National College of Ireland

BSC (HONOURS) IN COMPUTING

2016/2017

Bobo

Final Year Project: Technical Report

10th May 2017

YOUCEF O'CONNOR

INTERNET OF THINGS

X13114557

X13114557@student.ncirl.ie

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: **Youcef O'Connor**
Student ID: **X13114557**
Supervisor: **Dr Dominic Carr**

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature:____

Date:____10/05/2017_________

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

What constitutes plagiarism or cheating?

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a particular piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks

and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

Penalties for Plagiarism

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced

- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and
- the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.

ABSTRACT

The goal of the Bobo app was to develop an Android application that will help make taxi usage safer. The app will be used to identify whether a taxi is registered or not by entering the registration plate number or the license number. Once the taxi is validated and the user decides to use it, the user can send their contacts the taxi driver details and the pickup location. The pickup location is generated automatically by using Google Play Services and Google Maps to produce the current address of where the user sent the message. This information is helpful in the case of an emergency and if the police will need to know the last location of the user and last person who was in contact with him/her.

This application is using Firebase to store taxi and user data and runs part of the authentication procedure. WhatsApp is used to send the taxi driver details and the pickup location to the users WhatsApp contacts.

Acknowledgements

This work would not have been possible if it wasn't for the help and the great levels of support from my academic supervisor Dr Dominic Carr. Your help and advice throughout my final year project has been greatly appreciated.

To my mother, thank you for all the advice, support, patients and for feeding me these last four years.

To my family, thank you for supporting me through these difficult years.

To my IKEA manager, Desmond Shelly and team leaders Seamus McGann, Sean Cushen and Jenny O'Neill. Thank you for your support, understanding and giving me the time off that was needed to get me through these last few difficult months.

Table of Contents

Introduction

1.1 Background

The concept of this app was developed when a situation personal to myself was revealed. The app concept would be constructed in a way that would allow users of the app to feel safe whilst taking a taxi as a mode of transport.

I began starting to think of different scenarios of what could go wrong and what I could have done to prevent it. I came to the realisation that if I am in a taxi with a driver that plans to do me harm, then the driver would have assumed I have a phone and would go to use at the first sign of trouble. What would stop a driver from randomly pulling over, demands my mobile phone by threatening and then continues to drive? There would be no way for me to contact anyone. And then I would be left wishing I have told someone the taxi number sooner, when I had the chance.

That's when the idea hit me! What if there was an app that I can use to check if the taxi is registered and let my friends and family know the taxi driver details and pickup location all before or at the start of the taxi journey. This was just an idea at the time but now, two years later it will be in development. The name Bobo has no meaning, it was chosen simply because it sounds good and better than something like "The Safety Travel app".

Mobile applications are great help when looking for currency rates, booking flights or information on a location one would like to travel to and that is all you will get when you are looking up traveling apps on Google Play and Apple's App Store. There is one app found through Google search that helps to make traveling safer by letting a user's contact track their movements. It's call Companion (Companion, 2017).

There is an app in Ireland that lets the user to check is a taxi driver is registered or not but that is it. It will let the user inform friends and family. I was also surprised to see that the big taxi apps like Uber do not have this feature. This shows that user safety is not a primary goal for apps on the traveling app market.

During the Bobo app development, a new app called mytaxi has been launched in Ireland that has a feature that allows the user to share their taxi journey, but it only works if the user is using mytaxi taxis (GmbH, 2017). This shows that the Bobo idea is in demand.

1.2 Aims

The aim of this project is to create a fully functioning mobile application that will focus on making traveling safer for its user. This app will be developed for Android portable devices, mobile, tablet and smartwatch (Android Wear).

The main goals of this project are:

1. Use taxi verification feature to make taxi usage safer by allowing the user to enter the taxi license or registration number and then returning a message indicating whether the driver is registered or not.

2. The user will be able to message their friends and family the taxi driver details and the location of the pickup automatically.

3. The app will automatically send an email to the user’s emergency contacts if the user is under eighteen.

4. Try to get real taxi data from Transport Ireland

1.3 Description of technologies used

The Bobo app has different types of functions and as a result the app requires different types of technologies. For Example, the login function uses Firebase Authentication and the function for getting the current location uses Google Play Services. In this section there is a brief description on each type technology used. How Bobo uses these technologies will be explained in the System Architecture and Implementation sections of this report.

Android

Android is Google's Linux based mobile operating system (OS). It powers a lot of devices such as tablets, phones, smart TV's and smart watches (Karch, 2017).

Android Studio

Android Studio is an integrated development environment (IDE) that has been developed specifically for Android application development and requires the programming language to be Java (Eye, Tek). Alternatives to using the Android Studio IDE are IntelliJ IDE and Eclipse IDE. The reason why Android Studio was chosen over the other IDE's was because it is primarily focused on Android development and is ready to be used as soon as it is installed. Unlike the other IDE's where you will have to install Android software development kits (SDK) separately.

Firebase

Firebase is an online service that lets you run, test, store and deploy mobile application. In other words it is a mobile platform that has many features to suit development and business needs. The features used for the Bobo App are:

- Realtime Database: This is a NOSQL (Not Only SQL) database that lets users send and receive data at rapid speeds. This database only works with text.
- Authentication: Is used to make login and signup safe, fast and easy to apply to applications.
- Analytics: Allows businesses to monitor the usage of their applications and helps them with making

WhatsApp

WhatsApp is a social media messaging mobile application that allows users to send data to one another through the internet (WhatsApp Inc., 2017). This can be done in a one-to-one or one-to-many relationships (called group chats). WhatsApp can be used for sending text messages, images, video or audio. And it can make online phone calls.

Google Maps

Google Maps is a web service by Google that gives users information about locations and regions around the world (Margaret Rouse, 2017). It offers different types of views for many locations. These views are:

- Satellite view gives a geographical birds eye view of an area.
- Aerial view is showing the locations as if they are on a map, displaying features in different colours. For Example, rivers are blue, fields are green and roads are white. This is to help the user tell the difference between different types of terrain.
- Street view lets the user see an area in panoramic view at a height that gives the impression they are standing on the street themselves. This is only available in some cities.

System

2.1 Introduction

This sections purpose is to describe what the Bobo app needed and what was done throughout the development.

1.4 Requirements Definition

The user will want to feel safe while traveling and the requirements of the Bobo app will help do this. The two requirements that will make this happen for the user will be the Verification and Contacts requirements that will check if the taxi is registered and then inform their contacts of the taxi they are using.

1.4.1 Functional requirements

1. Signup – lets the user create an account by allowing them to enter a valid email address and password.
2. Login – this is the primary security of the user's details. The user cannot access their account if they don't have the details that were added when creating an account in the signup.
3. Validation – Lets the user know if the entered number is registered or not.
4. Contacts – The user can send their journey details with their WhatsApp contacts.
5. Contacts (Arrived): Let the users contacts know that the user got to his/her destination safely.
6. History – Allows the user to view their previously used taxi's in the History screen.
7. Smartwatch – This allows the user to receive the Bobo message to their watch

1.4.1.1 Use Case Model

The use case model is used to give a visualization of how all the requirements (oval shapes inside the box) work together with the services and contacts. The arrows indicate the flow of data. Each requirement is explained in more depth in the next sections. You can click on the number that represents a requirement if you wish to go straight to that particular requirement.

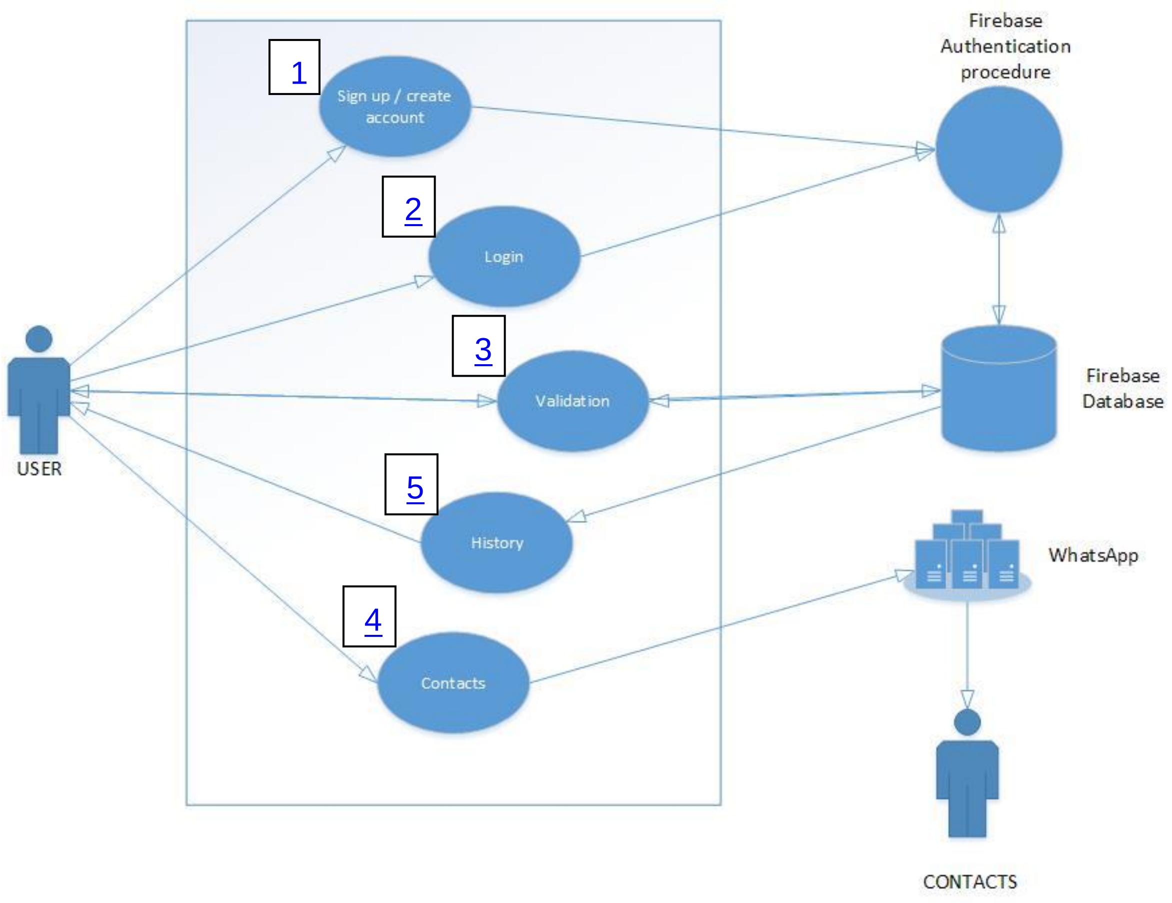


Figure 1: Use Case Model

1.4.1.2 Requirement 1 - Signup

Description & Priority

Signing up is necessary to start the application as there will be no way to tell the difference between the users without it. The user will also be required to fill in a form during the signup process to give extra details. For Example, first and last name. Some details entered by the user in this requirement will be needed to login.

Use Case

Scope

The scope of this use case is to demonstrate how the user can sign up or create an account on Bobo.

Use Case Diagram

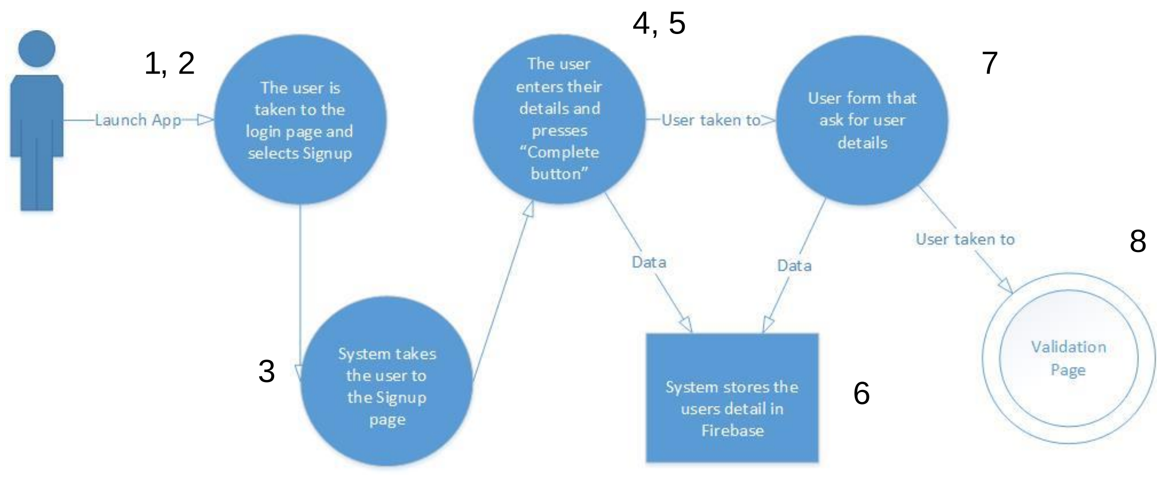

1, 2
Launch App
The user is taken to the login page and selects Signup
3
System takes the user to the Signup page
4, 5
The user enters their details and presses "Complete button"
User taken to
7
User form that ask for user details
Data
Data
6
System stores the users detail in Firebase
User taken to
8
Validation Page

Figure 2: Signup Use Case

Flow Description

Precondition

The user will need to be connected to internet and download the app from Google play to their android device.

Activation

This use case starts when the user choses to launch the Bobo application.

Main flow

1. User will be taken to the login page
2. The user selects the Signup link on the login screen
3. System takes user to the Signup page
4. The user enters their signup details.
5. Once entered the, the user will press the Register button.
6. The system stores the user details and takes the user to the user form screen where the user is asked to enter more details.
7. Once entered the, the user will press the Continue button.
8. The system stores the user details and takes the user to the user Validation page.

Alternate flow

A1: Unfilled fields.

1. System displays message pointing out that specific needs to filled.
2. The user enters / re-inputs their details.
3. The use case continues at position 5 of the main flow

Termination

The user can exit their account by signing out or closes the application.

Post condition

The user now has an account on the Bobo app and is now on Login page

1.4.1.3 Requirement 2 - Login

Description & Priority

The login of an app is very important, as it is the apps primary security of the users account and private details. The user must create an account by signing up to use the login function.

Use Case

Scope

The scope of this use case is to demonstrate how the user can start / launch the Bobo application.

Use Case Diagram

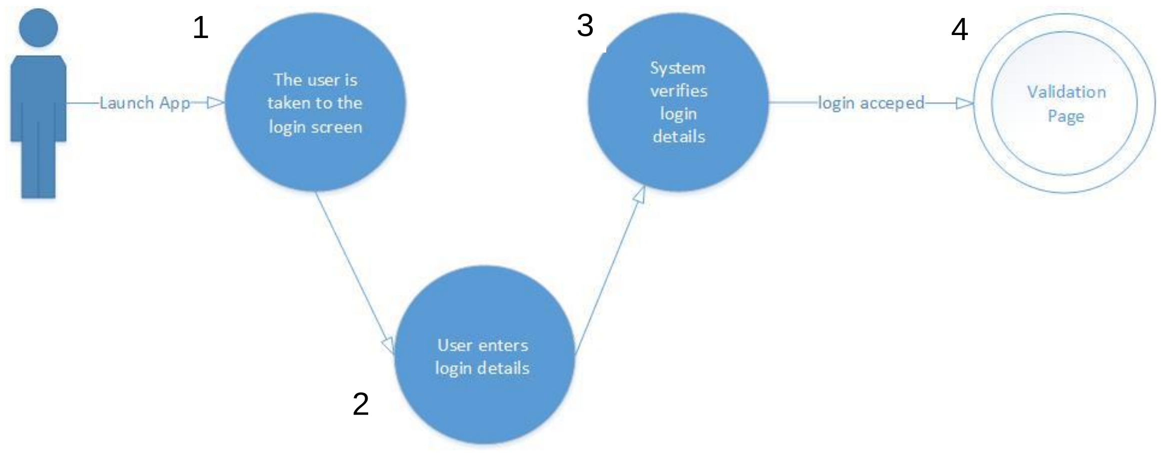

1
3
4
Launch App
The user is taken to the login screen
System verifies login details
login acceped
Validation Page
User enters login details
2

Figure 3: Login Use Case

Flow Description

Precondition

The user will need to be connected to internet and have signed up an account.

Activation

This use case starts when the user choses to launch the Bobo application.

Main flow

1. User will be taken to the login screen
2. Here the user will enter login details
3. System verifies login details
4. Once successfully logged in, the user will be taken to the verification page

Alternate flow

A1: Username or password is incorrect

1. System displays "invalid email or password entered" message.
2. The user re-enters their login details.
3. The use case continues at position 3 of the main flow

A2: Forgotten Password?

1. System displays wrong "username or password entered" message.
2. The user selects the Forgotten Password button
3. The system asks user to enter their email address.
4. The system sends the user an email with a link to change password.
5. The user changes their password.
6. The use case continues at position 1 of the main flow

Termination

The system goes to the main page (verification page). User can delete their account in the account page.

Post condition

The user is now logged into the Bobo app and is now on the Verification page.

1.4.1.4 Requirement 3 – Validation

Description & Priority

This requirement shows how the user can validate a taxi license and how the validation procedure works.

<u>Use Case</u>

Scope

The Scope of this requirement is to describe how the validation requirement works.

Use Case Diagram

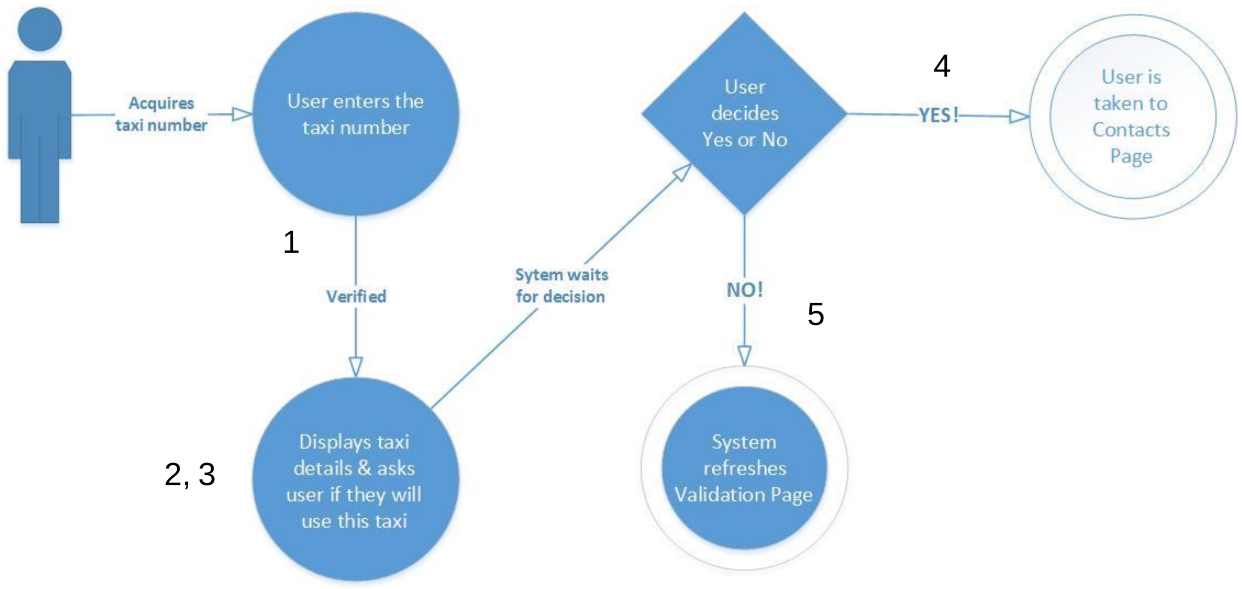

Acquires taxi number
User enters the taxi number
1
Verified
Displays taxi details & asks user if they will use this taxi
2, 3
Sytem waits for decision
User decides Yes or No
4
YES!
User is taken to Contacts Page
NO!
5
System refreshes Validation Page

Figure 4: Validation Use Case

<u>Flow Description</u>

Precondition

The user is connected to the internet, has created an account and has successfully logged in

Activation

This use case starts when a user enters the taxis registration / license number and hits the Verify button.

Main flow

1. The system verifies the taxi number.
2. The system displays that the taxi is registered and the driver's details
3. The system asks the user is they will use this taxi, Yes or No?
4. If the user chooses YES, the system will take the user to the contacts page.
5. If the user chooses NO, the system will take the user to the Validation page.

Alternate flow

A1: Yes! To a non-registered taxi number

1. System displays "number not is not registered! Are you going to use this unregistered taxi Yes or No?" message.
2. The user chooses Yes!
3. The use case continues at position 4 of the main flow

Termination

The user chooses not to use an unregistered taxi. This will stop the validation procedure and the user will remain on the Validation page. Or the user closes the application.

Post condition

The user is now on the Contacts Page.

1.4.1.5 Requirement 4 – Contacts

Description & Priority

This requirement describes how the Contacts procedure works by sending an email to the user's contacts of choice containing taxi driver details, the pickup location and the user's name.

Use Case

Scope

The scope of this use case is to demonstrate how the user can let their contacts know what taxi they are using and where they were picked up. WhatsApp is used to send the details to the users WhatsApp contacts

Use Case Diagram

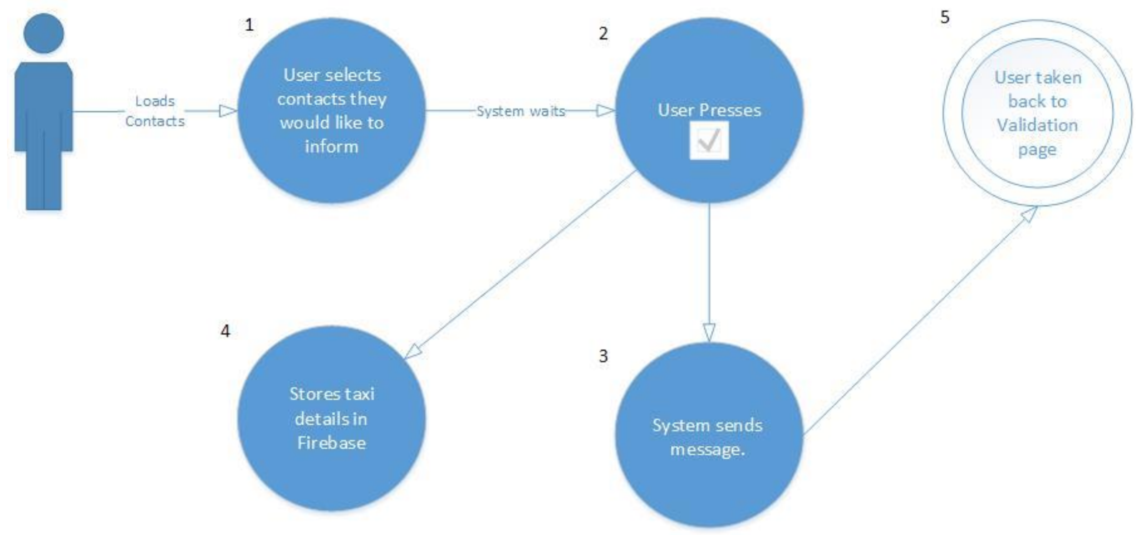

Loads Contacts
1
User selects contacts they would like to inform
System waits
2
User Presses
5
User taken back to Validation page
4
Stores taxi details in Firebase
3
System sends message.

Figure 5: Contacts Use Case

<u>Flow Description</u>

1 **Precondition**

The user will need to be connected to internet and have said Yes to using a taxi, which will let the system take the user to the Contacts page on WhatsApp. The user will also need a WhatsApp app installed on their device and have an active account.

Activation

This use case starts when a user selects the “YES” button.

Main flow

1. The user selects the contacts they would like to inform.

2. The user presses the “On the way” button
3. The system sends an email containing taxi driver and location details to selected contacts.
4. The system stores taxi details.
5. The user is sent to the contacts <arrived> page.

Alternate flow

A1: Under eighteen skipping

1. The user is under the age eighteen.
2. The system sends an email containing taxi driver and location details to the user’s parent/guardians
3. The use case continues at position 4 of the main flow

Termination

This process is terminated closes the application.

Post condition

The system waits on the Validation page.

1.4.1.6 Requirement 5 – Contacts (Arrived)

Description & Priority

This requirement describes how the “Arrived” procedure works by sending an email to the user’s contacts of choice containing taxi driver details and the drop off location. If the user is under eighteen, then the system will automatically add his/her parents/guardians to the list of contacts that the user would like to inform.

Use Case

Scope

The scope of this use case is to demonstrate how the user can let their contacts know they have arrived safely and where they were dropped off. The contacts were added to the contact list by the user when signing up. Users can also add or remove contacts from the view Contacts page.

Use Case Diagram

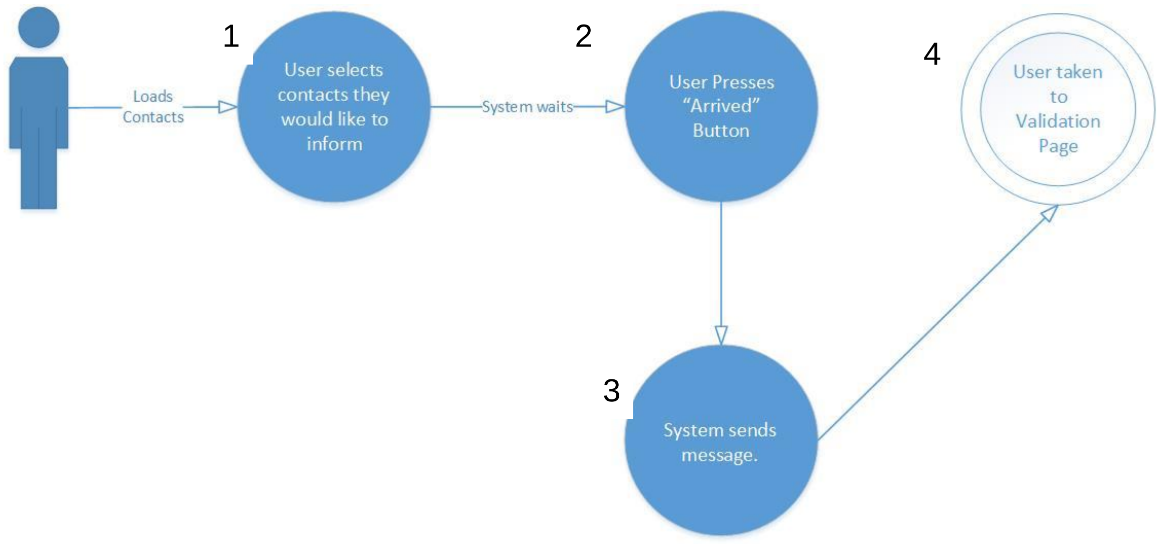

1
2
4
3
Loads Contacts
User selects contacts they would like to inform
System waits
User Presses "Arrived" Button
User taken to Validation Page
System sends message.

Figure 6: Contacts <Arrived> Use Case

Flow Description

1 **Precondition**

The user will need to be connected to internet and have pressed the On the way button on the contacts page, which will let the system take the user to the Contacts page.

Activation

This use case starts when a user selects the "Arrived" button.

Main flow

1. The user selects the contacts they would like to inform.
2. The user presses the "Arrived" button
3. The system sends an email containing an automated message informing them that the user has arrived safe along with the drop off location.
4. The user is then sent back to the Validation page.

Alternate flow

A1: Skip

1. The user does not want to inform contacts and presses the "Skip" button

2. The use case continues at position 6 of the main flow

A2: Under eighteen

1. The system automatically selects the parent/guardian contacts
2. The use case continues at position 1 of the main flow.

A3: Under eighteen skipping

1 The user does not want to inform contacts and presses the “Skip” button

5. The system sends an email containing an automated message informing them that the user has arrived safe along with the drop off location.
6. The use case continues at position 6 of the main flow

Termination

This process is terminated if the user decides to close the Bobo application.

Post condition

The system takes the user back to the Validation page.

1.4.1.7 Requirement 5 - History

<u>Description & Priority</u>

This requirement will describe how the history feature will work and how the user can use it to view previously used taxis.

<u>Use Case</u>

Scope

The scope of this use case is to demonstrate how the user can view their taxi history.

Use Case Diagram

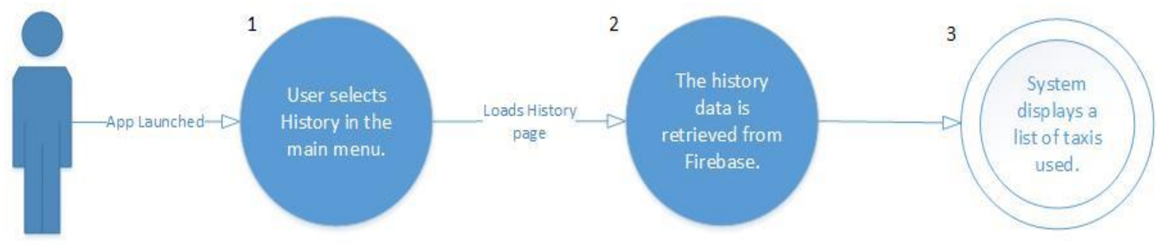

App Launched
1
User selects History in the main menu.
Loads History page
2
The history data is retrieved from Firebase.
3
System displays a list of taxis used.

Figure 7: History Use Case

Flow Description

Precondition

The user will need to be connected to internet and have successfully logged in.

Activation

The system will display a list of taxi numbers that the user has used with times and dates.

Main flow

1. The user Selects the History option on the main menu.
2. The system retrieves the taxi history data from the database
3. The system displays a list of taxi driver's numbers with time and dates.

Alternate flow

A1: No Taxis!

1. The system displays an empty screen.

Termination

This process is terminated if the user choses to go back to the Validation page or closes the Bobo application.

Post condition

The system goes into a wait state

1.4.1.8 Requirement 6 – Smartwatch

Description & Priority

This requirement will describe how the user's contacts can receive a Bobo notification on their Android Wear smartwatch. The notification will display the users taxi driver details and pickup or drop off location.

Use Case

Scope

The scope of this use case is to demonstrate how the user contacts can receive a Bobo notification on their Android Wear smartwatch.

Use Case Diagram

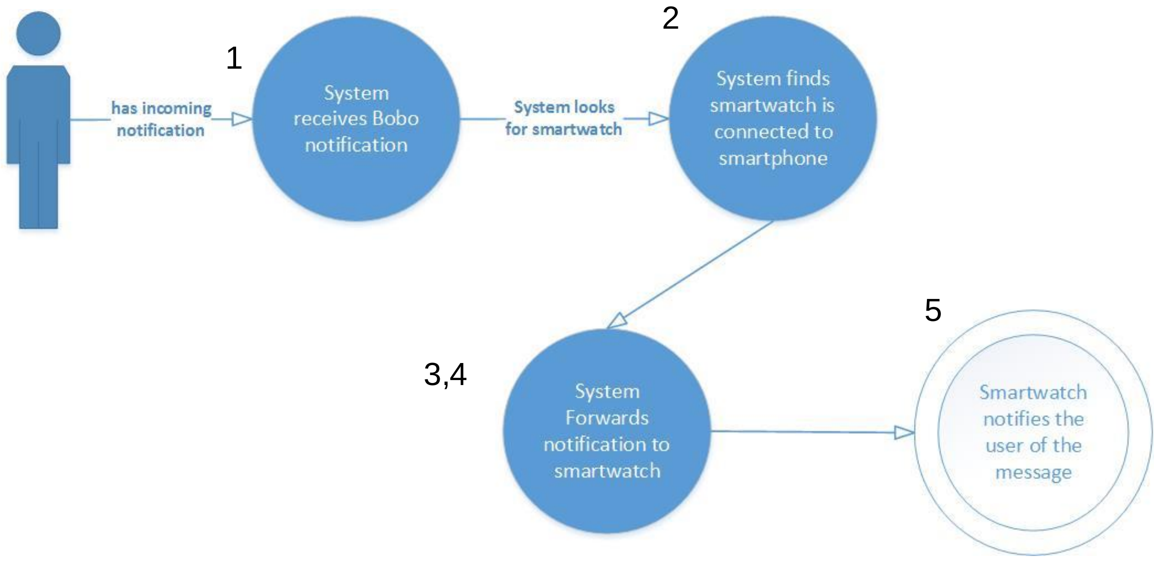

1
2
3,4
5
has incoming notification
System receives Bobo notification
System looks for smartwatch
System finds smartwatch is connected to smartphone
System Forwards notification to smartwatch
Smartwatch notifies the user of the message

Figure 8: Smartwatch Use Case

Flow Description

Precondition

The users contacts will need their phone and smartwatch to be connected to internet.

Activation

This use case starts when a contact receives a Bobo notification via “On the way” or “Arrived” to a contact who has an Android Wear smartwatch.

Main flow

1. The system receives the notification on the smartphone
2. The system reads that the smartphone is paired to a smartwatch
3. The system reads that the smartwatch is within Bluetooth range
4. The system forwards the notification on to the smartwatch using Bluetooth
5. The system notifies the contact (User) on the smartphone and on the smartwatch.

Alternate flow

A1: Too far apart
1. The system reads that the smartwatch is out of the Bluetooth reach
2. The system checks to see if the smartwatch is connected to the internet
3. The system forwards the notification to the smartwatch through the internet.
4. The use case continues at position 5 of the main flow

Termination

The user’s smartwatch is out of Bluetooth range and is not connected to the internet.

Post condition

The system waits for the user to react to receiving the notification

1.4.2 Non-Functional Requirements

In this section we will be describing the functions that are not mentioned or are in the functional requirements.

Performance/Response time requirement

The Bobo application will be made with simplicity in mind, so the user can move quickly from one screen to the next. This means navigation amongst pages within the app will be easy and fluid. The speed of retrieving data will depend on the user's internet coverage and speed. Also, the app is using Firebase's Real-time database which has a really fast response time that can send data in milliseconds (Firebase Realtime Database, 2017).

Availability requirement

The Bobo app is not available on the Google Play Store at the moment, as it is not ready for launch. The reasons why the app isn't ready is mentioned in the evaluation.

Data requirement

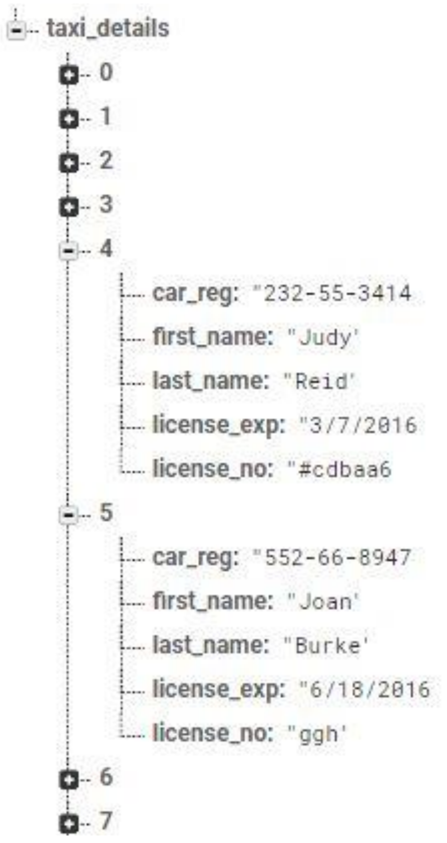
This app will be using Firebase's Real-time database to store the data. This database is a NOSQL database. NOSQL was developed to meet the demands for modern applications as relational databases are not designed to handle todays demand. The NOSQL provides a better performance and is more scalable than the relational database (mongoDB, 2017).

The Real-time database stores and sends data in a JSON (JavaScript Object Notation) format. JSON is made up of keys with values. The key has to be a name or sting type. With value you can store numbers, an object, an array or a string (Squarespace, 2017).

Unfortunately, the Bobo app is using mock data generated from mockaroo, as a result of not been able to retrieve real data. Phone calls were made to the Taxi regulators who told me to ring the Transport of Ireland. Transport of Ireland told me that givng out this data would be a breach of the data protection act.

There will be two NOSQL datasets. One called users, this is for user details and the other is called taxi_details, this for taxi driver details which will be used by the validation function.

Here is snippet of the taxi driver details in the database:



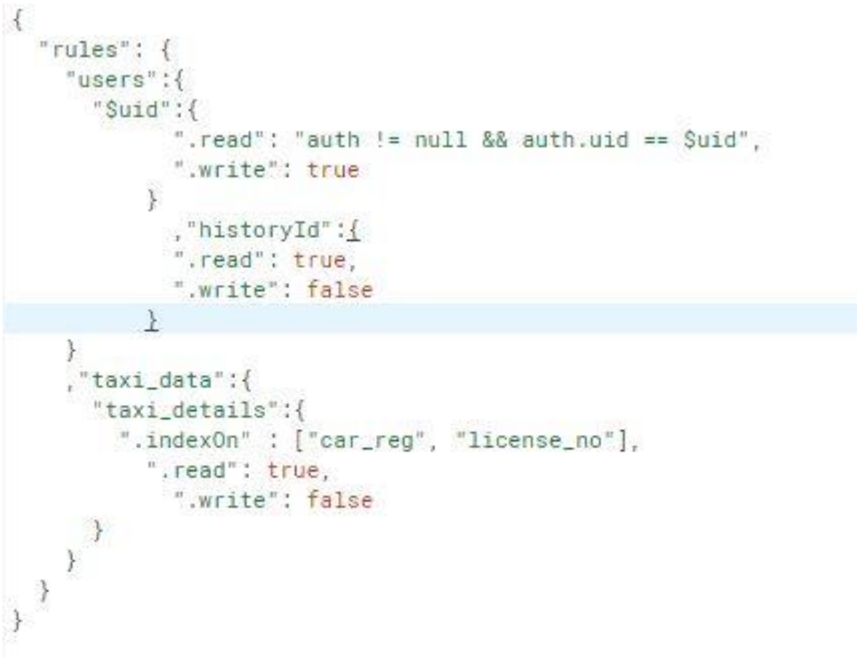
```
taxi_details
  0
  1
  2
  3
  4
    car_reg: "232-55-3414
    first_name: "Judy'
    last_name: "Reid'
    license_exp: "3/7/2016
    license_no: "#cdbaa6
  5
    car_reg: "552-66-8947
    first_name: "Joan'
    last_name: "Burke'
    license_exp: "6/18/2016
    license_no: "ggh'
  6
  7
```

Figure 9: taxi_data

Security requirement

The Bobo app focuses on safety for the user, not just when traveling but also in protecting their data. The user can only view the details of that user ID. This prevents a user from accessing other user’s data. This is done by setting rules in the Real-time database. In figure eight you can see that:

- Rules has datasets “users” and “taxi_data”.
- Within “users” it shows that “$uid” (user ID) is needed for the user to be able to read and write user data.
- And “historId” is needed by the user to view their history data.
- Within “taxi_data” there is “taxi_details”, and its rule says anyone can view the data but cannot write to it.



```
{
  "rules": {
    "users":{
      "$uid":{
            ".read": "auth != null && auth.uid == $uid",
            ".write": true
          }
            ,"historyId":{
            ".read": true,
            ".write": false
          }
      }
      ,"taxi_data":{
        "taxi_details":{
          ".indexOn" : ["car_reg", "license_no"],
            ".read": true,
              ".write": false
        }
      }
    }
}
```

Figure 10: Database rules

This app will also have a login page that will require the user’s details that only the user will know.

Reliability requirement

The connection reliability to the database and the speed of the data being transferred will depend on the signal strength and coverage of the user. Whatever plan the user is on, to get a more reliable connection they will need to contact their network provider.

Using the Real-time database optimises the apps offline use. What this means is that if the user loses connection after validating a taxi and then presses YES. The history data will be stored using a local cache on the device until the connection has being restored and the data will be sent to the database.

1.5 System Architecture

In this section, figure 9 is used to explain how the Bobo app works with all the services and devices. It also illustrates the direction of the data flow and that the devices are communicating wirelessly. The received devises are shown to have the Android and Apple logo, this means the received user can have an Android or Apple OS. This is one of the benefits of using WhatsApp. The Cloud Services show all the services used by the app and how they are used. Please note that the PC can run on any OS as long as it can access the Firebase website.

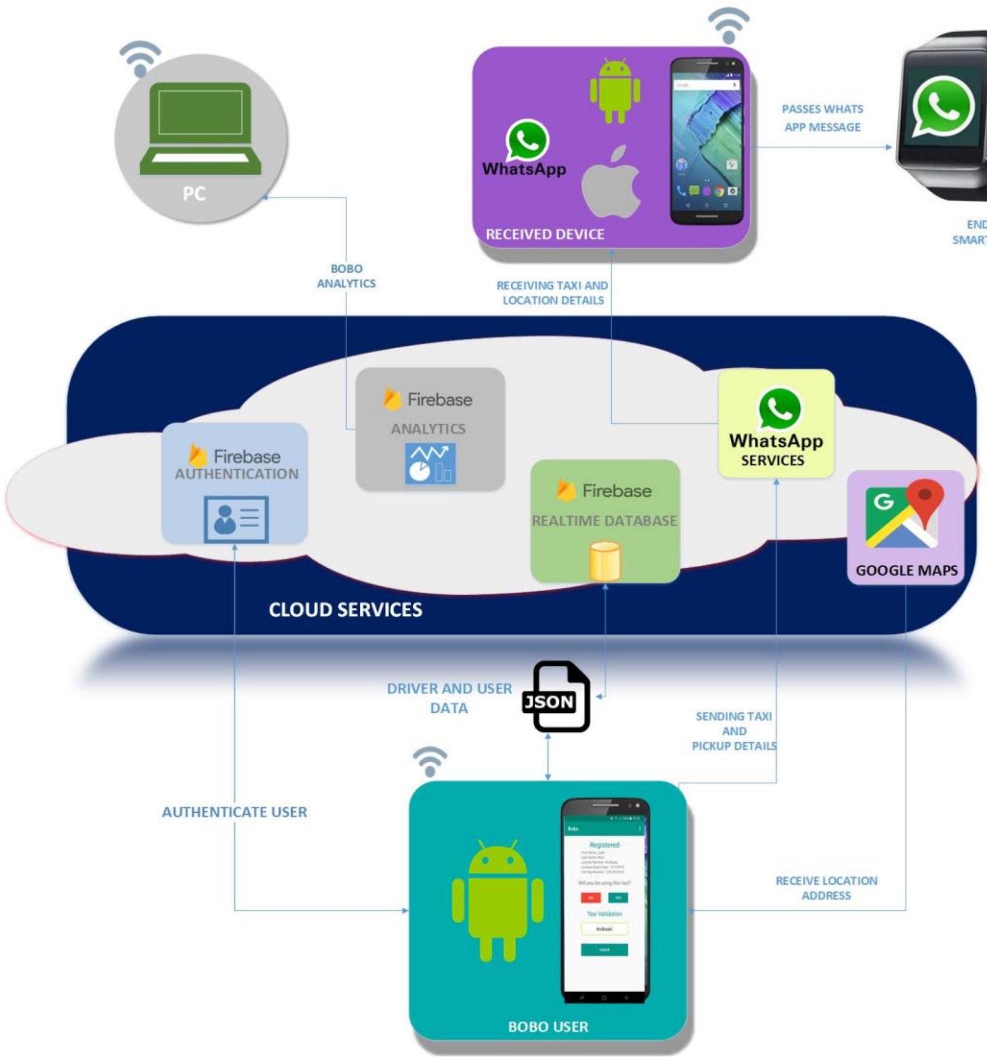


Figure 11: System architecture

1.6 Implementation

This section will explain how each function was implemented into the Bobo app with the aid of snippets containing code relevant to the function. There are 9 function implemented and they are:

- FIREBASE – How the Firebase features added to the app
- LOGIN – Explains how the user's details are authenticated for access
- SIGNUP – Describes how the user creates an account with Bobo
- USER FORM – How it gets additional information from the user and stores it in the Real-time database.
- VALIDATION – Explains how a taxi in checked and how the results is handled.
- CONTACTS – Describes how WhatsApp was implanted to use for contacts.
- PERMISSION – How the app asks the user for permission to use the location feature on their device
- LOCATION – Is the function that retrieves the user's current location.
- HISTORY – To display previously used taxi' for the user.

1.6.1 FIREBASE

In December 2016, it was decided that Firebase features will be used on this app after it was recommended by Lecturer Dr Dominic Carr. The features were researched and it was found to be very helpful with Android application development (Firebase, 2017).

To install features for the app, you only needed to copy and paste the library compile code given. There is a different library for each feature, so once the right libraries are added under dependencies in the build Gradle (app level), press Sync Now. This will install the SDK's (Software Development Kit's) needed to run the Firebase features. Firebase has great documentation that takes you step by step through the installation process.

The only issue I had with implementing Firebase, was the versioning issues of incompatible releases amongst Android OS, Android Studio and Firebase.

Whenever an issue like this occurs and a feature like authentication stops working, it is best to start looking for updates after checking the code, rather than Googling the specific problem. This method was useful in the Bobo development.

1.6.2 LOGIN

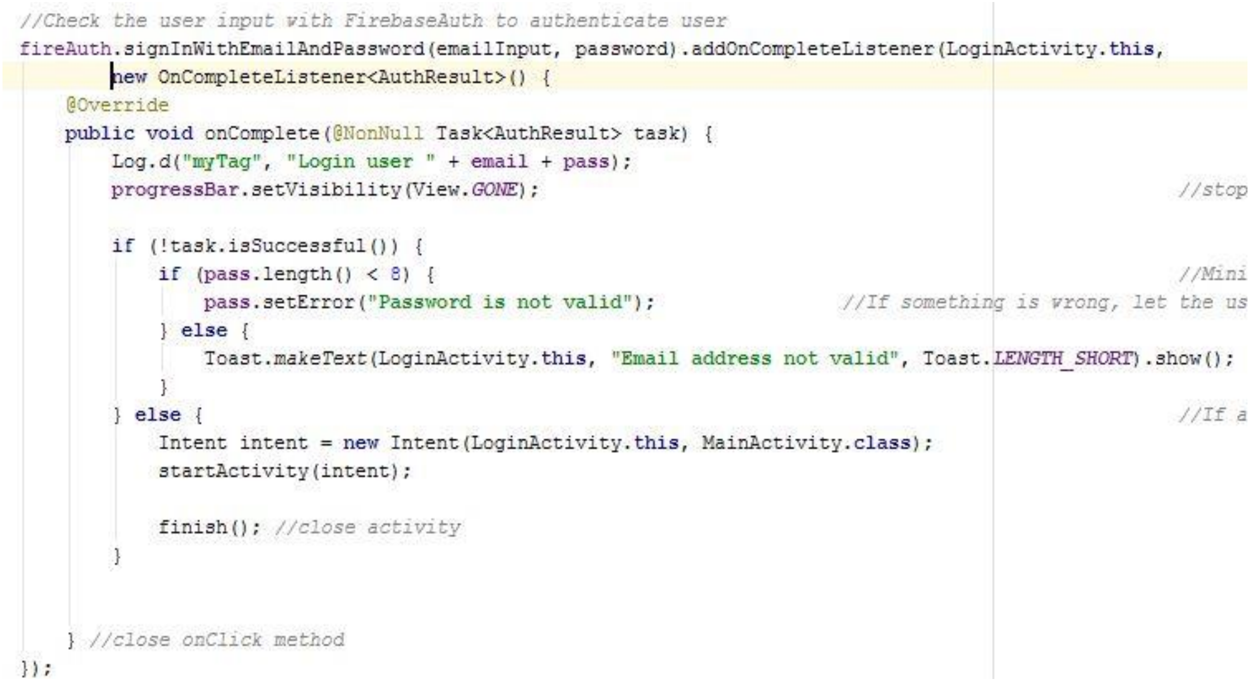
There are different approaches in Android when it comes to implementing a login function to an app. The first approach was using Firebase's AuthUI (Authentication User Interface). There wasn't much code needed for this approach and Firebase takes you through the steps. You can see the steps and code for the AuthUI on Firebase. It was believed at the time that this approach was a good idea as it saved a lot of time, but It was found later that it wasn't the best approach. Here is why:

1. Couldn't retrieve any user details, as this was needed for displaying the user name in the message that is sent to the users contacts.
2. It didn't look secure as it would sometimes at lunch still display the validation page even when the user is signed out.
3. It wasn't flexible and had no layout XML file (couldn't customise layout).

It was decided to code the login and signup functions without using UI librarys. This will bring more benefits to the Bobo apps security, flexibility and gives us the ability to run a user application form which we will go through later in the Application Form section 1.7.4. Firebase was still used for authentication, we were are just no longer using the authentication UI (authUI).

After it was decided what was needed, the xml layout was the first part of implementing the Login function. Once that was done the LoginActivity.Java class was created and coding began. The main part of this code would be the authentication method, also the most important part.

The authentication method in figure 9 uses a Firebase authentication instance (declared fireAuth). The signInWithEmailAndPasswod is used so that Firebase knows what type of authentication we are using and what type of input it is dealing with. This method also uses an on complete task. This means it will listen out for the authentication to complete. If authentication was a failure then a Toast message will notify the user “Password is not valid” or “Email address was not valid”. If authentication was a success, then open the Validation page.



```
//Check the user input with FirebaseAuth to authenticate user
fireAuth.signInWithEmailAndPassword(emailInput, password).addOnCompleteListener(LoginActivity.this,
        new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        Log.d("myTag", "Login user " + email + pass);
        progressBar.setVisibility(View.GONE);                                  //stop

        if (!task.isSuccessful()) {
            if (pass.length() < 8) {                                           //Mini
                pass.setError("Password is not valid");          //If something is wrong, let the us
            } else {
                Toast.makeText(LoginActivity.this, "Email address not valid", Toast.LENGTH_SHORT).show();
            }
        } else {                                                               //If a
            Intent intent = new Intent(LoginActivity.this, MainActivity.class);
            startActivity(intent);

            finish(); //close activity
        }


    } //close onClick method
});
```

Figure 12: Firebase Authenticate

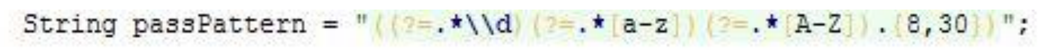
1.6.3 SIGNUP

The Signup function needs to take in the users details and make sure they are valid before sending them off to Firebase Authentication services.
The main features of the signup process are:

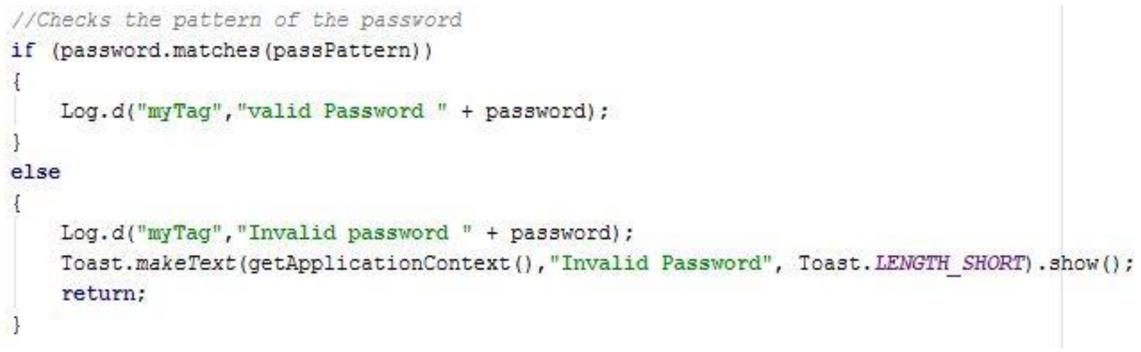
1. Making sure the password saved follows a certain pattern of having a minimum of eight characters, one uppercase letter, one lowercase letter and one number. The reason this was done, was to direct the users to make a more complex and secure password.

This was done with the aid of a regular expression (developer, 2017). The regular expression is used to make sure the password is using the right pattern. This is done by using Androids matcher method to compare or match the entered password with the regular expression.

Here are two snippets, the first one shows the regular expression and the second one shows how the regular expression and the password are matched.



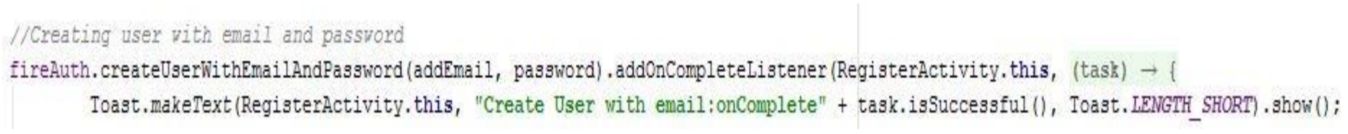
```
String passPattern = "((?=.*\\d)(?=.*[a-z])(?=.*[A-Z]).{8,30})";
```



```
//Checks the pattern of the password
if (password.matches(passPattern))
{
    Log.d("myTag","valid Password " + password);
}
else
{
    Log.d("myTag","Invalid password " + password);
    Toast.makeText(getApplicationContext(),"Invalid Password", Toast.LENGTH_SHORT).show();
    return;
}
```

Figure 13: Matching password with Regular Expression

2. Using Firebase authentication method just like the Login function but instead of signInWithEmailAndPasswod, the registration function uses createUserWithEmailAndPassword. This checks to make sure the email address is valid and creates the user account.



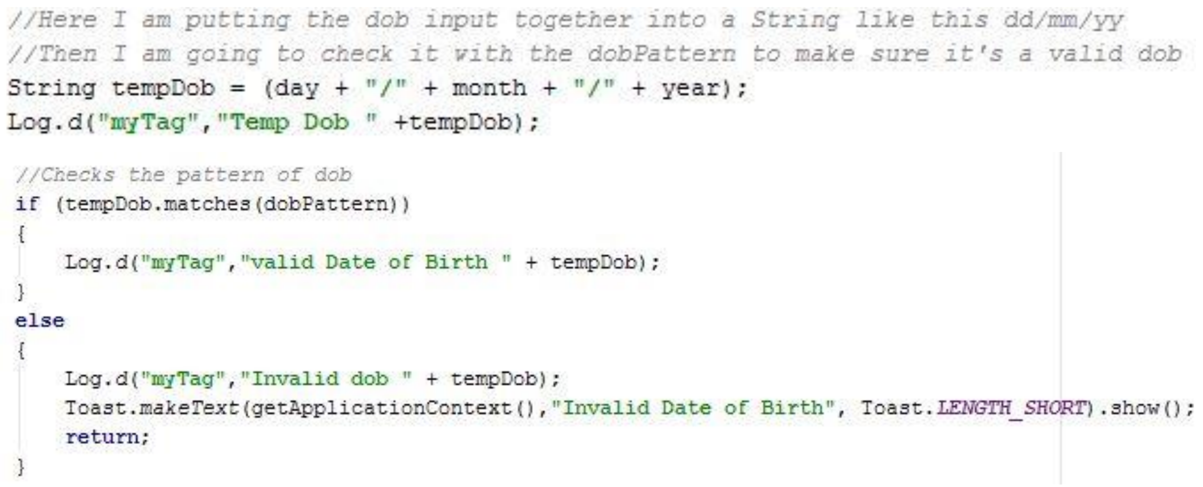
```
//Creating user with email and password
fireAuth.createUserWithEmailAndPassword(addEmail, password).addOnCompleteListener(RegisterActivity.this, (task) → {
      Toast.makeText(RegisterActivity.this, "Create User with email:onComplete" + task.isSuccessful(), Toast.LENGTH_SHORT).show();
```

Figure 14: create user with email and password

1.6.4 USER FORM

This was a function that I had added to the registration procedure so the app can get more data about the user. The main parts of the User Form activity was getting the date of birth to match the regular expression and sending the data to the firebase Database.

1. The date of birth was being taken from three different edit text field and then put together to create a date pattern dd/mm/yyyy. The reason the date of birth had to be put together like this, was so it can be matched with the date of birth Regular Expression.

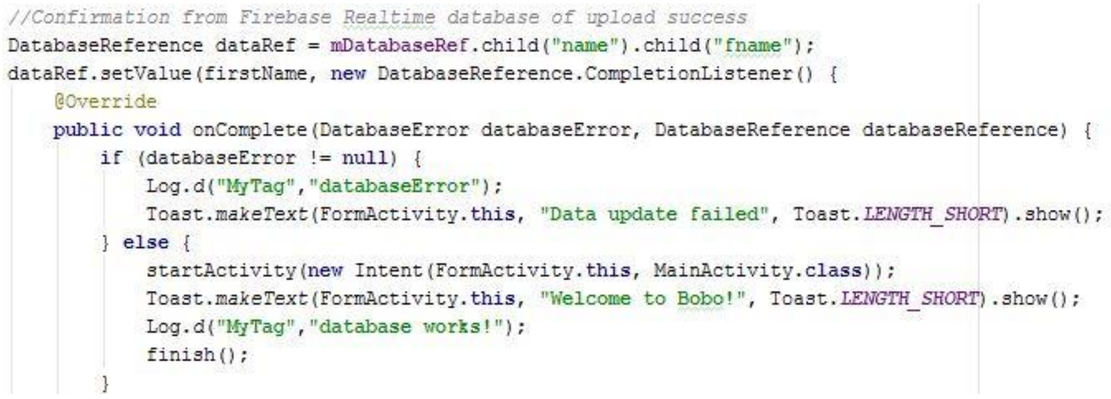


```java
//Here I am putting the dob input together into a String like this dd/mm/yy
//Then I am going to check it with the dobPattern to make sure it's a valid dob
String tempDob = (day + "/" + month + "/" + year);
Log.d("myTag","Temp Dob " +tempDob);
```

```java
//Checks the pattern of dob
if (tempDob.matches(dobPattern))
{
    Log.d("myTag","valid Date of Birth " + tempDob);
}
else
{
    Log.d("myTag","Invalid dob " + tempDob);
    Toast.makeText(getApplicationContext(),"Invalid Date of Birth", Toast.LENGTH_SHORT).show();
    return;
}
```

Figure 15: Matching dob and Regular Expression

2. There were a few stages to implement in order to send the user's details to the database. First get the Firebase Authenticate and Firebase Database instances. Next stage was to get the users ID and use it with the database instance to create a reference that represents where the data is going to be stored. Once the "user" node was referenced, three children nodes were created for contacts, dob and name. A **onComplete** method was the last stage, it sends the users first name as a data reference. The **onComplete** method lets the user know whether it was a successful upload or was there an error.



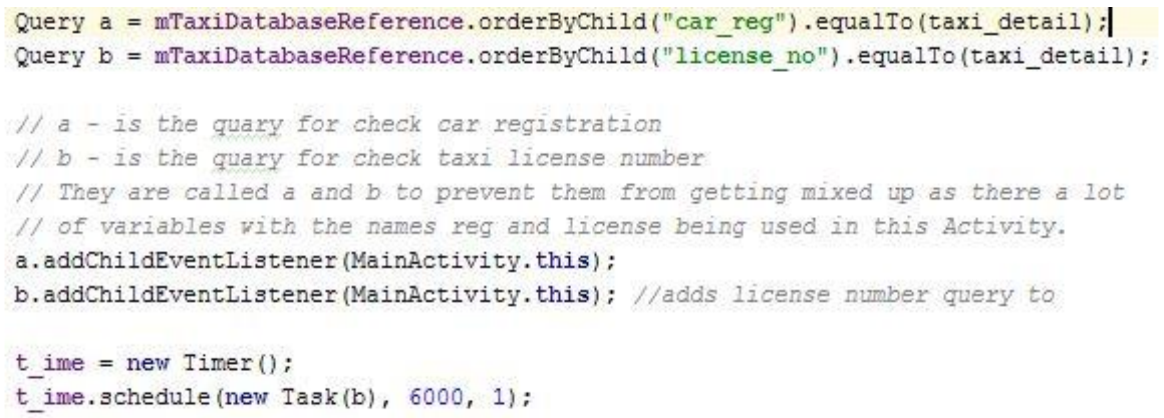
```java
//Confirmation from Firebase Realtime database of upload success
DatabaseReference dataRef = mDatabaseRef.child("name").child("fname");
dataRef.setValue(firstName, new DatabaseReference.CompletionListener() {
    @Override
    public void onComplete(DatabaseError databaseError, DatabaseReference databaseReference) {
        if (databaseError != null) {
            Log.d("MyTag","databaseError");
            Toast.makeText(FormActivity.this, "Data update failed", Toast.LENGTH_SHORT).show();
        } else {
            startActivity(new Intent(FormActivity.this, MainActivity.class));
            Toast.makeText(FormActivity.this, "Welcome to Bobo!", Toast.LENGTH_SHORT).show();
            Log.d("MyTag","database works!");
            finish();
        }
```

Figure 16: On complete task

1.6.5 VALIDATION

The Validation function is the most important part of the Bobo app and it is the most complex. The Validation activity is the main activity that acts as the center of the whole app, as seen in figure 9. The main parts of this class are:

1. The validation function is triggered when the user enters a license number or car registration number and then presses the check button. Two queries then run to check the child nodes of “taxi_details”. Query “a” checks for the car_reg node and query b checks for the “license_no” node. The reason there are two queries is to allow the user to enter the license number or the car registration number to validate the taxi.
   Each query has its own **addChildEventListener** that will pick up the data that the queries have found (Firebase, ChildEventListener, 2017).

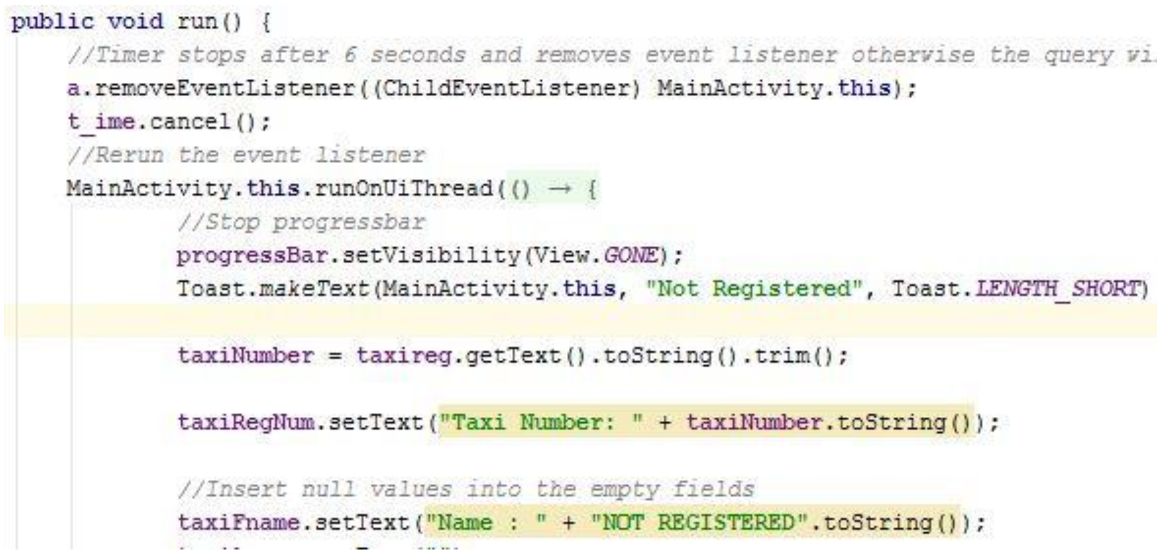


```
Query a = mTaxiDatabaseReference.orderByChild("car_reg").equalTo(taxi_detail);
Query b = mTaxiDatabaseReference.orderByChild("license_no").equalTo(taxi_detail);

// a - is the quary for check car registration
// b - is the quary for check taxi license number
// They are called a and b to prevent them from getting mixed up as there a lot
// of variables with the names reg and license being used in this Activity.
a.addChildEventListener(MainActivity.this);
b.addChildEventListener(MainActivity.this); //adds license number query to

t_ime = new Timer();
t_ime.schedule(new Task(b), 6000, 1);
```

Figure 17: Validation queries, add child event listeners, timer

Notice in the code snippet above that there’s a timer at the bottom. This timer starts when the button is clicked. It then creates a Task that will run when the timer is up in six seconds. The reason for this is to stop the add child event listeners from running constantly if the number entered isn’t in the database. The listeners will keep looking for something that isn’t there so they have been given six seconds to find the data, which is a lot of time for a real-time database. Once the six seconds are up the Task will activate the run method that will cancel the child event listeners and then start them again without the queries.
This method also displays the “NOT REGISTERED’ message to the user along with the number entered and the option buttons.

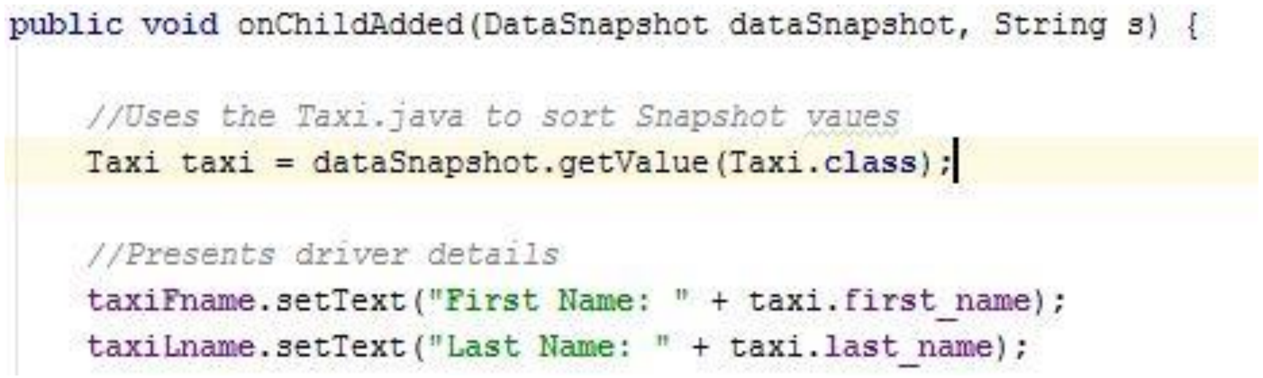
Here is a snippet of the run method:



```
public void run() {
    //Timer stops after 6 seconds and removes event listener otherwise the query vi
    a.removeEventListener((ChildEventListener) MainActivity.this);
    t_ime.cancel();
    //Rerun the event listener
    MainActivity.this.runOnUiThread(() → {
            //Stop progressbar
            progressBar.setVisibility(View.GONE);
            Toast.makeText(MainActivity.this, "Not Registered", Toast.LENGTH_SHORT)

            taxiNumber = taxireg.getText().toString().trim();

            taxiRegNum.setText("Taxi Number: " + taxiNumber.toString());

            //Insert null values into the empty fields
            taxiFname.setText("Name : " + "NOT REGISTERED".toString());
```

Figure 18: Run method

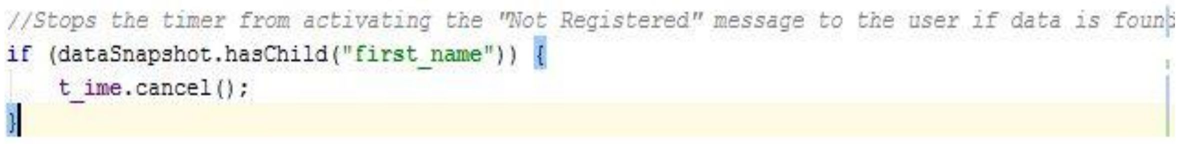
If the data was found then the **addChildEventListener** will store the data temporary as a snapshot. This will activate the **onChildAdded** method and then the driver be sent to the Taxi Java class to be sorted. The data can then be retrieved from the Taxi Java class so it can be set to text and displayed to the user (Read and Write Data on Android, 2017). The driver details are also declared in this method so they are ready to be stored in History. In the snippet below shows how data is retrieved from the Taxi class, which then set to text .



```
public void onChildAdded(DataSnapshot dataSnapshot, String s) {

    //Uses the Taxi.java to sort Snapshot vaues
    Taxi taxi = dataSnapshot.getValue(Taxi.class);

    //Presents driver details
    taxiFname.setText("First Name: " + taxi.first_name);
    taxiLname.setText("Last Name: " + taxi.last_name);
```

Figure 19: On child added method

Here is small if statement from the **onChildAdded** method. This if statement runs when there's a driver's first name. It will cancel the timer. In other words, if data has been found stop the timer displaying the not registered message.

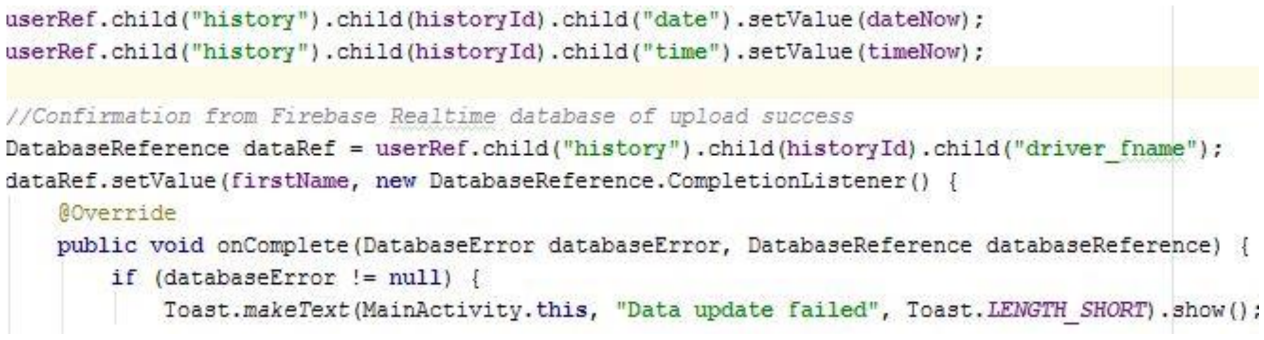


```
//Stops the timer from activating the "Not Registered" message to the user if data is foun
if (dataSnapshot.hasChild("first_name")) {
    t_ime.cancel();
}
```

Figure 20: Cancel Timer

There are two YES buttons in this app, the first one is the **yesBtn** that is shown to the user when the taxi is registered and the second one is the **yesBtn2** which is shown to the user when the taxi is not registered.
The **yesBtn** method has two main functions. The first is to take the user to their contacts using an intent. And the second function is to store the driver details, current location of the user and the time and date to the history node in the database. There is also an **onComplete** method to make sure the upload was successful. The only difference the **yesBtn2** has, is that it only stores the drivers name as NOT REGISTERED and the number that has been entered.

Here is a snippet of how the data is stored in **yesBtn**:

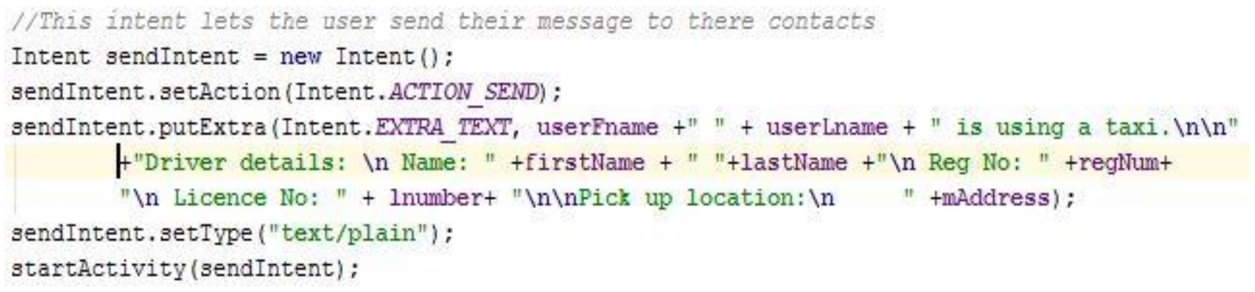


```
userRef.child("history").child(historyId).child("date").setValue(dateNow);
userRef.child("history").child(historyId).child("time").setValue(timeNow);

//Confirmation from Firebase Realtime database of upload success
DatabaseReference dataRef = userRef.child("history").child(historyId).child("driver_fname");
dataRef.setValue(firstName, new DatabaseReference.CompletionListener() {
    @Override
    public void onComplete(DatabaseError databaseError, DatabaseReference databaseReference) {
        if (databaseError != null) {
            Toast.makeText(MainActivity.this, "Data update failed", Toast.LENGTH_SHORT).show();
```

Figure 21: yesBtn

1.6.6 CONTACTS

An intent is used to send the username, taxi driver details and pickup location to the user’s contacts through WhatsApp (WhatsApp Inc, 2017). This code is very important. Without it, the user will not be able to send the information on to their contacts which is on of the main reason of this project. The snippet below shows how EXTRA_TEXT is used to send the information in a text layout.

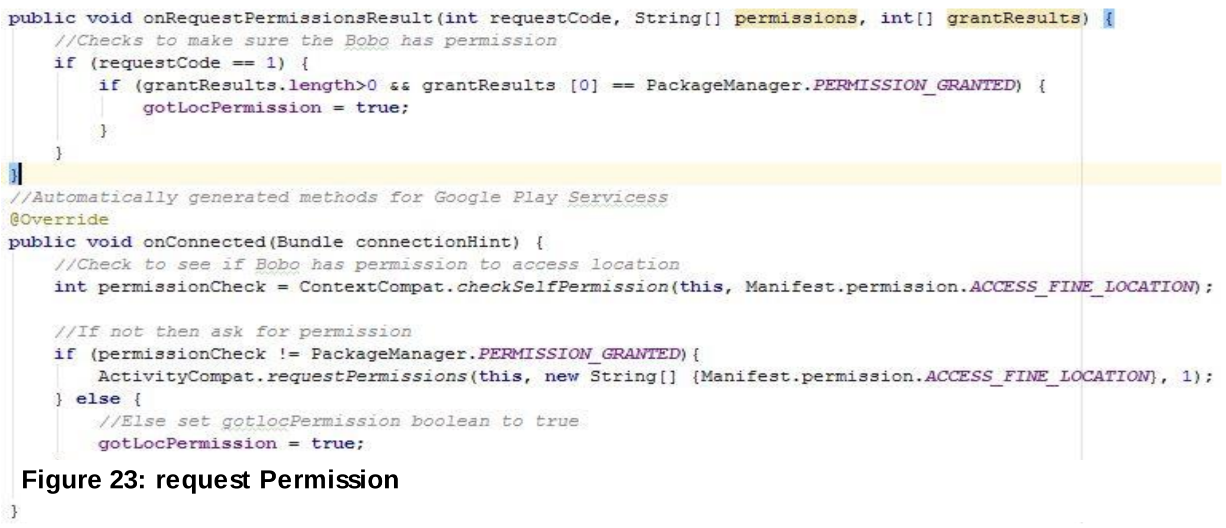


```java
//This intent lets the user send their message to there contacts
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, userFname +" " + userLname + " is using a taxi.\n\n"
        +"Driver details: \n Name: " +firstName + " "+lastName +"\n Reg No: " +regNum+
        "\n Licence No: " + lnumber+ "\n\nPick up location:\n     " +mAddress);
sendIntent.setType("text/plain");
startActivity(sendIntent);
```

Figure 22:Intent for WhatsApp

1.6.7 PERMISSION

This asks for permission from the user to use a certain feature of their device. To do this we need to declare our permission in the Manifest file (Android, Add Permissions to the Manifest, 2017). Next, we need an **onRequestPermissionResult** method that handles the permission request at launch (Requesting Permissions at Run Time). The **onConnected** method will ask for permission every time it tries to use the location (Marini, 2016).



```java
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    //Checks to make sure the Bobo has permission
    if (requestCode == 1) {
        if (grantResults.length>0 && grantResults [0] == PackageManager.PERMISSION_GRANTED) {
            gotLocPermission = true;
        }
    }
}
//Automatically generated methods for Google Play Servicess
@Override
public void onConnected(Bundle connectionHint) {
    //Check to see if Bobo has permission to access location
    int permissionCheck = ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION);

    //If not then ask for permission
    if (permissionCheck != PackageManager.PERMISSION_GRANTED){
        ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.ACCESS_FINE_LOCATION}, 1);
    } else {
        //Else set gotlocPermission boolean to true
        gotLocPermission = true;
}
```

Figure 23: request Permission

1.6.8 LOCATION

Adding location to the Bobo app at the start seemed to be easy to implement. Getting the longitude and latitude was done within two - three hours. After spending hours then trying to use the longitude and latitude to store the users location as a Google Maps pointer and send it, could not be done. The aim of this was to allow the user to send their location and the contact will be able to open the message using Google Maps and see where the pointer is to indicate where the pickup location was. The solution to this, is to send the users location an address, at least then the contacts receiving will know where the pickup location is.

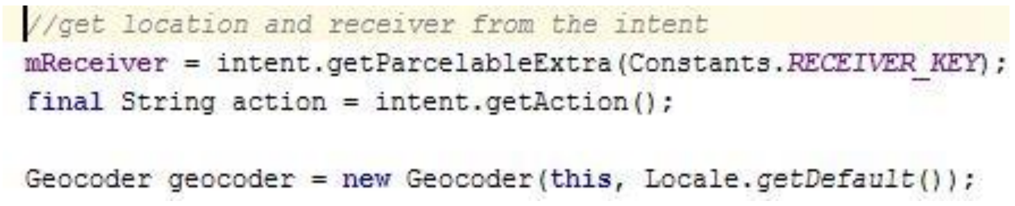
The implementation of location function became very difficult, especially with the newest update that need you to install Google Play API's. It took hours to get this feature to work. It was finally accomplished thanks to a video tutorial (Marini, 2016)

Two Java classes had to be made for the location to work, Constants and GeocodeServices.

- Constants class holds a set of constant values that the Main Activity will use. It passes data between main activity and the service.
- The GeocodeService class uses the **getfromLocation()** method to pass the longitude and latitude to the geocoder. The geocoder returns an array of addresses. To convert a location to an address to is called reverse geocoding. This is what the geocoder method does.

To get the get the address an **onHandleIntent** function has to be implemented within the GeocodeService. This function is where all the geocoding takes place.

The snippet below shows how the **onHandleIntent** function gets the RECEIVER_KEY from the Constants class. This is the package name that has being returned from Google Play Service API. You can also see in the code below, the Locale object being passed to the geocoder object. This is done to ensure the returned address is localized to the user's area.



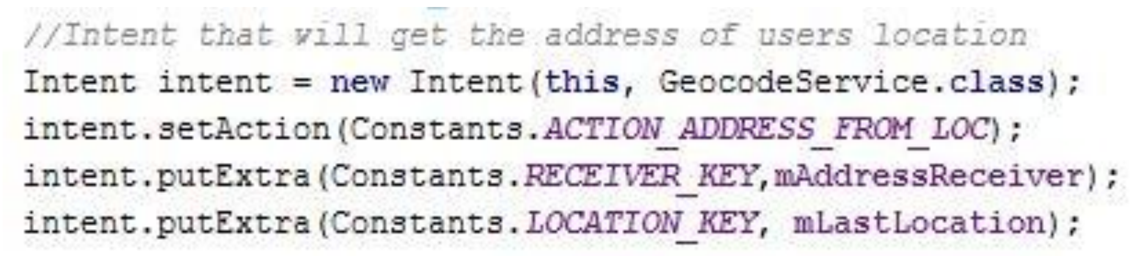
```
//get location and receiver from the intent
mReceiver = intent.getParcelableExtra(Constants.RECEIVER_KEY);
final String action = intent.getAction();

Geocoder geocoder = new Geocoder(this, Locale.getDefault());
```

When

Figure 24: Geocoder

the user clicks the Check button, this activates the **getAddressFromLoc** (get the address from location) method is activated. If the user has a location, then this triggers an intent in the main activity to gets the Constants class to retrieve the address from location and send to the GeocodeService class which will then store the address which can then be used as a text and sent to the user's contacts using the Contacts intent.

This is the Intent code used to get the Address:



```
//Intent that will get the address of users location
Intent intent = new Intent(this, GeocodeService.class);
intent.setAction(Constants.ACTION_ADDRESS_FROM_LOC);
intent.putExtra(Constants.RECEIVER_KEY,mAddressReceiver);
intent.putExtra(Constants.LOCATION_KEY, mLastLocation);
```

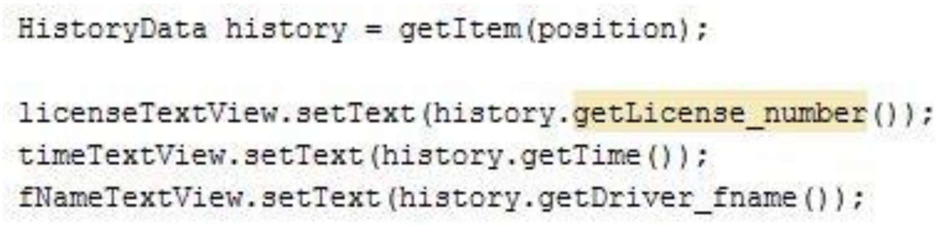
Figure 25: Get Address Intent

1.6.9 HISTORY

The History function needed to be able to display the stored details of the taxi's used, by the user along with when and where. This function is activated in the History activity (HistoryActivity) and uses two other classes.

- History Data (HistoryData) sorts the data from the Snapshot and sorts it by using setters and getters.

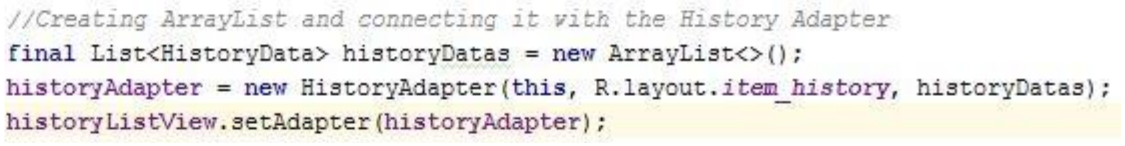
- History Adapter (HistoryAdapter) takes the data that is passed on from the History activity using an Array List and displays the data using List View (Firebase in a Weekend: Android). This is shown in figure 25:



```
HistoryData history = getItem(position);

licenseTextView.setText(history.getLicense_number());
timeTextView.setText(history.getTime());
fNameTextView.setText(history.getDriver_fname());
```

Figure 26: History data Array List

And here is the code that lets the History activity pass the data from History data class to the History Adapter class:



```
//Creating ArrayList and connecting it with the History Adapter
final List<HistoryData> historyDatas = new ArrayList<>();
historyAdapter = new HistoryAdapter(this, R.layout.item_history, historyDatas);
historyListView.setAdapter(historyAdapter);
```

Figure 27: Passing data to HistoryAdapter

1.7 *Graphical User Interface (GUI)*

LOGIN SCREEN:

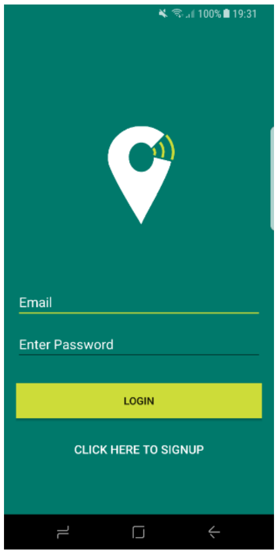
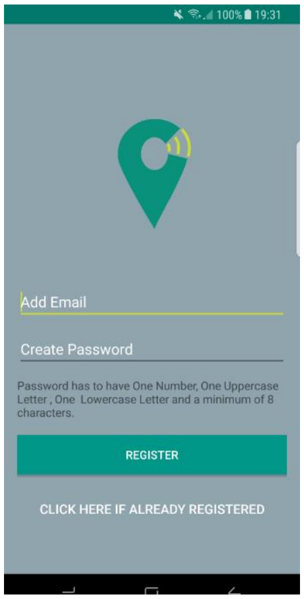


Figure 28:Login Screen

The Login screen consists two edit text field to allow the user to input their login details. The details needed are the email address and Password. There is also a Login button and Signup link. When pressed, the button activates the authentication procedure and the link open the Signup screen.

SIGNUP SCREEN:

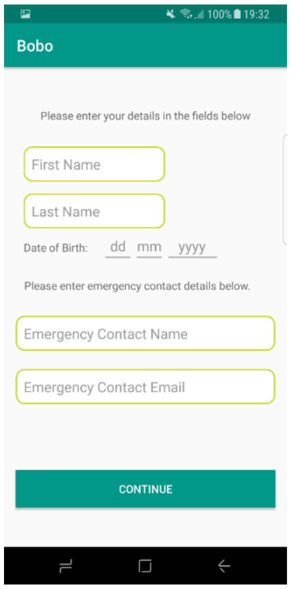
The Signup screen is where the users will create an account. Just like the Login screen, the Signup screen has two edit text fields, a button and a link. The Add Email and Create Password edit text field allow the user to add their details and the Register button will send these details to see if they are valid. The Already Registered link will bring the user to the Login screen. There is also a message that outlines the password pattern that is needed for it to be acceptable.



Password has to have One Number, One Uppercase Letter , One Lowercase Letter and a minimum of 8 characters.

Figure 29: Signup Screen

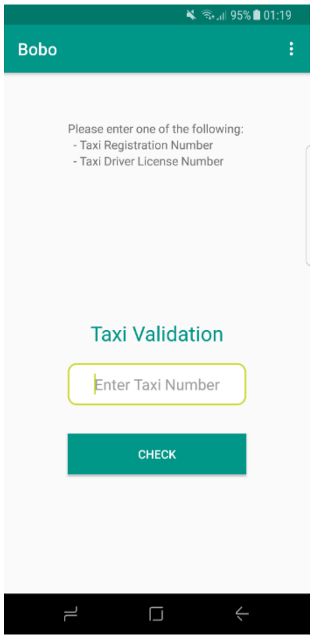
ACCOUNT FORM:


Bobo
Please enter your details in the fields below
First Name
Last Name
Date of Birth: dd mm yyyy
Please enter emergency contact details below.
Emergency Contact Name
Emergency Contact Email
CONTINUE

The Account Form takes personnel details of the user. There is seven edit text fields, two messages and a button. The first two fields require the users first and last name. The next three edit text fields are for the user's date of birth in the dd/mm/yyyy pattern. And the last two are for the user's emergency contact details (Name and Email address). The Continue button is checks to make sue the data entered is valid before sending the data to be stored. The messages just let the user know when to enter their own details or that of their emergency contact.

Figure 30: Account Form

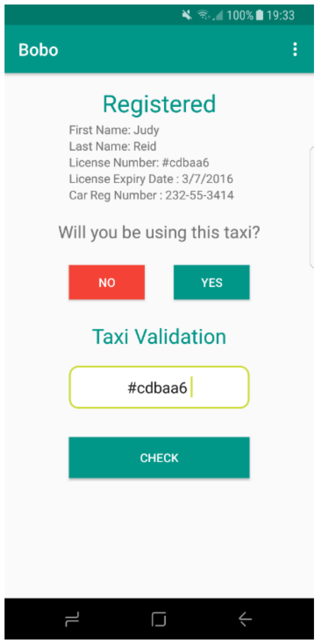
VALIDATION SCREEN:


Bobo
Please enter one of the following:
- Taxi Registration Number
- Taxi Driver License Number
Taxi Validation
Enter Taxi Number
CHECK

Validation screen is in the unchecked condition (hasn't tried to validate). This screen consists of one edit text field, one button and one message. The field allows the user to enter a taxi's registration number or license number. The button activates the validation and the message lets the user know what details are needed to verify the taxi.

Figure 31: Validation Screen

VALIDATION REGISTERED SCREEN:



Registered

First Name: Judy
Last Name: Reid
License Number: #cdbaa6
License Expiry Date : 3/7/2016
Car Reg Number : 232-55-3414

Will you be using this taxi?

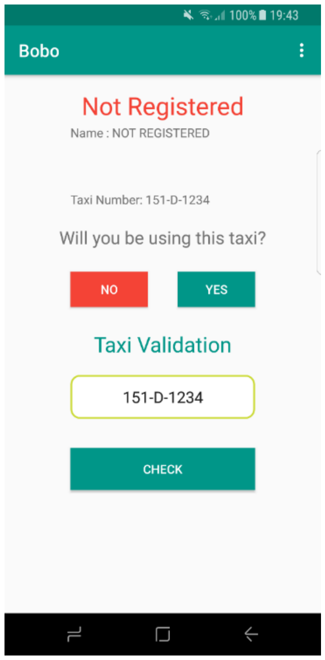
#cdbaa6

Figure 32Validation Registered Screen

The Registered screen is the Validation screen with additions to indicate that the taxi is registered. The additions are seven text views, used to display taxi driver details, a heading highlighting “Registered” and a message asking the user “Will you be using this taxi”. There are to buttons YES and NO. These are the options to the user.

VALIDATION NOT REGISTERED SCREEN:

The Not Registered screen is the Validation screen with additions to indicate that the taxi is not registered. The additions are four text views, used to display the registration number added by the user and a heading highlighting “Not Registered” and a message asking the user “Will you be using this taxi”. There are to buttons YES and NO. These are the options to the user.



Not Registered

Name : NOT REGISTERED

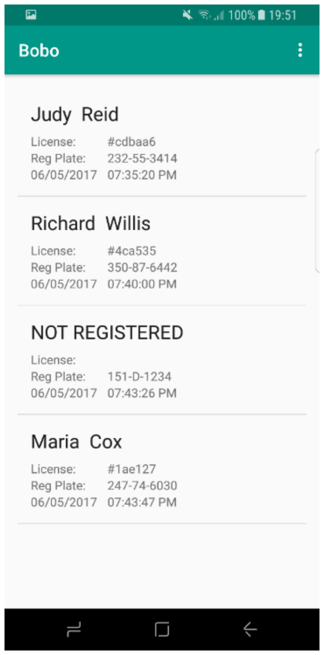
Taxi Number: 151-D-1234

Will you be using this taxi?

151-D-1234

Figure 33: Validation not registered screen

HISTORY SCREEN:


19:51

Bobo

Judy Reid
License: #cdbaa6
Reg Plate: 232-55-3414
06/05/2017 07:35:20 PM

Richard Willis
License: #4ca535
Reg Plate: 350-87-6442
06/05/2017 07:40:00 PM

NOT REGISTERED
License:
Reg Plate: 151-D-1234
06/05/2017 07:43:26 PM

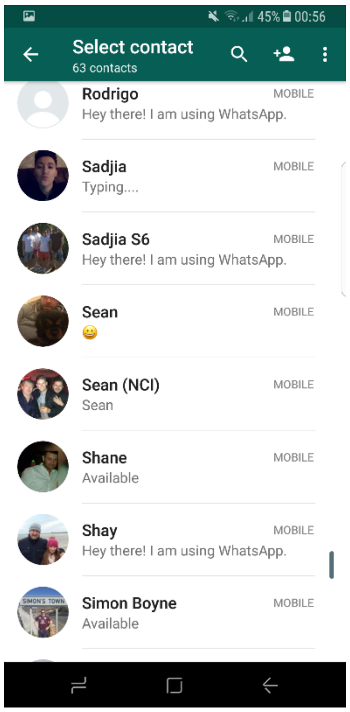
Maria Cox
License: #1ae127
Reg Plate: 247-74-6030
06/05/2017 07:43:47 PM

Figure 34: History Screen

The History screen displays the users previous taxi usage, along with the date and time of the pickup. This is useful in case’s such as the user has left something in the taxi and would like to have the necessary taxi driver information to find or get Gardaí to find the driver.

CONTACTS SCREEN (WHATSAPP):

Using the WhatsApp app, the users can send the taxi detail and pickup location to their WhatsApp contacts.


Select contact
63 contacts

Rodrigo MOBILE
Hey there! I am using WhatsApp.

Sadjia MOBILE
Typing....

Sadjia S6 MOBILE
Hey there! I am using WhatsApp.

Sean MOBILE
😀

Sean (NCI) MOBILE
Sean

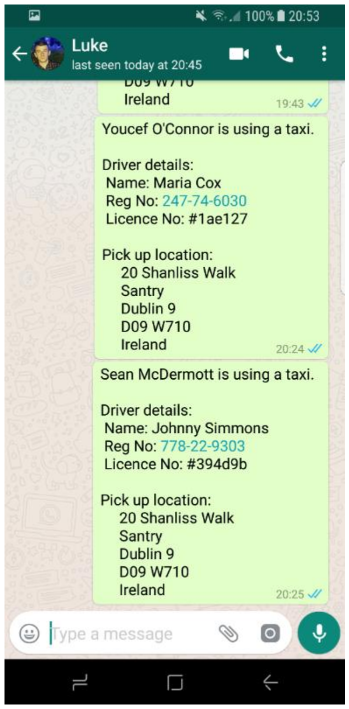
Shane MOBILE
Available

Shay MOBILE
Hey there! I am using WhatsApp.

Simon Boyne MOBILE
Available

Figure 35: Contacts screen (WhatsApp)

MESSAGE LAYOUT (WHATSAPP):


100% 20:53

Luke

last seen today at 20:45

D09 W710
Ireland 19:43

Youcef O'Connor is using a taxi.

Driver details:
Name: Maria Cox
Reg No: 247-74-6030
Licence No: #1ae127

Pick up location:
20 Shanliss Walk
Santry
Dublin 9
D09 W710
Ireland 20:24

Sean McDermott is using a taxi.

Driver details:
Name: Johnny Simmons
Reg No: 778-22-9303
Licence No: #394d9b

Pick up location:
20 Shanliss Walk
Santry
Dublin 9
D09 W710
Ireland 20:25

Figure 36: message layout (WhatsApp)

This is a message that has been sent to a contact through WhatsApp. Notice that the two message sent have two different names. That's the users name of whoever was signed into the Bobo app and sent it. The name is taken from the user name, the taxi driver detail are taken from the database and the location was taken from Google play API at the time this message was sent.

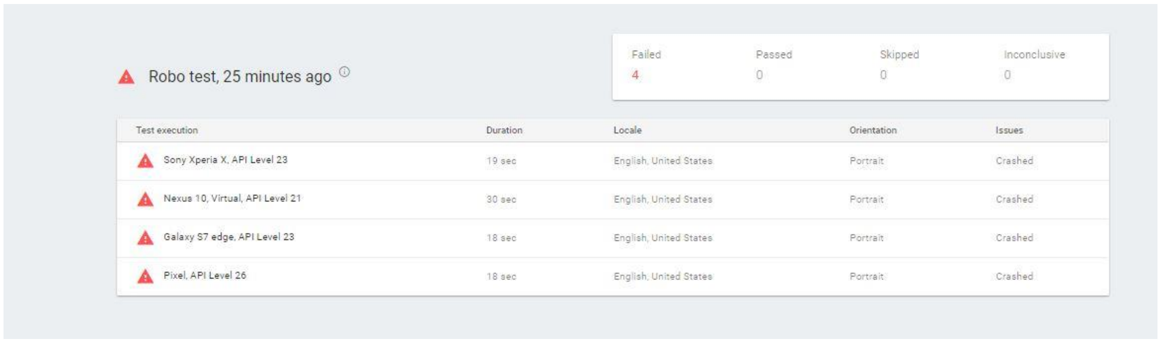
1.8 Testing

In this section, the testing is done using two methods. One method was using Firebase's Test Lab for Android and the other was a usability test. We are going to go through each method separately and explain the outcome of the tests and what was done solve any issues or errors.

Firebase Test Lab is a cloud-based platform for testing Android apps. In your Firebase console the app can be uploaded and tested and there's a Command Line Interface for testing continuous integration servers. The test results can be given as logs, screenshots and also video (Firebase Test Lab for Android).

FIREBASE TEST LAB

This process tests the app on different type of android devices and gives back a report. After running the test on four devices and one virtual machine(VM), the Bobo app failed all tests.



Robo test, 25 minutes ago

Failed	Passed	Skipped	Inconclusive
4	0	0	0

Test execution	Duration	Locale	Orientation	Issues
Sony Xperia X, API Level 23	19 sec	English, United States	Portrait	Crashed
Nexus 10, Virtual, API Level 21	30 sec	English, United States	Portrait	Crashed
Galaxy S7 edge, API Level 23	18 sec	English, United States	Portrait	Crashed
Pixel, API Level 26	18 sec	English, United States	Portrait	Crashed

Figure 37: Firebase test lab

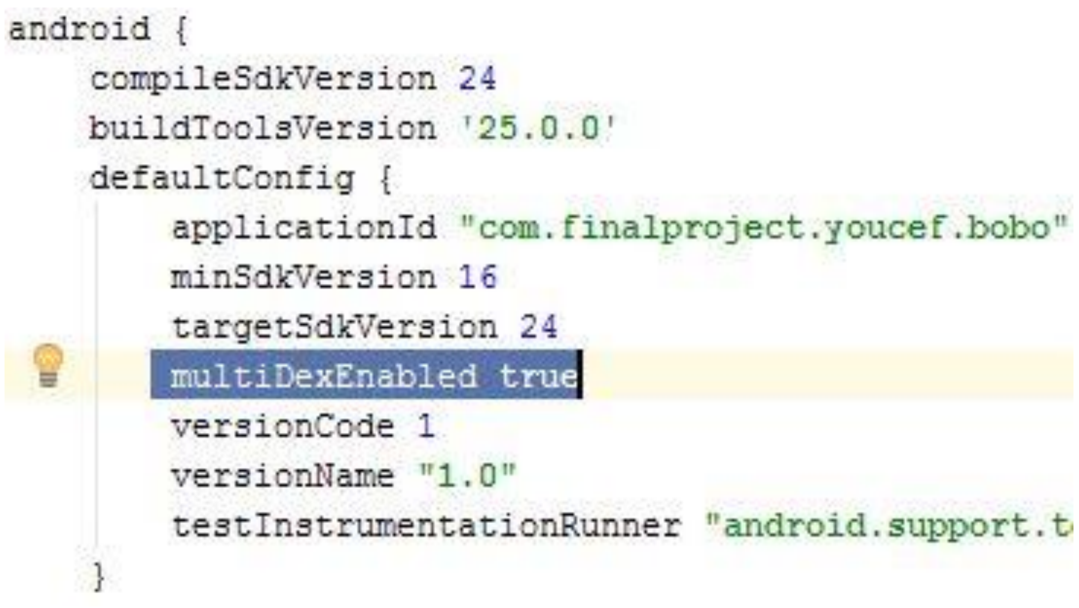
PROBLEM

After investigating these results, the main reason they all failed was due to a problem with finding the location of the DexPathList. After researching this problem, it was found that I needed to configure the Bobo app for multidex.

Multidex is when an app needs more than one Dalvik Executable (DEX). DEX contains the code needed for your app to work. It is a result of using the Google Play Service API's.

SOLUTION:

To solve this problem, multidex had to be enabled in the build gradle (app level) file (Android Studio, 2017). The code below shows this:

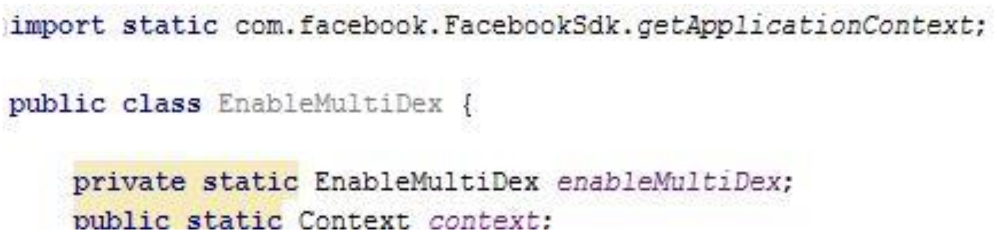


```
android {
    compileSdkVersion 24
    buildToolsVersion '25.0.0'
    defaultConfig {
        applicationId "com.finalproject.youcef.bobo"
        minSdkVersion 16
        targetSdkVersion 24
        multiDexEnabled true
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.t
    }
```

Figure 38: Adding multiDexEnabled

RESULTS

This solution did not work, the same error is still appearing. After more investigating, it was found on Stackoverflow that a user (Redman) has found a solution. On the first step of this new solution, a new Java class needed to be created and the contents from Redman's solution added to the new class. Before proceeding to the next stage, something stood out suspiciously. An import automatically generated as seen in figure 37 and it was a Facebook SDK.



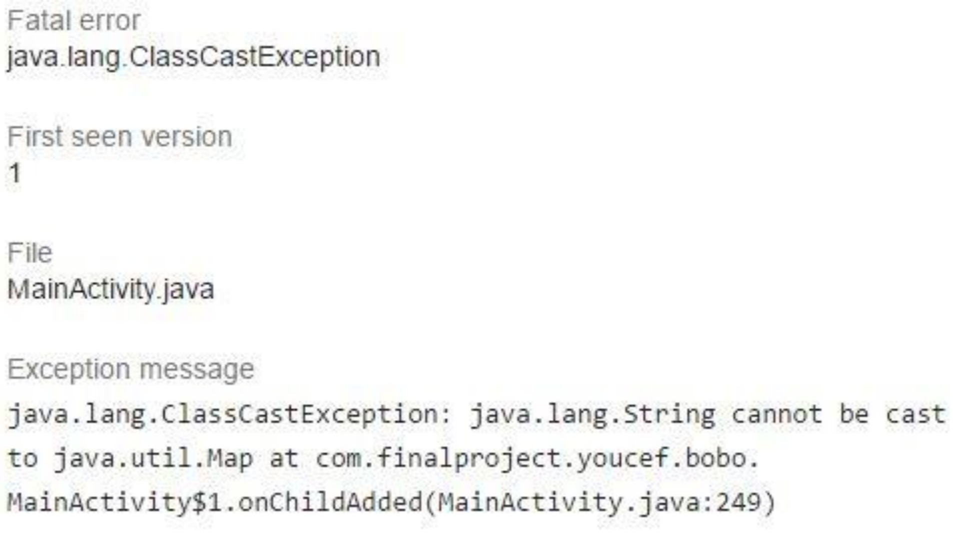
```
import static com.facebook.FacebookSdk.getApplicationContext;

public class EnableMultiDex {

    private static EnableMultiDex enableMultiDex;
    public static Context context;
```

Figure 39: Suspicious import

Because this is from an unreliable source and looks suspicious, we will not proceed with this method. This problem is still currently unresolved.

1. PROBLEM

Another error that has been found from the test, feedback shown below:



Fatal error
java.lang.ClassCastException

First seen version
1

File
MainActivity.java

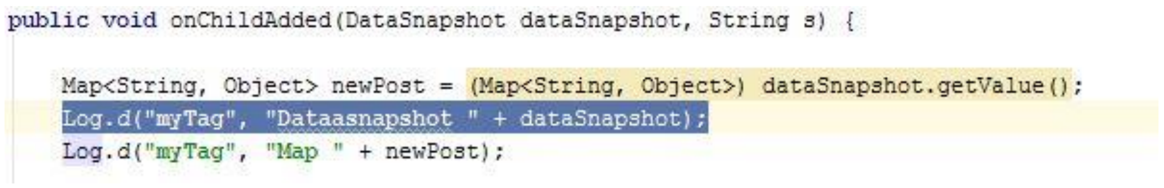
Exception message

```
java.lang.ClassCastException: java.lang.String cannot be cast
to java.util.Map at com.finalproject.youcef.bobo.
MainActivity$1.onChildAdded(MainActivity.java:249)
```

Figure 40: Class Exception

The problem here was how a Map was used to temporarily store the Firebase data in the **onChildAdded** method. After investigating the code, it was noticed that using a Map was unnecessary. The Snapshot already does this.

On Child Added method before changes:

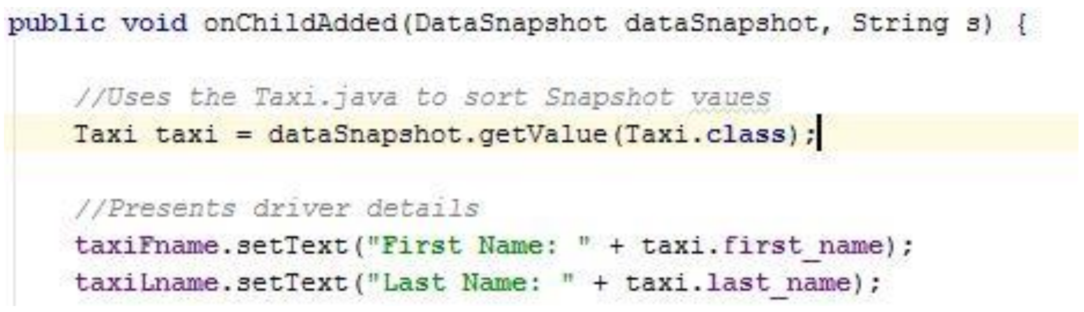


```
public void onChildAdded(DataSnapshot dataSnapshot, String s) {

    Map<String, Object> newPost = (Map<String, Object>) dataSnapshot.getValue();
    Log.d("myTag", "Dataasnapshot " + dataSnapshot);
    Log.d("myTag", "Map " + newPost);
```

Figure 41: Using Map

SOLUTION:

Remove the Map and create a Taxi Java class that can temporarily store and sort the data from **DataSnapshot**. Then in the method the data Snapshot values have been declared and now the values can be retrieved directly from the Snapshot instead of needing to be stored in a Map (Read and Write Data on Android). This problem has been solved.



```
public void onChildAdded(DataSnapshot dataSnapshot, String s) {

    //Uses the Taxi.java to sort Snapshot vaues
    Taxi taxi = dataSnapshot.getValue(Taxi.class);

    //Presents driver details
    taxiFname.setText("First Name: " + taxi.first_name);
    taxiLname.setText("Last Name: " + taxi.last_name);
```

Figure 42: Improved method

USEABILITY TESTING

For this test two participants in their early twenties who identified themselves as regular taxi users will test the app. Unfortunately they bought have the Galaxy S8 which is the same device used for development. This means we won't get to see Bobo app running on different android versions during the usability test.

Before we began, we sat down and they were a scenario. The scenario was: You are on a night out and got separated from your friends so you decided to head home alone.

I also gave them two taxi registration and license numbers that are valid to the Bobo App and one that is not. They were not shown how to use the app.

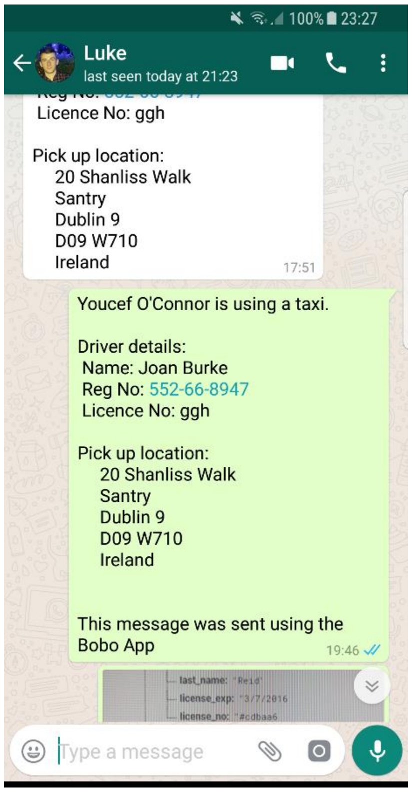
Feedback:

- Registering was easy, it didn't ask for too much.
- It would be great to able to scan the register plate
- I like the way I can let my WhatsApp contacts know.
- I don't like the way I have to type the taxi number in.
- The layout and look of the app is good, looks professional.
- Is there a way to say YES without sending to contacts?
- The main menu is straight forward.
- I like the message that's being sent but it would be good to mention that it is coming from the Bobo app.

Action:

After the usability test, action on the feedback has started immediately.

The first thing that was implemented from the test was adding "This message was sent using the Bobo App" to the message



Licence No: ggh

Pick up location:
20 Shanliss Walk
Santry
Dublin 9
D09 W710
Ireland
17:51

Youcef O'Connor is using a taxi.

Driver details:
Name: Joan Burke
Reg No: 552-66-8947
Licence No: ggh

Pick up location:
20 Shanliss Walk
Santry
Dublin 9
D09 W710
Ireland

This message was sent using the Bobo App
19:46

```
last_name: "Reid"
license_exp: "3/7/2016
license_no: "#cdbaa6
```

Figure 43: Updated message

The next task to be taken from the feedback is to create a registration scanning feature using Optical Character Recognition (OCR). The project supervisor Dr Dominic Carr found a great solution to this on GitHub.

Unfortunately due to time constraints this function or any of the functions from the feedback could not be implemented or even investigated.

2 Evaluation

Positive Implications

As outcomes from the usability test the apps layout is good, the design is modern and gives the Bobo app a professional look. It is easy to navigate between the validation screen and the history screen. The signup procedure is straight forward and didn't ask for much. And the feedback to the user after validating a taxi is direct, only gives the user what is needed and lets the contacts know what app is used to send the message which was requested from the usability feedback.

Being able to forward the details to WhatsApp contacts has many benefits:

- 31% of social messaging users in Ireland use WhatsApp (Smith, n.d.)
- It can be used on Android and Apple devices
- WhatsApp has a smartwatch app for Android Wear and Apple watches
- The users contacts are already integrated
- The Smartwatch requirement is completed on the case that it is using the WhatsApp for sending message.

There are also disadvantages to using WhatsApp, they are:

- No customization of how the contacts are displayed or used.
- Not being able to monitor who the user is sending the messages two or how many messages they send. The Firebase analytics cannot monitor WhatsApp as it is using its own servers.

The storing of the data is quick and fluid thanks to Firebase's Real-time database. The data in the History screen is clear and accurate. And the message that is sent only has the information needed.

Negative Effects

- Couldn't get real data, the workers at Transport Ireland said it would be a breach of data protection (Data Requirement).
- Failure to create emergency contact feature for users under 18 for the Contacts requirement.
- Failure to meet Contacts (Arrived) requirement for when the user arrives safely, to let their contacts know that they got home safe and sound.
- The testing could have been more in-depth.

With the exception of the real-time data acquisition, these in-complete elements are a result of time constraints.

3 Conclusion

The purpose of the Bobo app is to allow users to feel safe when using a taxi. This is done by informing their contacts with the taxi driver details and the pickup location. Using this app will allow users to see if the taxi driver is a registered driver or not. However, during the construction of the app, the developer encountered an obstacle of obtaining the real data deemed necessary to show how the application could be used in real life situations. So therefore, the project is using mock data.

It is felt that obtaining the real data for the app, Bobo, could have real life implications. This app could have the ability to ensure those using a taxi be it someone who is travelling during the day or night can check the legitimacy of the driver which would allow them to make an informed decision as to whether they would prefer to use that taxi or not, and due to the ever-increasing demand for this safety feature other apps have developed a similar feature that allows users to share information about their taxi journey (GmbH, 2017).

4 Further development or research

I see Bobo helping the user in numerous ways with other features like:

- “Travel Alarm” that uses GPS to set it off when the user is close to location.
- “Did You Forget Me” feature that lets the user know that their phone has being left behind.
- The smart phone will notice the distance from the smart watch is getting bigger, so smart phone will send an alert to the Android Wear watch when saying “Did you forget (Phone ID)?”. The user will be able to swipe left for Yes and right for No. If Yes, then the user will be able to ring their phone form the smart watch.

5 References

Android. (2017). *Add Permissions to the Manifest*. Retrieved from Android Developers: https://developer.android.com/training/permissions/declaring.html

Android. (2017). *Requesting Permissions at Run Time*. Retrieved from Android Developers: https://developer.android.com/training/permissions/requesting.html

Android Logo. (n.d.). Retrieved from media cache: https://s-media-cache-ak0.pinimg.com/originals/1b/62/a6/1b62a65f7e1afcd0fbe31e2a7af0310d.png

Android Studio. (2017). *Configure Apps with Over 64K Methods*. Retrieved from Android Studio: https://developer.android.com/studio/build/multidex.html#testing

Android studio Logo. (n.d.). Retrieved from APK Android Blogspot: http://net-apk-android.blogspot.ie/2016/02/android-studio-151-build-1412456560.html

anoop. (n.d.). *Android Move cursor from one EditText to another one if click any letter in field?* Retrieved from StackOverflow: http://stackoverflow.com/questions/12418324/android-move-cursor-from-one-edittext-to-another-one-if-click-any-letter-in-fiel

Baradia, R. (2012, 08 18). *email address validation in android on edittext*. Retrieved from Stack Overlow: http://stackoverflow.com/questions/12947620/email-address-validation-in-android-on-edittext

Companion. (2017, 05 09). Retrieved from Companion: https://www.companionapp.io/

developer, A. (2017). *Patter*. Retrieved from Android Developer: https://developer.android.com/reference/java/util/regex/Pattern.html

Eye, Tek. (2017). *List of IDEs for Android App Development, Which is Best for You?* Retrieved from Tek Eye: http://tekeye.biz/2014/list-of-android-app-development-ides

Firebase. (2017, 05 08). *Add Firebase to Your Android Project*. Retrieved from Firebase: https://firebase.google.com/docs/android/setup

Firebase. (2017). *ChildEventListener*. Retrieved from Firebase: https://firebase.google.com/docs/reference/android/com/google/firebase/database/ChildEventListener

Firebase. (2017). *Firebase Realtime Database*. Retrieved from Firebase: https://firebase.google.com/docs/database/

Firebase. (2017). *Firebase Test Lab for Android*. Retrieved from Firebase: https://firebase.google.com/docs/test-lab/

Firebase. (2017). *Read and Write Data on Android*. Retrieved from Firebase: https://firebase.google.com/docs/database/android/read-and-write

Firebase. (2017). *Work with Lists of Data on Android*. Retrieved from Firebase: https://firebase.google.com/docs/database/android/lists-of-data

Firebase. (n.d.). *Manage Users in Firebase*. Retrieved from Firebase: https://firebase.google.com/docs/auth/android/manage-users

GmbH, I. A. (2017, 05 08). *Welcome to mytaxi, the new way to Hailo!* Retrieved from mytaxi: https://ie.mytaxi.com/hailoisnowmytaxi?erid=1494286974573464203

Gojare, K. (n.d.). *NoSQL Databases*. Retrieved from Sandip Foundation's Students' Blog.: http://itsitrc.blogspot.ie/2014/08/nosql-databases.html

Google. (2016). *Android Wear*. Retrieved from Android Developer: https://developer.android.com/wear/index.html

Google. (2016). *Firebase in a Weekend: Android*. Retrieved from Udacity: https://www.udacity.com/course/firebase-in-a-weekend-by-google-android--ud0352

Google. (2016). *List View*. Retrieved from Android Developers: https://developer.android.com/guide/topics/ui/layout/listview.html

Google. (2016). *Meet Android studio*. Retrieved from Android Studio: https://developer.android.com/studio/intro/index.html

Google. (2016). *Menus*. Retrieved from Android Developers: https://developer.android.com/guide/topics/ui/menus.html

Google. (2017). *DateFormat*. Retrieved from Android Developers: https://developer.android.com/reference/java/text/DateFormat.html

Google. (2017). *Firebase Logo*. Retrieved from Firebase: https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html

Google. (2017). *Requesting Permissions at Run Time*. Retrieved from Android Developers: https://developer.android.com/training/permissions/requesting.html

Google. (2017). *Set Up Google Play Services*. Retrieved from Google API's for Android: https://developers.google.com/android/guides/setup

Google Maps logo. (2016). Retrieved from Vector Logo: http://vectorlogo4u.com/google-maps-2015-vector/

Java Logo. (n.d.). Retrieved from https://www.quora.com/Which-are-the-most-beautiful-logos-youve-ever-seen#!n=102

Karch, M. (2017, 05 08). *What Is Google Android?* Retrieved from Lifwire: https://www.lifewire.com/what-is-google-android-1616887

koceeng. (n.d.). *Using FirebaseUI to Populate a ListView*. Retrieved from GitHub: https://github.com/firebase/FirebaseUI-Android/blob/master/database/README.md

Margaret Rouse. (2017). *Google Maps*. Retrieved from WhatIs.com: http://whatis.techtarget.com/definition/Google-Maps

Marini, J. (2016, 06 24). *https://www.lynda.com/Google-Play-Services-tutorials/Convert-lat-long-address-geocoder/474086/503689-4.html*. Retrieved from Lynda.com from LinkedIn: https://www.lynda.com/Google-Play-Services-tutorials/Convert-lat-long-address-geocoder/474086/503689-4.html

mongoDB. (2017). *NoSQL Databases Explained*. Retrieved from mongoDB: https://www.mongodb.com/nosql-explained

ORACLE. (n.d.). *What is Java technology and why do I need it?* Retrieved from Java: https://www.java.com/en/download/faq/whatis_java.xml

pjiojhgyuiojklm. (n.d.).

Smith, C. (n.d.). *57 Amazing WhatsApp Statistics (March 2017)*. Retrieved from Expanded Ramblings: http://expandedramblings.com/index.php/whatsapp-statistics/4/

Squarespace. (2017). *What is JSON?* Retrieved from Squarespace: https://developers.squarespace.com/what-is-json/

Storey, J. (n.d.). *Stopping timer events in Java*. Retrieved from StackOverflow: http://stackoverflow.com/questions/11775908/stopping-timer-events-in-java

Tamada, R. (n.d.). *Android Location API using Google Play Services*. Retrieved from Android Hive: http://www.androidhive.info/2015/02/android-location-api-using-google-play-services/

Tamada, R. (n.d.). *Android working with Firebase Realtime Database*. Retrieved from Android Hive: http://www.androidhive.info/2016/10/android-working-with-firebase-realtime-database/

testmonk. (n.d.). *7 Testing Tips to Ready your App for Launch*. Retrieved from testmonk: https://blog.testmunk.com/7-testing-tips-to-ready-your-app-for-launch/

Udacity. (n.d.). *Android Development for Beginners*. Retrieved from Udacity: https://www.udacity.com/course/android-development-for-beginners--ud837

WhatsApp Inc. (2017). *I'm an Android developer, how can I integrate WhatsApp with my app?* Retrieved from WhatsApp: https://www.whatsapp.com/faq/en/android/28000012

WhatsApp Inc. (2017, 05 08). *Simple. Secure.* Retrieved from WhatsApp: https://www.whatsapp.com/

WhatsApp Logo. (n.d.). Retrieved from http://whatsapp.descargarapps.info/whatsapp-para-pc/

Yaseen, K. (n.d.). *Photoshop Tutorial* **|** *Logo Design*. Retrieved from Youtube: https://www.youtube.com/watch?v=il_PCSQ-KME

6 Appendix

6.1 ProjectProposal

6.1.1 Objectives

The aim of this project is to develop a fully functioning mobile application that will make traveling safer for the users. The name of this app is Bobo. The user will be able to use this application on Android devices, including smart watches (Android Wear).

Main objectives:

- Make traveling safer by using the taxi verification feature, which will allow the user to enter a taxi's registration or license number. Then the application will display to the user if the taxi is registered or not.
- Allowing the user to send the taxi details and location of where the user was picked up to selected contacts.
- If the user is under eighteen, then the application will automatically send the details to parents or guardians.
- Using real taxi driver data.

I see Bobo helping the user in numerous ways with other features like a "Travel Alarm" that uses GPS to set it off when the user is close to location. And a "Did You Forget Me" feature that lets the user know that their phone has being left behind. The smart phone will notice the distance from the smart watch is getting bigger, so smart phone will send an alert to the Android Wear watch when saying "Did you forget (Phone ID)?". The user will be able to swipe left for Yes and right for No. If Yes, then the user will be able to ring their phone form the smart watch.

Unfortunately, I am very time restricted with this project and will need to focus on the main taxi feature. I will make the other features if there is time after the main objectives have been completed.

6.1.2 Background

The idea of the Bob application hit me two years ago, when my mother was worried about my younger brother who was out one night with friends. My brother came home later than usual when on a night out so the next day I asked him what took him so long. He told me it was the taxi! His taxi driver was rude and clearly didn't know his way around Dublin or he just went the longer way on purpose to get a bigger fair. I asked my brother "why didn't you say anything?", but he felt intimidated by the driver. Also, the driver's picture on his taxi license display was OK but my brother had no way of checking the license number. I know my brother can take care of himself but I can tell he felt un safe in that taxi.

I said to myself, what if there was an app that can let family and friends know what taxi I am in, is the taxi registered and where I have been picked up with a press of a button. This was just an idea at the time and kind of skeptical that this idea was made before. A guy I knew who was a fourth Computing student told me that odds of having an idea that hasn't been made yet is very rare. So, I really didn't give it that much thought, until a year later when a guy I worked with told me of a situation he was in and my app idea would have been useful. My friend told me his wife was on a night out when she got separated from her friends and decided to head home without them. He hated it when she was in a taxi on her own and always told her to send him the taxi license number in case something went wrong.

I took a chance and told an old colleague of mine about my app idea. He told me he loved the idea and would definitely use it, getting a message with all that information will relax me when she is in a taxi on her own he said.

Getting this type of reaction and feedback has got me thinking, maybe there really isn't an app that can do this yet. So I began researching. I found that there is an app in Ireland that lets the user check a taxi registration and license number and that’s it! I also found that the big taxi apps like Uber to my surprise don’t have this feature which I believe is so important. It shows they only care about money rather than the well being of there users.

That’s where my Bobo app will come in. Focusing specifically on the user’s safety when traveling.

6.1.3 Technical Approach

After three years of studying computer science I developed a lot of experience with different types of coding languages, such as Java, HTML5, and C#. So, I feel confident I will able to do this project solo.

Using Plurasight, YouTube and Udacity videos as guides.

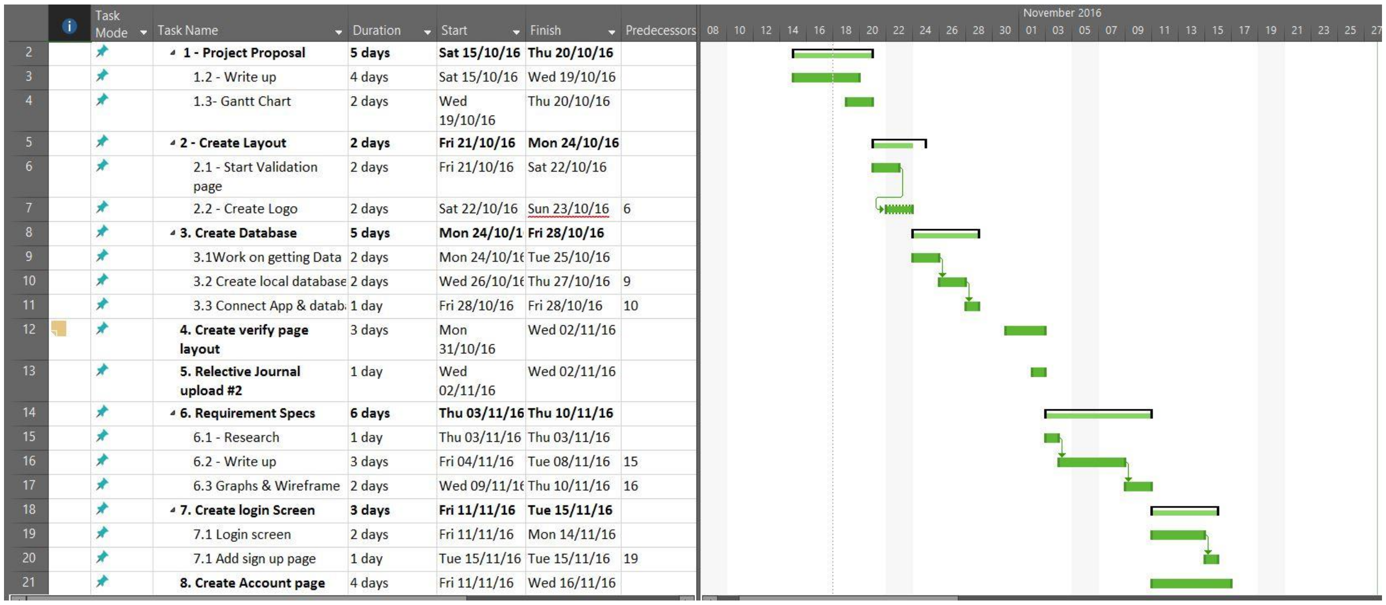
It is possible that this application will also be available to Apple’s iOS but the costs of buying a Mac that is capable of coding on and the €100 fee to get it up on the App Store is too expensive. This just means if I was able to afford it in the future then I will build this application for iOS too, but for now the Bobo app will be developed for Android and Android Wear devices.

6.1.4 Special Resources required:

This is probably the worst part of the project as the college does not provide Android Wear devices to students, which means I must buy one. It will roughly cost me about €350. There are cheaper ones but I will need a high-end smart watch that has GPS and Data Roaming. Also, if I’m paying that much I might as well get the best. At the moment, the Huawei W1 looks to be the best Android Wear device. It has great reviews but my problem is that its nearly 2 years old. So, I decided to wait until the rage of Android Wear devices are released before Christmas.

6.1.5 Project Plan:

Gantt Chart.



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
2	📌	**1 - Project Proposal**	**5 days**	**Sat 15/10/16**	**Thu 20/10/16**	
3	📌	1.2 - Write up	4 days	Sat 15/10/16	Wed 19/10/16	
4	📌	1.3- Gantt Chart	2 days	Wed 19/10/16	Thu 20/10/16	
5	📌	**2 - Create Layout**	**2 days**	**Fri 21/10/16**	**Mon 24/10/16**	
6	📌	2.1 - Start Validation page	2 days	Fri 21/10/16	Sat 22/10/16	
7	📌	2.2 - Create Logo	2 days	Sat 22/10/16	Sun 23/10/16	6
8	📌	**3. Create Database**	**5 days**	**Mon 24/10/1**	**Fri 28/10/16**	
9	📌	3.1Work on getting Data	2 days	Mon 24/10/1€	Tue 25/10/16	
10	📌	3.2 Create local database	2 days	Wed 26/10/1€	Thu 27/10/16	9
11	📌	3.3 Connect App & datab	1 day	Fri 28/10/16	Fri 28/10/16	10
12	📌	**4. Create verify page layout**	3 days	Mon 31/10/16	Wed 02/11/16	
13	📌	**5. Relective Journal upload #2**	1 day	Wed 02/11/16	Wed 02/11/16	
14	📌	**6. Requirement Specs**	**6 days**	**Thu 03/11/16**	**Thu 10/11/16**	
15	📌	6.1 - Research	1 day	Thu 03/11/16	Thu 03/11/16	
16	📌	6.2 - Write up	3 days	Fri 04/11/16	Tue 08/11/16	15
17	📌	6.3 Graphs & Wireframe	2 days	Wed 09/11/1€	Thu 10/11/16	16
18	📌	**7. Create login Screen**	**3 days**	**Fri 11/11/16**	**Tue 15/11/16**	
19	📌	7.1 Login screen	2 days	Fri 11/11/16	Mon 14/11/16	
20	📌	7.1 Add sign up page	1 day	Tue 15/11/16	Tue 15/11/16	19
21	📌	**8. Create Account page**	4 days	Fri 11/11/16	Wed 16/11/16	

6.1.6 Technical Details

So, I had put thought and careful consideration into how I am going to develop the Bobo application and found that using Java and Android Studios is the best option. I have been studying java every year in college and Android Studios is user friendly.

Mobile App Development:

- The mobile app will be developed using Android Studios and my Android smart phone (Samsung Galaxy S8).
- It will be able to run on the Android "Ice-Cream Sandwich" version or later
- Will need permission to use Contacts (to send taxi driver info).
- Will need permission to use the Camera (to scan taxi license barcode).
- Will need permission to use Location (GPS to get picked up location).
- Will need permission to use Data (to receive taxi details).
- Will need permission to use Bluetooth (to link to a smart watch).

Smart Watch Development:

- Will need permission to run an app on Wear device.
- Will need permission to run app from smart phone.
- Will need permission to use Data (to receive message).

6.1.7 Evaluation

I will use friends and family as testers of the Bobo app, to help me with the evaluation. I will be hoping to be evaluating with real data. The test users will be given a survey that will ask them to test specific parts of the app and ask questions on how they feel about the UI and what would they recommend will make the Bobo app better.

6.2 Monthly Journals

6.2.1 Reflective Journal #1 (Sep)

<u>19th September 2016</u>

Day one of my final year in college and feeling ready. The first class of the year was Software Project. This class had all the fourth-year Computing students from all streams present. Our lecturer Eamon Nolan briefed on everything we need to do in next two weeks. We need to have a project idea, run it by a lecturer of our stream (in my case Dominic Carr in IOT) and then get ready for project proposal.

Ideas:

I was eager to start my project! I had two ideas and was looking forward to start either one. My first idea was a Wi-Fi controlled car (WWC). The WWC would have been made from a broke down remote control car, raspberry pi 3, Arduino motor controller and a camera. An android app would have been made to control the car from a mobile device.

My second idea is the Bobo app. The bobo app helps to make traveling safer. It does this by letting the user check a taxi registration or license number is registered. If the taxi is registered, then the user is given an option. “Will you be using this taxi: Yes, or No?”. If the user hit Yes, then the user will be able to send the taxi details to their contacts. If the user is under 18 then the app will automatically send the taxi details including the location of the pick up to the parents / guardians.

It was not long into my first lecture till I started noticing a problem with the modules I have this year. You see this is the first-time Artificial Intelligence (AI) is done in forth year, so we are doing an extra subject than previous years. This is problem because Eamon is setting our project standard to previous years which I thought is unfair. We have less time! I have asked Eamon what has changed in the Software Project module to confiscate for the extra work load of AI. He said that is a good question and I should take it up with Paul Stynes.

<u>27th September 2016</u>

After having two meetings with Sam Cogan I finally got a sit down with Paul Stynes. I told him about the problem and concerns that not just me but all fourth-year students are worried about the amount of time AI is taking up. He was happy to hear this sooner rather than later and said he be looking into it.

<u>29th September 2016</u>

I decided to go with the Bobo app for my fourth-year project. I choose this because after doing some research I found that the WWC would have been too complicated and I felt that I would have spent a lot more time on hardware rather than software.

<u>30th September 2016</u>

Preparing my slides for the project proposal. I really hope Bobo gets the green light.

6.2.2 Reflective Journal #2 (Oct)

<u>3th October 2016</u>

Finishing slides for the project idea pitch presentation, that are due to be uploaded tonight. It is only two slides but I need to make sure they explain my app idea in a nutshell and the key functions are explained.

<u>5th October 2016</u>

Project Pitch today! I had to pitch my idea to three judges. Two judges were happy with the main function of the Bobo app idea but the third thought it would be a bit easy for a final year project. When I explained that there will be other features other than just verifying a taxi like checking history, all three judges were happy. The results of the project idea pitch were that the Bobo App got the green light and to achieve high marks it needs to be more difficult.

<u>10th October 2016</u>

Over a week after being given the go ahead to develop the app and I find it hard to get time to work on my project. I have assignments due nearly every week, next week I have three assignments due. I also notice that I spend my Software Project Independent study time working on my Artificial Intelligence project with the rest of the class.

<u>21st October 2016</u>

Finished my Project Proposal document that is due tonight! I explained exactly how my project will work and all the functions it will have. Made this document as clear as possible, so my final year project supervisor can understand my app idea.

<u>24th October 2016</u>

Found out today that Dominic Carr is my final year project supervisor. I am very happy with him being my supervisor as I am doing the internet of things (IoT) stream and he is the IoT lecturer.

<u>2nd November 2016</u>

Started to look for real taxi driver license data. I found it is extremely difficult to get this data in Ireland, so I started to look outside of Ireland. I realized if my app is aimed at people who travel, then I will need to get taxi driver data from other countries and if I am going to do that then I should focus on some of the big cities that have a lot of taxi's like New York.

<u>4th November 2016</u>

After doing some research I found an open data website called New York City Open Data. This site has all the licensed drivers of New York City and State (48,081 drivers).

I spoke with my supervisor today who gave me good advice on how I could retrieve taxi driver data. He also helped me to prioritize my tasks for this project and gave me great tips on how to get top marks.

6.2.3 Reflective Journal #3 (Nov)

24th November 2016

I had a meeting my supervisor today to let him know I was unable to complete the requirement spec by today. We didn't have much else to cover other than I will need to complete it ASAP

2nd December 2016

With all the time put into other modules, I was able to finally get my project Requirement Specifications completed. I wasn't 100% sure if the user case diagrams are ok, so I will ask Dominic (Supervisor) if they are OK.

3th December 2016

Made a start of my technical report.

5th December 2016

Today I had a meeting with Dominic to see if he is happy with my requirement spec and how I am doing with the tech report. He gave me great tips and I left the meeting with a list of what is needed to complete the tech report.

6th December 2016

Created GUI's using Android Studio.

6.2.4 Reflective Journal #4 (Dec)

9th December 2016

I have made changes to my projects technical report as advised by my supervisor Dominic. The changes varied from changing the way I referenced the report to a more professional Endnote referencing and I broke up my Login and Register functional Requirement to two separate requirements, Login requirement and Registration requirement. I have also made some small changes to the diagrams used.

11th December 2016

I have completed my project technical report and upload it to Moodle today.

13th December 2016

I have planned to start my mid-point presentation work today but I have three assignments due this week and find it very hard to make a start on preparing for my mid-point presentation.

17th December 2016

Today I have started to work on my mid-point presentation. I will not be able to put a working prototype together in this time frame.

18th December 2016

I used adobe experience design to develop a visual demo layout of how the Bobo app will look like when up and running. It was great way to demonstrate an apps prototype with a prototype. I completed my preparation for my mid-point presentation hat due tomorrow.

19th December 2016

I had my mid-point presentation today. I seemed confident in what I had to present would please the judges / lecturers. The presentation went well, I talked for over 20 mins straight and answered all questions asked.

6.2.5 Reflective Journal #5 (Jan)

18th January 2017

Today was the first day I got a chance to sit down and give my final project full attention now that the exams and semester one is over. I was deciding whether to use a SQL server database or an NoSQL server database. I weighed the pros and cons and found that Google's Firebase (NoSQL) would be the best server option for the Bobo app.

25th January 2017

After watching multiple Firebase tutorials on Udacity and YouTube I have become familiar with all of the features it has to offer such as real time database and authentication. Firebase seems to have what the Bobo app needs to be fully completed, especially with its messaging services.

3th February 2017

The results for semester two came out today. I was happy with the results but thought I could have done better. Really curious to find out where I went wrong.

6th February 2017

After two weeks of trying to use the firebase database for the verification function which is or should be in this case a simple search task of entering a taxi License number and returning the taxi drivers details has seemed to be a difficult challenge for me. With the firebase being an NoSQL database and that it is still relatively new, I cannot find any tutorials to help me.

9th February 2017

I have decided to use a SQL server database for my verification feature and continue to use Firebase for the other features. There seems to be a lot of tutorials and aids that can help me build a search function to a SQL server. I have created an Amazon Web Services (AWS) account and started work on the SQL database.

10th February 2017

Today I got a chance to sit with my supervisor and informed him of my situation with unsuccessfully using Firebase for my verification function and that I'm changing to SQL. He agreed that the tutorials online seemed useless and took upon himself to find a solution and thought me how it can be done.

6.2.6 Reflective Journal #6 (Feb)

17th February 2017

Today I have created a new GitHub account. The reason I created a new GitHub account was to get GitHub's student package. This package included a free two year private account which was necessary for my final year projects repository. I didn't want anyone to be able to see my code, especially the authentication code.

I have also created a new repository for Bobo and pushed the Bobo project to it. Now that the Bobo project is successfully pushed to GitHub, I have added my supervisor Dominic as collaborator. He is now able to view the development of the Bobo app as it progresses.

24th February 2017

Now that I am able to retrieve a taxi drivers details on the app by entering the drivers registration or license number, I got to work on how the information can be displayed. I spent two days trying to get the taxi info to be displayed in a popup, the popup would appear after the user presses the check button. I was able to have the popup appear once the button was pressed but the taxi info wouldn't appear. I realised I was wasting too much time on this so I have decided to come back to this later when the main requirements are completed.

1st March 2017

I found it hard to get time to focus on my final year project when I have two other projects to work on and I have an upcoming exam in data mining and visualisation. I am trying to get the Cloud application development project done as soon as I can so I can have more time to focus on the Bobo app.

2nd March 2017

I was able to add a login and signup functions to the Bobo app today using Firebase authentication. This is a great way to add login and signup to an android app but unfortunately the signup function just requires the users email address and password. I will need a lot more details from the user when signing up. For Example, I will need them too add at least two contacts.

6th March 2017

Today I began to study and experiment with firebase authentication code to add more options to the signup page which will create and new users table in the firebase database.

6.2.7 Reflective Journal #7 (Mar)

23th Feb 2017

After careful consideration, I have decided to change the login and signup screens. The reason I have done this is because the current Login and signup functions were not reliable and limited me to just use Firebase's authentication user interface (AthUI), this resulted in not being able to do any customisation. Even though AuthUI was easy to implement into the Bobo app, it didn't look very safe as it just looked like an overlay and it was slow. So I began researching for a better, safer and a more customable solution.

<u>9th March 2017</u>

Nearly two weeks from when I decided to change my login and signup functions and I have finally completed developing from scratch a new login and signup page. I have tested a multiple of different ways or types of Android authentication and found a great tutorial that took me step by step. I was happy that I took this approach and not the easy one.

<u>18th March 2017</u>

I got time today to work on the user's signup form. This is an Activity that a user will be brought to after registering or signing up. The form will ask the user for their first name, last name, age and emergency contact name and email.

<u>21st March 2017</u>

The form successfully and securely stores user's personal data in Firebase's Realtime database with the user ID. This means that only the user that stored the data can view it.

<u>26th March 2017</u>

Began work today on the contacts function to retrieve and display contacts that are stored in Firebase.

5th April 2017

Today I had a sit down with my supervisor and took a look at what I have done since our last meeting. Dominic seemed happy with the approach I have taken with changing from AuthUI. He was also happy with the user form and how it stored data securely.

We spoke about my next tasks and how I need to prioritised them. Dominic found great solutions for some of my concerns and gave me great tips.

After the meeting, I began adding a OCR (Optical Character recognition) scanner to the Bobo app. This will make it easier to enter a taxi's registration to the app for verification.