Summer 5-2019

B-Quest: Experience Your Gift

Mowhile Geek

Ailem Garcia

Ronald Torres

Juan Martin

Johan Quiroz

See next page for additional authors

Author

Mowhile Geek, Ailem Garcia, Ronald Torres, Juan Martin, Johan Quiroz, Robert Flores, and Maria F. Torres

B-Quest

Experience your gift

Module Title: Applied Technology Group Project

Group: Mowhile Geek

Team members:

Ailem Garcia	2016431
Ronald Torres	2016432
Juan Martin	2016435
Johan Quiroz	2016337
Robert Flores	2016313
Maria F. Torres	2016258

CONTENTS

CHAPTER I
INTRODUCTION

OBJECTIVES

GENERAL

DESIGN AND BUILD A WEB APP AND A NATIVE APPLICATION AIMED AT TECHNOPHILES FOR ORGANISING QUESTS LIKE EVENTS.

SPECIFICS

1. To implement the use of programming languages such as Java and JavaScript to create a Web App and Native application.
2. To develop new skills in the Integrated Development Environment Android Studio in order to learn how to create Android Apps.
3. To implement a secure Database 'back end' for the application to store/maintain user details.

INTRODUCTION

ANALYSIS OF SITUATION

Birthdays are special events for everybody and for most of people it is an important date in their life, it is the day where they feel honoured and appreciated by their friends, family, acquaintances…

The main reason for creating an application dedicated to make from the Birthday event a fun game, is generated after bearing in mind how the tradition of giving presents to honour a birthday person is something important in many people's live.

Another aspect considered during the conception of this application is how fast the world is moving, and how sometimes it could be beneficial to have different options to make from an event something different and special.

Under this knowledge it is importantly understood that creating an app is a work that requires significant amounts of time and attention. Something that in a fast pace world like this can be considered a luxury. Nevertheless, the opportunity to offer something different to a specific target of users, who enjoy gaming options, is an interesting and worthwhile challenge to be taken.

As a first step, a research about events organisation apps was done. The outcome showed that there is a lack of applications that can offer the option to organise an event and deliver in a dynamic way a gift for the birthday person.

From this first research some interesting facts were found, such as:

On a database of 10940 mobile devices sample, Events apps for Android OS take around 1.4% of the worldwide market according to the statistics in Favourite Smartphone App 2018 ((2018) S. , Statista.com, n.d.)

On IOS there is no even a sector for Events, the most similar will be Lifestyle. Lifestyle apps for IOS take around 8.32% of the worldwide market according to the statistics in Popular Categories In the App Store ((2018) S. , Statista.com, n.d.). Nevertheless, this sector is very wide, doesn't show a real statistics number for Events apps.

As a second step a researching for applications that could provide benefits for Birthday events was also performed (Fuenmayor, n.d.) (Google, n.d.) (Bullas, n.d.) (2018, n.d.) ((2018). W. , n.d.) ((2018). S. , n.d.) ((2018) G. , n.d.) ([online], Bloggingwizard.com. (2018). [online], n.d.) ([online], Bloggingwizard.com. (2018). [online] , n.d.). Some of the most similar were:

ADDWISH

This app allows people to create a list of gifts that they would like to receive, once created users can share the list with their contacts so they are aware of the user preferences on special dates. This application works perfectly in Spain, integrating stores so that the user experience for the app is rewarding. (Fuenmayor, n.d.)

GLEAM

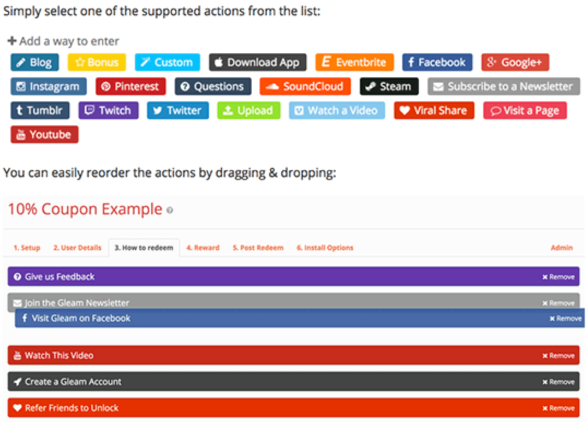
Gleam ((2018) G. , n.d.) is a growth marketing platform that uses 4 different apps to put in place different campaigns. Through them users are instructed to complete individual tasks such as follow a social media account or just answer a question in order to be included in a contest for a giveaway.

"One unique feature of Gleam is that, unlike many other platforms, there is verification of each action a user takes. Gleam tracks users on whether or not they complete the actions for your giveaway." (Bloggingwizard.com, 2018)

The app also provides better interaction with the users by embedding YouTube videos showing what they will win or how they can enter your giveaway.

The Business Growth Platform

Marketing apps designed to help you grow your business.



Simply select one of the supported actions from the list:

You can easily reorder the actions by dragging & dropping:

10% Coupon Example

After considering the first research and the knowledge about similar apps, B-Quest was born. An idea that arises from a team that was working on a research project for College. The mind after this idea is Ronald Torres, who came up with the proposal for an application that would offer the opportunity to give a birthday gift in a fun, attractive, dynamic and above all practical way.

After coming up with the idea the next step was performed and it consisted in doing a SWOT analysis.

During this stage the Strengths, Weaknesses, Opportunities and Threats for this project were evaluated by answering the following questions (Team, n.d.) (Anna, n.d.):

Strengths

- What advantages does your team have?
- What unique or lowest-cost resources can you draw upon that others can't?
- What do people in your market see as your strengths?
- What factors mean that you "get the sale"?
- What is your organization's **Unique Selling Proposition** (USP)?

Weaknesses

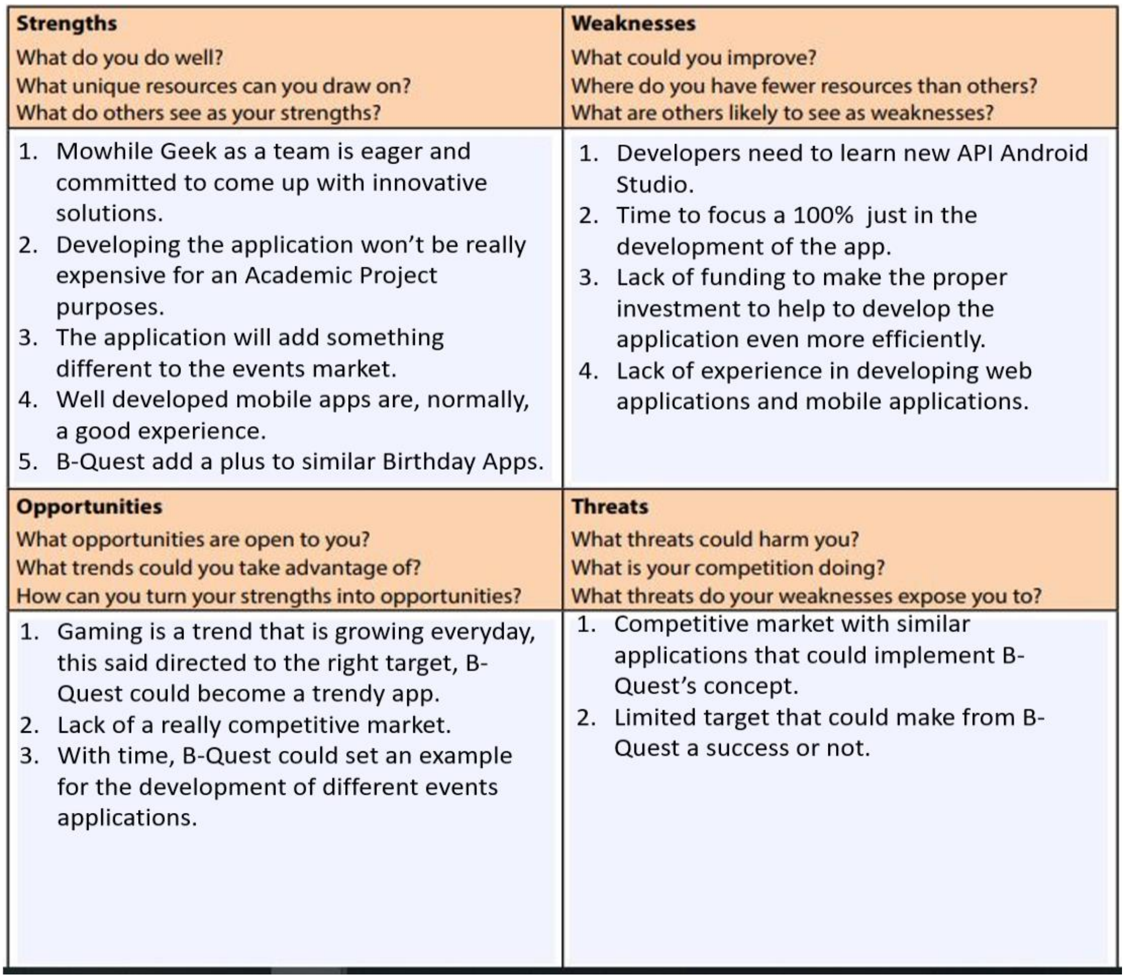
- What could you improve?
- What factors lose you sales?
- What are people in your market likely to see as weaknesses?

Opportunities

- What interesting trends are you aware of?
- What good opportunities can you spot?
- Changes in social patterns, population profiles, lifestyle changes, and so on.

Threats

- What obstacles do you face?
- What are your competitors doing?
- Could any of your weaknesses seriously threaten your business?



Strengths What do you do well? What unique resources can you draw on? What do others see as your strengths?	**Weaknesses** What could you improve? Where do you have fewer resources than others? What are others likely to see as weaknesses?
1. Mowhile Geek as a team is eager and committed to come up with innovative solutions. 2. Developing the application won't be really expensive for an Academic Project purposes. 3. The application will add something different to the events market. 4. Well developed mobile apps are, normally, a good experience. 5. B-Quest add a plus to similar Birthday Apps.	1. Developers need to learn new API Android Studio. 2. Time to focus a 100% just in the development of the app. 3. Lack of funding to make the proper investment to help to develop the application even more efficiently. 4. Lack of experience in developing web applications and mobile applications.
Opportunities What opportunities are open to you? What trends could you take advantage of? How can you turn your strengths into opportunities?	**Threats** What threats could harm you? What is your competition doing? What threats do your weaknesses expose you to?
1. Gaming is a trend that is growing everyday, this said directed to the right target, B-Quest could become a trendy app. 2. Lack of a really competitive market. 3. With time, B-Quest could set an example for the development of different events applications.	1. Competitive market with similar applications that could implement B-Quest's concept. 2. Limited target that could make from B-Quest a success or not.

Finally, as step 5 a questionnaire (QUESTIONNAIRE) will be used to survey a group of 100 people. This questionnaire is designed to establish were the trend goes regarding the idea of giving money or not for an event such as a Birthday. Once the results are analysed, it could give us a better idea about how to approach the marketing side of the application. For this reason the questionnaire is left as a last technique as the project is yet in the earlier stages of development.

B-QUEST: THE SOLUTION

The idea behind B-QUEST, is to bring together a group of people AKA Lords, that could contribute with certain amount of money to give to the birthday person, AKA The B-Hero, thus it can cash it and enjoys it in whatever best way it suits it. But, is not as simple as deposit and claiming the money. There is always a catch.

With B-Quest, guests will have as a duty to choose default quests for the B-Hero, who in turn will need to successfully meet each of these quest to claim the ultimate prize.

Each quest will have to be harmless and definitely fun for all the involved, including the B-Hero. The B-Hero will have to provide proof that all tasks were carried out effectively according to the requirements of the group. Evidence such as found treasures, pictures, videos… are just some of the ideas that are on mind in order to prove that the tasks were completed successfully.

Each quest will have to be reviewed and approved by a Chief, which will be the creator of the treasure hunt and the decision maker of the group.

(Please refer to Glossary for more information about the designated keywords for this application)

IMPLEMENT A WORKING PROTOTYPE

Prototyping is an iterative analysis technique in which users are actively involved in the mocking-up of screens and reports. The purpose of a prototype is to show people the possible designs for the user interface of an application.

There are 3 steps involved for Prototyping:

1) Determine the needs of your users.

The requirements of the users drive the development of your prototype as they define the business objects that your system must support. You can gather these requirements in interviews, in CRC (class responsibility collaborator) modelling sessions, in use-case modelling sessions, and in class diagramming sessions. In our case, this application is expecting to be an innovation, a new alternative way to give a birthday present to a colleague, a best friend or family member. Applying a survey could help us to know the possible needs of the user and also to understand what the user may be expecting to see on the app.

2) Build the prototype. Using a prototyping tool or high-level language we could develop the screens and reports needed by the users. The best advice during this stage of the process is to not invest a lot of time in making the code “good” because chances are high that you may just scrap your coding efforts anyway after evaluating the prototype. One of the techniques used to bring all the ideas together is the Brainstorming. It is commonly used in group settings to generate a large number of ideas about a topic through creative thinking. In our case it helps us to join up all the features that modelling the application involve.

The application basically allows the user to create an account or sing up to the app as a first step.

The new user AKA "Chief" could have the option to create a treasure hunt for the birthday friend, AKA "B-Hero". The treasure hunt describes the quests that the B-Hero will do: a series of tasks in different environments, like nature, supermarkets, shopping centre, etc. In this screen the user should be able to see the different options available to choose for the event. To complete each of the quests the B-Hero will have to upload all the activity by means of picture, video or audio as we show in one of the screen.

The Chief, has also the ability to invite a limited list of friends so they can join to each event later on. In the screen the user could see the list of friends available according with his/her list of Facebook's friend or contacts on the phone.

EVALUATE THE PROTOTYPE

After a version of the prototype is built it needs to be evaluated. The main goal is that we need to verify that the prototype meets the needs of the users: What's good about the prototype? What is bad about the prototype?-, and what's missing from the prototype?

One of the most common methods of testing an app is to arrange and set up a usability test. A usability test is often one-on-one and a crucial component in the process. A usability test exists to determine how usable your app is and not whether a user likes or dislikes an element. By implementing a usability test, we gain valuable information such as how users perform tasks, how they interact with functionality and how they interpret label names.

To implement a usability test, we decide to build Mockups that helps us to build an interactive wireframe. We consider Mockups the best choice to do faster interaction design. The whole interaction design is totally visualized. To bring the ideas to life, we uses MARVEL (MarvelApp, n.d.); Marvel is an excellent online tool which allow you to create prototypes of mobile applications and web projects.

The prototype includes the general and most important features that describe our main goal. You can visualize it HERE or in the following link: B-QUEST Prototype V1

The next steps to make the prototype fully functional it is to develop the code of the previous mockups previously indicated using Android Studio as a platform to help as to build the code and Firebase as a database to store and organize the data coming from the users.

WHAT THE PROJECT IS NOT ABOUT?

This is the initial stage of the application development and as it, it is important to take into consideration that by the end of this Academic Project the application is not expected to be a 100% functional. The main idea is to develop an advance prototype so the next step could be moving to a full working functional system.

Likewise, the prototype of this application will be designed for Android users, and will not be initially developed for IOS.

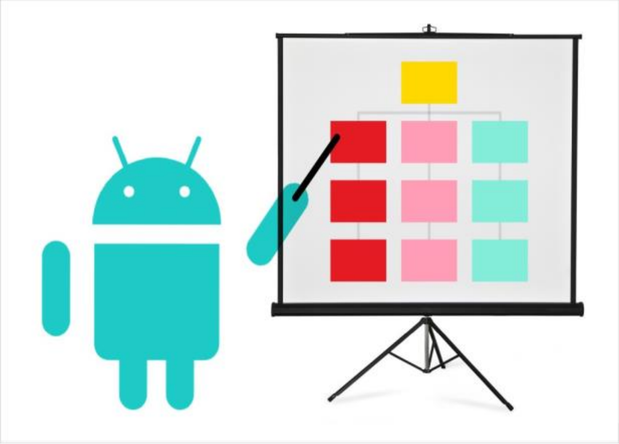
A database with encryption features will be applied to this prototype. However, the project does not involve the development of a new Database System, therefore an existing one will be implemented.

B-Quest will not be a birthday reminder app, a marketing platform or an online boutique shop. By the end of the project the B-Quest should be able to be considered an entertainment and event's organisation app.

KNOWLEDGE REQUIRED

Mobile application development contains a set of procedures involving programming and building software or applications designed for fast operating handheld devices such as smart phones and tablets. In many ways it is similar to Web application development, the difference being that mobile applications are often developed to be specifically tailored to the special functions of the device being used. For example, in the case of B-Quest for an Android platform phone, it may be advantageous to utilize the phone's accelerometer to offer a unique user experience.

Building Android applications requires a deep knowledge of programming and design. Building apps for mobile devices often requires mastery of a number of more fine distinction of concepts. Mobile devices have smaller screens, simpler processors, and – in the case of Android – many different manufacturers, meaning that Mowhile Geek as developers need to keep code flexible and account for a variety of user scenarios.



So, what does it take to Mowhile Geek to become an Android developer team?

1. **JAVA**

   Java is the language that underpins all Android development. For Mowhile Geek whose have gained most of their coding experience in this language, it will be a plus. Java, like JavaScript and Ruby, is object-oriented, but it is also stricter about the way it handles data types. Mowhile Geek as a team have to be much more thoughtful with the code, defining the types of data their applications plan to work with, and more carefully allocating scarce memory resources.

2. **ANDROID STUDIO**

   The integrated development environment (IDE) of choice for Android developers is called Android Studio. In this aspect, Mowhile Geek as a team, do not have previous experience using it. Android Studio is built on top of the well-respected IntelliJ IDE, and it comes with great out-of-the-box support for many of the most common Android SDKs.

   Android Studio also features many of the capability's developers expect of a full-featured IDE. Code completion helps make auto-complete suggestions as you type. Code debuggers let you step through the code to identify the source of errors. There are even more advanced tools like memory and CPU monitors,

   The Mowhile team have invested an important amount of time learning from scratch about how to use it efficiently. The team have completed an online course of 25 hours from Udemy in order to gain more experience.

3. **DATABASES**

   B-Quest will handle large amounts of data, most of it probably will not live on the user device at any given time. Instead, the app will likely interact with a database living outside of the user phone. Cloud services like Firebase or Parse provide simple APIs to store data in the cloud and make it available across devices.

   These platforms also often provide Java libraries that can plug into the B-Quest app, making it easy to cache some of the data on the user's device. This syncing of data between local storage and remote database is important if the functionality of the app let users use the app when they are offline.

   Another way to store data locally is through Android's built-in support for using SQL to interact with a SQLite database. Mowhile Geek as a team need to explore and understand how databases work, and the ways to query that data and use it in the project app. As a team we are studying deeply about the features of the different service to find the right one to the needs of B-Quest.

WHY B-QUEST IS A GOOD PROJECT

Mowhile Geek have defined from the beginning the most important points to consider when putting together the project and team. The followings are 7 golden rules that will guarantee the success of B-Quest and the reason why this could be a good project.

Do not Get to loose, Manage Scope Just as Tightly

All changes are managed closely, the scope, potential risks or any modification that have been made in our project. Track, track, and track some more is the key. To focus in one application objective at the time. Each idea must contribute to one of the Application's Specific Objectives, defined in the scope of the application.

Flexible BUT Manage the Schedule with an Iron Fist

Every member of the Mowhile Geek Team manages the schedule tightly. Keeping it on track to make sure everyone knows it and what is expected from them.

Know now, ask the tough questions!

As the rules says, it is tough in many ways. However, it is understood that making these kinds of questions on time, could save a lot of time in an advance phase of the develop process.

A well-defined need

The first condition to fund our project is that it will improve something on a territory and that a segment of the population could benefit from it in the short-long run. Have fun with a few clicks.

Measure, measure, measure

Mowhile Geek as a team use Basecamp as an Application's indicator system to measure if the project objectives have been achieved on time. How much and how well have done the project do?

Knowing the team deeply, Evaluating and applying lessons learned

Mowhile Geek as a team keep monitoring the project results being able to tell if anything has improved or anyone is better off thanks to ours practices, doing constantly informal evaluations of the action plan in a way that allows the team to learn from them and act accordingly during the project's life time.

Mowhile Geek are not reinventing the wheel with B-Quest

The research that have been done found similar apps to our idea. However, none of those matches up entirely and group all the features that B-Quest will have. We are taking advantage of their best functionality and adapting those to our goal and project.

NOVEL ASPECTS

To think that from a fun and unique ‘Quest like’ game people can obtain prizes during their birthdays is the foundation of B-Quest.

Giving presents is a traditional method to honour people on their birthday, and the way of doing it have become very traditional as well. With B-Quest people will have an additional way of making from this tradition something extra fun that along with the physical gift will leave memories.

B-Quest will combine all in one the best features of current apps such as Events, Giveaway, Gaming and even Social media apps to provide a better user experience for Birthday events.

Android Studio (Wikipedia, n.d.) is the official tool that is used for the development of applications for the Google’s Android Operating System. Officially launched to the public on December 2014 it is the IDE (Integrated Development Environment) that replaced Eclipse Android Development Tool (ADT) as the main IDE for native Android applications. Android studio is available for Windows, macOS and Linux.

Why Android Studio?

Our final goal is to build a mobile application; therefore, we need to use a tool that will help us in the process. Since we decided to build a native app for Android, we adopted Android Studio as that tool given that it is (as mentioned before) the official IDE for app development for this OS, at the same time Android Studio is based on JAVA, a programming language with which we have been working in college from semester one and feel more comfortable using.

However, this does not mean that it’s easy, as a team we are investing a big amount of time and effort to learn how to properly use this IDE, part of this self-training is being carried out by doing an online course in the well-known website **Udemy** and, as everything else, with lots of study and practice.

Firebase…maybe?

The Database system to be implemented in the project is not defined yet. Nevertheless, for the time being the team is leaning towards Firebase.

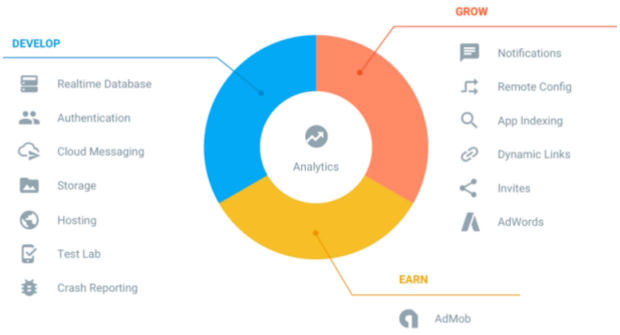
Feature Set

Firebase has a solution for everything that will be needed in order to develop the B-Quest Project. Looking at their site, they have ads, analytics, push notifications, full document storage, and more.

Real Time

The entire database is backed by a real-time connection to the back end, and if the Firebase SDK is used, it will get live updates in the B-Quest app any time something changes, that is one of the most important features that the project will have (when a quest has been completed, all the members of the party should be notified in real time).

The SDK handles all network traffic, which saves lots of time from having to write a networking layer. Since the Firebase database is schema-less, it will be possible to define the data model in B-Quest without the needs to make changes to any server code.



Firebase is unrivalled, as far as our research is concerned, when it comes to how quickly Mowhile Geek could move as a developer team. Its real-time interaction features could save the project from the needs to write huge parts of an app. In return, it may have to make a few sacrifices, it is not way to think about the data in the same way as a more typical API. In the case of building production apps, it is worth working through all the requirements before any team jump in, but for proof-of-concept, Firebase will definitely be the first choice.

Mowhile Geek as a team is investing a big amount of time and effort to learn how to properly use this Google's mobile platform, part of this self-training is being carried out by doing an online course in the well-known website **Udemy** and, as everything else, with lots of study and practice.

SYSTEM DELIVERABLES

To define the system deliverables it is important to consider the general and specific objectives of the project in order to determine what is going to be needed. Then figure out the requirements that will make the deliverable acceptable. That is why before starting the process of development a detailed and careful analysis was carried out in order to define the deliverables.

Once the deliverables were decided the whole project is ready to start the process. Considering budgets and having a good limited scope of what the application is going to do and what not, we can list them:

1. This project is trying to achieve the development of an application that will benefit the users when thinking or trying to find a nice gift for that special someone’s birthday.

2. The main goal of B-Quest as an app is to make the user feel better with the gift they are getting and to do something fun and enjoyable during their special day.

3. As an academic project B-Quest's main goal is to have as a result an Android Mobile Application and Web App.

4. B-Quest will be an application developed in such a way that makes the user experience and user interface as easy and straightforward as possible. This will help people to navigate easily and learn how to use the app, having as a possible result the user's engagement.

   This step is so important because it could benefit the positioning of the application in the online store. In order to do this, tedious and complicated designs are to be avoided.

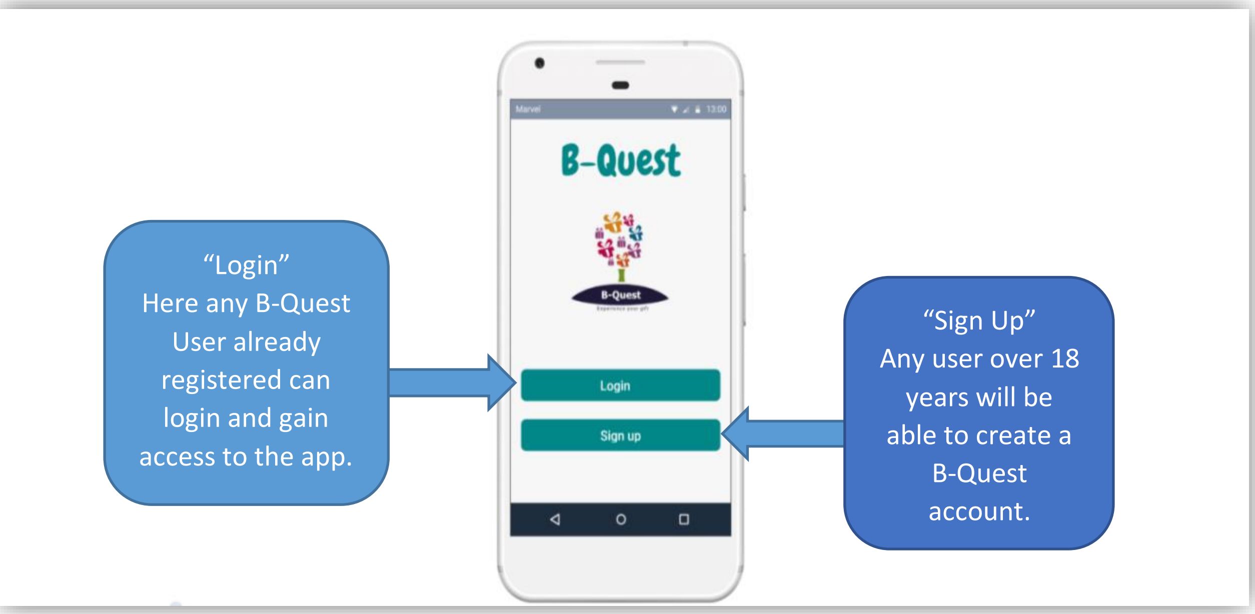
   The cost of acquisition of the last mentioned step is more time but with this a high tech product can be delivered, hoping more people will feel like using it every time it is someone's birthday.

MANUAL FOR PROTOTYPE

A first version of the prototype is done and based on this the manual is the following:

(Please consider since last version of this prototype was created some terms have been updated and will be seen in next version of prototype)

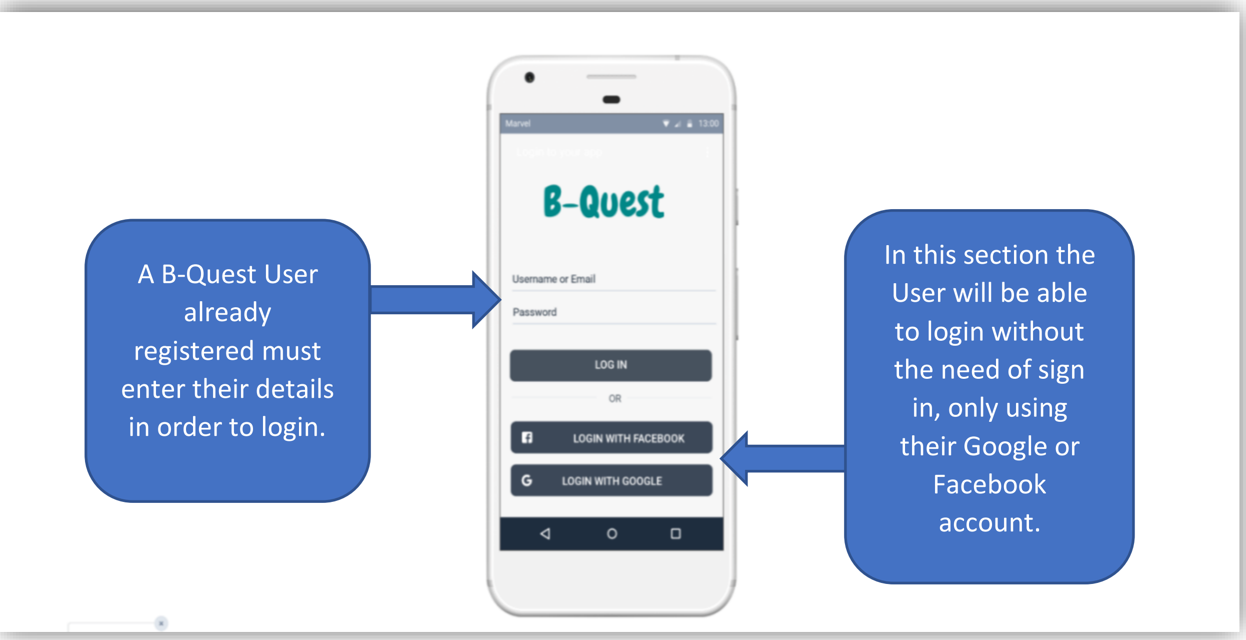
1.- Login or Sign up



"Login"
Here any B-Quest User already registered can login and gain access to the app.

"Sign Up"
Any user over 18 years will be able to create a B-Quest account.

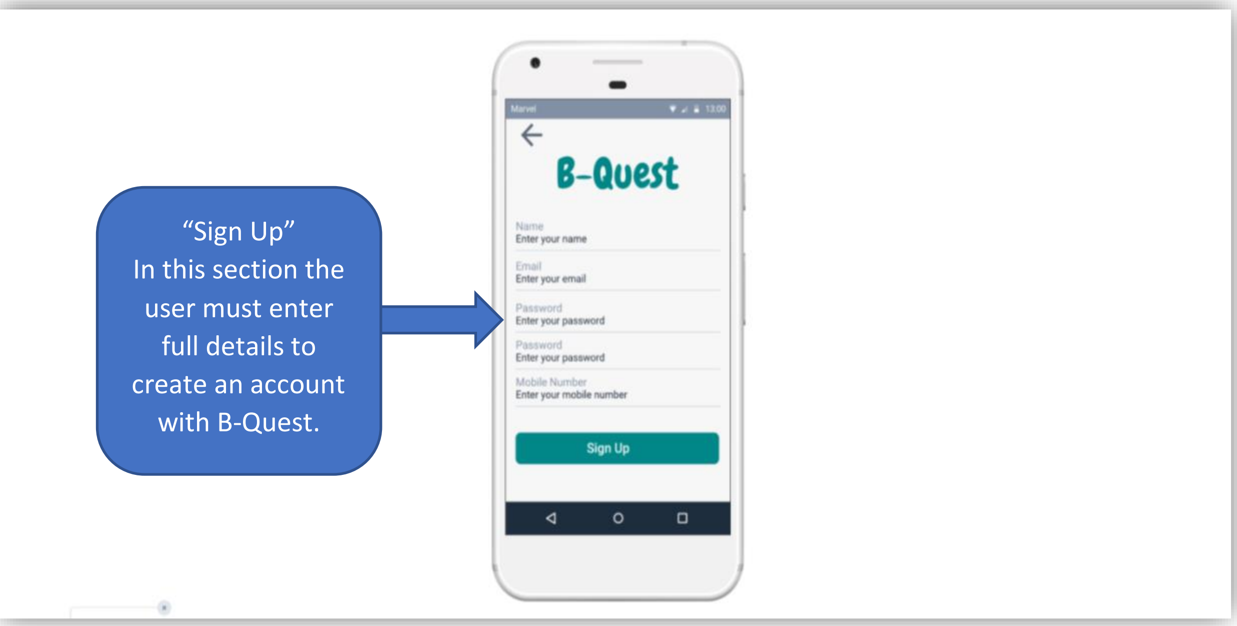
2.- Log in via Social Media or entering details



A B-Quest User already registered must enter their details in order to login.

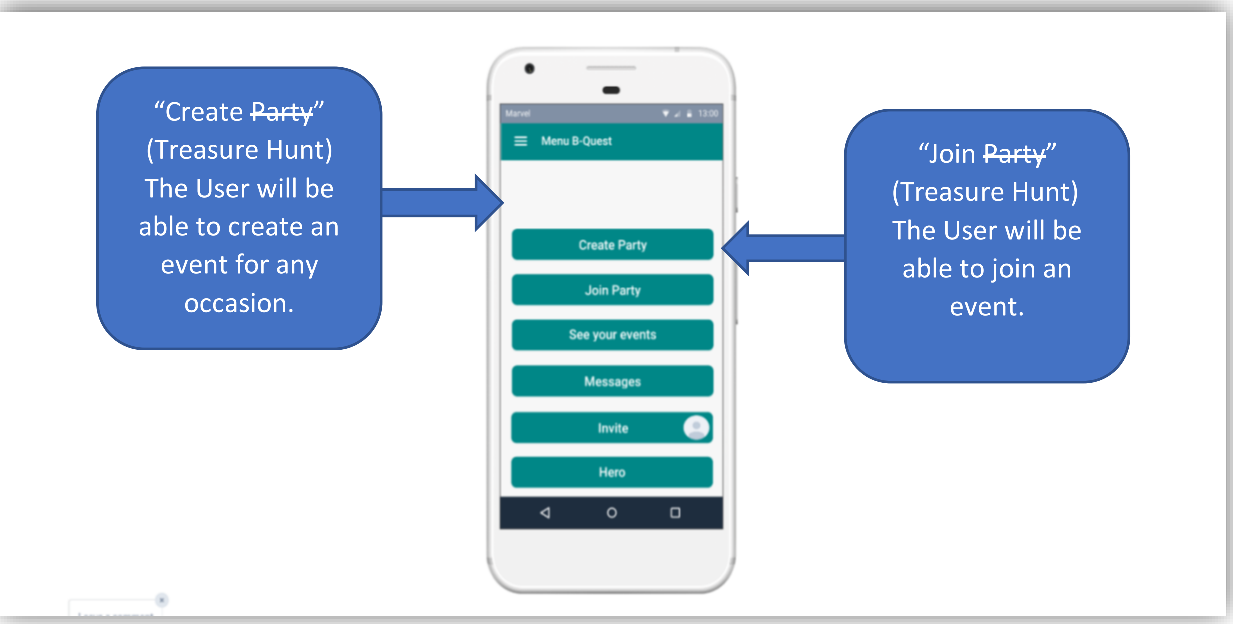
In this section the User will be able to login without the need of sign in, only using their Google or Facebook account.

3.- Create an account



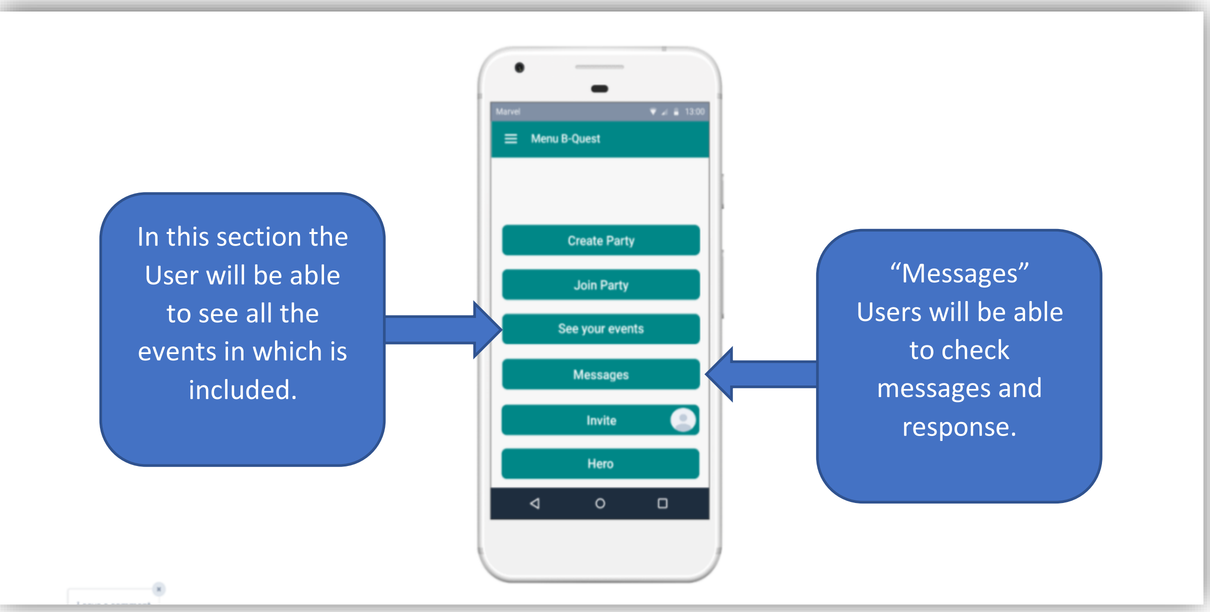
"Sign Up"
In this section the user must enter full details to create an account with B-Quest.

4.- Dashboard, where the user will have all the options about what to do once in the App. (***Party = Treasure Hunt, Hero=B-Hero)***



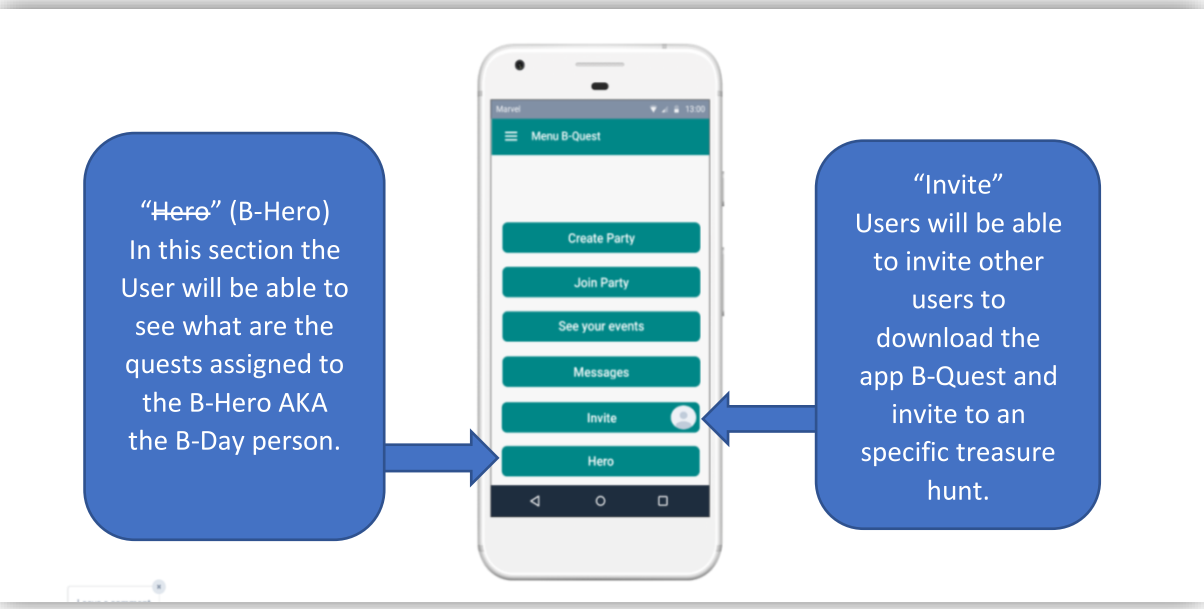
"Create ~~Party~~" (Treasure Hunt) The User will be able to create an event for any occasion.

"Join ~~Party~~" (Treasure Hunt) The User will be able to join an event.



In this section the User will be able to see all the events in which is included.

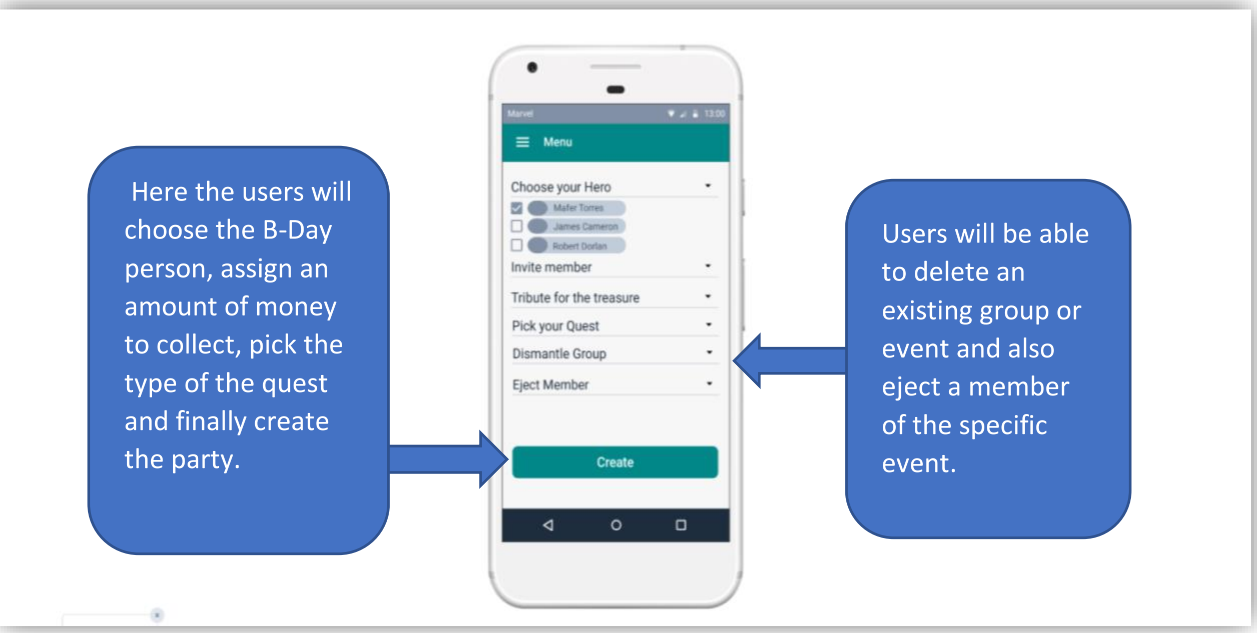
"Messages" Users will be able to check messages and response.



"~~Hero~~" (B-Hero) In this section the User will be able to see what are the quests assigned to the B-Hero AKA the B-Day person.

"Invite" Users will be able to invite other users to download the app B-Quest and invite to an specific treasure hunt.

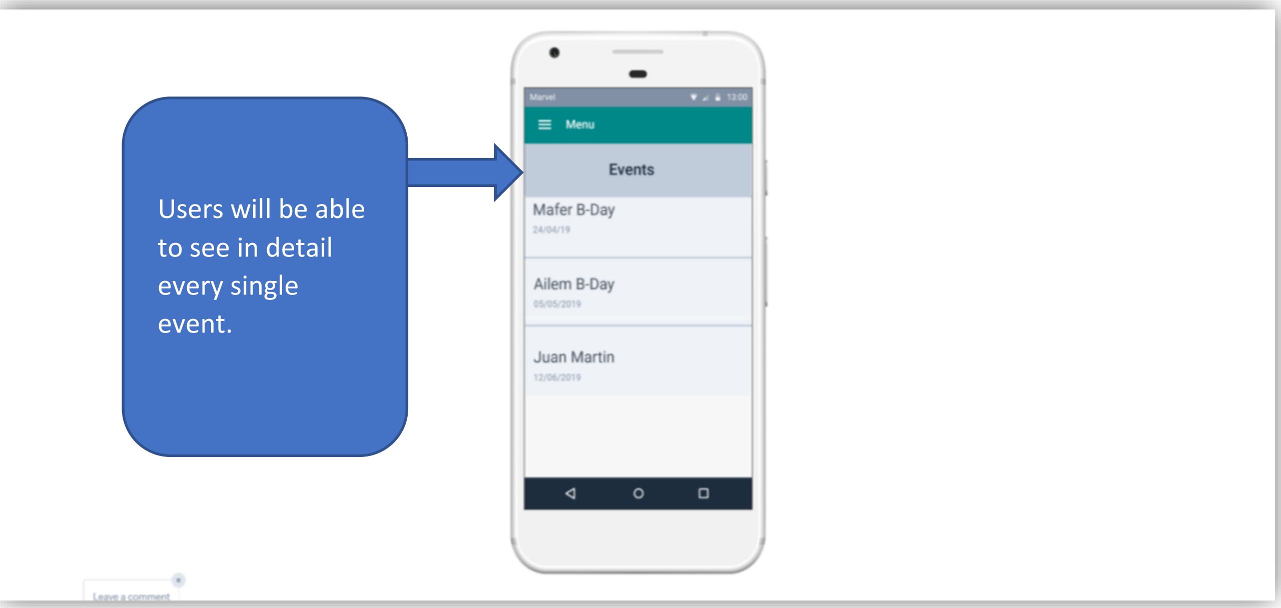
5.- The Create Menu, where the user will be able to set all the requirements for the event.



Here the users will choose the B-Day person, assign an amount of money to collect, pick the type of the quest and finally create the party.

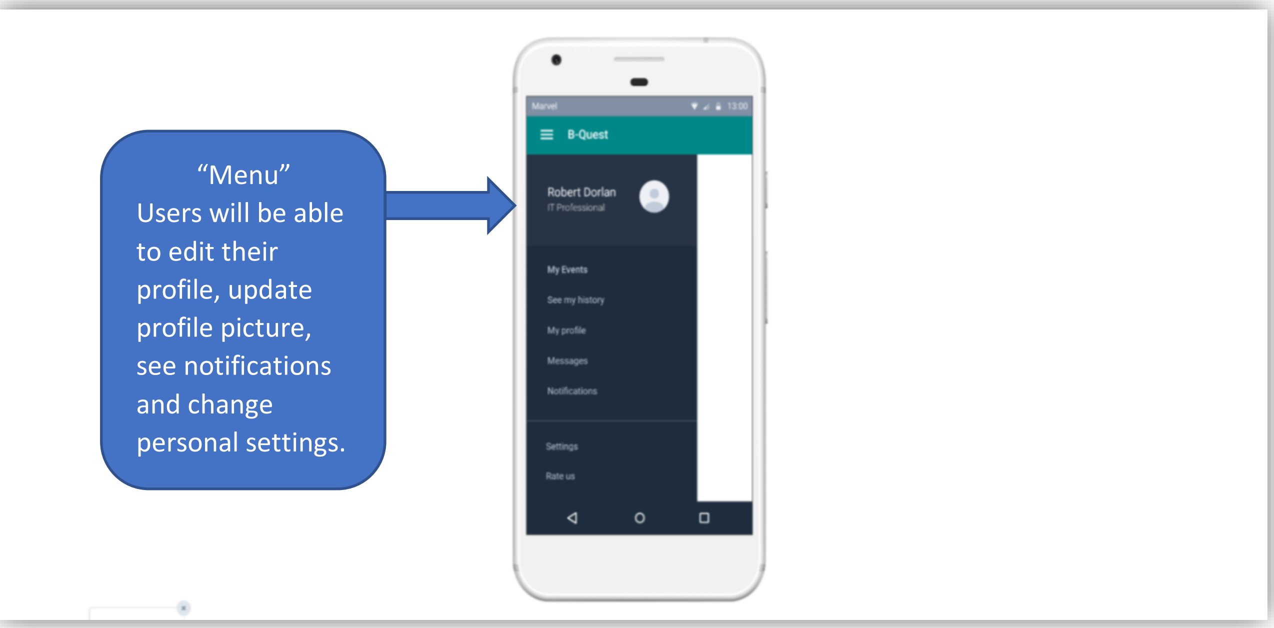
Users will be able to delete an existing group or event and also eject a member of the specific event.

6.- Events dashboard.



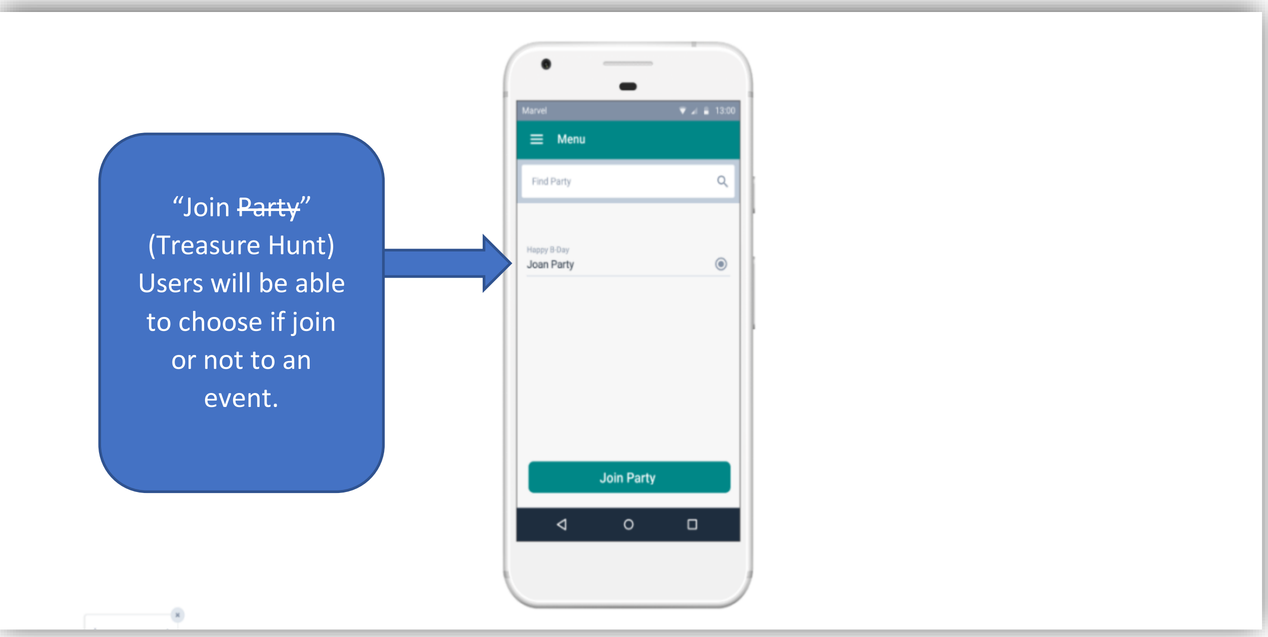
Users will be able to see in detail every single event.

7.- Easy access side Menu.



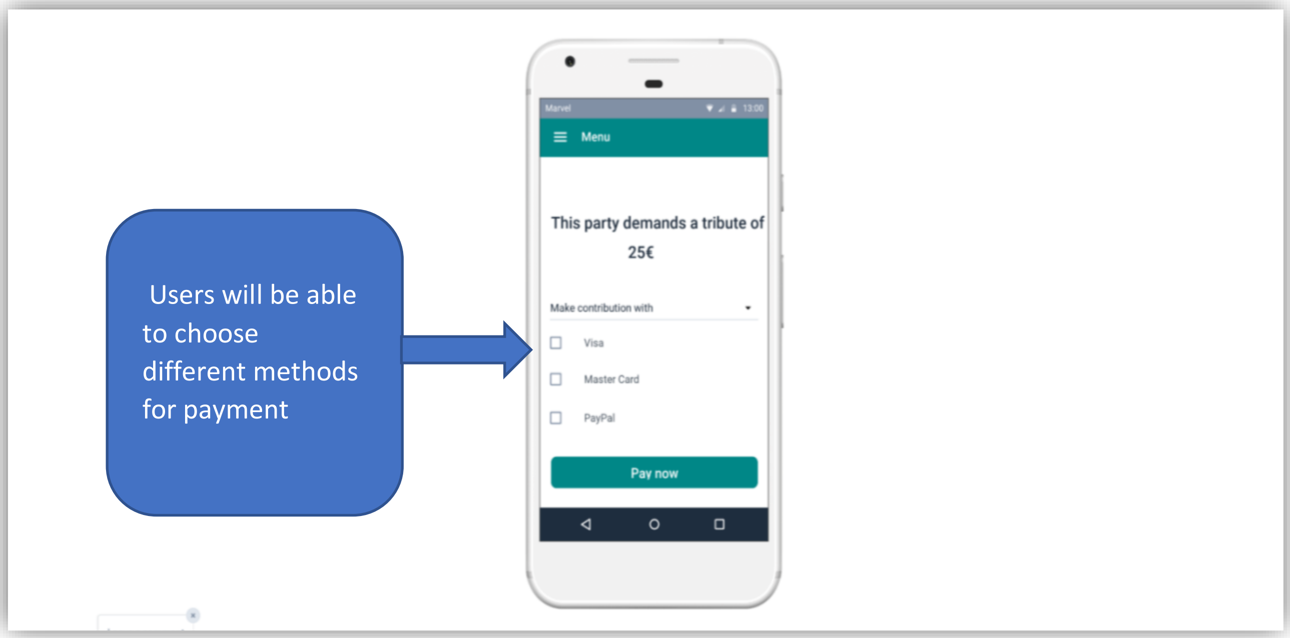
"Menu"
Users will be able to edit their profile, update profile picture, see notifications and change personal settings.

8.- Treasure Hunt Section. ***(Party = Treasure Hunt)***



"Join ~~Party~~"
(Treasure Hunt)
Users will be able to choose if join or not to an event.

9.- Payment section where Lords will be able to contribute in order to be finally included in the event.



Users will be able to choose different methods for payment

This party demands a tribute of 25€

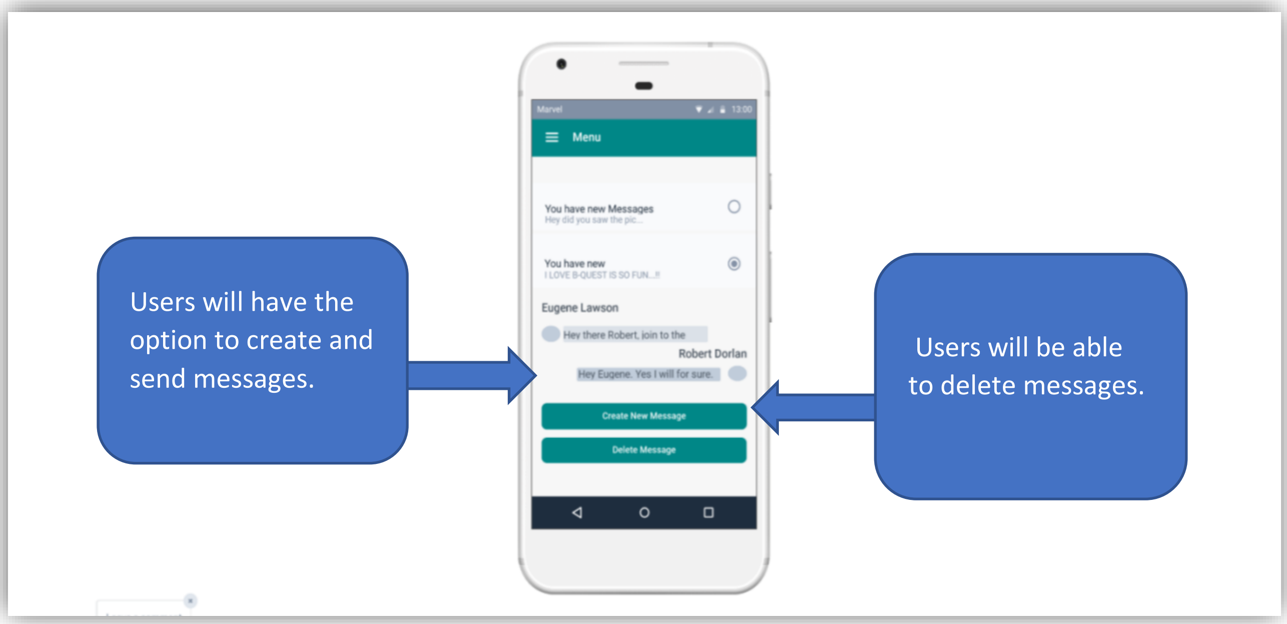
Make contribution with

□ Visa

□ Master Card

□ PayPal

10.- Communication centre, where the users will have the chance to communicate with other members of the contacts.



Users will have the option to create and send messages.

Users will be able to delete messages.

You have new Messages ○
Hey did you saw the pic...

You have new ●
I LOVE B-QUEST IS SO FUN...!!

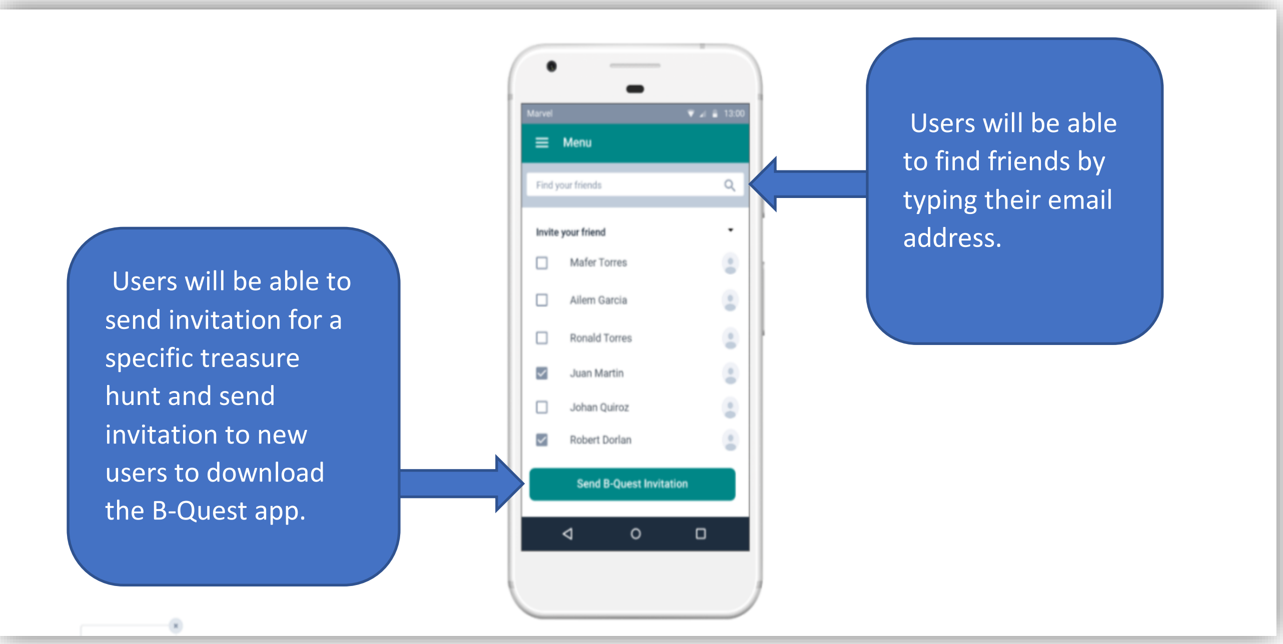
Eugene Lawson

Hey there Robert, join to the

Robert Dorlan

Hey Eugene. Yes I will for sure.

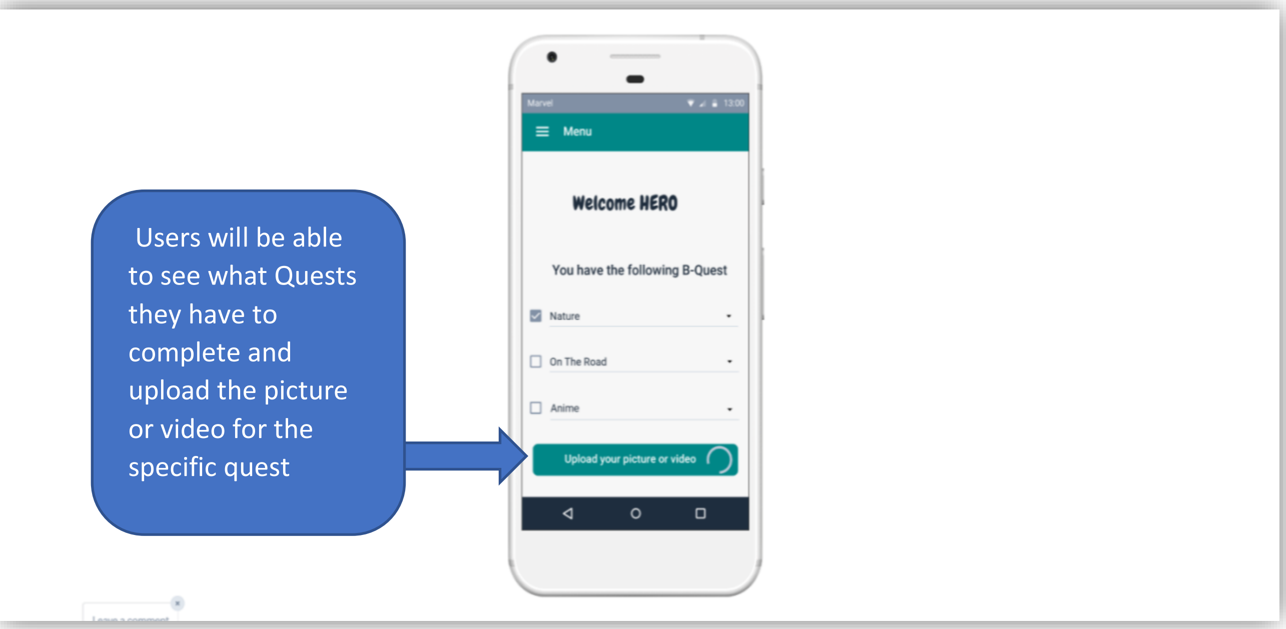
11.- List of Contacts section.



Users will be able to find friends by typing their email address.

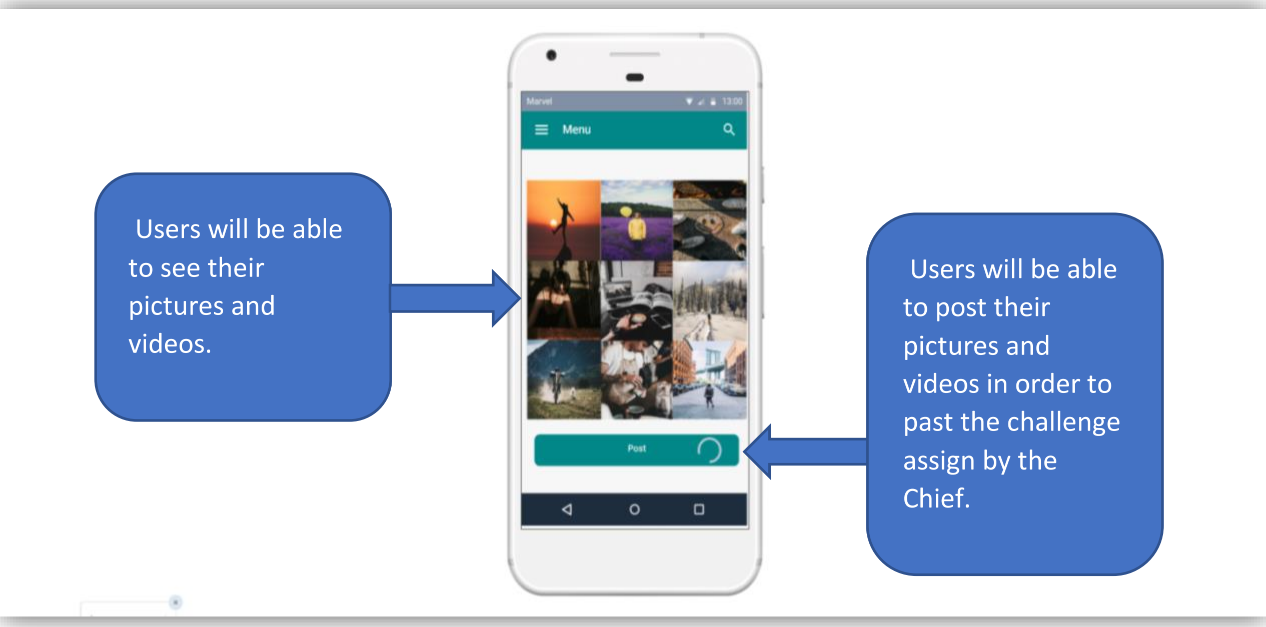
Users will be able to send invitation for a specific treasure hunt and send invitation to new users to download the B-Quest app.

12.- Activity centre for the B-Hero.



Users will be able to see what Quests they have to complete and upload the picture or video for the specific quest

13.- The media centre is where the users will be able to interact with the proof of tasks completed.



Users will be able to see their pictures and videos.

Users will be able to post their pictures and videos in order to past the challenge assign by the Chief.

CHAPTER II
SYSTEM ANALYSIS

FUNCTIONAL REQUIREMENTS

SYSTEM DESCRIPTION

B-Quest is a birthday application where a group of friends can create an event that involves a group of tasks to complete in order to win a prize that will be the birthday present. For our application purposes the prize is always money.

In our app the event is considered a Treasure Hunt, the birthday person is a hero so it's called a B-Hero and the birthday present is a Treasure.

This said, the main purpose of our application is to create a Treasure Hunt with 5 or more Quests for the B-Hero to complete in order to collect a Treasure. For each Quest completed a media proof has to be uploaded by the B-Hero to the system. Once uploaded, the creator of the event (called Chief) will have to approve or reject this proof until the 5 of them are completed and the treasure can be released.

Following is the description of how our system should function in order to achieve our application's main purpose.

Firstly, for the functionality of the system it is required one type of user:

- **Registered user**: which owns an account, is able to log in with an username and a password. The user will have different roles and levels of privileges, depending on how he/she participates in the treasure hunt. Such roles will be:
  - Chief *
  - Lord *
  - B-Hero *

*Note: Please refer to Glossary for terminology used.

GENERAL FUNCTIONAL REQUIREMENTS

*Note: Please refer to Glossary for terminology used.

Functional Requirements:

- When signing in for the first time, the application will guide you through a tutorial showing every step of the application.
- As a first step the user is on the role of "Chief" and it has full privileges. This means, the user will have full access to all the features of the application. It will be allowed to create, delete and view the treasure hunts and track the progress of each of them. Likewise, it will be able to invite Lords to participate in the treasure hunt at any moment and send the request to the B-Hero after at least one Lord accept the invitation. Finally, the "Chief" can review quests and have access to the internal chat between the members of the treasure hunt.
- The user with the role of 'Lord' has limited privileges. This means, limited access to the features of the application. This user is allowed to view in detail and track the progress of every quest completed. This user also has access to the internal chat between the members of the treasure hunt.
- The user with the role of 'B-Hero' has limited privileges. This means, limited access to the features of the application. He/she is able to complete quests and upload proof of quests completed. This user also has access to the internal chat between the members of the treasure hunt. Finally, the B-Hero will be the only one allowed to claim the treasure.
- The application will allow upload media files such as videos and images.
- The system should give the option to upload a profile picture.
- The system will allow all members to abandon the Treasure Hunt at any moment. The application also should allow to the registered user to decline or accept the invitation for the Treasure Hunt.
- Some Terms & Conditions will apply in case the B-Hero abandons and a refund has to be done.
- The application should be able to allow the user to create an account by submitting required personal details and agreeing to the Terms & Conditions. In addition, it should allow the registered user to login introducing the data username and password (submitted in the registration form).
- The system should have an authentication process to verify the user's details against the Database. Likewise, the system should offer the option to log out in case the user wants leave the session, otherwise, session should continue open until user indicate the opposite.
- The system should give the option to register/log in by using a Facebook or a Google account.
- At the time of creating a treasure hunt, the system has to allow the user with the role "Chief" to create the treasure hunt by assigning a name, selecting a minimum of 5 Quests from 5 categories, setting up a minimum amount of 5€ contribution for the

treasure and sending invitations to join the event to the registered and non-registered users.

- The system to invite non-registered users will be via email or sharing a link via WhatsApp.
- Each Treasure Hunt will have a preview once created. Additionally, each Treasure hunt will have 3 status:
    - Pending: When the user has been invited but hasn't accepted or rejected the invitation.
    - In Progress: When the Treasure Hunt has already commenced.
    - Completed: When the Treasure Hunt has been finalised and all Quests completed.
- The system will give the option to view a list of all the Treasure Hunts and it's progress. This view will give also the option to filter the Treasure Hunt according to the role assigned.
- When contributing with the treasure the application will show a PayPal icon that will lead the user to the more convenient PayPal site to register the payment to Mowhile Geek.
- A hidden menu will be available in all the screens of the application. Giving the option to have access to different shortcuts such as: My profile, Settings, Notifications and Log Out and showing the profile picture of the user, the nickname and the role.
- A "Go Back" button will be available in specific screens in case the user need to return to the previous page.

DETAILED FUNCTIONAL REQUIREMENTS

In order to provide a clearer view of the detailed functional requirements, a MoSCoW analysis will be applied.

MoSCoW Analysis

Moscow analysis is one of the easiest methods and most common techniques used by Business Analysts for prioritizing requirement. It will assist in reaching a mutual or common understanding between the developers and the stakeholders of the proposed software application. MoSCoW analyses are usually performed to ensure the most important requirement from stakeholders are ranked or placed in order of development of each condition. MoSCoW analysis can be explained as the following below:

M: Must

Must Requirements could be usually considered as mandatories and non-negotiable for the project to succeed; the software development project is subject to failure if these requirements are not meet to an agreed satisfaction. Prioritization is mainly for high requirement.

S: Should

This requirement should be implemented to the software development if it can be possible at all, they are the next high requirement after the must requirement.

C: Could

These are the desirable requirement that would be desire or charming to have by the stakeholders, not necessarily important but these would be pleasant to have if there is time, capacity, resources or budget available to implement them.

W: Won’t

They are the requirement that has been agreed that won’t be included before the prearranged release. However, they would be implemented at a future stage or updated version of the product.

Functional Requirement ID <CHIEF-001>	
Description: For the low cost, treasure hunt system is to be designed as simple as possible. The menus collect all kinds of treasure hunt quests available for the event. View, Modify, Delete functions are just like Create function.	
Assumptions	The Chief could create, modify or delete a treasure hunt activity
Persona	CHIEF (treasure hunt's owner)
Impact	MUST HAVE
Primary Actor	The chief would be the only user with the capacity of modifying a treasure hunt.
Input	Accessing the "manage treasure hunt" will trigger this function.
Action	**Create a Treasure hunt:** The Chief requests to create a new treasure hunt to be supplied to the lords; One treasure hunt can have many quests, but one quest belongs to only one treasure hunt. **View a treasure hunt or quest:** When a treasure hunt or quest, Chief can request to view it. Usually, chief views the treasure hunt to find whether the quests can satisfy the lords demand. **Modify a treasure hunt or quest:** When Chief views the treasure hunts, he can choose to modify it. But he can't modify the treasure hunt with date of current date because they are in service. **Delete a treasure hunt or quest:** If a treasure hunt or quest cannot be completed any more, the Chief needs to delete the treasure hunt and the related quests other than modifies them. When deleting a treasure hunt, quests belonging to the deleted treasure hunt need to be deleted
Output	It will make changes to the database by creating, deleting, modifying or querying the treasure hunt.
Business Rules	A user needs to have an account and be part of a treasure hunt as a chief.
Exceptions	NA
Dependencies	NA

Functional Requirement ID <CHIEF-002>	
Description: The Chief will decide if a quest submitted for the hero was completed or if it need to be done again.	
Assumptions	The Chief has already invited a B-Hero and created the treasure hunt with the quests to be completed.
Persona	CHIEF
Impact	MUST HAVE
Primary Actor	The Chief would be the only user with the capacity of approving or rejecting the proofs of a quest submitted by the B-Hero
Input	Accessing the submission of the quest for the B-Hero
Action	**Quest was completed:** The Chief will review the submission of the quest for the B-Hero and he will check it as passed. **Quest was not completed:** The Chief will review the submission of the quest for the B-Hero and he will reject it.
Output	**Quest was completed:** It will unlock the next quest on the treasure hunt or if the quest was the last one to be completed (in both scenarios, it will notify to the B-Hero about it). It will release the treasure for the B-Hero. **Quest was not completed:** It will notify the B-Hero that he needs to complete the quest again in order to progress.
Business Rules	A user needs to have an account and be part of a treasure hunt as a chief.
Exceptions	NA
Dependencies	Completing a submitting a quest.

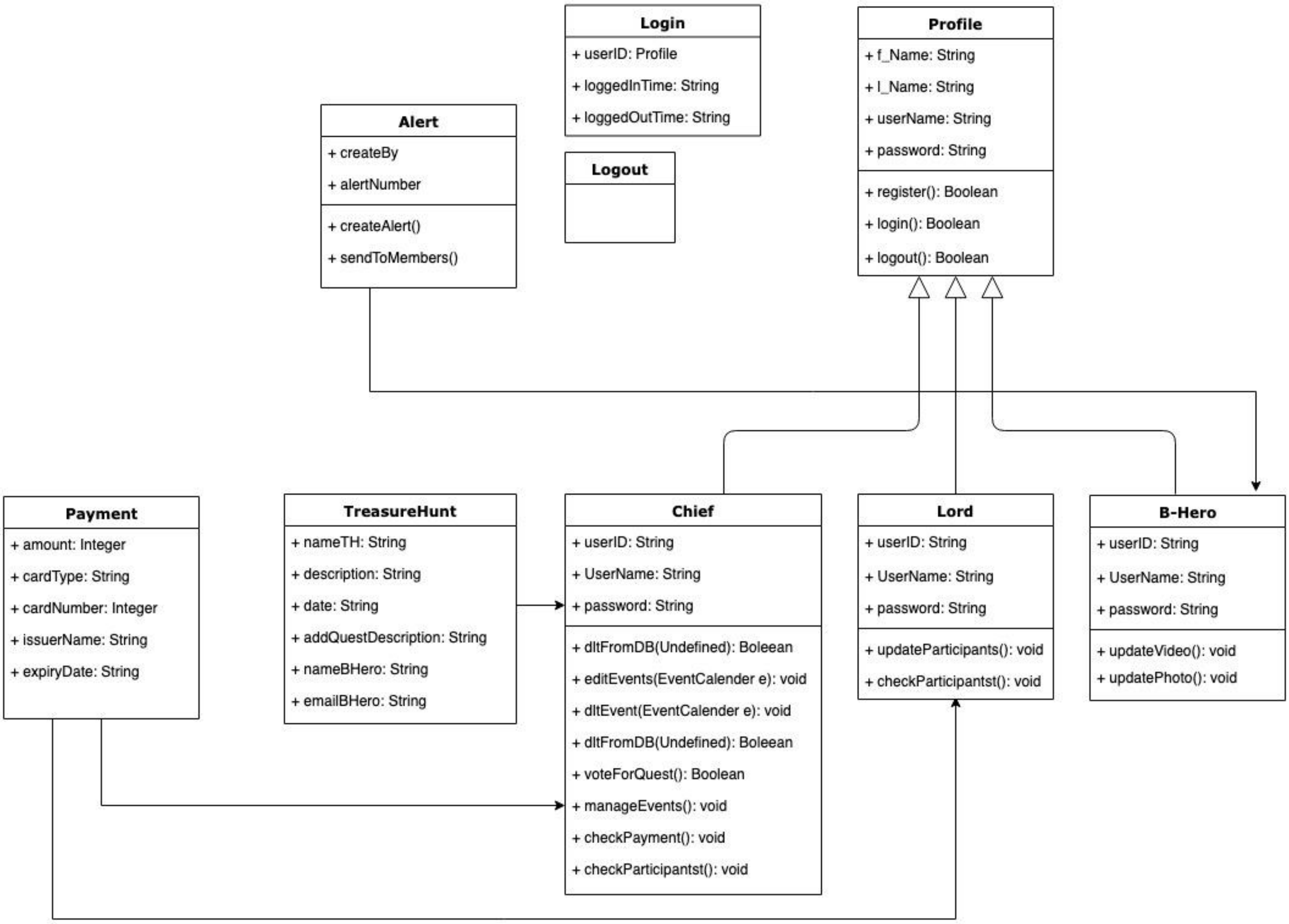
Functional Requirement ID <USER-001>	
Description: New users will need to create an account in order to use and access the app	
Assumptions	New users of the app with not previous account, first time that they access the app
Persona	New users
Impact	MUST HAVE
Primary Actor	A new user of the application that hasn't access the application previously.
Input	New user presses the create account button.
Action	The system should display a window where the new user will insert the information asked in order to have a new account. An email, username, full name and a password will be required in order to complete this action.
Output	A confirmation email will be sent to the new user to confirm its email. A new user will be created in the database.
Business Rules	A new user that will access the app for first time, a user that does not have a previous account.
Exceptions	An email will be assigned for a single account. This one will not be able to use in more than one.
Dependencies	NA

Functional Requirement ID <USER-003->	
Description: The system should give the option to upload a profile picture.	
Assumptions	User already created an account on the system
Persona	All the users
Impact	SHOULD
Primary Actor	All the users that want to modify their account avatar
Input	User will select the option to modify its profile/avatar.
Action	User will select a picture from his/her gallery or use the camera from the phone to take a photo. Also, the system should upload it to the database.
Output	The system should modify the default avatar picture with the new photo selected for the user
Business Rules	A user must have an account and internet connection.
Exceptions	NA
Dependencies	Create an account

Functional Requirement ID <USER-002->	
Description: New users will be able to create an account using their social media accounts (Facebook/Gmail).	
Assumptions	The new user will create an account. The user hasn’t accessed he app previously.
Persona	New user
Impact	MUST HAVE
Primary Actor	The new user will be able to use their social media account to join the app.
Input	The new user will select the option for sign up using his/her social medial account. Two buttons will appear, one associated with Facebook and another one with Gmail. The user must select one of those buttons in order to advance in the process.
Action	The system should open another window from the social media provider that the user had selected. It will ask for confirmation about the process.
Output	The system should open the dashboard of the user with the account already created.
Business Rules	A user must have a social media account in Facebook or a Gmail account.
Exceptions	If the user has already associated a Facebook or Gmail account to a previous, he will not be able to associate it with a new account on the app.
Dependencies	Sign up

Functional Requirement ID <HERO-001->	
Description: The application will allow upload videos and images from the Hero for each quest that was assigned to him in the treasure hunt.	
Assumptions	The user must have an account. Also, the B-Hero must be already part of a treasure hunt.
Persona	B-Hero
Impact	MUST HAVE
Primary Actor	The B-Hero will be able to upload videos or pictures as parts of the activities of the treasure hunt that he is a participant.
Input	B-Hero will complete a quest, he/she will access to the treasure hunt and access to the quest that has completed, will press the upload bottom to submit the proofs.
Action	The system should upload the video or picture to the database. Also, it will send a notification to the members of the treasure hunt. The system should allow to the Hero to make any modification to the quest before it has been reviewed for the Chief.
Output	The system should display a message confirming that the video or picture was uploaded satisfactorily.
Business Rules	A user must have an account, the user must have been nominated as a Hero for the treasure hunt. User must have accepted it.
Exceptions	If the video or picture was reviewed and approved for the Chief. Hero will not be able to modify that specific quest.
Dependencies	NA

CLASS DIAGRAMS



DATA REQUIREMENTS

Firebase is a Google platform that provides different services, such as: storage, hosting and 2 databases, real time Database and Firestore. It also provides tools for authentication and security. We have chosen to work with firebase for different reasons, it has a fairly good free bracket, which means not a significant amount (or none) money have to be spend ahead of the launch and success of the app, also the documentation provided is very well made, easy to read and to understand, also being owned by Google, Firebase services synergise perfectly with Android and Android Studio (IDE we are using for the development of our application).

Firestore is a NoSQL database provided by Firebase. NoSQL databases do not use the table/columns/rows schema that we know from SQL databases. Firestore is a document database, this means that stores data in Documents that live inside Collections, also opposite to SQL databases, redundancy of data is allowed and the three rules for normalization **do not apply**.

A document is a record that contains a series of key-value fields in a Json like format, data can be stored using different datatypes, some of them are Boolean, number, string, timestamp. Documents also support the usage of arrays and maps to store and structure data. It also important to know that documents within Firestore are limited to 1 MB in size.

We can think of a COLLECION as a big container to organize various documents (up to 20.000), and that is all really, a collection cannot be loaded with data if a document does not exist in it, also a collection cannot contain another collection, however, a document can point to another collection known as a sub-collection. A Sub-collection is a collection associated with a specific document and can be queried only through that document.

Firestore provides three basic ways of storing that data:

- Root collections
- Nested data in documents
- Sub-collections within documents

Given that our knowledge come from a SQL world where everything is organized in tables, we have decided to adopt the sub-collections approach since provides a better and easier way of visualizing the structure of the data and at the same time allows us to organize the data in a hierarchical manner.

Queries in Firestore are built specifying the path to the document you want to reach, this means that the query will be composed of a series of calls to collections, documents and sub-collection associated with documents.

OVERVIEW OF ENTITIES

Database description (Stage 1):

NOTE: bear in mind that we can use Json files, maps or arrays to organise data inside a document, chances are that a combination of those formats will be used when the database is finalized.
NOTE: Authentication information (i.e.: username and password) will be kept separated from the rest of the data.

NOTE: All id's will be unique and autogenerated by the database.

User is a collection that stores individual documents for each registered user, each user will have the following attributes:

```
{
   user_id: ObjectID

   name: string

   email: string

   phone_number: string

   friends {

      user_id(friend): ObjectID (will get name here)

      user_id(friend): ObjectID (will get name here)

      user_id(friend): ObjectID (will get name here)

      user_id(friend): ObjectID (will get name here)

   }

}
```

Attribute	**Data type**	**Description**
user_id	ObjectID (auto generated)	required / unique
name	String	required
Email	String	required / validation applies
phone number	String	required / validation applies
friends	map / array	not required

The ***friends*** attribute inside the user ***document*** will be set (array/map) of users that have been added as friends. At the same time each individual user ***document*** will point to a sub-collection that will contain information for each friend in the set, this means that the user can have a list of friends and then query an individual friend if needed

User friend list is a sub-collection of a specific user, it will contain information about users listed as friends by other users. Will contain similar attributes to those on the user document, such as user name and contact details: (this might be redundant)

```
{
   user_id: ObjectID
   name: string
   email: string
   phone_number: string
}
```

Attribute	Data type	Description
user_id	ObjectID	required / unique
name	String	required
email	String	required / validation applies
phone number	String	required

Quest category is a collection that contains documents for each different category, inside each document there will be a category id along with a title or category name and a brief description of the category itself:

```
{
   category_id: ObjectID
   category_name: string
   description: string
}
```

Attribute	Data type	Description
category_id	ObjectID (auto generated)	required / unique
category_name	string	required
description	string	required

Each quest category document will point to a sub-collection ***quest*** that will contain individual documents for each quest related to that category. We could just add all quest inside the category document rather than creating a sub-collection. This would be fine when or while we have a couple of quest for each category. However, is not a good practice if we think about scalability. At some point the app will grow and more quest will be added, all this data will then be thrown to the client with no filters, we read full documents, there is no way of reading only a fraction of the document. Keeping each quest as a document in a sub-collection allow us to control the amount of data that will be given to the client using pagination.

Quest is a sub-collection that contains documents for each quest that will be provided to the user:

```
{
   quest_id: ObjectID
   title: string
   description: string
   difficulty: number
}
```

Attribute	Data type	Description
quest_id	ObjectID (auto generated)	required / unique
title	string	required
description	string	required
difficulty	number	required

Event is a collection that contains individual documents for each event, the event document will contain an event id, a title and possibly a list of members and quests. This could be added as attribute or on a sub-category, here description based on a sub-collection will be added:

```
{
   event_id: ObjectID
   title: string
}
```

Attribute	Data type	Description
event_id	objectID (auto generated)	required / unique
title	string	Required

Each event document will point to two sub-collections one storing private data that will remain hidden from the client and other storing public data accessible by the client, in both cases the data will be stored as key/value set within a single document inside private or public sub-collection:

Private data

```
role {
   user_id: "chief"
   user_id: "lord"
   user_id: "lord"
}
payment {
   user_id: "yes"
   user_id: "yes"
   user_id: "no"

}
```

Attribute	Data type	Description
role	map / array	required
payment	map/array of Booleans	required

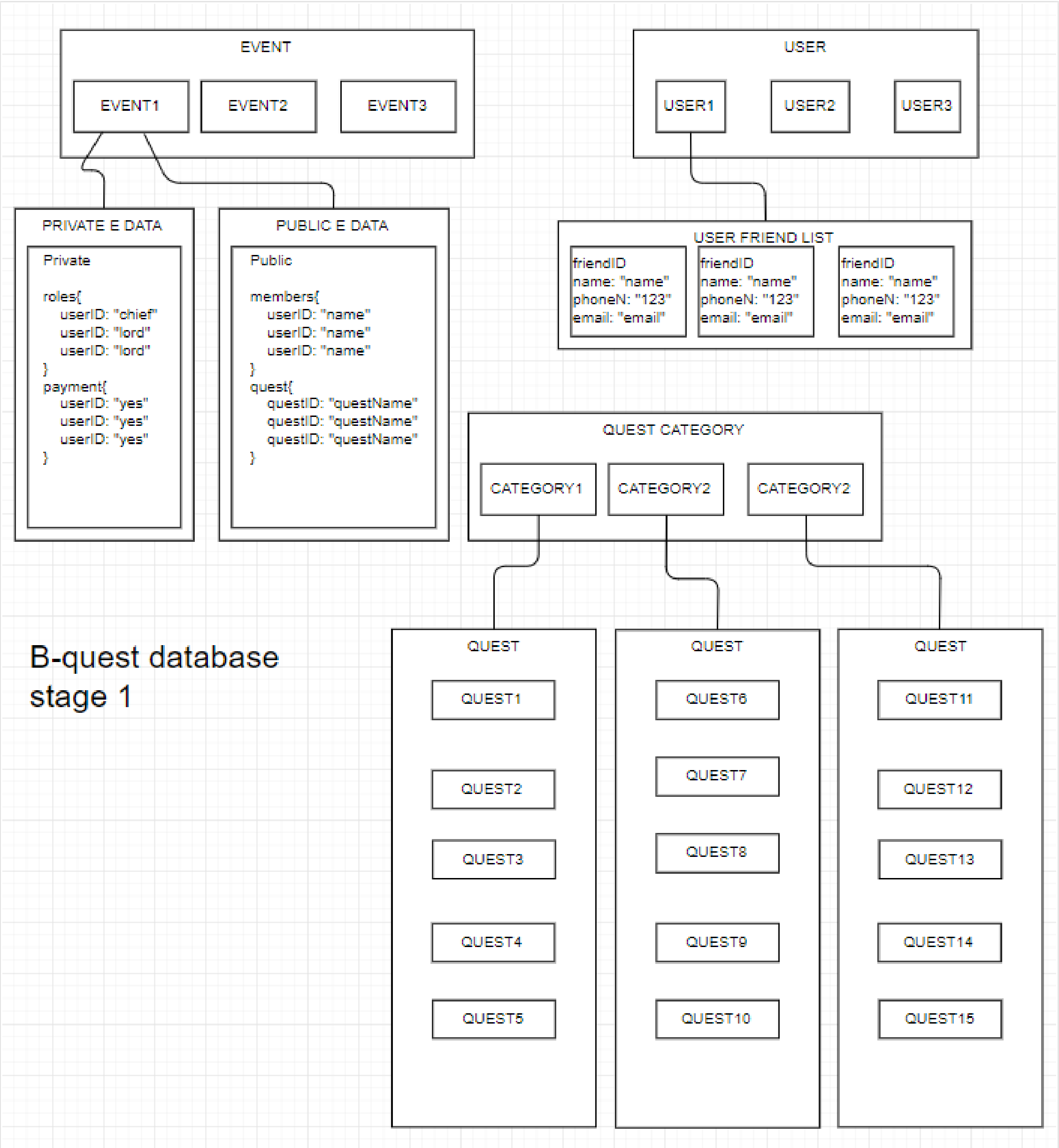
This will allow to set rules and permissions based on the “role”, I.e. you can only delete an event before competition if your role is “chief” or participate (have access/follow) an event if your payment status is positive.

Public data

```
members {

   user_id: "name"

   user_id: "name"

   user_id: "name"

}

quest_list {

   quest_id: title

   quest_id: title

   quest_id: title

   quest_id: title

}
```

Attribute	Data type	Description
members	map / array	required
quest_list	map / array	required

ENTITY-RELATIONSHIP DIAGRAM



```
Private

roles{
    userID: "chief"
    userID: "lord"
    userID: "lord"
}
payment{
    userID: "yes"
    userID: "yes"
    userID: "yes"
}
```

```
Public

members{
    userID: "name"
    userID: "name"
    userID: "name"
}
quest{
    questID: "questName"
    questID: "questName"
    questID: "questName"
}
```

```
friendID
name: "name"
phoneN: "123"
email: "email"
```

```
friendID
name: "name"
phoneN: "123"
email: "email"
```

```
friendID
name: "name"
phoneN: "123"
email: "email"
```

CHAPTER III
SYSTEM ANALYSIS

USER INTERFACE DESIGN

Web application UI Design

The ability to design user interface so that it attracts the user attention and enhances the user experience is one of the biggest challenges during the development process. To design the Web application a decision was made to use Semantic UI. Semantic UI is an User Interface component framework for theming websites. Semantic UI enables developers to build websites with fast and concise HTML, along with a complete mobile responsive experience. Semantic UI treats words and classes as exchangeable concepts. Classes use syntax from natural languages like noun/modifier relationships, word order, and plurality to link concepts intuitively. It is important to remember that everything stems from knowing the users; we need to understand their goals skills, preferences and tendencies. As we have all this aspect clear we consider the following when designing the interface:

Keep the interface simple, Login page.

The best interfaces are almost invisible to the user (Usability.gov, n.d.). The login page is one sample of it. It contains only the necessary information needed for the user to log into the account. During this process it is important to avoid unnecessary elements and keep clear the language and labels used in messaging. Also, the login page is built like a modal. Modals put an overlay on the screen, and therefore take a higher visual precedence over all other elements. If the user introduces an invalid input, validation warnings is displayed highlighting in red colour, with purpose of making the user aware of any possible mistake made on the input, minimizing the chances of the system throwing errors.

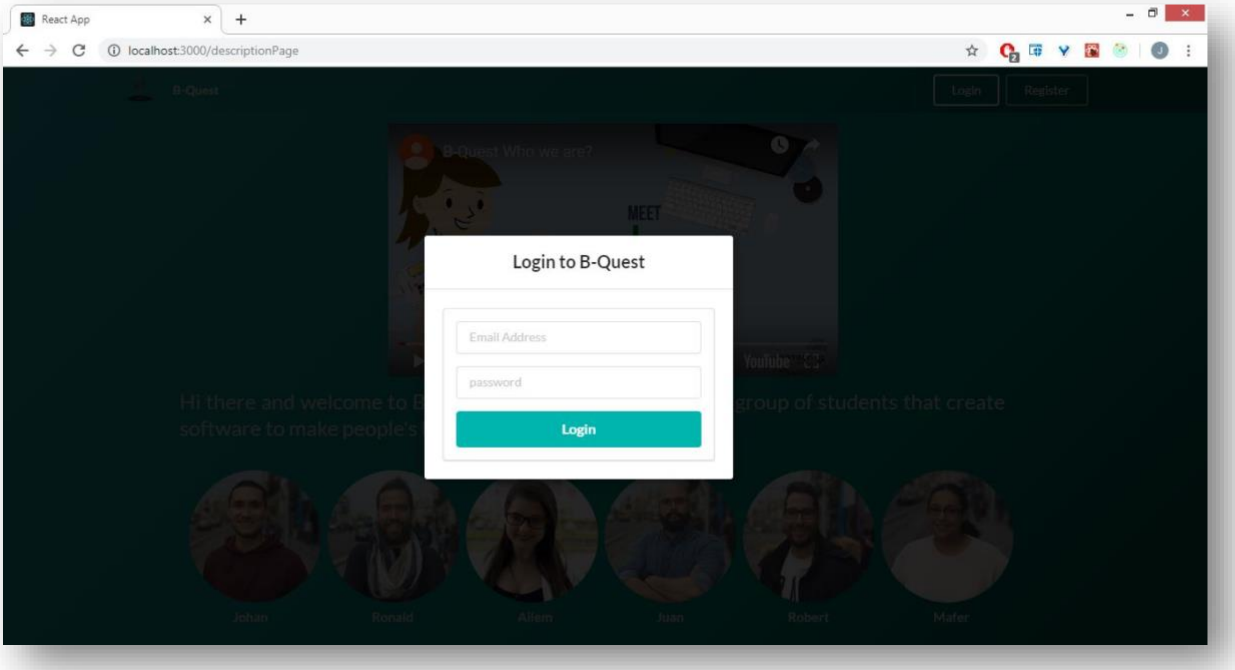
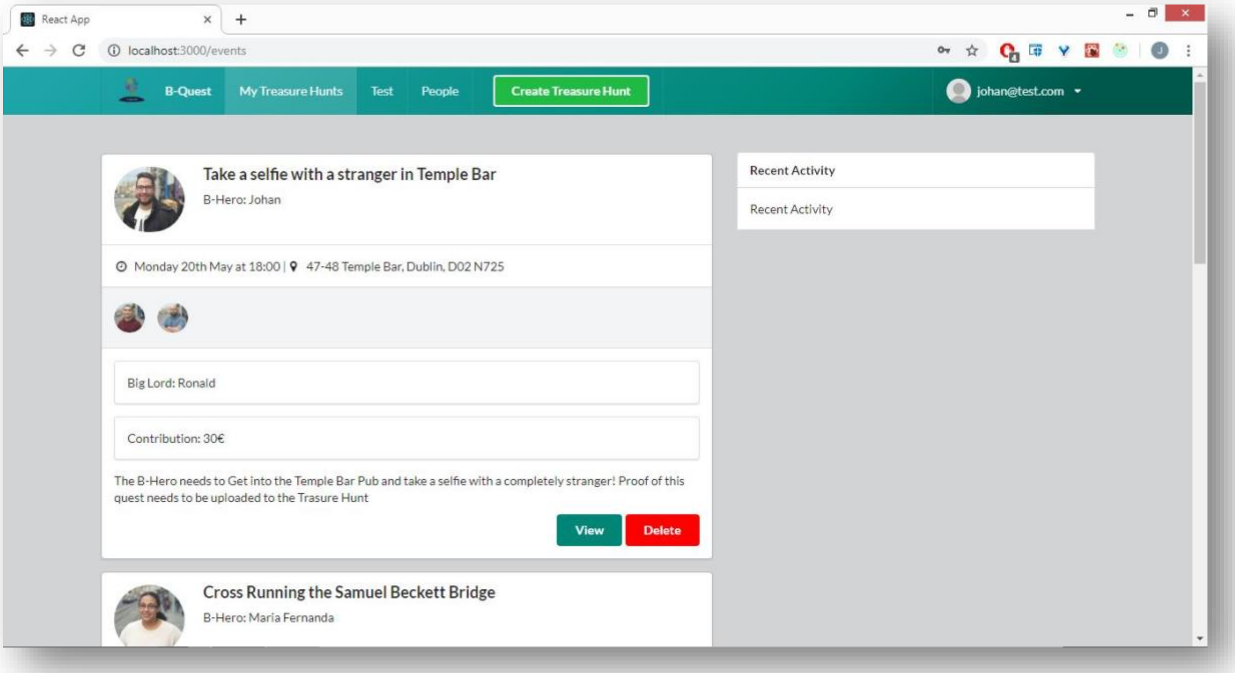


Figure 1.- Web Application - Login Page

Create consistency and use common UI elements. Dashboard.

By using common elements in the UI, users can feel more comfortable and they are able to get things done more quickly. It is also important to create patterns in language, layout and design throughout the site to help facilitate efficiency (Usability.gov, n.d.). For B-Quest an Intuitive dashboard is defined. After login the dashboard page it is displayed. The purpose of this page is to present the user an overview of all the Treasure Hunts they may have joined or have been invited. On the left side, a list of Treasure Hunts is displayed with description of each event and a little icon with the profile picture of the people who has already enjoyed the Treasure Hunt. Each display of the Treasure Hunt contains title, users joined, a small description of the Treasure Hunt, B-Hero's name and birthday and a View/play button that will lead to another page. On the left side a window of notifications is displayed.

On top of the page there's a navbar. This element contains a button of Create Treasure hunt and a display of the username with a dropdown menu, which the user can access by clicking on the small "arrow" icon. This menu is populated with options to changes profile, settings and sign out. The user has the option to see all the Treasure Hunts (Lord), create a new Treasure Hunt (Chief), or take part in his/her own adventure (B-Hero); depending on their role.



Take a selfie with a stranger in Temple Bar

B-Hero: Johan

Monday 20th May at 18:00 | 47-48 Temple Bar, Dublin, D02 N725

Big Lord: Ronald

Contribution: 30€

The B-Hero needs to Get into the Temple Bar Pub and take a selfie with a completely stranger! Proof of this quest needs to be uploaded to the Trasure Hunt

Cross Running the Samuel Beckett Bridge

B-Hero: Maria Fernanda

Recent Activity

Recent Activity

Figure 2.- Web Application - Dashboard

Strategically use colour and texture. Main page

Attention can be directed or redirect away from items using colour, light, contrast, and texture for better advantage (Usability.gov, n.d.). When selecting a colour scheme for B-Quest the Adobe's colour wheel tool was used to help deciding which the best option were. The selection of specific colours, not random ones, were considered to implement on the website. In order to do that, a few guidelines were followed to generate the most attractive results.

1. Choose the primary colour for your website.

It is important to have clear where the primary colour should be used. This is an essential step throughout the website, using it for prominent elements such as your header, background, or title text will help us to get the attention of the right consumers.

2. Select a colour scheme using a colour wheel tool

Confidences, technology, reliable and fun are mainly the adjective that describe B-Quest should show to the user. Having this in mind, an investigation about which colour could be associated with most of these adjectives was made.

Blue is the most universally favoured colour when talking about technology. Blue is unique and versatile, plus each shade of blue can mean a different thing. Blue is known to be linked with creativity and when we look at a few words that is associated with blue, we come up with words like trust, dependability, serenity, intelligence, confidence, aloofness, and unappetizing.

In design, the exact shade of blue you select will have a huge impact on how the designs are perceived. Light blues are often relaxed and calming. Bright blues can be energizing and refreshing. Dark blues, like navy, are excellent for corporate sites or designs where strength and reliability are important. Knowing this the selection of blue shades that ended in green was made.

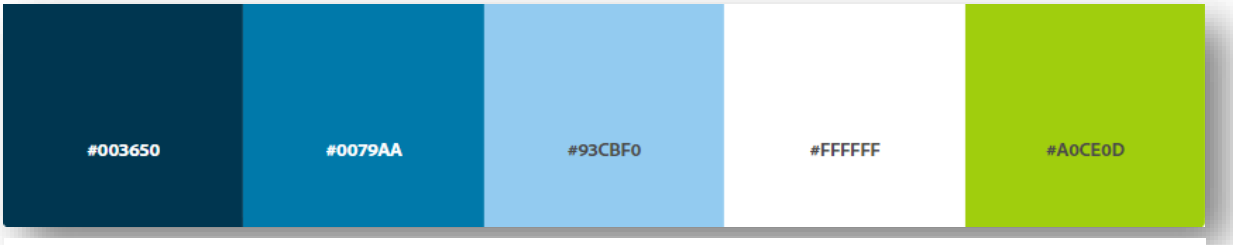
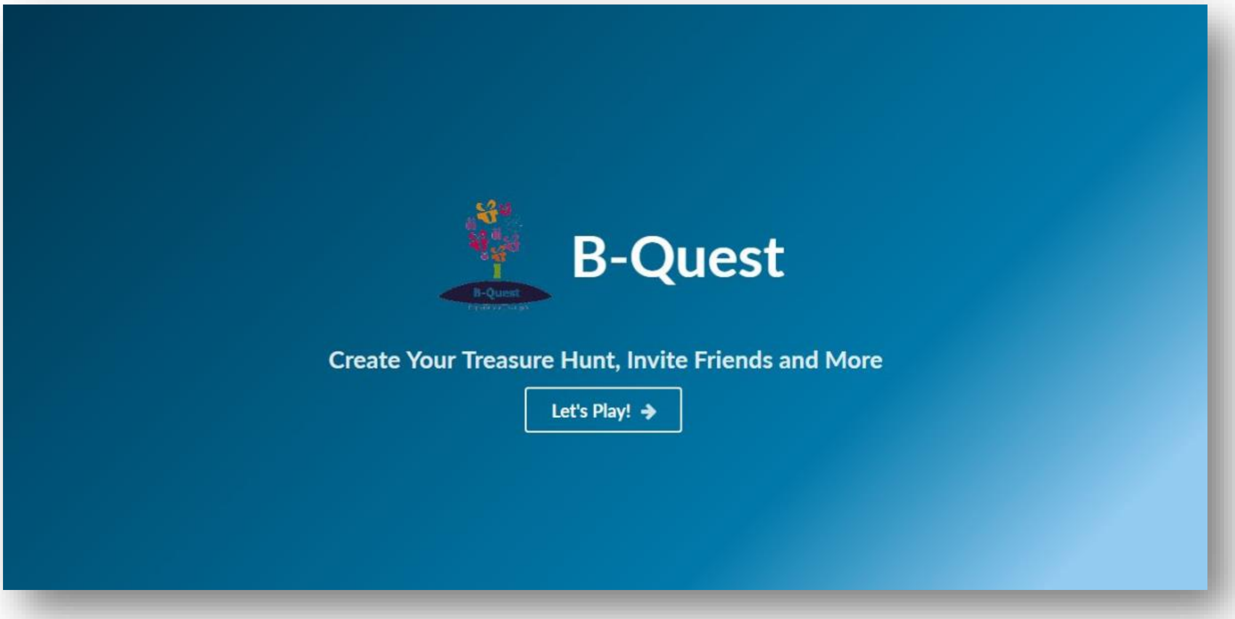


Figure 3.- Colour scheme used for B-Quest with Adobe's colour wheel tool



B-Quest

Create Your Treasure Hunt, Invite Friends and More

Figure 4.- Web Application - Main Page

Make sure that the system communicates what is happening. Are you chief, lord or a B-Hero?

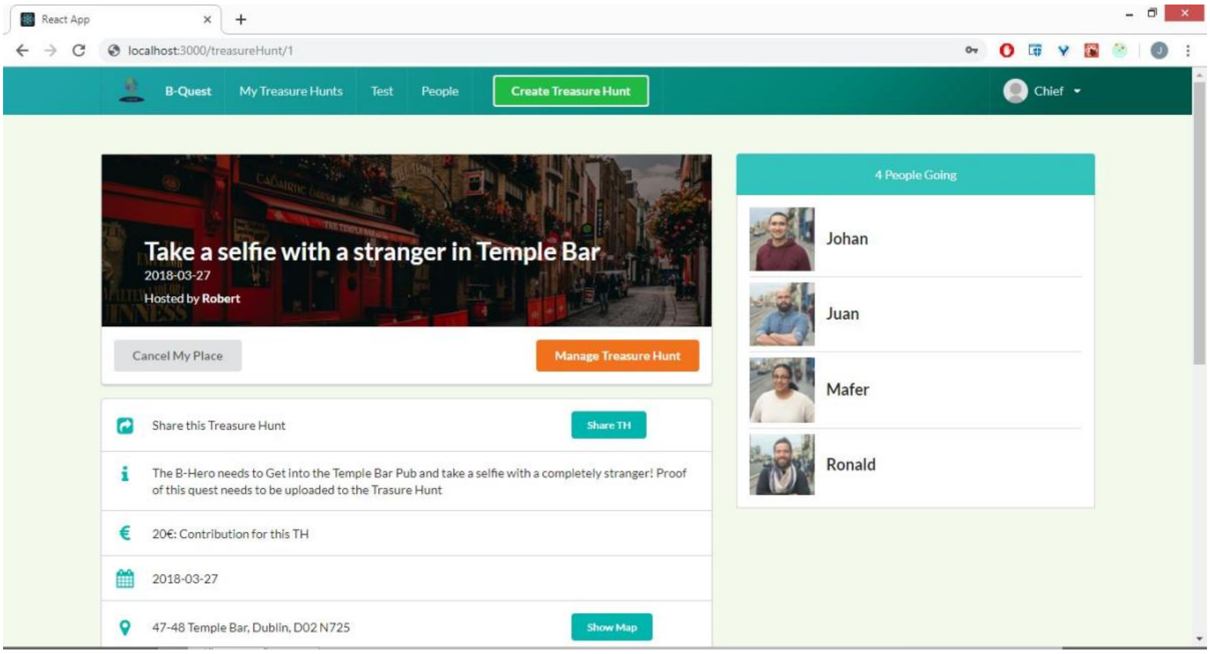
Keeping in mind that users must be informed of actions, changes in state, or errors, the use of various UI elements to communicate status and, if necessary, next steps can reduce frustration for the user. There are 4 types of users when using the web application. To make aware the user of his/her role the use of different colours was made to distinguish which functions the users is allowed to execute.

Guest: when the user visits the website for the first time, the main page is displayed and options like login and register. After the user is register, he/she will see the display of the dashboard with the list of activities that he/she has done. Until here the user doesn't have any label. He is still a guest in the website.

Chief: The role called "chief" will start when the user presses the button CREATE TREASURE HUNT on the dashboard. Now the user is a chief and he/she is allowed to create and modify a Treasure Hunt. The page, create Treasure Hunt, has different text input where the user is able to introduce the details of the event: title, description of the Treasure Hunt, B-Hero's name, birthday and email. Scroll down menus are available for the chief to choose the Quest, the location of the Treasure Hunt and the contribution for the event. The treasure hunt is created when the user clicks the create button, and automatically the event is added to the list of treasure hunt and display on the dashboard.

B-Hero: The main character. After the user accepts de invitation to the treasure hunt, this event is displayed in his/her dashboard. But one important detail is that when the user presses the button view event, the next page displayed will have different option comparing with the chief or lord role. The B-hero page has a description of the quest to achieve and an upload bottom that allow to the user uploading picture or video depending of the quest. Another very important visual difference of the page is that colours of the buttons are different comparing to the chief and lord pages.

Lord: Once the treasure hunt starts the Lord is allow seeing all the different photos and videos that the B-Hero has loaded. Lord's page has a different colour on the buttons comparing with the chief and B-Hero.



Take a selfie with a stranger in Temple Bar

2018-03-27

Hosted by **Robert**

Share this Treasure Hunt

The B-Hero needs to Get into the Temple Bar Pub and take a selfie with a completely stranger! Proof of this quest needs to be uploaded to the Trasure Hunt

20€: Contribution for this TH

2018-03-27

47-48 Temple Bar, Dublin, D02 N725

Figure 5.- Web Application – Dashboard with Chief privileges.

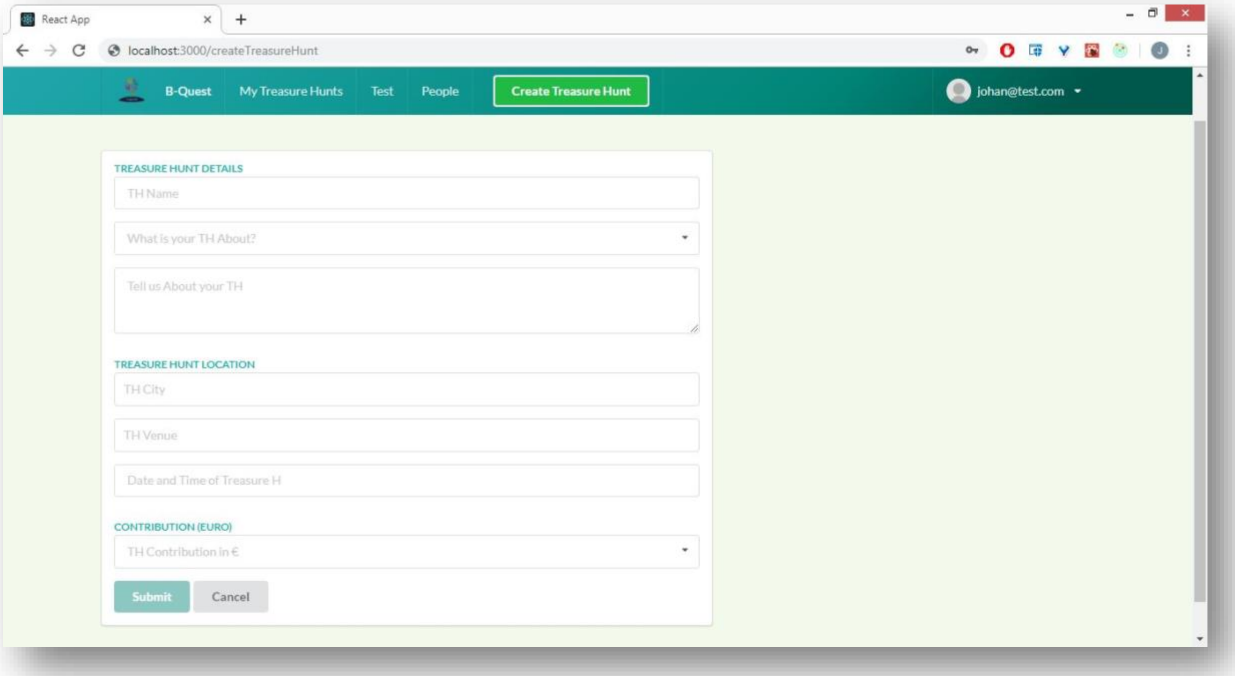
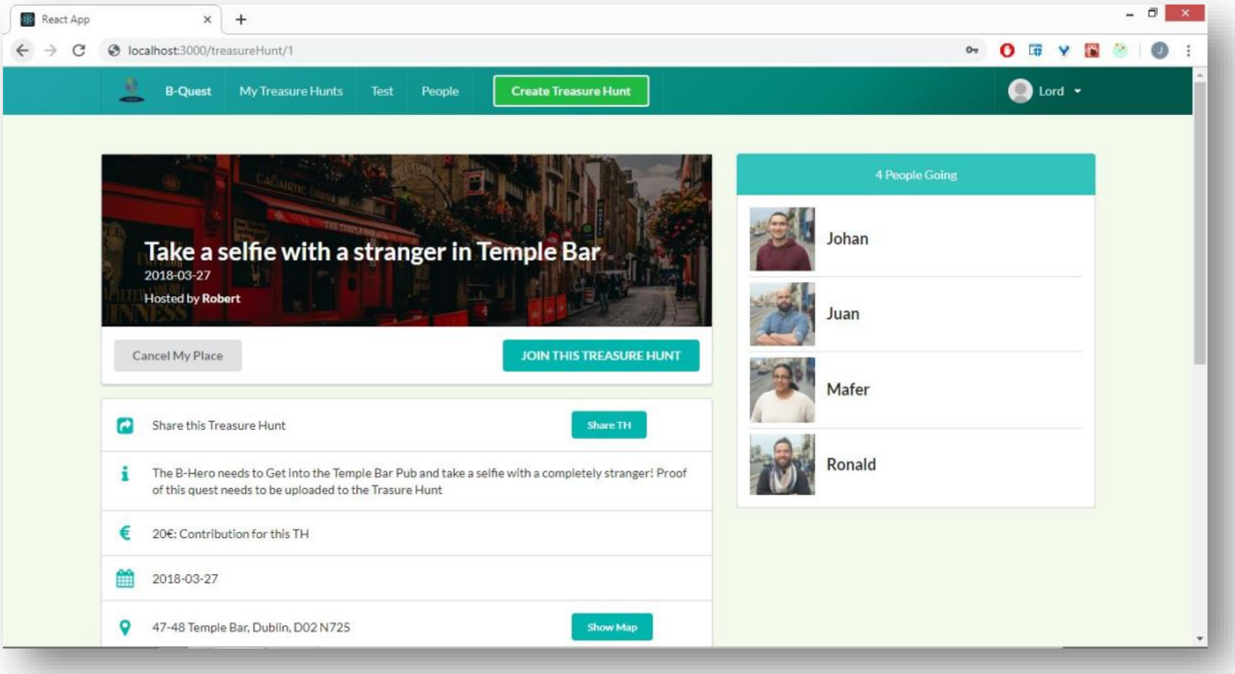


Figure 6.- Web Application - Chief.Create Treasure Hunt page



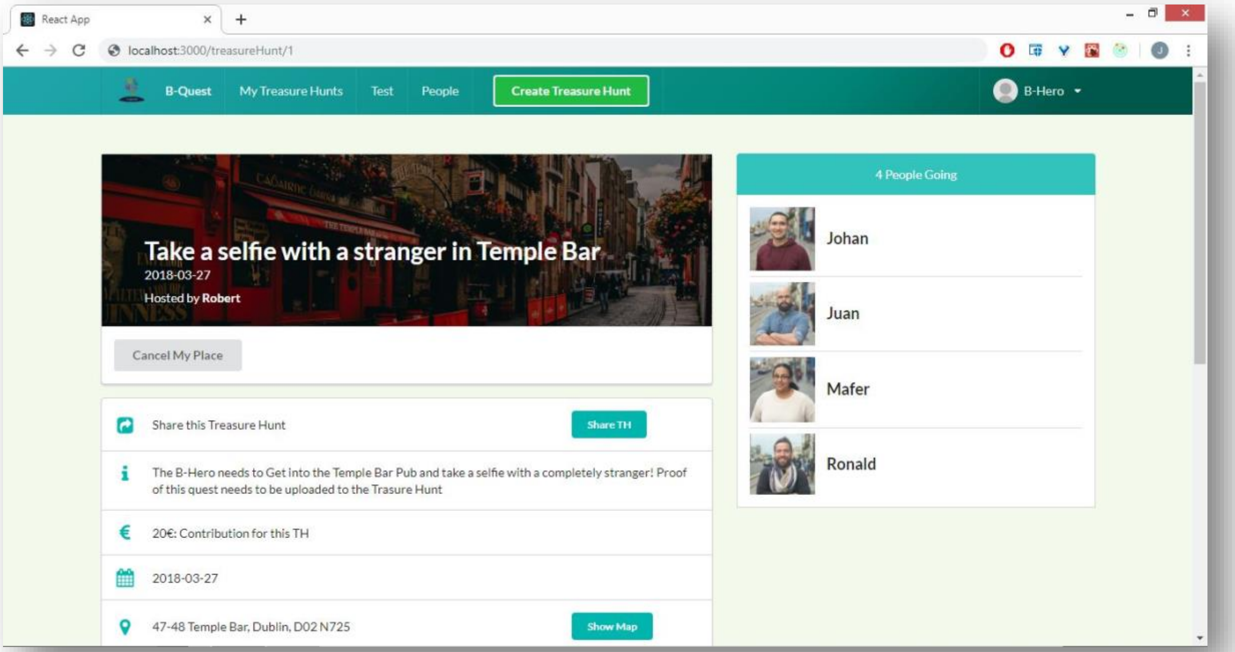
The B-Hero needs to Get into the Temple Bar Pub and take a selfie with a completely stranger! Proof of this quest needs to be uploaded to the Trasure Hunt

20€: Contribution for this TH

2018-03-27

47-48 Temple Bar, Dublin, D02 N725

Figure 7.- Web Application - Dashboard with Lord privileges.



Take a selfie with a stranger in Temple Bar
2018-03-27
Hosted by **Robert**

Share this Treasure Hunt

The B-Hero needs to Get into the Temple Bar Pub and take a selfie with a completely stranger! Proof of this quest needs to be uploaded to the Trasure Hunt

20€: Contribution for this TH

2018-03-27

47-48 Temple Bar, Dublin, D02 N725

Figure 8.- Web Application - Dashboard with B-Hero privileges

Same style and principles were applied throughout the whole web application in order to keep it as simple and functional as possible without forgetting the main objective. More examples of the flow of the web application can be found in the appendix.

Web Application Backend

Using Firestore in the Backend Side of the Web Application

The Web Application developed requires to store data in the cloud because every time a new Treasure Hunt is created, the details of this Treasure hunt needs to be shared with all the participants and also the Web Application will be synchronized with the Android application this means that access to the data should be available from more than one device. Storing data in the cloud is not easy as it looks, normally a set-up of own servers to host the data and keep them up running day and night has to be done which also means dealing with networking issues. Here is where Firestore has a number of useful functions to help in this regard.

I-Firestore Fabric Newsletter - https://blog.prototypr.io/seven-principles-for-effective-content-design-8b3f82840d3b

This tool works in near real time, automatically fetching changes from the database as they happen, or the data can also be requested manually.

It Also provides Authentication which is really important to the B-Quest Web Application, Users cannot participate, create or see any event unless they are registered and logged in the Application.

J-Firestore - https://blog.prototypr.io/seven-principles-for-effective-content-design-8b3f82840d3b

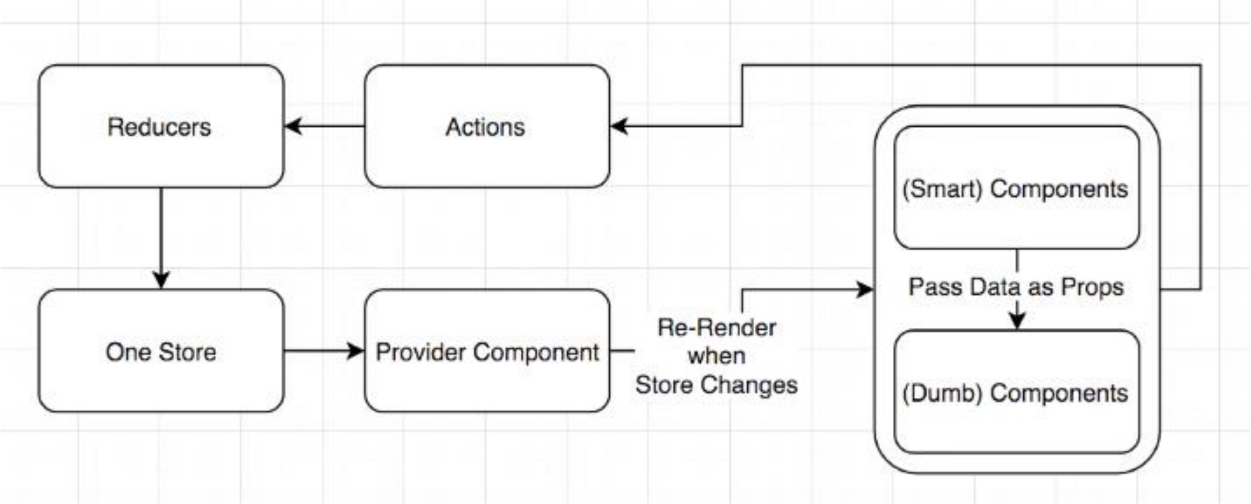
REDUX

Redux is a Predictable State container for JavaScript Apps, it is just not for React. It’s designed for any JavaScript application (REDUX, n.d.).

When developing B-Quest Web Application, Redux was the chosen one because its simplicity when managing the state of the data displayed and how the web Application responds to the user’s actions.

One of the most important features of Redux is it’s DevTools which makes it easy to trace when, where, why, and how our application’s state changed. This means that it really helps to avoid a lot of complicated bugs. Redux’s architecture lets our log changes, use “time-travel debugging”, and sends the specific error in detail to a server.

Redux is small, it’s just 2kb including dependencies. It’s like having a local database in the client.



- **Provider**: wraps your application, injecting the store

- **Store**: one large store that contains the state for your entire application

- **Reducers**: reducers listen to actions and make changes on the store values. They also cannot mutate the data on the store in any way but must return a new set of data.

- **Actions**: pretty much just like flux actions, the only difference is that async can be handled in multiple different ways depending on store "middleware"

- **Components**: React components can be injected with various pieces of store data. React components also trigger Redux actions. This is what makes it all come together.

Android application UI Design

A mobile user interface (mobile UI) is the graphical and usually touch-sensitive display on a mobile device that allows the user to interact with the device's apps, in fact is everything that the user can see and interact with. Because of that it is critical to plan every step of the design, and at some point, retreat and examine what it's been building to accomplish the goal of simplicity and usability. A golden rule is established, and it is that the users has to be able to understand a command icon and its meaning whether through legible text or comprehensible graphical representation.

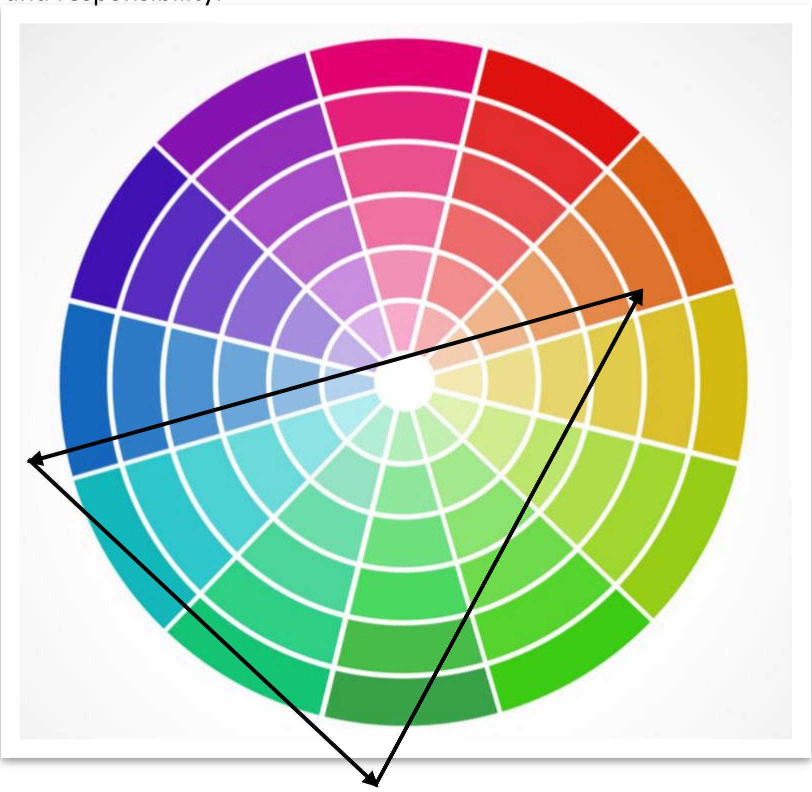
Colours

A colour complementary combination statement was used for the selection of these (GRAF1X, n.d.). This consists in a variation of the complementary combination, but instead of the opposite colour the two colours that are next to it are used. This combination is applied for a base colour that in the case of B-Quest is blue and two additional colours green and orange. These colours appear to be closely linked to words like:

Orange: The colour of encouragement. The combination of yellow and red makes orange conveys excitement, warmth and enthusiasm. Social and inviting, this is the colour of the extrovert, exuding happiness and joy, releasing inhibitions.

Green: The colour of growth and health. Think of nature and see green in all its glory expressing renewal and life. Green has a strong association as a refreshing and peaceful colour.

Blue: The colour of trust. Blue, the shade of the sea and the sky, is thought to induce calm and convey tranquillity, serenity and peace. The popular colour instils confidence and inspires feelings of trust, loyalty, integrity and responsibility.



This combination gave B-Quest the final colour for its UI Design and was the bluish green.

Bluish green is not only the combination of the colours is also the combination of values that we want to transmit to the users which are encouragement, enthusiasm, trust, tranquillity, serenity, peace, loyalty, integrity and responsibility.

This bluish green is to be applied on those components were users should focus on, such as, logo and buttons, and relevant information shown on text.

As for the background, a light background (white) was chosen, which relates to purity, cleanliness, transparency and honesty. It offers a sense of peace and calm while creating a sense of order and efficiency.

Content

The main idea for this process is to provide the user with all the information necessary to create Treasure Hunts, add Quests to the Treasure Hunts, invite people to it and have access to their own Treasure Hunts. For this it we considered the following principles recommended by Rachel Mc Connell, 2017, "Seven principles for effective content design" on the blog Prototypr, in order to make sure the content is effective and efficient.

1. *Purpose*

When taking into consideration the purpose of the application some questions should be answered from the beginning of the development. Some examples of these are:

- "What is the issue or opportunity we're trying to address?" – In the case of B-Quest the opportunity to learn how to and develop an Android application that could add an extra in the resume of knowledges not included in the Academic Curricula. Also is the opportunity to develop something with the guidance of a Supervisor, teachers and other without less risks than out in any workplace.
- What do we want our user to think, feel or do? – The application is for a target in specific, it is for a group of people inclined to games, adventures and technology but most of all people who likes sharing nice experiences with other people.
- What will success look like? – The application will be available in the Google Play Store, so people could start planning Birthday Parties in a different way.
- Has anything already been designed or tested? – Not yet. After researching, nothing exactly like it was found.
- Are there any constraints we need to know about? – Knowledge of some technologies, specific target, timelines.

2. *Context*

Why the user lands on the content of the application? – It is because of the its context. They are looking for something in specific that could make their context more manageable.

When thinking about this user's context has to be taken into consideration. For example, in the case of B-Quest Mobile application the user most probably will be a multitasking one. Therefor some features in the app are to be considered as main goals: clear navigation, quick response, easy to understand and use.

That's the reason why B-Quest will include an initial video talking about the purpose of the application. And if it catches the user's attention because it could be efficient for them, then the user will register on the application.

3. *Accessibility*

Different people, different abilities. The content should be accessible for everyone. The application should think about adding not only text but images, diagrams, videos… for this reason it would be ideal to consider working with a designer.

For B-Quest unfortunately, a designer was not an option, but these aspects were taken on consideration when developing the application. B-Quest is keeping everything short, simple and explicit enough for the user to understand.

4. *Clarity*

Language on the application should be clear for a better understanding of the user. The principle indicates not using jargon, nevertheless part of the soul of B-Quest is its own terminology for this type of application.

Most of the action labels are kept simple and the jargon used is explained on a tutorial where the user will be able to see the definition for the not so simple labels. For example: user will be able to see that in order to create an event they will have to think of it as a Treasure Hunt.

5. *Brevity*

"Less is more when it comes to content."

Only the most important content should be kept on each screen as users normally only take in a low percentage of that content. This another reason why making it more accessible could be beneficial, as more images the user can see more information may be taken in.

6. *Relevance*

When talking about relevance, the user experience context and the purpose of them downloading the application are to be taken into consideration. Relevance refers to what should be included to make the application more user-friendly, welcoming and warm. Is all about the application connection with the human side.

B-Quest is developed for a specific target, and it is like that because most of the team relates on the same category that the target. People who likes learning new things, technology, games, adventures and most of all for pro-people.

7. *Flexibility*

Once the application is developed is important to keep in mind that it may be subject to changes, and this is where the flexibility kicks in. Processes like testing or even changes in the laws may make the application to change some of its features and the developers should be able to make those changes without altering the app itself too much. The application needs to be flexible in order to be successful and long-lasting.

Layouts

B-Quests layouts were designed thinking first in the users, and following **Nick Babich's** fundamentals in "The Guide to Mobile App Design", found in the blog Uxpin.

For the layouts it is important to keep in mind the number of actions required to complete a task, users nowadays tend to get bored easily of an application if this requires high levels of interactions to be successful on a task.

The position of every component on this mobile application has been carefully selected to make it as much intuitive as possible.

Some of the fundamentals followed to design the layout B-Quest Mobile application are:

1. *Minimize Cognitive Load*

Make the application as clear as possible for the user. The most user-friendly the better.

2. *Optimized User Flow*

In order to make an ideal application, the user's flow has to be taken into consideration and this should lead successfully to the user's goal. Some steps could be included in this process such as:

- Provide natural next steps for the user to finish a task. The UI should guide naturally the user.
- Keep one primary action per screen. This will make the UI more user-friendly.

By studying the flow of points of friction can be highlighted and therefor amended.

When designing B-Quest this aspect was a vital one taken into consideration. B-Quest doesn't require the user to go through many steps to finally reach a goal inside the application.

3. *Cut Out the Clutter.*

Similar to the principle of Brevity recommended by Rachel Mc Connell, 2017, "Seven principles for effective content design", the idea is to provide the user with only the relevant information in term of user interface design. Keep only the buttons needed, the images, icons… specially on a mobile application where the size of the screen limits even more the user interaction with the application.

4. *Make Navigation Self-Evident*

Keep the features visible and obvious for the user, so it can make clear for it what is its current location for example, or where the menus are without having to figure it out by leaving the application.

A good example for B-Quest is the side menu, that is indicated by a burger icon and is accessible from most of the screens.

5. *Optimize Interactions for the Medium*

Take into consideration that a mobile application development has to differ with a web application as mobile devices have some constraints in different aspects. For example: screen size.

This is one of the main reasons why B-Quest Web application is different in some aspects from the mobile application. On the web application more features could be shown per page.

6. *Design Finger-friendly Tap-Targets*

The user has to be able to easily tap the actionable elements in the mobile application. In order to do this, these elements should be big enough and for this a "Rule of Thumb" was considerated. This rule indicates that the Tap-Targets should have a measurement area of 7-10 mm to make it accurate when tapped and visible enough for the users to find.

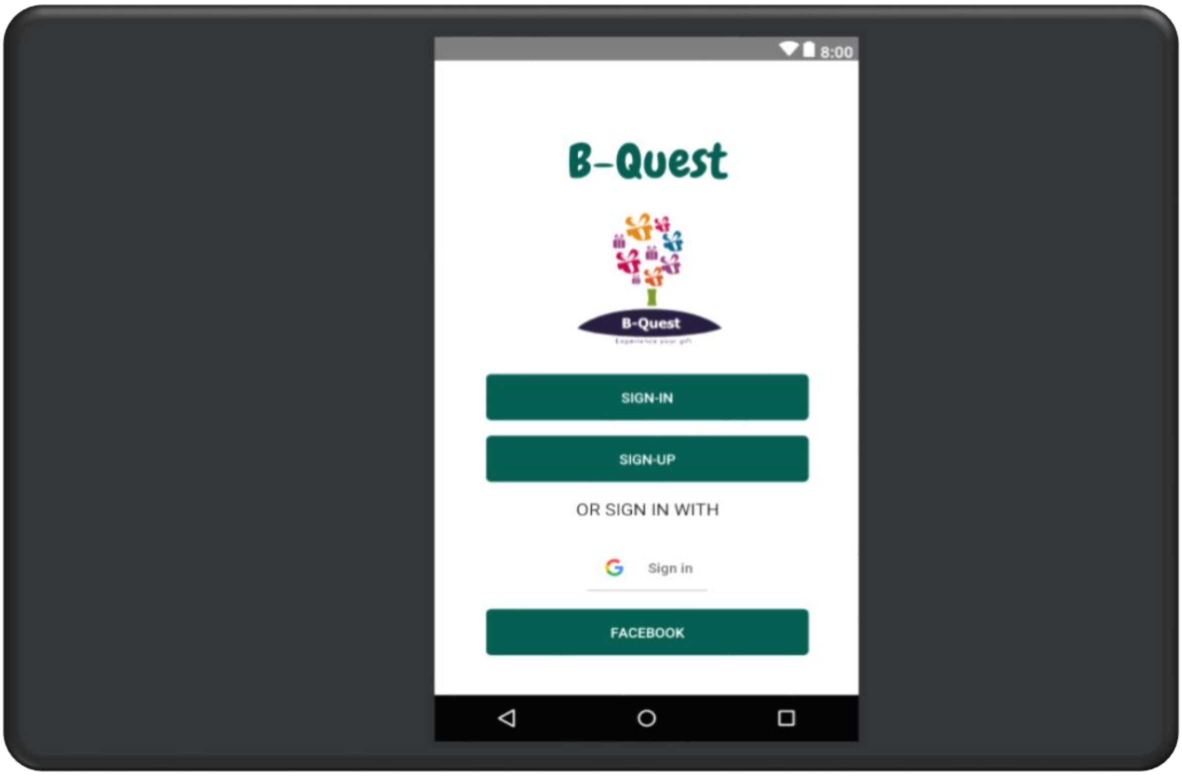
Android Activity

First and foremost is important to define the meaning of an Activity in an Android Development context. An Activity refer to the window provided where the application can draw its UI.

"An Android activity is one screen of the Android app's user interface. In that way an Android activity is very similar to windows in a desktop application. An Android app may contain one or more activities, meaning one or more screens. The Android app starts by showing the main activity, and from there the app may make it possible to open additional activities" - Jakob Jenkov, 2014. "Jenkov.com Tech and Media Labs"

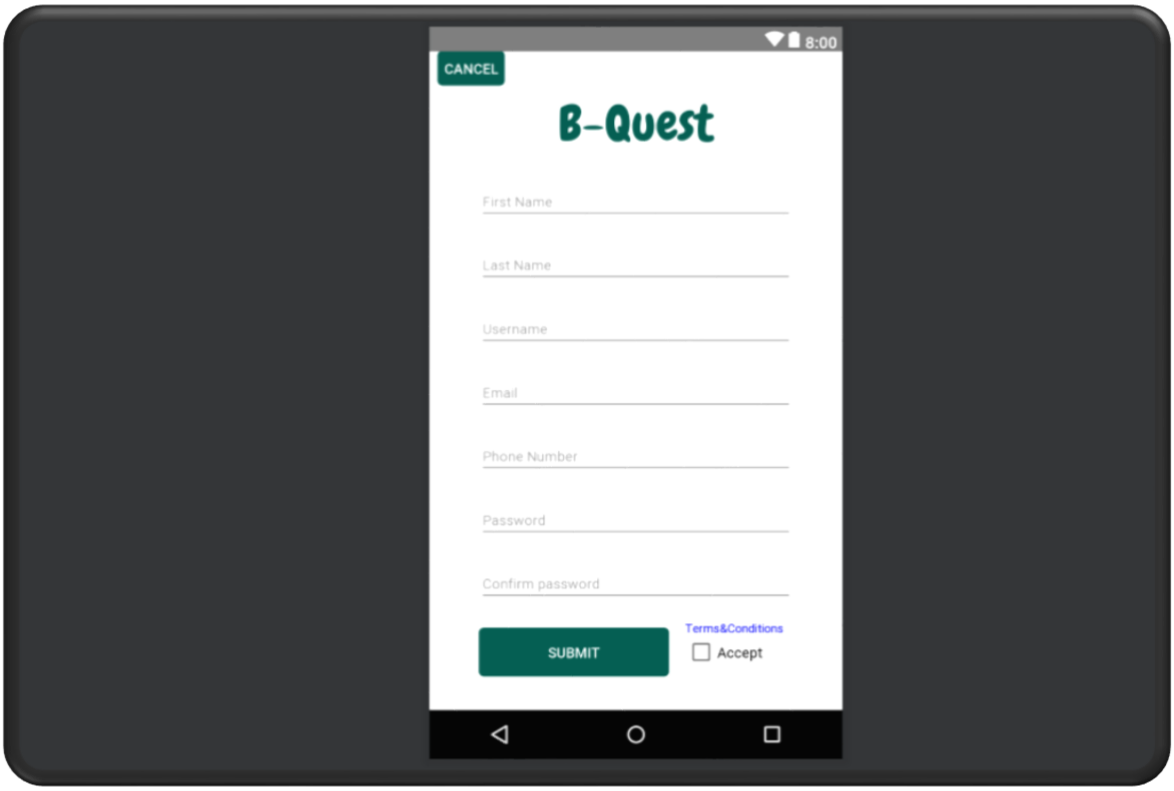
In the following images, you will see a series of mock-ups in order to better illustrate these layouts.

Landing Page Activity



The Landing Page provides the user with access to the Sign In, Sign Up, Sign In with Google account and also Sign In with Facebook account. The idea is to make it as simple as possible but at the same time make it as much intuitive as possible.

Sign Up Activity

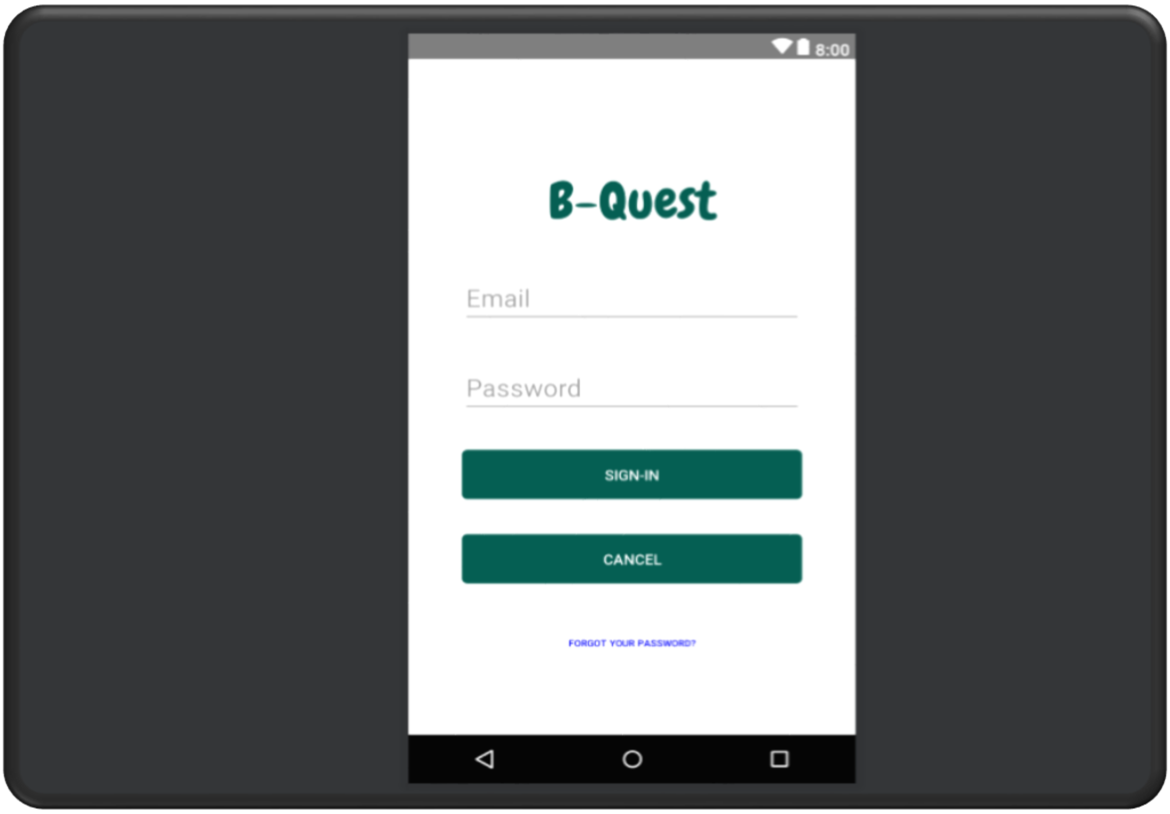


The Sign-Up Page provides the user with a series of fields First Name, Last Name, Username, Email, Phone Number, Password and Confirm Password, all this fields are required to create an account.

This page complies with the User Control and Freedom principle by providing a way for the user to go back to the Landing Page in this case pressing the cancel button, also contains a check box in which the user must click to accept our terms and conditions.

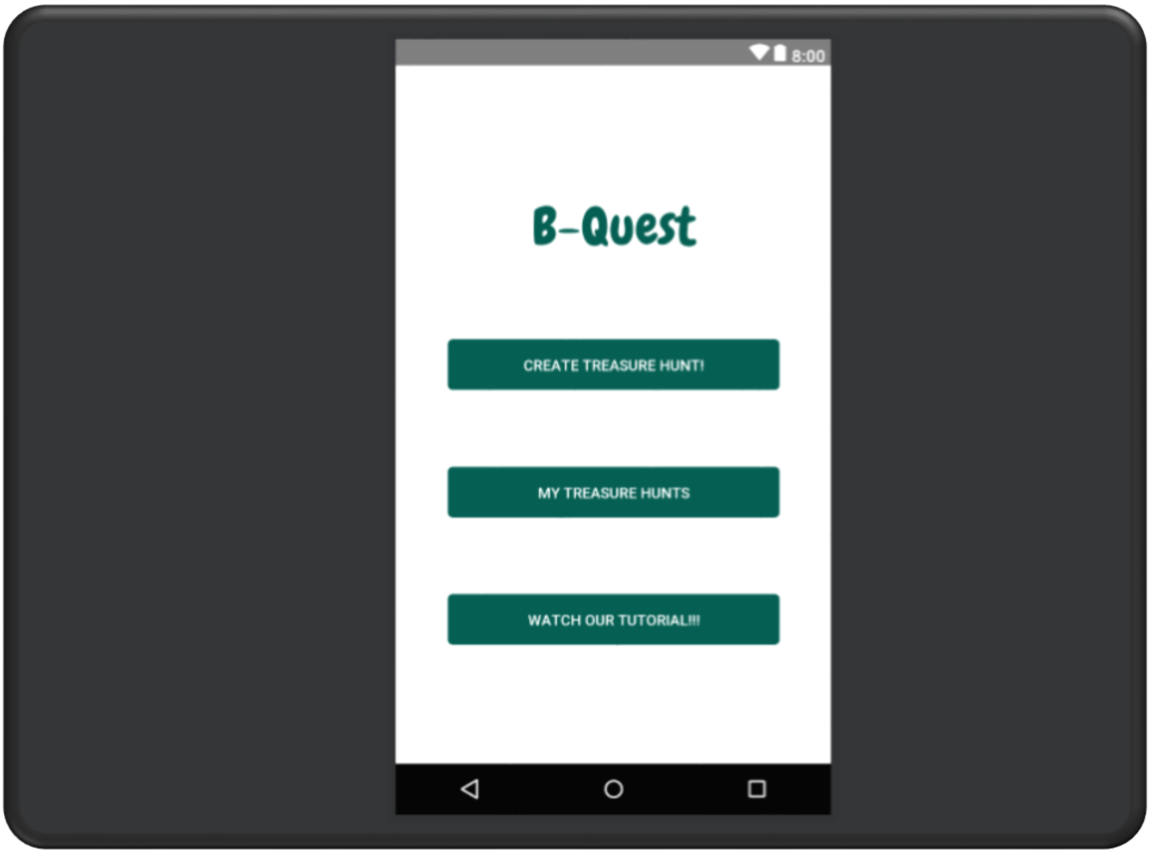
Validations warning are to be displayed highlighting the respective input control with a pop-up for a few seconds, making aware of any invalid input the may be typing, hence, minimizing the chances of the system throwing errors due to invalid inputs.

Sign in Activity



The Sign-In Page provides the user with two fields Email and Password these two are required in order to Log-In into the application. This page also complies with the User Control and Freedom principle by providing a way for the user to go back to the Landing Page in this case pressing the cancel button and a Sign-In button that allows the user Log-in into the application.

Dashboard Activity

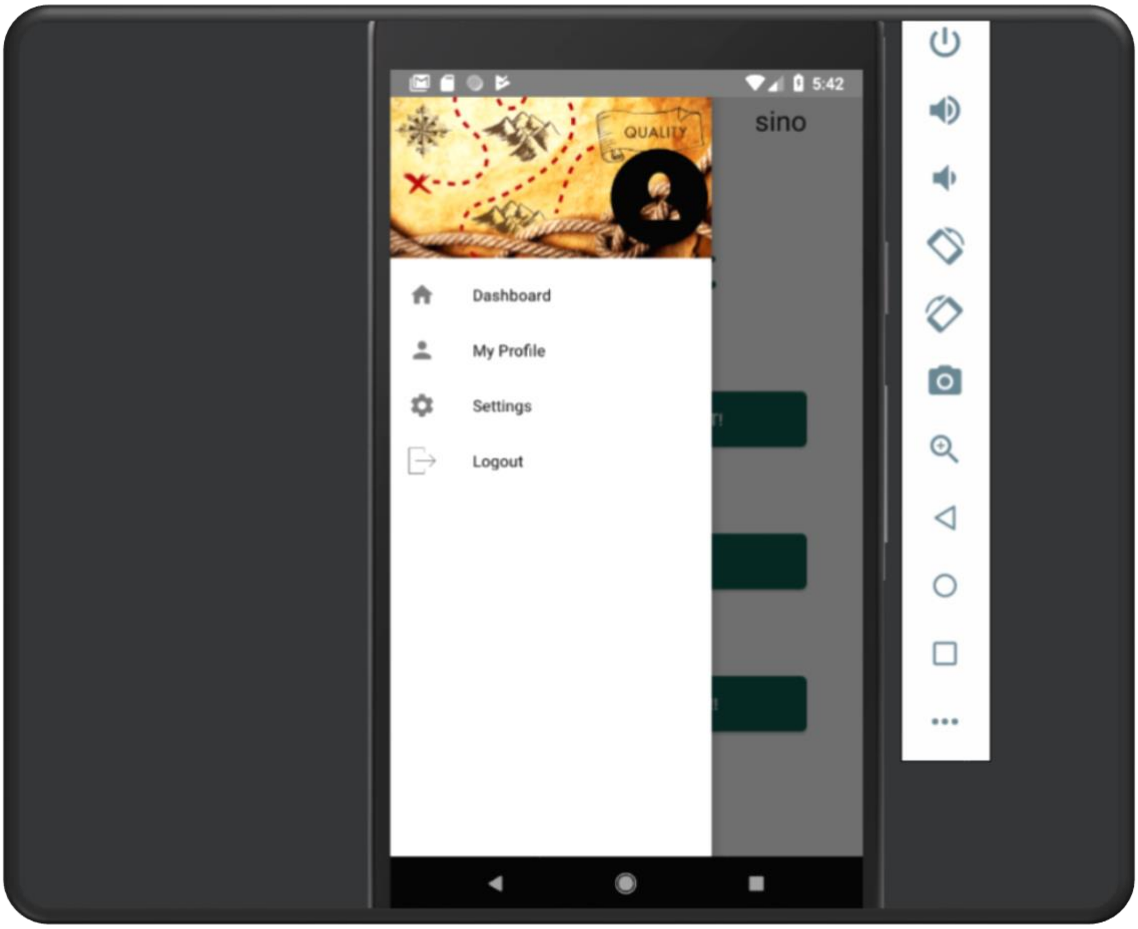


The Purpose of this page is to present the user in a very simplistic and intuitive way the options available at this stage. The user is presented with three buttons:

1. ***Create Treasure Hunt (Button):*** It allows users to go and create an event for the birthday person. By clicking this button, the user will be redirected to the Creating Event Activity.
2. ***My Treasure Hunts (Button):*** It allows users to go and check all the events in which the user is included and in which the user could join.
3. ***Watch Our Tutorial (Button):*** This button allows users to watch the tutorial video redirecting the user to YouTube.

Complying with the User Control and Freedom principle the user can Log-out at any time by pressing the menu button at the top left and clicking the Log-out button.

Menu Activity

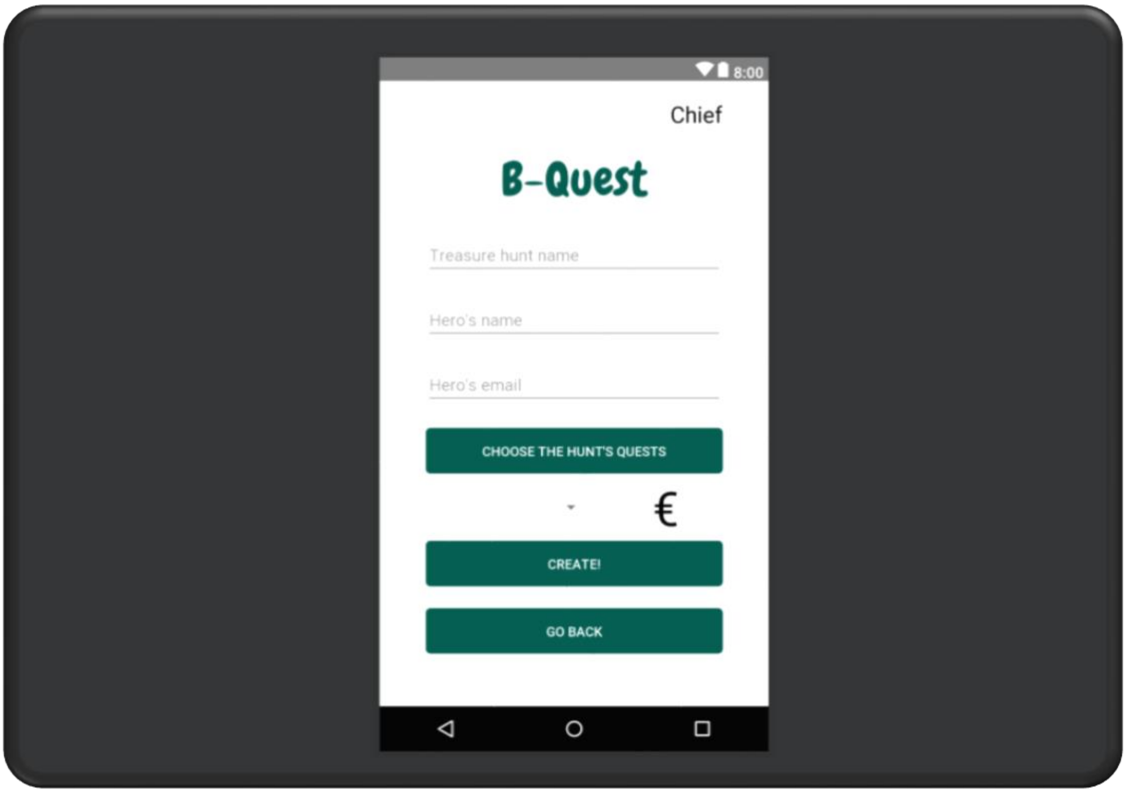


On this activity, the user is presented with a very simplistic an intuitive menu that show an image view (profile picture) that will be displayed once the user updates the profile.

Four options on the menu will be displayed:

1. ***Dashboard:*** Once this is pressed the user will be redirected to the Dashboard Activity.
2. ***My Profile:*** Once this is pressed the user will be redirected to the Profile Activity where further options will be available.
3. ***Settings:*** Once this is pressed the user will be redirected to the Settings Activity where further options will be available.
4. ***Logout:*** Once this is pressed the user will be redirected to the Sign in Activity letting the user know the end of the session with a pop-up. Is important to highlight the importance of the Logout because this ensures that user access and user credentials are safe after the login session.

Creating Event Activity



On this activity is very important to highlight the status of the user on the top right (Chief) meaning that this person is now responsible for creating this event. This page provides the user with three very important fields:

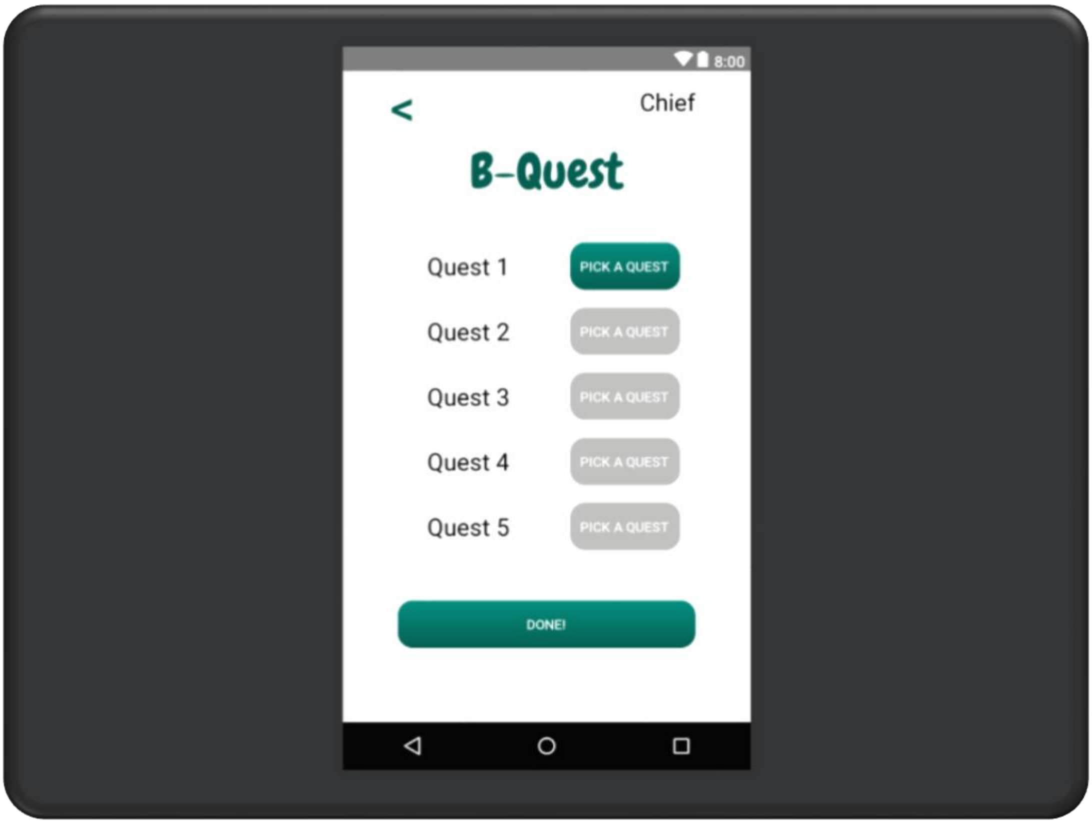
1. ***Treasure Hunt Name:*** This field will define the name of the event for the birthday person.
2. ***B-Hero's Name:*** This field will hold the birthday's name, and this will be the name to show on the invitation for others user.
3. ***B-Hero's Email:*** This field will hold the birthday's email and is one of the most important because with that information an email will be sent to the birthday person with the description of the event.

Choose ***The Hunt Quest (Button)*** will redirect the user to the Choosing Category and Quest Activity.

On this activity the user is also provided with an arrow button which displays a dropdown list of different amounts of money that the user can choose as a present for the birthday's person.

Complying with the User Control and Freedom principle the user can Log-out at any time by pressing the menu button at the top left and clicking the Log-out button, also the user is presented with a go back button which redirect the user to the Dashboard Activity.

Choosing Category and Quest Activity

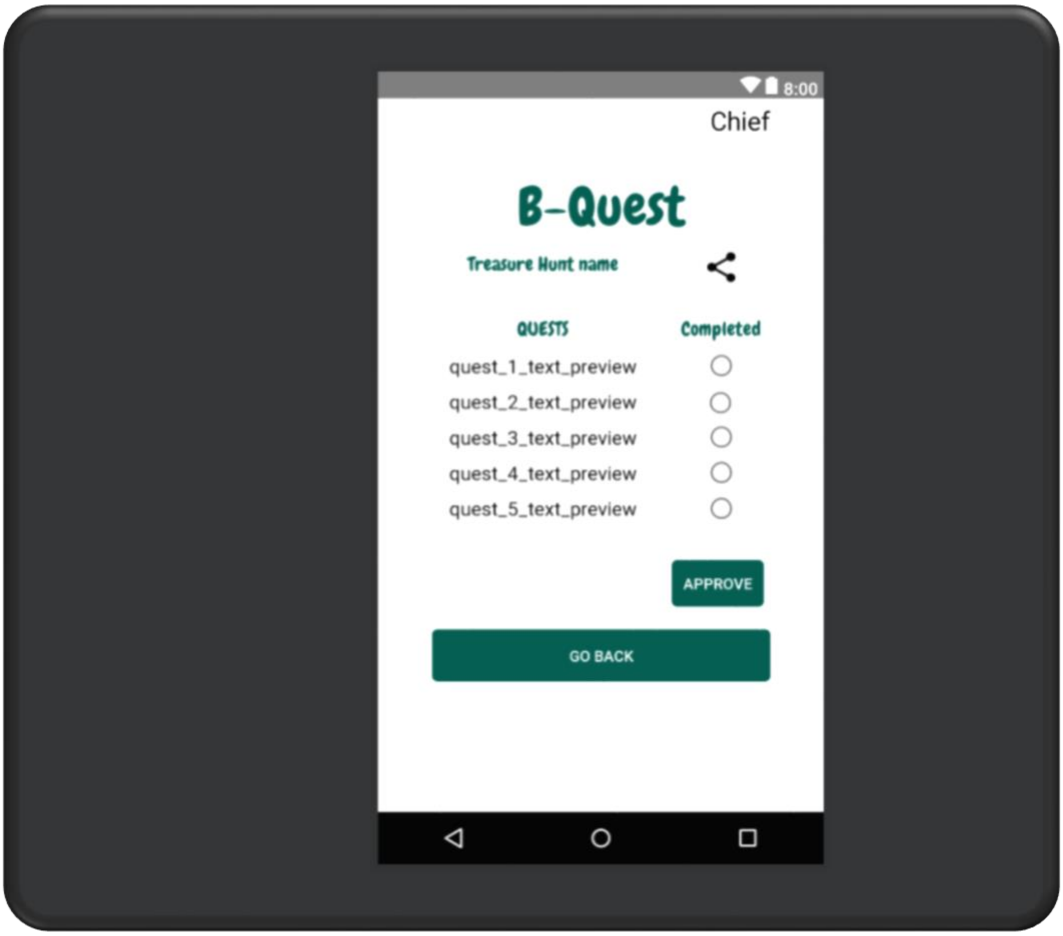


On this activity the user is presented with five buttons in which each belongs to a specific Quest. Is also important to highlight the status of the user on the top right (Chief) meaning that this person is now responsible for creating this event.

Each button will redirect the user to another activity and this activity will display a series of Categories (five) and inside of those categories five different quests that the user must pick to create every quest.

Complying with the User Control and Freedom principle the user can Log-out at any time by pressing the menu button at the top left and clicking the Log-out button, also the user is presented with a DONE button which redirect the user to the Dashboard Activity and an arrow button to go back to the Creating Event Activity.

Approving Quest Chief Activity



QUESTS	Completed
quest_1_text_preview	○
quest_2_text_preview	○
quest_3_text_preview	○
quest_4_text_preview	○
quest_5_text_preview	○

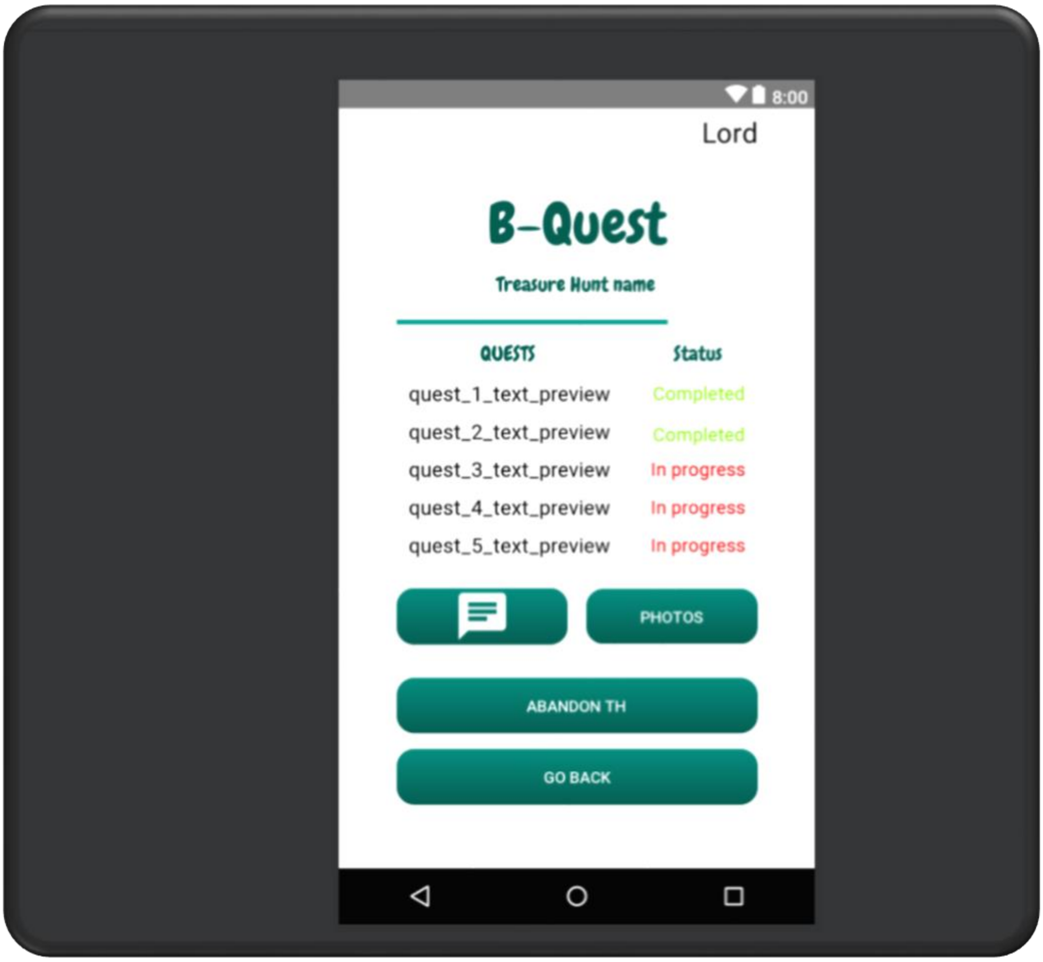
On this activity the user is presented with the Treasure Hunt Name (name for the event) and just beside the name a share button which will allow the user to share this event with any other user. Is also important to highlight the status of the user on the top right (Chief) meaning that this person is now responsible for creating this event.

On this screen the user is presented with five radio buttons that belongs to a specific quest, the user can click on every radio button in order to decide if the quest had been completed.

An Approve button is also presented to the user, once this button is pressed the birthday person will get a notification that the person in charge of this event has approved all the quests and the user will be redirect to the Dashboard activity.

Complying with the User Control and Freedom principle the user can Log-out at any time by pressing the menu button at the top left and clicking the Log-out button, also the user is presented with a go back button which redirected to the Dashboard Activity.

Lord Activity



QUESTS	Status
quest_1_text_preview	Completed
quest_2_text_preview	Completed
quest_3_text_preview	In progress
quest_4_text_preview	In progress
quest_5_text_preview	In progress

On this activity is very important to highlight the status of the user on the top right (Lord) meaning that this person is a participant on this event. But this title do not give certain privileges to this user.

On this activity the user is presented with the Treasure Hunt Name (name for the event) and just below a progress bar that shows the how the birthday person is evolving on their tasks.

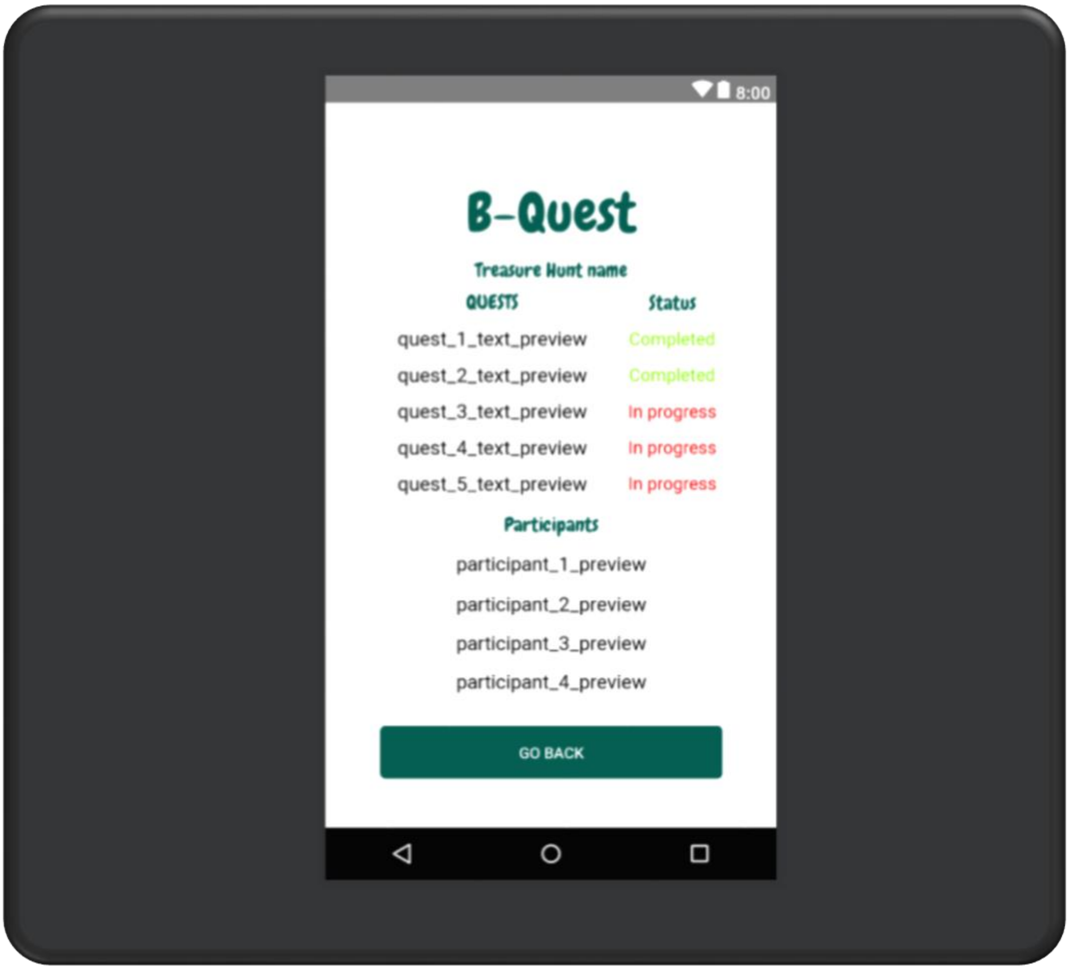
The user is presented with a list of quest and beside a list of the status of the quest (this status is either completed or in progress) and this occurs when the birthday person uploads a picture.

A chat button is also presented to the user once the person clicks on this button it will be redirect to the chat room, this will be another activity.

A photo button was created with the purpose of showing the user all the uploads that the birthday person in doing just by clicking this button the user will be redirect to the photo’s activity.

Complying with the User Control and Freedom principle the user can Log-out at any time by pressing the menu button at the top left and clicking the Log-out button, also the user is presented with a go back button which redirected to the Dashboard Activity.

Preview Events Activity



QUESTS	Status
quest_1_text_preview	Completed
quest_2_text_preview	Completed
quest_3_text_preview	In progress
quest_4_text_preview	In progress
quest_5_text_preview	In progress

Participants

participant_1_preview

participant_2_preview

participant_3_preview

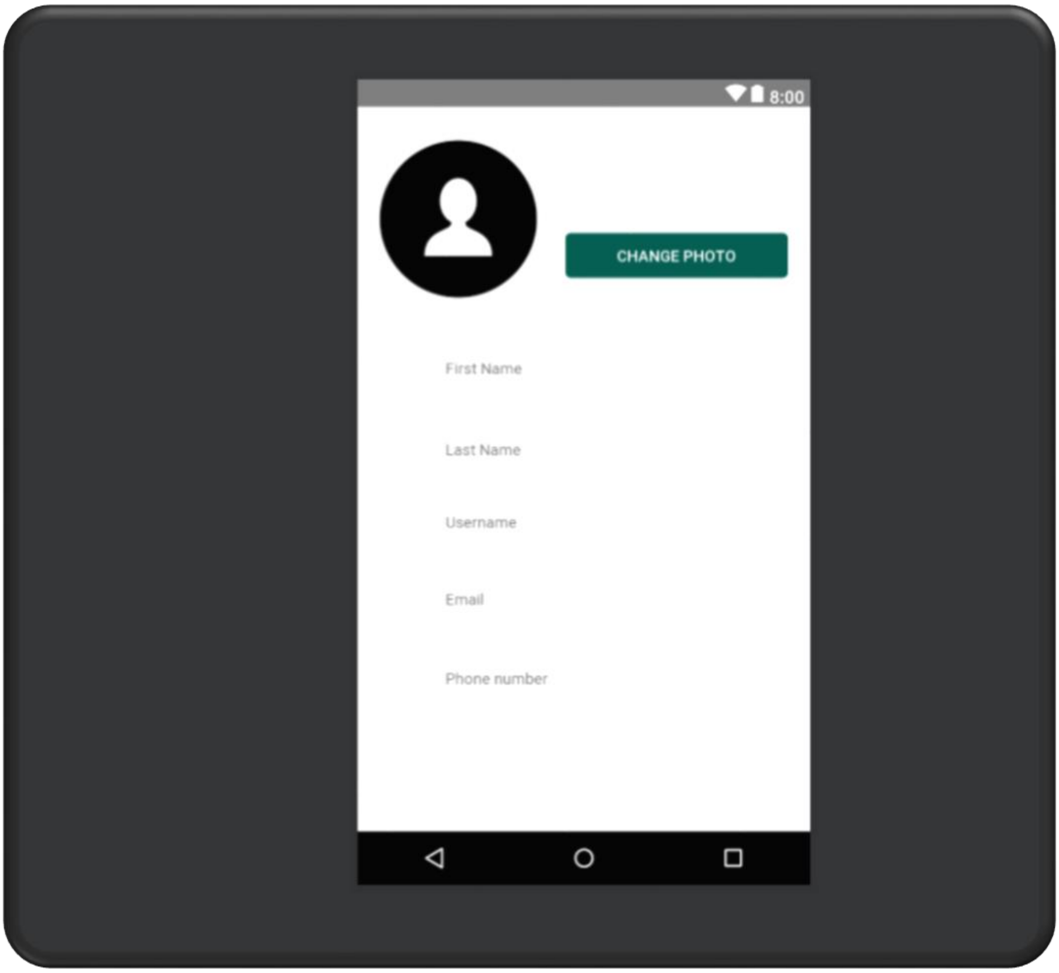
participant_4_preview

On this activity the user is presented with the Treasure Hunt Name (name for the event), just below and very similar to the Lord Activity the user will encounter list of quest and beside a list of the status of the quest (this status is either completed or in progress) and this occurs when the birthday person uploads a picture.

Right below a list of participants will shows the user all the people involve on this event.

Complying with the User Control and Freedom principle the user can Log-out at any time by pressing the menu button at the top left and clicking the Log-out button, also the user is presented with a go back button which redirected to the Dashboard Activity.

Profile Activity



On this activity the user is presented with image view (profile picture) and a Change Photo button, once this button is pressed it will ask the user for permission to gain access the internal storage device (photos, videos, gif etc), the user will be able to select any picture and set it as a profile picture.

Right below the user will be able to see the First Name, Last Name, Username, Email and Phone Number, we took this information from the Sign-Up activity where the person is creating an account and now this account is shown to the user in this format.

Complying with the User Control and Freedom principle the user can Log-out at any time by pressing the menu button at the top left and clicking the Log-out button, at the same time the user would be able to press the menu button just to hide the menu at any stage in the app.

FUNCTIONAL DESIGN

It is very important for us to highlight the purpose of the functional design is to specify a system's actions in a form that allows developers, users, stakeholders, and sponsors to reach a consensus. In order to create a product that works, there are seven questions that has to be kept in mind about the product.

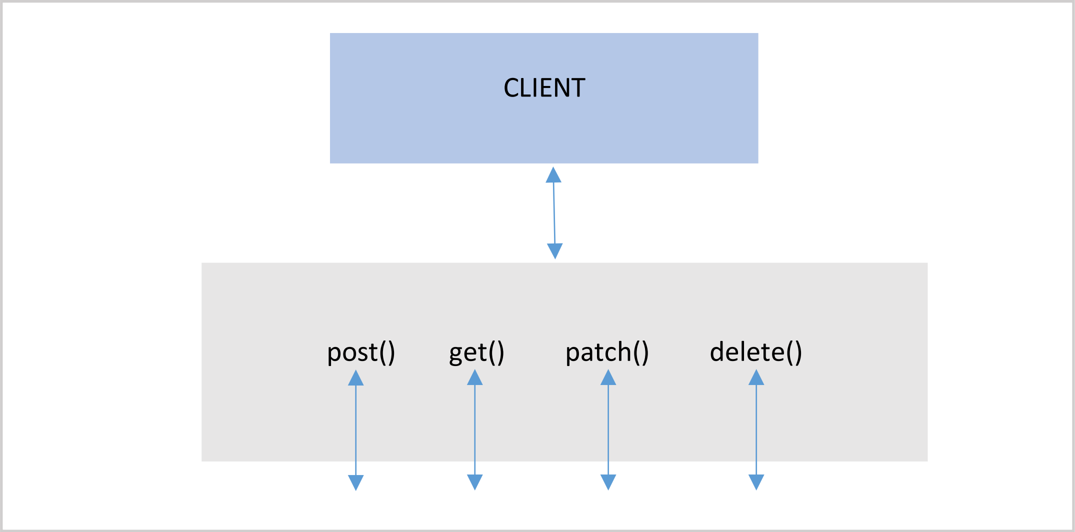
- What is the final product's goal – For B-Quest the final goal is to create an application that will allow users to plan and enjoy Birthday parties in a different way.
- Consider who will be using it – Mostly Techies and Gamers.
- Consider what your audience intends to do with it – Keep it simple and keep on mind what's does the user really wants to do if uses the application.
- Is it clear how to use it? – Keep the application accessible for everyone.
- How does your user know it's working? – The application has to be clear, user-friendly and intuitive enough to assure the user it is working with minimal instructions.
- Is it engaging to your user? – If the application is successful in terms of relevance, clarity, and self-evident and it accomplish the user goals the application will be successful.
- How does it handle mistakes? – The application should be flexible and with a good design to make room for mistakes to handle.

B-Quest is a server-less application called "Native Mobile Frontend" and a backend composed by a platform called "Firebase" that provides automatic data synchronization, authentication services, messaging, file storage, analytics, and more, whose role is that of acting as a RESTful online file web service.

The RESTful API is an architecture style that allowed the developer to design the components of this system as isolated units, totally decoupled from one another.

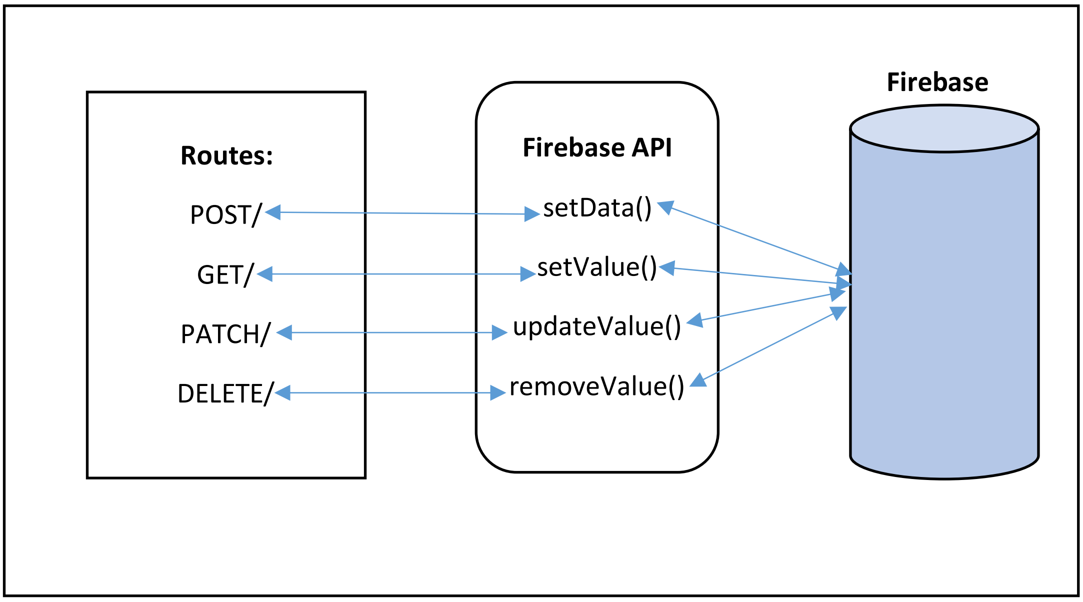
Frontend

The frontend is composed by different components. Think of this components as clients since they will require some sort of data service in order to carry out their tasks. This service is provided by the backend; since they form part of a server-less application they can't communicate with the backend directly.



Backend

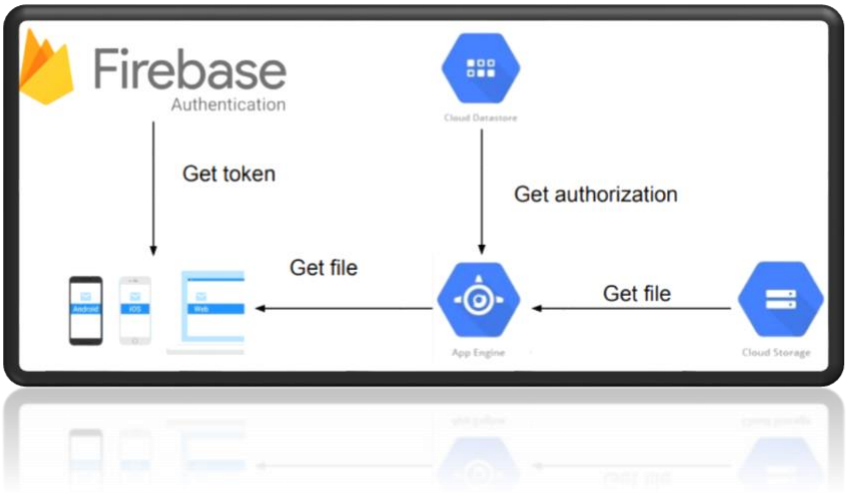
For the backend the Authentication function from Firebase is being used, which is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.



How does it work?

To sign a user into the application, first authentication credentials from the user are needed. These credentials can be the user's email address and password, or an OAuth token from a federated identity provider. Then, these credentials has to be passed to the Firebase Authentication SDK. B-Quest's backend services will then verify those credentials and return a response to the client.

After a successful sign in, access to the user's basic profile information is available, so a control to the user's access to data stored in other Firebase products can be done. Additionally, a use of the provided authentication token to verify the identity of users is in the backend services.



DATA DESIGN

When selecting a database for the mobile application and the web application a scalable and flexible one was necessary. It is for that reason, and few more, that the selected one is Firestore (Firestore, 2019), which is a NoSQL database. With Cloud Firestore and its elements, both mobile and web applications are able to connect simultaneously so all the data coming from both ends will be storage in this DB.

Firestore is a scalable document NoSQL Data Store in the cloud, that provides live synchronization, offline support, storing and querying data services to mobile, web and IoTs applications. Additionally, it includes the same security features than Firebase and Google Cloud platform as they are integrated.

As a Cloud Hosted Document Database, Firestore offers different advantages that will benefit B-Quest making it a solid application infrastructure. For example:

- Firestore is **Serverless** as it uses the same services than Firebase related to the management of Servers. This means that data will be saved directly in Firestore and pull from the same place, avoiding to the end user having to go through any server as they also connect directly to Firestore in the cloud.
- **Result Set Scaling**: Firestore offers a High Scaling Performance as it can handle heavy database workloads. It takes advantage of the Google Cloud Platform infrastructure to scale horizontally in and out, depending on the application load.
- **Offline Support**: Firestore uses the data cached to read, listen to, write or query even if the device is offline and then sync all the changes once the device is online.
- **Realtime Updates**: Once the data in the backend chances Firestore provides real-time update in the application across multiple devices, maintaining the data synchronized in all the devices an individual user could have.
- **Complex Queries**: It doesn’t take into consideration how complex the query is or how many records could be in a Collection as it uses the Zigzag Merge Join Algorithm and Composite Queries functionality. Additionally, in Firestore the time it takes to run a query is proportional to the number of results to be received, and not to the number of Documents to be searched through.

Firestore also offers additional functions due to its integration with Firebase and Google Cloud Platform products, providing most of the benefits of the mentioned platforms in one place.

B-Quest Database Design

First, let’s talk about the structure of Firestore. As a NoSQL Document Collection database, it doesn’t use tables, columns, rows… Instead, Firestore uses a strict pattern of Collections, Documents and Sub-Collections. Normalization is not needed either as redundancy is allowed. What’s the structure’s pattern:

1. Define the COLLECTION
2. Add DOCUMENTS
3. To each DOCUMENT Add SUB-COLLECTIONS.
4. Each SUB-COLLECTION could contain its own DOCUMENTS.
5. And so on…

In the case of B-QUEST the structure of the Database will be Collections, Documents and the use of Sub-Collection will be implemented to keep details more organized.

As Firestore is the chosen Database for B-Quest, the structure will be the following:

*Note: please refer to Chapter II for specific details about functionality.

User will be a collection with documents for each registered user.

```
{
   user_id: ObjectID
   name: string
   email: string
   phone_number: string
   friends {
      user_id(friend): ObjectID (will get name here)
      user_id(friend): ObjectID (will get name here)
      user_id(friend): ObjectID (will get name here)
      user_id(friend): ObjectID (will get name here)
   }
}
```

Quest category will be a collection that contains documents per category. Each of this documents will have a sub-collection ***quest*** that will contain individual documents for each quest related to that category.

```
{
   category_id: ObjectID
   category_name: string
   description: string
}
```

Quest

```
{
   quest_id: ObjectID
   title: string
   description: string
   difficulty: number
}
```

Event will be a collection that contains the documents per event. For each event document will two sub-collections will be stored, one storing private data (invisible for the client) and one public data (visible for the client), in both cases the data will be stored as key/value set within a single document inside private or public sub-collection. This will allow to set rules and permissions based on the "role", I.e. you can only delete an event before competition if your role is “chief” or participate (have access/follow) an event if your payment status is positive.

```
{
   event_id: ObjectID
   title: string
}
```

Private data

```
role {
   user_id: "chief"
   user_id: "lord"
   user_id: "lord"
}
payment {
   user_id: "yes"
user_id: "yes"
   user_id: "no"
}
```

Public data

```
members {
   user_id: "name"
   user_id: "name"
   user_id: "name"
}
quest_list {
   quest_id: title
   quest_id: title
   quest_id: title
   quest_id: title          }
```

CHAPTER IV
IMPLEMENTATION OF THE SYSTEM

TECHNOLOGIES INVOLVE MOBILE APPLICATION

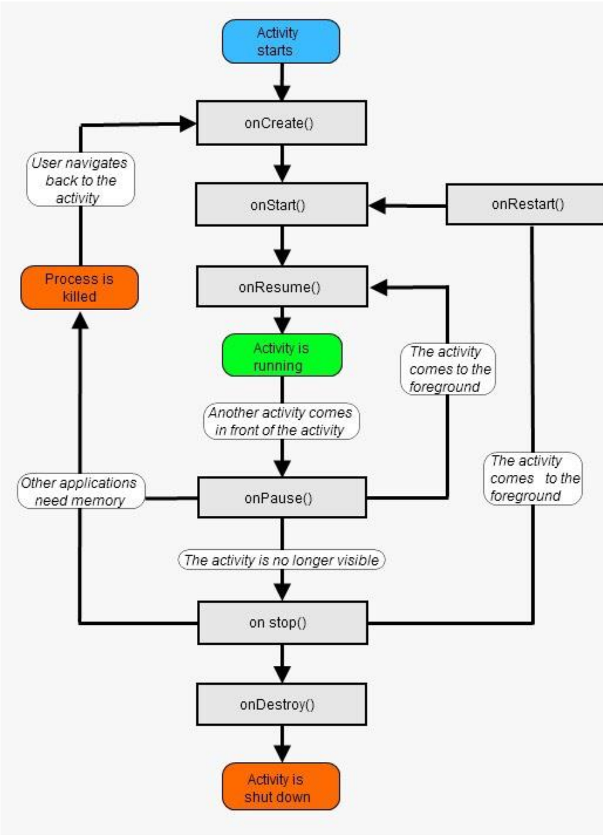
Java API Framework (1)

The entire feature-set of the Android OS is available through APIs written in the Java language. These APIs form the building blocks that need to create Android apps by simplifying the reuse of core, modular system components and services, which include the following:

- A rich and extensible **View System** can be used to build an app's UI, including lists, grids, text boxes, buttons, and even an embeddable web browser.
- A **Resource Manager**, providing access to non-code resources such as localized strings, graphics, and layout files.
- A **Notification Manager** that enables all apps to display custom alerts in the status bar.
- An **Activity Manager** that manages the lifecycle of apps and provides a common navigation back stack.
- **Content Providers** that enable apps to access data from other apps, such as the Contacts app, or to share their own data.

Activity Lifecycle

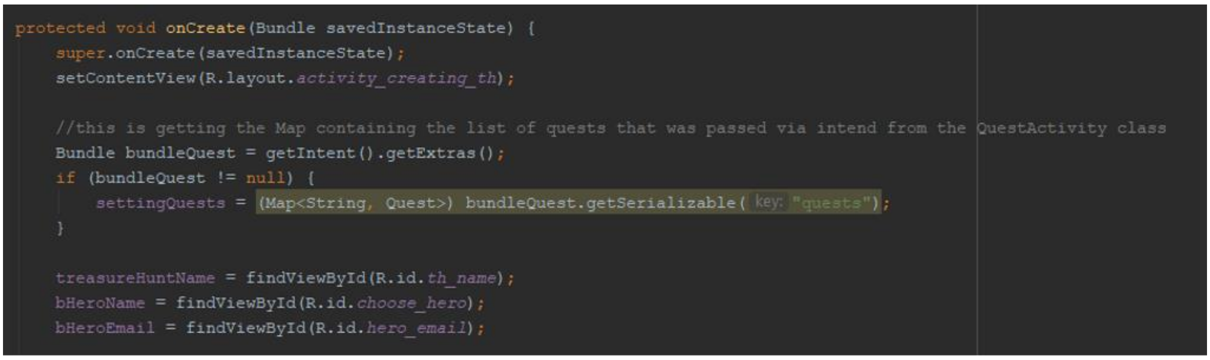
Activities are one of the fundamental building blocks of apps on the Android platform. They serve as the entry point for a user's interaction with an app, and are also central to how a user navigates within an app (as with the Back button) or between apps (as with the recent button).


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
on stop()
onDestroy()
Activity is shut down

The Activity instances in an app transition through different states in their lifecycle. The Activity class provides a number of callbacks (Android, Developers - Documentation, n.d.) that allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). The system invokes each of these callbacks as an activity enters a new state.

An example of onCreate() functionality is the creation of the Quest in a Treasure Hunt. The user (Chief) has the possibility to choose between a numbers of Quests already set in the database. When the Chief choose one of this quest the method onCreate is triggered. This method goes through the Quest storage in a hash map. An Arrays store items as an ordered collection, and it is accessible via an index number (int type). However, HashMap store items in "key/value" pairs, and can be accessible by an index of another type (e.g. a String). Each quest has a unique key, and this two values will be sent to the database when the Chief finishes choosing the total of Quests that it will like to add to the event.

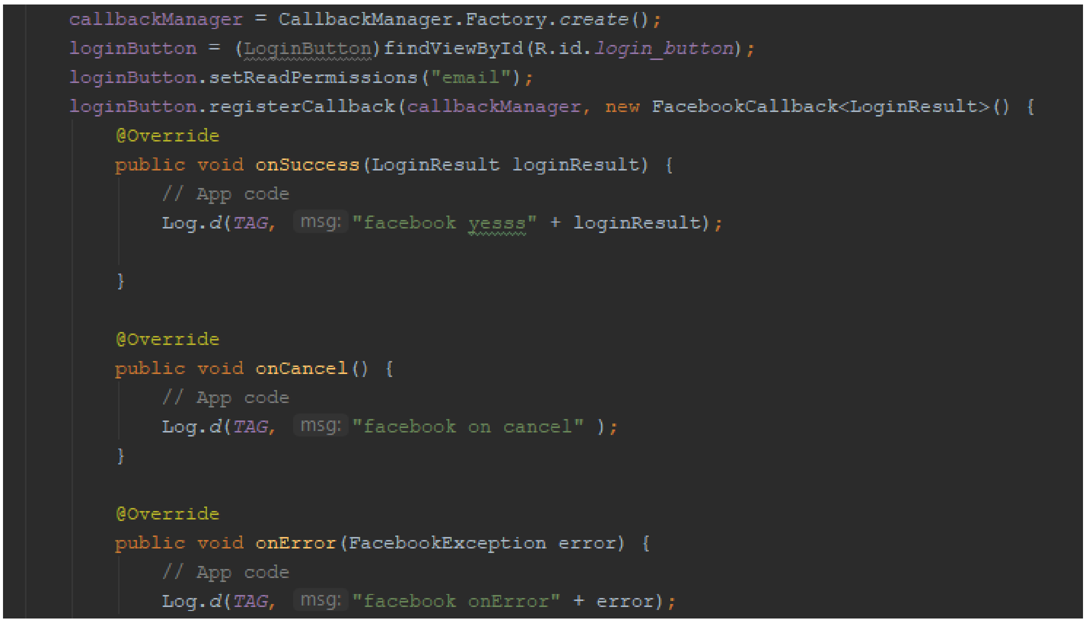


```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_creating_th);

    //this is getting the Map containing the list of quests that was passed via intend from the QuestActivity class
    Bundle bundleQuest = getIntent().getExtras();
    if (bundleQuest != null) {
        settingQuests = (Map<String, Quest>) bundleQuest.getSerializable( key: "quests");
    }

    treasureHuntName = findViewById(R.id.th_name);
    bHeroName = findViewById(R.id.choose_hero);
    bHeroEmail = findViewById(R.id.hero_email);
```

Figure 1.-Example code onCreate(). Activity lifecycle.

Another example of activity lifecycle is the action login with Facebook where a callbackManager is performed. The CallbackManager manages the callbacks into the FacebookSdk from an Activity's or Fragment's onActivityResult() method. CallbackManager use to route calls back to the Facebook SDK and registered the callbacks. When people log into the app with Facebook they can grant permissions to the app so it can retrieve information or perform actions on Facebook on their behalf. If login succeeds, the LoginResult parameter has the new AccessToken, and the most recently granted or declined permissions.

Then in onActivityResult() forward the login results to the callbackManagercreated in onCreate()



```
    callbackManager = CallbackManager.Factory.create();
    loginButton = (LoginButton)findViewById(R.id.login_button);
    loginButton.setReadPermissions("email");
    loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
        @Override
        public void onSuccess(LoginResult loginResult) {
            // App code
            Log.d(TAG,  msg: "facebook yesss" + loginResult);

        }

        @Override
        public void onCancel() {
            // App code
            Log.d(TAG,  msg: "facebook on cancel" );
        }

        @Override
        public void onError(FacebookException error) {
            // App code
            Log.d(TAG,  msg: "facebook onError" + error);
```

Figure 2.- Example code of onSuccess(), onCancel(), onError(). Activity lifecycle

Example code of onSuccess(), onCancel(), onError(). Activity lifecycle

Project Structure (2)

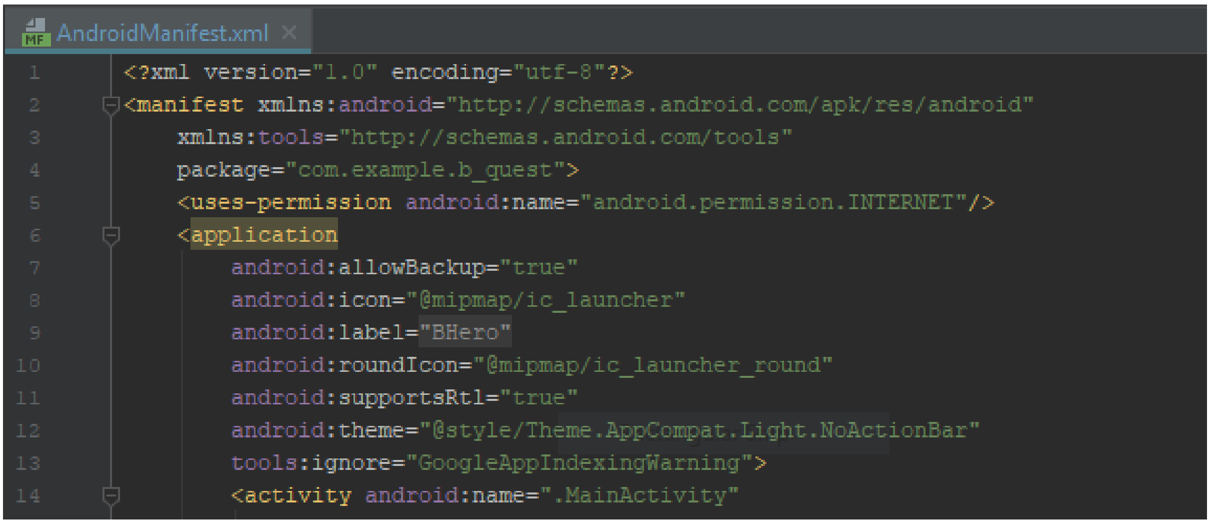
Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- ✓ Android app modules
- ✓ Library modules
- ✓ Google App Engine modules

By default, Android Studio displays your project files in the Android project view. This view is organized by modules to provide quick access the project's key source files.

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

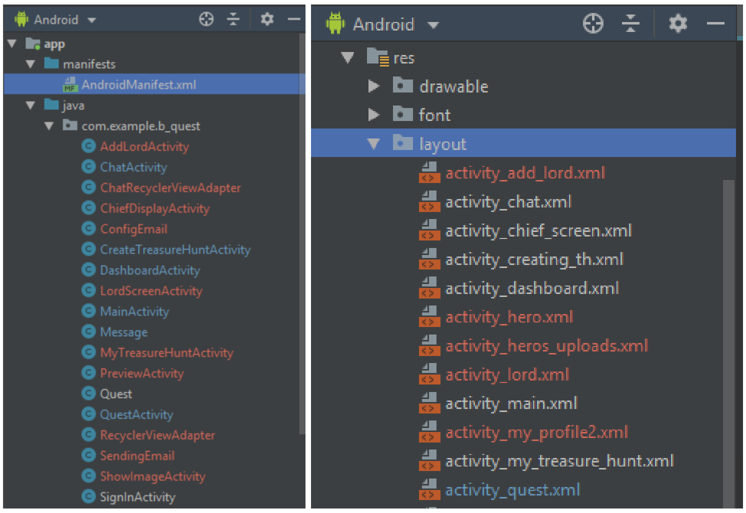
- Manifests: Contains the AndroidManifest.xml file. The manifest file describes essential information about the app to the Android build tools, the Android operating system, and Google Play. The most important characteristics of the app are reflected in the manifest file. Like Package name and application ID, App components, Permissions and Device compatibility



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.b_quest">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="BHero"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar"
        tools:ignore="GoogleAppIndexingWarning">
        <activity android:name=".MainActivity"
```

Figure 11.- Manifest document. B-Quest project.

- **Java**: Contains the Java source code files, including JUnit test code. Contains the Java source code files, separated by package names, including JUnit test code.
- **Res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images, divided into corresponding sub-directories.



```
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▼ com.example.b_quest
        AddLordActivity
        ChatActivity
        ChatRecyclerViewAdapter
        ChiefDisplayActivity
        ConfigEmail
        CreateTreasureHuntActivity
        DashboardActivity
        LordScreenActivity
        MainActivity
        Message
        MyTreasureHuntActivity
        PreviewActivity
        Quest
        QuestActivity
        RecyclerViewAdapter
        SendingEmail
        ShowImageActivity
        SignInActivity
```

```
▼ res
  ▶ drawable
  ▶ font
  ▼ layout
      activity_add_lord.xml
      activity_chat.xml
      activity_chief_screen.xml
      activity_creating_th.xml
      activity_dashboard.xml
      activity_hero.xml
      activity_heros_uploads.xml
      activity_lord.xml
      activity_main.xml
      activity_my_profile2.xml
      activity_my_treasure_hunt.xml
      activity_quest.xml
```

Figure 12.- Java and res document. B-Quest project.

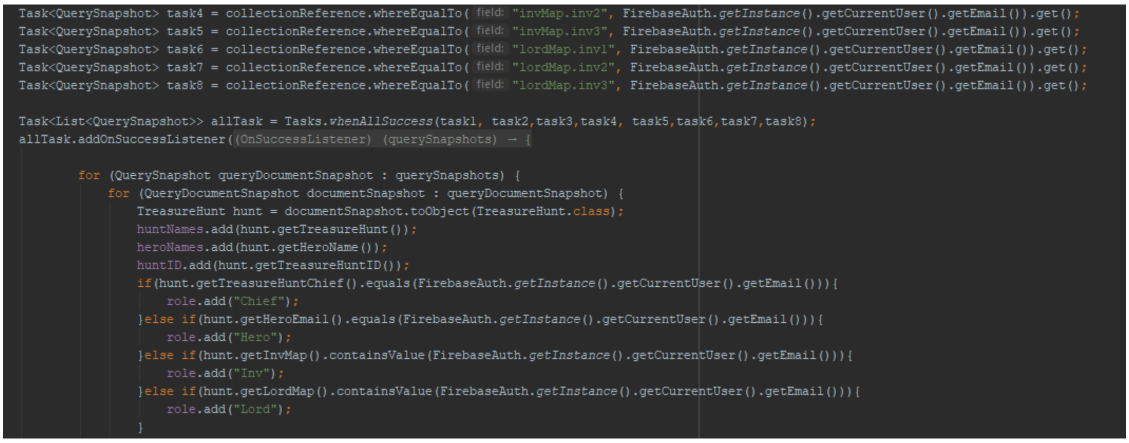
FireBase

To explain how the data stored in Cloud Firestore is retrieved, is important to keep on mind the structure of the data in the Cloud. For B-Quest documents and multiples collections to organize the distribution of the data was the final structure used. There are two ways to retrieve data stored in Cloud Firestore. Either of these methods can be used with documents or collections of documents:

- Call a method to get the data.
- Set a listener to receive data-change events.

In B-Quest a listener was set. Once a listener is set, Cloud Firestore sends the listener to an initial snapshot of the data, and then another snapshot, each times the document changes. One example of this event is the activity MyTreasureHuntActivity, in this activity all the Treasure Hunt that the user is joined in are displayed. Keep on mind that the user has the label of Chief, B-Hero or Lord depending on the activity that is going to perform in the Treasure Hunt. That is why when the user reviews the activity history, it can see with different colours the title of the Treasure Hunt depending on the role that is performing. Each role (Chief, B-Hero, Lord) will demand from the application a different display of information of the event. For example the Chief will always have the option to modify the event, on the contrary the Lord will only be able to see the evidence of each Quest that the B-Hero has pass (video, photo, etc.) And finally the B-Hero can only have the option to upload the proof for the Quest.

The following image shows a complex query that shows the whole process. Depending of the role every user will see the description of the Treasure Hunt with different colours and limited information.



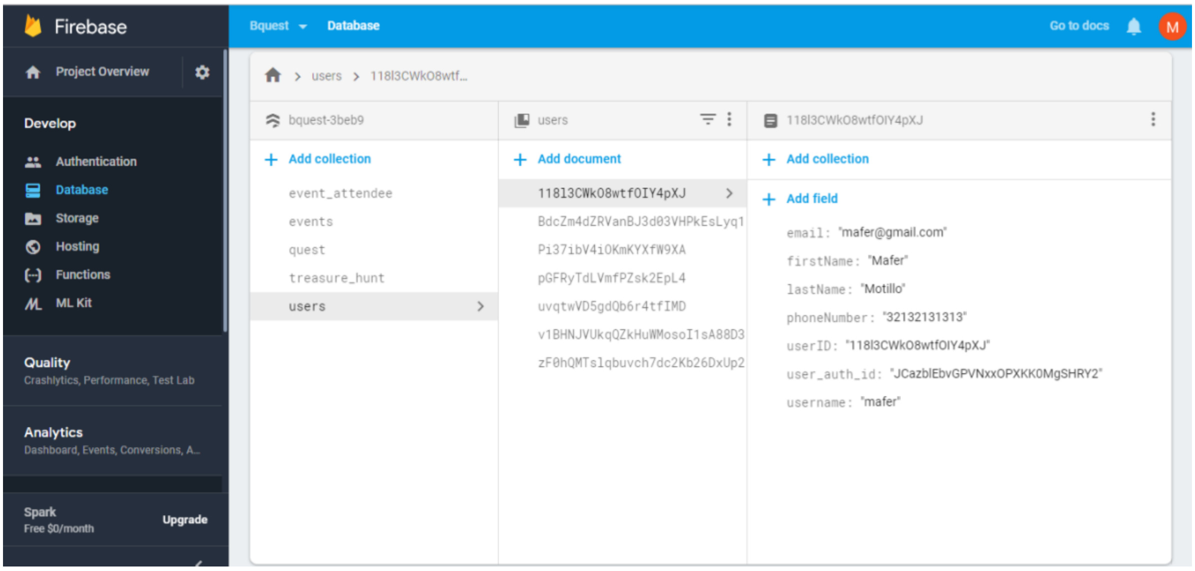
```
Task<QuerySnapshot> task4 = collectionReference.whereEqualTo( field: "invMap.inv2", FirebaseAuth.getInstance().getCurrentUser().getEmail()).get();
Task<QuerySnapshot> task5 = collectionReference.whereEqualTo( field: "invMap.inv3", FirebaseAuth.getInstance().getCurrentUser().getEmail()).get();
Task<QuerySnapshot> task6 = collectionReference.whereEqualTo( field: "lordMap.inv1", FirebaseAuth.getInstance().getCurrentUser().getEmail()).get();
Task<QuerySnapshot> task7 = collectionReference.whereEqualTo( field: "lordMap.inv2", FirebaseAuth.getInstance().getCurrentUser().getEmail()).get();
Task<QuerySnapshot> task8 = collectionReference.whereEqualTo( field: "lordMap.inv3", FirebaseAuth.getInstance().getCurrentUser().getEmail()).get();

Task<List<QuerySnapshot>> allTask = Tasks.whenAllSuccess(task1, task2,task3,task4, task5,task6,task7,task8);
allTask.addOnSuccessListener((OnSuccessListener) (querySnapshots) → {

        for (QuerySnapshot queryDocumentSnapshot : querySnapshots) {
            for (QueryDocumentSnapshot documentSnapshot : queryDocumentSnapshot) {
                TreasureHunt hunt = documentSnapshot.toObject(TreasureHunt.class);
                huntNames.add(hunt.getTreasureHunt());
                heroNames.add(hunt.getHeroName());
                huntID.add(hunt.getTreasureHuntID());
                if(hunt.getTreasureHuntChief().equals(FirebaseAuth.getInstance().getCurrentUser().getEmail())){
                    role.add("Chief");
                }else if(hunt.getHeroEmail().equals(FirebaseAuth.getInstance().getCurrentUser().getEmail())){
                    role.add("Hero");
                }else if(hunt.getInvMap().containsValue(FirebaseAuth.getInstance().getCurrentUser().getEmail())){
                    role.add("Inv");
                }else if(hunt.getLordMap().containsValue(FirebaseAuth.getInstance().getCurrentUser().getEmail())){
                    role.add("Lord");
                }
```

Figure 13.- Complex query. MyTreasureHuntActivity.

Cloud Firestore provides powerful query functionality for specifying which documents are to be retrieved from a collection.

The **where()** method (like in an SQL database) filter the information that is requested from the database, it takes three parameters: a field to filter on, a comparison operation, and a value. In the case of B-Quest there is a comparison between the roles of each user, and with this label, a filter of the information to be displayed is possible per treasure hunt.

This information is taken from the documents and collections from the firebase database. On the following image a display of the distribution of the data is seen. There are 5 collections, each of them storage different type of information, for example the users collection is made up of documents which storage the user information as email, firstname, lastname, phonenumber, userID, user_auth_id and username.



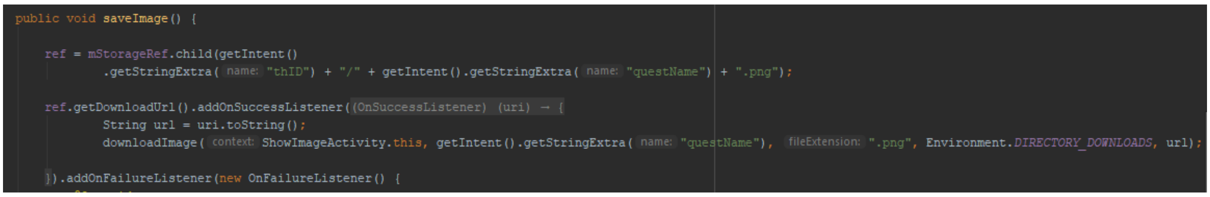
```
event_attendee
events
quest
treasure_hunt
users
```

```
118l3CWkO8wtfOIY4pXJ
BdcZm4dZRVanBJ3d03VHPkEsLyq1
Pi37ibV4iOKmKYXfW9XA
pGFRyTdLVmfPZsk2EpL4
uvqtwVD5gdQb6r4tfIMD
v1BHNJVUkqQZkHuWMosoI1sA88D3
zF0hQMTslqbuvch7dc2Kb26DxUp2
```

```
email: "mafer@gmail.com"
firstName: "Mafer"
lastName: "Motillo"
phoneNumber: "32132131313"
userID: "118l3CWkO8wtfOIY4pXJ"
user_auth_id: "JCazblEbvGPVNxxOPXKK0MgSHRY2"
username: "mafer"
```

Figure 14.- Current status of the database B-Quest project. Firebase.

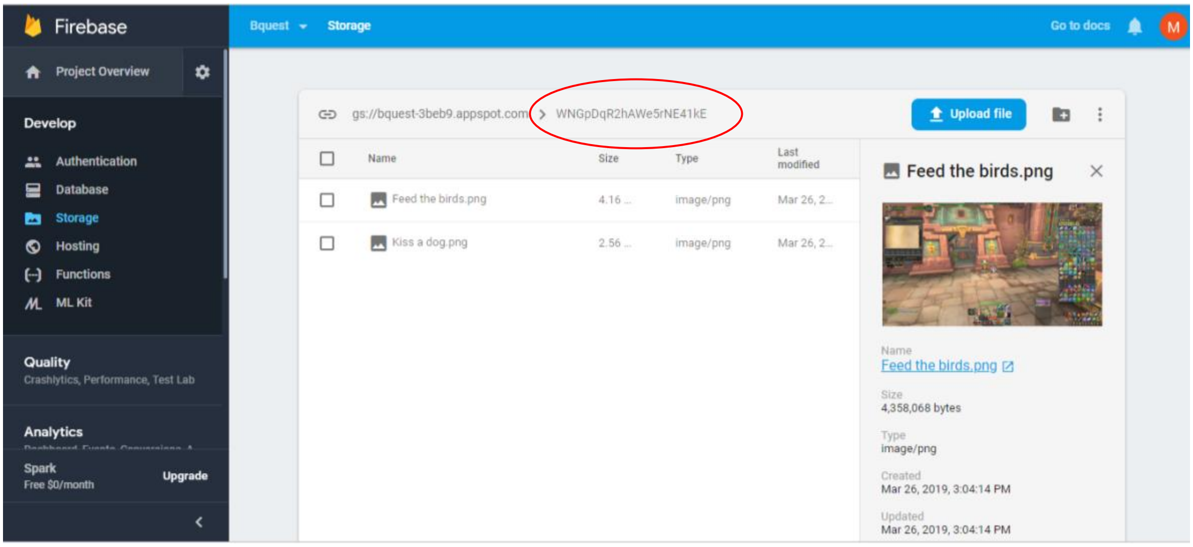
On the other hand, it is important to mention that information is not only stored in a String type, as there is also the storage of photos or videos sent as an evidence of a completed Quest. In this case the information is safe in the storage. Every time a user uploads a photo or video it will create a reference. These references can then be used to download, get or update metadata or delete the file. Here an example of the code:



```
public void saveImage() {

    ref = mStorageRef.child(getIntent()
            .getStringExtra( name: "thID") + "/" + getIntent().getStringExtra( name: "questName") + ".png");

    ref.getDownloadUrl().addOnSuccessListener((OnSuccessListener) (uri) → {
            String url = uri.toString();
            downloadImage( context: ShowImageActivity.this, getIntent().getStringExtra( name: "questName"),  fileExtension: ".png", Environment.DIRECTORY_DOWNLOADS, url);

    }).addOnFailureListener(new OnFailureListener() {
```

Figure 15.- Example code save image/evidence of the quest.

The following image demonstrate how the media information, is stored in Firebase. In this case, images are stored in its original format and the name of the folder where the image is stored is the document key of the Theasure Hunt.



gs://bquest-3beb9.appspot.com > WNGpDqR2hAWe5rNE41kE

□	Name	Size	Type	Last modified
□	Feed the birds.png	4.16 ...	image/png	Mar 26, 2...
□	Kiss a dog.png	2.56 ...	image/png	Mar 26, 2...

Feed the birds.png

Name
Feed the birds.png

Size
4,358,068 bytes

Type
image/png

Created
Mar 26, 2019, 3:04:14 PM

Updated
Mar 26, 2019, 3:04:14 PM

Figure 16.- Current status of the storage. Firebase

PROBLEMS ENCOUNTERED MOBILE APP

During the process of implementation a more realistic comprehension of the possibility of implementing the initial design was evaluated, mainly because even when the initial idea is great, to implement it is not as functional or possible as expected.

B-Quest is no exception. When diving into the codding for the mobile application, a few aspects from the design did not go according to plan, however, after some extra analysis and few stressful coding sessions the problems were solved.

For example, in the original design it was established that the user participating as a **CHIEF** would be able to select the Quests for the Treasure Hunt by moving between two activities, in the first activity the CHIEF was shown all categories of the Quests and on the second activity would see all Quests related to the particular category that was chosen, once a Quest was selected the CHIEF would have been sent back to the category activity and so on and so forth until all Quests were selected.

That was the first problem. When the user shifted between activities, for one time it worked fine but at the time of picking the next Quest, the second one was replacing the first Quest rather than adding it to the quest list.

When this was happening, data was been transferred by using an INTENT. Intents are used to go from one activity to another and so data can be passed between activities. However, the data that is transferred with an intent can be used in the destination activity but cannot be stored for later usage.

Few solutions came into play for this, the first thought was to keep the data bouncing from activity to activity using intents to move the data forward and backwards until all Quest were selected, this idea was discarded as it did not sound very promising for the application purposes and bouncing data could be victim of data corruptions or other problems.

The second idea, was to use an adapter. An adapter class acts as a connection between what is to be displayed and how is going to be displayed, this would have been between the data coming form the database and the data displayed that the user would had access to. So using an adapter, all categories and all Quests related to them could be displayed in one layout, getting rid of the shifting between activities but adapters are recommended for data that will be updated constantly and have to be updated in “real time” in the user screen. Since the number of Quests is limited and will only be updated when adding new Quests, an adapter might not be ideal for the situation.

Finally, the solution came by combining two activities into one and attaching two layouts to it, by doing this the results of an adapter were but the implementation would be done in a simpler way. In this case the user will remain in the same activity at the time of selecting the Quests for the Treasure Hunt and will be presented with two layouts. One layout will be hidden while the user interact with the other, so the user will have the feeling of being moved from one screen to another but in reality will remain in the same, this gets rid of the need of transferring data from an activity to another and back again as everything is done in one place.

The second problem encountered was related to the database. At first the database was design with a set of collections, documents and sub-collections. The problem came at the time of querying sub collections, Firestore does not allow to query a particular document inside a sub collection but the whole document will be retrieved instead, so even if a particular document or even a particular field inside a document was requested the retrieval would have been of the whole document containing that sub-collection. This was solved by changing the design of the database. Now it will use root collections only, this means that there is no nested information, therefore any piece of data can be accessed, modified or displayed at any given time.

The last relevant problem encountered is related to both the database and the code. At first Array Lists were used to store the Quests that were chosen for a Treasure Hunt, but when requesting access to a particular Quest to change its status to completed, Firestore didn't allow to do so, it will only allow to add elements to the array or replace the full array.

This was solved by changing the arrays and using hash maps instead. Hash map makes accessible any Quest, and since each Quest is an object, it also makes accessible any particular attribute in a Quest, making possible to manipulate the state of a quest, to set its status as completed whenever it needs to be done.

TECHNOLOGIES INVOLVE WEB APPLICATION

Web-App

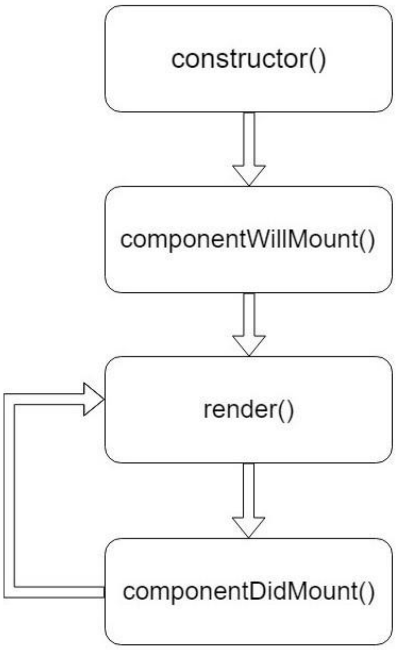
While building B-Quest, it is important to find ways to always keep updated its components, e.g. adding items to ListView dynamically. In order to do that in the right way, is necessary to understand the lifecycle API of React Native.

What is a lifecycle API?

It is a group of functions provided by React Native to instantiate, mount, render, and eventually update, unmount, and destroy components. This API helps to initialise, update data in the right way.

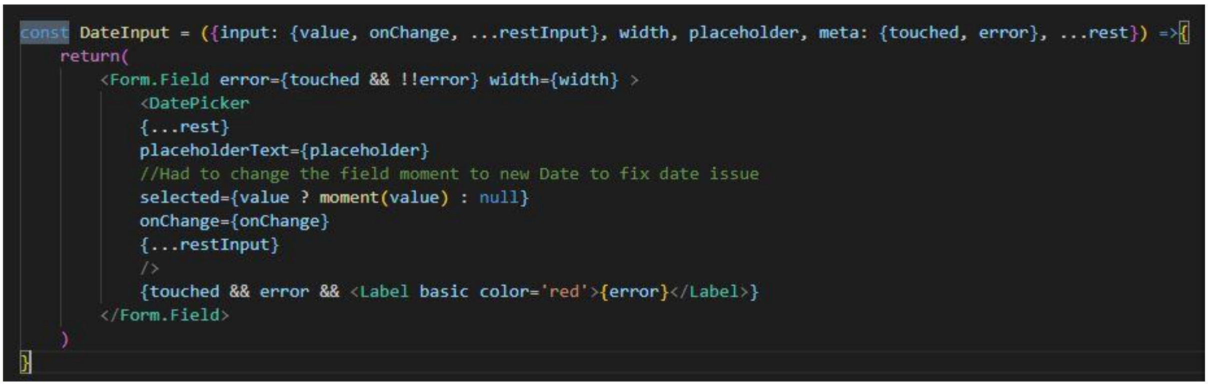
The Component Lifecycle API

Let's look at each phase of the component lifecycle. In React Native the API contains 4 major phases: mounting phase, updating phase and unmounting phase and error handle, so the order should be:



Mounting methods

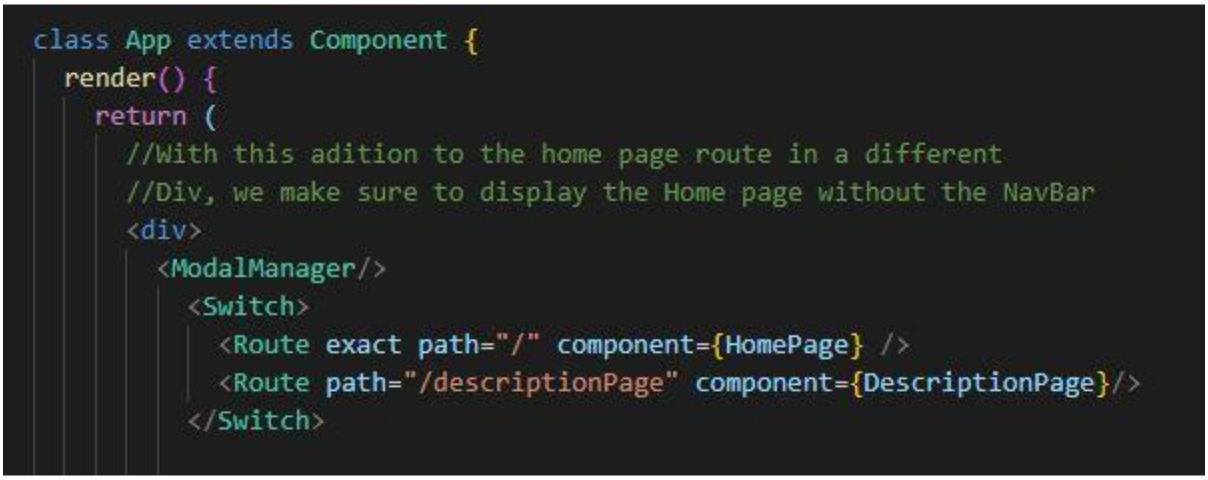
constructor(props): this method is used to initialise components with initial state, no native UI element has been rendered yet. Constructor() is always called at the first time when react native application will start in mobile; it is mostly used to create States in react native application class.



```
const DateInput = ({input: {value, onChange, ...restInput}, width, placeholder, meta: {touched, error}, ...rest}) =>{
    return(
        <Form.Field error={touched && !!error} width={width} >
            <DatePicker
            {...rest}
            placeholderText={placeholder}
            //Had to change the field moment to new Date to fix date issue
            selected={value ? moment(value) : null}
            onChange={onChange}
            {...restInput}
            />
            {touched && error && <Label basic color='red'>{error}</Label>}
        </Form.Field>
    )
}
```

Figure 17.- Example of contructor(props) in B-Hero

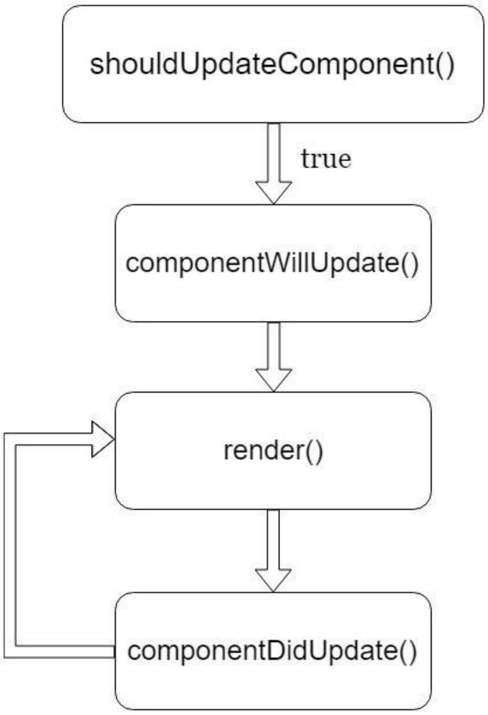
render(): the render method must return a React Native component (JSX element) to render (or null, to render nothing). Render function is one of the most important function of a class because it is used to render View's or any graphical representation on screen using its return block. Without the use of render function in a class it is more difficult to make stable Views or Screens in react native applications.



```
class App extends Component {
  render() {
    return (
      //With this adition to the home page route in a different
      //Div, we make sure to display the Home page without the NavBar
      <div>
        <ModalManager/>
          <Switch>
            <Route exact path="/" component={HomePage} />
            <Route path="/descriptionPage" component={DescriptionPage}/>
          </Switch>
```

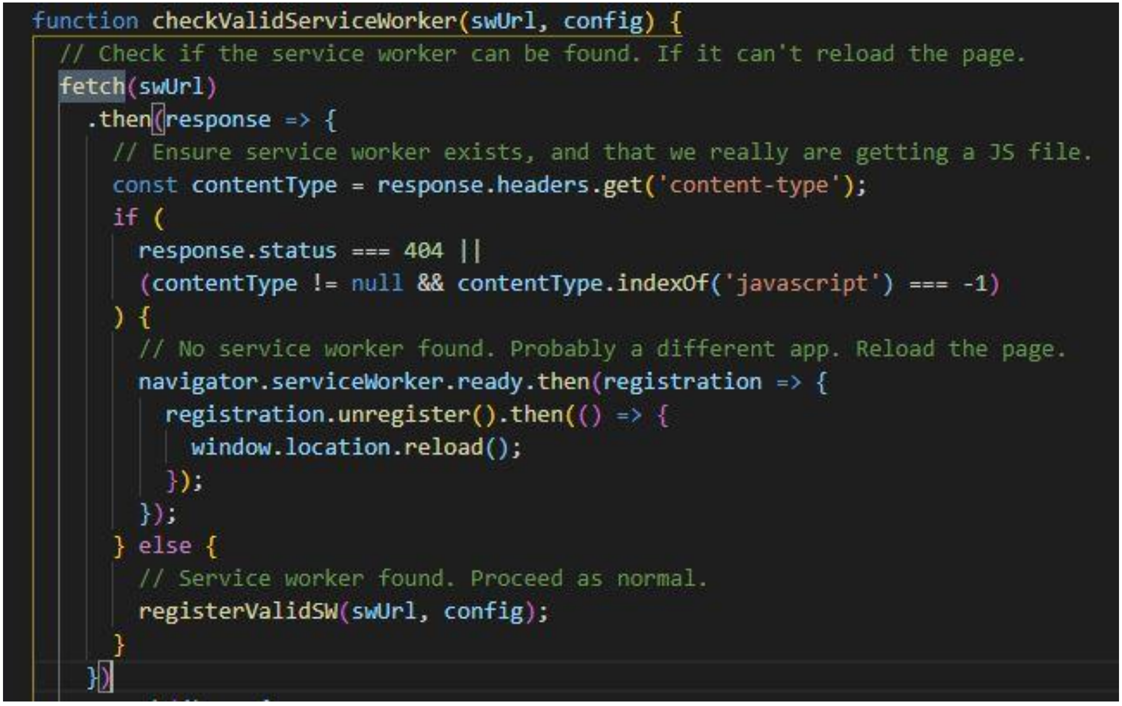
Figure 10.- Example of render() in B-Hero

After this step it will enter to runtime loop, where the B-Quest app will be in updating cycle, so its component will wait for updates occurred either in props or state.



Unmounting method:

componentWillMount(): this method is invoked only once before rendering the component for the first time. Used generally to fetch data from external API, e.g.: This function is just called right after constructor() called, it is mostly used to call asynchronous functions or web calls from react native apps:

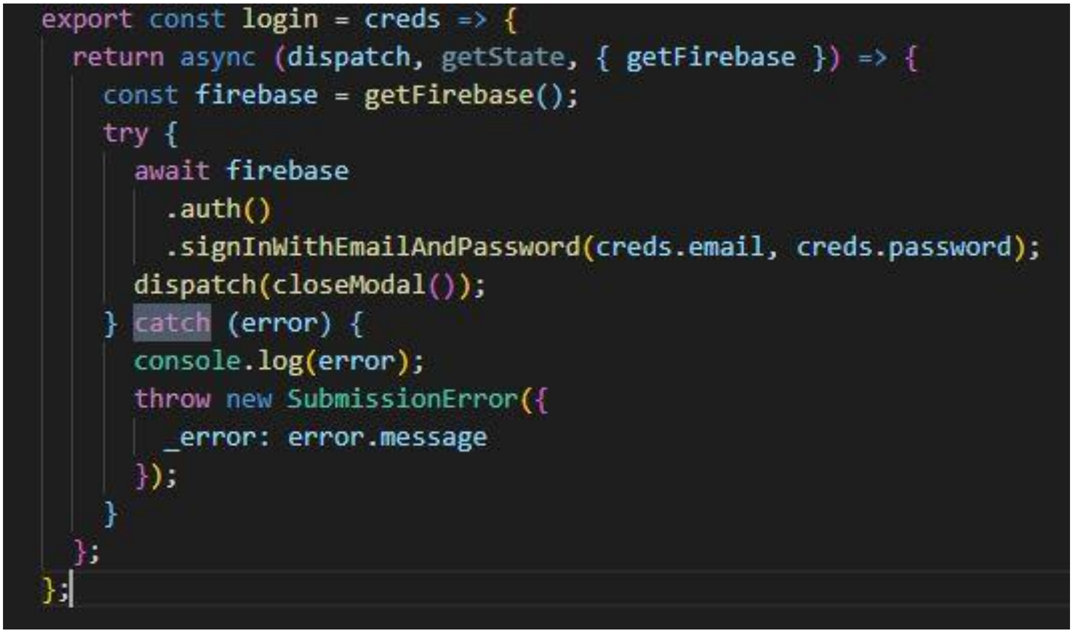


```
function checkValidServiceWorker(swUrl, config) {
  // Check if the service worker can be found. If it can't reload the page.
  fetch(swUrl)
    .then(response => {
      // Ensure service worker exists, and that we really are getting a JS file.
      const contentType = response.headers.get('content-type');
      if (
        response.status === 404 ||
        (contentType != null && contentType.indexOf('javascript') === -1)
      ) {
        // No service worker found. Probably a different app. Reload the page.
        navigator.serviceWorker.ready.then(registration => {
          registration.unregister().then(() => {
            window.location.reload();
          });
        });
      } else {
        // Service worker found. Proceed as normal.
        registerValidSW(swUrl, config);
      }
    })
```

Figure 11.- Example of componentWillMount() in B-Hero

Error handling method:

The **componentDidCatch()** method is a part of error handling method. It is used to find error between JavaScript code and handle them by passing correct message or argument with them. It will help to use any cache or try method to handle any error.



```
export const login = creds => {
  return async (dispatch, getState, { getFirebase }) => {
    const firebase = getFirebase();
    try {
      await firebase
        .auth()
        .signInWithEmailAndPassword(creds.email, creds.password);
      dispatch(closeModal());
    } catch (error) {
      console.log(error);
      throw new SubmissionError({
        _error: error.message
      });
    }
  };
};
```

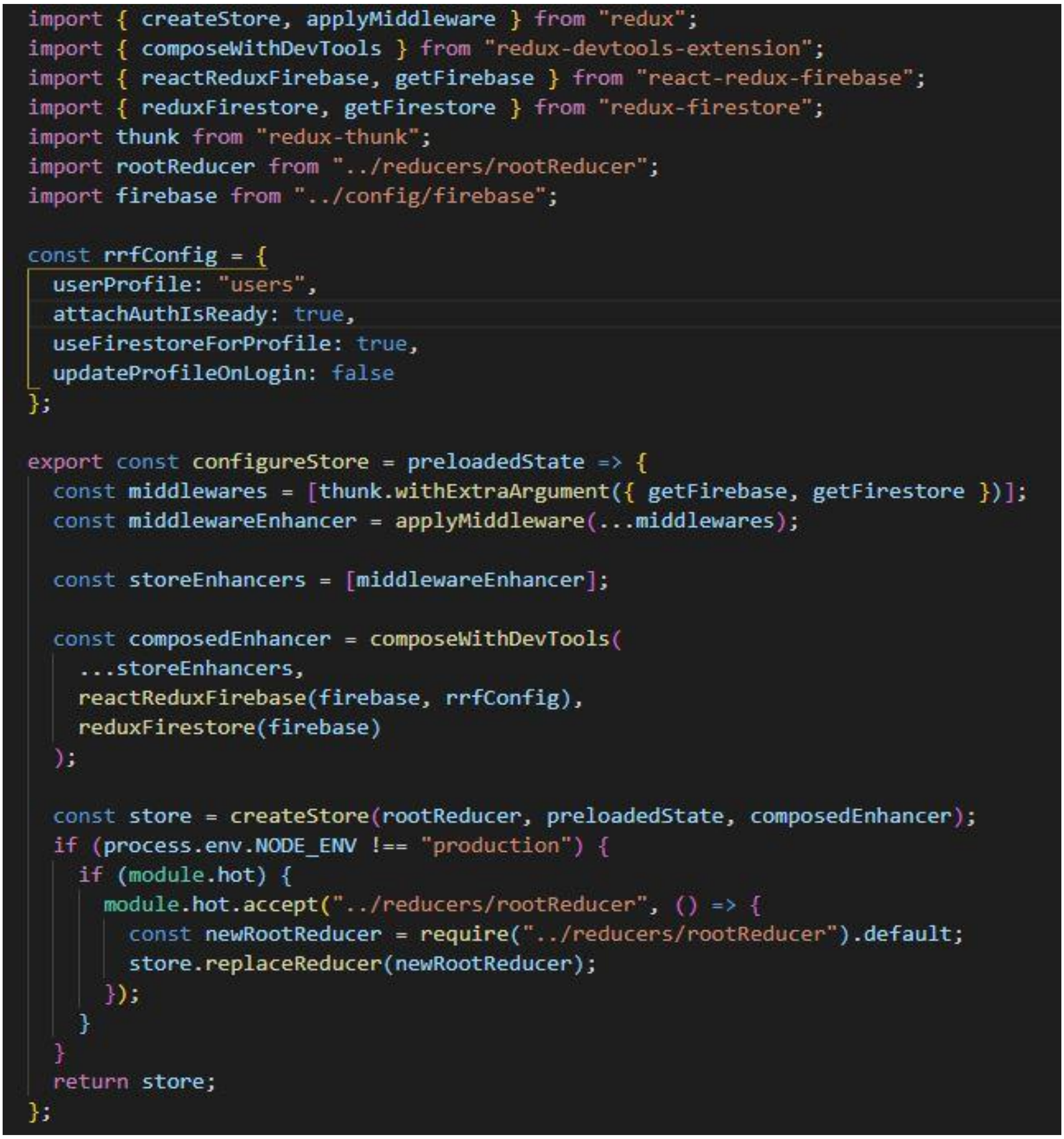
Figure 12.- Example of componentDidCatch() in B-Hero

Redux

Redux is a state management library built for maintaining the data of JavaScript applications.

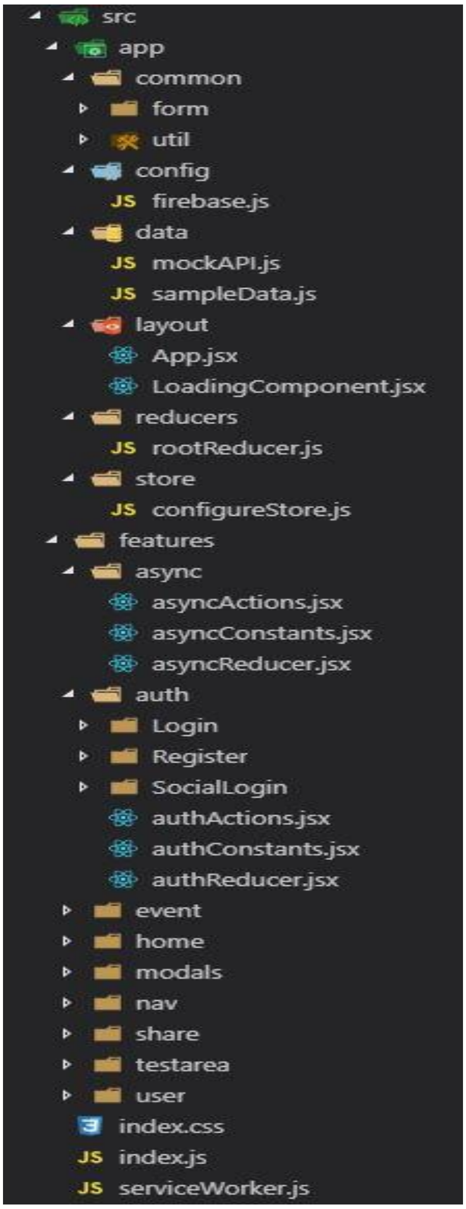
The main thing to know about Redux is that it will keep all the data of the B-Quest application in a single place named **configureStore.js**. When any component of the Web app requires some data, Redux will simply take it from the store and give it to the component.

To begin, it was needed to install the redux library into B-Quest app. Along with this, it was installed the react-redux library to help connect React with Redux, and the redux-thunk library to act as a middleware and handle asynchronous actions in Redux. This is important since the B-Quest app is using Firebase as the database, so chances are that it will need to perform some asynchronous request to fetch the data from there.



```
import { createStore, applyMiddleware } from "redux";
import { composeWithDevTools } from "redux-devtools-extension";
import { reactReduxFirebase, getFirebase } from "react-redux-firebase";
import { reduxFirestore, getFirestore } from "redux-firestore";
import thunk from "redux-thunk";
import rootReducer from "../reducers/rootReducer";
import firebase from "../config/firebase";

const rrfConfig = {
  userProfile: "users",
  attachAuthIsReady: true,
  useFirestoreForProfile: true,
  updateProfileOnLogin: false
};

export const configureStore = preloadedState => {
  const middlewares = [thunk.withExtraArgument({ getFirebase, getFirestore })];
  const middlewareEnhancer = applyMiddleware(...middlewares);

  const storeEnhancers = [middlewareEnhancer];

  const composedEnhancer = composeWithDevTools(
    ...storeEnhancers,
    reactReduxFirebase(firebase, rrfConfig),
    reduxFirestore(firebase)
  );

  const store = createStore(rootReducer, preloadedState, composedEnhancer);
  if (process.env.NODE_ENV !== "production") {
    if (module.hot) {
      module.hot.accept("../reducers/rootReducer", () => {
        const newRootReducer = require("../reducers/rootReducer").default;
        store.replaceReducer(newRootReducer);
      });
    }
  }
  return store;
};
```

Figure 13.- Example from code configureStore.js



```
src
  app
    common
      form
      util
    config
      firebase.js
    data
      mockAPI.js
      sampleData.js
    layout
      App.jsx
      LoadingComponent.jsx
    reducers
      rootReducer.js
    store
      configureStore.js
  features
    async
      asyncActions.jsx
      asyncConstants.jsx
      asyncReducer.jsx
    auth
      Login
      Register
      SocialLogin
      authActions.jsx
      authConstants.jsx
      authReducer.jsx
    event
    home
    modals
    nav
    share
    testarea
    user
  index.css
  index.js
  serviceWorker.js
```

Project Structure

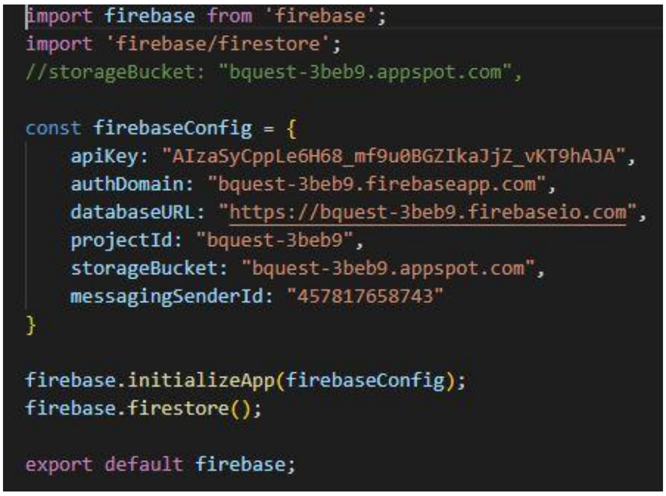
Create-react-app is a flexible tool that allows developers to structure project codes anyway their like as long as the index.js file is available in the root of the /src directory. Small projects do not need a sophisticated structure, and it contains pretty much all the components in the /src directory. However, this will not scale for larger projects. It is imperative that the code structure of B-Quest stays maintainable as it scales.

Each route in the B-Hero web application will have one directory, and each directory will have Redux code.

Additionally, a common directory will contain components that are shared throughout the B-Hero app, such as buttons, modals, navbars.

Firebase is a Backend as a Service (*BaaS*) that provides an advantage to mobile developers who use React Native for developing mobile applications. As a React Native developer, by using Firebase, it is possible to start building an MVP (minimum viable product), keeping the costs low and prototyping the application fast. Firebase is a platform that got acquired by Google and has a healthy and active community. Most users in this community are web and mobile developers as Firebase can help with mobile analytics, push notification, crash reporting and out of the box provides email as well as social authentication.

The image below shows how the B-Quest app is connected to the Database in Firebase.



```
import firebase from 'firebase';
import 'firebase/firestore';
//storageBucket: "bquest-3beb9.appspot.com",

const firebaseConfig = {
    apiKey: "AIzaSyCppLe6H68_mf9u0BGZIkaJjZ_vKT9hAJA",
    authDomain: "bquest-3beb9.firebaseapp.com",
    databaseURL: "https://bquest-3beb9.firebaseio.com",
    projectId: "bquest-3beb9",
    storageBucket: "bquest-3beb9.appspot.com",
    messagingSenderId: "457817658743"
}

firebase.initializeApp(firebaseConfig);
firebase.firestore();

export default firebase;
```

Figure 18 Example from code in the WebApp

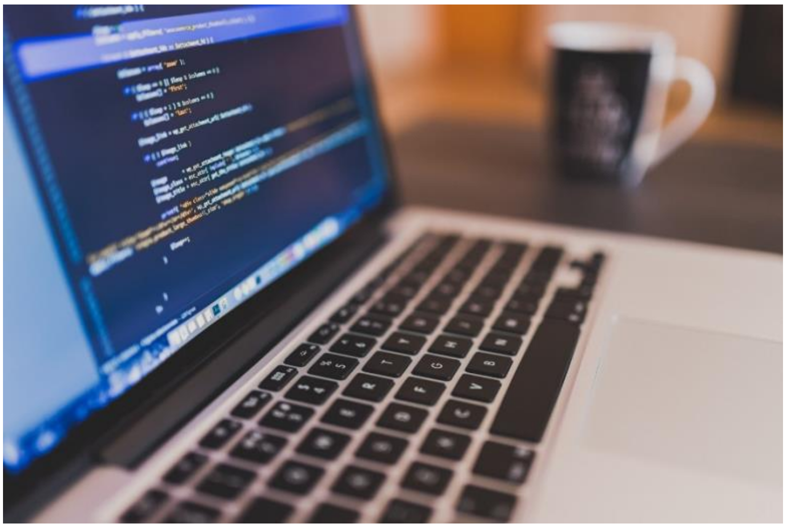
PROBLEMS ENCOUNTERED WEB APPLICATION

Learning curve - Even for experienced programmers who are good at programming using Javascript, it would require understanding the Redux/Flux thing in detail. The team had to spent about 25 hours completing an Udemy course to be able to complete the B-Quest Web Application.

Thinking in React - It is different to work with JS framework than just using jQuery, so that thinking needs to be learned and it takes time. (Like, we have MVC in Rails, MTV in DJango, the team must understand and think the ReactJS way just for UI/UX flow).

In general, several issues were found about the specifics ways to write code in React.js. Most of them were common problems that a beginner React.js learner will make. Those affected considerable when the deadline was coming, it delayed significantly the whole process. In some cases, the issues were cryptic, not finding a proper way to solve them in several hours.

Different Screen Size was a Serious Problem - While designing the B-Hero app, it was not possible to design for the latest platform because mobile devices have limitations based on screen sizes, embedded technologies, pixel intensities, OS requirements, and so on. And then again, it is not just about the devices alone, as a team, we realized that Android, Windows and iOS have their own UI objects as well that will affect the whole project at the end.



Software Fragmentation - As there are many Android versions, it is not easy to make apps run the same way in different devices, even if it is the same OS. The problem is not because as developers we are not able to develop software that runs on various devices, but because of the various other software that can be found on them. This software read the applications differently causing glitches which further leads to major problems when customers use them. Testing all the possible combinations was one of the main things in order to ensure that B-Quest was going to work properly.

Losing information - ReactNative application will continue working if the device loses connectivity momentarily. But once the B-Quest application or tab is closed, the data will be gone. It was not possible to implement a cache with persistence, this still in developing. It could be a serious issue, specially on mobile.

CHAPTER V

TESTING AND EVALUATION

TEST PLAN

The Test Plan is designed to prescribe the scope, approach, resources, and schedule of all testing activities of the project B-Hero app.

The plan identifies the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.

TEST STRATEGY

Scope of Testing

Features to be tested
All the features of B-Hero app which were defined in software requirement specs are need to bested.

Module Name	Applicable Roles	Description
Create an Account with a valid Email	New Users	New users: a new user can create an account filling the details asked
Create an Account with Social Media account	New Users	New users: a new user can create an account using his social medial account.
Login in Account	All users	Users: a user will be able to login into its account.
Create Treasure Hunt	Chief	Chief: a chief can create a treasure hunt selecting the quests that need to be completed
Join Treasure Hunt	Lords, B-Hero	Lords: a user can join a treasure hunt as a Lord after it was created using the invitation that he/she received. B-Hero: A user can join a Treasure Hunt as a B-Hero using the invitation that he/she received.

Vote for Quest	Chief	Chiefs: a chief can vote in a quest after the B-Hero completed and submitted it. The chief could approve or reject it.
Submit a Quest	B-Hero	B-Hero: a B-Hero can submit a quest (photo) to be approved.to be approved.
Complete a Treasure Hunt	B-Hero	B-Hero: a B-Hero can complete a Treasure Hunt after completing all the quests.
Invite a Lord	Chief, Lord	Chief: a Chief can invite a person to join a Treasure Hunt as a Lord.
Invite a B-Hero	Chief, B-Hero	Chief: a Chief can invite a person to join a Treasure Hunt as a B-Hero.
Interruption by Calls	All users	User: a User should able to accept Phone calls when application is running and should continue from the same point.
Interruption by Messages	All users	User: a User should able to accept messages when application is running and should continue from the same point after reading the message.
Exit application	All users	User: a User should able to exit from application if we click on end key.
Charge	All users	Application should run when inserting the charger. It will not affect the application.

Features not to be tested

These features are not be tested because they are not included in the software requirement specs:

- Hardware Interfaces.

- Database logical.

- Website Security and Performance.

TEST TYPE

In the project B-Quest, there are 3 types of testing should be conducted.

- **Integration** Testing (Individual software modules are combined and tested as a group).

- **System** Testing: Conducted on a **complete**, **integrated** system to evaluate the system's compliance with its specified requirements.

- **API testing:** Test all the APIs create for the software under tested.

Risks and Mitigations: refers to the issues and plans of attack that the testing process can involve. Some could be:

Risk	Mitigation
Team member lack the required skills for website testing.	Plan **training course** to skill up your members
The project schedule is too tight; it's hard to complete this project on time	Set **Test Priority** for each of the test activity.
A lack of cooperation negatively affects team member's productivity	**Encourage** each team member in his task and **inspire** them to greater efforts.

TEST LOGISTICS

Who will test?

The project should be tested for team members of Mowhile Geek.

When will test occur?

The tester will start the test execution when all the following inputs are ready

- Software is available for testing.
- Test Specification is created.
- Test Environment is built.
- Enough human resource for testing.

TEST OBJECTIVE

The test objectives are to verify the Functionality of Web-app and Android app, the project should focus on testing the critical operations such as Create an account, Login into the app, Manage Account, Create Treasure Hunt, Manage Treasure Hunt, Join Treasure Hunt as a Lord, Join Treasure Hunt as a B-Hero, Etc… to guarantee all these operation can work normally in real business environment.

TEST CRITERIA

Suspension Criteria

If the team members report that there are **40%** of test cases **failed**, suspend testing until the development team fixes all the failed cases.

Exit Criteria

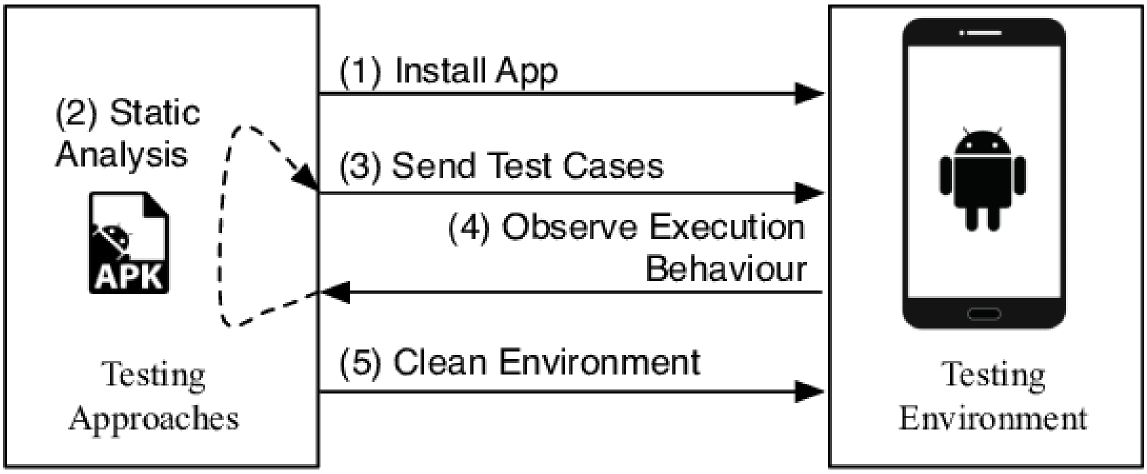
Specifies the criteria that denote a **successful** completion of a test phase

- **Run** rate is mandatory to be **95%** unless a clear reason is given.

- **Pass** rate is **80%,** achieving the pass rate is **mandatory**.

TEST ENVIRONMENT

The Test Environment should be setup as figure below:

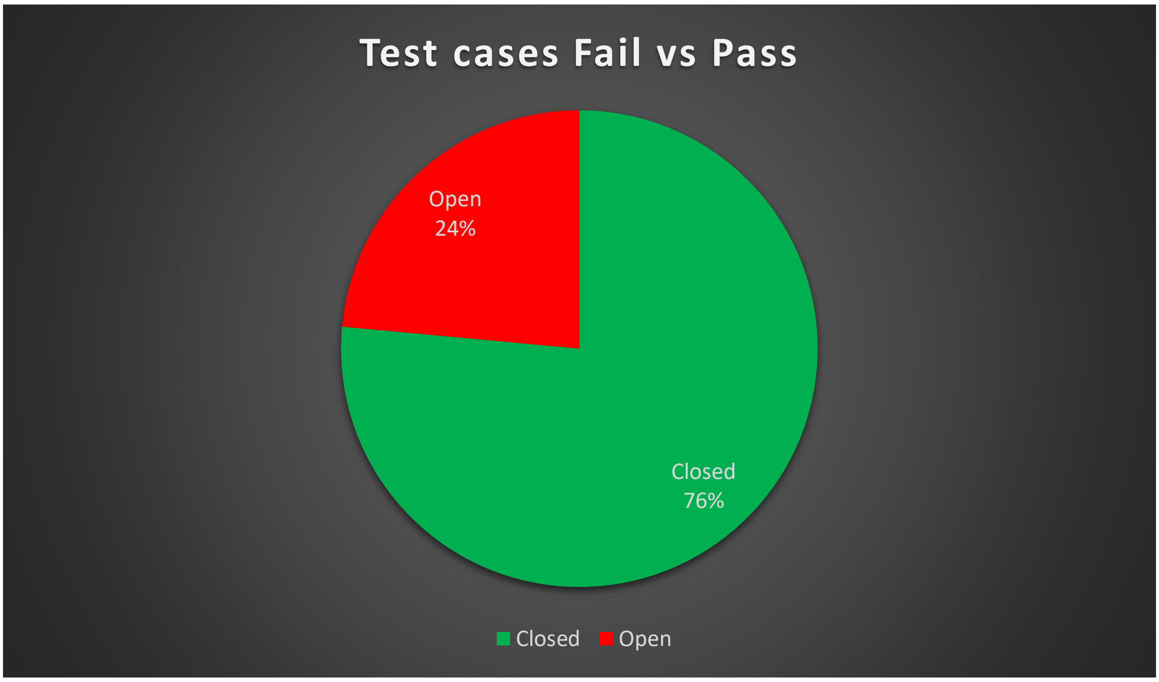


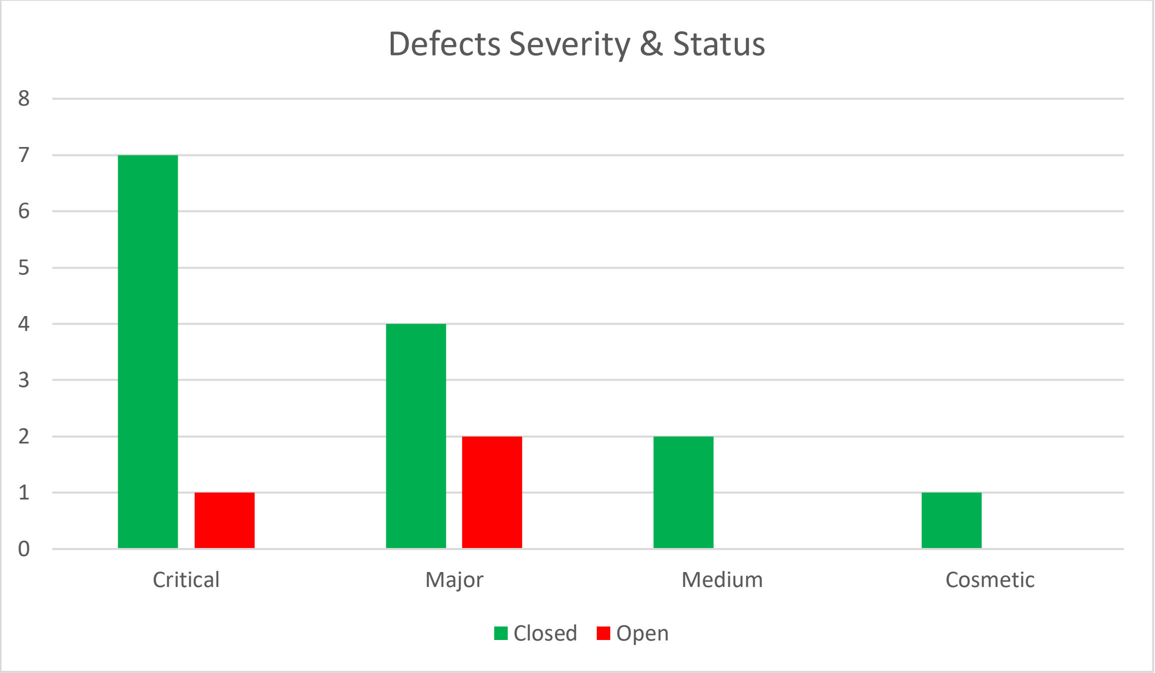
SCHEDULE & ESTIMATION

All project task and estimation

Task	Members	Estimate effort
Create the test specification	Ronald	7 hours man-hour
Perform Test Execution	Ronald, Ailem	5 hours man-hour
Test Report	Ailem, Ronald	5 hours man-hour
Test Delivery	Ailem, Ronald	5 hours man-hour
Total		22 hours man-hour

TEST RESULTS





Performing exhaustive testing, checking the test results, metrics, best practices, lessons learned, conclusions on 'Go Live' etc. are extremely important steps to produce evidence for the Testing performed and the Testing conclusion.

As a conclusion, we noticed that the B-Quest application Mobile app and Web-App are in an early stage and its deploying won't be recommended until these have been tested deeply and with results of success over 95%. Also, a deeply review of the code is needed to fix few Cosmetic bugs that were not comtemplate in the current testing plan.

Please refer to Appendix to see specific details of scenarios tested and results

Please refer to original document of Testing (Excel) to see specific details of scenarios tested and results

COMMERCIALISATION / MARKETING

B-Quest is an application that is in its early stages. During the development of this project an Alpha version of the application was created, therefore the application is still not stable enough for general deployment but for internal use only.

Nevertheless, the next step will be a status 2 for release candidate, during this stage the application will be registered in the Google Apps Playstore by creating a Google Play Developers account.

Similarly, different and very basic strategies will be applied to make the application more popular. So far, some initial ideas are:

- The investment and use of publicity among different Social Media platforms such as Facebook, Instagram…
- Investment in YouTube and Twitch influencers to test and make reviews of the application for future recommendations. Mainly influencer from those two platforms as they have a specific target market which is Gamers. The idea is for them to expose B-Quest when they are streaming live and give positive reviews so other followers are suggested to download. This strategy has been implemented many times before for gaming applications.
- Additionally, investment in traditional media such as Radio, Gamers magazines and gaming sites is important to do. The mouth to mouth advertising is something that is expected by using this type of media and that should never be underrated.

On the other hand, as developers the implementation of Ads inside B-Quest will be done in order to generate profits. Nevertheless, the initial idea is not to be too aggressive with the use of 3rd parties Ads as it normally causes bad ratings in the Playstore.

Finally, the idea of B-Quest can be implemented for multiple occasions in life, that's why this first product will set an example for following versions that could relate to different social events such as bachelor/bachelorette party, sports events, corporative competitions, fundraising or charity, exploring or just for fun events… There are multiple opportunities out there.

CHAPTER VI
CONCLUSIONS

The importance of technology in our daily lives is undeniable. Technology, which refers to the use of tools, that make, use, and exchange of information, to make your life easier and safer. One of the greatest uses is communication, to connect people no matter where they are.

The main objective of this project has been to develop an application and a web application that allows digitally gathering a group of friends on an occasion like a birthday day.

Whenever you are in the situation of an innovative project as this one, it is difficult not to deviate from the objectives. For a project like this, we had to make many cuts in order to produce a final working product with an Alpha level of quality.

However, with a lot of dedication, as a team we tried to hold on and never forget our main object. A functional and intuitive application, a web application dynamic and attractive to the user has been the result of a great effort and work.

When the project was envisioned for the first time a large scale one was the goal. This represented a big obstacle during the development as it was the first time working on a project like this. The learning curve was one, if not, the biggest challenge for this project in many aspects. No only for the technical or hard skill necessaries but also for the soft skills.

Skills such as communication, empathy, consideration, creativity and even fun were necessary to develop this application. Taking into consideration that Mowhile geek is a group of students in their last year of a demanding career such as Information Technology, it was important to never feel the project was impossible, boring or a chaotic idea that could generate stress.

Thankfully, It wasn’t the case of B-Quest, even in the hardest moments skills were developed and a good example of this is the development of features such as the Chat or the Venue Map, that weren’t thought in the initial plan but that were possible to add thanks to the knowledge acquired when learning the new technologies. The team managed to communicate successfully, to help each other and to trust to make this job possible in an enjoyable way.

Then the challenge of learning new technologies was very demanding, nevertheless, as a team the focus on being positive and thinking that each new technology is something that will stay with us for our professional development was extremely helpful. To develop this application and web app we made use of tools such as Android Studio, React and Firebase. One of the biggest challenges has been acquiring the necessary skills and knowledge of these technologies to be able to create the application and the web app.

Thinking about the users, we have studied and designed a friendly interface and the most intuitive for any type of user. However, we believe that many improvements could be applied by doing a deeper study in the development of the next versions, where deadlines and college responsibilities will be less restrictive.

Yet, so far, B-Quest uses top point technologies that provides multiple benefits for the development of future versions, is for that reason that it may not be necessary apply many changes in the following versions regarding this aspect.

Also, we were taking special attention to ensure that the user feels confident in using the application and to provide personal information without any risk. One of the advantages when using Firebase is that this support authentication using a password, or phone number, popular federated identity providers like Google or Facebook. And because most apps and web apps need to know the identity of a user, firebase allows the app to securely save user data in the cloud and provide the same personalized experience across all of the users' devices.

Finally, working in groups can be challenging at times and it demands a set of skills, however with this project, we gained fair knowledge about the task-oriented groups work and what complexity it involves. Some of the most important things we learned when working in a group is to keep the communication flowing, stay organized, finish off tasks before the deadline, and the opportunity to get to know our colleagues. This has been such a rich experience and a remarkable opportunity that will shape and influence our professional life.

INDIVIDUAL CONTRIBUTIONS

Ronald Torres aka “ElPiberats” is the mastermind behind the concept of B-Quest.

Teams don't work well without teamwork! Teamwork is important for the success of all relationships, no matter if it is in business or college. To have a meaningful and lifelong career, I have learnt the need to work well with others which is why teamwork is so important in the professional world and my life. It was a pleasure to work with such of nice people, every single of them gave their best to finish this project on time.

As an individual, my contribution from beginning to end is the following:

- Implementation of the Backend in the WebApp (Developing phase):
  - Setting up the project.
  - Creation of snippets used in the project.
  - Break up the UI into a component Heriarchy.
  - Creation of forms and submittin form data.
  - Implementation of the React Route.
  - Redux store configuration (Creating reducers and connecting the store to our application).
  - Adding Google maps integration into B-Quest app.
  - Deploying of Firebase in B-Quest webapp.
  - Redux Thunk, creating an async reducer.
- Support in the implementation of the FrontEnd in the WepApp (Developing phase).
- Mobile Application video for documentation submission.

Chapter I:

- I participated in the elaborations of the wireframes for the Web-App.
- Oversaw the redaction of Why B-Quest is a good project.
- Research of two similar apps to B-Quest.
- Along with other members of the team had a part on the design of Use -Cases diagrams.
- Testing of the first prototype in chapter I.
- Design of class Diagrams for the mobile app.
- Creation of the glossary for the project.
- Research of the tools for being used in both projects (Web-app/Mobile app).
- Research of courses to be complete for the members of the group to help on the process of development of the project.
- Why B-Quest is a good project and Novel Aspects for Chapter I.

Chapter II

- Functional requirements for the Mobile App in Chapter.

Chapter IV

- Write the documentation for the Implementation of the WebApp.

Chapter V

- Create the test specifications for the testing process.
- Perform Test Execution along with Ailem.
- Write along with Ailem the Test Report.
- Test Delivery in Chapter.

Finally, it is worth to mention the completion of the courses "Android O & Java - The Complete Android Development Bootcamp" (link), "The Complete JavaScript Course 2018: Build Real Projects" (link) and "Build an app with React, Redux and Firestore from scratch" (link) since it is proof of the effort and dedication to the project.

Juan Martin aka "The Pelon"

From start to end this is my participation within our project:

Part I

- Developed the wireframes.
- Direct participation in the design of the mock-ups for the B-QUEST mobile application.
- Participated on research tasks related with the database that the application is going to use, the IDE to develop the application (Android Studio).
- Carried out a research with the purpose of finding applications somehow related with B-QUEST (the specific application found I researched was "Birthday Gift List").
- Had part on the design of Use -Cases diagrams.
- Completion of the course "Android O & Java - The Complete Android Development Bootcamp" (link).

Part II Development (Mobile App).

- Database structure.
- Front end:
  - Design of he GUI and implementation of the following:
    - Home page.
    - Sign in / Sign up page.
    - Dashboard.
    - Create treasure hunt.
    - Choosing the quest for the treasure hunt.
    - "My treausure hunt" where the user will be able to see TH where its participating.
    - Display for Cheifs and Lords.
- Back end:
  - Implementation of a drawable menu.
  - Creating user authentication information (using the Firebase Auth service).
  - Creating user on the app.
  - Storing new users on the database (sign up).
  - User sign in.
  - Retreaving user information from the databse and displaying that on the app.
  - Createing THs on the app and storing them on the database (displaying the quest list, managing the users process of choosing a quest, display the name of the quest and everything related to the creation of a TH).
  - Sending an automatic invitation email to the B-HERO when the event is created (JavaEmailAPI).
  - Retreaving TH from the database (with all its information) and displaying them acording to the role that the user have.
  - Inviting lords to the TH.
  - Chief votes.
  - (Cheif and Lord) show the media that has been uploaded by the B-Hero.
  - Implemetantion of a chat room for each individual treasure hunt.
- Documentation regarding the database, Firebase and Firestore.
- Documentation regarding problems encountered during the development and how were solved.

Robert Flores aka “Dorlan”

As an individual myself, as well as being part of B-Quest in this project, I understood that working together is a part of life. A family is a team, almost every type of work is based on teams, and on this occasion, I had the opportunity to be part of this great team.

My role on this project was to be in charge of the development of the Android Application together with Juan Martin from beginning to end. As a Developer on the Android application I was in charge of the following tasks:

1. **Creating the sketches of the app.**
2. **Turning sketches into Wireframes.**
3. **Turning Wireframes into a Prototype.**

The use of Marvel tool allowed me to create the prototype for the B-Quest app. All of this is shown in chapter one.

4. **Developing the app (Coding – Frontend & Backend).**
    - Profile Activity, Setting Activity, Help Activity, About Us Activity, B-Hero Activity, B-Hero Images Activity, B-Hero Uploads Activity, Link to our YouTube video.
5. **Contribution on the documentation.**
    - Manual for Prototype (App).
    - Writing the User Interface Design (App).
    - Writing the Functional Design (App).

The most challenging aspect for me was having to learn new technologies in such a short period of time. One other aspect I found challenging was having a large group of 6 people, where everyone has different views and ways of facing problems. Learning how to deal with this was a challenge.

Managing and organizing my time in a more efficient way, being in contact with my colleagues more frequently and definitely having a backup plan for every step of the project would be the 3 and most important things that I would do different next time, as during this process I realized that with organization and communication, it can be more effective. I have learned the hard way of not having a backup in any project is a huge mistake.

For me, the most challenging aspect of this project turned out to be the most interesting, because learning new technologies is one of the things I love most. Sharing ideas with my colleagues from the beginning and making them happen was one of the best experiences I ever had on the beginning of my career.

This would not have been possible without the help and collaboration of our project supervisor Ken Healy and my B-Quest team.

Johan Quiroz aka "Sportsman" researched about what apps were similar, we were curious about what competitors we will be facing in the market but as a result we had that some apps were alike B-Quest in some ways. Johan also contributed and helped Robert to implement the digital design of the wireframes with its mock-ups to view how the application will look like when is finished.

As a Front-End developer I've contribuited in the User Interface to provide a high quality system and also a platform easy to understand. It was very challening but at the same time very fun because the Front-End side is what I like the most.

At the beginning my role was only the User Interface of the Web Application but then I acquire some knoledge in the Back-End side so I helped and support Ronald with some connections and features in order to make our Web Application work. It was great because I learned so much from it and despite the plan was different at the beginning, we realise we could help each other as a team. The comunication was really important between us, was esential to connect both Back-End and Front-End sides which defined the final product.

In the Front-End I was involved in the develop and design of several pages such as:

1. Landing Page
2. Description About Us Page
3. Login and Signup Pages
4. The Dashboard Page
5. Treasure Hunt Details Page
6. The User Details Page

In the Back-End I supported Ronald with:

1. Connecting the application with the Database
2. Setting Up the Redux to track the entire state of the Application
3. Creating the Upload Photo feature which required Node Packages
4. Routing the pages to render the correct data

Problems encountered in the Web-App

I also documented some information about the problems encountered in the Web-App and the troubleshooting I had to perform to find a solution for some of these problems.

Also, I did the Web Application video for documentation submission.

Finally, it is worth to mention the completion of the course "Build an app with React, Redux and Firestore from scratch" (link) since it is proof of the effort and dedication to the project.

Maria Torres aka “Mafer”

My contribution as a Product Manager on this project can be resuming as I describe bellow:

- Chapter 1: I was responsible for a deep research of any other application similar with the B-Quest app. I wrote a report about an application that could be similar to B-Quest application. I help to develop the wireframes of the application, and also with the design of the mock-up of the application, collaboration with ideas relate with the structure of the document. Also, writing general and specifics object of the project and implement a working prototype. Also collaborate in the process of making a promotional video of the application.

- Chapter 2: I design use cases diagrams of the application. I collaborated with the structure of application and how will work the application, I wrote Functional Requirements and General Functional Requirements of the application and web app.

- Chapter 3: I collaborate with the design of the application and the interface of the application, research about the right colours and structure of the web application, participation with the design of the web app, and writing front end and user interface design of the web application for the documentation.

- Chapter 4: Writing technologies involves on the mobile application and implementation of the system.

- Chapter 6: Collaboration with writing the conclusions and making the final presentation of the project (video and slideshow).

In every group reunion I participated in the brain storms of ideas for the application and web application, most of the time relate with the functionality, structure and design of them.

Ailem Garcia Aka "La Chiquitik". As Project Manager in charge of planning and managing time resources, organising people and integrating cross-functional information important for the scope of the development of B-Quest by conducting regular meetings.

From the beginning I researched and put on action Project Management Tools such as Monday to have a method for sharing, measuring, organizing and assigning tasks to each team member. Once Basecamp (Basecamp, n.d.) was recommended I was in charge of creating the account and setting it up by migrating previous data in Monday to this new PMT and adding new and future tasks to be carried out.

As PM some of the tasks I had to keep in mind were:

- Planning and Defining Scope
- Resource Planning
- Developing Schedules
- Time Estimating
- Monitoring Progress
- Team Leadership
- Controlling Quality
- Assigning topics to each member so everybody would be fully involved and providing the necessary guidance.

My personal contribution for each chapter is the following:

Chapter I - INTRODUCTION:

- Creation of the YouTube Video "B-Quest Who we are".
- General and Specific Objectives
- Analysis of Situation, SWOT analysis
- Research of similar applications to that could influence B-Quest such as Wishpond and Gleam.
- B-Quest the Solution
- What the project is not about
- USE-CASE Diagrams
- Revision and assembling of full Chapter I

Chapter II System Analysis:

- System Description
- General Functional Requirements
- Class Diagrams
- Revision and assembling of full Chapter II

Chapter III System Design:

- Android Application UI Design.
- Data Design.
- Revision and assembling of full Chapter III.

Chapter IV:

- Revision and assembling of full Chapter IV.

Chapter V:

- Elaboration and Execution of test scenarios with the collaboration of Ronald Torres.
- Testing of Mobile Application.
- Commersialization and Marketing.
- Revision and assembling of full Chapter V.

Chapter VI:

- Conclusions with the collaboration of Mafer.
- Revision and assembling of full document.
- References

Finally, I would personally like to give my reflection about working with a team in the role of Project Manager. From the beginning the fear was that our team is a large one and that some members will no be involved in the project at all. I have to say that I'm very proud that it wasn't the case of B-Quest, the whole team put their heart on this project. We had our ups and downs, for each team member there were times of demotivation and times of enthusiasm. Thankfully, we managed to keep the team in sync by using the right skills of communication and motivation.

It wasn't an easy task to keep a big group motivated and active during challenging times, I think this was the hardest part of my role, dealing and leading 6 (including myself) different personalities, opinions and mind-sets. It wouldn't have been possible without the effort of each one of us. And for that, I'm grateful.

If I were to do things differently, I would only make one thing different, and it is the selection of the project. As most of the technologies for our project were new, the selection of a smaller project would have been better.

APPENDIX

TIMELINE

				March				April		
	Task Name WEEK	1	2	3	4	1	2	3	4	5
CHAPTER 3	Writing: User interface Desing (App)	🟩	🟩	🟩						
	Writing: Funtional Desing (App)	🟩	🟩	🟩						
	Coding: FrontEnd App	🟧	🟧	🟧	🟧					
	Coding: BackEnd App	🟧	🟧	🟧	🟧					
	Writing: Data Design	🟦	🟦	🟦						
	Building DataBase									
	Writing: User interface Desing (WebApp)	🟦	🟦	🟦						
	Writing: Funtional Desing (WebApp)	🟦	🟦	🟦						
	Coding: FrontEnd WebApp	🟥	🟥	🟥	🟥					
	Coding: BackEnd WebApp	🟫	🟫	🟫	🟫					
	Put everythig together				⬛					
CHAPTER 4	Writing: Technologies Involved (App)			🟩	🟩	🟩				
	Writing: Technologies Involved (WebApp)			🟪	🟪	🟪				
	Writing: Implemetation of the system (App)			🟩	🟩	🟩				
	Writing: Implemetation of the system (WebApp)			🟪	🟪	🟪				
	Writing: Problems encountered (App)			🟩	🟩	🟩				
	Writing: Problems encountered (WebApp)			🟪	🟪	🟪				
	Testing the aplication						🟨	🟨		
	Testing the WebApp						🟧	🟧		
	Put everythig together						⬛			
CHAPTER 5	Writing: Testing. Methodology App and WebApp						🟨	🟨	🟨	
CHAPTER 6	Writing: Conclusions									🟧
	Put everythig together									⬛

QUESTIONNAIRE

Mowhile Geek

Project: B-Quest

- How do you find choosing a birthday present for your colleagues, friends or family?

Very difficult/ Difficult/ Neither difficult nor easy/ Easy / Very easy

- Would you prefer to give money for birthday presents if there was a fun way to do it?

Strongly Disagree / Disagree / Neither disagree Nor Agree / Agree / Strongly Agree

- Do you often enjoy the presents you receive for your birthday?

Strongly Disagree / Disagree / Neither disagree Nor Agree / Agree / Strongly Agree

- Would you consider receiving money for your birthday as a present?

Strongly Disagree / Disagree / Neither disagree Nor Agree / Agree / Strongly Agree

- Would you participate in activities that your friends have created to get you your birthday present?

Strongly Disagree / Disagree / Neither disagree Nor Agree / Agree / Strongly Agree

GLOSSARY

Chief: is the user in charge of the group, He will be the ruler of the clan. A new **chief** for the treasure hunt could be appointed or else the creator of the clan will hold this position by default. The last vote for approve or decline the result of a quest will be in his/her fingers.

Lord: Member of the treasure hunt. Will have a vote to decide if a quest is approved or not. Also, the lord will contribute with a donation for the treasure reward.

B-Hero: The honouree guest of the Treasure hunt, the Birthday-Hero. This person will receive notifications of the special gift that will be receive after completing all the quests included on the treasure hunt.

Guest: user that visits the website for the first time and haven't registered yet.

Quest: A special activity assigned by the Chief and Lords that needs to be complete for the B-quest hero in order to get the Treasure.

Treasure: The final prize that will receive the B-Hero after completing the treasure hunt.

Treasure Hunt: an event/activity in which the B- Hero search for hidden objects or completing different Quests by following a trail of clues or instructions.

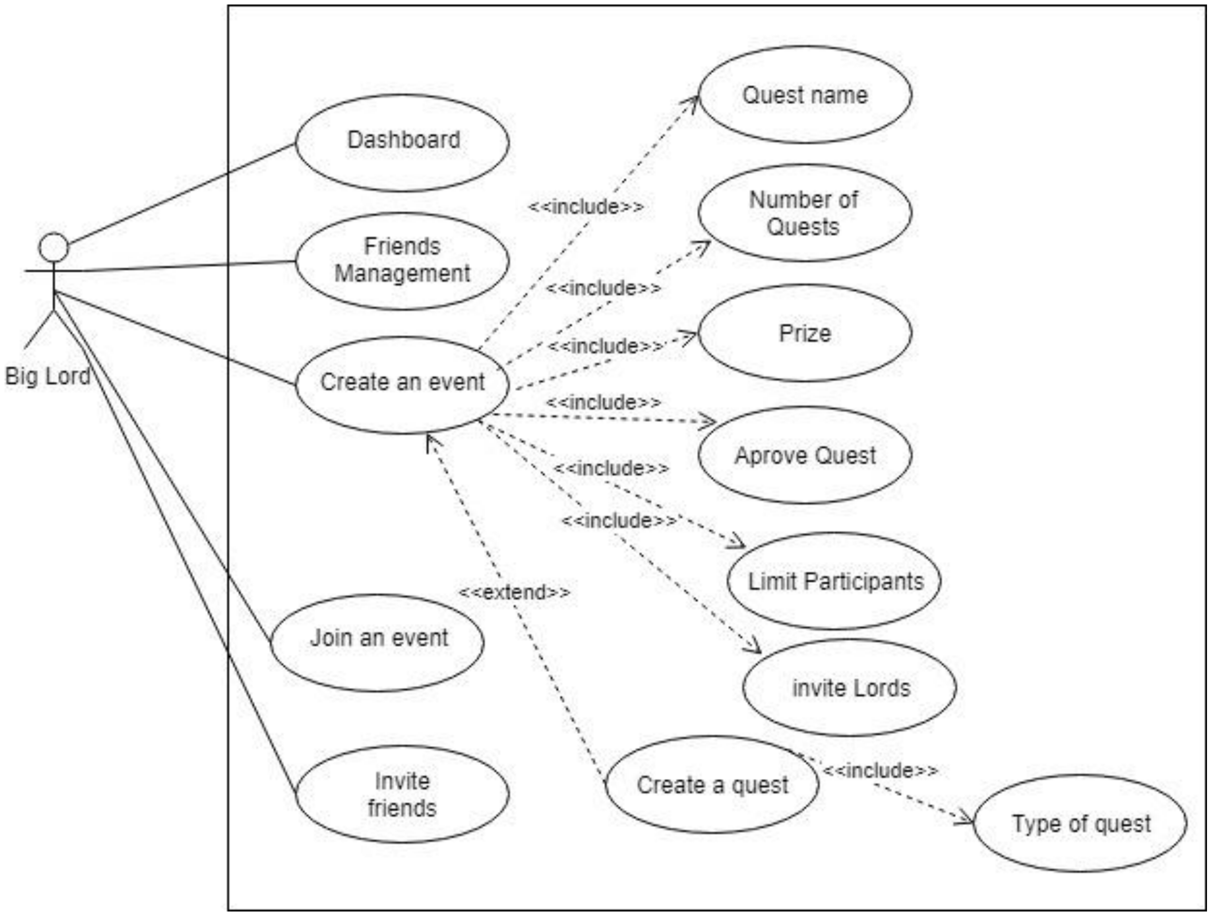
Clan: a group of people with a strong common interest, members of the treasure hunt.

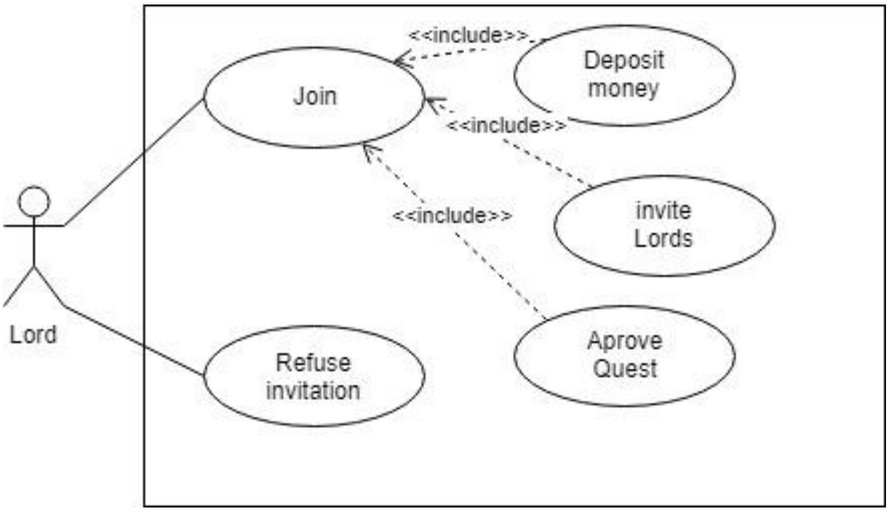
DIAGRAMS

Use Cases

Use case: Create an event.

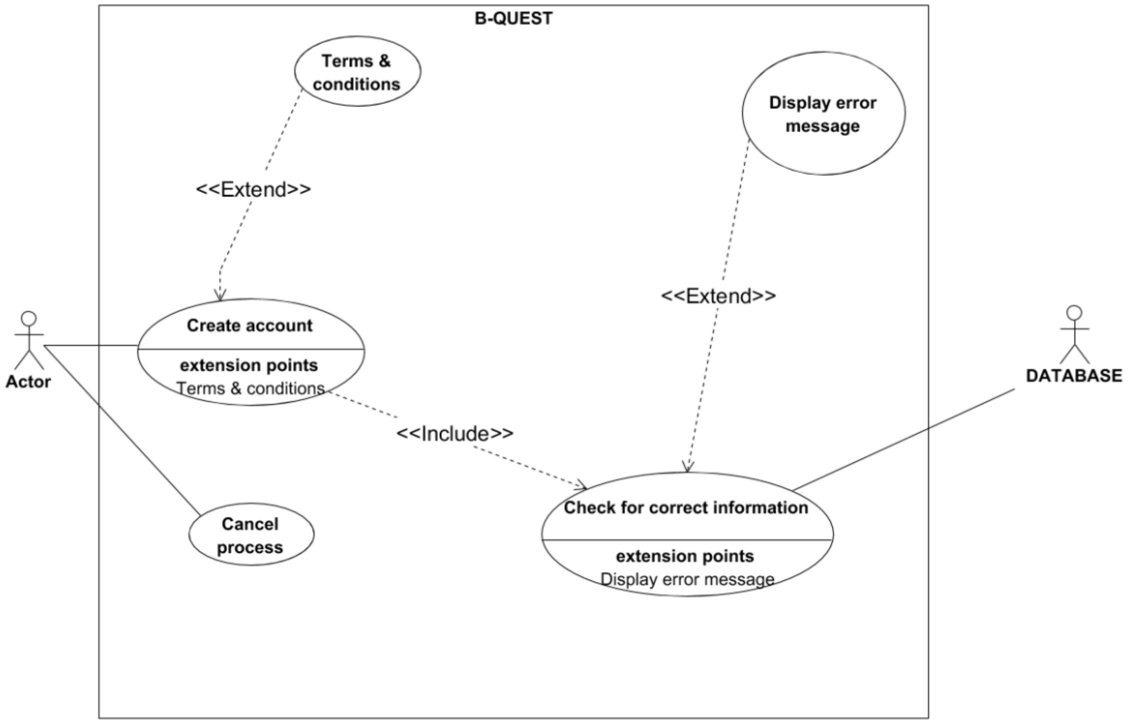
Pre-requisite: Must have created an account first.





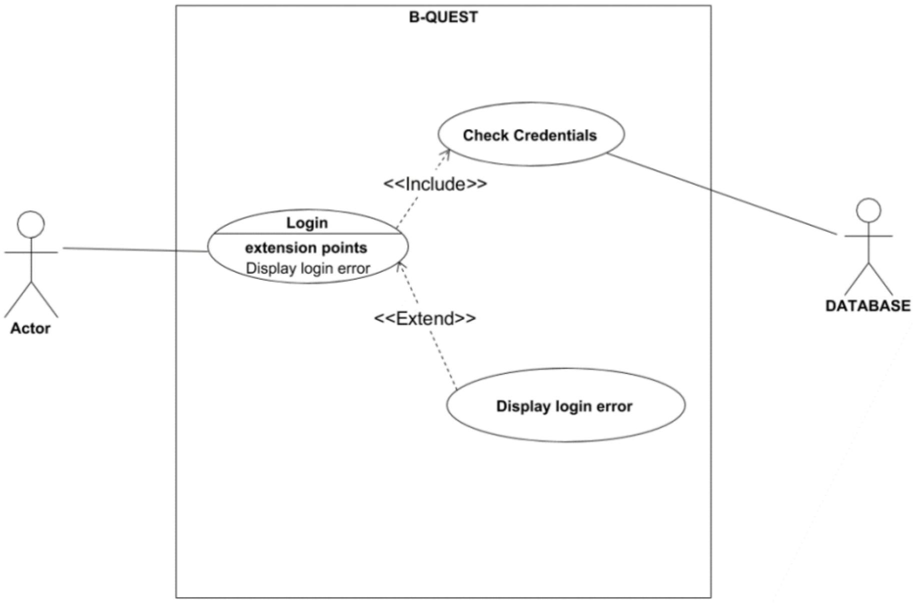
Use case: Sign up

Pre-requisite: Must not have an account.



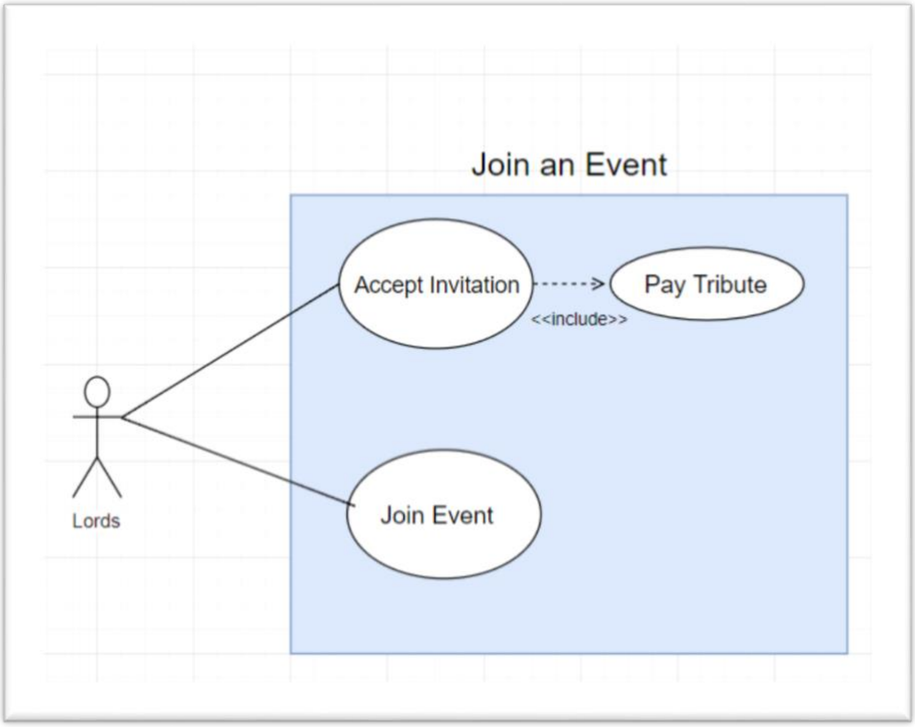
Use case: Login.

Pre-requisite: must have been created an account.



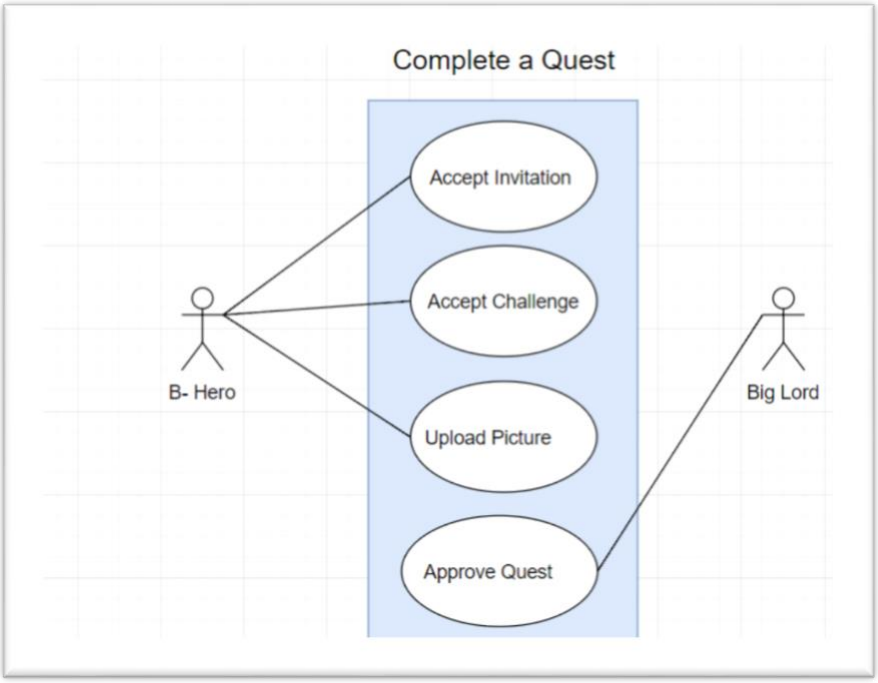
Use case: Join an Event

Pre-requisite: Must have created an account



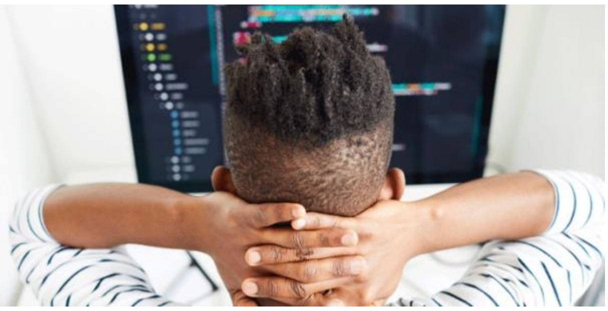
Use case: Complete a quest

Pre-requisite: B-Hero must have created an account and received an invitation to join it.



WEB APP FIRST STAGES

Initial Research



Web development tools have come a long way in just a few short years. Thanks to this progress, it is thinkable to harness the power of highly tested libraries to improve our workflow and benefit from greater possibilities when it comes to responsive design. Not only that, it is possible to build things together thanks to ever-improving version control systems. From browser add-ons and plugins, to processors that streamline the code, there have never been more possibilities for creating awesome web applications.

But with the number of web dev tools increasing almost daily, finding the best software to get the job done can sometimes feel daunting.

Chrome Developer Tools

Google's built-in Chrome Developer Tools let our team do just that. Bundled and available in both Chrome and Safari, they allow developers access into the internals of their web application. On top of this, a palette of network tools can help optimize the loading flows, while a timeline gives a deeper understanding of what the browser is doing at any given moment.

Angular.js

HTML is usually the cornerstone of any front-end developer's toolbox, but it has what many perceive to be a serious flaw: it was not designed to manage dynamic views.

This is where AngularJS, an open-source web application framework, comes in. Developed by Google, AngularJS lets extend the application's HTML syntax, resulting in a more expressive, readable, and quick to develop environment that could otherwise not have been built with HTML alone.

GitHub

It is every developer’s worst nightmare – The team has been working on a new project feature and something bad happens. Enter version control systems (VCS) – and more specifically, GitHub.

By rolling out the B-Quest project with the service, it will be able to view any changes that the team have made or even go back to the previous state (making pesky mistakes a thing of the past). The repository hosting service also boasts a rich open-source development community (making collaboration between teams as easy as pie), as well as providing several other components such as bug tracking, feature requests, task management, and wikis for every project.

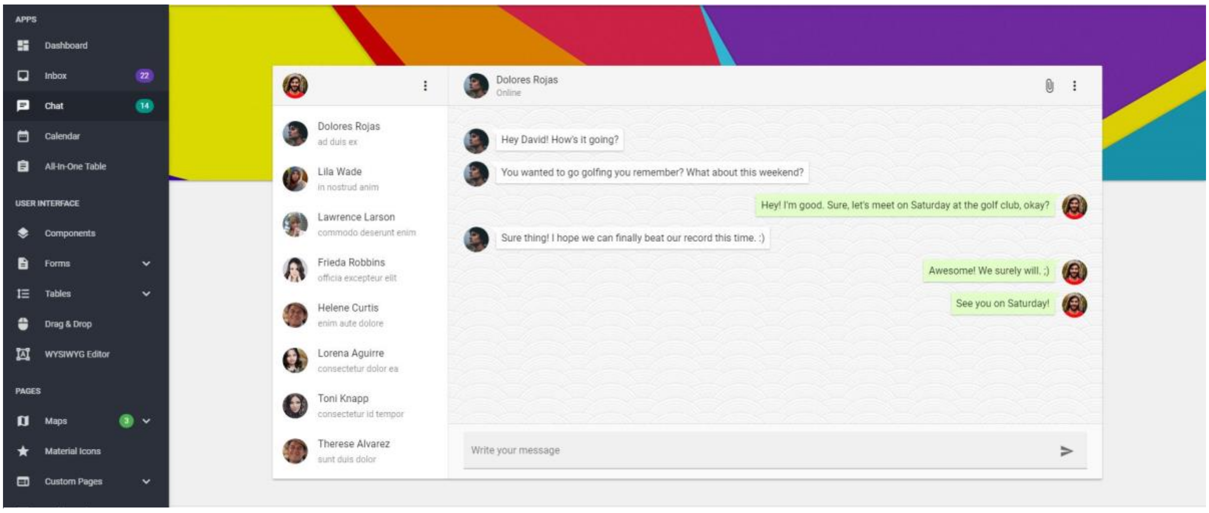
It is a great way to get involved and learn from the best with a wide array of open-source projects to work on.

Bootstrap

UI frameworks are an attempt to solve problems by abstracting the common elements into reusable modules - meaning the Mowhile developers’ team could scaffold the elements of new applications with speed and ease.

The most widely used of these frameworks is Bootstrap, a comprehensive UI package developed by the team at Twitter. Complete with tools to normalize stylesheets, build modal objects, add JavaScript plugins, and a plethora of other features, Bootstrap can dramatically cut down on the amount of code (and time) needed to build the B-Quest project.

Wireframes Web App V1



Dolores Rojas
ad duis ex

Lila Wade
in nostrud anim

Lawrence Larson
commodo deserunt enim

Frieda Robbins
officia excepteur elit

Helene Curtis
enim aute dolore

Lorena Aguirre
consectetur dolor ea

Toni Knapp
consectetur id tempor

Therese Alvarez
sunt duis dolor

Dolores Rojas
Online

Hey David! How's it going?

You wanted to go golfing you remember? What about this weekend?

Hey! I'm good. Sure, let's meet on Saturday at the golf club, okay?

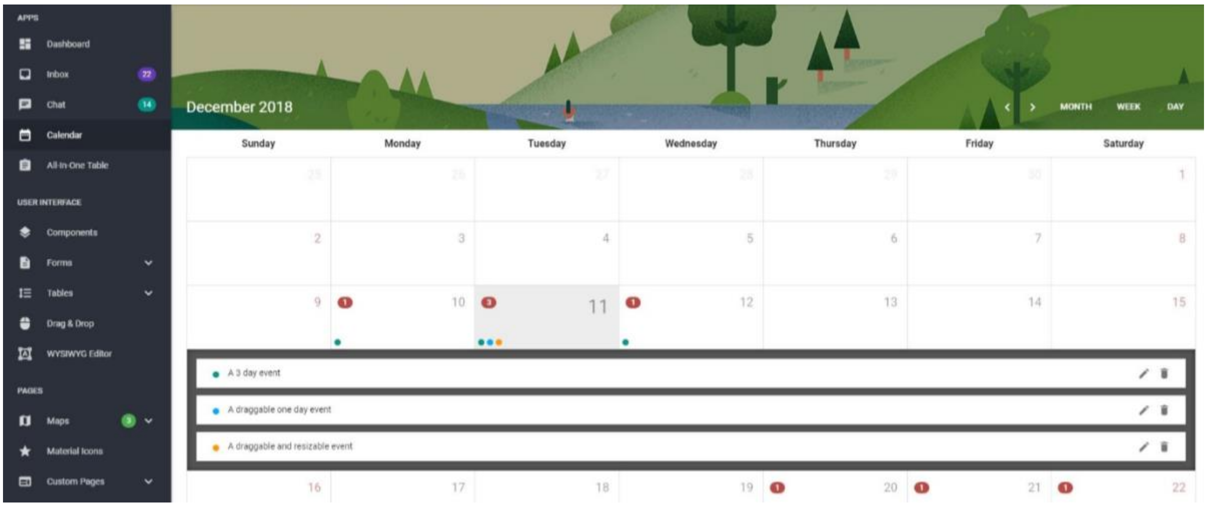
Sure thing! I hope we can finally beat our record this time. :)

Awesome! We surely will. ;)

See you on Saturday!

Write your message

Figure 1.- The Chief will be able to chat with each member of the group



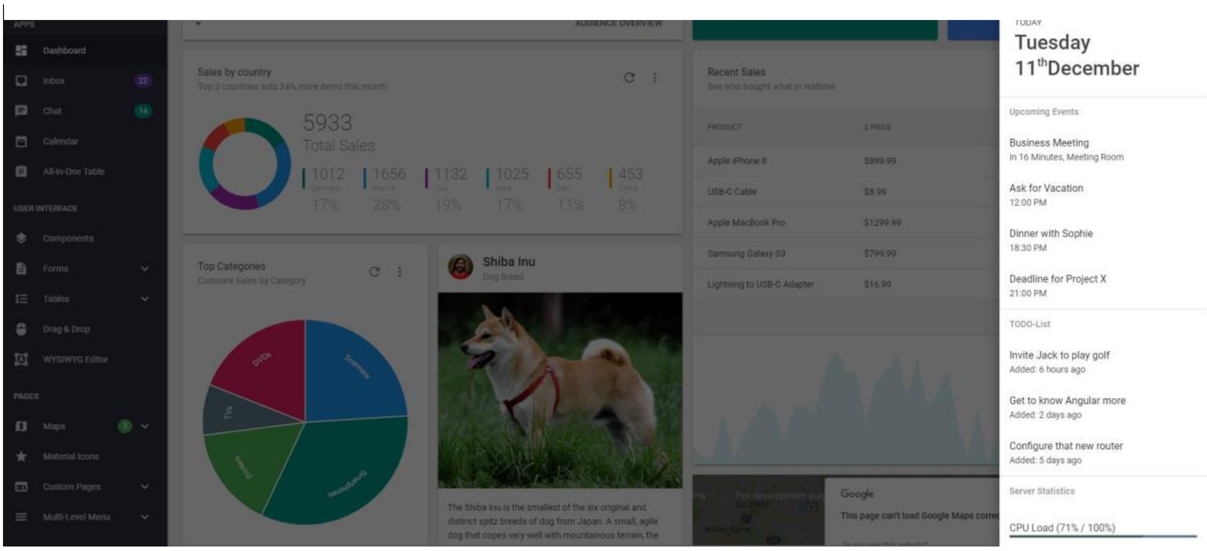
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22

● A 3 day event

● A draggable one day event

● A draggable and resizable event

Figure 2.- The calendar will be a helpful tool to follow the treasure hunt



Sales by country
Top 3 countries sold 34% more items this month

5933
Total Sales

1012	1656	1132	1025	655	453
Germany	France	USA	India	Italy	China
17%	28%	19%	17%	11%	8%

Top Categories
Compare Sales by Category

Shiba Inu
Dog Breed

The Shiba Inu is the smallest of the six original and distinct spitz breeds of dog from Japan. A small, agile dog that copes very well with mountainous terrain, the

Recent Sales
See who bought what in realtime

PRODUCT	$ PRICE
Apple iPhone 8	$899.99
USB-C Cable	$8.99
Apple MacBook Pro	$1299.99
Samsung Galaxy S9	$799.99
Lightning to USB-C Adapter	$16.99

This page can't load Google Maps correc

TODAY
Tuesday
11thDecember

Upcoming Events

Business Meeting
In 16 Minutes, Meeting Room

Ask for Vacation
12:00 PM

Dinner with Sophie
18:30 PM

Deadline for Project X
21:00 PM

TODO-List

Invite Jack to play golf
Added: 6 hours ago

Get to know Angular more
Added: 2 days ago

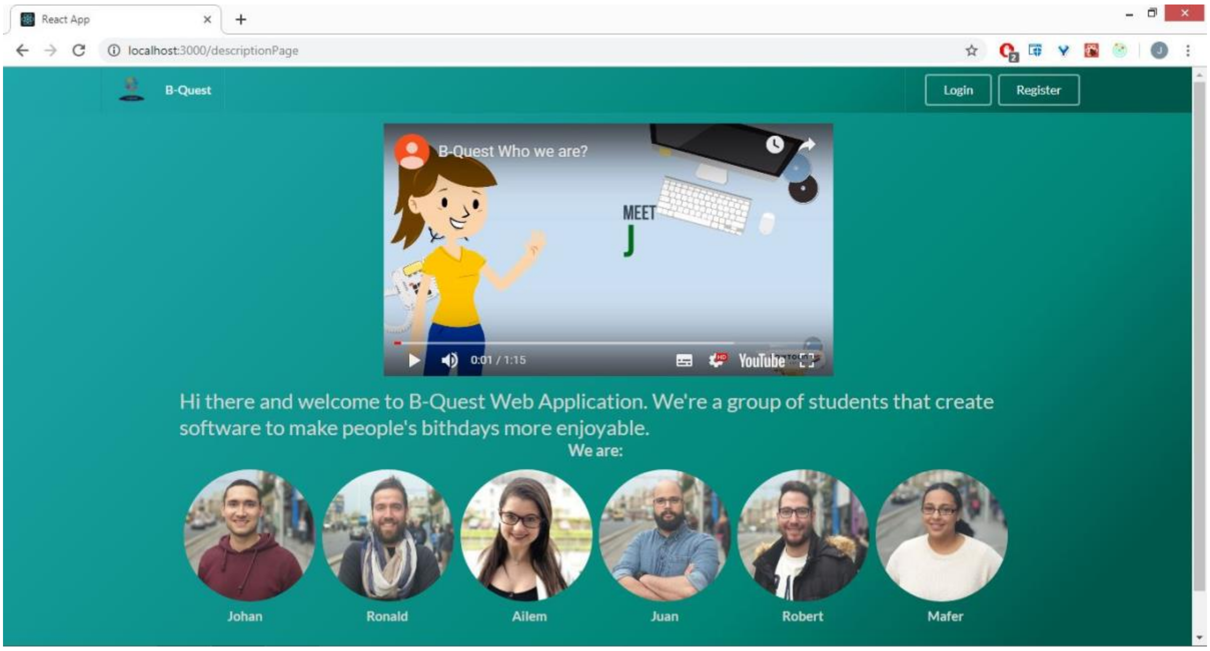
Configure that new router
Added: 5 days ago

Server Statistics

CPU Load (71% / 100%)

Figure 3.- Many tools will be able for the users that will improve their experience using B-Quest

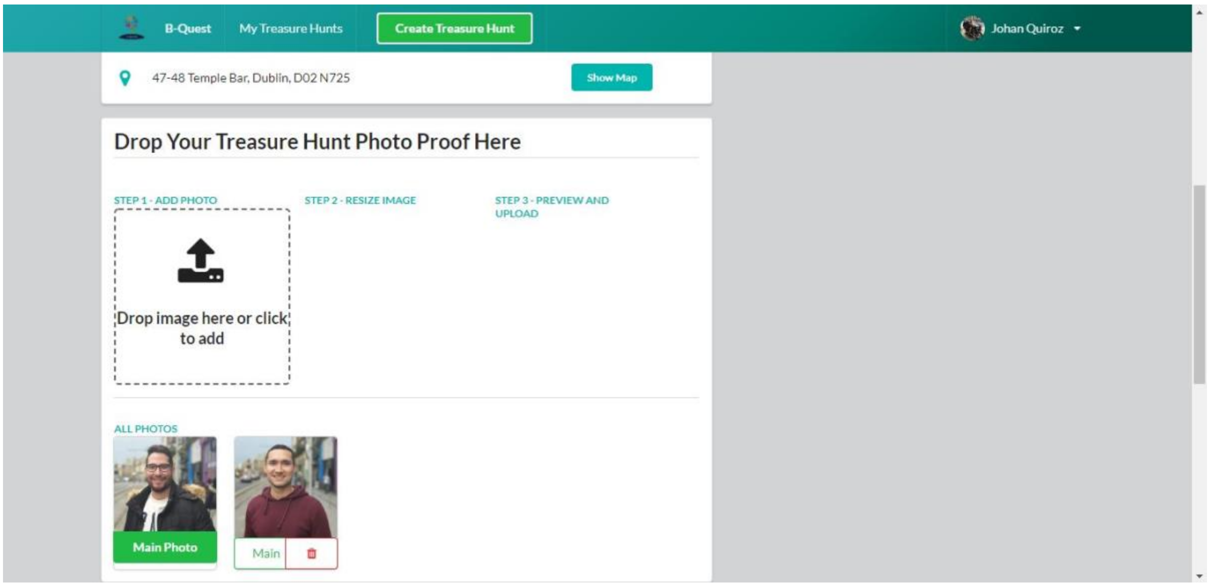
Web Application UI



Hi there and welcome to B-Quest Web Application. We're a group of students that create software to make people's bithdays more enjoyable.

We are:

Figure 1.- Description Page



B-Quest
My Treasure Hunts
Create Treasure Hunt
Johan Quiroz
47-48 Temple Bar, Dublin, D02 N725
Show Map
Drop Your Treasure Hunt Photo Proof Here
STEP 1 - ADD PHOTO
STEP 2 - RESIZE IMAGE
STEP 3 - PREVIEW AND UPLOAD
Drop image here or click to add
ALL PHOTOS
Main Photo
Main

Figure 2.- Web Application - Upload photo for Treasure Hunt

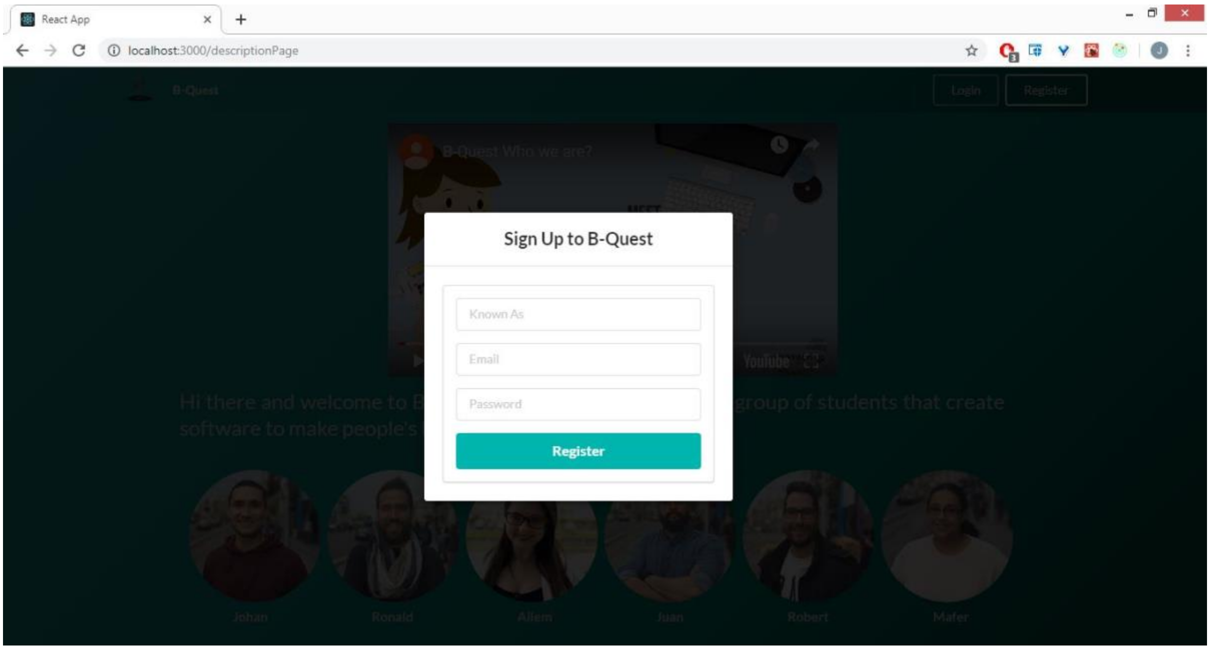
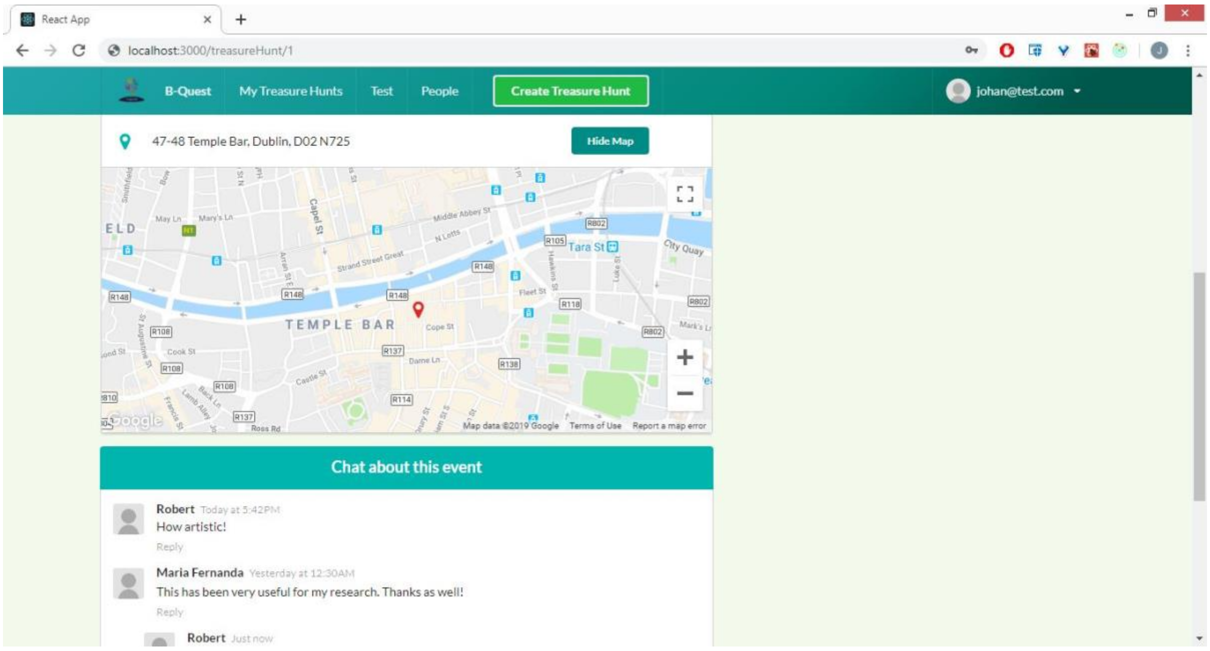


Figure 3.- Web Application - Registration Page



47-48 Temple Bar, Dublin, D02 N725

Chat about this event

Robert Today at 5:42PM
How artistic!
Reply

Maria Fernanda Yesterday at 12:30AM
This has been very useful for my research. Thanks as well!
Reply

Robert Just now

Figure 4.- Web Application - Meeting point of the Treasure Hunt

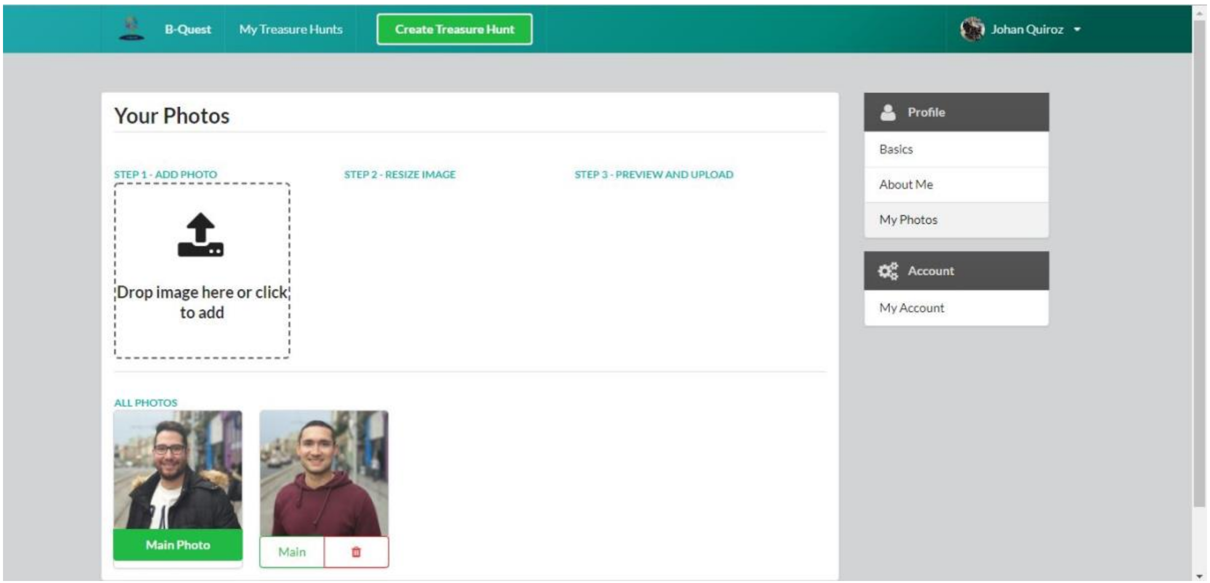
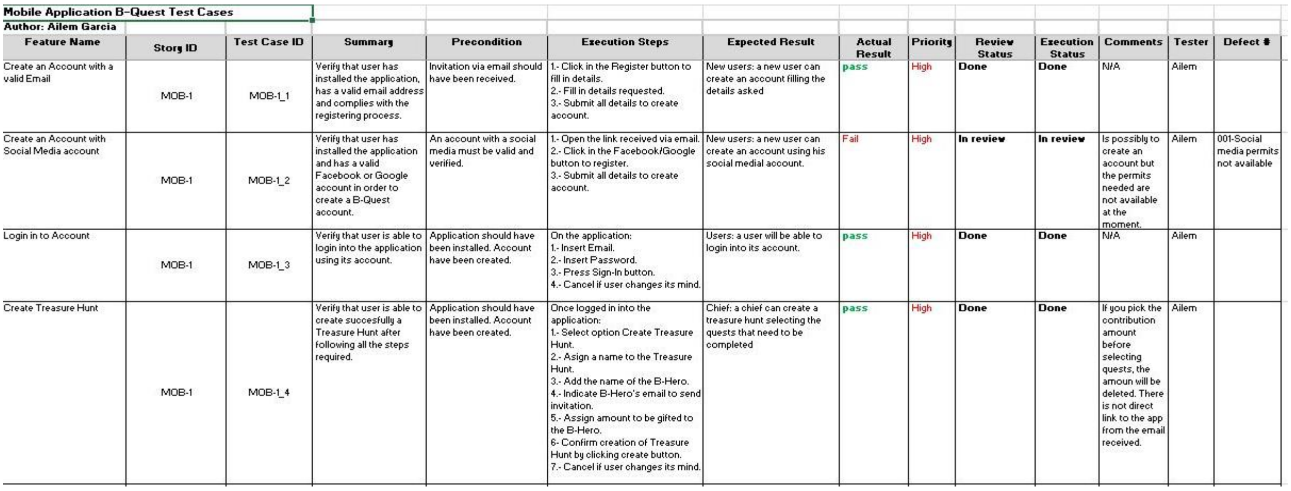


Figure 5.- Web Application - Profile Picture page

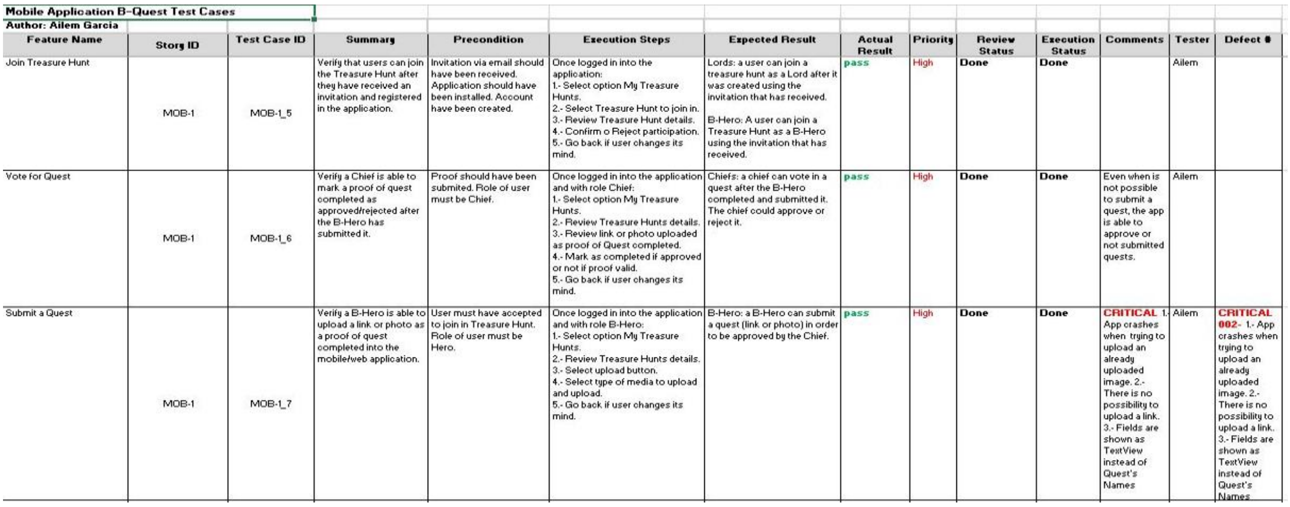
VERSION I – TESTING RESULTS



Mobile Application B-Quest Test Cases

Author: Ailem Garcia

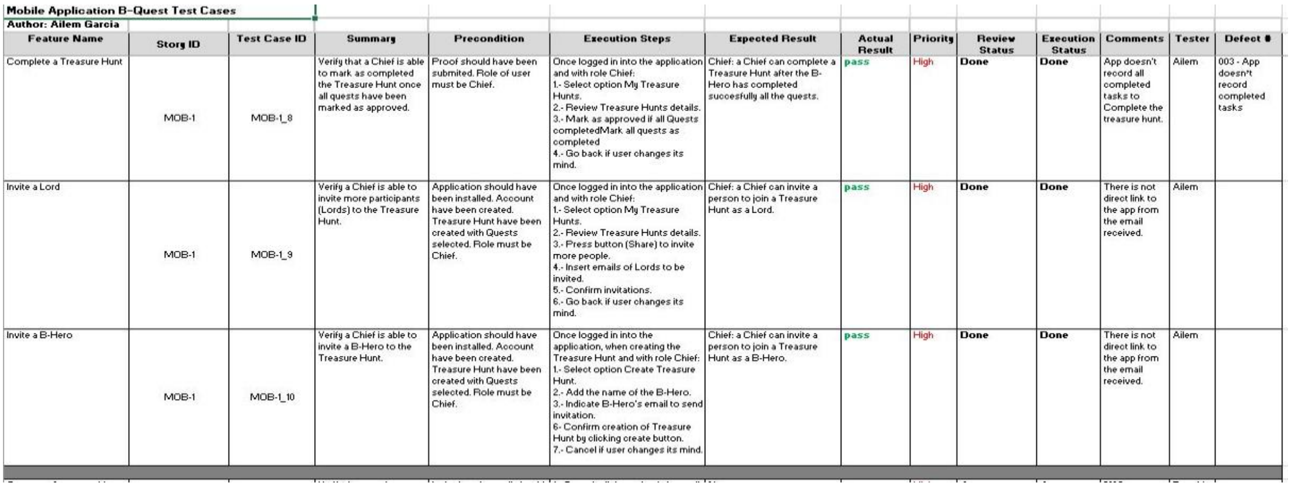
Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result	Priority	Review Status	Execution Status	Comments	Tester	Defect #
Create an Account with a valid Email	MOB-1	MOB-1_1	Verify that user has installed the application, has a valid email address and complies with the registering process.	Invitation via email should have been received.	1.- Click in the Register button to fill in details. 2.- Fill in details requested. 3.- Submit all details to create account.	New users: a new user can create an account filling the details asked	pass	High	**Done**	**Done**	N/A	Ailem	
Create an Account with Social Media account	MOB-1	MOB-1_2	Verify that user has installed the application and has a valid Facebook or Google account in order to create a B-Quest account.	An account with a social media must be valid and verified.	1.- Open the link received via email. 2.- Click in the Facebook/Google button to register. 3.- Submit all details to create account.	New users: a new user can create an account using his social medial account.	Fail	High	**In review**	**In review**	Is possibly to create an account but the permits needed are not available at the moment.	Ailem	001-Social media permits not available
Login in to Account	MOB-1	MOB-1_3	Verify that user is able to login into the application using its account.	Application should have been installed. Account have been created.	On the application: 1.- Insert Email. 2.- Insert Password. 3.- Press Sign-In button. 4.- Cancel if user changes its mind.	Users: a user will be able to login into its account.	pass	High	**Done**	**Done**	N/A	Ailem	
Create Treasure Hunt	MOB-1	MOB-1_4	Verify that user is able to create succesfully a Treasure Hunt after following all the steps required.	Application should have been installed. Account have been created.	Once logged in into the application: 1.- Select option Create Treasure Hunt. 2.- Asign a name to the Treasure Hunt. 3.- Add the name of the B-Hero. 4.- Indicate B-Hero's email to send invitation. 5.- Assign amount to be gifted to the B-Hero. 6- Confirm creation of Treasure Hunt by clicking create button. 7.- Cancel if user changes its mind.	Chief: a chief can create a treasure hunt selecting the quests that need to be completed	pass	High	**Done**	**Done**	If you pick the contribution amount before selecting quests, the amoun will be deleted. There is not direct link to the app from the email received.	Ailem	



Mobile Application B-Quest Test Cases

Author: Ailem Garcia

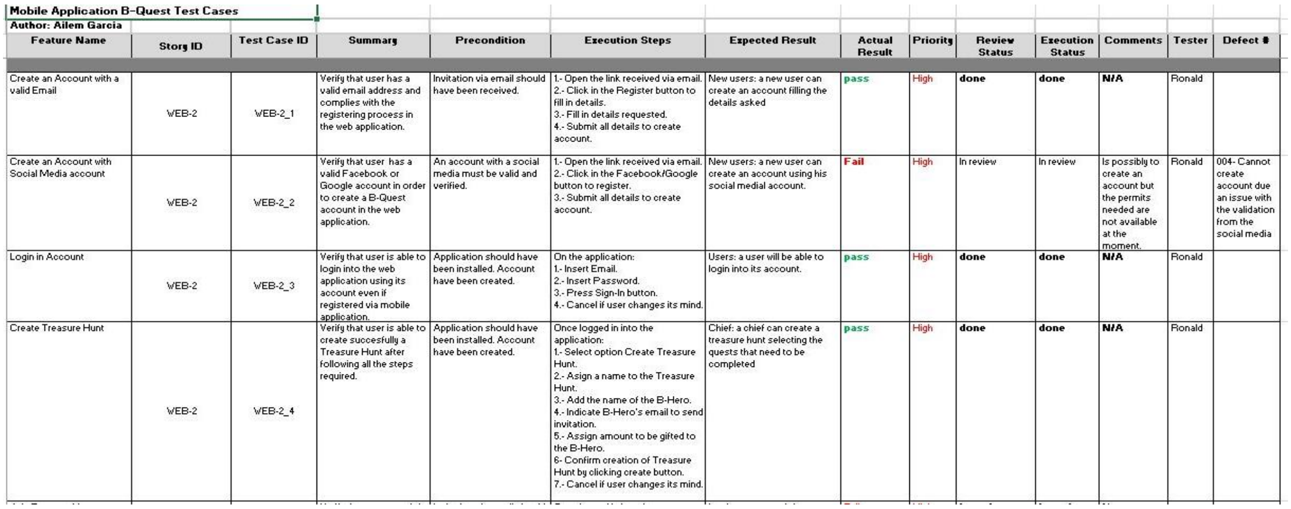
Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result	Priority	Review Status	Execution Status	Comments	Tester	Defect #
Join Treasure Hunt	MOB-1	MOB-1_5	Verify that users can join the Treasure Hunt after they have received an invitation and registered in the application.	Invitation via email should have been received. Application should have been installed. Account have been created.	Once logged in into the application: 1.- Select option My Treasure Hunts. 2.- Select Treasure Hunt to join in. 3.- Review Treasure Hunt details. 4.- Confirm o Reject participation. 5.- Go back if user changes its mind.	Lords: a user can join a treasure hunt as a Lord after it was created using the invitation that has received. B-Hero: A user can join a Treasure Hunt as a B-Hero using the invitation that has received.	pass	High	**Done**	**Done**		Ailem	
Vote for Quest	MOB-1	MOB-1_6	Verify a Chief is able to mark a proof of quest completed as approved/rejected after the B-Hero has submitted it.	Proof should have been submited. Role of user must be Chief.	Once logged in into the application and with role Chief: 1.- Select option My Treasure Hunts. 2.- Review Treasure Hunts details. 3.- Review link or photo uploaded as proof of Quest completed. 4.- Mark as completed if approved or not if proof valid. 5.- Go back if user changes its mind.	Chiefs: a chief can vote in a quest after the B-Hero completed and submitted it. The chief could approve or reject it.	pass	High	**Done**	**Done**	Even when is not possible to submit a quest, the app is able to approve or not submitted quests.	Ailem	
Submit a Quest	MOB-1	MOB-1_7	Verify a B-Hero is able to upload a link or photo as a proof of quest completed into the mobile/web application.	User must have accepted to join in Treasure Hunt. Role of user must be Hero.	Once logged in into the application and with role B-Hero: 1.- Select option My Treasure Hunts. 2.- Review Treasure Hunts details. 3.- Select upload button. 4.- Select type of media to upload and upload. 5.- Go back if user changes its mind.	B-Hero: a B-Hero can submit a quest (link or photo) in order to be approved by the Chief.	pass	High	**Done**	**Done**	**CRITICAL** 1.- App crashes when trying to upload an already uploaded image. 2.- There is no possibility to upload a link. 3.- Fields are shown as TextView instead of Quest's Names	Ailem	**CRITICAL 002-** 1.- App crashes when trying to upload an already uploaded image. 2.- There is no possibility to upload a link. 3.- Fields are shown as TextView instead of Quest's Names



Mobile Application B-Quest Test Cases

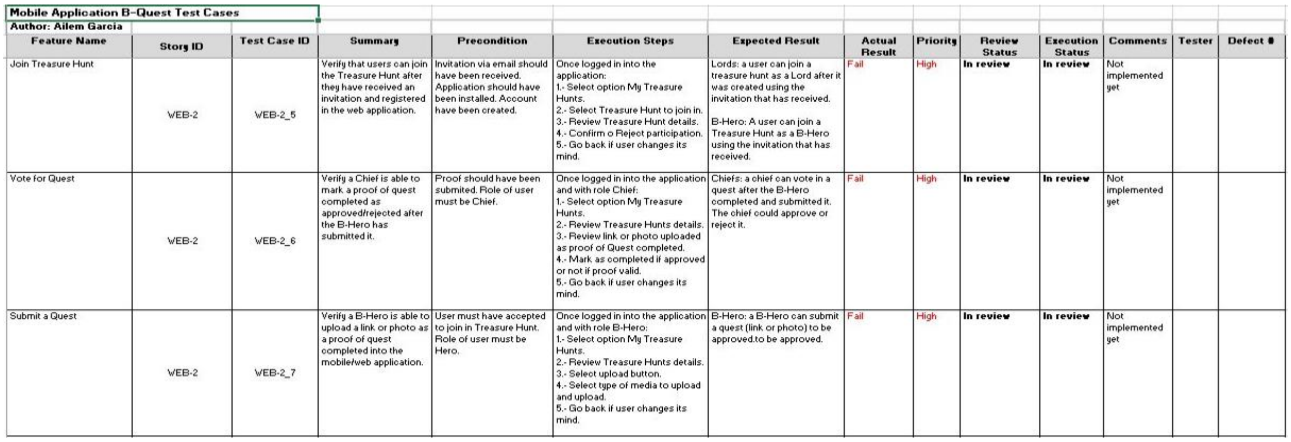
Author: Ailem Garcia

Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result	Priority	Review Status	Execution Status	Comments	Tester	Defect #
Complete a Treasure Hunt	MOB-1	MOB-1_8	Verify that a Chief is able to mark as completed the Treasure Hunt once all quests have been marked as approved.	Proof should have been submited. Role of user must be Chief.	Once logged in into the application and with role Chief: 1.- Select option My Treasure Hunts. 2.- Review Treasure Hunts details. 3.- Mark as approved if all Quests completedMark all quests as completed 4.- Go back if user changes its mind.	Chief: a Chief can complete a Treasure Hunt after the B-Hero has completed succesfully all the quests.	pass	High	**Done**	**Done**	App doesn't record all completed tasks to Complete the treasure hunt.	Ailem	003 - App doesn't record completed tasks
Invite a Lord	MOB-1	MOB-1_9	Verify a Chief is able to invite more participants (Lords) to the Treasure Hunt.	Application should have been installed. Account have been created. Treasure Hunt have been created with Quests selected. Role must be Chief.	Once logged in into the application and with role Chief: 1.- Select option My Treasure Hunts. 2.- Review Treasure Hunts details. 3.- Press button (Share) to invite more people. 4.- Insert emails of Lords to be invited. 5.- Confirm invitations. 6.- Go back if user changes its mind.	Chief: a Chief can invite a person to join a Treasure Hunt as a Lord.	pass	High	**Done**	**Done**	There is not direct link to the app from the email received.	Ailem	
Invite a B-Hero	MOB-1	MOB-1_10	Verify a Chief is able to invite a B-Hero to the Treasure Hunt.	Application should have been installed. Account have been created. Treasure Hunt have been created with Quests selected. Role must be Chief.	Once logged in into the application, when creating the Treasure Hunt and with role Chief: 1.- Select option Create Treasure Hunt. 2.- Add the name of the B-Hero. 3.- Indicate B-Hero's email to send invitation. 6- Confirm creation of Treasure Hunt by clicking create button. 7.- Cancel if user changes its mind.	Chief: a Chief can invite a person to join a Treasure Hunt as a B-Hero.	pass	High	**Done**	**Done**	There is not direct link to the app from the email received.	Ailem	



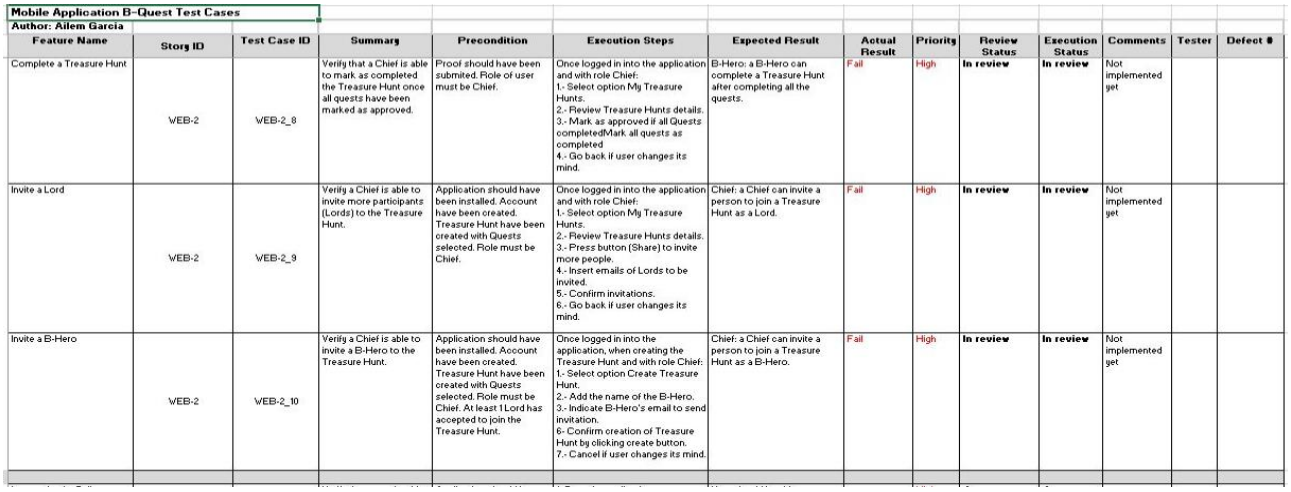
Mobile Application B-Quest Test Cases
Author: Ailem Garcia

Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result	Priority	Review Status	Execution Status	Comments	Tester	Defect #
Create an Account with a valid Email	WEB-2	WEB-2_1	Verify that user has a valid email address and complies with the registering process in the web application.	Invitation via email should have been received.	1.- Open the link received via email. 2.- Click in the Register button to fill in details. 3.- Fill in details requested. 4.- Submit all details to create account.	New users: a new user can create an account filling the details asked	pass	High	done	done	N/A	Ronald	
Create an Account with Social Media account	WEB-2	WEB-2_2	Verify that user has a valid Facebook or Google account in order to create a B-Quest account in the web application.	An account with a social media must be valid and verified.	1.- Open the link received via email. 2.- Click in the Facebook/Google button to register. 3.- Submit all details to create account.	New users: a new user can create an account using his social medial account.	Fail	High	In review	In review	Is possibly to create an account but the permits needed are not available at the moment.	Ronald	004- Cannot create account due an issue with the validation from the social media
Login in Account	WEB-2	WEB-2_3	Verify that user is able to login into the web application using its account even if registered via mobile application.	Application should have been installed. Account have been created.	On the application: 1.- Insert Email. 2.- Insert Password. 3.- Press Sign-In button. 4.- Cancel if user changes its mind.	Users: a user will be able to login into its account.	pass	High	done	done	N/A	Ronald	
Create Treasure Hunt	WEB-2	WEB-2_4	Verify that user is able to create succesfully a Treasure Hunt after following all the steps required.	Application should have been installed. Account have been created.	Once logged in into the application: 1.- Select option Create Treasure Hunt. 2.- Asign a name to the Treasure Hunt. 3.- Add the name of the B-Hero. 4.- Indicate B-Hero's email to send invitation. 5.- Assign amount to be gifted to the B-Hero. 6- Confirm creation of Treasure Hunt by clicking create button. 7.- Cancel if user changes its mind.	Chief: a chief can create a treasure hunt selecting the quests that need to be completed	pass	High	done	done	N/A	Ronald	



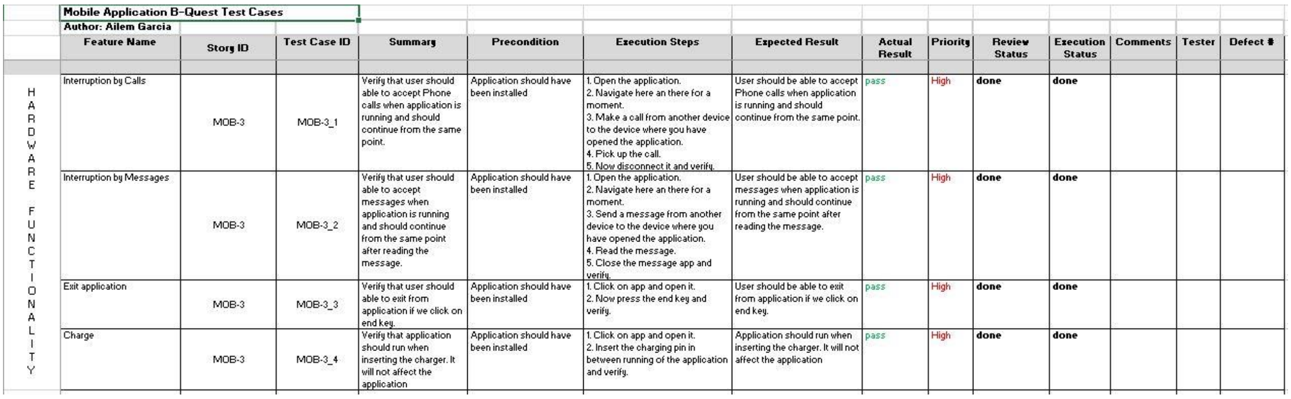
Mobile Application B-Quest Test Cases
Author: Ailem Garcia

Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result	Priority	Review Status	Execution Status	Comments	Tester	Defect #
Join Treasure Hunt	WEB-2	WEB-2_5	Verify that users can join the Treasure Hunt after they have received an invitation and registered in the web application.	Invitation via email should have been received. Application should have been installed. Account have been created.	Once logged in into the application: 1.- Select option My Treasure Hunts. 2.- Select Treasure Hunt to join in. 3.- Review Treasure Hunt details. 4.- Confirm o Reject participation. 5.- Go back if user changes its mind.	Lords: a user can join a treasure hunt as a Lord after it was created using the invitation that has received. B-Hero: A user can join a Treasure Hunt as a B-Hero using the invitation that has received.	Fail	High	In review	In review	Not implemented yet		
Vote for Quest	WEB-2	WEB-2_6	Verify a Chief is able to mark a proof of quest completed as approved/rejected after the B-Hero has submitted it.	Proof should have been submited. Role of user must be Chief.	Once logged in into the application and with role Chief: 1.- Select option My Treasure Hunts. 2.- Review Treasure Hunts details. 3.- Review link or photo uploaded as proof of Quest completed. 4.- Mark as completed if approved or not if proof valid. 5.- Go back if user changes its mind.	Chiefs: a chief can vote in a quest after the B-Hero completed and submitted it. The chief could approve or reject it.	Fail	High	In review	In review	Not implemented yet		
Submit a Quest	WEB-2	WEB-2_7	Verify a B-Hero is able to upload a link or photo as a proof of quest completed into the mobile/web application.	User must have accepted to join in Treasure Hunt. Role of user must be Hero.	Once logged in into the application and with role B-Hero: 1.- Select option My Treasure Hunts. 2.- Review Treasure Hunts details. 3.- Select upload button. 4.- Select type of media to upload and upload. 5.- Go back if user changes its mind.	B-Hero: a B-Hero can submit a quest (link or photo) to be approved.to be approved.	Fail	High	In review	In review	Not implemented yet		



Mobile Application B-Quest Test Cases
Author: Ailem Garcia

Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result	Priority	Review Status	Execution Status	Comments	Tester	Defect #
Complete a Treasure Hunt	WEB-2	WEB-2_8	Verify that a Chief is able to mark as completed the Treasure Hunt once all quests have been marked as approved.	Proof should have been submited. Role of user must be Chief.	Once logged in into the application and with role Chief: 1.- Select option My Treasure Hunts. 2.- Review Treasure Hunts details. 3.- Mark as approved if all Quests completedMark all quests as completed 4.- Go back if user changes its mind.	B-Hero: a B-Hero can complete a Treasure Hunt after completing all the quests.	Fail	High	In review	In review	Not implemented yet		
Invite a Lord	WEB-2	WEB-2_9	Verify a Chief is able to invite more participants (Lords) to the Treasure Hunt.	Application should have been installed. Account have been created. Treasure Hunt have been created with Quests selected. Role must be Chief.	Once logged in into the application and with role Chief: 1.- Select option My Treasure Hunts. 2.- Review Treasure Hunts details. 3.- Press button (Share) to invite more people. 4.- Insert emails of Lords to be invited. 5.- Confirm invitations. 6.- Go back if user changes its mind.	Chief: a Chief can invite a person to join a Treasure Hunt as a Lord.	Fail	High	In review	In review	Not implemented yet		
Invite a B-Hero	WEB-2	WEB-2_10	Verify a Chief is able to invite a B-Hero to the Treasure Hunt.	Application should have been installed. Account have been created. Treasure Hunt have been created with Quests selected. Role must be Chief. At least 1 Lord has accepted to join the Treasure Hunt.	Once logged in into the application, when creating the Treasure Hunt and with role Chief: 1.- Select option Create Treasure Hunt. 2.- Add the name of the B-Hero. 3.- Indicate B-Hero's email to send invitation. 6- Confirm creation of Treasure Hunt by clicking create button. 7.- Cancel if user changes its mind.	Chief: a Chief can invite a person to join a Treasure Hunt as a B-Hero.	Fail	High	In review	In review	Not implemented yet		



Mobile Application B-Quest Test Cases

Author: Ailem Garcia

	Feature Name	Story ID	Test Case ID	Summary	Precondition	Execution Steps	Expected Result	Actual Result	Priority	Review Status	Execution Status	Comments	Tester	Defect #
HARDWARE FUNCTIONALITY	Interruption by Calls	MOB-3	MOB-3_1	Verify that user should able to accept Phone calls when application is running and should continue from the same point.	Application should have been installed	1. Open the application. 2. Navigate here an there for a moment. 3. Make a call from another device to the device where you have opened the application. 4. Pick up the call. 5. Now disconnect it and verify.	User should be able to accept Phone calls when application is running and should continue from the same point.	pass	High	**done**	**done**			
	Interruption by Messages	MOB-3	MOB-3_2	Verify that user should able to accept messages when application is running and should continue from the same point after reading the message.	Application should have been installed	1. Open the application. 2. Navigate here an there for a moment. 3. Send a message from another device to the device where you have opened the application. 4. Read the message. 5. Close the message app and verify.	User should be able to accept messages when application is running and should continue from the same point after reading the message.	pass	High	**done**	**done**			
	Exit application	MOB-3	MOB-3_3	Verify that user should able to exit from application if we click on end key.	Application should have been installed	1. Click on app and open it. 2. Now press the end key and verify.	User should be able to exit from application if we click on end key.	pass	High	**done**	**done**			
	Charge	MOB-3	MOB-3_4	Verify that application should run when inserting the charger. It will not affect the application	Application should have been installed	1. Click on app and open it. 2. Insert the charging pin in between running of the application and verify.	Application should run when inserting the charger. It will not affect the application	pass	High	**done**	**done**			

BIBLIOGRAPHY

(2018), G. (n.d.). *Gleam: Business Growth Platform. [online]* . Retrieved November 1, 2018, from https://gleam.io/

(2018), S. (n.d.). *Statista.com*. Retrieved November 2018, from https://www.statista.com/statistics/200855/favourite-smartphone-app-categories-by-share-of-smartphone-users/

(2018), S. (n.d.). *Statista.com*. Retrieved November 2018, from https://www.statista.com/statistics/270291/popular-categories-in-the-app-store/

(2018)., S. (n.d.). *Wishpond Software - 2018 Reviews, Pricing & Demo. [online]* . Retrieved November 1, 2018, from https://www.softwareadvice.com/crm/wishpond-profile/

(2018)., W. (n.d.). *Wishpond. [online]* . Retrieved November 1, 2018, from https://blog.wishpond.com/

[online], B. (. (n.d.). *Bloggingwizard.com. (2018). [online]*. Retrieved November 1, 2018, from https://bloggingwizard.com/social-media-contest-tools/

[online], B. (. (n.d.). *Bloggingwizard.com. (2018). [online]* . Retrieved November 1, 2018, from Available at: https://bloggingwizard.com/social-media-contest-tools/

2018, J. R. (n.d.). *Giftagram App: Gift Giving Made Easy | Send Gifts Online. [ONLINE]* . Retrieved November 1, 2018, from https://www.giftagram.com/

Android, D. (n.d.). *Developer Android*. Retrieved November 2018, from https://developer.android.com/training/basics/firstapp/

Anna. (n.d.). *The App Solutions*. Retrieved November 2018, from https://theappsolutions.com/blog/marketing/swot-for-mobile-app/

Basecamp. (n.d.). *Basecamp*. Retrieved November 2018, from https://basecamp.com/

Bullas, J. (. (n.d.). *The Top 5 Apps For Your Next Social Media Contest. [online] Jeffbullas's Blog.* . Retrieved November 2, 2018, from https://www.jeffbullas.com/top-five-social-contest-apps/

DeMuro, J. (n.d.). *Techradar.Pro*. Retrieved November 2018, from https://www.techradar.com/news/best-web-development-tool

Esplin, C. (2018, March 9). *Firebase Realtime DB vs Firestore*. Retrieved from YouTube: https://www.youtube.com/watch?v=TmXct7seeBY

FARR, C. (n.d.). *Venturabeat*. Retrieved November 2018, from https://venturebeat.com/2013/11/06/10-best-practices-from-top-coders-at-google-pinterest-more/

Firebase. (2017, October 3). *Firebase on Android - Tutorials Playlist*. Retrieved from YouTube: https://www.youtube.com/watch?v=kDZYIhNkQoM&list=PLl-K7zZEsYLmxfvI4Ds2Atko79iVvxlaq

Firestore, G. C. (2019, 03). *Cloud Firestore*. Retrieved from Database Products: https://cloud.google.com/firestore/

Fuenmayor, L. (. (n.d.). *Mejores Apps para regalos TOP Apps (iOS / Android).* . Retrieved November 2, 2018, from https://appaplicacionpara.com/regalos/

Google, (. (n.d.). *Birthday Gift List* . Retrieved November 3, 2018, from https://play.google.com/store/apps/details?id=it.xabaras.android.giftlist.birthday

GRAF1X. (n.d.). *Color Meaning and Psychology*. Retrieved from graf1x.com: https://graf1x.com/color-psychology-emotion-meaning-poster/

Jackson, B. (n.d.). *KeyCDN*. Retrieved November 2018, from https://www.keycdn.com/blog/web-development-tools

MarvelApp. (n.d.). Retrieved November 25, 2018, from https://marvelapp.com/

Mono. (2017, November 29). *Offline App for iOS App with Firestore*. Retrieved from Medium: https://medium.com/@mono0926/firestore-offline-edee47ee72cc

Mono. (2018, May 1). *Effective Cloud Firestore — Part 1*. Retrieved from Medium: https://medium.com/@mono0926/firestore1-5d04cdb683bc

React Bootstrap. (n.d.). Retrieved 3 15, 2019, from https://react-bootstrap.github.io/components/modal/

React-semantic-ui-tutorial. (n.d.). Retrieved 3 14, 2019, from https://www.robinwieruch.de/react-semantic-ui-tutorial/

REDUX. (n.d.). *Getting Started with Redux*. Retrieved from redux.js.org: https://redux.js.org/introduction/getting-started

Sierra, F. G. (2018, June 18). *Firebase Android Series: Firestore*. Retrieved from ProAndroidDev: https://proandroiddev.com/firebase-android-series-firestore-17e8951c574e

Sierra, F. G. (2018, May 19). *Working with Firestore: Building a simple database model*. Retrieved from ProAndroidDev: https://proandroiddev.com/working-with-firestore-building-a-simple-database-model-79a5ce2692cb

Team, M. T. (n.d.). *Mindtools.com*. Retrieved November 2018, from https://www.mindtools.com/pages/article/newTMC_05.htm

Usability.gov. (n.d.). *User Interface Design Basics*. Retrieved from www.usability.gov: https://www.usability.gov/what-and-why/user-interface-design.html

Varshneya, R. (n.d.). *Entrepeneur*. Retrieved November 2018, from https://www.entrepreneur.com/amphtml/231145

Visitors, W. (. (n.d.). *Wishpond Introduces Three Powerful New Tools to Convert Website Visitors. [online] PRWeb.* . Retrieved November 2, 2018, from http://www.prweb.com/releases/2017/08/prweb14594231.htm#!

Wikipedia, (. (n.d.). *Wikipedia*. Retrieved November 2018, from Wikipedia, (2018) Android Studio