OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Project Title:

Trip Planner

Wiktor Lesiak

Department of Informatics, School of Informatics and Engineering,
Technological University Dublin

Submitted to Technological University Dublin in partial fulfilment of the requirements for the degree of

Bachelor of Science (Honours) in Computing

Supervisor:
Marie Brennan

May 2019

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Degree of **Honours B.Sc. in Computer Science** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed: Wiktor Lesiak Dated: 05/05/2019

Abstract

This project aims to tackle the problem of creating a android-based App to plan cycling trips. The problem to be solved was how to create an app that can calculate the calories burned before the trip. In this report a description is presented of the specification and analysis of the problem, the review of relevant research conducted, and the life cycle of the system that will be developed to solve the problem.

The App will be developed in an object-oriented programming language Java using the Firebase database to store the data, the calculations will be based on Basal metabolic rate (BMR) using the information provided by one user on their profile page.

Key features of the developed system are GPS tracking using the google maps location and the calorie burning calculations.

The result of this project will result in a and mobile application that can be used by cyclist to plan and track their cycling trips, help to provide the information about the nutrition intake. It also can be used by anyone that is planning or run. Based on the calculation user would be able to know how many calories they will burin during the workout. GPS will help to track the user’s trip and store it in the profile, so he can use it any time they want.

Contents

List Of Figures

Chapter 1: Introduction

1.1 Background

The growth of smartphone users increased dramatically over the past few years from 2.1 billion phone users in 2016 to 2.5bilion in 2019, in which google shares that there are over 2 billion android active mothy users in 2017 *[1]*

There is a vast range of cycling/fitness apps which allows users to perform a monitoring, GPS tracking but usually they are limited, cost money or require subscription for full potential of the app.

The main job is to develop an app that will be appropriate for cyclist that will plan their trips for a specific distance, the other major feature in the app will be to calculate the amount of the nutrition intake that is needed based on the distance planned beforehand.

Most of the cycling applications in today's market are fully based on the map/GPS side and doesn't allow for the fitness part which would let users to track their calorie intake during the trip planning.

1.2 Main research question(s)

- What are the features that are used in the most popular cycling fitness apps?
- What is the databased that will be used to develop this app?
- Can you calculate calorific burning?
- Can you set up GPS and location on the app?
- What are the features that users dislike and why?
- Which features are missing in the popular apps and which features would users like use?

What do popular apps have that other apps don't have, and what do people like and dislike about those apps.

All the apps that are available in the market have their own advantages and disadvantages based on user experience, some people might like something that other doesn't, the option may vary and finding the perfect features is difficult.

1.3 Feasibility

The IDE that I will be using during this project will be Android Studio and the technologies, languages and tools required to complete this project are:

• Android: Java (primary development language), C

• Database: Firebase(primary development database), SQLite(maybe).

For this project an Android was the chosen OS, it is more popular, and more users will have access to it. Also programming in Android is more suitable because it uses a common languages like java. The figure below illustrates difference between Android, iOS in Operating System shares. [2]

1.4 Expected Results

The aim of this project is to develop a completely functional mobile application that will allow users to plan a trip based on GPS location, users will be able to set initial starting point and a final destination plus wind speed.

Based on the distance application will calculate the average calories burned during the trip and a calorie intake that should be consumed before the trip. Users will be able to track their calorie intake and set the weight goal.

Many factors come into the calculation, before starting on how many calories does a person burn on a bike we first need to know how much daily calorie intake a person needs. Basal Metabolic Rate (BMR) calculation shows how much of the calories you need to take daily based on your lifestyle and activities. [3]

Chapter 2: Literature Review

2.1 Overview of fields reviewed, and sources consulted

Nowadays, due to the popularity of end users, the mobile developments for smartphones have evolved tremendously.

In today's world almost, every person in the world have a smartphone and right now there are millions smartphone users and the applications that are downloaded are used throughout the world every single day.

Three different app stores that offer free or paid mobile applications (e.g., Apple App Store, Android Google Play and Windows Phone Store). *[4]*

Larger development companies are investing money into a mobile app development. The mobile phones can be used in any purpose It can be used in a various of ways from simple utility apps like camera, calculator or flashlight to more advanced utility apps like dictionary, car navigation, or even a tv remote.

For a mobile app market and large companies one of the key components of a solid app strategy is the knowledge of a specific demand for categories of apps. A lot of data is needed to know this kind of insight, the following graph form 2018 comScore research shows the percentage of the time people spend with mobile apps from various apps categories. *[5]*

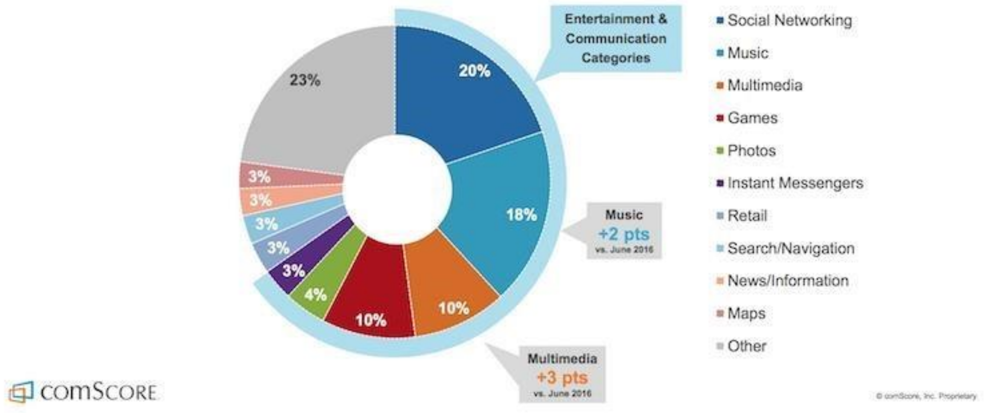


Figure 1 - Time Spent per Mobile Category (Joorabchi et al., 2013)

Many difficult questions come into play when developing a mobile application e.g., which platform should be used to develop the application. Android and Apple's iOS are the most popular platforms around the globe, and it is a difficult to choose one platform over other. This app will be based on Android platform, which causes the use of the app to be limited to the Apple's iOS users.

2.2 *Software Engineering Challenges*

2.2.1 Global Market

As we know Smart mobile devices are one of the fastest growing computing market platform in the world with estimated over 3.5 billion unique mobile internet users in 2017. (Source: Statista) This rapid increase of mobile devices alerted the marketing and is utilized for social, entertainment, business, gaming and productivity.

Allowing for wireless global positioning GPS, various connectivity, build in browser, and other kind of seasons allowed for mobile application development to provide rich, content-aware.

That combination of easy access, computing power, use of sensors and easy access to all sorts of application on the market has made mobile application the new computing development platform. [6]

2.2.2 Hardware

Many different platforms with different operating systems and different versions are available in the market, the hardware may differ from different hardware makers for the platforms. Android versions that can be found of Google, Samsung or HTC. That's why making an app that will be able to satisfy all the platforms and fully utilize the hardware that is available on the device may be rather challenging. [7]

It is crucial to build an application with good UI; mobile application should be easy and simple to use so the user has the best experience. Creating an application with easy and clean UI will create less confusion from users.

2.2.3 Maintenance

A lot of work comes to initially designing and developing but many app owners often forget to maintain their applications. Mobile applications require server that can hots the application's API, assets (images or videos), and database. The best way to host the app is to use cloud-based environment like Amazon Web Services (AWS). Depending on the size of the application the cost may vary. [8]

2.3 Health and Fitness Apps

Fitness apps are beneficial to those that don't know where to start and don't know what to do. Physical activity is important for both physique and metal health. Physical activity in a workplace increases the of efficiency and productivity.

Despite all the benefits that comes with physical activity, a lot of people still choose the sedentary lifestyle. The inactivity is still a huge prevalence today across an all the difference ages. There are many reasons for sedentary lifestyle including limited access to facilities and time, large work load, or no physical activity motivation. [9]

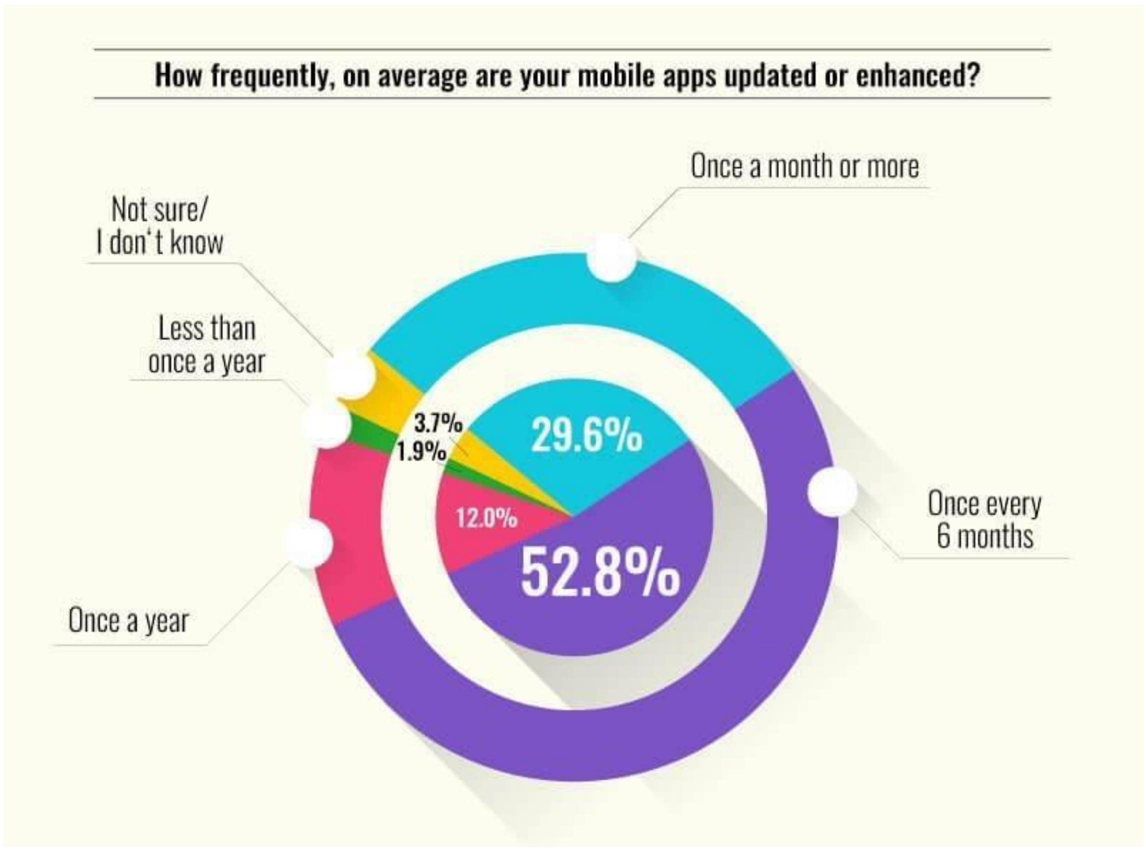


Figure 2 - How frequently are mobile apps updated? (Wasserman, 2010)

Mobile app encourages and motivate people to exercise, setting a goal in an app or organizing and planning workout program helps in consistent and regular exercises. Keeping track and record your key workouts on the app and competing with yourself and beating your best time will increase the desire for improvement.

2.4 Conclusions of review

Smartphone users have increased dramatically over past few years, with that many users there is a large market for phone apps, which may be challenging to create an app that competes with over 2 million apps that are available in the app store.

Creating an app that will satisfy users with the look and the functionality is rather expensive and time consuming. Simple apps can be created using the web application method which is easy to develop and compatible with all platforms

Developing an app for a large variety of platforms and operating systems might get challenging as they all differ is the size or hardware. Users might bet limited due to the platform in which the app was released and the requirements that are need for the app to function property. No app is perfect and maintaining is important as an app developer.

Chapter 3: Method

3.1 Methodologies

During the research of phase for this project there was a number of steps that needed to be completed to fully and successfully complete the application.

- First step was to research the most popular and similar applications on the market, and what application features are important to the users, and why users like them.
- Another step is to plan a design the look and feel of the app. The aim was to develop an application to reach large number of people by making quick and easy to navigate.
- Implement a GPS to the application and, a backend system to calculate the calorie loss and intake based on the distance traveled.
- The next step is to develop a prototype, and the final step is app testing.

There are three main popular development models like Agile Development which evolve through cooperative effort [10] Waterfall Development usually considered as more linear which flows in one direction based on phases [11] and Spiral Development known as risk-driven process, it guides a team to adopt elements of one or more development process models [12].

For this project the best fitting model would be the Waterfall Model, this model is a traditional software development model. The main aim of this model is to go through the phases and completing them fully before moving to another phase. It allows for better time management and it ensures that the application is being right direction to completion.

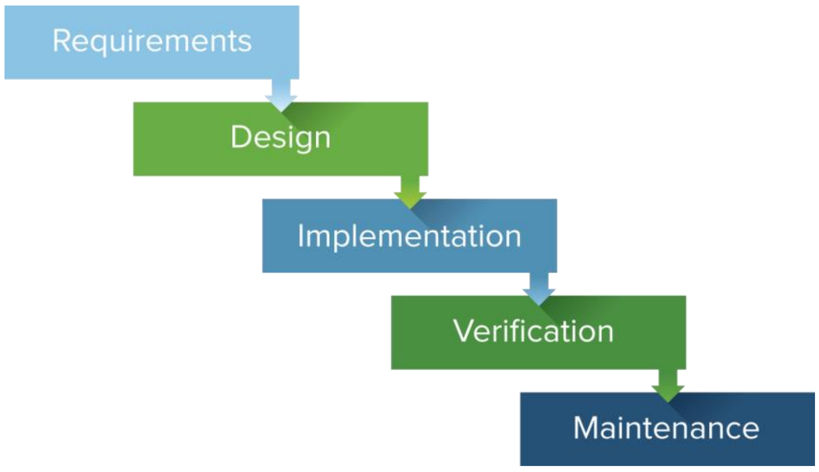


Figure 3 - Waterfall model

3.2 *Web vs Native vs Hybrid*

There are three main methods of development in mobile application: native, web, and hybrid.

3.2.1 Native application Method

The native mobile applications are the most common and this is the method that was included in this project. It runs on the mobile device and on its operating system, it provides compatibility for all the available user devices. This method allows developers to fully exploit the abilities of the user’s device, in which they have the better performance, availability and maintenance via platform’s dedicated app store.

Creating the native application method for many platforms requires high programming skills, and it tend to be more expensive and time consuming. *[13]*

Advantages:

- Better performance optimizations and fast response because applications are built on the specific platform
- They are distributed dedication platforms in application stores
- No internet connection required, depends on the applications functionally

Disadvantages

- Requires high programming language skills
- Expensive
- Less useful for a very simple app

3.2.2 Web application Method

Mobile Web application is the second method, it is created to run on the web browser. It is available through mobile web browser. This method is Hight portable across multiple mobile platforms in which it is decreasing the cost and time of the development. However, this method is very limited, using the web browser restricts the app of using the hardware and the device features like camera or accelerometer. *[14]*

Advantages:

- Easy to develop
- Easy to maintain
- An Inexpensive build processes
- One application can be available on all platforms (browser is required)

Disadvantages:

- Requires Brower to run
- Requires extra step to type the URL (poor user experience)
- Low performance
- Cannot use device hardware features.

3.2.3 Hybrid application Method

The third method is the Hybrid mobile applications, those tend to be stored on the mobile devices, and it is stored within the browser control of a platform. These applications can use the hardware that is available on the device and can be downloaded over the application distribuend app store. *[15]*

Advantages:

- Easier to build based on HTML/JavaScript and CSS
- Cheaper than native
- No browser required
- Can access hardware features of the device
- Faster to develop using single code base.

Disadvantages:

- Slower than native
- Less interactive than native
- Dependent on a third-party platform

The decision to develop either a native, web or hybrid mobile applications can be based on the requirements, skills and time. There are some questions you need to ask yourself before developing an application: *[16]*

- How fast is your application needs to be?
- What's your budget?
- What's the quality of users experience this app should have?
- What's the difficulty of the features you will work on?

This project will be build using the native method which outstands the other apps in both performance and the user experience. Native apps are easily accessible on all the and can utilize all the user's device hardware features in which allows for better, and more features to be used.

Chapter 4: System Requirements and Specification

4.1 User Requirements

There are serval tasks that user need to complete in order to be able to complete and make the app fully functional. They will need to navigate throughout each section of the app. To do this they need an easy and clear user interface design and clearly explained functions and of tasks to be completed.

Login Screen – This screen will allow user to login if they have an account if not they will be able to create an account.

Registration – If user have not account, they will be able to create one using their email address

Forgot Password – if user forgets the password, they will have a chance to change it using the email address that was provided during the registration

Profile – In profile page user can change password, remove account and set up their profile details.

Home screen – home screen should be easy to navigate and clear to read it allows user to go their profile, start new ride or log out.

Start Ride – Allows user to select initial starting point and destination based on GPS.

Saved Rides – Rides that was saved by user will be stored in their profile of reuse.

Navigation menu – navigation button on the top of the app or bottom of the app.

4.1.1 Who is the app for and what does it do?

he app is designed for users who love to cycle, run or walk. Even though the application is focused on the cycling, users who exercise by running or walking can use the app. The app calculates the lost calories over the distance, and the amount of calorie that will be burned during the exercise and shows the nutrition's that are needed to provide the body with enough energy to get through the distance that was set as goal.

This app needs to be available to users of all age. It should be easy to navigate and use, clean look of the app allows for easy access for users of all ages. .

The database will hold information like: users age, weight, height, username, password, starting point and destination. Database will collect data from GPS: distance, time. This data will allow for daily recommended daily calorie intake, and calories that will be burned during the trip.

4.2 Calculations

BMR Many factors come into the calculation, before starting on how many calories does a person burn on a bike, we first need to know how much daily calorie intake a person needs. Basal Metabolic Rate (BMR) calculation shows the amount of the calories you need to take daily based on your lifestyle and activities.

You can calculate the BMD based on those calculations:

- Men: (13.75 × weight) + (5 × height) - (6.76 × age) + 66
- Women: (9.56 × weight) + (1.85 × height) - (4.68 × age) + 655

The BMR is based on activity and how much exercise you get daily:

- inactive and have rather no exercise BMR x 1.2
- active 1-3 days/week BMR x 1.3
- active 3-5 days/week BMR x 1.55
- active most days BMR x 1.725
- active everyday BMR x 1.9

To calculate the calories burned per minute during the trip we need to know the Metabolic Equivalent of Task (MET)/h, person's bodyweight and time it took/going to cycle. [9]

(MET * (weight) * 3.5 / 200) * (time of activity)

For example, person who weights 80kg wants to cycle at standard cycling pace of 7 Mets/h which is about 16.3~19.2km/h for 1 hour and 40 minutes.

How many calories burned?

1hour and 40 minutes = 100 minutes

(7 * 80kg * 3.5 / 200) * 100 = 980 calories burned

Chapter 5: System Design

5.1 Choosing an operating system

As mentioned in 1.3 the operation system that was chosen for this project was Android which is the most common and most used operating system. Android is created by google and its based on modified version of Linux Kernel. Android is used on various devices such as tablets, smartphones Wear OS, TV, also used on some of game consoles, cameras and PCs.

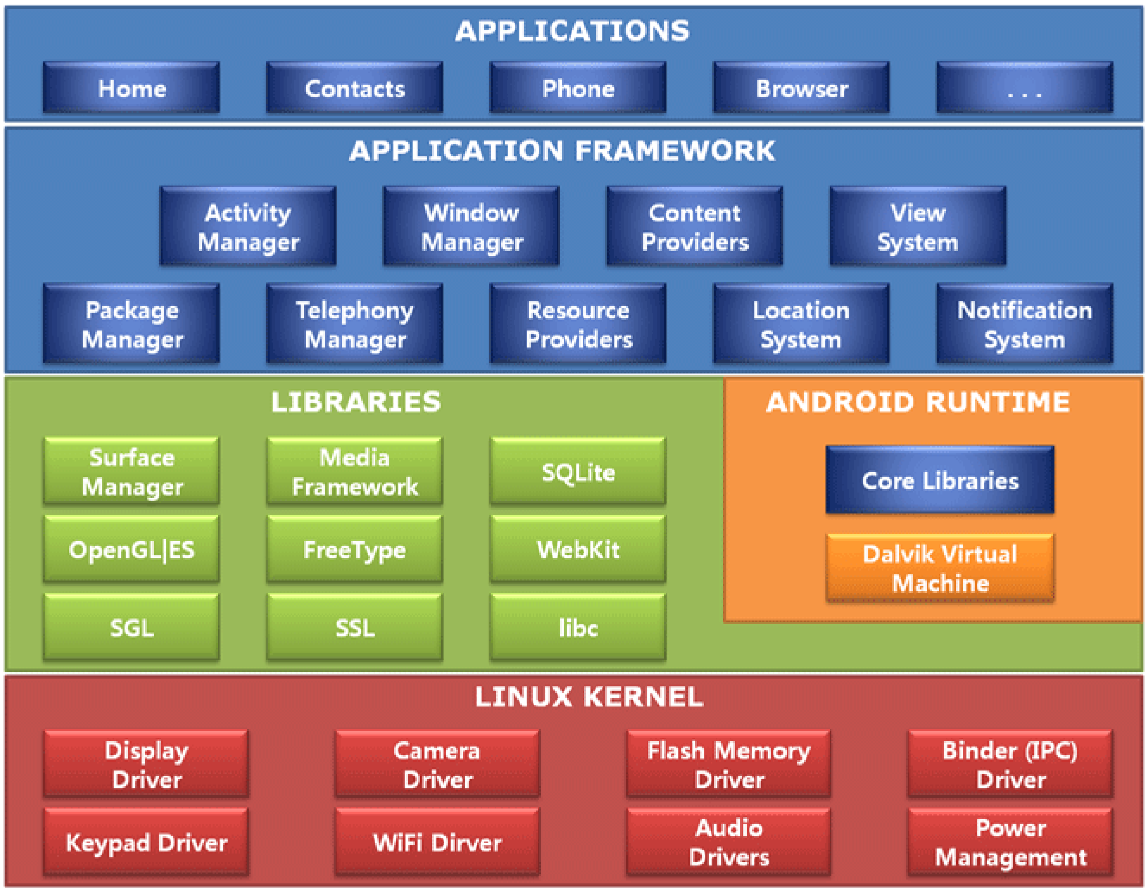


Figure 4 - Android Structure

5.2 Application Design

Finished application should look clean, neat and easy to navigate through. As comes witch navigation the user should be able to see on which page he is currently and everything should be clearly visible whiteout being lost. Pages should be easily noticeable with correct font without user having to remember where pages are located.

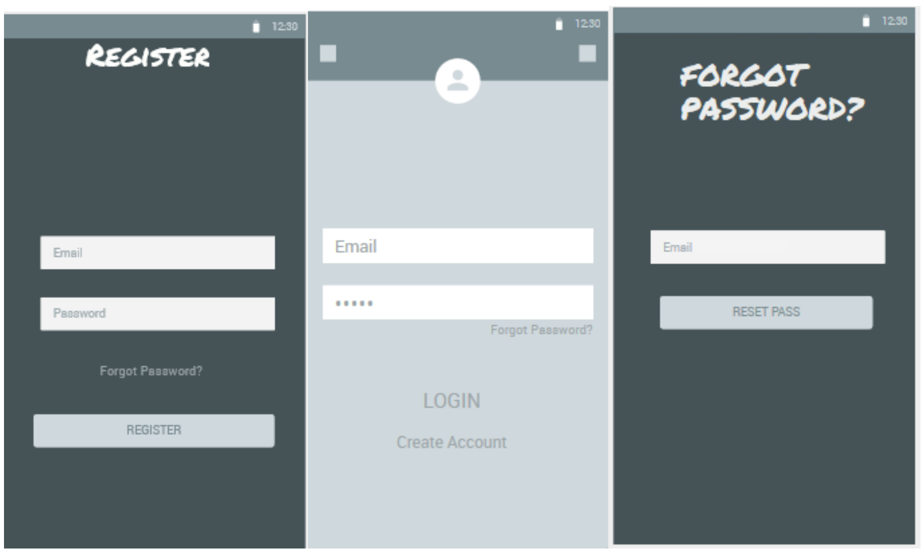


Figure 5 - Wireframe - Login/Register/Forgot Password

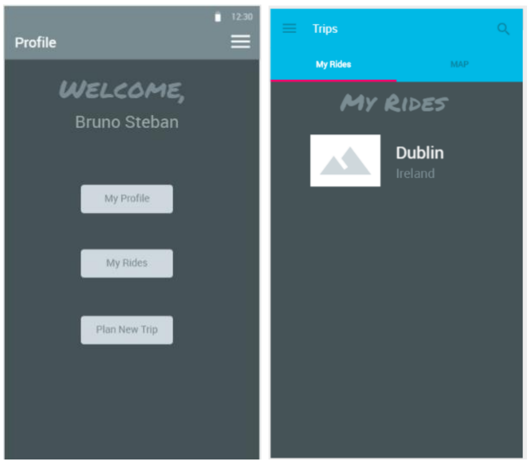


Figure 6 - Wireframe - Home Screen/My Rides

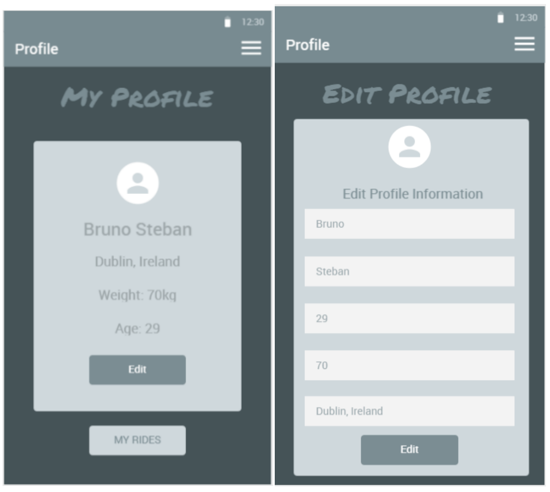


Figure 7 - Wireframe - My Profile

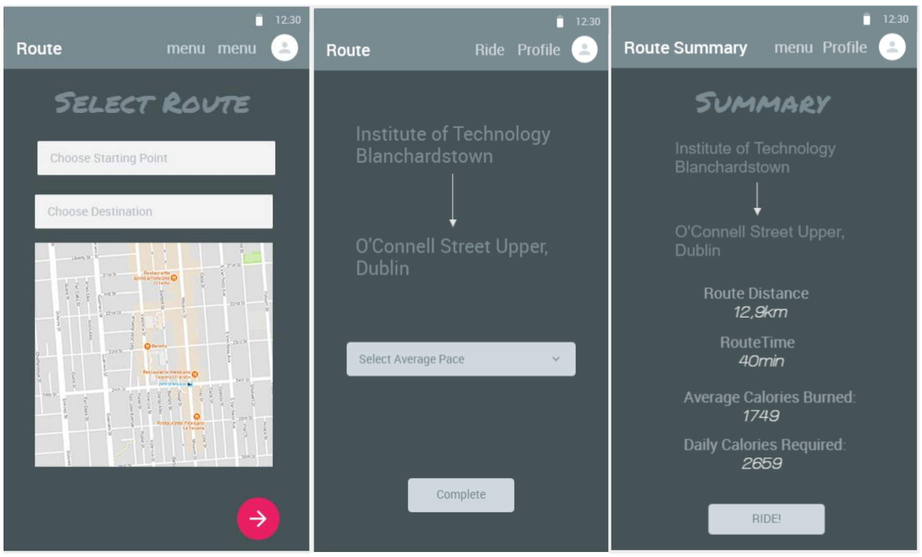


Figure 8 - Wireframe - Select Route

Register allows for creating an account based on user email address. Login Screen allows user to login using their email address. If user forgets the password, they can use the “Forgot Password’ which will allow them to reset the password.

In the home screen you will be able to go to the Profile page see your “Previous Rides” and “Start new Trip”

On your profile you will be able to edit your information which will be stored in database and used in the calories calculations

User will be able to set initial starting point and a destination, based on the distance application will calculate the average calories burned during the trip and a calorie intake that should be consumed before the trip. Users will be able to track their calorie intake and set the weight goal.

5.3 UML Diagrams

Unified Modeling Language (UML) is a rich language model software solution it helps with app planning, application structures and system behavior between the front end and a back end of the software or business processes. Creating an UML Diagram in an early stage of a software enjoining project helps with understating the structure of a program (app in this case) and allows for better programming in the late stages of the project.

5.3.1 Activity

Activity Diagram shows user interactions with the app and where does each activity lead to. The diamonds represented in the diagram displays user decisions, what activity they will go to next. and where they will finish. This Diagram is really helpful when programming the app, knowing the structure based on user decisions helps to understand the application and how it supposed to respond to each user decision.

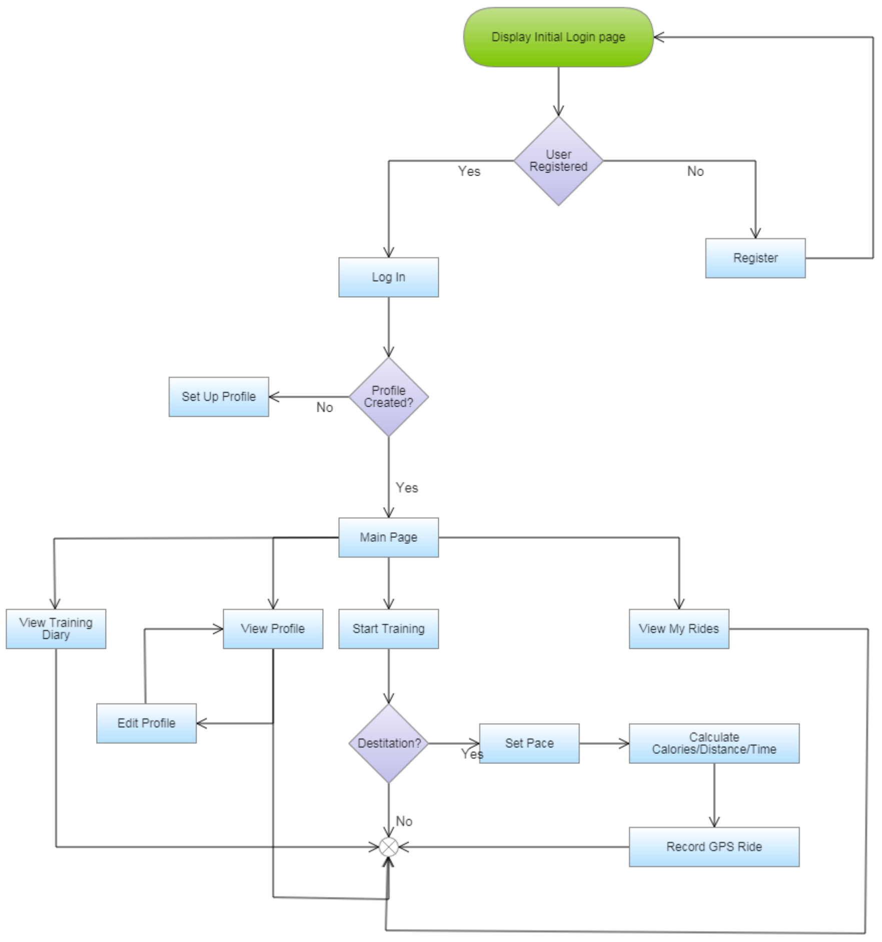


Figure 9 - Activity Diagram

5.3.2 Sequence Diagram

This diagram shows interactions arranged in a timed sequence from top right to down from the objects and classes involved in the application. Different stages represent the messages exchanged between user and a backed of the program that needed to carry of the functionality of the app.

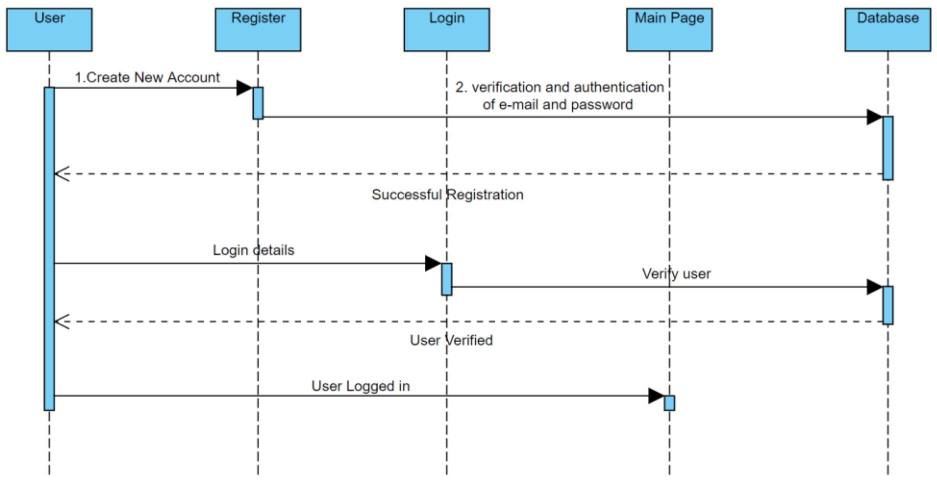


Figure 10 - Sequence Diagram

5.3.3 Use case Diagram

A use case diagram is represented with users' actions and relationship with the system it displays the different use of the cases in which the users is involved. It gives a closer look with interaction for a user (the actor) within the app (the system).

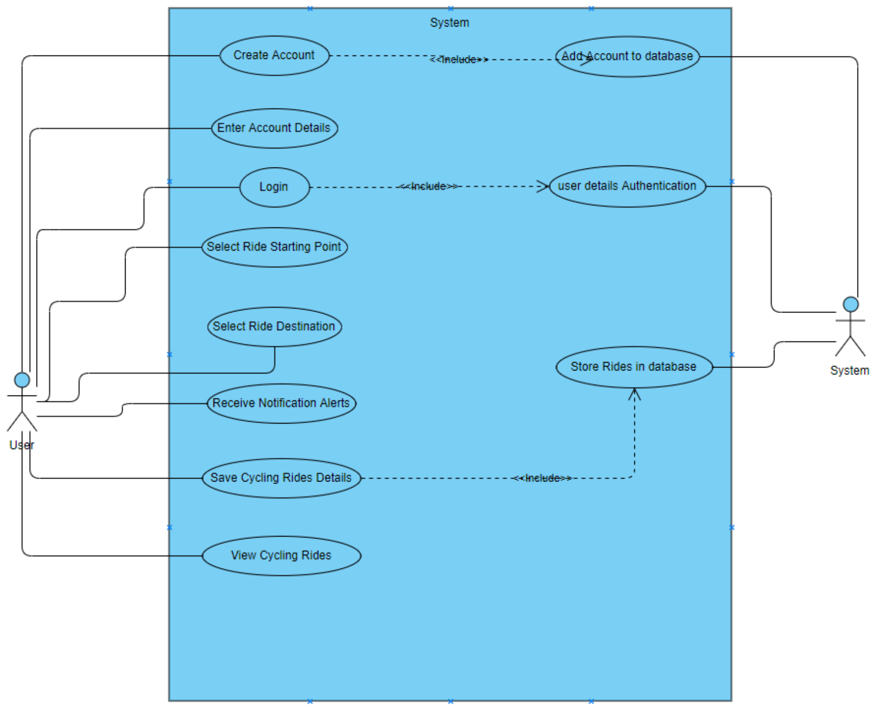


Figure 11 - Use Case Diagram

5.4 Logo

I created the logo using Photoshop. I designed this logo to match the theme of the app. U choose green colour because this colour represents the nature and healthy lifestyle. The logo design changed to match the design of Application.

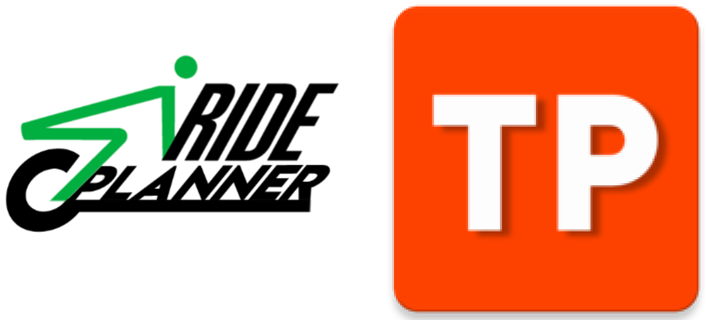


Figure 12 – Old Logo(left) & New Logo(right)

5.5 Database Design

Data that is not hardcoded or stored in code is usually stored in a database. It is a key component when designing an app or any other software project that requires user information or dynamic storage data that can be stored and accessed anywhere in the project. There any few options to consider then deciding for database in Mobile application development

5.5.1 SQL or NoSQL

One of the options is to use Not Only SQL (NoSQL) which is cloud based meaning it can be easier to set up and access the data because the data is stored in real-time. NoSQL it stores data differently. This SQL has no schema, so it be more flexible in data storage. Because of the unstructured nature of NoSQL, it can be easy scaled without redesigning the whole structure of database.

The other more old and traditional option is to use Structured Query Database (SQL) which is more structured, and rule based based on Database Management System (DMMS). The data is stored in tables where information can be related to other tables. Set up of SQL require more work and requires to be stored on a server.

Decided solution for this project was to use Real-time Firebase Database (NoSQL)

5.5.2 Firebase Realtime Database

Firebase Database is based on could-hosted server. The data is synchronized in real-time to every client and the data is stored in JSON format. Firebase and Android Studio are owned by Google, so it makes great compatibility for an Android Application development.

Key capabilities for firebase database are:

5.5.2.1 Realtime

Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.

5.5.2.2 Offline

Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is re-established, the client device receives any changes it missed, synchronizing it with the current server state.

5.5.2.3 Accessible from Client Devices

The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there’s no need for an application server. Security and data validation are available through the Firebase Realtime Database Security Rules, expression-based rules that are executed when data is read or written.

5.5.2.4 Scale across multiple databases

With Firebase Realtime Database on the Blaze pricing plan, you can support your app's data needs at scale by splitting your data across multiple database instances in the same Firebase project. Streamline authentication with Firebase Authentication on your project and authenticate users across your database instances. Control access to the data in each database with custom Firebase Realtime Database Rules for each database instance.

5.6 Conclusion

In the design chapter we’ve looked on what was the best system for out application, and the structure of the android app and how each of the layer interact with each other. We looked at the wireframe design of the project and how to implement this theme throughout the app.

We looked at the UML diagrams to give us clear vision on the project backend and client end.

We then looked at the Database options and how it works with the app. This stage helps implementing of the application

Chapter 6: Implementation

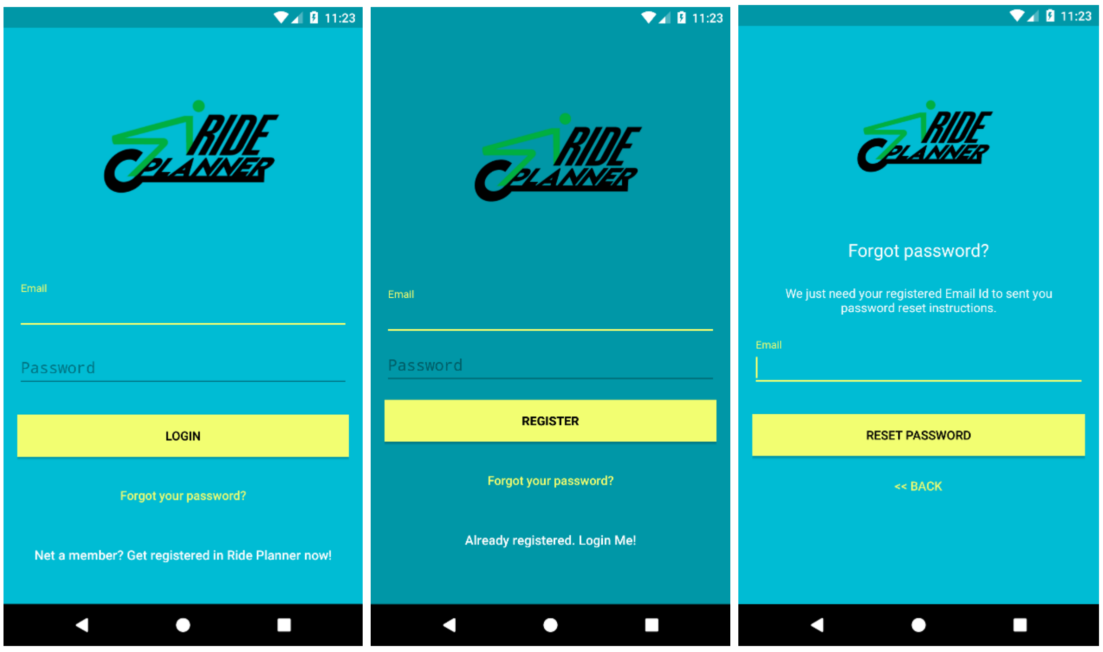
We looked at the design and the structure of the application in chapter 4. We also looked on how the user can interact with the app using the UML diagrams and how user can move through various of screens and menus.

In this chapter we will investigate implementation of the end-product which is fully functional application. We will examine components that make android application, how we will use those components in out application and what prosses went into making each stage of application complete.

6.1 *Prototype*

The colours that were used for the theme of this application are contrasted nicely by using bright and simple design. Buttons what are yellow nicely stand out from the background and are simple to navigate. Font colour clearly stand out making an easy to read. Logo as mentioned 5.3 was created using photoshop.

Login Screen / Register / Forgot Password



We just need your registered Email Id to sent you password reset instructions.

Figure 13 - Prototype - Login Screen/Register/Forgot Password

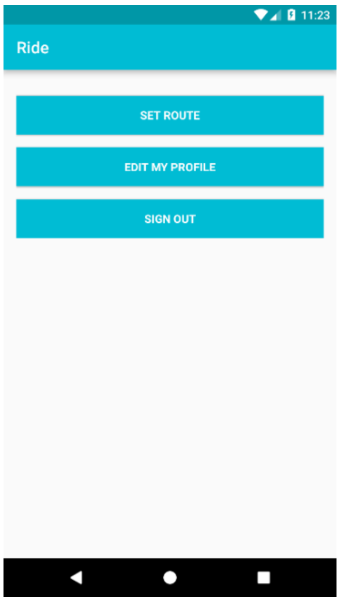


Figure 14 - Prototype - Home Screen

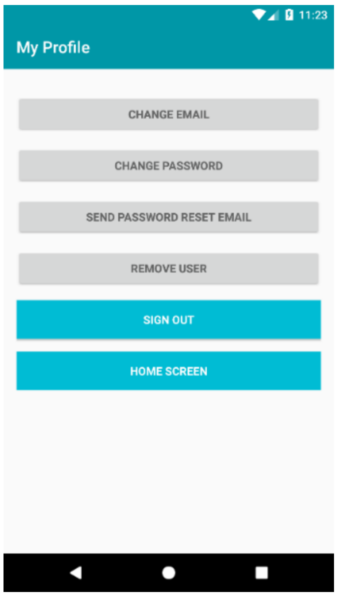


Figure 15 - Prototype - My Profile

The Login Screen allows for user to login using email and password, the register allows to create a user account using email and password to verify.

If user forgot the password, they can send the forgot password verification to an email address.

Home allows for navigation to the main activities and features of the app. You can access the “Set Route” which allows for creating new ride, the “Edit My Profile” will display the my profile activity, and sign out button will log out user from their account and set the screen to Login Activity.

On my profile you can change email and password that was provide during the registration. You can also delete your account. “Home Screen” button redirects user back to the home screen.

This page allows user to start new ride or look at the previous rides that they have saved on their profile account.

Firebase

Firebase is a Realtime database used in mobile applications which has 18 products one of the products that were used in this application is Firebase Auth which allows for the login using the email address which are stored in a Firebase database

The real-time database is a backend for an app, to store and synchronise data on the Firebase cloud. Also allows for push notifications from the server.

6.2 Getting Started

Before we start to program our application, we need to look on which languages and tools do we need. To start developing with android is quite simple compared to other operating systems. No external tools are required, and the IDE software is made by Google and free to download and use.

Kotlin and Java are two languages that are available to write Android Applications in Android Studio. As mentioned before Java will be the language used in this project. It is the most popular language used in computer programming it may get tricky initially, but overall good basic understanding is enough to start developing Android Applications. The layout is written in XML language which is easy, and the Android Studio GUI will do most of the work.

6.3 Tools & Setup

In order to start creating android application we will need to have Java Development Kit installed and up to date and installed Android Studio IDE. It contains a lot of options when designing the app and testing. Using Android Studio, you don’t have to type everything out because it will suggest a lot of options during the coding.

We can download Phone Emulator with android via Android Studio we can choose which model of the phone and version of the operating system we would like to have on the phone. The setup of the SDK might be quite long.

Using the Emulator is pretty hand and fast but it has high consumption and It can slow out machine so the physical device is crucial for testing and compatibly because one thing might work on the emulator device and it might not work on the physical device.

The emulator comes with many different versions of android and different phones, so it is good for testing for many devices with different operating systems and different screen sizes.

Physical devices benefits form knowing how the device does will feel in out hands, and helps us decide if the design was poorly made for one handed use because user experience is important in application development.

6.3.1 Structure of an Android App

When creating android project android studio creates environment for an application it creates files that we will use further into the project

Steps involved intro create android studio project are listed below.

Open Android Studio

Select New Project

Select type of activity

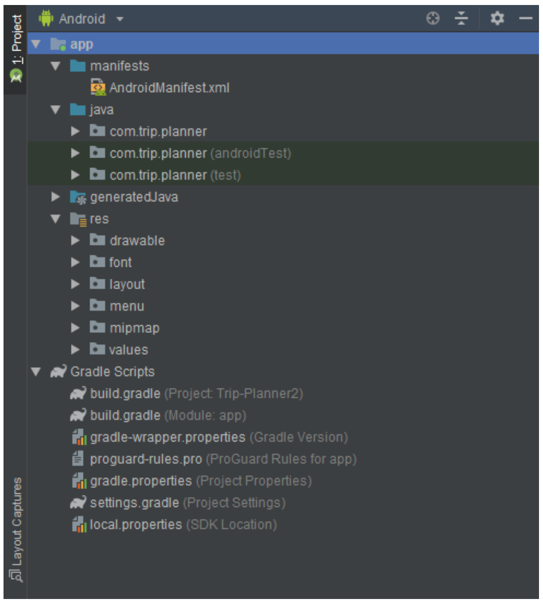
Enter name of the project, package name and choose where to save

Select version of the API and language

Select name for activity

And you're finished.

Fig. below represents android studio structure



```
Android
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▶ com.trip.planner
    ▶ com.trip.planner (androidTest)
    ▶ com.trip.planner (test)
  ▶ generatedJava
  ▼ res
    ▶ drawable
    ▶ font
    ▶ layout
    ▶ menu
    ▶ mipmap
    ▶ values
▼ Gradle Scripts
    build.gradle (Project: Trip-Planner2)
    build.gradle (Module: app)
    gradle-wrapper.properties (Gradle Version)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

Figure 16 - Android Studio Structure

6.3.2 Manifests

It is an XML file that tells what's inside the app, contains name of the app, version, domain name, package, user permissions, a list of out activities and logo. It also tells which activity to start first

6.3.3 Java

Folder where all activities are stored as the name of the folder indicates it contains java files

6.3.4 Res or Resources

Contains resources that the app needs to display there are usually JPEG or PNG files. Drawable folder contains main icons and image that are used throughout the app e.g. background pictures, icons, buttons images and main app logo etc.

Layout folder stores all activity layouts files they are XML files telling the location of all elements on the screen.

Menu contains XML and all the information about the menus we’ve added to our project

Mipmap folder contains all of out icons shown in all different resolutions, when uploading icon to android it creates different resolutions for the icon such as MHPI (160 DIP), HDPI (240 DPI) XHDPI (320 DPI), XXHDPI (480 DPI) and XXXHDPI (640 DPI).

Lastly the values contain all the strings, dimensions, style documents and colours that are also stored in XML format

6.3.5 Grade Scripts

Grade files indicates that version of the SDK is using, all libraries, and dependencies.

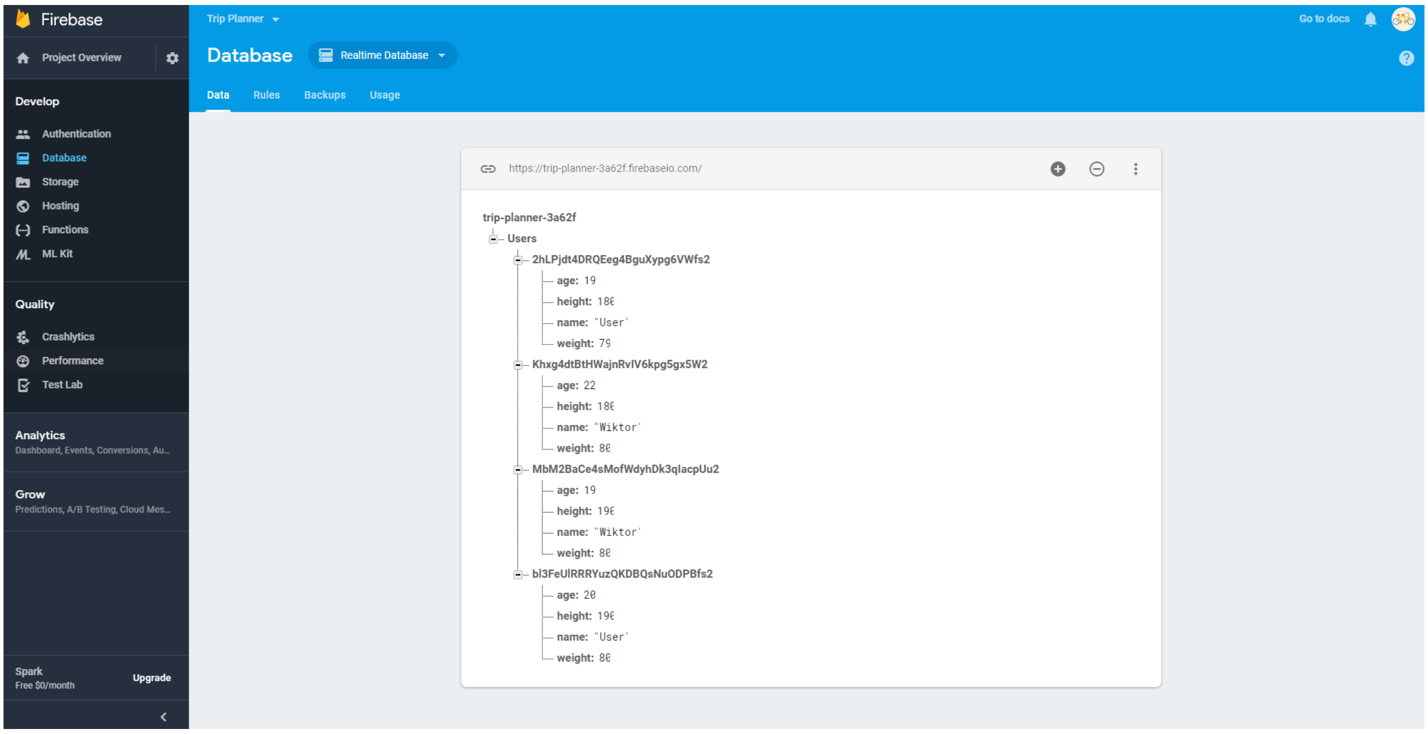
6.4 Project implementation

Last section we looked at the structure of the project and how each of the folders and components work in application. In this section we will look at the coding aspect of the project and describe all activities that are created in our application.

Firstly, we need to look on the database and describe how does it communicate with the code. Without the firebase this application would lack the dynamic functionality and it would limit our usage of this application.

6.4.1 Firebase

Here we will on look how does the database interacts with our code and the overall structure of the firebase Realtime database.



```
https://trip-planner-3a62f.firebaseio.com/

trip-planner-3a62f
  Users
    2hLPjdt4DRQEeg4BguXypg6VWfs2
      age: 19
      height: 18€
      name: "User'
      weight: 79
    Khxg4dtBtHWajnRvlV6kpg5gx5W2
      age: 22
      height: 18€
      name: "Wiktor'
      weight: 8€
    MbM2BaCe4sMofWdyhDk3qlacpUu2
      age: 19
      height: 19€
      name: "Wiktor'
      weight: 8€
    bl3FeUlRRRYuzQKDBQsNuODPBfs2
      age: 20
      height: 19€
      name: "User'
      weight: 8€
```

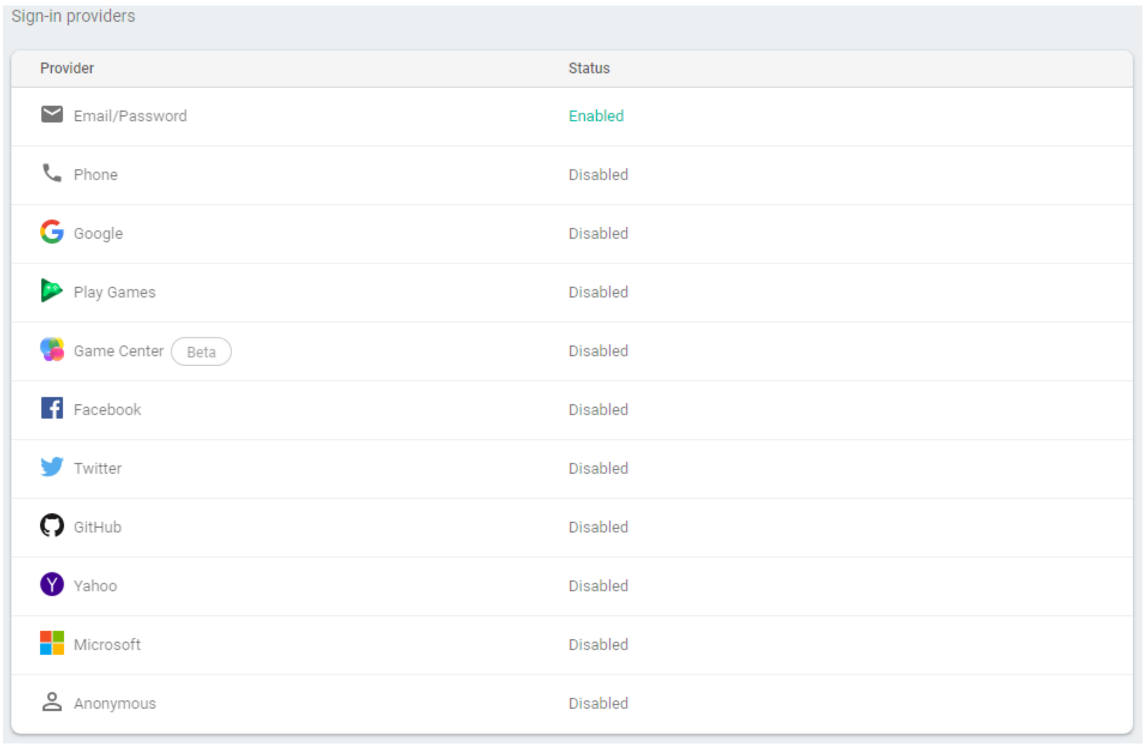
Figure 17 - Firebase Realtime Database console

Firebase offers a lot of tools for developing and testing our application, as you can see there is many different options that are free of use for our application, during this project we used Realtime Database and Firebase Authentication.

The fist Firebase feature that we implemented in our final application and in our prototype as mentioned in section 6.1 was Firebase Authentication. This feature allowed us to Authenticate users by creating an account using email and password. This method helps to prevent unwanted bots spamming our application and unwanted uses targeting the app. It also allows us to assign unique ID to each of the users that create account so we can separate all of the users and assign data to each individual user profile.

There any many sign in Authenticator options that are available in firebase. In this project we used Email/Password Sign in option as it is most popular and easiest option. Not all the users have accounts in e.g. Google, Facebook or Microsoft, that's why the email and password authentication is most compatible and most popular around the users.

Below figure shows all the Sign In providers that are available in Firebase Authenticator



Sign-in providers

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center Beta	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Yahoo	Disabled
Microsoft	Disabled
Anonymous	Disabled

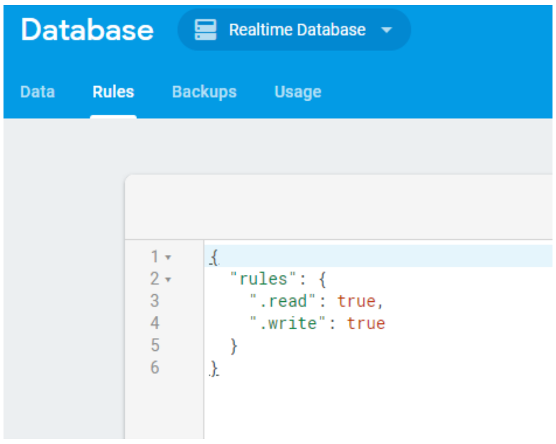
Figure 18 - Firebase Authentication providers

Using the other Sign In options for authentication is useful and fast because users don't have to create an account, they just link the account with the one existing in different application. Some users might not be comfortable linking their private Facebook account to some new application. Once user has created the account using email and password, they are free to sign in using giver credential, firebase also offers "Forget Password" option that lets user change the password using given address email.

Realtime database allows us for an easy way to use database. The database is written in JSON and working in real time. One of the best features of the Firebase Realtime database is that all the data is written in real-time and all the database updates are instantly pushed to the cloud and synched in real-time this allows users to use any device they wanted without worrying about losing data. If user has no connection to the interned the app will store the data and once user is connected to the internet the app will send the data to the cloud making no loss in the data.

As shown is fig. 17 the outline of the database displays different layers of information that are stored in the database. Unlike SQL firebase has no tables, it's written in JSON format. Once data is stored it created a node in the existing JSON structure with an associated key. We can provide our own keys such as authenticator user ID that allows us associate unique data to each individual user.

Firebase database also allows us to set rules for the database allowing users to only read or write data to their own node making sure that the data is secure.



```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Figure 19 - Database Rules

In fig. 17 you can notice that each user ID key is associated with data of age height weight and name.

Firebase Console was a bit difficult to learn and set up, it took a bit of time and there was a lot of learning process involved in working with database. Discovering the database and how to store and retrieve data was quite unique and different from the other databases that we discovered before. But once learned it was quite handy and allowed us to reuse the data throughout the application.

6.4.2 Splash

The first activity that you notice when entering the app will be the splash screen. Splash screen is a graphical control element consisting of a window with an image (logo). It creates more professional look to the app. Splash screens typically serve to enhance the look and feel of an application or web site, hence they are often visually appealing.

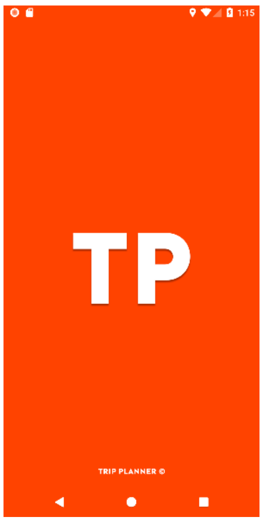


Figure 20 - Splash Activity

At the top of the code we will notice the package which we've named at the start of the application and under this there are import libraries.

```
package com.trip.planner;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.Window;
import android.view.WindowManager;
```

Then we define the class name just like in normal java program. On create is to initialise the app and setContentView(); link us to the XML file which is layout of the app.

```
public class Splash extends Activity {
            /** Duration of wait **/
            private final int SPLASH_DISPLAY_LENGTH = 1000;
            /** Called when the activity is first created. */
            @Override
            public void onCreate(Bundle icicle) {
                super.onCreate(icicle);
                //remove toolbar
                requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);
        setContentView(R.layout.activity_splash);
```

6.4.3 Sign Up

The second activity that we notice after running the app will be Sign Up screen called MainAcitivity this activity uses Firebase Authenticated for signing up options.

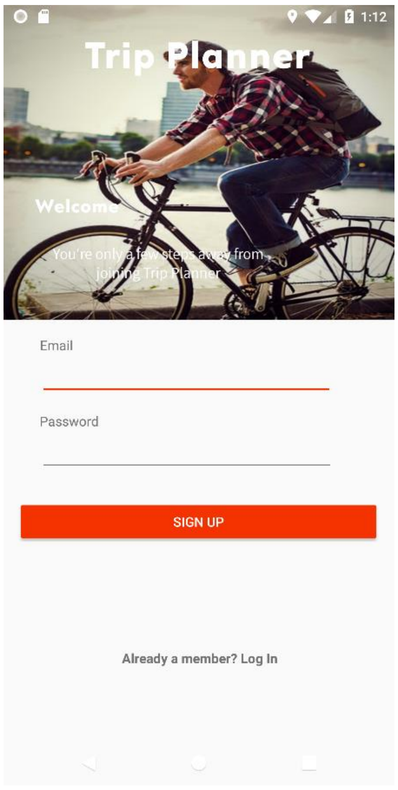


Figure 21- Sign Up Activity

At the top of the file we can see import option that are the firebase libraries. These can be standard or third-party.

```
import android.app.Activity;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ViewFlipper;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
```

After declaring the class of the activity class, we can create global variables

```
public class MainActivity extends Activity {
ViewFlipper v_flipper;
private Button buttonRegister;
private EditText editEmail;
private EditText editPassword;
private TextView textLogin;
private ProgressDialog progressDialog;
private FirebaseAuth firebaseAuth;
```

The layout consists of EditText, TextView, and Buttons

```
buttonRegister = findViewById(R.id.btn_reset_pass);
editEmail = findViewById(R.id.email);
editPassword = findViewById(R.id.password);
textLogin = findViewById(R.id.reset);
```

In code below we can see FirebaseAuth checking if the user has been already created. If the user is created, it will return toast with information that the user already exits

```
firebaseAuth.createUserWithEmailAndPassword(email, password)
public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            //user is successfully registered and logged in
            //start activity profile activity
            Toast.makeText(MainActivity.this, "Complete User Profile",
Toast.LENGTH_LONG).show();
            progressDialog.cancel();
            startActivity(new Intent(MainActivity.this, UserActivity.class));
        }else if(firebaseAuth.getCurrentUser() != null) {
            progressDialog.cancel();
            Toast.makeText(MainActivity.this, "User already exits... Try to
Log In", Toast.LENGTH_SHORT).show();
        }else{
            progressDialog.cancel();
            Toast.makeText(MainActivity.this, "Could not register... please
try again", Toast.LENGTH_SHORT).show();
        }
    }
});
```

If the details are correct onClickListener will be activated and will run the registerUser() method.

```
buttonRegister.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(v == buttonRegister){
            registerUser();
        }
    }
});
```

```
public void registerUser(){
    String email = editEmail.getText().toString().trim();
    String password = editPassword.getText().toString().trim();
    if(TextUtils.isEmpty(email)){
        //email is empty
        Toast.makeText(this, "Please enter email",
Toast.LENGTH_SHORT).show();
        return;
    }
    if(TextUtils.isEmpty(password)){
        //password is empty
        Toast.makeText(this, "Please enter password",
Toast.LENGTH_SHORT).show();
        return;
    }
```

6.4.4 Login

This activity prompts user to enter email and password they given when creating their account.

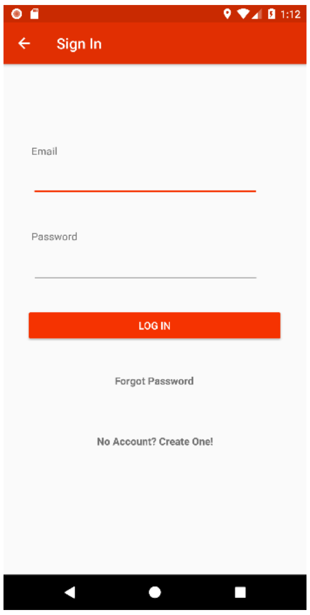


Figure 22- Login Activity

The login activity checks for user authentication if user enter incorrect password or email an error message will return, and if authentication is validated user will be moved to home activity.

```
//authenticate user
auth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(LoginActivity.this, new
OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

                if (!task.isSuccessful()) {
                    // there was an error
                    if (password.length() < 6) {
                        progressDialog.cancel();
                        inputPassword.setError("Incorrect Password!");
                    } else {
                        progressDialog.cancel();
                        Toast.makeText(LoginActivity.this, "Incorrect Email
or Password", Toast.LENGTH_LONG).show();
                    }
                } else {
                    progressDialog.cancel();
                    Intent intent = new Intent(LoginActivity.this,
HomeActivity.class);
                    startActivity(intent);

                    finish();
                }
            }
        });
```

6.4.5 Forgot Password

The forgot password activity allows user to reset their password using Firebase Authenticator. Once user enters their email. Instructions will be send to the email address with the reset password link created by firebase authenticator.

```
auth.sendPasswordResetEmail(email)
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(ResetPasswordActivity.this, "We have sent
you instructions to reset your password!", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(ResetPasswordActivity.this, "Failed to
send reset email!", Toast.LENGTH_SHORT).show();
                }

                progressBar.setVisibility(View.GONE);
            }
        });
```

6.4.6 User Details

This activity collects user data and stores in database. For each unique user ID data I collected and sored separately

User class using getters and setters to collect and store data.

```
public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public int getWeight() {
        return weight;
    }
    public void setWeight(int weight) {
        this.weight = weight;
    }
    public int getHeight() {
        return height;
    }
    public void setHeight(int height) {
        this.height = height;
    }
}
```

UserActivity collects data and stores them in Firebase Realtime Database

```
private void addName(){
    final String name = editName.getText().toString().trim();

    final String age = editAge.getText().toString().trim();
    int agetoint = Integer.parseInt(age);

    final String weight = editWeight.getText().toString().trim();
    int weighttoint = Integer.parseInt(weight);

    final String height = editHeight.getText().toString().trim();
    int heighttoint = Integer.parseInt(height);
```

```
User user = new User(
        name,
        agetoint,
        weighttoint,
        heighttoint
);
```

```
FirebaseDatabase.getInstance().getReference("Users").child(FirebaseAuth.getIn
stance().getUid())
        .setValue(user).addOnCompleteListener(new OnCompleteListener<Void>()
{
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        if(task.isSuccessful()){
            Toast.makeText(UserActivity.this, "Register Successful!",
Toast.LENGTH_LONG).show();
            progressDialog.cancel();
            startActivity(new Intent(UserActivity.this, HomeActivity.class));
        }
```

6.4.7 Profile

In User profile you can change email, change password, reset password, remove user and sign out. Layout of this activity is based on CoordinatiorLayout

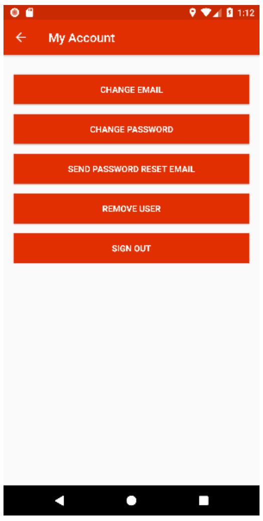


Figure 23 - Profile Activity

The change email option can be acceses using **user.updateEmail** once task is successful user will be singed out and ready to log in with their new email address.

```
if (user != null && !newEmail.getText().toString().trim().equals("")) {
    user.updateEmail(newEmail.getText().toString().trim())
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText(Profile.this, "Email address is 
updated. Please sign in with new email id!", Toast.LENGTH_LONG).show();
                        signOut();
                    }                 }
            });
```

Change password can be used the same as email using **user.updatePassword**

```
user.updatePassword(newPassword.getText().toString().trim())
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(Profile.this, "Password is updated, sign 
in with new password!", Toast.LENGTH_SHORT).show();
                    signOut();
                }
```

Password reset works the same as the “Forgot Password” activity it sends email address with instruction on how to change password.

```
auth.sendPasswordResetEmail(oldEmail.getText().toString().trim())
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(Profile.this, "Reset password email is 
sent!", Toast.LENGTH_SHORT).show();
                }
```

To remove user **user.delete()** method is used. Once user has deleted their account they will be logged out and ready to create their new account with new or the same credentials

```
user.delete()
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(Profile.this, "Your profile is deleted:(
Create a account now!", Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(Profile.this,
MainActivity.class));
                    finish();
                }
```

6.4.8 Home

Home activity is the first activity that displays after you create account. It displays user data from firebase database that was given during registration. In this activity you can select the amount to exercise you do weekly and based this information and given details during registration user will be able to discover how much calorie intake they need daily

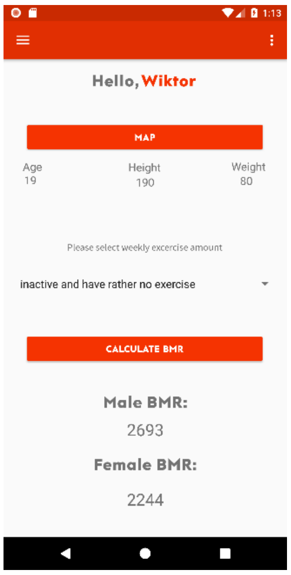


Figure 24 - Home Activity

showData() method allows us to collect data from firebase database and set TextView to given data.

```
private void showData(DataSnapshot dataSnapshot) {
    for(DataSnapshot ds : dataSnapshot.getChildren()){
        uInfo.setName(ds.child(userID).getValue(User.class).getName());
        uInfo.setAge(ds.child(userID).getValue(User.class).getAge());
        uInfo.setHeight(ds.child(userID).getValue(User.class).getHeight());
        uInfo.setWeight(ds.child(userID).getValue(User.class).getWeight());
        //set firebase data to textview
        profileName.setText(uInfo.getName());
        ageNum.setText(String.valueOf(uInfo.getAge()));
        heightNum.setText(String.valueOf(uInfo.getHeight()));
        weightNum.setText(String.valueOf(uInfo.getWeight()));
```

The exercise can be collected using the spinner which is a dropdown menu that can store values for each category using switch.

```
String names[] = {"inactive and have rather no exercise", "active 1-3
days/week","active 3-5 days/week","active most days","active everyday"};
switch (position)
{
    case 0:
        exercise = 1.2;
        break;
    case 1:
        exercise = 1.3;
        break;
    case 2:
        exercise = 1.55;
        break;
    case 3:
        exercise = 1.725;
        break;
    case 4:
        exercise = 1.9;
        break;
}
```

Based on those values we can create formula which was mentioned in section 4.2 and calculate user daily calorie intake.

```
weight = uInfo.getWeight();
//CALCULATING BMR FOR MALE
double sumMale = (13.75 * weight) + (5 * uInfo.getHeight()) + (6.76 *
uInfo.getAge()) + 66;
finalSumMale = sumMale * exercise;
String maleFormatted = String.format("%.0f", finalSumMale);
sumNum.setText(maleFormatted);
//CALCULATING BMR FOR FEMALE
double sumFemale = (9.56 * weight) + (1.85 * uInfo.getHeight()) + (4.68 *
uInfo.getAge()) + 665;
finalSumFemale = sumFemale * exercise;
String femaleFormatted = String.format("%.0f", finalSumFemale);
sumNum2.setText(femaleFormatted);
```

Using the Drawer Layout, we can access more options leaving more space on the screen.

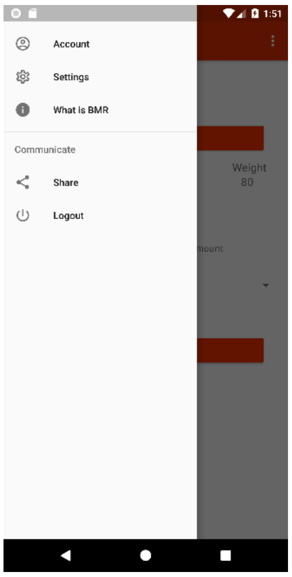


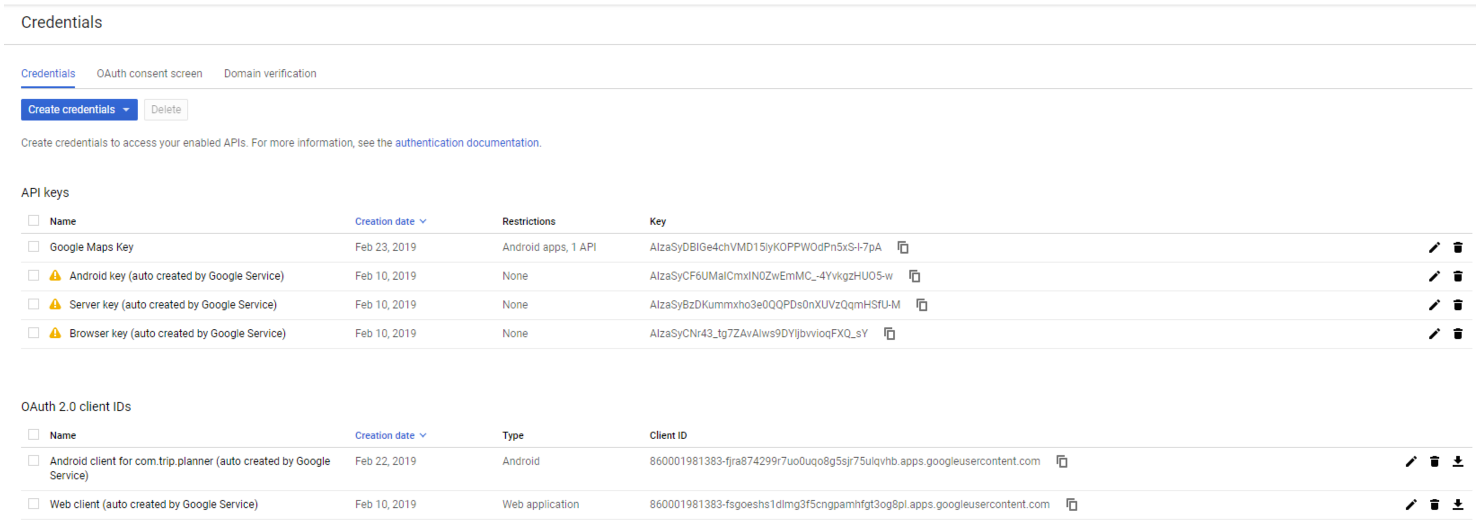
Figure 25 - Drawer Layout

When button was clicked Intent was used to send data between two activities. It allowed for communication between activities.

```
Intent i = new Intent(getApplicationContext(), MapsActivity.class);
i.putExtra("maleBMR", finalSumMale);
i.putExtra("femaleBMR", finalSumFemale);
i.putExtra("weight", weight);
startActivity(i);
```

6.4.9 Map Activity

Map Activity shows user location and allows user to select two points on the map and calculate distance between them. To connect to Google Maps API, we had to create API key


Credentials

Credentials | OAuth consent screen | Domain verification

Create credentials | Delete

Create credentials to access your enabled APIs. For more information, see the authentication documentation.

API keys

Name	Creation date	Restrictions	Key
Google Maps Key	Feb 23, 2019	Android apps, 1 API	AIzaSyDBIGe4chVMD15lyKOPPWOdPn5xS-I-7pA
Android key (auto created by Google Service)	Feb 10, 2019	None	AIzaSyCF6UMalCmxIN0ZwEmMC_-4YvkgzHUO5-w
Server key (auto created by Google Service)	Feb 10, 2019	None	AIzaSyBzDKummxho3e0QQPDs0nXUVzQqmHSfU-M
Browser key (auto created by Google Service)	Feb 10, 2019	None	AIzaSyCNr43_tg7ZAvAlws9DYljbvvioqFXQ_sY

OAuth 2.0 client IDs

Name	Creation date	Type	Client ID
Android client for com.trip.planner (auto created by Google Service)	Feb 22, 2019	Android	860001981383-fjra874299r7uo0uqo8g5sjr75ulqvhb.apps.googleusercontent.com
Web client (auto created by Google Service)	Feb 10, 2019	Web application	860001981383-fsgoeshs1dlmg3f5cngpamhfgt3og8pl.apps.googleusercontent.com

When we created the key, we had to provide the key into the values/google_maps_api.xml file so that the application could display Google Maps API. We needed app permission to use location finding. User location is marked by pink marker.

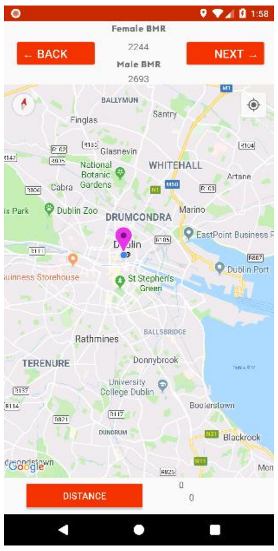


Figure 26 - User Locaion

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

To use the distance between two location we needed to acquire point a “ORIGIN” and point b “DESTINATION”. Code below displays how the markers are created

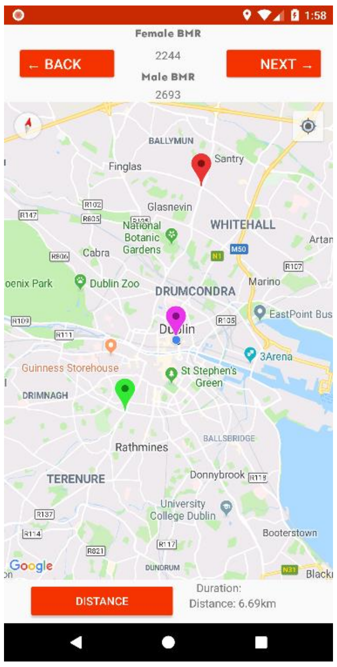


Figure 27 - Distance Between Two points

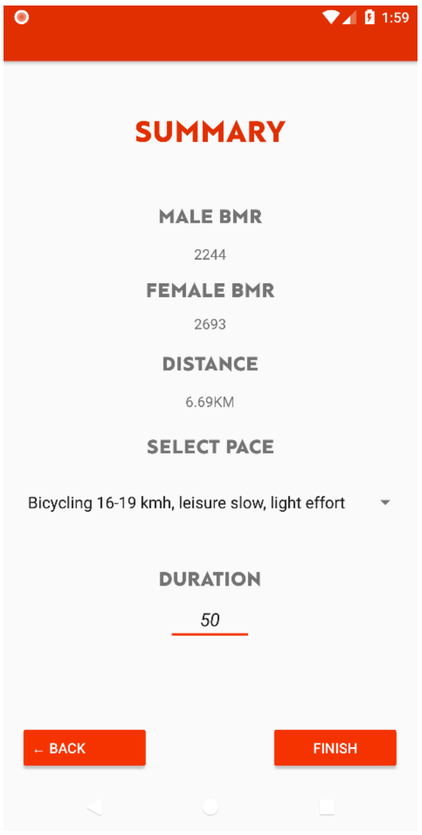
```
if (MarkerPoints.size() > 1) {
    MarkerPoints.clear();
    mMap.clear();
}
MarkerPoints.add(point);
MarkerOptions options = new MarkerOptions();
options.position(point);
if (MarkerPoints.size() == 1) {
options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HU
E_GREEN));
} else if (MarkerPoints.size() == 2) {
options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HU
E_RED));
}
```

Having those values we could use Google’s API method **distanceTo()** to get distance between two location in meters and convert it into km by multiplying by 1000

```
Location locationA = new Location("point A");
locationA.setLatitude(origin.latitude);
locationA.setLongitude(origin.longitude);

Location locationB = new Location("point B");
locationB.setLatitude(dest.latitude);
locationB.setLongitude(dest.longitude);

distanceP = locationA.distanceTo(locationB)/1000;//To convert Meter in
Kilometer
//Formal value to two decimal numbers
formattedDistance = String.format("%.2f", distanceP);

distanceText.setText("Distance: " + formattedDistance + "km");
```

6.4.10 Summary Activity

Summary activity take data from both Firebase Database and Home Activity. In this activity user can select speed they would like to bicycle which is calculated into METS e.g. 16-19 km/h is equivalent to 3 METS. Users also enter the duration which is speed that they think they can cycle the route in minutes.


SUMMARY

MALE BMR

2244

FEMALE BMR

2693

DISTANCE

6.69KM

SELECT PACE

Bicycling 16-19 kmh, leisure slow, light effort

DURATION

50

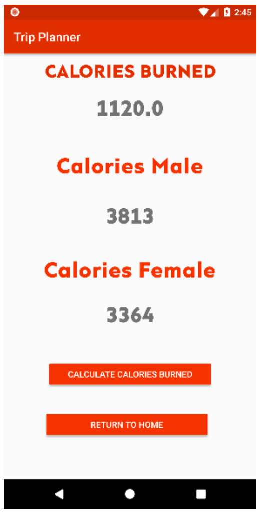
Then this activity then sends the collected data to next activity to create formula.

```
String stringDuration = dur.getText().toString();
duration = Integer.parseInt(stringDuration);

Intent i = new Intent(getApplicationContext(), FinishActivity.class);
i.putExtra("mB", maleBMR);
i.putExtra("fB", femaleBMR);
i.putExtra("disTT", dist);
i.putExtra("metts", mets);
i.putExtra("weig", weight3);
i.putExtra("durrr", duration);
startActivity(i);
```

6.4.11 Calories Burned

In this activity we use formula mentioned in section 4.2 based on the given data user can calculate how much calories he has burned and how many more he can take during the day.

All the main code is located below.



CALORIES BURNED

1120.0

Calories Male

3813

Calories Female

3364

Figure 28 - Final Activity

```
Intent intent = getIntent();
maleBMR = intent.getDoubleExtra("mB", 0);
femaleBMR = intent.getDoubleExtra("fB", 0);
dist = intent.getFloatExtra("distTT", 0);
mets = intent.getDoubleExtra("metts", 0);
weig = intent.getIntExtra("weig", 0);
duration = intent.getIntExtra("durrr", 0);

calc.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        BMRFINAL = (mets * (weig) * 3.5 / 200) * (duration);
        weightkg.setText(Double.toString(BMRFINAL));
        //(MET * (weight) * 3.5 / 200) * (time of activity)

        //CALORIES MALE
        finalmale = maleBMR + BMRFINAL;
        String maleFormatted = String.format("%.0f", finalmale);
        bmrMale.setText(maleFormatted);

        //CALORIES FEMALE
        finalfemale = femaleBMR + BMRFINAL;
        String femaleFormatted = String.format("%.0f", finalfemale);
        bmrFemale.setText(femaleFormatted);

    }
});
```

Chapter 7: Testing and evaluation

7.1 Testing

The most important phase of any software project is testing. This is where we test if the software we designed and programmed does exactly what is asked for. Many different testing's come into play when testing software application, one of the testing is automated testing, its used to test any software app. This testing makes use of special software tools to control the execution of tests and then compares the actual test results with predicted or expected results. However due to the lack of time and contains with the project this would not be possible in this occasion.

Many ways went into testing this application. For the start, each feature of the app was tested directly by developer/author, and each of the elements were tested individually to see if it works as desired. Before major change in the project author ensured that the application was fully functional and tested, these tests were mainly before moving further into developing the application.

Using emulator allowed for testing on completed on many different devices and versions of android, different screen sized were tested to ensure compatibility. Testing on physical phone to ensure that all the features are working as they should.

Author also tested the device using number of friends and family members to ensure that the user experience was working as wanted. Getting feedback from testing users allowed to eliminate major bugs and design flaws. Some noticed that the buttons were not sized properly, and some noticed that the pages were inconsistent with their looks. Few of people were crashing during the map experience.

7.2 Testing process

Author asked test users to install the application on their phones, given basic information on how the application work, how it should behave, and what are features that are available and what should they do.

The expected time to finish testing was about 6-7 min without telling the test user how each menu work or where different screens were. Many test users reported 4-5 min to finish the tests.

The application was designed for easy feel and use, so if user got lost in the application the app would need to change the design or to better readdress each function.

Many feedbacks were collected during the testing and author asked on how does the application feel, and work form a scale 1 to 10. Most of the users responded with positive feedback with the score of 8-9 on average.

Chapter 8: Conclusions and Further Work

8.1 Conclusion

The main goal for this project to gather information about the project and develop fully functional application that is designed to plan trips for users. This app would allow user to calculate their daily calorie intake based on given details.

There is a huge market for android apps and many of them are cycling fitness apps there are features that people like and dislike in every app. Not all the apps provide the functions for free, some of them require subscriptions.

Most popular and similar apps line “Strava” and “Map My Ride” does not provide all the functions and require purchases and subscriptions.

Android will be use as the base Operating system for this project, it is the largest and most popular operating systems and it has the most produced applications in the store.

From the start and finish author had good idea on what the users wanted form the application, and what was the main target the app. To ensure that the project was a success learning, planning and development went into the project to ensure that the finished product was fit for end user.

8.2 Future Work

If more time were available, I would like to implement serval functions for this application. First function to add would be time measurement from the map based on Google API. This would allow for faster and better experience in Trip Planning.

Second feature that would be great for this app is to make the app running in smart watch. This would make the app more accessible and more innovative.

8.3 Personal Reflections

This process of deployment of the application gave me good insight on the overall software development and allowed me to discover stages that are required in order to complete each process. From never developing in android to creating a functional app was a huge step and learning curve for me.

Chapter 9: Appendix A: Project Planning

9.1 Project Planning Chart

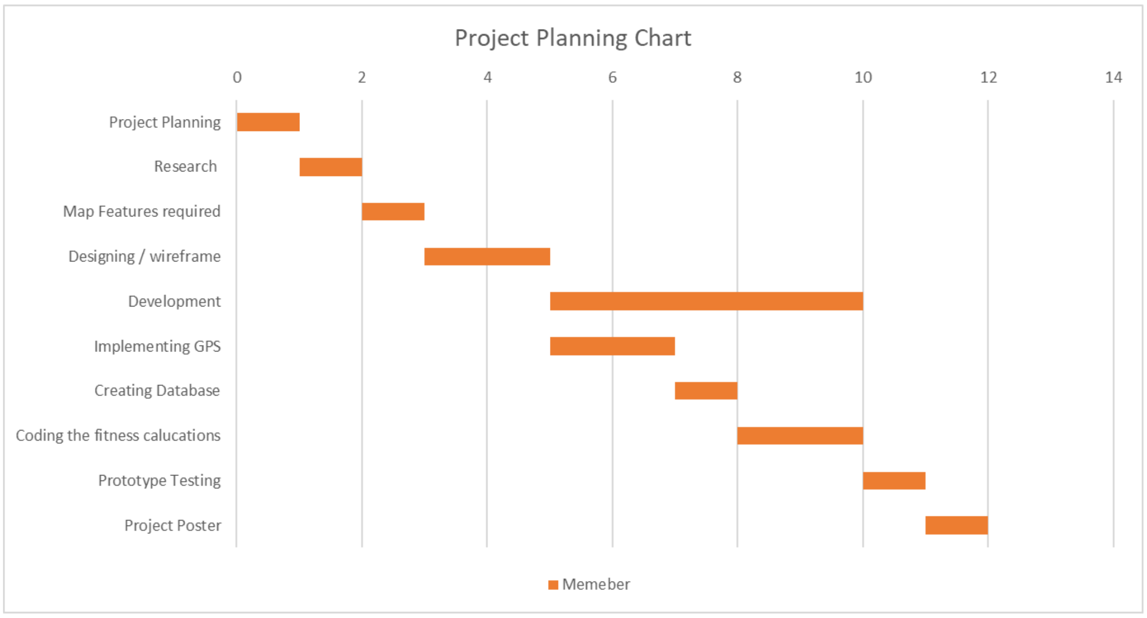


Figure 29 - Project Plan for Semester

9.2 Project Diary

01.11.2018 – Created design for the project

25.11.2018 – added Firebase Authenticator Sign Up

30.11.2018 – added Firebase Authenticator Login

3.12.2018 – added Firebase Authenticator Forgot Password

24.01.2019 – added Firebase database to the project

3.02.2019 – added Google Map to the project

01.03.2019 – added Maps Distance Between Two points

20.04.2019 – Completed all activities.

List of References

[1] Fattoh Al-Qershi, Muhammad Al-Qurishi, Sk Md Mizanur Rahman, and Atif Al-Amri. Android vs. ios: The security battle. In Computer Applications and Information Systems (WCCAIS), 2014 World Congress on, pages 1–8. IEEE, 2014.

[2] Herbert D Benington. Production of large computer programs. Annals of the History of Computing, 5(4):350–361, 1983.

[3] Barry Boehm and Wilfred J Hansen. Special report cmu/sei-2000-sr-008. 2000.

[4] Andre Charland and Brian Leroux. Mobile application development: web vs. native. Queue, 9(4):20, 2011.

[5] Ron Kwok Chi-Wai, Tania Mak So-Ning, Ken Lee Wing-Kuen, Stanley Sai-Chuen Hui, Peter Wu Ka-Shun, and Clara Choi-Ki Wong. Can mobile virtual fitness apps replace human fitness trainer? In Information Science and Service Science (NISS), 2011 5th International Conference on New Trends in, volume 1, pages 56–63. IEEE, 2011.

[6] Ken Collier. Agile analytics: A value-driven approach to business intelligence and data warehousing. Addison-Wesley, 2012.

[7] Artyom Dogtiev. App download and usage statistics 2017, 2018.

[8] J Arthur Harris and Francis G Benedict. A biometric study of human basal metabolism. Proceedings of the National Academy of Sciences, 4(12):370–373, 1918.

[9] 2009 HSS. Estimate how many calories you are burning with exercise. URL https://www.hss.edu/conditions_ 332009 HSS. Estimate how many calories you are burning with exercise.

[10] Mona Erfani Joorabchi, Ali Mesbah, and Philippe Kruchten. Real challenges in mobile app development. In Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on, pages 15–24. IEEE, 2013.

[11] Soo Ling Lim, Peter Bentley, Natalie Kanakam, Fuyuki Ishikawa, and Shinichi Honiden. Investigating country differences in mobile app user behavior and challenges for software engineering. IEEE Transactions on Software Engineering, (1):1–1, 2015.

[12] Ibtisam Mohamed and Dhiren Patel. Android vs ios security: A comparative study. In Information Technology-New Generations (ITNG), 2015 12th International Conference on, pages 725–730. IEEE, 2015.

[13] Ben Popper. Google announces over 2 billion monthly active devices on android. Luettavissa

[14] https://www. theverge. com/2017/5/17/15654454/android-reaches-2- billionmonthly-active-users. Luettu, 27:2017, 2017.

[15] Mark Rowan and Josh Dehlinger. Research trends and open issues in mobile application software engineering.

[16] Proceedings of the International Conference on Software Engineering Research and Practice (SERP), page 1. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2013.

[16] Anthony I Wasserman. Software engineering issues for mobile application development. In Proceedings of the FSE/SDP workshop on Future of software engineering research, pages 397–400. ACM, 2010.

Chapter 10: Appendix B CODE

10.1 Splash.java

```java
package com.trip.planner;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.Window;
import android.view.WindowManager;

public class Splash extends Activity {
            /** Duration of wait **/
            private final int SPLASH_DISPLAY_LENGTH = 1000;
            /** Called when the activity is first created. */
            @Override
            public void onCreate(Bundle icicle) {
                super.onCreate(icicle);
                //remove toolbar
                requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);
        setContentView(R.layout.activity_splash);
        /* New Handler to start the Menu-Activity
         * and close this Splash-Screen after some seconds.*/
        new Handler().postDelayed(new Runnable(){
            @Override
            public void run() {
                /* Create an Intent that will start the Menu-Activity. */
                Intent mainIntent = new Intent(Splash.this,
MainActivity.class);
                Splash.this.startActivity(mainIntent);
                Splash.this.finish();
            }
        }, SPLASH_DISPLAY_LENGTH);

    }
}
```

10.2 MainActivity.java

```
package com.trip.planner;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.text.TextUtils;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ViewFlipper;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class MainActivity extends Activity {

    ViewFlipper v_flipper;
    private Button buttonRegister;
    private EditText editEmail;
    private EditText editPassword;
    private TextView textLogin;
    private ProgressDialog progressDialog;
    private FirebaseAuth firebaseAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        requestWindowFeature(Window.FEATURE_NO_TITLE);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);
        setContentView(R.layout.activity_main);

        progressDialog = new ProgressDialog(this);
        firebaseAuth = FirebaseAuth.getInstance();

        int images[] = {R.drawable.img1, R.drawable.img2, R.drawable.img3,
R.drawable.img4, R.drawable.img5};

        v_flipper = findViewById(R.id.v_flipper);
        buttonRegister = findViewById(R.id.btn_reset_pass);
        editEmail = findViewById(R.id.email);
        editPassword = findViewById(R.id.password);
        textLogin = findViewById(R.id.reset);


        buttonRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(v == buttonRegister){
                    registerUser();
                }
            }
        });
        textLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                    //will open login activity here
                startActivity(new Intent(MainActivity.this,
LoginActivity.class));
            }
        });

        //fot loop
        for(int i = 0; i < images.length; i++){
            flipperImages(images[i]);
        }
    }

    public void registerUser(){
        String email = editEmail.getText().toString().trim();
        String password = editPassword.getText().toString().trim();

        if(TextUtils.isEmpty(email)){
            //email is empty
            Toast.makeText(this, "Please enter email",
Toast.LENGTH_SHORT).show();
            return;
        }

        if(TextUtils.isEmpty(password)){
            //password is empty
            Toast.makeText(this, "Please enter password",
Toast.LENGTH_SHORT).show();
            return;
        }
        //if validation ok
        //show progress bar
        progressDialog.setMessage("Registering...");
        progressDialog.show();

        firebaseAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if(task.isSuccessful()){
                            //user is successfully registered and logged in
                            //start activity profile activity
                            Toast.makeText(MainActivity.this, "Complete User
Profile", Toast.LENGTH_LONG).show();
                            progressDialog.cancel();
                            startActivity(new Intent(MainActivity.this,
UserActivity.class));

                        }else if(firebaseAuth.getCurrentUser() != null) {
                            progressDialog.cancel();
                            Toast.makeText(MainActivity.this, "User already
exits... Try to Log In", Toast.LENGTH_SHORT).show();

                        }else{
                            progressDialog.cancel();
                            Toast.makeText(MainActivity.this, "Could not
register... please try again", Toast.LENGTH_SHORT).show();
                        }
                    }
                });

    }

    public void flipperImages(int image){
        ImageView imageView = new ImageView(this);
        imageView.setBackgroundResource(image);

        v_flipper.addView(imageView);
        v_flipper.setFlipInterval(5000); //6 seconds interval
        v_flipper.setAutoStart(true);
        //animation
        v_flipper.setInAnimation(this, android.R.anim.slide_in_left);
        v_flipper.setOutAnimation(this, android.R.anim.slide_out_right);

    }
}
```

10.3 LoginActivity.java

```
package com.trip.planner;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;



public class LoginActivity extends AppCompatActivity {

    private EditText inputEmail, inputPassword;
    private FirebaseAuth auth;
    private Button btnLogin;
    private TextView textReset, signup;
    private ProgressDialog progressDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //requestWindowFeature(Window.FEATURE_NO_TITLE);

//getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);
        //Get Firebase auth instance
        auth = FirebaseAuth.getInstance();
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        progressDialog = new ProgressDialog(this);
        // set the view now
        setContentView(R.layout.activity_login);

        inputEmail = (EditText) findViewById(R.id.email);
        inputPassword = (EditText) findViewById(R.id.password);
        btnLogin = (Button) findViewById(R.id.btn_reset_pass);
        textReset = (TextView) findViewById(R.id.reset);
        signup = (TextView) findViewById(R.id.signup);


        //Get Firebase auth instance
        auth = FirebaseAuth.getInstance();

        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //will open login activity here
                startActivity(new Intent(LoginActivity.this,
HomeActivity.class));
            }
        });

        textReset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(LoginActivity.this,
ResetPasswordActivity.class));
            }
        });


        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = inputEmail.getText().toString();
                final String password = inputPassword.getText().toString();

                if (TextUtils.isEmpty(email)) {
                    Toast.makeText(getApplicationContext(), "Enter email
address!", Toast.LENGTH_SHORT).show();
                    return;
                }

                if (TextUtils.isEmpty(password)) {
                    Toast.makeText(getApplicationContext(), "Enter password!",
Toast.LENGTH_SHORT).show();
                    return;
                }

                progressDialog.setMessage("Logging In...");
                progressDialog.show();

                //authenticate user
                auth.signInWithEmailAndPassword(email, password)
                        .addOnCompleteListener(LoginActivity.this, new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull Task<AuthResult>
task) {
                                /*If sign in fails, display a message to the
user. If sign in succeeds
                                the auth state listener will be notified and
logic to handle the
                                signed in user can be handled in the
listener.*/

                                if (!task.isSuccessful()) {
                                    // there was an error
                                    if (password.length() < 6) {
                                        progressDialog.cancel();
                                        inputPassword.setError("Incorrect
Password!");

                                    } else {
                                        progressDialog.cancel();
                                        Toast.makeText(LoginActivity.this,
"Incorrect Email or Password", Toast.LENGTH_LONG).show();
                                    }
                                } else {
                                    progressDialog.cancel();
                                    Intent intent = new
Intent(LoginActivity.this, HomeActivity.class);
                                    startActivity(intent);

                                    finish();
                                }
                            }
                        });
                }
            });
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if(id == android.R.id.home){
            this.finish();
        }
        return super.onOptionsItemSelected(item);
    }
}
```

10.4 ResetPasswordActivity.java

```
package com.trip.planner;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;



public class ResetPasswordActivity extends AppCompatActivity {

    private EditText inputEmail;
    private Button btnReset;
    private FirebaseAuth auth;
    private ProgressBar progressBar;
    private TextView signup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //requestWindowFeature(Window.FEATURE_NO_TITLE);

        setContentView(R.layout.activity_reset_password);

//getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        inputEmail = (EditText) findViewById(R.id.email);
        btnReset = (Button) findViewById(R.id.btn_reset_pass);
        progressBar = (ProgressBar) findViewById(R.id.progressBar);
        signup = (TextView) findViewById(R.id.signup);
        auth = FirebaseAuth.getInstance();
        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //open login activity here
                startActivity(new Intent(ResetPasswordActivity.this,
MainActivity.class));
            }
        });

        btnReset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = inputEmail.getText().toString().trim();
                if (TextUtils.isEmpty(email)) {
                    Toast.makeText(getApplication(), "Enter your registered
email id", Toast.LENGTH_SHORT).show();
                    return;
                }
                progressBar.setVisibility(View.VISIBLE);
                auth.sendPasswordResetEmail(email)
                        .addOnCompleteListener(new OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void> task) {
                                if (task.isSuccessful()) {
                                    Toast.makeText(ResetPasswordActivity.this,
"We have sent you instructions to reset your password!",
Toast.LENGTH_SHORT).show();
                                } else {
                                    Toast.makeText(ResetPasswordActivity.this,
"Failed to send reset email!", Toast.LENGTH_SHORT).show();
                                }

                                progressBar.setVisibility(View.GONE);
                            }
                        });
            }
        });

    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if(id == android.R.id.home){
            this.finish();
        }
        return super.onOptionsItemSelected(item);
    }
}
```

10.5 Profile.java

```
package com.trip.planner;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;


public class Profile extends AppCompatActivity {

    private Button btnChangeEmail, btnChangePassword, btnSendResetEmail,
btnRemoveUser,
            changeEmail, changePassword, sendEmail, remove, signOut;

    private EditText oldEmail, newEmail, password, newPassword;
    private ProgressBar progressBar;
    private FirebaseAuth.AuthStateListener authListener;
    private FirebaseAuth auth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        setContentView(R.layout.activity_profile);
        //setSupportActionBar(toolbar);


        //get firebase auth instance
        auth = FirebaseAuth.getInstance();

        //get current user
        final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

        authListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth)
{
                FirebaseUser user = firebaseAuth.getCurrentUser();
                if (user == null) {
                    // user auth state is changed - user is null
                    // launch login activity
                    startActivity(new Intent(Profile.this,
LoginActivity.class));
                    finish();
                }
            }
        };

        btnChangeEmail = (Button) findViewById(R.id.change_email_button);
        btnChangePassword = (Button) findViewById(R.id.change_password_button);
        btnSendResetEmail = (Button)
findViewById(R.id.sending_pass_reset_button);
        btnRemoveUser = (Button) findViewById(R.id.remove_user_button);
        changeEmail = (Button) findViewById(R.id.changeEmail);
        changePassword = (Button) findViewById(R.id.changePass);
        sendEmail = (Button) findViewById(R.id.send);
        remove = (Button) findViewById(R.id.remove);
        signOut = (Button) findViewById(R.id.sign_out);


        oldEmail = (EditText) findViewById(R.id.old_email);
        newEmail = (EditText) findViewById(R.id.new_email);
        password = (EditText) findViewById(R.id.password);
        newPassword = (EditText) findViewById(R.id.newPassword);

        oldEmail.setVisibility(View.GONE);
        newEmail.setVisibility(View.GONE);
        password.setVisibility(View.GONE);
        newPassword.setVisibility(View.GONE);
        changeEmail.setVisibility(View.GONE);
        changePassword.setVisibility(View.GONE);
        sendEmail.setVisibility(View.GONE);
        remove.setVisibility(View.GONE);

        progressBar = (ProgressBar) findViewById(R.id.progressBar);

        if (progressBar != null) {
            progressBar.setVisibility(View.GONE);
        }

        btnChangeEmail.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                oldEmail.setVisibility(View.GONE);
                newEmail.setVisibility(View.VISIBLE);
                password.setVisibility(View.GONE);
                newPassword.setVisibility(View.GONE);
                changeEmail.setVisibility(View.VISIBLE);
                changePassword.setVisibility(View.GONE);
                sendEmail.setVisibility(View.GONE);
                remove.setVisibility(View.GONE);
            }
        });

        changeEmail.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                progressBar.setVisibility(View.VISIBLE);
                if (user != null &&
!newEmail.getText().toString().trim().equals("")) {
                    user.updateEmail(newEmail.getText().toString().trim())
                            .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void>
task) {
                                    if (task.isSuccessful()) {
                                        Toast.makeText(Profile.this, "Email
address is updated. Please sign in with new email id!",
Toast.LENGTH_LONG).show();
                                        signOut();
                                        progressBar.setVisibility(View.GONE);
                                    } else {
                                        Toast.makeText(Profile.this, "Failed to
update email!", Toast.LENGTH_LONG).show();
                                        progressBar.setVisibility(View.GONE);
                                    }
                                }
                            });
                } else if (newEmail.getText().toString().trim().equals("")) {
                    newEmail.setError("Enter email");
                    progressBar.setVisibility(View.GONE);
                }
            }
        });

        btnChangePassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                oldEmail.setVisibility(View.GONE);
                newEmail.setVisibility(View.GONE);
                password.setVisibility(View.GONE);
                newPassword.setVisibility(View.VISIBLE);
                changeEmail.setVisibility(View.GONE);
                changePassword.setVisibility(View.VISIBLE);
                sendEmail.setVisibility(View.GONE);
                remove.setVisibility(View.GONE);
            }
        });

        changePassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                progressBar.setVisibility(View.VISIBLE);
                if (user != null &&
!newPassword.getText().toString().trim().equals("")) {
                    if (newPassword.getText().toString().trim().length() < 6) {
                        newPassword.setError("Password too short, enter minimum
6 characters");
                        progressBar.setVisibility(View.GONE);
                    } else {

user.updatePassword(newPassword.getText().toString().trim())
                                .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                    @Override
                                    public void onComplete(@NonNull Task<Void>
task) {
                                        if (task.isSuccessful()) {
                                            Toast.makeText(Profile.this,
"Password is updated, sign in with new password!", Toast.LENGTH_SHORT).show();
                                            signOut();

progressBar.setVisibility(View.GONE);
                                        } else {
                                            Toast.makeText(Profile.this,
"Failed to update password!", Toast.LENGTH_SHORT).show();

progressBar.setVisibility(View.GONE);
                                        }
                                    }
                                });
                    }
                } else if (newPassword.getText().toString().trim().equals(""))
{
                    newPassword.setError("Enter password");
                    progressBar.setVisibility(View.GONE);
                }
            }
        });

        btnSendResetEmail.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                oldEmail.setVisibility(View.VISIBLE);
                newEmail.setVisibility(View.GONE);
                password.setVisibility(View.GONE);
                newPassword.setVisibility(View.GONE);
                changeEmail.setVisibility(View.GONE);
                changePassword.setVisibility(View.GONE);
                sendEmail.setVisibility(View.VISIBLE);
                remove.setVisibility(View.GONE);
            }
        });

        sendEmail.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                progressBar.setVisibility(View.VISIBLE);
                if (!oldEmail.getText().toString().trim().equals("")) {

auth.sendPasswordResetEmail(oldEmail.getText().toString().trim())
                            .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void>
task) {
                                    if (task.isSuccessful()) {
                                        Toast.makeText(Profile.this, "Reset
password email is sent!", Toast.LENGTH_SHORT).show();
                                        progressBar.setVisibility(View.GONE);
                                    } else {
                                        Toast.makeText(Profile.this, "Failed to
send reset email!", Toast.LENGTH_SHORT).show();
                                        progressBar.setVisibility(View.GONE);
                                    }
                                }
                            });
                } else {
                    oldEmail.setError("Enter email");
                    progressBar.setVisibility(View.GONE);
                }
            }
        });

        btnRemoveUser.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                progressBar.setVisibility(View.VISIBLE);
                if (user != null) {
                    user.delete()
                            .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void>
task) {
                                    if (task.isSuccessful()) {
                                        Toast.makeText(Profile.this, "Your
profile is deleted:( Create a account now!", Toast.LENGTH_SHORT).show();
                                        startActivity(new Intent(Profile.this,
MainActivity.class));
                                        finish();
                                        progressBar.setVisibility(View.GONE);
                                    } else {
                                        Toast.makeText(Profile.this, "Failed to
delete your account!", Toast.LENGTH_SHORT).show();
                                        progressBar.setVisibility(View.GONE);
                                    }
                                }
                            });
                }
            }
        });

        signOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                signOut();
            }
        });
    }

    //sign out method
    public void signOut() {
        auth.signOut();
    }




    @Override
    protected void onResume() {
        super.onResume();
        progressBar.setVisibility(View.GONE);
    }

    @Override
    public void onStart() {
        super.onStart();
        auth.addAuthStateListener(authListener);
    }
    @Override
    public void onStop() {
        super.onStop();
        if (authListener != null) {
            auth.removeAuthStateListener(authListener);
        }
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if(id == android.R.id.home){
            this.finish();
        }
        return super.onOptionsItemSelected(item);
    }
}
```

10.6 User.class

```
package com.trip.planner;

public class User {
    public String name;
    public int age, weight, height;

    public User(){

    }

    public User(String name, int age, int weight, int height) {
        this.name = name;
        this.age = age;
        this.weight = weight;
        this.height = height;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    public int getHeight() {
        return height;
    }

    public void setHeight(int height) {
        this.height = height;
    }
}
```

10.7 UserActivity.java

```
package com.trip.planner;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class UserActivity extends AppCompatActivity {
    private EditText editName, editAge, editWeight, editHeight;
    private Button add;

    private FirebaseAuth mAuth;
    private ProgressDialog progressDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user);

        editName = (EditText) findViewById(R.id.editName);
        editAge = (EditText) findViewById(R.id.editAge);
        editHeight = (EditText) findViewById(R.id.editHeight);
        editWeight = (EditText) findViewById(R.id.editWeight);
        progressDialog = new ProgressDialog(this);

        add = (Button) findViewById(R.id.add);


        mAuth = FirebaseAuth.getInstance();

        add.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (v.getId()) {
                    case R.id.add:
                        addName();
                        break;
                }
                progressDialog.setMessage("Registering User...");
                progressDialog.show();
            }
        });

    }


    protected void onStart() {
        super.onStart();

        if(mAuth.getCurrentUser() != null){
            //handle already login user
        }
    }
    private void addName(){
        final String name = editName.getText().toString().trim();

        final String age = editAge.getText().toString().trim();
        int agetoint = Integer.parseInt(age);

        final String weight = editWeight.getText().toString().trim();
        int weightoint = Integer.parseInt(weight);

        final String height = editHeight.getText().toString().trim();
        int heighttoint = Integer.parseInt(height);


        User user = new User(
                name,
                agetoint,
                weightoint,
                heighttoint
        );


FirebaseDatabase.getInstance().getReference("Users").child(FirebaseAuth.getInst
ance().getUid())
                .setValue(user).addOnCompleteListener(new
OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if(task.isSuccessful()){
                    Toast.makeText(UserActivity.this, "Register Successful!",
Toast.LENGTH_LONG).show();
                    progressDialog.cancel();
                    startActivity(new Intent(UserActivity.this,
HomeActivity.class));
                }else{
                    progressDialog.cancel();
                    Toast.makeText(UserActivity.this, "Error",
Toast.LENGTH_LONG).show();

                }
            }
        });

    }



    }
```

10.8 MapsActivity.java

```
package com.trip.planner;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.Location;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
```

```java
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolylineOptions;
import com.trip.planner.DirectionsJSONParser;

import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationListener {

    private GoogleMap mMap;
    ArrayList<LatLng> MarkerPoints;
    GoogleApiClient mGoogleApiClient;
    Location mLastLocation;
    Marker mCurrLocationMarker;
    LocationRequest mLocationRequest;
    TextView distanceText, bmrF, bmrM, DistanceDuration;
    Button distButton, nextButton, backButton;
    float distanceP;
    String femaleFormatted, maleFormatted, formattedDistance;
    int weightt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);

        distanceText = (TextView) findViewById(R.id.distanceText);
        bmrF = (TextView) findViewById(R.id.bmrF);
        bmrM = (TextView) findViewById(R.id.bmrM);
        DistanceDuration = (TextView) findViewById(R.id.duration);


        distButton = (Button) findViewById(R.id.distButton);
        nextButton = (Button) findViewById(R.id.nextButton);
        backButton = (Button) findViewById(R.id.backButton);

        Intent intent = getIntent();
        final Double maleBMR = intent.getDoubleExtra("maleBMR", 0);
        final Double femaleBMR = intent.getDoubleExtra("femaleBMR", 0);
        weightt = intent.getIntExtra("weight", 0);

        maleFormatted = String.format("%.0f", maleBMR);
        femaleFormatted = String.format("%.0f", femaleBMR);

        bmrM.setText(maleFormatted);
        bmrF.setText(femaleFormatted);


        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            checkLocationPermission();
        }
        // Initializing
        MarkerPoints = new ArrayList<>();

        // Obtain the SupportMapFragment and get notified when the map is ready
to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MapsActivity.this,
HomeActivity.class));
            }
        });

        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(distanceP == 0){
                    Toast.makeText(MapsActivity.this, "Calculate Distance
before continuing!", Toast.LENGTH_SHORT).show();

                }else if(distanceP > 0){
                    startActivity(new Intent(MapsActivity.this,
SummaryActivity.class));

                    Intent i = new Intent(getApplicationContext(),
SummaryActivity.class);
                    i.putExtra("mBMR", maleBMR);
                    i.putExtra("fBMR", femaleBMR);
                    i.putExtra("distance", distanceP);
                    i.putExtra("weightt", weightt);
                    startActivity(i);
                }
            }
        });
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        //Initialize Google Play Services
        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
                buildGoogleApiClient();
                mMap.setMyLocationEnabled(true);
            }
        }
        else {
            buildGoogleApiClient();
            mMap.setMyLocationEnabled(true);
        }

        // Setting onclick event listener for the map
        mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
            @Override
            public void onMapClick(LatLng point) {

                // Already two locations
                if (MarkerPoints.size() > 1) {
                    MarkerPoints.clear();
                    mMap.clear();
                }

                // Adding new item to the ArrayList
                MarkerPoints.add(point);

                // Creating MarkerOptions
                MarkerOptions options = new MarkerOptions();

                // Setting the position of the marker
                options.position(point);
                if (MarkerPoints.size() == 1) {

options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_
GREEN));
                } else if (MarkerPoints.size() == 2) {

options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_
RED));
                }


                // Add new marker to the Google Map Android API V2
                mMap.addMarker(options);

                // Checks, whether start and end locations are captured
                if (MarkerPoints.size() >= 2) {
                    LatLng origin = MarkerPoints.get(0);
                    LatLng dest = MarkerPoints.get(1);

                    // Getting URL to the Google Directions API
                    String url = getUrl(origin, dest);
                    Log.d("onMapClick", url.toString());
                    FetchUrl FetchUrl = new FetchUrl();

                    // Start downloading json data from Google Directions API
                    FetchUrl.execute(url);

                    //move map camera
                    mMap.moveCamera(CameraUpdateFactory.newLatLng(origin));
                    mMap.animateCamera(CameraUpdateFactory.zoomTo(10));
                }

            }
        });

    }

     String getUrl(final LatLng origin, final LatLng dest) {

        // Origin of route
        String str_origin = "origin=" + origin.latitude + "," +
origin.longitude;

        // Destination of route
        String str_dest = "destination=" + dest.latitude + "," +
dest.longitude;

        //button to calculate route distance between two points
        distButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //https://readyandroid.wordpress.com/calculate-distance-
between-two-latlng-points-using-google-api-or-math-function-android/
                Location locationA = new Location("point A");

                locationA.setLatitude(origin.latitude);
                locationA.setLongitude(origin.longitude);

                Location locationB = new Location("point B");

                locationB.setLatitude(dest.latitude);
                locationB.setLongitude(dest.longitude);

                distanceP = locationA.distanceTo(locationB)/1000;//To convert
Meter in Kilometer
                //Formal value to two decimal numbers
                formattedDistance = String.format("%.2f", distanceP);

                distanceText.setText("Distance: " + formattedDistance + "km");

            }
        });


        // Sensor enabled
        String sensor = "sensor=true";

        // Building the parameters to the web service
        String parameters = str_origin + "&" + str_dest + "&" + sensor;

        // Output format
        String output = "json";

         String url = "https://maps.googleapis.com/maps/api/directions/" +
output + "?" + parameters +"&key=" +"AIzaSyDBIGe4chVMD15lyKOPPWOdPn5xS-l-7pA";

        return url;
    }

    private String downloadUrl(String strUrl) throws IOException {
        String data = "";
        InputStream iStream = null;
        HttpURLConnection urlConnection = null;
        try {
            URL url = new URL(strUrl);

            // Creating an http connection to communicate with url
            urlConnection = (HttpURLConnection) url.openConnection();

            // Connecting to url
            urlConnection.connect();

            // Reading data from url
            iStream = urlConnection.getInputStream();

            BufferedReader br = new BufferedReader(new
InputStreamReader(iStream));

            StringBuffer sb = new StringBuffer();

            String line = "";
            while ((line = br.readLine()) != null) {
                sb.append(line);
            }

            data = sb.toString();
            Log.d("downloadUrl", data.toString());
            br.close();

        } catch (Exception e) {
            Log.d("Exception", e.toString());
        } finally {
            iStream.close();
            urlConnection.disconnect();
        }
        return data;
    }

    // Fetches data from url passed
    private class FetchUrl extends AsyncTask<String, Void, String> {

        @Override
        protected String doInBackground(String... url) {

            // For storing data from web service
            String data = "";

            try {
                // Fetching the data from web service
                data = downloadUrl(url[0]);
                Log.d("Background Task data", data.toString());
            } catch (Exception e) {
                Log.d("Background Task", e.toString());
            }
            return data;
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);

            ParserTask parserTask = new ParserTask();

            // Invokes the thread for parsing the JSON data
            parserTask.execute(result);

        }
    }

    private class ParserTask extends AsyncTask<String, Integer,
List<List<HashMap<String, String>>>> {

        // Parsing the data in non-ui thread
        @Override
        protected List<List<HashMap<String, String>>> doInBackground(String...
jsonData) {

            JSONObject jObject;
            List<List<HashMap<String, String>>> routes = null;

            try {
                jObject = new JSONObject(jsonData[0]);
                Log.d("ParserTask",jsonData[0].toString());
                DirectionsJSONParser parser = new DirectionsJSONParser();
                Log.d("ParserTask", parser.toString());

                // Starts parsing data
                routes = parser.parse(jObject);
                Log.d("ParserTask","Executing routes");
                Log.d("ParserTask",routes.toString());

            } catch (Exception e) {
                Log.d("ParserTask",e.toString());
                e.printStackTrace();
            }
            return routes;
        }

        @Override
        protected void onPostExecute(List<List<HashMap<String, String>>>
result) {
            ArrayList<LatLng> points;
            PolylineOptions lineOptions = new PolylineOptions();
            String duration = "";

            // Traversing through all the routes
            for (int i = 0; i < result.size(); i++) {
                points = new ArrayList<>();
                lineOptions = new PolylineOptions();

                // Fetching i-th route
                List<HashMap<String, String>> path = result.get(i);

                // Fetching all the points in i-th route
                for (int j = 0; j < path.size(); j++) {
                    HashMap<String, String> point = path.get(j);

                    if(j==1){ // Get duration from the list
                        duration = (String)point.get("duration");
                        continue;
                    }

                    double lat = Double.parseDouble(point.get("lat"));
                    double lng = Double.parseDouble(point.get("lng"));
                    LatLng position = new LatLng(lat, lng);

                    points.add(position);
                }

                // Adding all the points in the route to LineOptions
                lineOptions.addAll(points);
                lineOptions.width(5);
                lineOptions.color(Color.RED);

                Log.d("onPostExecute","onPostExecute lineoptions decoded");

            }

            if(lineOptions != null) {
                mMap.addPolyline(lineOptions);
            }
            else {
                mMap.addPolyline(lineOptions);
                Log.d("onPostExecute","without Polylines drawn");
            }
        }
    }

    protected synchronized void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();
    }
    @Override
    public void onConnected(Bundle bundle) {

        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);

mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {

LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
    }
    @Override
    public void onConnectionSuspended(int i) {

    }
    @Override
    public void onLocationChanged(Location location) {

        mLastLocation = location;
        if (mCurrLocationMarker != null) {
            mCurrLocationMarker.remove();
        }
        LatLng latLng = new LatLng(location.getLatitude(),
location.getLongitude());
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
        markerOptions.title("Current Position");

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactor
y.HUE_MAGENTA));
        mCurrLocationMarker = mMap.addMarker(markerOptions);

        //move map camera
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(11));

        //stop location updates
        if (mGoogleApiClient != null) {

LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
this);
        }

    }

    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {

    }

    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;
    public boolean checkLocationPermission(){
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)) {

                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION);


            } else {
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION);
            }
            return false;
        } else {
            return true;
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           String permissions[], int[]
grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_LOCATION: {
```

```java
                // If request is cancelled, the result arrays are empty.
                if (grantResults.length > 0
                        && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    if (ContextCompat.checkSelfPermission(this,
                            Manifest.permission.ACCESS_FINE_LOCATION)
                            == PackageManager.PERMISSION_GRANTED) {

                        if (mGoogleApiClient == null) {
                            buildGoogleApiClient();
                        }
                        mMap.setMyLocationEnabled(true);
                    }

                } else {
                    Toast.makeText(this, "permission denied",
Toast.LENGTH_LONG).show();
                }
                return;
            }
        }
    }
}
```

10.9 SummaryActivity.java

```java
package com.trip.planner;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class SummaryActivity extends AppCompatActivity {

    TextView distance, fBMR, mBMR;
    Button backButton, finishButton;
    Spinner sumS;
    double mets, femaleBMR, maleBMR;
    EditText dur;
    Float dist;
    int duration, weight3;
    String femaleFormatted, maleFormatted, distanceFormatted;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);
        setContentView(R.layout.activity_summary);
        sumS = (Spinner) findViewById(R.id.metSpinner);
        fBMR = (TextView)findViewById(R.id.fBMR);
        mBMR = (TextView)findViewById(R.id.mBMR);
        distance = (TextView)findViewById(R.id.distance);

        dur = (EditText) findViewById(R.id.dur);

        backButton = (Button)findViewById(R.id.backButton);
        finishButton = (Button)findViewById(R.id.finishButton);
        Intent intent = getIntent();
        maleBMR = intent.getDoubleExtra("mBMR", 0);
        femaleBMR = intent.getDoubleExtra("fBMR", 0);
        dist = intent.getFloatExtra("distance", 0);
        weight3 = intent.getIntExtra("weightt", 0);

        maleFormatted = String.format("%.0f", maleBMR);
        femaleFormatted = String.format("%.0f", femaleBMR);
        distanceFormatted = String.format("%.2f", dist);

        fBMR.setText(maleFormatted);
        mBMR.setText(femaleFormatted);
        distance.setText(distanceFormatted + "KM");
        //source: https://community.plu.edu/~chasega/met.html
        String names[] = {"Please select PACE",
                "Bicycling, stationary, 50 watts, very light effort",
                "Bicycling, < 16 kmh, leisure, to work or for pleasure",
                "Bicycling 16-19 kmh, leisure slow, light effort",
                "Bicycling 19.3-22.3 kmh leisure, moderate effort",
                "Bicycling 22.5-25.5 kmh, racing or recreational, fast,
vigorous effort",
                "Bicycling 25.7-30.5 kmh, racing/not drafting or >30.5 kmh
drafting, very fast, general racing",
                "Bicycling >32 kmh, racing, not drafting",};

        ArrayAdapter<String> adapter;

        adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, names);

        sumS.setAdapter(adapter);

        sumS.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
{
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {

                switch (position)
                {
                    case 0:
                        mets = 0;
                        break;
                    case 1:
                        mets = 3;
                        break;
                    case 2:
                        mets = 4;
                        break;
                    case 3:
                        mets = 6;
                        break;
                    case 4:
                        mets = 8;
                        break;
                    case 5:
                        mets = 10;
                        break;
                    case 6:
                        mets = 12;
                        break;
                    case 7:
                        mets = 16;
                        break;
                }
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });

        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(SummaryActivity.this,
MapsActivity.class));
            }
        });

        finishButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String stringDuration = dur.getText().toString();
                duration = Integer.parseInt(stringDuration);

                Intent i = new Intent(getApplicationContext(),
FinishActivity.class);
                i.putExtra("mB", maleBMR);
                i.putExtra("fB", femaleBMR);
                i.putExtra("disTT", dist);
                i.putExtra("metts", mets);
                i.putExtra("weig", weight3);
                i.putExtra("durrr", duration);
                startActivity(i);



            }
        });



    }



}
```

10.10 *FinishActivity.java*

```
package com.trip.planner;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class FinishActivity extends AppCompatActivity {
    TextView weightkg,bmrMale, bmrFemale;
    double maleBMR, femaleBMR, mets, BMRFINAL, finalmale, finalfemale;
    Float dist;
    int duration, weig;
    Button calc;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_finish);
        final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        weightkg = (TextView) findViewById(R.id.weightkg);
        bmrMale = (TextView) findViewById(R.id.bmrMaleF);
        bmrFemale = (TextView) findViewById(R.id.bmrFemaleF);
        calc = (Button) findViewById(R.id.calc);



        Intent intent = getIntent();
        maleBMR = intent.getDoubleExtra("mB", 0);
        femaleBMR = intent.getDoubleExtra("fB", 0);
        dist = intent.getFloatExtra("distTT", 0);
        mets = intent.getDoubleExtra("metts", 0);
        weig = intent.getIntExtra("weig", 0);
        duration = intent.getIntExtra("durrr", 0);

        calc.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                BMRFINAL = (mets * (weig) * 3.5 / 200) * (duration);
                weightkg.setText(Double.toString(BMRFINAL));
                //(MET * (weight) * 3.5 / 200) * (time of activity)

                //CALORIES MALE
                finalmale = maleBMR + BMRFINAL;
                String maleFormatted = String.format("%.0f", finalmale);
                bmrMale.setText(maleFormatted);

                //CALORIES FEMALE
                finalfemale = femaleBMR + BMRFINAL;
                String femaleFormatted = String.format("%.0f", finalfemale);
                bmrFemale.setText(femaleFormatted);

            }
        });

    }
}
```