and Improvement Practices”, (2018). Retrieved from <https://www.altexsoft.com/blog/engineering/12-techniques-of-website-speed-optimization-performance-testing-and-improvement-practices/>
- “Best Practices for Speeding Up Your Web Site - Yahoo Developer Network”, (n.d.). Retrieved from <https://developer.yahoo.com/performance/rules.html?guccounter=1>
- “Basic Use” (n.d.). Retrieved from <https://fontawesome.com/how-to-use/on-the-web/setup/getting-started>
- “CSS Flexbox”, (n.d.). Retrieved from [https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)
- “CSS Media Queries”, (n.d.). Retrieved from [https://www.w3schools.com/css/css3\\_mediaqueries.asp](https://www.w3schools.com/css/css3_mediaqueries.asp)
- “feedback”, (n.d.) Retrieved from <http://www.businessdictionary.com/definition/feedback.html>
- “feedback”, (n.d.) Retrieved from <https://www.techopedia.com/definition/7159/feedback>
- “Front-end performance for web designers and front-end developers, (2013). Retrieved from <https://csswizardry.com/2013/01/front-end-performance-for-web-designers-and-front-end-developers/>
- “What is a Budget? Budgeting Terms and Tips” (n.d.) Retrieved from <https://www.investopedia.com/terms/b/budget.asp>
- “What is a Budget?” (n.d.) Retrieved from <https://www.myaccountingcourse.com/accounting-dictionary/budget>

“What is Budgeting? What is a Budget?” (n.d.) Retrieved from

<https://www.mymoneycoach.ca/budgeting/what-is-a-budget-planning-forecasting>

# Appendixes

## Abbreviations and Definitions

All definitions were retrieved from [technopedia.com](http://technopedia.com), except the ones where the source is mentioned.

### **ARIA : Accessible Rich Internet Applications**

ARIA is a set of attributes that define ways to make Web content and Web applications (especially those developed with JavaScript) more accessible to people with disabilities.

Source: <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA>

### **AJAX: Asynchronous JavaScript and XML**

AJAX is a method of building interactive applications for the Web that process user requests immediately. It combines several programming tools including JavaScript, dynamic HTML (DHTML), Extensible Markup Language (XML), cascading style sheets (CSS), the Document Object Model (DOM), and the Microsoft object, XMLHttpRequest. AJAX allows content on Web pages to update immediately when a user performs an action, unlike an HTTP request, during which users must wait for a whole new page to load.

### **API: Application Programming Interface**

An API is a set of protocols, routines, functions and/or commands that programmers use to develop software or facilitate interaction between distinct systems. APIs are available for both desktop and mobile use, and are typically useful for programming graphic user interface components, as well as allowing a software program to request and accommodate services from another program.

### **APP: Application**

An app is computer software, or a program, most commonly a small, specific one used for mobile devices. The term app originally referred to any mobile or desktop application, but as more app stores have emerged to sell mobile apps to smartphone and tablet users, the term has evolved to refer to small programs that can be downloaded and

installed all at once. There are thousands of apps designed to run on today's smartphones and tablets. Some apps can be downloaded for free, while others must be purchased from an app store.

## **CSS: Cascading Style Sheets**

CSS is used to apply styling to HTML content, for example setting background colors and fonts, and laying out the content in multiple columns. It describes how HTML elements are to be displayed on screen, paper, or in other media and can control the layout of multiple web pages all at once.

## **CSS3: Cascading Style Sheets, revision 3**

CSS3 is the latest evolution of the Cascading Style Sheets language and aims at extending CSS. It brings a lot of long-awaited novelties, like rounded corners, shadows, gradients, transitions or animations, as well as new layouts like multi-columns, flexible box or grid layouts. Experimental parts are vendor-prefixed and should either be avoided in production environments, or used with extreme caution as both their syntax and semantics can change in the future.

## **DNS: Domain Name System**

A DNS server is a type of name server that manages, maintains and processes Internet domain names and their associated records. In other words, a DNS server is the primary component that implements the DNS protocol and provisions domain name resolution services to Web hosts and clients on an IP-based network.

## **HTML: Hypertext Markup Language**

HTML is a standardized system for tagging text files to achieve font, color, graphic, and hyperlink effects on World Wide Web pages. It is the markup language used to structure and give meaning to web content, for example defining paragraphs, headings, and data tables, or embedding images and videos in the page.

## **HTML5: Hypertext Markup Language, revision 5**

HTML5 is markup language for the structure and presentation of World Wide Web contents. HTML5 supports the traditional HTML and XHTML-style syntax and other new features in its markup, New APIs, XHTML (Extensible Hypertext Markup Language) and error handling.

## **HTTP: HyperText Transfer Protocol**

HTTP is an application-layer protocol used primarily on the World Wide Web. HTTP uses a client-server model where the web browser is the client and communicates with the webserver that hosts the website. The browser uses HTTP, which is carried over TCP/IP to communicate to the server and retrieve Web content for the user. HTTP is a widely used protocol and has been rapidly adopted over the Internet because of its simplicity. It is a stateless and connectionless protocol.

## **HTTPS: HyperText Transfer Protocol Secure**

HTTPS is a variant of the standard web transfer protocol, HTTP, that adds a layer of security on the data in transit through a secure socket layer (SSL) or transport layer security (TLS) protocol connection. HTTPS enables encrypted communication and secure connection between a remote user and the primary web server.

## **IDE: Integrated Development Environment**

An IDE is an application that facilitates application development. In general, an IDE is a graphical user interface (GUI)-based workbench designed to aid a developer in building software applications with an integrated environment combined with all the required tools at hand. Most common features, such as debugging, version control and data structure browsing, help a developer quickly execute actions without switching to other applications. Thus, it helps maximize productivity by providing similar user interfaces (UI) for related components and reduces the time taken to learn the language. An IDE supports single or multiple languages.

## **IP: Internet Protocol**

IP provides a standard set of rules for sending and receiving data over the Internet. It allows devices running on different platforms to communicate with each other as long as they are connected to the Internet. In order for an Internet-connected host to be recognized by other devices, it must have an IP address.

## **JavaScript**

JavaScript is a scripting language, primarily used on the Web to enhance HTML pages. It is an interpreted language, thus it does not need to be compiled. JavaScript renders web pages in an interactive and dynamic fashion and allows the implementation of complex things on web pages. JavaScript enables interaction since it allows the pages to

react to events, exhibit special effects, accept variable text, validate data, create cookies, detect a user's browser, etc. It is the third layer of the layer cake of standard web technologies, two of which (HTML and CSS).

### **JSON: JavaScript Object Notation**

JSON is a text-based, human-readable data interchange format used for representing simple data structures and objects in Web browser-based code. JSON is also sometimes used in desktop and server-side programming environments. JSON was originally based on the JavaScript programming language and was introduced as the page scripting language for the Netscape Navigator Web browser.

### **MySQLi**

The MySQLi Extension is a relational database driver used in the PHP scripting language to provide an interface with MySQL databases.

Source: <https://www.tutorialspoint.com/mysqli/>

### **PHP: Hypertext Preprocessor**

PHP is a script language and interpreter that is freely available and used primarily on Linux Web servers. PHP, originally derived from Personal Home Page Tools, now stands for PHP: Hypertext Preprocessor, which the PHP FAQ describes as a “recursive acronym”.

### **SQL: Structured Query Language**

SQL is a standard computer language for relational database management and data manipulation. SQL is used to query, insert, update and modify data. Most relational databases support SQL, which is an added benefit for database administrators, as they are often required to support databases across several different platforms.

### **TCP: Transmission Control Protocol**

TCP is a network communication protocol designed to send data packets over the Internet. TCP is a transport layer protocol in the open systems interconnection layer and is used to create a connection between remote computers by transporting and ensuring the delivery of messages over supporting networks and the Internet.

## **TLS: Transport Layer Security**

TLS is a protocol that provides communication security between client/server applications that communicate with each other over the Internet. It enables privacy, integrity and protection for the data that's transmitted between different nodes on the Internet. TLS is a successor to the secure socket layer (SSL) protocol.

## **UI: User Interface**

UI is a broad term for any system, either physical or software based, that allows a user to connect with a given technology. Many different kinds of user interfaces come with various devices and software programs. Many of them have some basic similarities, although each one is unique in key ways.

## **URL: Uniform Resource Locator / Universal Resource Locator**

A URL is the address of a resource on the Internet and indicates the location of a resource as well as the protocol used to access it and contains information regarding the protocol used to access the resource, the location of the server (whether by IP address or domain name), the port number on the server (optional), the location of the resource in the directory structure of the server, and a fragment identifier (optional). Also known as a Universal Resource Locator (URL) or Web address, a URL is a type of uniform resource identifier.



## Resources for Development

A list with all tools and programs used for the development of the prototype system.

### APIs, Plugins and Libraries

blob-select: <https://github.com/Blobfolio/blob-select>

Chart.js: <https://www.chartjs.org/>

Fetch API: <https://fetch.spec.whatwg.org/>

Font Awesome: <https://fontawesome.com/>

Grudus Timepicker: <https://github.com/grudus/Timepicker>

md-date-time-picker: <https://github.com/puranjayjain/md-date-time-picker>

Roboto Font Family: <https://fonts.google.com/specimen/Roboto>

### Software and Tools

Adobe Illustrator: [https://www.adobe.com/gr\\_en/products/illustrator.html](https://www.adobe.com/gr_en/products/illustrator.html)

Atom: <https://atom.io/>

FilleZilla Client: <https://filezilla-project.org/>

Gulp.js: <https://gulpjs.com/>

Node.js: <https://nodejs.org/en/>

NPM: <https://www.npmjs.com/>

XAMPP: <https://www.apachefriends.org/index.html>

Table A1 includes a comprehensive list with all packages required to utilize the Gulp build tool.

Table A1: Gulp packages

Package	Version	License	Description
del	3.0.0	MIT	Deletes files and folders using globs. Includes a Promise API and support for multiple files and globbing. It also protects against deleting the current working directory and above.
gulp-autoprefixer	6.0.0	MIT	Parses CSS and add vendor prefixes to CSS rules.
gulp-babel	7.0.1	MIT	Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible

			version of JavaScript in current and older browsers or environments.
babel-core	6.26.3	MIT	Babel compiler core.
@babel/core	7.1.2	MIT	Babel compiler core.
babel-cli	6.26.0	MIT	Babel command line. In addition, various entry point scripts live in the top-level package at babel-cli/bin. There are some shell-executable utility scripts, babel-external-helpers.js and babel-node.js, and the main Babel cli script, babel.js.
babel-preset-es2015	6.24.1	MIT	Babel preset for all es2015 plugins.
@babel/preset-env	7.1.0	MIT	A Babel preset for each environment.
gulp-clean-css	3.10.0	MIT	Plugin to clean CSS files.
gulp-concat	2.6.1	MIT	This concatenates files by the operating system's newLine. It takes the base directory from the first file that passes through it.
gulp-htmlmin	5.0.1	MIT	Minifies HTML files.
gulp-html-replace	1.6.2	MIT	Replaces build blocks in HTML.
gulp-imagemin	4.1.0	MIT	Minifies and optimizes PNG, JPEG, GIF and SVG images.
gulp-install	1.1.0	MIT	Automatically installs npm, bower, tsd, typings, composer and pip packages/dependencies if the relative configurations are found in the gulp file stream respectively
gulp-json-modify	1.0.2	MIT	Replaces data in a JSON file with the specified values.
gulp-minify	3.1.0	ISC	Minifies JavaScript files.
gulp-rename	1.4.0	MIT	Plugin to rename files.
gulp-replace-task	0.11.0	MIT	Replaces text patterns.
gulp-strip-css-comments	2.0.0	MIT	Strips comments from CSS files.
gulp-strip-comments	2.5.2	MIT	Removes both single and multi-line comments from JSON, JavaScript and CSS/Text, without changing the layout / formatting of the original document The library does not support mixed content - HTML with JavaScript or CSS in it
gulp-sourcemaps	2.6.4	ISC	Writes inline source maps embedded in the source file.
require-dir	1.1.0	MIT	Node helper to require() directories. The directory's files are examined, and each one that can be required is required and returned as part of a

			hash from that file's basename to its exported contents.
run-sequence	2.2.1	MIT	Runs a sequence of gulp tasks in the specified order.
through2	3.0.0	MIT	A tiny wrapper around Node.js streams. Used to make a stream out of a function to set up the prototype chain properly.
gulp-uglify	3.0.1	MIT	Minifies JavaScript with UglifyJS3 and changes names of variables and functions.
gulp-uglify-es	1.0.4	MIT	Gulp stream to uglify with “terser” (ES6 supported).
minimatch	3.0.4	ISC	This is the matching library used internally by npm. It works by converting glob expressions into JavaScript RegExp objects.
@types/graceful-fs	4.1.2	MIT	This package contains type definitions for graceful-fs.

# Descriptive Statistics

## Tables

Table A2: Descriptive statistics of nominal variables

<b>Descriptive statistics of nominal variables</b>			
<b>Variables</b>	<b>Categories</b>	<b>N</b>	<b>f%</b>
Gender	Male	61	46.56
	Female	70	53.44
Age	16-26	28	21.37
	27-37	83	63.36
	38-48	15	11.45
	≥49	5	3.82
Affective Feedback	Yes	66	50.38
	No	65	49.62
Budget adherence 1st week	Success	41	31.30
	Fail	90	68.70
Budget adherence 2nd week	Success	69	52.67
	Fail	62	47.33
Budget adherence 3rd week	Success	101	77.10
	Fail	30	22.90
Budget goal	Yes	39	29.77
	No	92	70.23

N: Frequency

f%: Relative frequency

Table A3: Age groups based on Gender

		<b>Age groups based on Gender</b>			
<b>Age</b>		<b>Gender</b>		<b>Total</b>	
		<b>Male</b>	<b>Female</b>		
16-26	Count	10	18	28	
	% within Age	35.71%	64.49%	100.0%	
	% within Sample	7.63%	13.74%	21.37%	
27-37	Count	40	43	83	
	% within Age	48.19%	51.81%	100.0%	
	% within Sample	30.53%	32.82%	63.36%	
38-48	Count	8	7	15	
	% within Age	53.33%	46.67%	100.0%	
	% within Sample	6.11%	5.34%	11.45%	
≥49	Count	3	2	5	
	% within Age	60.0%	40.0%	100.0%	
	% within Sample	2.29%	1.53%	3.82%	
Total	Count	61	70	131	
	% within Sample	46.56%	53.44%	100.0%	

Table A4: Feedback groups based on Gender

		<b>Feedback groups based on Gender</b>			
<b>Gender</b>		<b>Affective Feedback</b>		<b>Total</b>	
		<b>Yes</b>	<b>No</b>		
Male	Count	31	30	61	
	% within Gender	50.82%	49.18%	100.0%	
	% within Sample	23.66%	22.90%	46.56%	
Female	Count	35	35	70	
	% within Gender	50.0%	50.0%	100.0%	
	% within Sample	26.72%	26.72%	53.44%	
Total	Count	66	65	131	
	% within Sample	50.38%	49.62%	100.0%	

Table A5: Feedback groups based on Age

		<b>Feedback groups based on Age</b>			
		<b>Affective Feedback</b>			
<b>Age</b>		Yes	No	Total	
16-26	Count	16	12	28	
	% within Age	57.14%	42.86%	100.0%	
	% within Sample	12.21%	9.16%	21.37%	
27-37	Count	38	45	83	
	% within Age	45.78%	54.22%	100.0%	
	% within Sample	29.01%	34.35%	63.36%	
38-48	Count	8	7	15	
	% within Age	53.33%	46.67%	100.0%	
	% within Sample	6.11%	5.34%	11.45%	
≥49	Count	3	2	5	
	% within Age	60.0%	40.0%	100.0%	
	% within Sample	2.29%	1.53%	3.82%	
Total	Count	66	65	131	
	% within Sample	50.38%	49.62%	100.0%	

Table A6: Budget adherence of first week in correlation to affective feedback

		<b>Budget adherence 1st week in correlation to Affective Feedback</b>			
		<b>Affective Feedback</b>			
<b>Budget adherence 1st week</b>		Yes	No	Total	
Success	Count	24	17	41	
	% within Budget adherence 1st week	58.54%	41.46%	100.0%	
	% within sample	18.32%	12.98%	31.30%	
Fail	Count	42	48	90	
	% within Budget adherence 1st week	46.67%	53.33%	100.0%	
	% within sample	32.06%	36.64%	68.70%	
Total	Count	66	65	131	
	% within sample	50.38%	49.62%	100.0%	

Table A7: Budget adherence of second week in correlation to affective feedback

<b>Budget adherence 2nd week in correlation to Affective Feedback</b>			<b>Affective Feedback</b>		
<b>Budget adherence 2nd week</b>	Success	Count	Yes	No	Total
		Count	40	29	69
		% within Budget adherence 2nd week	57.97%	42.03%	100.0%
		% within sample	30.53%	22.14%	52.67%
	Fail	Count	26	36	62
		% within Budget adherence 2nd week	41.94%	58.06%	100.0%
		% within sample	19.85%	27.48%	47.33%
	Total	Count	66	65	131
		% within sample	50.38%	49.62%	100.0%

Table A8: Budget adherence of third week in correlation to affective feedback

<b>Budget adherence 3rd week in correlation to Affective Feedback</b>			<b>Affective Feedback</b>		
<b>Budget adherence 3rd week</b>	Success	Count	Yes	No	Total
		Count	55	46	101
		% within Budget adherence 3rd week	54.46%	45.54%	100.0%
		% within sample	41.98%	35.11%	77.10%
	Fail	Count	11	19	30
		% within Budget adherence 3rd week	36.67%	63.33%	100.0%
		% within sample	8.40%	14.50%	22.90%
	Total	Count	66	65	131
		% within sample	50.38%	49.62%	100.0%

Table A9: Budget adherence 1st week in correlation to Gender - Affective Feedback

<b>Budget adherence 1st week in correlation to Gender - Affective Feedback</b>			
<b>Gender - Affective Feedback</b>	<b>Budget adherence 1st week</b>		
	<b>Success</b>	<b>Fail</b>	<b>Total</b>
<b>Male - Yes</b>	12	19	31
% within Budget adherence 1st week	29.27%	21.11%	50.38%
% feedback group	18.18%	28.79%	46.97%
% within sample	9.16%	14.50%	23.66%
<b>Female - Yes</b>	12	23	35
% within Budget adherence 1st week	29.27%	25.56%	54.82%
% feedback group	18.18%	34.85%	53.03%
% within sample	9.16%	17.56%	26.72%
<b>Male - No</b>	7	23	30
% within Budget adherence 1st week	17.07%	25.56%	42.63%
% feedback group	10.77%	35.38%	46.15%
% within sample	5.34%	17.56%	22.90%
<b>Female - No</b>	10	25	35
% within Budget adherence 1st week	24.39%	27.78%	52.17%
% feedback group	15.38%	38.46%	53.85%
% within sample	7.63%	19.08%	26.72%
<b>Total</b>	41	90	131
% within sample	31.3%	68.7%	100.0%



Table A10: Budget adherence 2nd week in correlation to Gender - Affective Feedback

<b>Budget adherence 2nd week in correlation to Gender - Affective Feedback</b>			
<b>Gender - Affective Feedback</b>	<b>Budget adherence 2nd week</b>		
	<b>Success</b>	<b>Fail</b>	<b>Total</b>
<b>Male - Yes</b>	20	11	31
% within Budget adherence 2nd week	28.99%	17.74%	46.73%
% feedback group	30.30%	16.67%	46.97%
% within sample	15.27%	8.40%	23.66%
<b>Female - Yes</b>	20	15	35
% within Budget adherence 2nd week	28.99%	24.19%	53.18%
% feedback group	30.30%	22.73%	53.03%
% within sample	15.27%	11.45%	26.72%
<b>Male - No</b>	13	17	30
% within Budget adherence 2nd week	18.84%	27.42%	46.26%
% feedback group	20.00%	26.15%	46.15%
% within sample	9.92%	12.98%	22.90%
<b>Female - No</b>	16	19	35
% within Budget adherence 2nd week	23.19%	30.65%	53.83%
% feedback group	24.62%	29.23%	53.85%
% within sample	12.21%	14.50%	26.72%
<b>Total</b>	69	62	131
% within sample	57.67%	47.33%	100.0%

Table A11: Budget adherence 3rd week in correlation to Gender - Affective Feedback

<b>Budget adherence 3rd week in correlation to Gender - Affective Feedback</b>			
<b>Gender - Affective Feedback</b>	<b>Budget adherence 3rd week</b>		
	<b>Success</b>	<b>Fail</b>	<b>Total</b>
<b>Male - Yes</b>	26	5	31
% within Budget adherence 3rd week	25.74%	16.67%	42.41%
% feedback group	39.39%	7.58%	46.97%
% within sample	19.85%	3.82%	23.66%
<b>Female - Yes</b>	29	6	35
% within Budget adherence 3rd week	28.71%	20.00%	48.71%
% feedback group	43.94%	9.09%	53.03%
% within sample	22.14%	4.58%	26.72%
<b>Male - No</b>	21	9	30
% within Budget adherence 3rd week	20.79%	30.00%	50.79%
% feedback group	32.31%	13.85%	46.15%
% within sample	16.03%	6.87%	22.90%
<b>Female - No</b>	25	10	35
% within Budget adherence 3rd week	24.75%	33.33%	58.09%
% feedback group	38.46%	15.38%	53.85%
% within sample	19.08%	7.63%	26.72%
<b>Total</b>	101	30	131
% within sample	77.1%	22.9%	100.0%

Table A12: Budget adherence 1st week in correlation to Age - Affective Feedback

<b>Budget adherence 1st week in correlation to Age - Affective Feedback</b>			
<b>Age - Affective Feedback</b>	<b>Budget adherence 1st week</b>		
	<b>Success</b>	<b>Fail</b>	<b>Total</b>
<b>16-26 - Yes</b>	8	8	16
% within Budget adherence 1st week	19.51%	8.89%	28.40%
% feedback group	12.12%	12.12%	24.24%
% within sample	6.11%	6.11%	12.21%
<b>27-37 - Yes</b>	13	25	38
% within Budget adherence 1st week	31.71%	27.78%	59.49%
% feedback group	19.70%	37.88%	57.58%
% within sample	9.92%	19.08%	29.01%
<b>38-48 - Yes</b>	1	8	9
% within Budget adherence 1st week	2.44%	8.89%	11.33%
% feedback group	1.52%	12.12%	13.64%
% within sample	0.76%	6.11%	6.87%
<b>≥49 - Yes</b>	2	1	3
% within Budget adherence 1st week	4.88%	1.11%	5.99%
% feedback group	3.03%	1.52%	4.55%
% within sample	1.53%	0.76%	2.29%
<b>16-26 - No</b>	1	11	12
% within Budget adherence 1st week	2.44%	12.22%	14.66%
% feedback group	1.54%	16.92%	18.46%
% within sample	0.76%	8.40%	9.16%
<b>27-37 - No</b>	12	33	45
% within Budget adherence 1st week	29.27%	36.67%	65.93%
% feedback group	18.46%	50.77%	69.23%
% within sample	9.16%	25.19%	34.35%
<b>38-48 - No</b>	3	3	6
% within Budget adherence 1st week	7.32%	3.33%	10.65%
% feedback group	4.62%	4.62%	9.23%
% within sample	2.29%	2.29%	4.58%
<b>≥49 - No</b>	1	1	2
% within Budget adherence 1st week	2.44%	1.11%	3.55%
% feedback group	1.54%	1.54%	3.08%
% within sample	0.76%	0.76%	1.53%
<b>Total</b>	41	90	131
% within sample	31.30%	68.70%	100.0%

Table A13: Budget adherence 2nd week in correlation to Age - Affective Feedback

<b>Budget adherence 2nd week in correlation to Age - Affective Feedback</b>			
<b>Age - Affective Feedback</b>	<b>Budget adherence 2nd week</b>		
	<b>Success</b>	<b>Fail</b>	<b>Total</b>
<b>16-26 - Yes</b>	10	6	16
% within Budget adherence 2nd week	14.49%	9.68%	24.17%
% feedback group	15.15%	9.09%	24.24%
% within sample	7.63%	4.58%	12.21%
<b>27-37 - Yes</b>	22	16	38
% within Budget adherence 2nd week	31.88%	25.81%	57.69%
% feedback group	33.33%	24.24%	57.58%
% within sample	16.79%	12.21%	29.01%
<b>38-48 - Yes</b>	6	3	9
% within Budget adherence 2nd week	8.70%	4.84%	13.53%
% feedback group	9.09%	4.55%	13.64%
% within sample	4.58%	2.29%	6.87%
<b>≥49 - Yes</b>	2	1	3
% within Budget adherence 2nd week	2.90%	1.61%	4.51%
% feedback group	3.03%	1.52%	4.55%
% within sample	1.53%	0.76%	2.29%
<b>16-26 - No</b>	4	8	12
% within Budget adherence 2nd week	5.80%	12.90%	18.70%
% feedback group	6.15%	12.31%	18.46%
% within sample	3.05%	6.11%	9.16%
<b>27-37 - No</b>	21	24	45
% within Budget adherence 2nd week	30.43%	38.71%	69.14%
% feedback group	32.31%	36.92%	69.23%
% within sample	16.03%	18.32%	34.35%
<b>38-48 - No</b>	3	3	6
% within Budget adherence 2nd week	4.35%	4.84%	9.19%
% feedback group	4.62%	4.62%	9.23%
% within sample	2.29%	2.29%	4.58%
<b>≥49 - No</b>	1	1	2
% within Budget adherence 2nd week	1.45%	1.61%	3.06%
% feedback group	1.54%	1.54%	3.08%
% within sample	0.76%	0.76%	1.53%
<b>Total</b>	69	62	131
% within sample	52.67%	47.33%	100.0%

Table A14: Budget adherence 3rd week in correlation to Age - Affective Feedback

<b>Budget adherence 3rd week in correlation to Age - Affective Feedback</b>			
<b>Age - Affective Feedback</b>	<b>Budget adherence 3rd week</b>		
	<b>Success</b>	<b>Fail</b>	<b>Total</b>
<b>16-26 - Yes</b>	11	5	16
% within Budget adherence 3rd week	10.89%	16.67%	27.56%
% feedback group	16.67%	7.58%	24.24%
% within sample	8.40%	3.82%	12.21%
<b>27-37 - Yes</b>	34	4	38
% within Budget adherence 3rd week	33.66%	13.33%	47.00%
% feedback group	51.52%	6.06%	57.58%
% within sample	25.95%	3.05%	29.01%
<b>38-48 - Yes</b>	7	2	9
% within Budget adherence 3rd week	6.93%	6.67%	13.60%
% feedback group	10.61%	3.03%	13.64%
% within sample	5.34%	1.53%	6.87%
<b>≥49 - Yes</b>	3	0	3
% within Budget adherence 3rd week	2.97%	0.00%	2.97%
% feedback group	4.55%	0.00%	4.55%
% within sample	2.29%	0.00%	2.29%
<b>16-26 - No</b>	7	5	12
% within Budget adherence 3rd week	6.93%	16.67%	23.60%
% feedback group	10.77%	7.69%	18.46%
% within sample	5.34%	3.82%	9.16%
<b>27-37 - No</b>	31	14	45
% within Budget adherence 3rd week	30.69%	46.67%	77.36%
% feedback group	47.69%	21.54%	69.23%
% within sample	23.66%	10.69%	34.35%
<b>38-48 - No</b>	6	0	6
% within Budget adherence 3rd week	5.94%	0.00%	5.94%
% feedback group	9.23%	0.00%	9.23%
% within sample	4.58%	0.00%	4.58%
<b>≥49 - No</b>	2	0	2
% within Budget adherence 3rd week	1.98%	0.00%	1.98%
% feedback group	3.08%	0.00%	3.08%
% within sample	1.53%	0.00%	1.53%
<b>Total</b>	101	30	131
% within sample	77.10%	22.90%	100.00%

Table A15: Correlation of “Expense Amount” with the variables “Affective Feedback”, “Gender - Affective Feedback”, and “Age - Affective Feedback”

<b>Correlation of “Expense Amount” with the variables “Affective Feedback”, “Gender - Affective Feedback”, and “Age - Affective Feedback”</b>			
<b>Variables</b>	<b>Affective Feedback (t-test)</b>	<b>Gender - Affective Feedback (ANOVA)</b>	<b>Age - Affective Feedback (Kruskal-Wallis)</b>
Expense amount 1st week	0.796	0.526	0.343
Expense amount 2nd week	0.363	0.815	0.979
Expense amount 3rd week	0.062	0.285	0.438

## Graphs

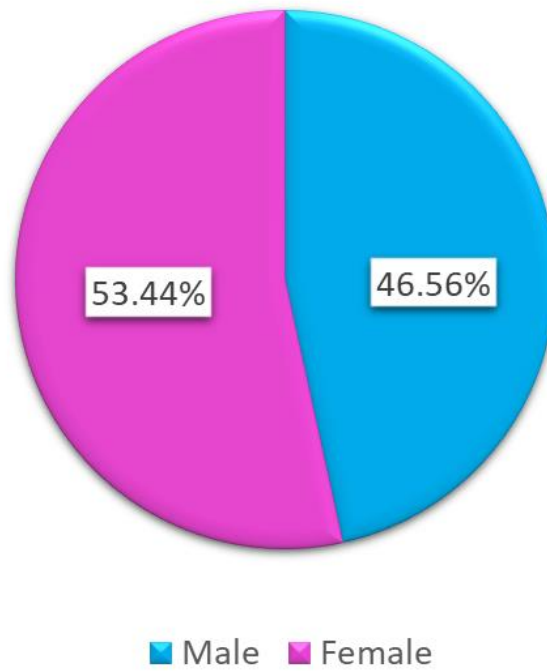


Figure A1: Graph regarding the gender of participants

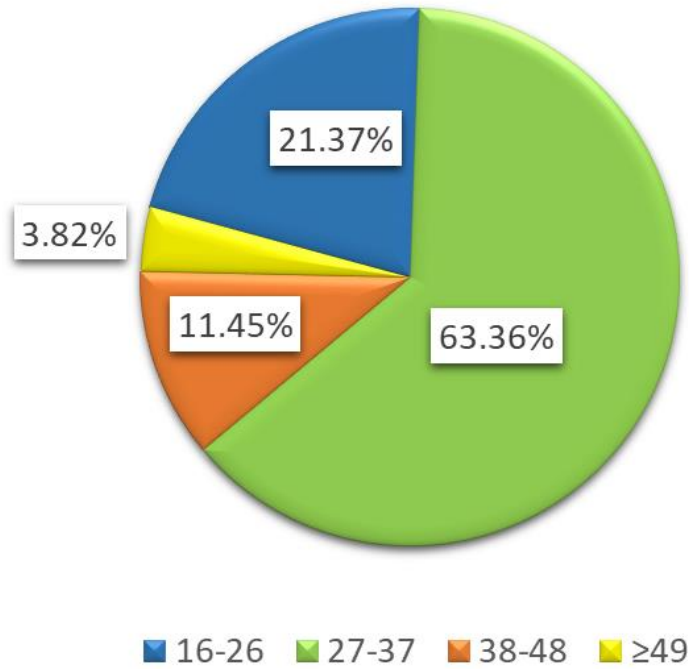


Figure A2: Graph regarding the age of participants

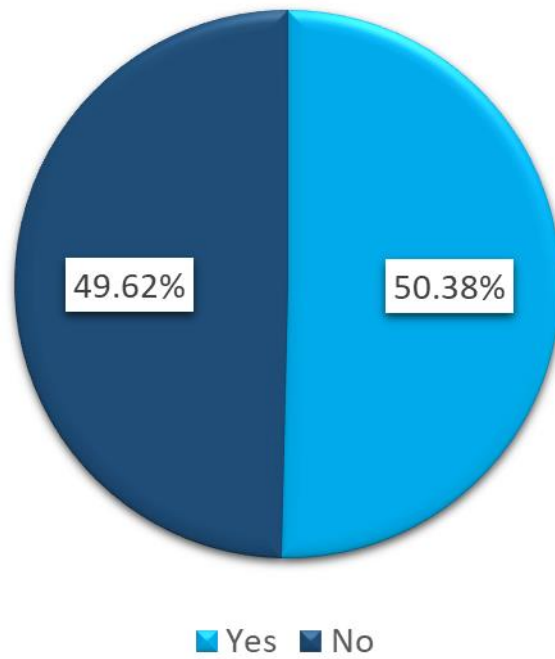


Figure A3: Graph regarding the participants who received affective feedback

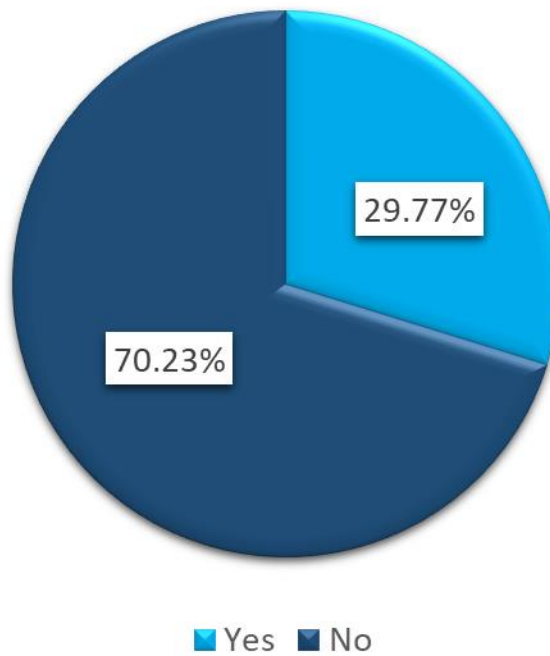


Figure A4: Graph regarding the participants who had set at least one budget goal over the three-week period



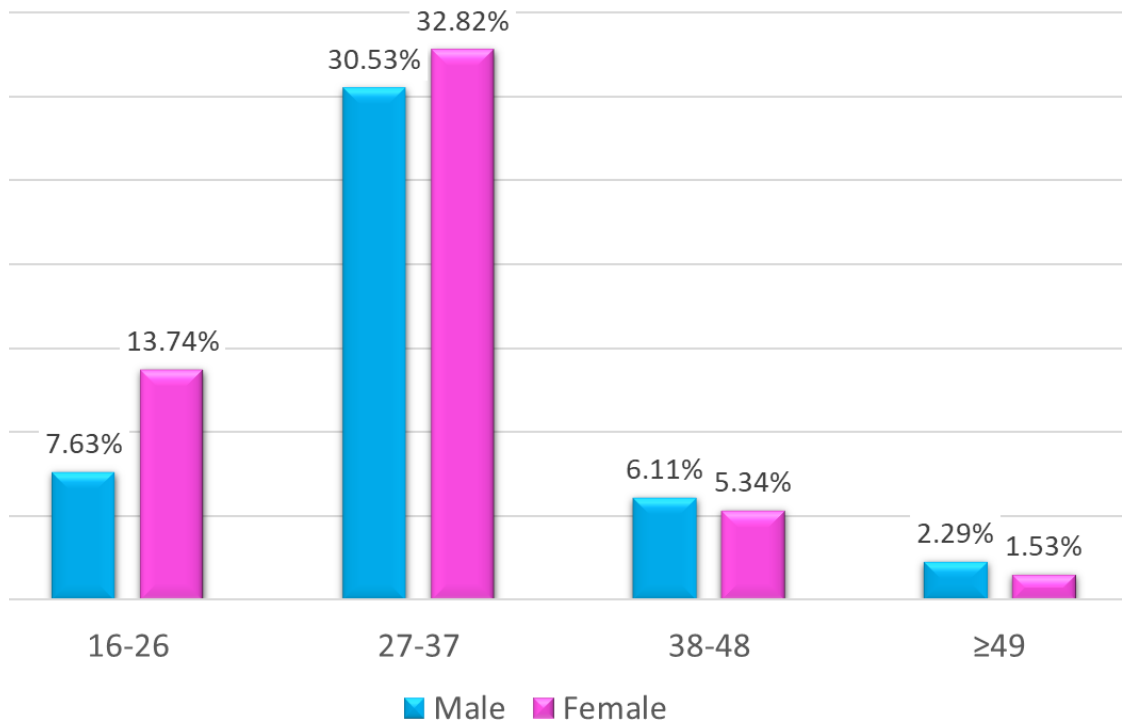


Figure A5: Graph regarding the age groups based on gender

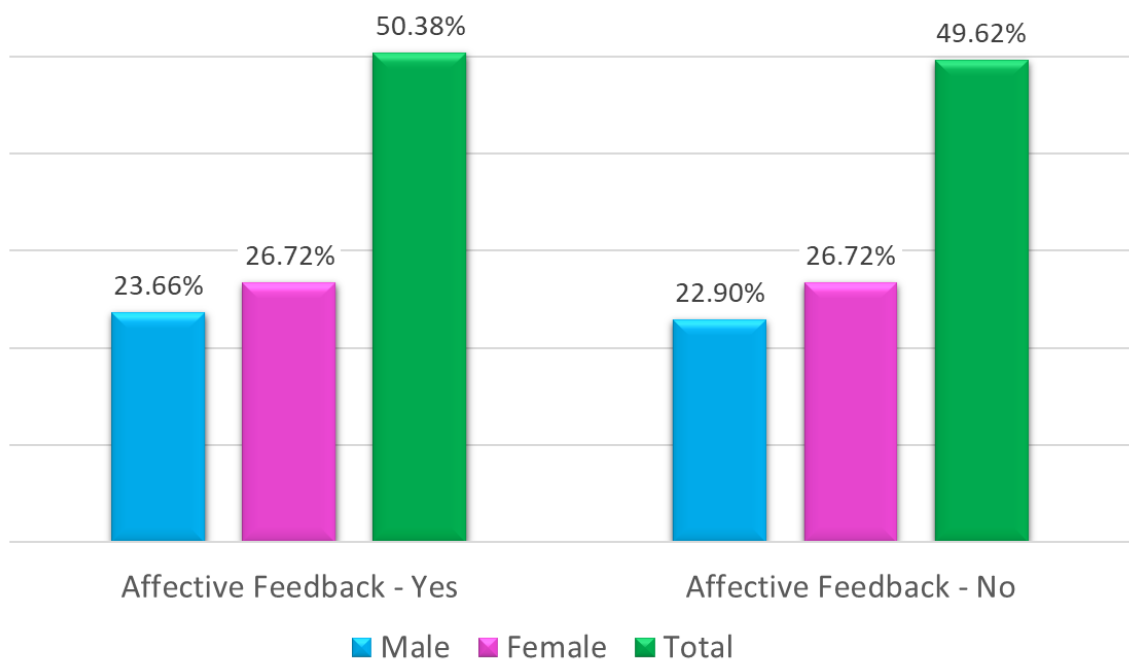


Figure A6: Graph regarding the feedback groups based on gender

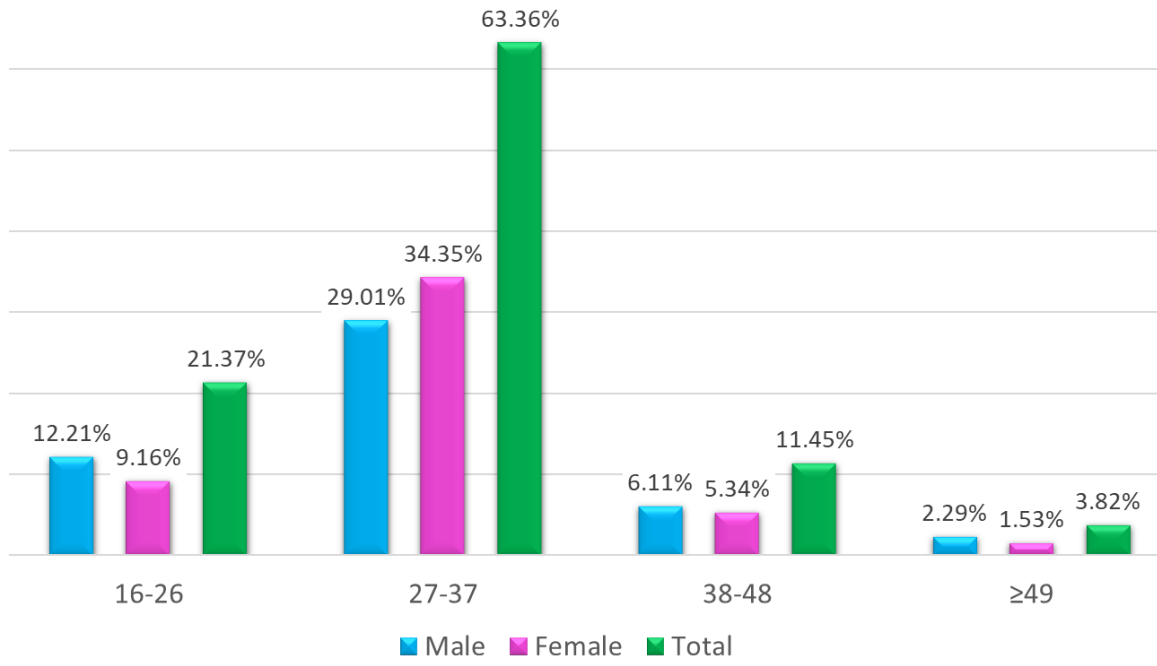


Figure A7: Graph regarding the feedback groups based on age

## List of Tables

Table 1: Functional requirements of the <i>Budget Manager</i> prototype system.....	27
Table 2: Non-functional requirements of the <i>Budget Manager</i> prototype system .....	29
Table 3: ISO 9126 quality characteristics.....	29
Table 4: Description of the entities in the ERD .....	38
Table 5: Description of interfaces for unauthorized users .....	53
Table 6: Description of interfaces for authorized users .....	56
Table 7: Descriptive statistics of participants who defined at least one budget goal in each week .....	100
Table 8: Descriptive statistics of quantitate variables .....	104
Table 9: Correlation of “Budget Adherence” with the variables “Affective Feedback”, “Gender - Affective Feedback”, and “Age - Affective Feedback” .	105
Table 10: Confidence Intervals 95% for percentages of successful budget adherence.....	105
Table 11: Multiple comparisons for expense amount via Bonferonni.....	106
Table A1: Gulp packages.....	129
Table A2: Descriptive statistics of nominal variables .....	132
Table A3: Age groups based on Gender.....	133
Table A4: Feedback groups based on Gender .....	133
Table A5: Feedback groups based on Age .....	134
Table A6: Budget adherence of first week in correlation to affective feedback	134
Table A7: Budget adherence of second week in correlation to affective feedback .....	135
Table A8: Budget adherence of third week in correlation to affective feedback .....	135
Table A9: Budget adherence 1st week in correlation to Gender - Affective Feedback.....	136
Table A10: Budget adherence 2nd week in correlation to Gender - Affective Feedback.....	137

Table A11: Budget adherence 3rd week in correlation to Gender - Affective Feedback .....	138
Table A12: Budget adherence 1st week in correlation to Age - Affective Feedback .....	139
Table A13: Budget adherence 2nd week in correlation to Age - Affective Feedback .....	140
Table A14: Budget adherence 3rd week in correlation to Age - Affective Feedback .....	141
Table A15: Correlation of "Expense Amount" with the variables "Affective Feedback", "Gender - Affective Feedback", and "Age - Affective Feedback" ..	142

## List of Figures

Figure 1: Eight frames of the affective feedback card that was delivered to users who managed to stay on budget .....	32
Figure 2: Eight frames of the affective feedback card that was delivered to users who did not manage to stay on budget .....	33
Figure 3: Feedback on budget for users in the control group who successfully stayed on budget.....	34
Figure 4: Feedback on budget for users in the control group who failed to stay on budget.....	34
Figure 5: Feedback on budget for users in the experimental group who successfully stayed on budget .....	35
Figure 6: Feedback on budget for users in the experimental group who failed to stay on budget .....	35
Figure 7: Architecture of the <i>Budget Manager</i> prototype system.....	36
Figure 8: ERD (Entity Relationship Diagram) of the system's database.....	38
Figure 9: Structure of table <i>user</i> .....	39
Figure 10: A sample containing five random rows of the <i>user</i> table .....	39
Figure 11: Structure of table <i>budget</i> .....	40
Figure 12: A sample containing five random rows of the <i>budget</i> table .....	40
Figure 13: Structure of table <i>category</i> .....	40
Figure 14: Basic categories of expenses .....	41
Figure 15: Structure of table <i>expense</i> .....	42
Figure 16: A sample containing five random rows of the <i>expense</i> table.....	42
Figure 17: Structure of table <i>feedback</i> .....	43
Figure 18: A sample containing five random rows of the <i>feedback</i> table .....	43
Figure 19: Structure of table <i>goal</i> .....	44
Figure 20: A sample containing five random rows of the <i>goal</i> table .....	44
Figure 21: Structure of table <i>log_activity</i> .....	45
Figure 22: A sample containing five random rows of the <i>log_activity</i> table .....	45

Figure 23: Structure of table <i>log_user</i> .....	46
Figure 24: A sample containing five random rows of the <i>log_user</i> table.....	46
Figure 25: Structure of table <i>log_category</i> .....	46
Figure 26: A sample containing five random rows of the <i>log_category</i> table .....	47
Figure 27: Structure of table <i>log_budget</i> .....	48
Figure 28: Structure of table <i>log_expense</i> .....	48
Figure 29: Structure of table <i>log_goal</i> .....	48
Figure 30: A sample containing five random rows of the <i>log_budget</i> table .....	49
Figure 31: A sample containing five random rows of the <i>log_expense</i> table .....	49
Figure 32: A sample containing five random rows of the <i>log_goal</i> table .....	49
Figure 33: Class diagram of the php models .....	51
Figure 34: Interfaces for unauthorized users .....	52
Figure 35: Use cases of interaction between the system and unauthorized users .....	54
Figure 36: Interfaces for authorized users.....	55
Figure 37: Use cases of interaction between the system and authorized users, part 1.....	58
Figure 38: Use cases of interaction between the system and authorized users, part 2.....	59
Figure 39: Use cases of interaction between the system and authorized users, part 3.....	59
Figure 40: Use cases of interaction between the system and authorized users, part 4.....	60
Figure 41: Percentage of programming languages used.....	61
Figure 42: The Software Development Cycle (Source: <a href="https://images.mendix.com/wp-content/uploads/Artboard-1@2x-701x700.png">https://images.mendix.com/wp-content/uploads/Artboard-1@2x-701x700.png</a> ).....	62
Figure 43: Creating table <i>expense</i> with SQL .....	63
Figure 44: Adding constraints for tables <i>expense</i> and <i>feedback</i> .....	64
Figure 45: Inserting predefined categories of expenses .....	64
Figure 46: Functions to sanitize string, date, and id values.....	65

Figure 47: Functions to sanitize float and category values .....	65
Figure 48: Functions to allocate users into the experimental and control groups .....	66
Figure 49: Setting session parameters based on cookie existence.....	67
Figure 50: Accessing the LoggedUser class and executing its functions .....	67
Figure 51: Accessing the User class and executing its functions .....	68
Figure 52: Expenses of a user returned in JSON format from the server.....	69
Figure 53: Response returned in JSON format from the server after successfully adding an expense.....	69
Figure 54: Personal information returned in JSON format from the server for a user that is signed into the system .....	69
Figure 55: Personal information returned in JSON format from the server for a user that is not signed into the system .....	70
Figure 56: PWA logo (Source: <a href="https://responsivedesign.is/wp-content/uploads/2018/08/PWA-Progressive-Web-App-Logo-800x460.png">https://responsivedesign.is/wp-content/uploads/2018/08/PWA-Progressive-Web-App-Logo-800x460.png</a> ) .....	74
Figure 62: The loadStyle function – loading dynamically CSS code for interfaces .....	81
Figure 63: The loadScript function – loading dynamically JavaScript code for interfaces .....	81
Figure 64: The application on mobile phone and tablet with different viewports and orientations.....	82
Figure 65: Enhancing the meaning of the content with “Font Awesome” icons.	84
Figure 66: Implementation of blob-select, md-date-time-picker, and grudus timepicker plugins/libraries .....	85
Figure 67: Charts regarding the expenses of a week with the use of Chart.js...	85
Figure 68: Implementation of the Fetch API to send data to the server.....	86
Figure 69: Implementation of the Fetch API to retrieve data from the server ....	87
Figure 70: The manifest.json file.....	88
Figure 71: Manifest tab of Chrome DevTools.....	88
Figure 72: The “Add to Home Screen” feature of the “Budget Manager” PWA.	89
Figure 73: Launching the “Budget Manager” PWA from the home screen.....	90

Figure 74: The registered Service Worker as displayed in Chrome DevTools...	91
Figure 75: Static files saved in the cache storage of the browser by the Service Worker.....	91
Figure 76: Images saved in the cache storage of the browser by the Service Worker.....	92
Figure 77: User experience with and without Service Workers .....	93
Figure 78: Array of CSS resources of the “Home” interface.....	95
Figure 79: The CSS file required from the browser for the presentation of the “Home” interface.....	95
Figure 80: Parameters of the application required for distribution.....	96
Figure 81: Graph regarding the budget adherence of participants.....	100
Figure 82: Graph regarding the budget adherence in correlation to affective feedback.....	101
Figure 83: Graph regarding budget adherence in correlation to the feedback groups based on gender.....	101
Figure 85: Graph regarding budget adherence in correlation to users who received affective feedback based on age .....	102
Figure 86: Graph regarding budget adherence in correlation to users who did not receive affective feedback based on age .....	103
Figure 87: Graph of descriptive statistics of quantitate variables .....	103
Figure A1: Graph regarding the gender of participants.....	143
Figure A2: Graph regarding the age of participants.....	143
Figure A3: Graph regarding the participants who received affective feedback	144
Figure A4: Graph regarding the participants who had set at least one budget goal over the three-week period.....	144
Figure A5: Graph regarding the age groups based on gender .....	145
Figure A6: Graph regarding the feedback groups based on gender.....	145
Figure A7: Graph regarding the feedback groups based on age .....	146



