Anthony Awobasivwe 6442385

University of Surrey
Faculty of Engineering and Physical Sciences

A Fitness Online Chatting Application For Students at the University of Surrey

Words = 12,481

Acknowledgements

I would first like to thank my project sponsor Francois Dupressoir for his continued support during the course of the project. I would also like to thank my family for giving me the opportunity to study in the UK through my three years as an international student. Lastly, I would like to thank the University of Surrey and its staff members who gave me the platform to learn and experience new coding languages.

Declaration of Originality

"'I confirm that the submitted work is my own work and that I have clearly identified and fully acknowledged all material that is entitled to be attributed to others (whether published or unpublished) using the referencing system set out in the programme handbook. I agree that the University may submit my work to means of checking this, such as the plagiarism detection service Turnitin® UK. I confirm that I understand that assessed work that has been shown to have been plagiarised will be penalised".

Abstract

Recent advances in mobile development have brought about changes to the lifestyle of individuals in the society which has lead to an increase in the demand for mobile and development of mobile applications. Hence this rapid evolution of technology has made way for a growing variety of applications relating to health and healthcare have offered access to a variety of applications and technologies to healthcare professionals. New features in the mobile industry have revolutionized the health and the social industry due to the advanced software of smartphones.
This dissertation looks into building an application for students at the University of Surrey to communication and fitness purposes through features such as a pedometer.

Table Of Contents

Table of Figures

Statement of Ethics

This report will be developed with a high level of thought going into the legal and ethical considerations regarding the project. The project in itself involves managing sensitive data of the users such as the results which are gotten from the survey, the messages sent between users etc. The project will follow the Data Protection Act 1998. All data collected from users will be stored and processed Fairly & lawfully, data gotten from users will be processed securely and with a sound motivation to protect user interest. Ethics are the norms that distinguish rights from wrongs. Users who partake in the survey and user acceptance test will have full informed consent towards what they are going to be doing. This involves letting users know what they will be doing and for how long , giving users the option to withdraw at any time of their choosing , letting users know the purpose of the research to be conducted, potential benefits to the society, potential harms or discomfort in terms of questions asked and a means for users to get their copy of the results of the survey and user acceptance test . Concerning the social considerations, this looks into the issues and trends of the application on society. The application currently is in a prototype phase however I do strongly feel it could be nurtured and developed to leave a positive impact on the target population which could also grow based on the success of the project.

Anthony Awobasivwe 6442385

Section 1 - Introduction

1.1 Problem Background

From my research and being at the University Of Surrey for a considerable period of time, it came to my attention that the university did not have any clear mobile form of interaction between students. The argument could be made that the universities email platform could be used for communication however emailing is a more formal method of communicating which this application does not exhibit. The mobile application the university had related to social networking was the Surrey Alumni Hub however this form of communication was only available to alumni and graduate students of the university. It could be argued most university students spend the most time on their phones communicating, hence the fitness part of the app was added in order to give users a lure to use its features for their own health and well-being benefits while communicating on the app.

1.2 Project Description

This project aims to provide a more versatile means of communication among university students. This could be useful as it will provide a means of communication between students aside from the already existing forms such as Facebook, WhatsApp etc. It will also prove useful for students who are for example allocated to a team/group in a certain course, users can start communication on this app rather than having to exchange personal information such as phone numbers via WhatsApp to fellow group members on a randomly assigned group.
The project also aims to integrate a platform where users can manage forms of cardio exercise such as running.

1.3 Motivational factor

I was motivated to proceed with this topic as I am really intrigued by social media applications. It also came to my notice that the university didn't really have any relaxed means of communication between students other than outside parties such as Facebook, snap chat etc. When I first came to university it was initially hard to communicate with other students as most where cautious about giving out personal information initially so this app could also help me on a personal level knowing fresher students will have an easier way to communicate with fellow students.

Concerning the fitness aspect of the application, I felt motivated to proceed with it as I am really intrigued by fitness and body care in general. When I first started studying at the University Of Surrey I didn't really know much about the area or in terms of anything as I am an International student so exercise kept me occupied for the most part. This combined with my ever-growing interest in mobile development made this topic a suitable choice for my coding skills as well as my interest

1.4 Aims and Objectives

This project aims to create a new system which could potentially be used by the university as a means of communication between students while also creating a platform for university students to exercise on. Students will be able to sign into the application by providing their university username and a password of their choosing, students can also add fellow students based on their university email.

Objectives :

1. To create an application suitable for users in a real-world scenario
2. To conduct research into firebase in order to apply it to my application
3. To conduct research into how designing a system can help user approval
4. Integrating fitness features into the chat application presentable to users
5. To learn new features in android studio such as firebase authentication, learning more complex forms of services, receivers and fragments and integrating these features into the coding aspect in android studio
6. Setting out a project management plan in order to have all application deliverables ready on schedule

1.5 Success Criteria

The success criteria for the project will be based on :

1. Providing a working application suitable for use by the target population
2. Improving my skills in Android development to implement features into my application thus strengthening my coding while giving me more insight into android development.
3. The user acceptance testing feedback gotten from users
4. The testing of all core functional requirements being met
5. Finishing the project on time and within the scope and quality level of coding practice I've been taught at the University of Surrey

1.6 Project Stages

This section identifies the key phases of the project needed in order for a successful project to be delivered.

- Section 1 - Project Introduction - This section of the report will aim to give an introduction to the objectives and aims, the success criteria of the project I will be developing as well as the motivational factor of my chosen topic.
- Section 2- Literature review - This section of the report will aim to introduce terms related to the project such as firebase. Existing systems related to my application will also be discussed in this section of the report.
- Section 3 - System analysis - This section of the report will aim to provide an analysis of the system to be built. The relevant methods of gathering requirements, the target population for the project as well as the functional and nonfunctional requirements of the project will be addressed in this section

- Section 4- System design-This section looks into creating a system design that will be best suited to our target population.

- Section 5 - Project Management- This section will aim to create a project plan for the project in order to maximise the time for the project. A risk assessment will also be conducted in this section of the report.
- Section 6 - implementation - This section will focus on the actual implementation of the project. The java classes used for the project as well as any external libraries used will be highlighted.
- Section 7 - System Testing - This section of the project will aim to test if the system built from the implementation phase is working. A user acceptance test will also be conducted in order to gain users thoughts on the application.
- Section 8 - System Evaluation - This section of the project will aim to give an overall conclusion for the projects, lessons learnt and future improvements in the project. The objectives and success criteria will also be assessed in this section.

Section 2 - Literature Review

2.1 Smartphone Ecosystem

The current ecosystem of application stores provides an opportunity for independent developers to publish their applications online. It is estimated about a third of the world's population will have smartphones in 2020. The increase in people possessing smartphone is a welcome sign for this project and all mobile development projects as the steady increase in mobile devices available to consumers will also lead to an increase in the consumption rate for new development applications like my final year project application.

2.2 Firebase

Firebase is a mobile and web application development platform created in 2011 as a startup called Envolve. It provided developers with an API that enabled functionalities such as chatting. It was later separated into two parts due to the contradicting ways in which the application was being used. Firebase provides users with lots of functionalities such as Authentication through fingerprints, email/phone number and passwords authentications, 3rd party authentications through google/facebook. It also provides a real-time database and storage for users. Firebase stores data through JSON contrary to other popular servers for Android applications such as Oracle SQL, Microsoft SQL Server, and MySQL which are connected to the server with PHP files.

Firebase was used in my application for authentication, real-time database and the chatting features present in the application.

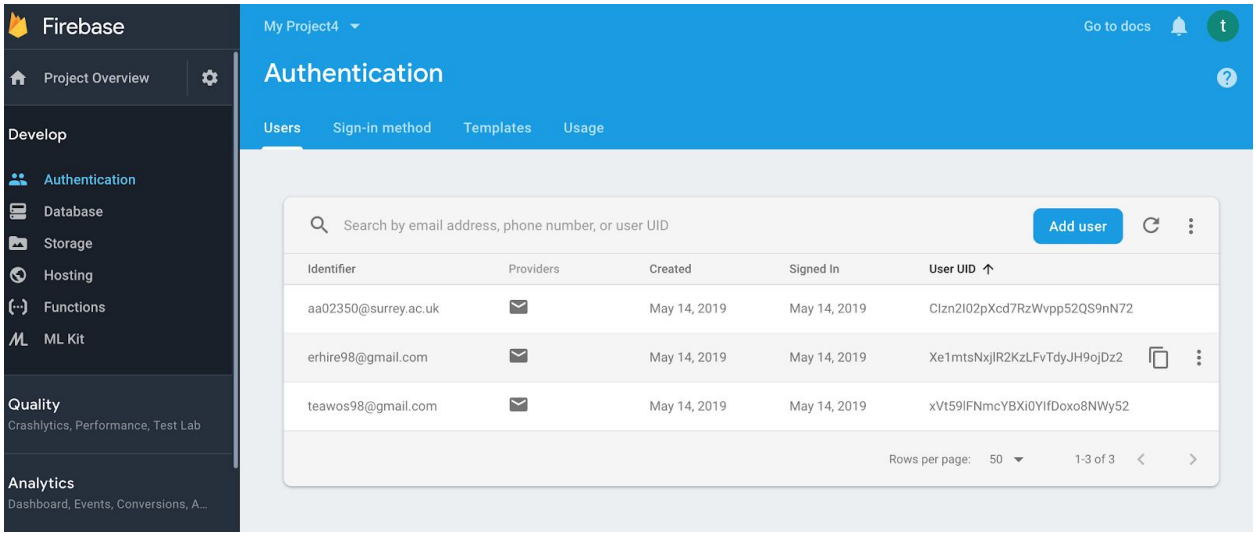


Figure 1

Firebase can also be used to analyze the level of activity in the application by providing details such as daily active users, users in a certain amount of time, retention percentage etc. This could help provide a

good idea of the level of interaction with the application by users assuming it was available online for downloads.

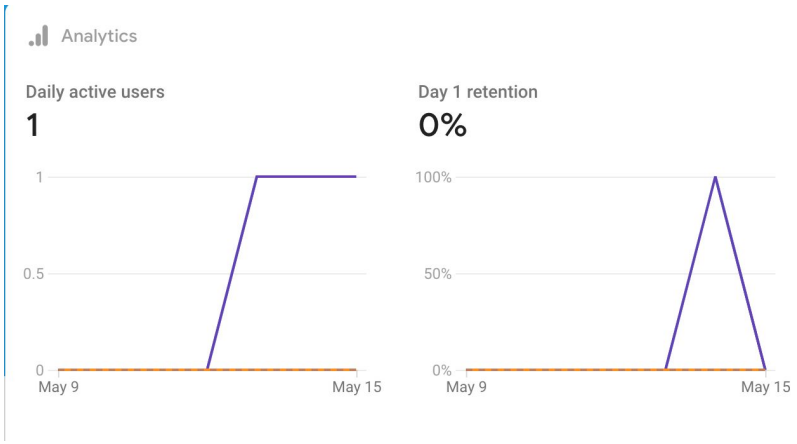


Figure 2

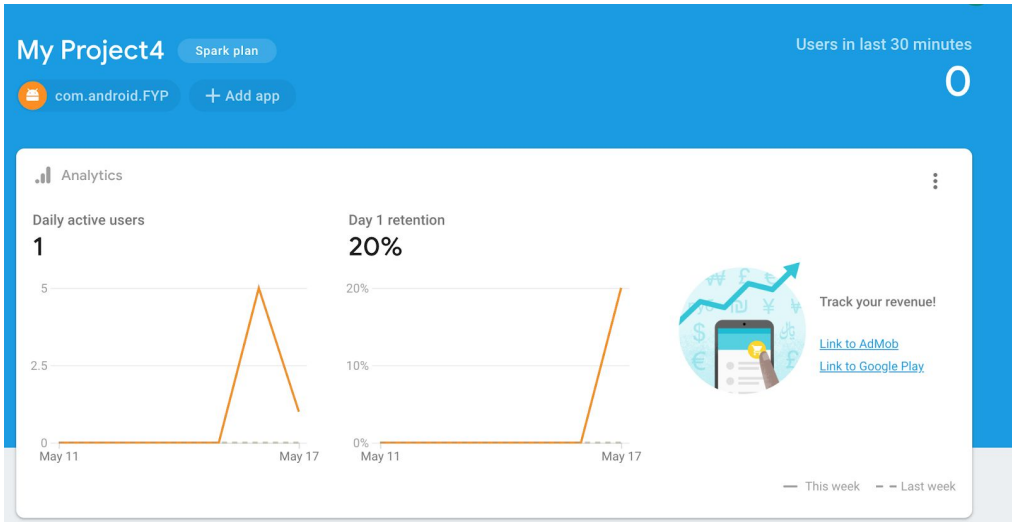


Figure 3

2.3 RealTime Database vs Cloud Firestore

Firebase offers two types of cloud-based client-accessible database solutions for mobile apps that support real-time data syncing which are real-time Database and Cloud Firestore**.** Both are NoSQL databases. Real-time databases store data as a large JSON tree which makes application storing simpler forms of data suitable to use a real-time database.

Contrary to the real-time database, Cloud Firestore stores data in documents arranged in collections. The Simple forms of data are stored in these documents while more complex forms are stored using subcollections within these documents.

2.4 Existing Methods

Existing solutions already available was similar to the problem my project is solving. I would be considering the two main functionalities of my application (chat and fitness) when looking for similar solutions to my application. The argument could be made that Facebook could be a good alternative however, the argument could be made that Facebook requires the sharing of personal information between students which in a case such as being assigned to a group project could compel a student to give their personal information e.g phone number out to the group.

2.4.1 Google Fit

Google Fit is a health-tracking platform developed by Google for the Android operating system. Google Fit makes use of sensors in order to track and record activities performed such as walking, running etc. According to Wikipedia “Google Fit provides a single set of APIs for apps and device manufacturers to store and access activity data from fitness apps and sensors on Android and other devices (like wearables, heart rate monitors or connected scales”.
Users of google fit have the option to choose who they share information related to their fitness with and also have the option to delete that information at any time of their choosing. Google Fit was originally exclusive to android however it has recently been added to IOS operating systems as well.

2.4.2 Canvas Student

The University of Birmingham also has a similar IOS application provided for their students called “Canvas Student“ which allows communication between fellow students on a more balanced none formal level. The application also provides information for students such as courses and timetable similar to “Surreylearn”, however, its main focus is communication between students rather than academics.

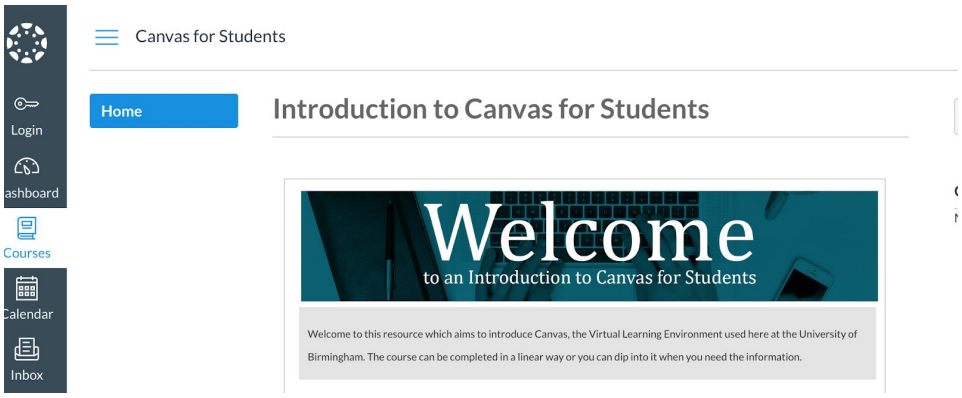


Figure 4

2.3.2 Whatsapp

WhatsApp Messenger is a freeware cross-platform messaging application. It allows users to send messages, receives messages, makes voice and video calls, it also allows users to send files and document. Whatsapp is similar to my application in the sense that both applications can send and receive messages. However, WhatsApp is far more complex than my application understandably so.

2.4.3 Samsung Health

Samsung Health Originally was a free application developed by Samsung that serves to track various aspects of daily life contributing to well being such as physical activity, diet, and sleep. Main features of Samsung health include dietary monitoring, sleep monitoring, heart rate checks, pedometer. Samsung health is similar to my application in the sense that both applications make use of pedometer functions for its users.

2.5 Market for Chat Applications

Mobile chat applications have growingly been on the rise in popularity in recent years with currently over a billion users communicating via WhatsApp, Facebook and Instagram. These applications such as Facebook provide users with enhanced means to communicate such as gifs, emojis, videos etc.

Recent research concluded that more than half of the world population make use of at least one chat application. The leading candidates are WhatsApp and facebook with about 1.2 billion and 1.9 billion respectively. These numbers also give the suggestion that chat applications are overshadowing social networking application however recently applications such as Instagram have now added chatting features to their applications in order to keep up with the market. Below is a percentage obtained from www.agriya.com on the percentage of features uses use chatting features for excluding private messaging.

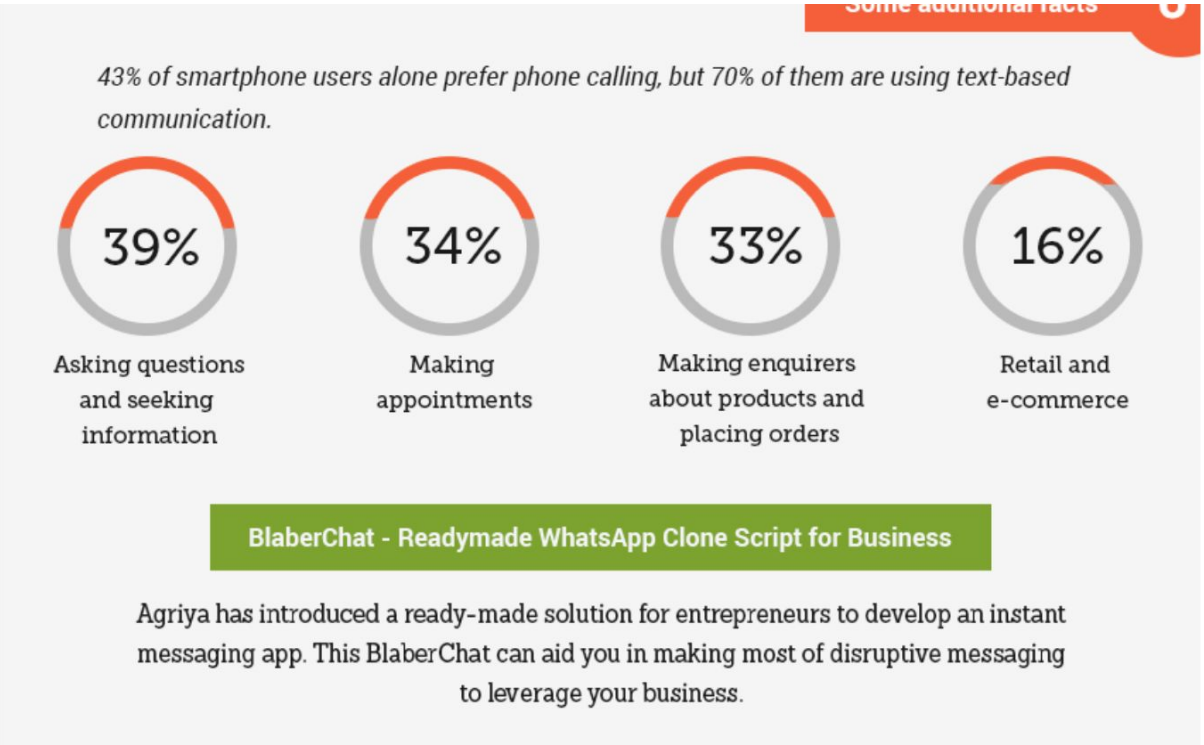


Figure 5

IOS VS Android

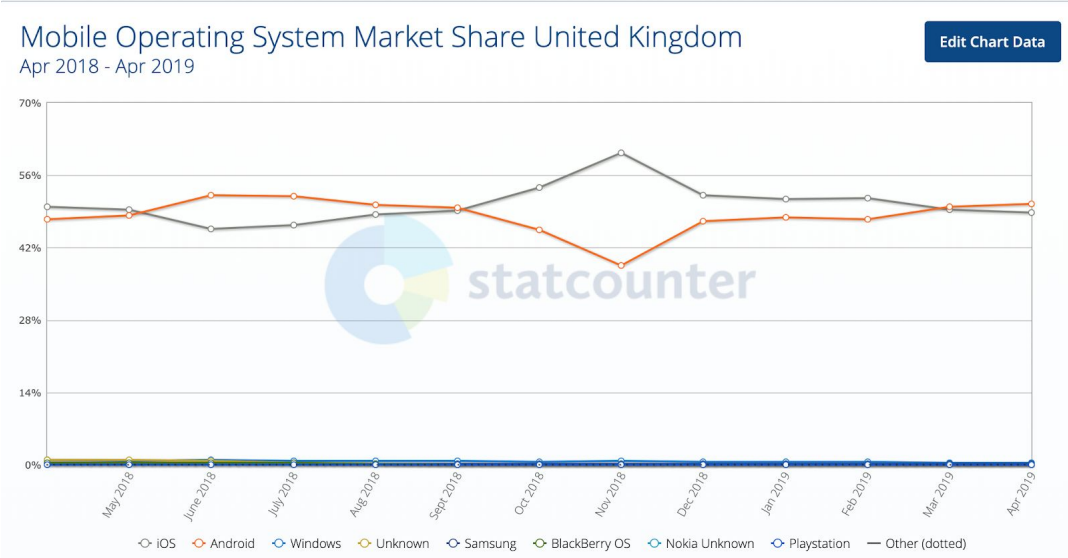


Figure 6

As seen from the charts android as well as ios are the leading choice in the market. Android vs IOS has always been a topic for mobile developers on what platform to invest their applications in. My decision to work with android for the development of my application was also economically influenced. Although the market share as seen from the picture above is quite similar the price of the android phone on an average is significantly less than the price of the iPhone. This suits my target population as students often don't have the money for the more expensive phones and settle for android phone.

Section 3: System Analysis

3.1 Introduction

System analysis is an important part of software development. It involves analyzing the system in order to get feedback on the system. The functional and nonfunctional requirements of the project will also be considered in this section.

3.2 Investigation Techniques

1. Interviews

An interview is a form of interaction used to gain more in-depth knowledge of potential users of the application I designed. An unstructured interview was carried out during the course of the project which targeted students particularly international first-year students who had not yet adjusted to the country

to try and find out if the application would be a good way for them to maintain their body health. Possible questions asked during an interview would be " how their social life is going so far at the university , "would this app be beneficial to their social university life in a case where the student answered no to the 1st question" , Do they think this application could provide a good balance between their social and fitness lifestyles respectively .

2. Surveys

A survey is a methodology used to gather information from a large group of individuals while also making sure the individuals remain anonymous. This form of interaction with potential users will prove to be the most functional as it ensures users anonymity which could steer users into giving honest opinions about the application. Hence survey was the chosen form of communication with the audience.

3. Activity Observations

Activity observation is a form of data collection in which participants are observed and information is gathered based on feedback from the observations performed. Activity observations are split into two parts participant and non-participant observations Participant observations involve when I the user will be involved in observing other similar applications while non-participant observation involves using other tools to observe. Activity observations are useful as they give a good understanding of the background of a project however it will not be considered as it is quite a time consuming compared to other forms of data gathering.

3.3 Survey Objectives

The objectives of the survey are :

- To find out what features users find interesting in the app
- To know how the target population responds to the application
- To give a visual representation and provide key findings from the application

3.4 Survey Design

A survey is a crucial part of gathering information in any business sector. However to increase the effectiveness of the survey to be carried out first we consider other factors affecting the survey such as the target population for the application, the distribution process of the survey etc.

Certain points have to be considered when designing a survey in order to maximise the results obtained from the survey. The survey design should focus on creating a user experience suitable to casual users

who just want to keep it simple. The structure of the design should be kept as simple as possible by avoiding asking too many questions, making sure every question asked is necessary, avoiding redundancy in questions asked etc. With the following considerations in mind a few questions asked from the survey were “have you ever experienced an application similar to this one “ , “ do you think in a real world scenario this application would sell “ , “ How do you think combining fitness and social media will benefit the user “ , “ what features would you like to be added to the application in the future “ . The full layout of the survey is found in the Appendix section. The survey was also hosted online and has since been removed to avoid late responses or false responses from people who already completed the original survey.

3.5 Target Population

As the application is social media based for the most part it would be targeted towards the University of Surrey students particularly those in the first year of studies as international students who haven't adjusted to life on campus. Concerning the fitness aspect of the application, it has also been discovered by “ncbi.nlm.nih.gov” who studied and concluded that obesity amongst young people causes lifetime cardiovascular problems by performing a survey to determine the extent of obesity by sampling university students from 22 universities in 22 low, middle income and emerging economy countries. The results obtained estimated about a third of the students to be obese or at risk of being overweight.

3.6 Ethical Considerations

The survey was conducted as an anonymous survey to maximise the truthfulness of results gotten from participants. As a survey was conducted and user information was taken it is important to consider the following points:

Anonymity: The survey as stated earlier is designed to remain anonymous. This helps the overall preciseness of the answers gotten from the audience.

Storage: The answers gotten from participants will only be used for the gathering of information purposes

Participation: None of the participants in the survey was forced to partake the survey. Every individual who took part in the survey did it out of their own free will.

Awareness: Survey participants were made aware beforehand of what the survey was about and what their answers would be used for in order to give the participants a clear view of what they were signing up for.

3.7 Survey Response Analysis

The survey was made available on social media and around passed out around the international house building which is where most international students reside on campus. This was in line with the targeted group for the application. The full survey questions are available in the appendix section. In total I got about 20 people to participate in the survey. Although this is a small sample size it is still big enough to have an idea of how users will feel about the application

Question 1: Would you consider this app suitable for university students?

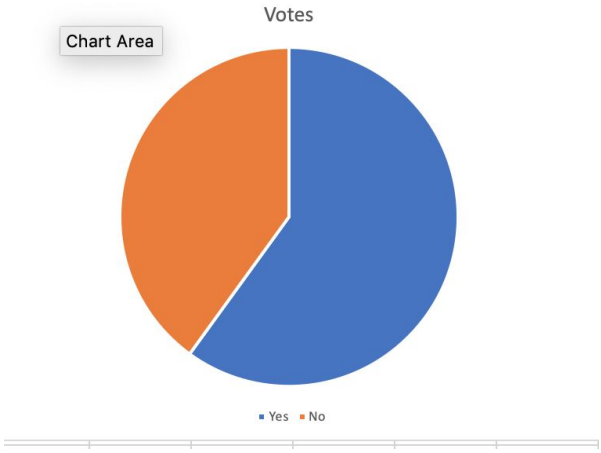


Figure 7

The feedback gotten from this question was mixed. The percentage of students answered yes to this question was about 60 per cent compared to the 40 per cent who said no. The answers gotten from this question leads on to the next question on why a survey participant answered yes or no.

Question 2: Could you please explain the reasoning behind your answer
The answers got from this section as to why users answered a yes or no was quite similar for most survey participants. Users who answered no for the most said the application wasn't suitable as they would rather just have two separate applications one for fitness and one for chatting. Users who the option of yes said they chose yes as they preferred multitasking so found it useful to have these two functionalities in the same application.

Question 3: What features would you like to be added to the application?

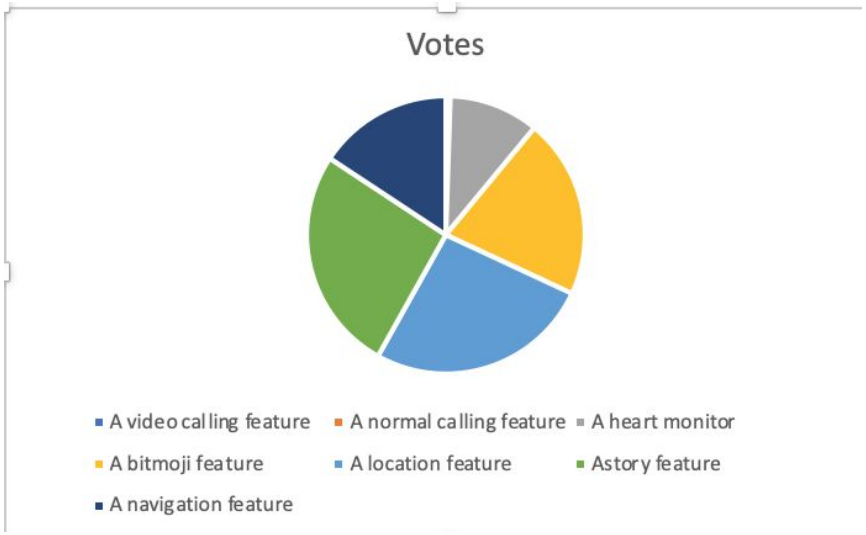


Figure 8

As seen from the results gotten most users had an inclination towards adding more features related to communication rather than fitness. This was expected as the survey participants comprise mostly of university students.

Question 4: Have you ever used any applications similar to this one?

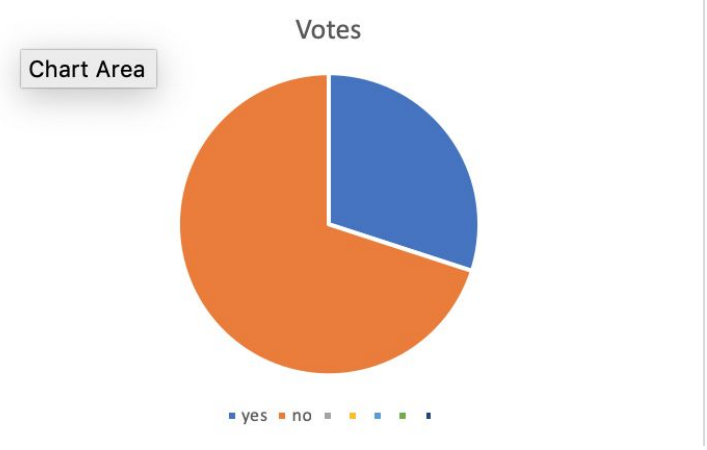


Figure 9

The answers obtained from this section were quite skewed towards no. This could be because as noted earlier most application now focuses on one core feature e.g communication which is understandable.

Question 5. If yes how do they differ from this application

The answers gotten from this section were not admissible due to the low amount of people who replied yes to question 4. The few users who answered yes made the case that other similar applications were more advanced in terms of the features that they provided in relation to my application.

Question 6: Do you think this application could be financially successful in a real-world market

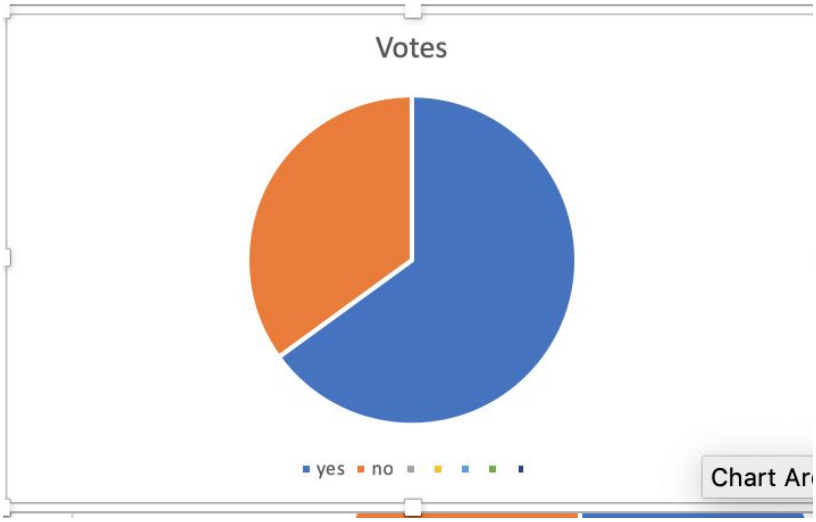


Figure 10

As seen from the chart above the answers gotten from this question were positive in nature. Most users viewed the application as having the chance to be a financially successful one.

Question 7: Could you please explain your answer to the previous section

The answers gotten from this question were similar .the anonymity aspect of the survey was crucial to this question as users were allowed to give their honest opinion to what was a very sensitive question. Many users viewed the application had a chance to succeed if it was revamped to include more features as it was something new and not many similar applications are out there. The users who said no also gave valid reasoning behind their answers saying my target population are already too heavily invested into other chat applications such as Facebook, snapchat and WhatsApp for my application to make a real push towards success.

Question 8: What Current features do you like in the application

The results obtained from this question showed a wide range of users found the summary of training section feature most likeable. This is understandable as the target population are university students who in a sense prefer things to be summarized. Users who answered yes to this question noted they preferred this for the reason stated above.

Features such as the individual chat feature of the application were not very fund to users as they stated that they already have chatting applications such as WhatsApp and Facebook with more complex chat features available which is understandable.

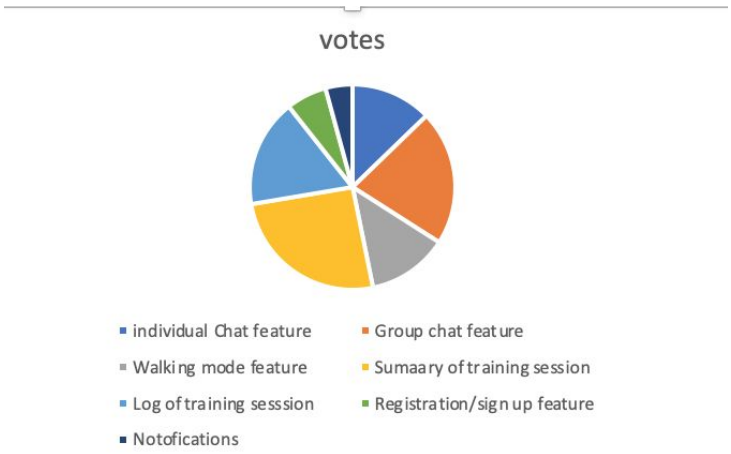


Figure 11

3.8 Survey Overall Feedback

Objective	Was it achieved	Comments
• To find out what features users find interesting in the app	Yes	This objective was met as the highest percentage of users expressed a liking towards the summary of training session feature
To give a visual representation and provide key findings from the application	Yes	This objective was met using the pie charts gotten from user feedback
To know how the target population responds to the application	Yes	The overall response to the application was mixed however the current version proposed iis only a prototype so this could change

3.9 System Requirements

System requirements are divided into two sections. Functional and non-functional requirements. Functional requirements are requirements that are user based and the user has the option of using. Non-functional requirements are the requirements the system must have to operate at the accepted level defined.

3.9.1 Functional Requirements

Requirement ID	Requirement	Description	Priority
F 1	Sign up/Register	Users can sign up by providing their username and choosing and confirming a password	Core
F 2	Sign out/Login	Already logged in users can sign out of the account and log back in at their choosing	Core
F 3	Sending and receiving messages	The system is designed to allow sending and receiving of messages between users	Core
F 4	Create a Group Chat	The system should allow users to create group chats. The user who created the group will be the group admin	Core
F 5	Add users	Users are able to add other users as friends	Core
F 6	Add users to a group	Users can add other users to a group and communicate via the group	Core
F 7	Edit Group	The Group admin can edit the group to change the group name / add or delete	Core

		members from the group	
F 8	View user profile	Users can view their personal profile	Core
F 9	Edit user profile	Users can edit their profile by adding a profile picture, change username	Core
F 10	Change password	Users can reset their password by choosing the option and the user will have a link sent to their actual email to continue the password reset	Core
F 11	Application trained to give a summary of the users training session	Users can request a new training session and will receive the time, distance in km, calories burned and the number of steps taken as feedback	Core
F 12	Log of activities	The system produces a log of a users training sessions in the form of months, weeks or days	Core
F 13	General settings	Users can adjust the settings for the application by choosing their weight, step goal, gender etc.	Non-core
F 14	Summary of the users training session	The system should provide a summary of a users training session	Core
F 15	Notifications	Notifications on what time to do a health check, time to train, distance walked etc.	Non-core
F 16	Help	Users have the option	Non-core

		to go to help page if any functions in the fitness section are not clear	
F 17	Email uniqueness	If email is already taken the system will let the user know	Core
F 18	Group requirement	At Least two other users in total are required to form a group	Core
F 19	Group admin	Only group admin can edit or delete members from a group	Core
F20	Walking modes	This feature will allow users to select a walking mode of their choosing	Non-core

3.9.2 Non Functional Requirements

Requirement ID	Requirement	Description
NF 1	Startup efficiency	The app should run within the first five seconds of execution
NF 2	Accuracy	The app should provide accurate results when calculating steps taken, calories burned etc.
NF 3	Availability	The app should be available to users if they are connected to wifi
NF 4	Compatibility	The app should be compatible with all android phones higher SDK version than 19

NF 5	Interface	The app should provide a good level of appeal to users while also being easy to navigate around
NF 6	Confidentiality	Users messages should remain confidential between the two parties sharing those messages

Section 4 - System Design

4.1 Introduction

System design is an important aspect of software development. This section explores defining the architecture, modules, interfaces, and data for our system to satisfy specified all requirements. This design phase would help up when executing the implementation phase of the project

4.2 Use case diagram of core application features

This is a good visual representation used in order to demonstrate the different ways that a user might interact with a system. The usage of a use case diagram is important as it gives a simpler identification of the various interactions between the users and the systems within an environment.

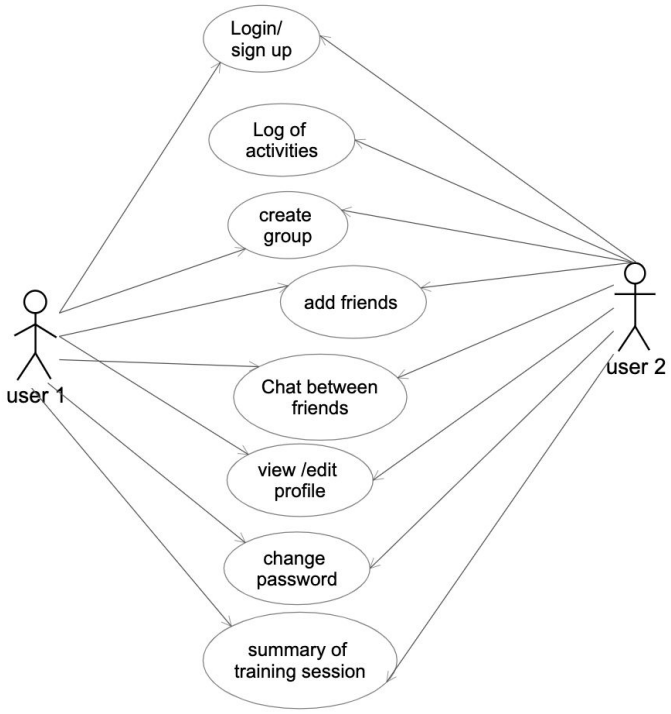


Figure 12

4.3 System Architecture

The system architecture was built using the android studio. Android is structured in the form of a software stack comprising applications, an operating system, run-time environment, middleware, services and libraries. Each layer of the stack and the corresponding elements within each layer are tightly integrated and tuned to provide the optimal environment for developing an application. A visual representation is seen below. Looking at the use case in section 4.1 all users have the same level of functionalities within the system.

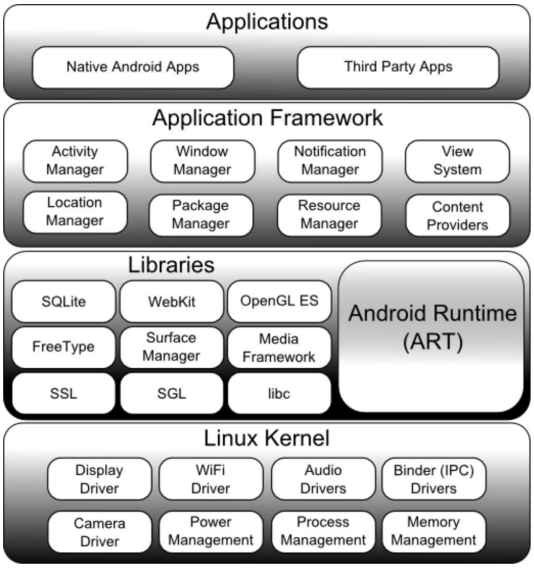


Figure 13

4.3.1 Manifest and overall Structure

When the application is first to run the first page available to users is the login page. A transition will then lead users who haven't signed up the option to register. After a user's logs in an Intent is passed and users are transferred to the main activity, From the main activity users are then free to go to any part of the application of their choosing.

Every application in the android studio has an AndroidManifest.xml file in its root directory. The manifest file presents information essential for the application structure. The manifest file also describes the components of the application such as the activities, fragments, services, broadcast receivers used etc. The manifest file is also responsible for determining the minimum API level the application will run at

The manifest file as seen below was built to follow this format.

```
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="FitChat"
android:supportsRtl="true"
android:theme="@style/AppTheme">
<activity
    android:name="com.android.FYP.MainActivity"
    android:label="chat"
    android:launchMode="singleTop"
    android:screenOrientation="portrait"
    android:theme="@style/AppTheme.NoActionBar"/>
<activity
    android:name="com.android.FYP.ui.LoginActivity"
    android:launchMode="singleTop"
    android:screenOrientation="portrait"
    android:theme="@style/TranslucentBackground">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name="com.android.FYP.ui.RegisterActivity"
    android:launchMode="singleTop"
    android:screenOrientation="portrait"
    android:theme="@style/Translucent" />
<activity
    android:name="com.android.FYP.ui.ChatActivity"
    android:screenOrientation="portrait"
    android:parentActivityName="com.android.FYP.MainActivity"/>
<activity android:name="com.android.FYP.ui.AddGroupActivity"
    android:theme="@style/MyThemeNoActionbar"/>
<service android:name="com.android.FYP.service.FriendChatService"
    android:exported="false"
```

Figure 14

In order to reduce the complexity of work, the main classes used were stored in relation to its purposes. This helped the project have a good structure and organisation. As seen from the diagram below classes that related to the user interface were placed in the UI package. Java classes related to the broadcast receivers were placed in the receiver's package and so on.

4.3.2 How classes are placed in packages

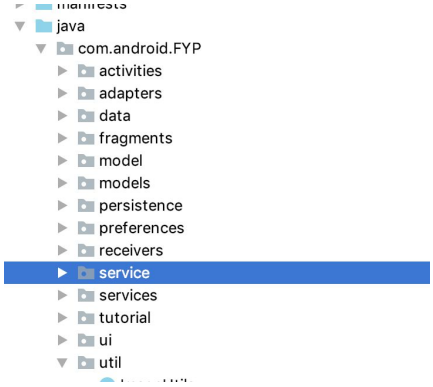


Figure 15

The architecture for the application as seen from the diagram above contains several packages such as the activities were activities are stored.

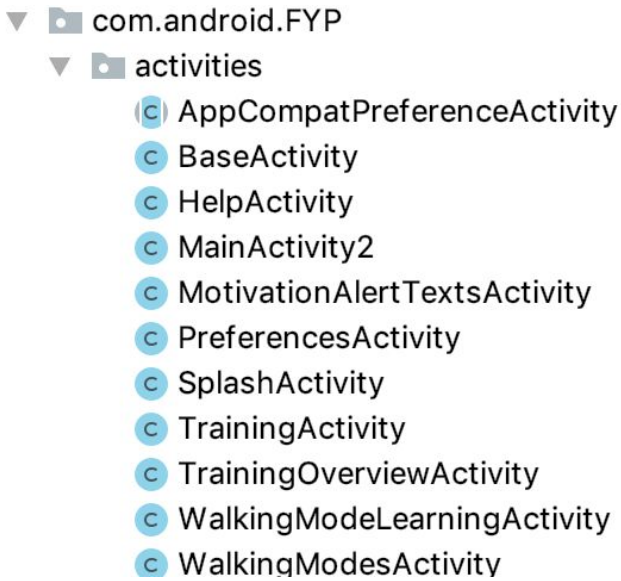


Figure 16

4.4 Mobile Application Design

When considering the design for the application I came across several websites detailing the importance of good design in any mobile app. Important aspects need to be considered when designing an application for users such as cutting out clutter on the screen in order to avoid overloading users with too much information. In all the application should be displayed to users as simple as possible. The figure below shows a good representation of managing what is displayed to users on the screen in order to avoid the complexity of the user interface.

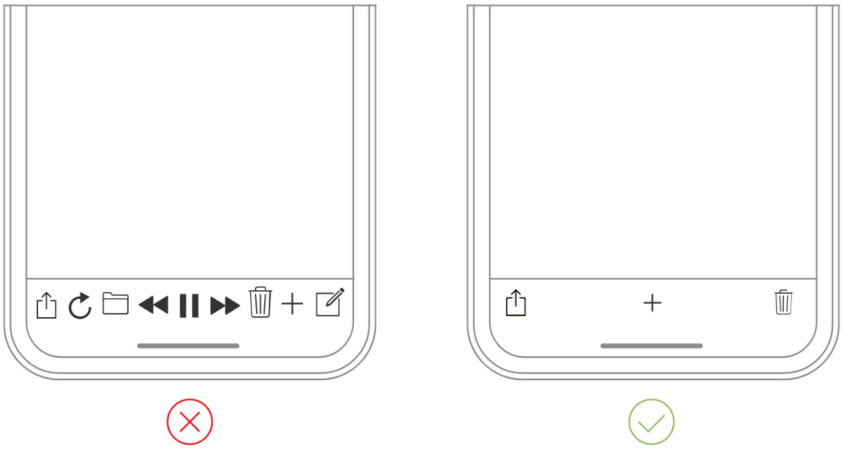


Figure 17

The user interface should also contain terms suitable and easily understandable to users. This is good to avoid so as to not steer some users away from the application. Below is an example of using clear understandable terms in the design vs unclear terms.

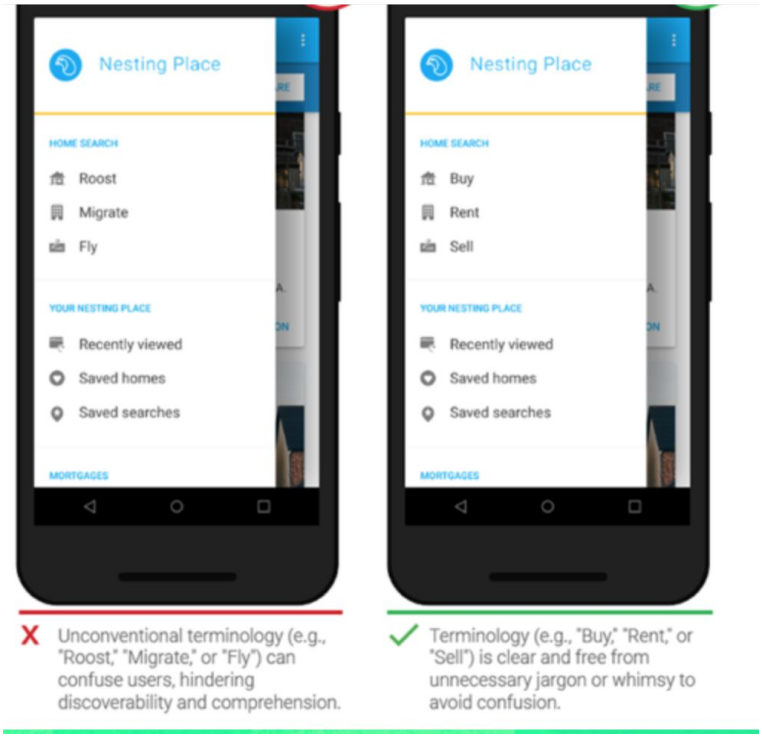


Figure 18

According to “steve hobber” of UX matters another important feature for designing the interface of an application is making sure the key content for the application is in the middle part of the page while also accounting in the design to ensure people’s fingers and thumbs don’t obscure content, so they can see what they are touching on the screen .

4.4.1 Breaking Tasks into fewer chunks

A section of the application which requires a lot of steps needed to be taken by the user should be broken down into subtasks. This is useful and will be considered when designing features in the application such as user password retrieval/user login/registration. Doing this overall helps the user avoid losing interest in the application which is one of the main focuses of the application as it requires users to remain intrigued.

These points will be considered when designing the UI of the application.

4.4.2 Using visual weight to convey the importance

This involves using the size of items on the screen as a way of conveying how important a certain icon or button is. Users of mobile applications tend to focus on the bigger things displayed in an application so maintaining a good balance when choosing the size of buttons in the application will be of importance. In the example below users will be drawn towards the 3 min icon seen as it is displayed to maximise user attention.

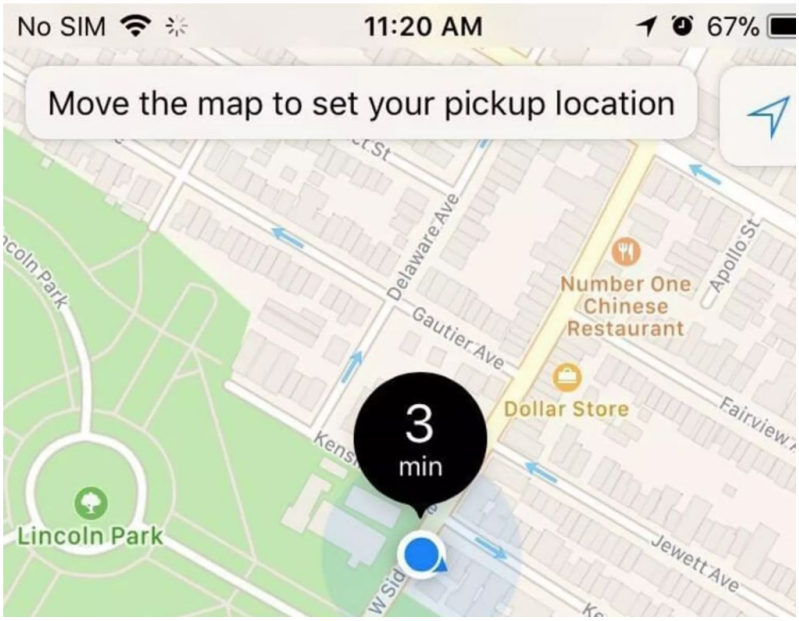


Figure 19

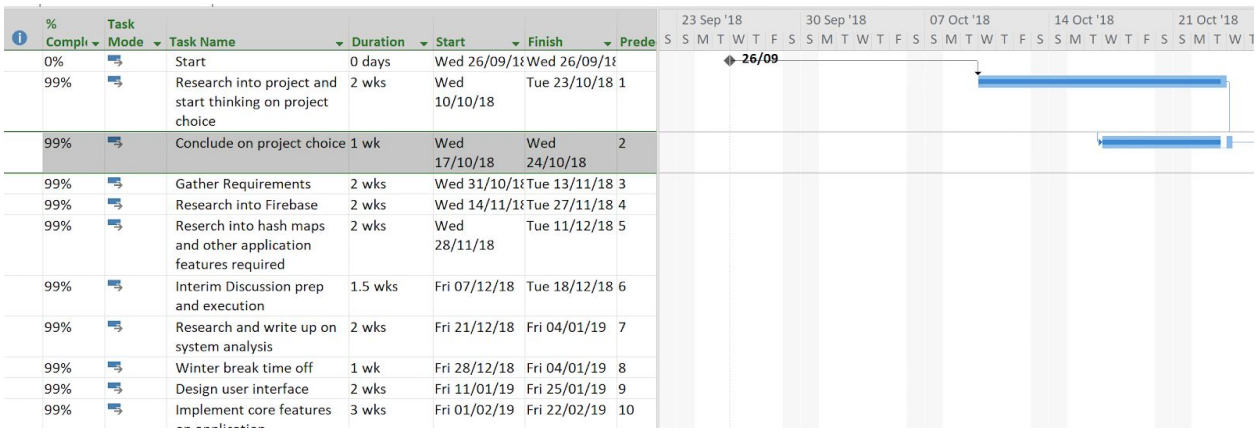
Section 5 Project Management

5.1 Introduction

Project management is important and plays a key part in the success of any project. This section provides a risk assessment determining the level of risk the project faces and how these risks can be projected. A timetable of the project lifeline will also be presented in this section.

5.2 Project Lifeline

As the project was a time consuming one with lots of milestones required in order to move on with certain parts of the project it was important to create a Gantt chart at the start of the project which helped me maintain a good state of mind with the huge bulk load of activities needed for the project to be completed successfully .Ms project was used for creating the Gantt chart, this was a skill I was lucky to pick up as I was also studying it in another module concurrently running in semester 2. The Gantt chart shows a good balance between the time taken for research the project, implement the code and document it. Using the Gantt chart was very helpful as I was able to tell the system how much per cent of a task is complete which gave me a clear view of how far complete my project was.



% Compl	Task Mode	Task Name	Duration	Start	Finish	Prede
0%		Start	0 days	Wed 26/09/18	Wed 26/09/18	
99%		Research into project and start thinking on project choice	2 wks	Wed 10/10/18	Tue 23/10/18	1
99%		Conclude on project choice	1 wk	Wed 17/10/18	Wed 24/10/18	2
99%		Gather Requirements	2 wks	Wed 31/10/18	Tue 13/11/18	3
99%		Research into Firebase	2 wks	Wed 14/11/18	Tue 27/11/18	4
99%		Reserch into hash maps and other application features required	2 wks	Wed 28/11/18	Tue 11/12/18	5
99%		Interim Discussion prep and execution	1.5 wks	Fri 07/12/18	Tue 18/12/18	6
99%		Research and write up on system analysis	2 wks	Fri 21/12/18	Fri 04/01/19	7
99%		Winter break time off	1 wk	Fri 28/12/18	Fri 04/01/19	8
99%		Design user interface	2 wks	Fri 11/01/19	Fri 25/01/19	9
99%		Implement core features	3 wks	Fri 01/02/19	Fri 22/02/19	10

Figure 20

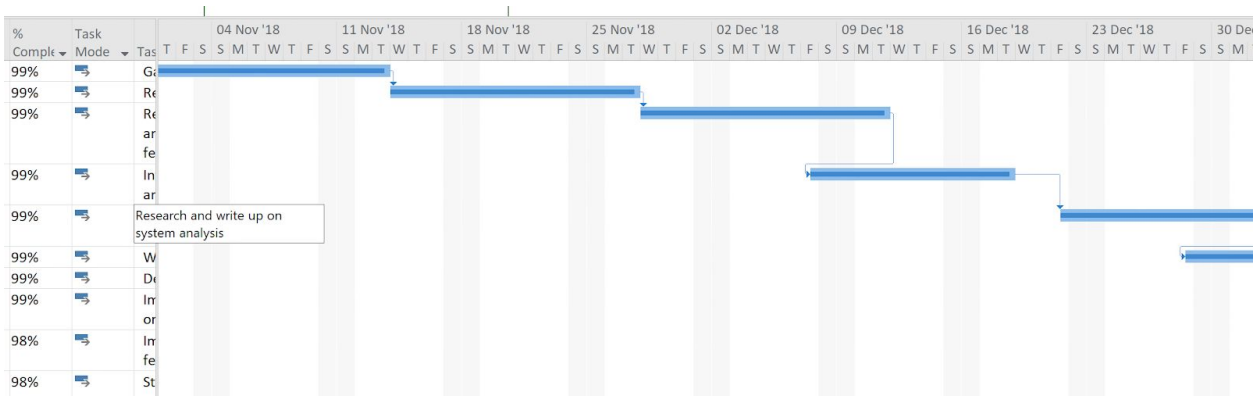


Figure 21

5.2 Software Development Process

The project will follow the software development lifecycle (SDLC). The SDLC consists of 6 phases outlined in the figure below.

Figure 22

Using this lifecycle will form the bases of how the project will run. The SDLC is also split up into different models such as the waterfall model, V-shaped model, incremental, iterative model. These different models will be discussed to determine which would best suit the application

5.3 Waterfall Method of Development

The Waterfall approach is a software development process divided into different phases. The outcome of a prior phase acts as the input for its preceding phase. The waterfall method requires all activities to complete in one phase in order to progress to the next phase meaning this methodology will not be sufficient for use in this project as different phases of the project need to be checked backed on and updated when required. It is also not suitable as once the application is in the testing stage, it will be difficult to make changes from the implementation phase assuming some tests failed.

5.4 Iterative Method of Development

Iterative development is a form of development which involves breaking the work packages of a large application into smaller units. This form of development can have each cycle being repeated with new features being added until the system is fully functional and ready to be deployed to customers. This form of development will work best for this project as certain aspects of the project will need to be revisited to be altered when needed. Hence this method of development will allow for more flexibility when changes need to be made to the application in the case of any unforeseen complication that doesn't arise initially.

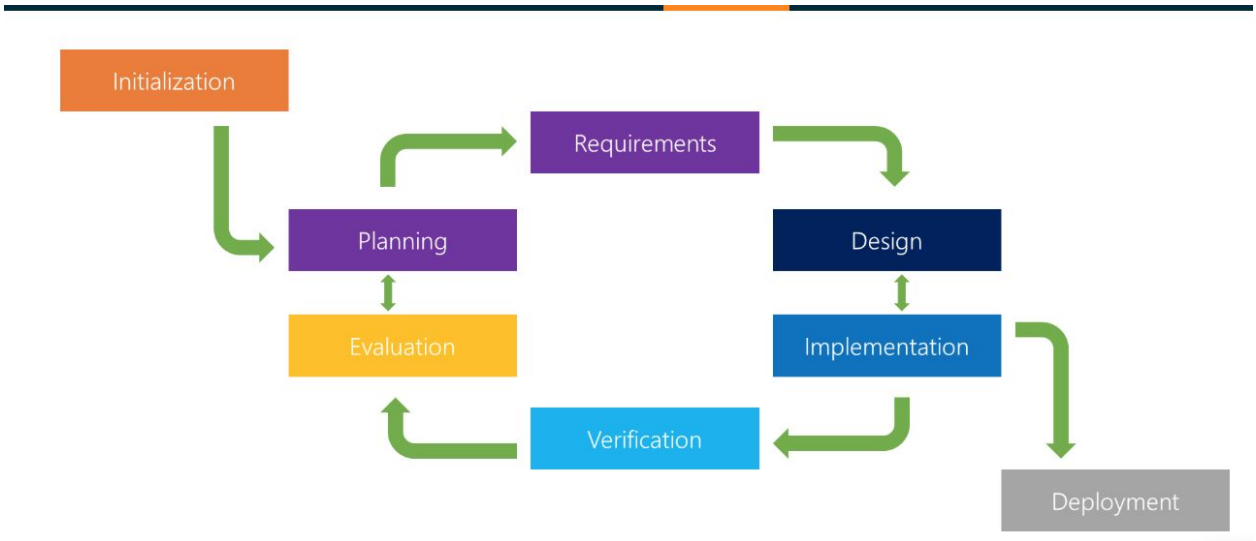


Figure 23

5.5 Risk analysis

Risk analysis is the process in which risks are identified and analyzed in order to manage any issues that could negatively impact a project. Risk analysis is an important aspect of any project so it will be considered for this project. The project will be held to standards following the CIA triad.

Confidentiality - Ensure user data on the application stays secure and inaccessible to any unauthorized users. As the application contains sensitive data such as messages sent across users it is important for all information to remain confidential to the user. For the application to remain confidential messages which are exchanged by users through a communication channel should be readable only to the intended parties.

Integrity - For an application to meet the requirements of the CIA triad they also have to have a high level of integrity. Features in the application such as the pedometer are required to have a high level of integrity in the results obtained from them in order not to mislead users. For the chatting aspect of the application to have integrity messages during the transfer of messages between two parties, the messages should not be altered. Attackers can possibly eavesdrop on messages and modify them before they get to the receiver. Hashing is a common mechanism used in information security to prevent this from occurring.

Availability- The application and all its features should be available to users at all times. This can be done by proper testing of the application to identify that all features work. The application should be available to users connected to the internet. Features in the application should also be easily accessible to users

Risks analysis will be used in the context of this project to :

- Anticipate harmful events to the project
- Plan for a response to the risk in the case of a risk occurring
- Identify the likelihood and impact a risk would have on the project

5.5.1 Risk identification assessment

Risks are identified through qualitative analysis. Qualitative analysis of risks will be the best way to evaluate the risks of this project as the qualitative analysis does not require specific numbers in order to analyse risks. As the project did not possess sufficient data for a more detailed quantitative analysis qualitative analysis is justified in its use for the project.

5.5.2 Risk Matrix

The following risk matrix is used with risks with a value of 1-3 considered as of low risk. Risks with a value of 4-7 will be considered as medium and risks with a score higher than 7 will be considered as of high risk.

	Negligible	**Minor**	**Moderate**	**Critical**	**Catastrophic**
Almost Certain	5	6	7	8	9
Likely	4	5	6	7	8
Possible	3	4	5	6	7
Unlikely	2	3	4	5	6
Rare	1	2	3	4	5

Risk id	Risk	Likelihood	Impact	Risk Score	Risk Level
R1	Unauthorized users gain access to the firebase storage used for the application	Unlikely	Catastrophic	6	Medium
R2	User forgets their login details	Possible	Moderate	5	Medium
R3	A person could purposely sign up to the	Likely	Critical	7	High

	application with another user's university email to impose a said person				
R4	Pedometer provides users with inaccurate results	Unlikely	Critical	5	Medium
R5	Accidental/un authorized deletion of user data from firebase	Unlikely	Catastrophic	6	Medium
R6	System crashes	Possible	Moderate	5	Medium
R7	The designed system is not in line with a defined scope	Possible	Critical	6	Medium
R8	Time is taken to complete a certain phase of the project overruns	Possible	Critical	6	Medium

5.5.3 Risk Response strategies

Once risks have been identified the next phase is certain up measures to respond to those risks. These measures include :

1. Mitigate - This involves reducing the level of impact or likelihood of the risk should a vulnerability be exploited
2. Transfer - Transferring of the risk to a 3rd party
3. Acceptance - This is done when the identified risk is considered low enough to be left unattended to.
4. Avoid - Change project scope and plans to eliminate the risk entirely

Applying these methods of response we come up with a risk order to counteract the identified risk from the previous section.

Risk id	Response Method	Justification
R1	Mitigate	Apart from firebases security measures, extra firewalls could be put in in place to prevent unauthorized access
R2	Mitigate	Having a recovery system in place in case a user forgets their login details
R3	Mitigate	Having a verification system in place in to prevent users from impersonating others
R4	Mitigate	By having a feature in the application to confirm from the user if the amount of steps is right or logical
R5	Mitigate	Backup of user data
R6	Retain	Depending on how often this occurs this risk is expected as the system is not of the highest standards
R7	Retain	Changes in the scope of a project of this magnitude are expected this could be based on actual skill needed to complete certain tasks
R8	Mitigate	Use of Gantt charts to carefully plan and follow a work schedule to prevent time overlaps

Section 6 - Implementation

This section of the report focuses on the implementation phase of the project. The relevant software and any complex forms of the software, as well as all external libraries used, will be identified and justified in this section. This section also gives an overview of any problems which came across during the implementation of the application.

6.1 User interface

6.1.1 Login vs Registration Activities

The first step in the implementation of the user interface was designing the login and register pages. A yellow background was used for clarity purposes and to create a bright style for the application. A transition.xml file was created and initialised when a user from the login page clicks on the new user button (green circle with a white person from figure 24 below) to create an effect to direct users to the registration page.

Login vs Register Activities

These activities followed the design technique indicated in the system design section. I Made sure the activities were broken down into smaller units in order to suit the users. A transition button was used to break down the login activity from the registration activity.

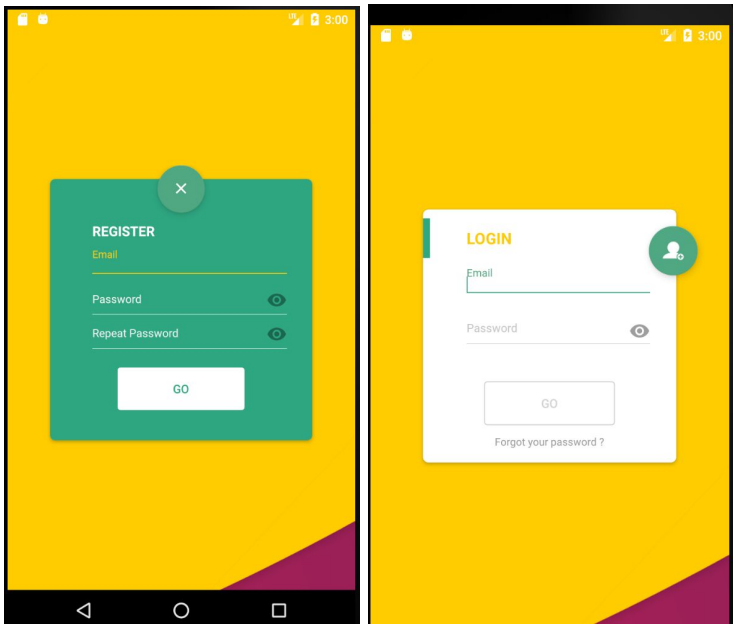


Figure 24

```
<?xml version="1.0" encoding="utf-8"?>
<transitionSet
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:interpolator/linear_out_slow_in"
    android:duration="300">

    <changeBounds>
        <arcMotion
            android:maximumAngle="0"
            android:minimumHorizontalAngle="90"
            android:minimumVerticalAngle="60"/>
    </changeBounds>
</transitionSet>
```

Figure 25

6.1.2 Main Chat page Activities

The main page of the UI was built to maintain the brightness standard set in the login and registration activities. To justify the use of this I conducted some research on brighter colours in UI design, according to UX(uxplanet.org) the use of brighter colours in an applications user interface leads users having increased readability and legibility towards the application due to contrast layout elements becoming more distinguishable and noticeable. I also noted in my research that bright colours in an application could potentially influence the mood of certain individuals. The UI created comply with the Android standard of designing and were stored in XML files(an seen in figure 27). These XML files were then called upon in the java activities when required.

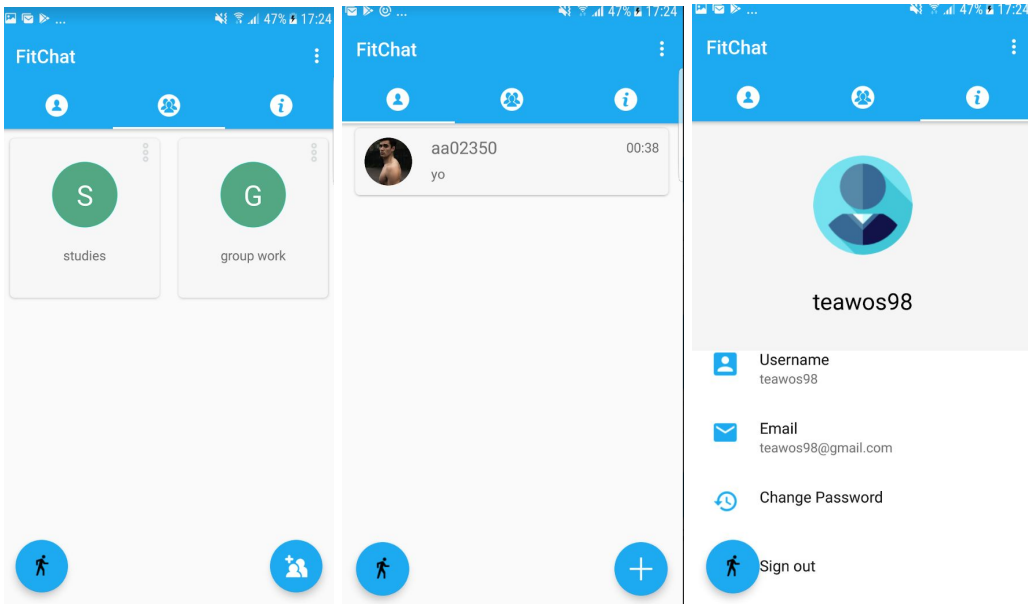


Figure 26

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="com.android.FYP.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:minHeight="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            android:id="@+id/toolbar" />

        <android.support.design.widget.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:layout_gravity="bottom"
            app:tabGravity="fill"
            app:tabMode="fixed" />

    </android.support.design.widget.AppBarLayout>
```

Figure 27

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

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        if(toolbar != null) {
            setSupportActionBar(toolbar);
            getSupportActionBar().setTitle("FitChat");
```

Figure 28

6.1.3 Fitness Activities

The user interface design aspect for the fitness activities also complied with the standards set from the chat activities. Relevant bits of information such as calories, kilometres and time spent training wee all split across the screen to create a visual user might appreciate.

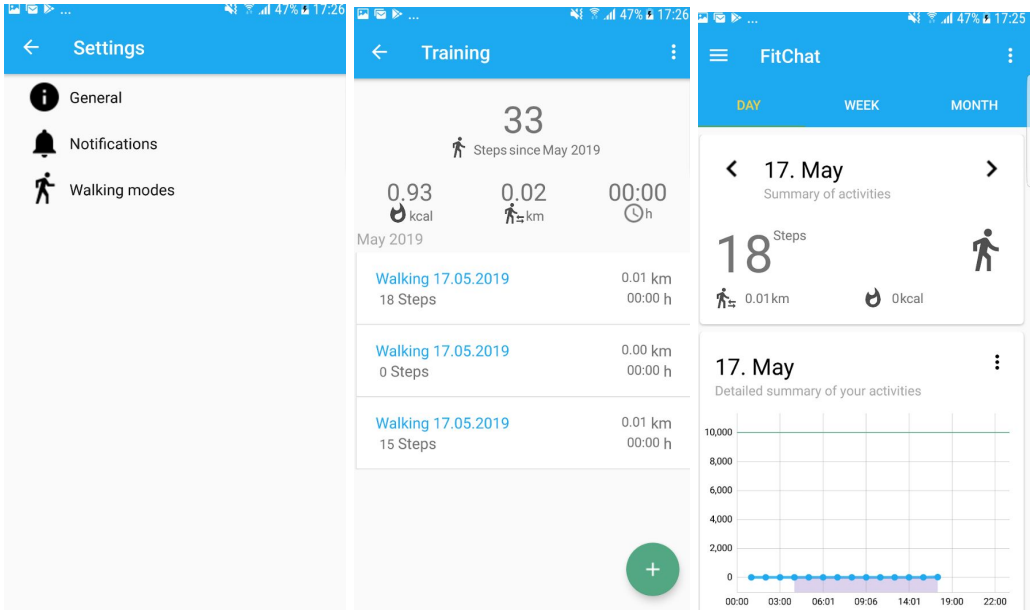


Figure 29

6.2 Implementation

The implementation of the project I would say was the most tasking aspect of the project. This required research into skills that I have learned and forgotten such as hash maps and use of services. Features that were new to me in this section were using firebase as this was the first project done that I came across it

6.2.1 Implementation of firebase into the application

Android studio allows the user to register their application an initialize it in the Gradle build. The google services.json file which contains the package name and client id of the project is obtained from the firebase console, stored in a file and initialized in the Gradle build of the application as seen below.

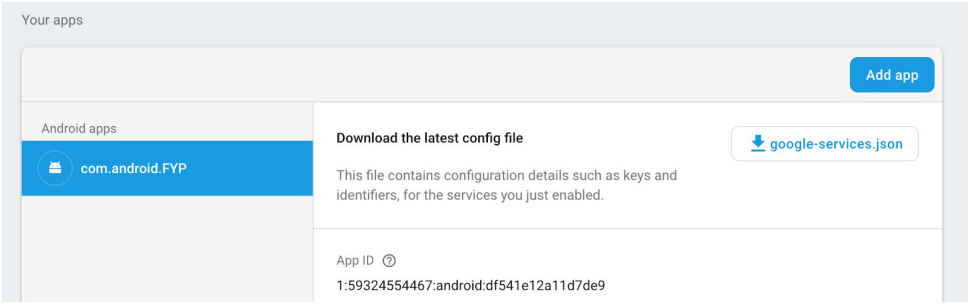


Figure 30

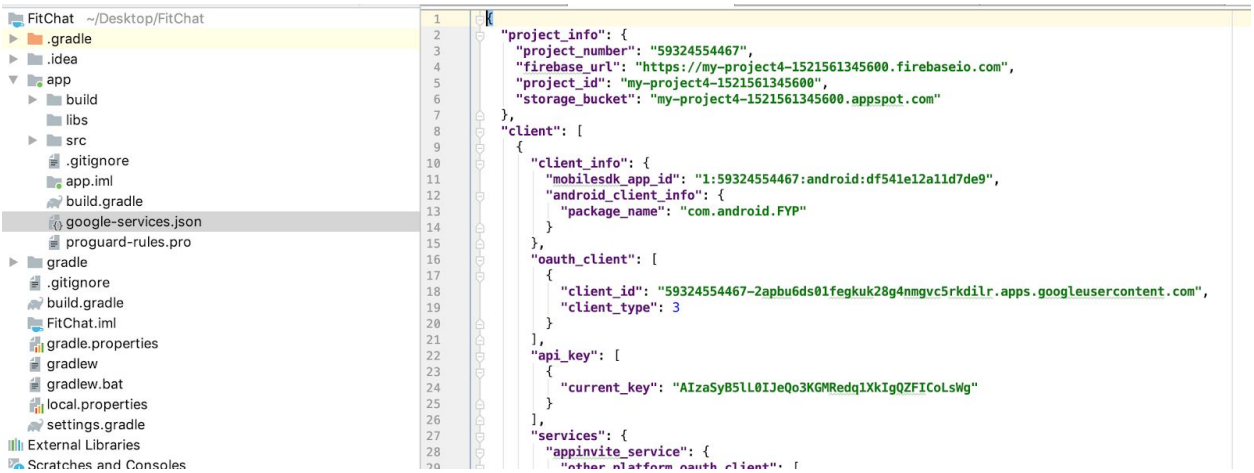


Figure 31

6.2.2 Build.gradle /External Libraries used

In order for some features in the application to work some external libraries were added to the gradle build file. Apart from the default libraries used by android studio ('com.android.support:appcompat-v7:24.2.1' , 'com.android.support:design:24.2.1') the following libraries were added to the build.gradle file.

1. Implementation of 'com.google.firebase:firebase-core:9.8.0'
2. implementation of 'com.google.firebase:firebase-database:9.8.0'
3. implementation of 'com.google.firebase:firebase-auth:9.8.0'
4. implementation of 'de.hdodenhof:circleimageview:2.1.0'
5. implementation of 'com.yarolegovich:lovely-dialog:1.0.4'
6. implementation of 'com.android.support:cardview-v7:25.0.1'
7. Implementation of 'com.android.support:percent:25.0.1'
8. Implementation of 'com.android.support:recyclerview-v7:25.2.0'
9. implementation of 'com.github.PhilJay:MPAndroidChart:v3.0.0-beta1'

The google services plugin was also applied in order to link the google services.json file obtained from firebase into the application was the

6.2.3 Java Implementation

The application had many activities so for the implementation phase, I will be focusing on the core activities and grouping them based on functionalities in order to reduce the bulk load of writing. Java was the main programming language used to implement the application.

6.3 Chat Implementation

6.3.1 Models

The models are simple java classes built which contain the fields of a specific model we are building. For example, the fields in the message model class will be sender, receiver, text and timestamp. Models depending on which also contained getters and setter methods in some instance. Apart from the message model, several other models such as friends, Group, status, training etc were used when implementing the project

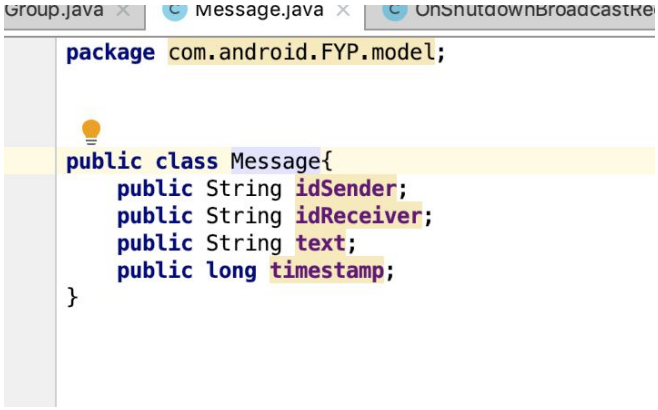


Figure 32

6.3.2 Chat Services

A service is a component that runs in the background to perform long-running operations without needing to interact with the user. Services also run if the application is destroyed. The services extend Service

6.3.3 Friend Chat Service

Among other features, this java class instantiates a hashmap containing the list and size of a user's friends.

```
public void onCreate() {
    super.onCreate();
    mapMark = new HashMap<>();
    mapQuery = new HashMap<>();
    mapChildEventListenerMap = new HashMap<>();
    listFriend = FriendDB.getInstance(this).getListFriend();
    listGroup = GroupDB.getInstance(this).getListGroups();
    listKey = new ArrayList<>();
    mapBitmap = new HashMap<>();
    updateOnline = new CountDownTimer(System.currentTimeMillis(), StaticConfig.TIME_TO_REFRESH)
        @Override
        public void onTick(long l) { ServiceUtils.updateUserStatus(getApplicationContext()); }

        @Override
        public void onFinish() {

        }
    };
    updateOnline.start();

    if (listFriend.getListFriend().size() > 0 || listGroup.size() > 0) {
        //Register to listen to the rooms here
        for (final Friend friend : listFriend.getListFriend()) {
            if (!listKey.contains(friend.idRoom)) {
                mapQuery.put(friend.idRoom, FirebaseDatabase.getInstance().getReference().child
                mapChildEventListenerMap.put(friend.idRoom, new ChildEventListener() {
                    @Override
                    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
                        if (mapMark.get(friend.idRoom) != null && mapMark.get(friend.idRoom)) {
                            Toast.makeText(FriendChatService.this, friend.name + ": " + ((Has
                            if (mapBitmap.get(friend.idRoom) == null) {
```

Figure 33

6.3.4 ServiceUtils

This java class controls the service for connections between friends. It checks and makes updates to the connection based on a users status. The java class contains a method called updateFriendstaus in this method the system performs checks to find out if a users friend is online.

```
    private static ServiceConnection connectionServiceFriendChatForStart = null;
    private static ServiceConnection connectionServiceFriendChatForDestroy = null;

@   public static boolean isServiceFriendChatRunning(Context context) {
        Class<?> serviceClass = FriendChatService.class;
        ActivityManager manager = (ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE);
        for (ActivityManager.RunningServiceInfo service : manager.getRunningServices(Integer.MAX_VALUE))
            if (serviceClass.getName().equals(service.service.getClassName())) {
                return true;
            }
        }
        return false;
    }

    public static void stopServiceFriendChat(Context context, final boolean kill) {
        if (isServiceFriendChatRunning(context)) {
            Intent intent = new Intent(context, FriendChatService.class);
            if (connectionServiceFriendChatForDestroy != null) {
                context.unbindService(connectionServiceFriendChatForDestroy);
            }
            connectionServiceFriendChatForDestroy = new ServiceConnection() {
                @Override
                public void onServiceConnected(ComponentName className,
                                               IBinder service) {
                    FriendChatService.LocalBinder binder = (FriendChatService.LocalBinder) service;
                    binder.getService().stopSelf();
                }

                @Override
                public void onServiceDisconnected(ComponentName arg0) {
                }
            };
            context.bindService(intent, connectionServiceFriendChatForDestroy, Context.BIND_NOT_FOREGROUN
```

Figure 34

```
public static void updateFriendStatus(Context context, ListFriend listFriend){
    if(isNetworkConnected(context)) {
        for (Friend friend : listFriend.getListFriend()) {
            final String fid = friend.id;
            FirebaseDatabase.getInstance().getReference().child("user/" + fid + "/status").addListene
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    if (dataSnapshot.getValue() != null) {
                        HashMap mapStatus = (HashMap) dataSnapshot.getValue();
                        if ((boolean) mapStatus.get("isOnline") && (System.currentTimeMillis() - (long
                            FirebaseDatabase.getInstance().getReference().child("user/" + fid + "/sta
                        }
                    }
                }

                @Override
                public void onCancelled(DatabaseError databaseError) {
erviceUtils
```

Figure 35

6.3.5 UI

The main UI implementation is made up of the following Java classes :

1. AddGroup Activity
2. Chat Activity
3. LoginActivity
4. RegisterActivity

AddGroup Activity

This activity is used to add and deletes a group made by a user. It calls on firebase to get an instance in order to create an id before the group can be added. In the class is a method called creteGroup which gets the id of the group and then puts in the name of the group which the user defines and the admin of the group(the person who created the group). This class also makes use of the external library **'com.yarolegovich: lovely-dialogue:1.0.4'** for the design of the buttons.

```
private void createGroup() {
    //Show dialog wait
    dialogWait.setIcon(R.drawable.ic_add_group_dialog)
            .setTitle("Registering....")
            .setTopColorRes(R.color.colorPrimary)
            .show();

    final String idGroup = (StaticConfig.UID + System.currentTimeMillis()).hashCode() + "";
    Room room = new Room();
    for (String id : listIDChoose) {
        room.member.add(id);
    }
    room.groupInfo.put( k: "name", editTextGroupName.getText().toString());
    room.groupInfo.put( k: "admin", StaticConfig.UID);
    FirebaseDatabase.getInstance().getReference().child("group/" + idGroup).setValue(room).addOnComp
            addRoomForUser(idGroup,  userIndex: 0);
    });
}
```

Figure 36

Chat Activity

This activity is used for sending and receiving messages in the application. When a message is sent the system acknowledges the user sending the message, passes it on to the receiver with the text and a timestamp is given for the time the message was delivered. This activity extends AppCompatActivity and implements View.OnClickListener. This class also allow users to write and send messages

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_chat);
    Intent intentData = getIntent();
    idFriend = intentData.getCharSequenceArrayListExtra(StaticConfig.INTENT_KEY_CHAT_ID);
    roomId = intentData.getStringExtra(StaticConfig.INTENT_KEY_CHAT_ROOM_ID);
    String nameFriend = intentData.getStringExtra(StaticConfig.INTENT_KEY_CHAT_FRIEND);

    consersation = new Consersation();
    btnSend = (ImageButton) findViewById(R.id.btnSend);
    btnSend.setOnClickListener(this);

    String base64AvataUser = SharedPreferenceHelper.getInstance(this).getUserInfo().avata;
    if (!base64AvataUser.equals(StaticConfig.STR_DEFAULT_BASE64)) {
        byte[] decodedString = Base64.decode(base64AvataUser, Base64.DEFAULT);
        bitmapAvataUser = BitmapFactory.decodeByteArray(decodedString,  offset: 0, decodedString.lengt
    } else {
        bitmapAvataUser = null;
    }

    editWriteMessage = (EditText) findViewById(R.id.editWriteMessage);
    if (idFriend != null && nameFriend != null) {
        getSupportActionBar().setTitle(nameFriend);
        linearLayoutManager = new LinearLayoutManager( context: this, LinearLayoutManager.VERTICAL,
```

Figure 37

LoginActivity

This activity is used for users who already have accounts and are looking to log back into the application. There is a validation method as seen below which matches email addresses to make sure that email has previously been signed up to the application. Firebase is also initialised in this class to allow users to log back into the application. Firebase saves an ID for each individual user, this ID is what is used to link users back to their accounts.

```
private boolean validate(String emailStr, String password) {
    Matcher matcher = VALID_EMAIL_ADDRESS_REGEX.matcher(emailStr);
    return (password.length() > 0 || password.equals(";")) && matcher.find();
}

public void clickResetPassword(View view) {
```

Figure 38

```
 *
 * Initialize components necessary for log management
 */
private void initFirebase() {
    //Initialize the component to login, register
    mAuth = FirebaseAuth.getInstance();
    authUtils = new AuthUtils();
    mAuthListener = (AuthStateListener) (firebaseAuth) → {
            user = firebaseAuth.getCurrentUser();
            if (user != null) {
                // User is signed in
                StaticConfig.UID = user.getUid();
                Log.d(TAG,  msg: "onAuthStateChanged:signed_in:" + user.getUid());
                if (firstTimeAccess) {
                    startActivity(new Intent( packageContext: LoginActivity.this, MainActivity.class));
                    LoginActivity.this.finish();
                }
            } else {
                Log.d(TAG,  msg: "onAuthStateChanged:signed_out");
            }
            firstTimeAccess = false;
    };
```

Figure 39

Register Activity

The registered activity is called when a new user wants to register to the application. A REGEX check is done to make sure users emails match the required format. Currently, the format is allowing all email types. In practice, I could have made this just for university of surrey emails however as I don't have the permissions from the university I opted not to do this. The registered activity also contains a method to validate a users password after it is entered twice to make sure the two passwords are matching.

```
public class RegisterActivity extends AppCompatActivity {
    FloatingActionButton fab;
    CardView cvAdd;
    private final Pattern VALID_EMAIL_ADDRESS_REGEX =
            Pattern.compile( regex: "^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$", Pattern.CASE_INSENSITIVE)
    private EditText editTextUsername, editTextPassword, editTextRepeatPassword;
    public static String STR_EXTRA_ACTION_REGISTER = "register";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        fab = (FloatingActionButton) findViewById(R.id.fab);
        cvAdd = (CardView) findViewById(R.id.cv_add);
        editTextUsername = (EditText) findViewById(R.id.et_username);
        editTextPassword = (EditText) findViewById(R.id.et_password);
        editTextRepeatPassword = (EditText) findViewById(R.id.et_repeatpassword);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            ShowEnterAnimation();
        }
        fab.setOnClickListener((v) → { animateRevealClose(); });
    }
    private void ShowEnterAnimation() {
```

Figure 40

```
     */
    private boolean validate(String emailStr, String password, String repeatPassword) {
        Matcher matcher = VALID_EMAIL_ADDRESS_REGEX .matcher(emailStr);
        return password.length() > 0 && repeatPassword.equals(password) && matcher.find();
    }
}
```

Figure 41

6.3.6 UI Fragments

The java classes UserProfileFragment, FriendsFragment and Group Fragment create the fragments the three main screens displayed in the chat section.

```
public class GroupFragment extends Fragment implements SwipeRefreshLayout.OnRefreshListener{
    private RecyclerView recyclerListGroups;
    public FragGroupClickFloatButton onClickFloatButton;
    private ArrayList<Group> listGroup;
    private ListGroupsAdapter adapter;
    private SwipeRefreshLayout mSwipeRefreshLayout;
    public static final int CONTEXT_MENU_DELETE = 1;
    public static final int CONTEXT_MENU_EDIT = 2;
    public static final int CONTEXT_MENU_LEAVE = 3;
    public static final int REQUEST_EDIT_GROUP = 0;
    public static final String CONTEXT_MENU_KEY_INTENT_DATA_POS = "pos";

    LovelyProgressDialog progressDialog, waitingLeavingGroup;

    public GroupFragment() {
        // Required empty public constructor
    }

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

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View layout = inflater.inflate(R.layout.fragment_group, container,  attachToRoot: false);

        listGroup = GroupDB.getInstance(getContext()).getListGroups();
        recyclerListGroups = (RecyclerView) layout.findViewById(R.id.recycleListGroup);
        mSwipeRefreshLayout = (SwipeRefreshLayout) layout.findViewById(R.id.swipeRefreshLayout);
        mSwipeRefreshLayout.setOnRefreshListener(this);
        GridLayoutManager layoutManager = new GridLayoutManager(getContext(),  spanCount: 2);
        recyclerListGroups.setLayoutManager(layoutManager);
        adapter = new ListGroupsAdapter(getContext(), listGroup);
        recyclerListGroups.setAdapter(adapter);
        onClickFloatButton = new FragGroupClickFloatButton();
        progressDialog = new LovelyProgressDialog(getContext())
                .setCancelable(false)
```

Figure 42

These three fragments are then initialized in the setupViewPager method by using adapters to add the fragments.

```
private void setupViewPager(ViewPager viewPager) {
    adapter = new ViewPagerAdapter(getSupportFragmentManager());
    adapter.addFrag(new FriendsFragment(), STR_FRIEND_FRAGMENT);
    adapter.addFrag(new GroupFragment(), STR_GROUP_FRAGMENT);
    adapter.addFrag(new UserProfileFragment(), STR_INFO_FRAGMENT);

    floatButton.setOnClickListener(((FriendsFragment) adapter.getItem( position: 0)).onClickFloatButton.

    viewPager.setAdapter(adapter);
    viewPager.setOffscreenPageLimit(3);
    viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
        @Override
        public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {

        }

        @Override
        public void onPageSelected(int position) {
            ServiceUtils.stopServiceFriendChat(MainActivity.this.getApplicationContext(), kill: false)
            if (adapter.getItem(position) instanceof FriendsFragment) {
                floatButton.setVisibility(View.VISIBLE);
                floatButton.setOnClickListener(((FriendsFragment) adapter.getItem(position)).onClickF
                floatButton.setImageResource(R.drawable.plus);
            } else if (adapter.getItem(position) instanceof GroupFragment) {
                floatButton.setVisibility(View.VISIBLE);
                floatButton.setOnClickListener(((GroupFragment) adapter.getItem(position)).onClickFlo
                floatButton.setImageResource(R.drawable.ic_float_add_group);
            } else {
                floatButton.setVisibility(View.GONE);
            }
```

Figure 43

6.3.7 Main Activity

The main activity is where all the previous fragments talked about are initialised and used to build the core of the application. Firebase is also initialised in the main activity as seen below

```
private void initFirebase() {
    //Initialize the component to login and register
    mAuth = FirebaseAuth.getInstance();
    mAuthListener = (AuthStateListener) (firebaseAuth) → {
            user = firebaseAuth.getCurrentUser();
            if (user != null) {
                StaticConfig.UID = user.getUid();
            } else {
                MainActivity.this.finish();
                // User is signed in
                startActivity(new Intent( packageContext: MainActivity.this, LoginActivity.class));
                Log.d(TAG, msg: "onAuthStateChanged:signed_out");
            }
            // ...
    };
}
```

Figure 44

6.4 Fitness Implementation

6.4.1 Database

Training data, as well as other forms of data, are stored locally using SQLite. SQLite stores the database and all its definitions, tables and data as a single file on the host machine.SQLite read operations can be multitasked and write operations can be performed sequentially.

```
public class StepCountDbHelper  extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 2;
    public static final String DATABASE_NAME = "StepCount.db";

    private static final String INTEGER_TYPE = " INTEGER";
    private static final String COMMA_SEP = ",";
    private static final String SQL_CREATE_ENTRIES =
            "CREATE TABLE " + StepCountEntry.TABLE_NAME + " (" +
                    StepCountEntry._ID + " INTEGER PRIMARY KEY," +
                    StepCountEntry.COLUMN_NAME_STEP_COUNT + INTEGER_TYPE + COMMA_SEP +
                    StepCountEntry.COLUMN_NAME_WALKING_MODE + INTEGER_TYPE + COMMA_SEP +
                    StepCountEntry.COLUMN_NAME_TIMESTAMP + INTEGER_TYPE +
            " )";

    public StepCountDbHelper(Context context) {
        super(cont + DATABASE_NAME  factory: null, DATABASE_VERSION);
    }
constructor 'StepCountDbHelper()' more... (⌘F1)
    public void onCreate(SQLiteDatabase db) { db.execSQL(SQL_CREATE_ENTRIES); }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Fill when upgrading DB
    }
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }

    /* Inner class that defines the table contents */
    public static abstract class StepCountEntry implements BaseColumns {
        public static final String TABLE_NAME = "stepcount";
        public static final String COLUMN_NAME_STEP_COUNT = "stepcount";
        public static final String COLUMN_NAME_WALKING_MODE = "walking_mode";
        public static final String COLUMN_NAME_TIMESTAMP = "timestamp";
```

Figure 45

```
*/
public class TrainingDbHelper extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "TrainingSessions.db";

    private static final String INTEGER_TYPE = " INTEGER";
    private static final String STRING_TYPE = " TEXT";
    private static final String REAL_TYPE = " REAL";

    private static final String COMMA_SEP = ",";
    private static final String SQL_CREATE_ENTRIES =
            "CREATE TABLE " + TrainingSessionEntry.TABLE_NAME + " (" +
                    TrainingSessionEntry._ID + " INTEGER PRIMARY KEY," +
                    TrainingSessionEntry.COLUMN_NAME_NAME + STRING_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_DESCRIPTION + STRING_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_STEPS + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_DISTANCE + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_CALORIES + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_START + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_END + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_FEELING + REAL_TYPE +
            " )";

    public TrainingDbHelper(Context context) {
        super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) { db.execSQL(SQL_CREATE_ENTRIES); }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Fill when upgrading DB
    }
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

Figure 46

6.4.2 Broadcast Receivers

Broadcast receivers are Android components which allow registration of application events. Once an event occurs the receivers for the event are notified and a notification is sent out.
The implementing class for a receiver extends the BroadcastReceiver class

If the event for which the broadcast receiver has registered happens, The onReceive method is called by android when the event of the broadcast receiver has been registered

They are then kept and registered in the manifest file and activated to work in the background

```
*/
public class MotivationAlertReceiver extends WakefulBroadcastReceiver {
    public static final int NOTIFICATION_ID = 0;
    private static final String LOG_CLASS = MotivationAlertReceiver.class.getName();
    private Context context;
    private AbstractStepDetectorService.StepDetectorBinder myBinder = null;


    private ServiceConnection mServiceConnection = new ServiceConnection() {

        @Override
        public void onServiceDisconnected(ComponentName name) { myBinder = null; }

        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            myBinder = (AbstractStepDetectorService.StepDetectorBinder) service;
            motivate();

            context.getApplicationContext().unbindService(mServiceConnection);
        }
    };

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i(LOG_CLASS,  msg: "Motivate the user!");

        this.context = context;

        SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(context);
        float criterion = Float.parseFloat(sharedPref.getString("com.android.FYP.pref.motivation_alert
        if (criterion < 0 || criterion > 100) {
```

Figure 47

```
    </activity>

    <receiver android:name="com.android.FYP.receivers.OnBootCompletedBroadcastReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
    </receiver>
    <receiver android:name="com.android.FYP.receivers.OnShutdownBroadcastReceiver">
        <intent-filter>
            <action android:name="android.intent.action.ACTION_SHUTDOWN" />
            <!-- this is for some htc devices (and others) -->
            <action android:name="android.intent.action.QUICKBOOT_POWEROFF" />
        </intent-filter>
    </receiver>
    <receiver android:name="com.android.FYP.receivers.OnPackageReplacedBroadcastReceiver">
        <intent-filter>
            <action android:name="android.intent.action.MY_PACKAGE_REPLACED" />
        </intent-filter>
    </receiver>
    <receiver
        android:name="com.android.FYP.receivers.StepCountPersistenceReceiver"
        android:enabled="true"
```

Figure 48

6.4.3 Persistent Classes

Difference persistent classes were used to store data related to the fitness features such as steps, walking modes etc. All persistent classes extend SqliteOpenHelper. They are currently six persistence classes used for the fitness features these include :

1. SteCountDB Helper - This database helper is responsible for storing steps of a user. The database stores the number of steps taken along with a timestamp and the number of steps since the last entry.

```
public class StepCountDbHelper  extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 2;
    public static final String DATABASE_NAME = "StepCount.db";

    private static final String INTEGER_TYPE = " INTEGER";
    private static final String COMMA_SEP = ",";
    private static final String SQL_CREATE_ENTRIES =
            "CREATE TABLE " + StepCountEntry.TABLE_NAME + " (" +
                    StepCountEntry._ID + " INTEGER PRIMARY KEY," +
                    StepCountEntry.COLUMN_NAME_STEP_COUNT + INTEGER_TYPE + COMMA_SEP +
                    StepCountEntry.COLUMN_NAME_WALKING_MODE + INTEGER_TYPE + COMMA_SEP
                    StepCountEntry.COLUMN_NAME_TIMESTAMP + INTEGER_TYPE +
            " )";

    public StepCountDbHelper(Context context) {
        super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) { db.execSQL(SQL_CREATE_ENTRIES); }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

Figure 49

2. Step Count Persistence Helper - Helper to save and restore step count from the database. The method public static List<step Count> in this class creates the list of a users steps, sets the start and the timestamp between start and finish of the training session.

```
/**
 * Returns the stepCount models for the steps walked in the given interval.
 *
 * @param start_time The start time
 * @param end_time   The end time
 * @param context    The application context
 * @return The @{see StepCount}-Models between start and end time
 */
public static List<StepCount> getStepCountsForInterval(long start_time, long end_time, Context conte
    if (context == null) {
        Log.e(LOG_CLASS,  msg: "Cannot get step count - context is null");
        return new ArrayList<>();
    }
    Cursor c = getDB(context).query(StepCountDbHelper.StepCountEntry.TABLE_NAME,
            new String[]{StepCountDbHelper.StepCountEntry.COLUMN_NAME_STEP_COUNT, StepCountDbHelper.
             selection: StepCountDbHelper.StepCountEntry.COLUMN_NAME_TIMESTAMP + " >= ? AND " + StepCou
                    String.valueOf(end_time)},  groupBy: null,  having: null,  orderBy: null);
    List<StepCount> steps = new ArrayList<>();
    long start = start_time;
    int sum = 0;
    while (c.moveToNext()) {
        StepCount s = new StepCount();
        s.setStartTime(start);
        s.setEndTime(c.getLong(c.getColumnIndex(StepCountDbHelper.StepCountEntry.COLUMN_NAME_TIMESTA
        s.setStepCount(c.getInt(c.getColumnIndex(StepCountDbHelper.StepCountEntry.COLUMN_NAME_STEP_C
        //Log.w("ASDF", "Getting walking mode " + c.getLong(c.getColumnIndex(StepCountDbHelper.StepC
        s.setWalkingMode(WalkingModePersistenceHelper.getItem(c.getLong(c.getColumnIndex(StepCountDb
        steps.add(s);
```

Figure 50

3. Training DBHelper - This database was used for storing walking modes

```
/**
 * Database helper class for storing walking modes
 *
 * @author Anthony Awobasivwe
 */
public class TrainingDbHelper extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "TrainingSessions.db";

    private static final String INTEGER_TYPE = " INTEGER";
    private static final String STRING_TYPE = " TEXT";
    private static final String REAL_TYPE = " REAL";

    private static final String COMMA_SEP = ",";
    private static final String SQL_CREATE_ENTRIES =
            "CREATE TABLE " + TrainingSessionEntry.TABLE_NAME + " (" +
                    TrainingSessionEntry._ID + " INTEGER PRIMARY KEY," +
                    TrainingSessionEntry.COLUMN_NAME_NAME + STRING_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_DESCRIPTION + STRING_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_STEPS + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_DISTANCE + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_CALORIES + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_START + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_END + REAL_TYPE + COMMA_SEP +
                    TrainingSessionEntry.COLUMN_NAME_FEELING + REAL_TYPE +
```

Figure 51

4. Training Persistence Helper- This helper class was used to save and restore training sessions from the database. Among other methods this helper class contained methods responsible for storing data the database, inserting data of a new training session into the database and deleting items from the database

```
/**
 * Stores the given training session to database.
 * If id is set, the training session will be updated else it will be created
 *
 * @param item    the training session to store
 * @param context The application context
 * @return the saved training session (with correct id)
 */
public static Training save(Training item, Context context) {
    if (item == null) {
        return null;
    }
    if (item.getId() <= 0) {
        long insertedId = insert(item, context);
        if (insertedId == -1) {
            return null;
        } else {
            item.setId(insertedId);
            return item;
```

Figure 52

```
/**
 * Deletes the given training session from database
 *
 * @param item    the item to delete
 * @param context The application context
 * @return true if deletion was successful else false
 */
public static boolean delete(Training item, Context context) {
    if (item == null || item.getId() <= 0) {
        return false;
    }
    String selection = TrainingDbHelper.TrainingSessionEntry._ID + " = ?";
    String[] selectionArgs = {String.valueOf(item.getId())};
    return (0 != getDB(context).delete(TrainingDbHelper.TrainingSessionEntry

}

/**
 * Inserts the given training session as new entry.
 *
 * @param item    The training session which should be stored
 * @param context The application context
 * @return the inserted id
 */
protected static long insert(Training item, Context context) {
    ContentValues values = item.toContentValues();
    long insertedId = getDB(context).insert(
            TrainingDbHelper.TrainingSessionEntry.TABLE_NAME,
            nullColumnHack: null,
```

Figure 53

6.4.4 Models

The models used for the fitness aspect were simple java classes containing getters and setters. Example of classes was training model, step count model, activity summary model, walking mode model etc.

6.4.5 Adapters

Adapter classes in Java provide the default implementation of all methods in an event listener interface. Adapter classes are implemented in cases where you want to process only a few of the events that are handled by a particular event listener interface. All adapter classes extend RecyclerView.Adapter.
The implemented application made use of four key adapter classes :

1. Motivation Alerts Text Adapter - This adapter is used for motivation-alert texts.
2. Report Adapter - This adapter creates a Report View of a users training and invoked by the layout manager. This adapter also contains methods to generate the chart summary of a users training.

```
/**
 * This adapter is used for ReportView.
 *
 * @author Anthony Awobasivwe
 */
public class ReportAdapter extends RecyclerView.Adapter<ReportAdapter.ViewHolder> {
    private static final int TYPE_SUMMARY = 0;
    private static final int TYPE_DAY_CHART = 1;
    private static final int TYPE_CHART = 2;
    private List<Object> mItems;
    private OnItemClickListener mItemClickListener;

    /**
     * Creates a new Adapter for RecyclerView
     *
     * @param items The data displayed
     */
    public ReportAdapter(List<Object> items) { mItems = items; }

    // Create new views (invoked by the layout manager)
    @Override
    public ReportAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                                                       int viewType) {
        View v;
        ViewHolder vh;
        switch (viewType) {
            case TYPE_CHART:
                v = LayoutInflater.from(parent.getContext())
                        inflate(R.layout.card_activity_bar_chart, parent, attachToRoot: false);
```

Figure 54

```
            mChart = (LineChart) itemView.findViewById(R.id.chart);
            mChart.getXAxis().setPosition(XAxis.XAxisPosition.BOTTOM);
            mChart.getAxisRight().setEnabled(false);
            mChart.setTouchEnabled(false);
            mChart.setDoubleTapToZoomEnabled(false);
            mChart.setPinchZoom(false);
            mChart.setDescription("");
        }
    }

    public class CombinedChartViewHolder extends AbstractChartViewHolder {
        public CombinedChart mChart;

        public CombinedChartViewHolder(View itemView) {
            super(itemView);
            mChart = (CombinedChart) itemView.findViewById(R.id.chart);
            mChart.getXAxis().setPosition(XAxis.XAxisPosition.BOTTOM);
            mChart.getAxisRight().setEnabled(false);
            mChart.setTouchEnabled(false);
            mChart.setDoubleTapToZoomEnabled(false);
            mChart.setPinchZoom(false);
            mChart.setDescription("");
            mChart.setDrawOrder(new CombinedChart.DrawOrder[]{
                    CombinedChart.DrawOrder.BAR, CombinedChart.DrawOrder.BUBBLE, CombinedC
            });
        }
    }
```

Figure 55

3. Training overview Adapter - Adapter used for training overview
4. WalkingModesAdapter - Adapter used for walking modes

6.4.5 Fitness Services

The services extend IntentService and implement SensorEventListener, SharedPreferences.OnSharedPreferenceChangeListener .
The fitness aspect of the application utilised three service classes :

1. Abstract Step Detector Service - Generic class for a step detector.
   which does not detect steps itself. The step detection has to be done in the subclasses below this

2. Accelerometer Step Detection Service - Uses the accelerometer to detect steps.
3. HardwareStepDetectorService - Uses the hardware step counter sensor to detect steps.

6.4.6 Activities

1. Training Activity - this activity is responsible for the training aspect of the application. Features implemented in this activity include getting the step count of the user, updating the steps of a user. This activity also allows the user to manage the training phases. The onCreate method in this activity is called to open persistent service if no current training session is active.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_training);
    // get current training instance
    training = TrainingPersistenceHelper.getActiveItem( context: this);
    if (training == null) {
        // if no training is active, start persistence service
        StepDetectionServiceHelper.startPersistenceService( context: this);
        // Now wait for steps saved broadcast message and than create a new training session.
        // We have to wait to ensure, that only the steps since now are counted.
    }
    StepDetectionServiceHelper.startAllIfEnabled( context: this);

    mTextViewSteps = (TextView) findViewById(R.id.training_steps);
    mTextViewDistance = (TextView) findViewById(R.id.training_distance);
    mTextViewDistanceTitle = (TextView) findViewById(R.id.training_distance_title);
    mTextViewCalories = (TextView) findViewById(R.id.training_calories);
    mTextViewDuration = (TextView) findViewById(R.id.training_duration);
    mTextViewVelocity = (TextView) findViewById(R.id.training_velocity);
    mTextViewVelocityTitle = (TextView) findViewById(R.id.training_velocity_title);
    Button buttonStop = (Button) findViewById(R.id.training_stop_button);
    if (buttonStop != null) {
        buttonStop.setOnClickListener(this);
    }

    IntentFilter filterRefreshUpdate = new IntentFilter();
    filterRefreshUpdate.addAction(StepCountPersistenceHelper.BROADCAST_ACTION_STEPS_SAVED);
    filterRefreshUpdate.addAction(AbstractStepDetectorService.BROADCAST_ACTION_STEPS_DETECTED)
    LocalBroadcastManager.getInstance(this).registerReceiver(broadcastReceiver, filterRefreshU
    // Bind to stepDetector
    Intent serviceIntent = new Intent( packageContext: this, Factory.getStepDetectorServiceClass(
    getApplicationContext().bindService(serviceIntent, mServiceConnection, Context.BIND_AUTO_C
    this.getStepCounts();
```

Figure 55

2. Training Overview Activity

This java class provides an overview of a users training session. It extends AppCompatActivity and implements Training Overview Adapter.OnItemClickListener. The activity allows users to manage the training phases. It contains methods such as showing a training session of a user.

```
protected void showTrainings() {
    // Load training sessions
    List<Training> trainingsLoadFromDatabase = TrainingPersistenceHelper.getAllItems( con
    trainings = new ArrayList<>();
    int steps = 0;
    double distance = 0;
    double duration = 0;
    double calories = 0;

    // Add month labels
    Calendar cal = Calendar.getInstance();
    int month = -1;
    for(int i = 0; i < trainingsLoadFromDatabase.size(); i++){
        Training training = trainingsLoadFromDatabase.get(i);
        cal.setTimeInMillis(training.getStart());
        if(month != cal.get(Calendar.MONTH)){
            month = cal.get(Calendar.MONTH);
            DateFormat df = new SimpleDateFormat( pattern: "MMMM yyyy", getResources().getC
            // create dummy training entry to display the new month
            Training monthHeadline = new Training();
            monthHeadline.setName(df.format(cal.getTime()));
            monthHeadline.setViewType(TrainingOverviewAdapter.VIEW_TYPE_MONTH_HEADLINE);
            trainings.add(monthHeadline);
        }
        steps += training.getSteps();
        distance += training.getDistance();
        duration += training.getDuration();
        calories += training.getCalories();
```

Figure 56

3. Motivational Texts Activity - This activity allows the user to manage the motivation texts. The class extends AppCompatActivity and implements Motivation Alert Texts Adapter.OnItemClickListener. In the protected void show Motivation Alert Texts() of the class, the motivation texts are loaded from shared preferences. If no preference exists then the default texts will be given in the resources. If the motivation texts are empty it is set to empty view.

```
protected void showMotivationAlertTexts() {
    SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences( context: this);
    Set<String> defaultStringSet = new HashSet<>(Arrays.asList(getResources().getStringArray(R.array.pr
    Set<String> stringSet = sharedPref.getStringSet("com.android.FYP.pref.motivation_alert_texts", defa
    motivationTexts = new ArrayList<>(Arrays.asList(stringSet.toArray(new String[stringSet.size()])));

    this.mAdapter.setItems(motivationTexts);
    if (motivationTexts.size() == 0) {
        mEmptyView.setVisibility(View.VISIBLE);
    } else {
        mEmptyView.setVisibility(View.GONE);
    }
}

/**
 * Stores the motivation texts to shared preferences
 */
protected void save() {
    SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences( context: this);
    SharedPreferences.Editor editor = sharedPref.edit();
    editor.putStringSet("com.android.FYP.pref.motivation_alert_texts", new HashSet<>(motivationTexts));
    editor.apply();
}
```

Figure 57

6.4.7 Fitness Fragments

1. Main Fragment - This was used to display the main application view in terms of daily, monthly and weekly reports.
2. Monthly Report Fragment - This was used to create report fragments of a specific month. Activities In this class implemented the Monthly Report Fragment.OnFragmentInteractionListener} interface
3. Weekly Report Fragment - This was used to create report fragments of a specific week. Activities In this class implemented the Weekly Report Fragment.OnFragmentInteractionListener} interface

4. Daily Report Fragment - This was used to create report fragments of a specific day. Activities In this class implemented the Daily Report Fragment.OnFragmentInteractionListener} interface

Problems faced during Implementation

Implementation of an application is always fun but tricky topic. Implementing the application required me to perform additional research on concepts such as firebase which were not taught in detail at the university. I found this difficult as I had to adjust the timing on the Gantt chart to consider this.

Section 7 - System Testing

System testing is an important part of the success of any project. In order to meet the requirements of the system, it is important to test all functionalities are working

7.1 Requirements testing

This section looks into the testing of all requirements listed from section 3.2.1 in order to ensure the product handed out to future customers meets the functional requirements level of the project. As the application

Test 1

Evaluation

Users will be able to sign up by providing their email and choosing and confirming a password
The provided details will be stored in the firebase

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Screenshot/Proof

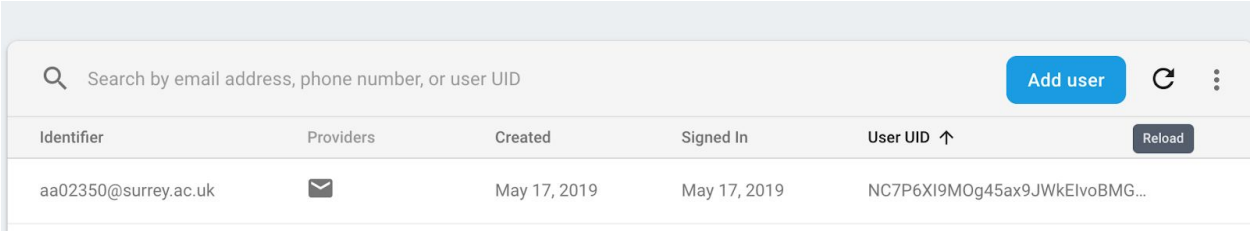


Figure 58

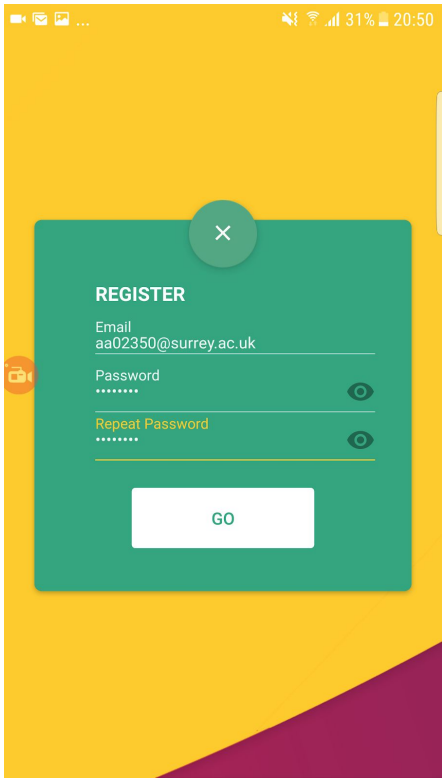


Figure 59

Test Passed = Yes

Test 2

Evaluation
If email is already taken the system will let the user know

Screenshots/Proof

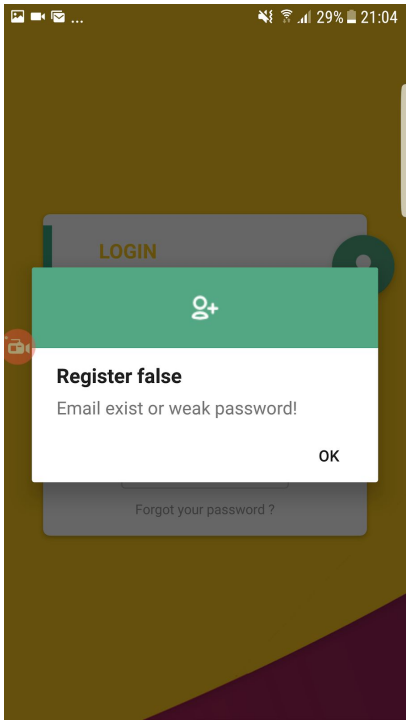


Figure 60

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test 3

Evaluation
When a user enters a password for registration the password is entered twice and must be matching in order to complete registration

Screenshots/Proof

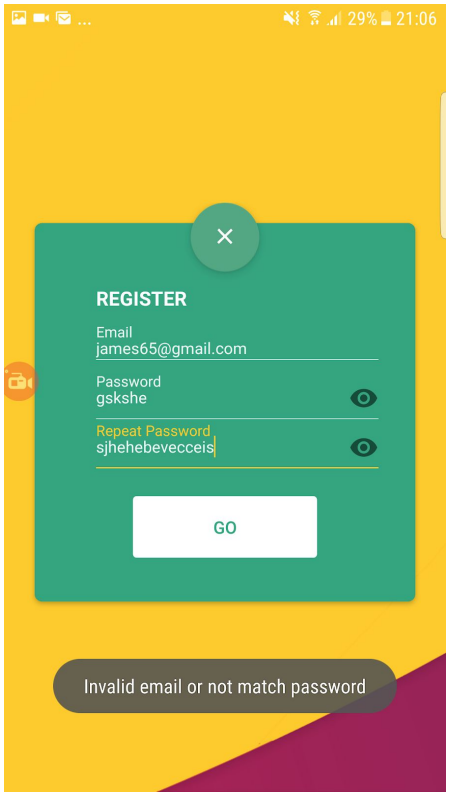


Figure 61

Outcome

This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed = Yes

Test 4

Evaluation

When logging in password must match original or else user is denied entry

Screenshots/Proof

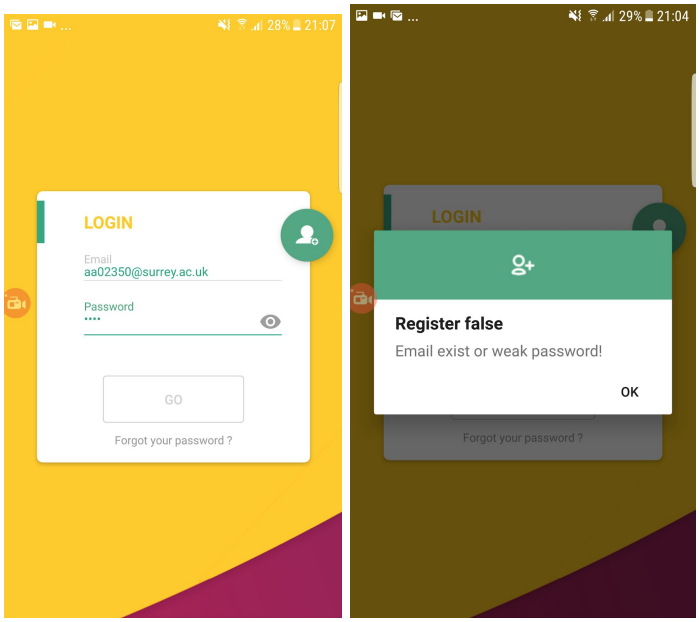


Figure 62

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed = Yes

Test 5

Evaluation
Already logged in users can sign out of the account and log back in at their choosing

Screenshots/Proof

This test cannot be proved via screenshot. However, the application was tested and users could log back in based on the test performed.

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

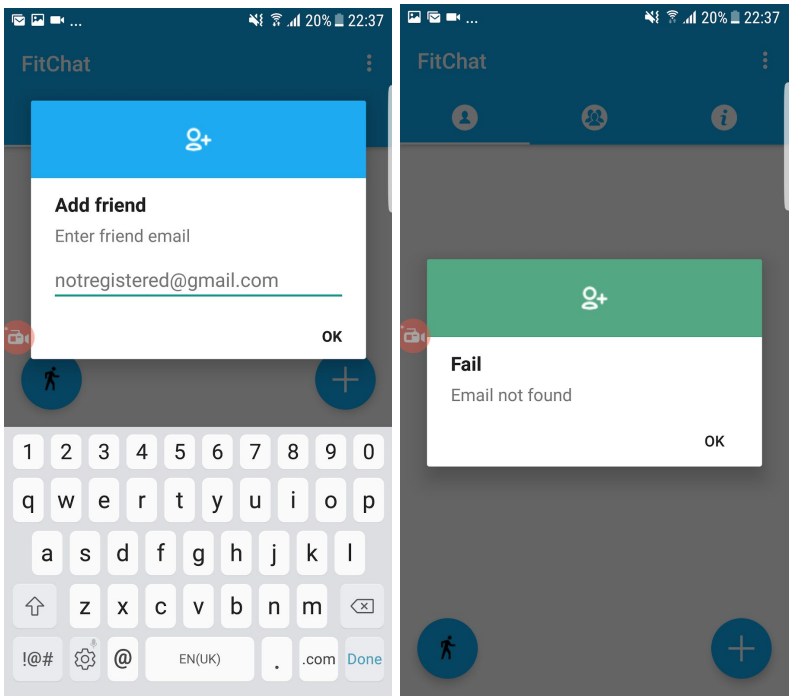
Test Passed = Yes

Test 6

Evaluation
Users can only add users as friends if the friend also has an accounts

Screenshots/Proof

The user adds a friend not registered on the application



The user adds a friend who is also registered on the application

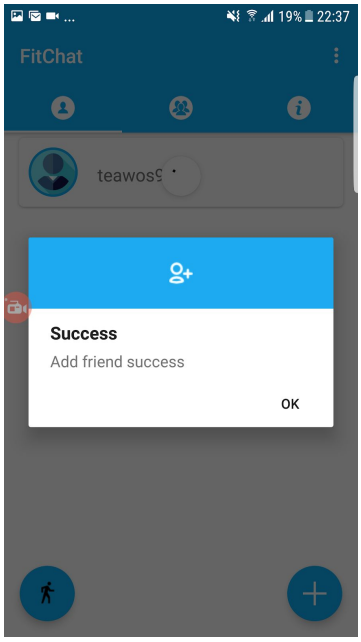


Figure 63

Outcome

This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed =yes

Anthony Awobasivwe 6442385

Test 7

Evaluation
The system will allow sending and receiving of messages between users

Screenshots/Proof

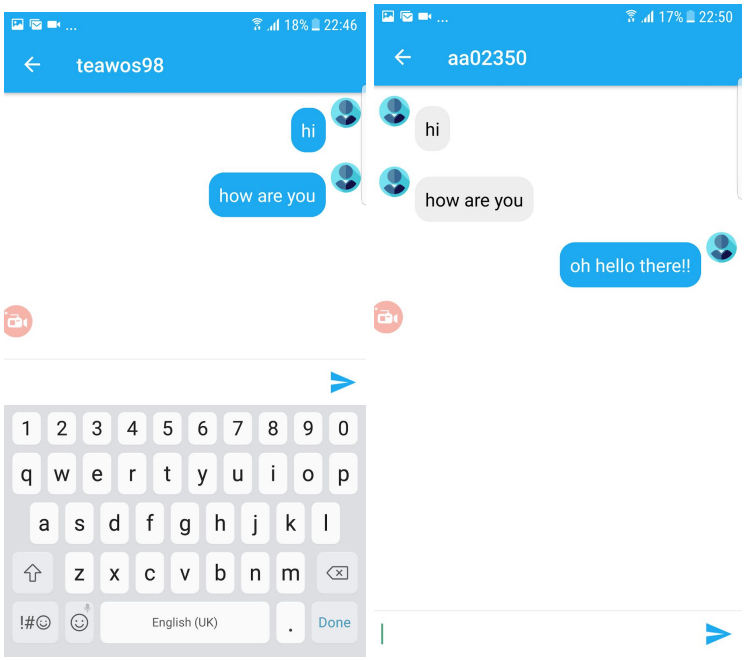


Figure 64

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed = Yes

Test 8

Evaluation
At Least two other users in total are required to form a group

Screenshots/Proof

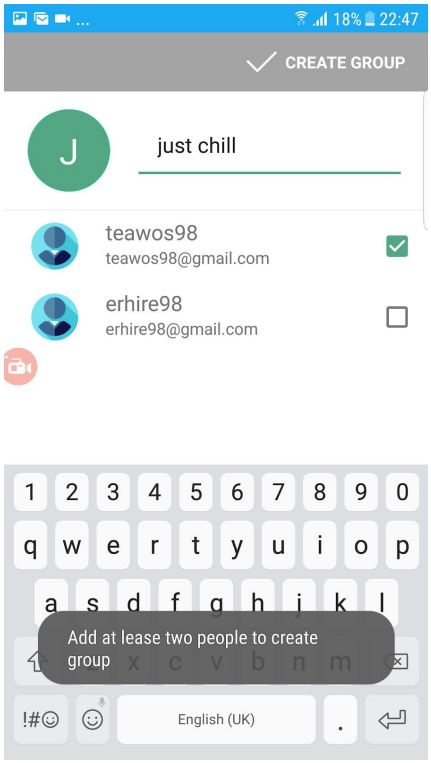


Figure 65

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed = Yes

Test 9

Evaluation

Users can add other users to a group and communicate via the group

Screenshots/Proof

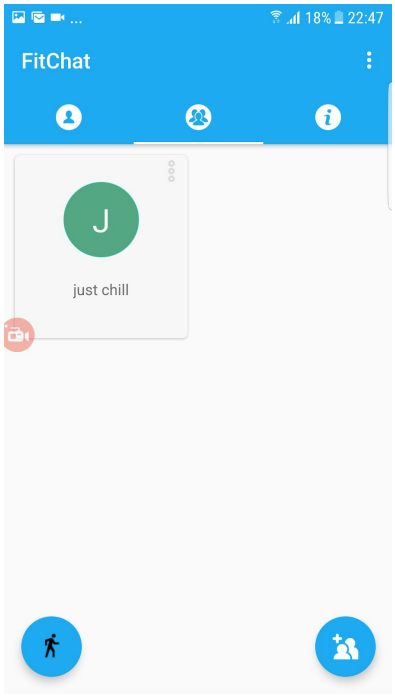


Figure 66

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test 10

Only group admin can edit or delete members from a group

Screenshots/Proof

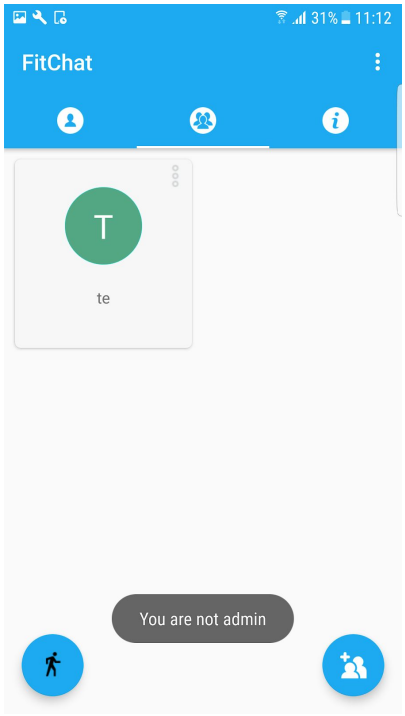


Figure 67

Outcome

This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed = Yes

Test 11

Evaluation

The Group admin can edit the group to change the group name / add or delete members from the group

Screenshots/Proof

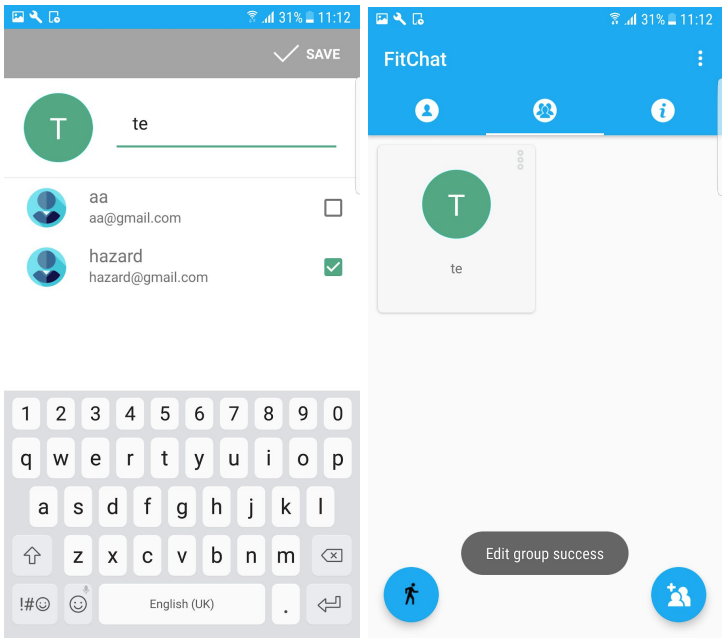


Figure 68

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed = Yes

Test 12

Evaluation
Users can view their personal profile

Screenshots/Proof

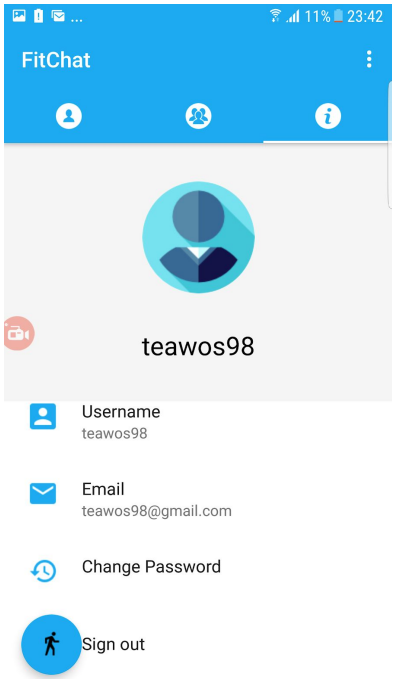


Figure 69

Outcome

This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed = Yes

Test 13

Evaluation

Users can edit their profile by adding a profile picture, change username

Screenshots/Proof

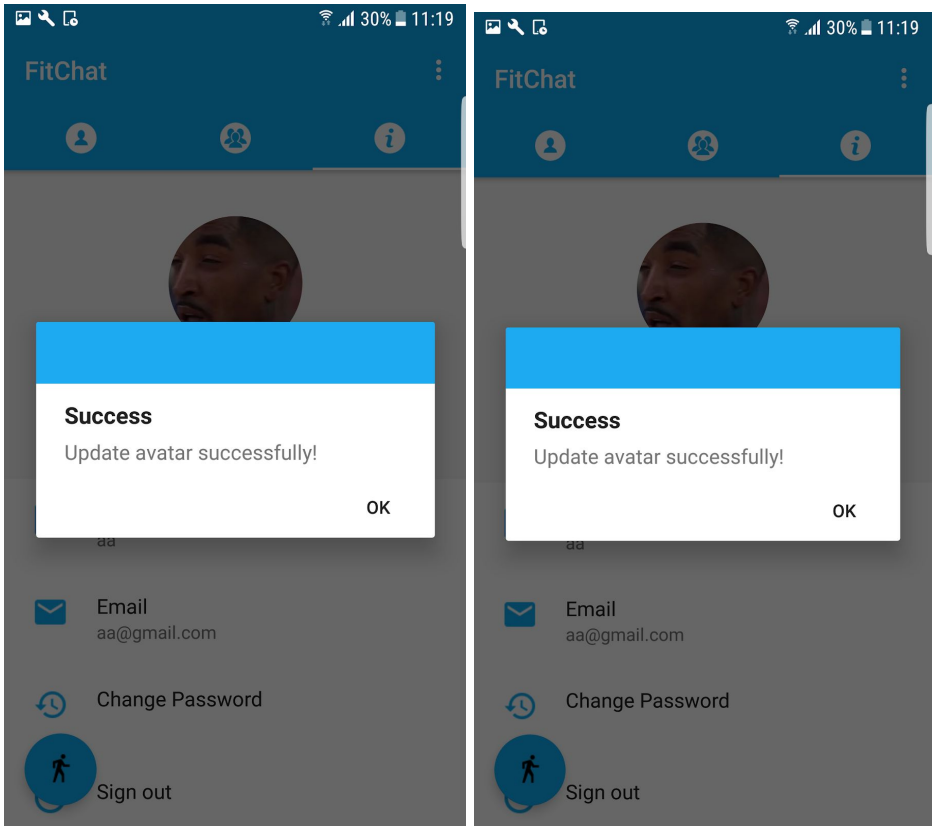


Figure 70

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 14

Evaluation
Users can reset their password by choosing the option and the user will have a link sent to their actual email to continue the password reset

Screenshots/Proof

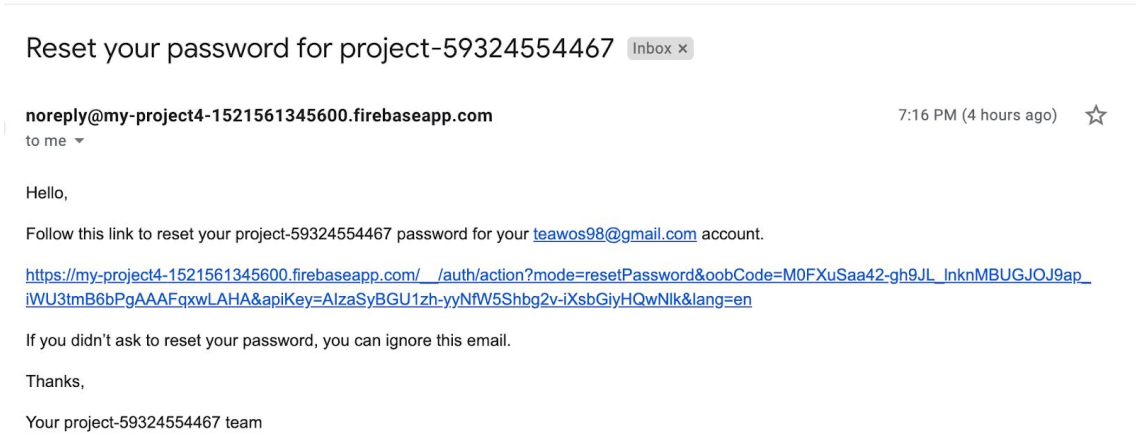


Figure 71

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation

Test Passed = Yes

Test 15

Evaluation

Users can request a new training session and will receive the time, distance in km, calories burned and the number of steps taken and a star rating.

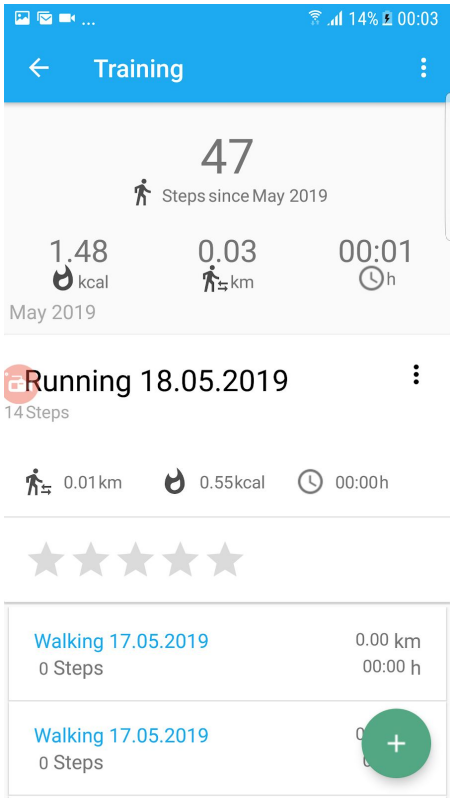


Figure 72

Screenshots/Proof

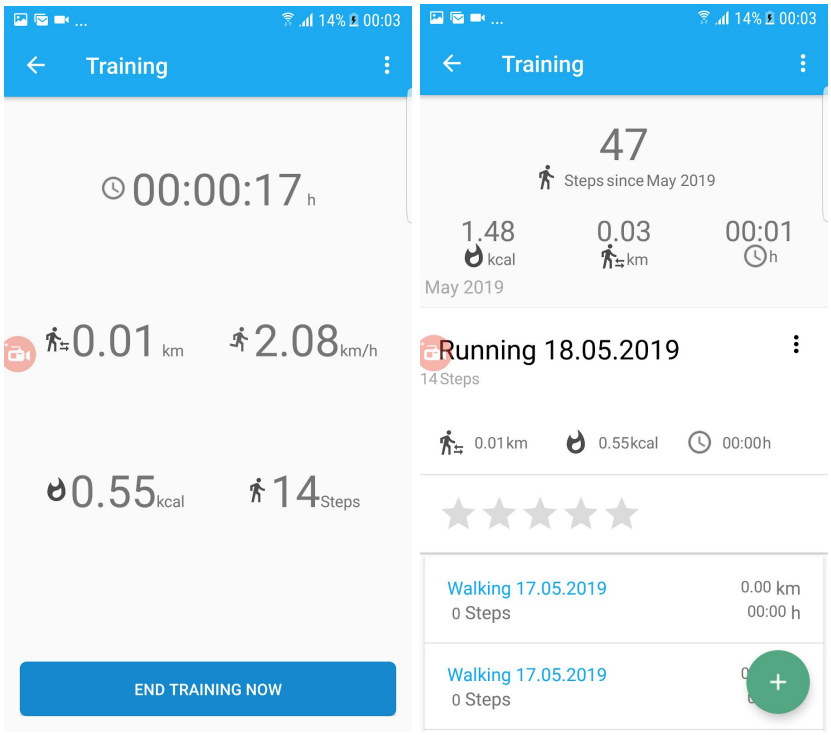


Figure 73

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 16

Evaluation
Users can adjust general settings for the application such as choosing their weight, step goal, gender etc.

Screenshots/Proof

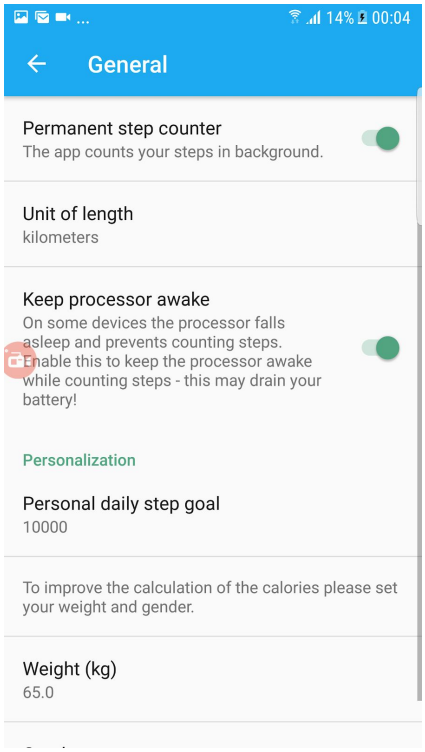


Figure 74

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 17

Evaluation
The system should provide a summary of a users training session based on day, week or months

Screenshots/Proof

Anthony Awobasivwe 6442385

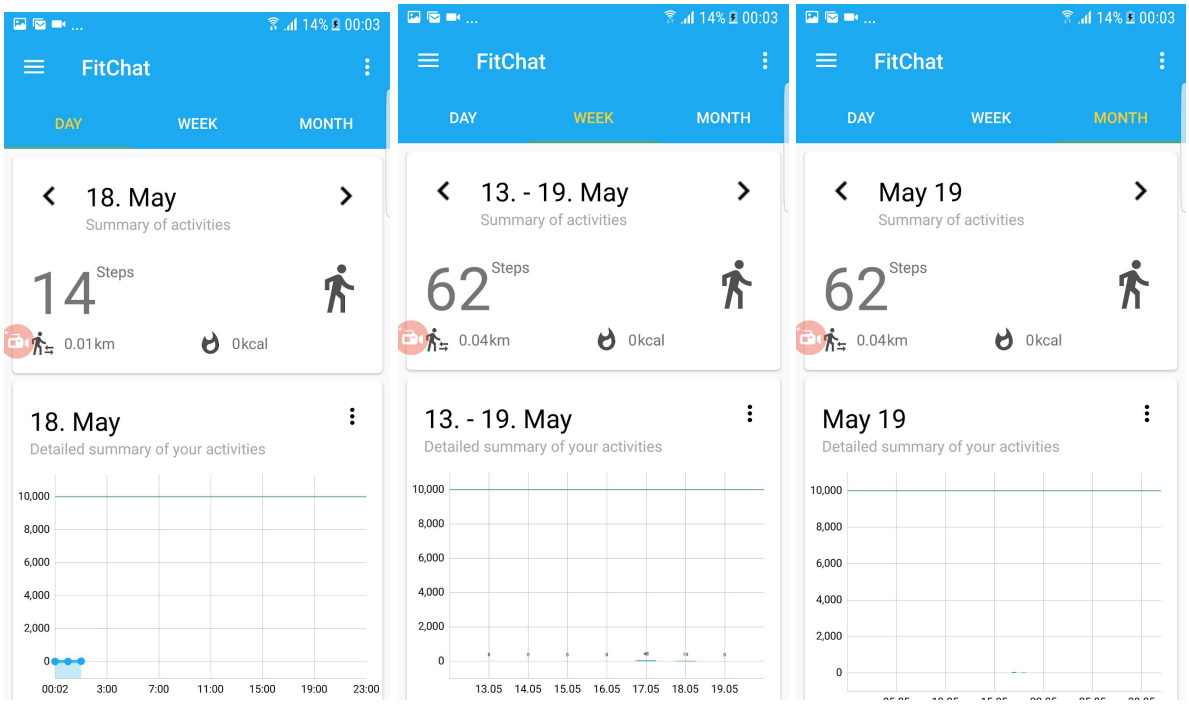


Figure 75

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes
Test 18

Evaluation
The system can provide notifications on what time to do a fitness check, time to train, distance walked today etc.

Screenshots/Proof

Anthony Awobasivwe 6442385

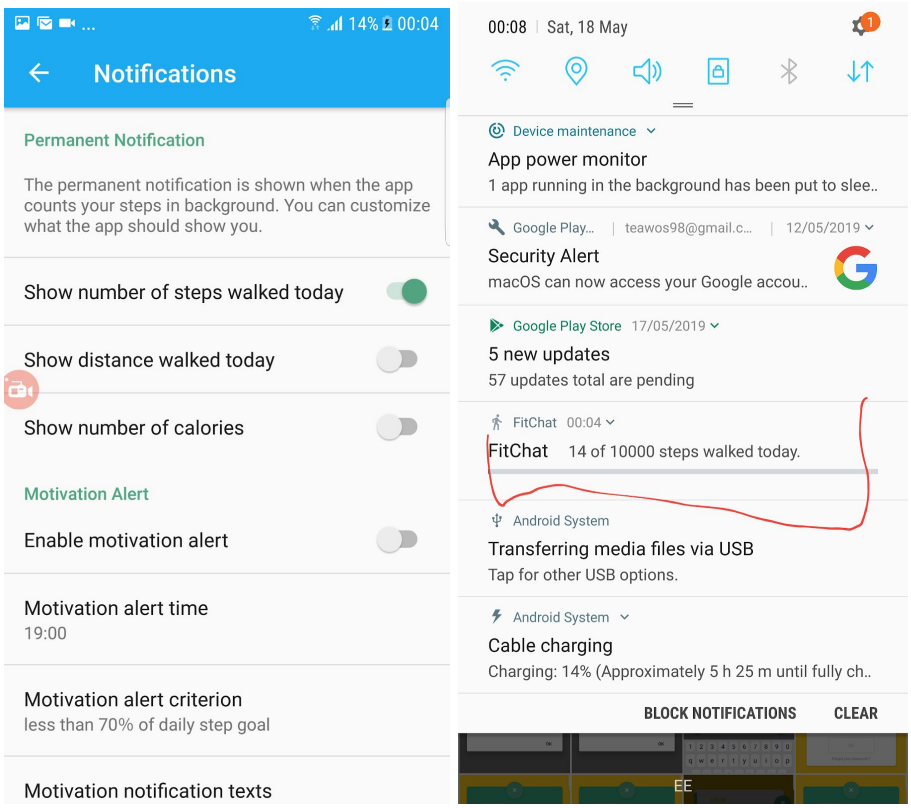


Figure 76

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 19

Evaluation
Users have the option to go to the help page if any functions in the fitness section are not clear

Screenshots/Proof

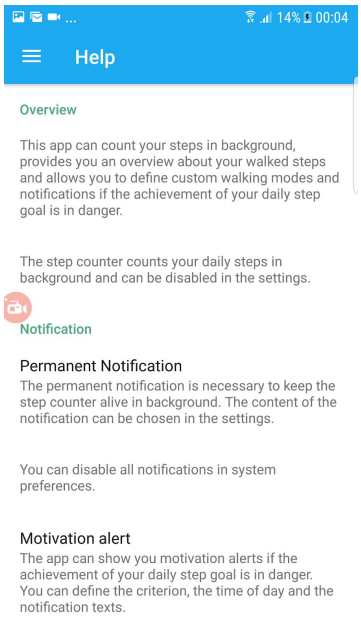


Figure 77

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 20
Evaluation

Walking mode feature to allow users to select a walking mode of their choosing

This test failed, however, the code imputed was left for the section in the case it could be fixed before the project deadline. This feature was of low relevance and not a core feature of the application so it shouldn't have a big impact on the success of the project
Test Passed = No

7.1.1 Non Functional Requirement Testing

Test 1

Evaluation
The app should run within the first five seconds of execution
Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 2

Evaluation
The app should provide accurate results when calculating steps taken, calories burned etc.
Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 3
Evaluation
The app should be available to users if they are connected to wifi

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 4
Evaluation
The app should be compatible with all android phones higher SDK version than 21
Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 5
Evaluation
The app should provide a good level of appeal to users while also being easy to navigate around

Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

Test 6
Evaluation
Users messages should remain confidential between the two parties sharing those messages
Outcome
This requirement was tested and it was concluded that the outcome was consistent with the evaluation
Test Passed = Yes

7.1.2 J unit Testing

Junit testing was used in the android application and placed in the build. gradle file. The resulting test gotten when the project was built showed there were no coding errors in the application

```
implementation 'com.google.firebase:firebase-auth:9.8.0'
testImplementation 'junit:junit:4.12'
implementation 'de.hdodenhof:circleimageview:2.1.0'
implementation 'com.yarolegovich:lovely-dialog:1.0.4'
```

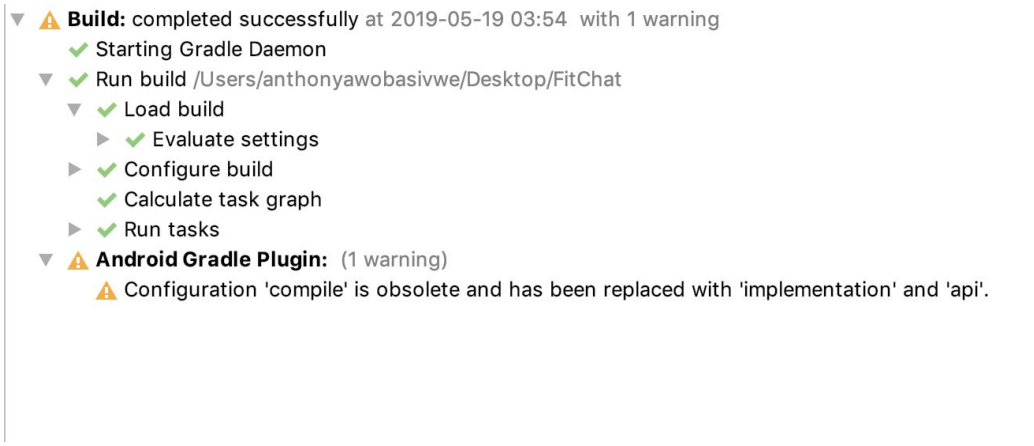


Figure 78

7.2 User Acceptance Testing

User acceptance testing (UAT) is the final phase in software development testing. This phase involves making the application available to real users who test the software to determine its feasibility of success in a real-life scenario and if the application can handle or adapt to real-world scenarios according to specifications.

User acceptance was prepared with the primary goal being to test how the system functioned in a real-life situation. Users were given the link to download the application on their various devices and log in with their student email and a password of their choosing. The users then tested out different functionalities within the system and replied to various questions asked in the User Acceptance Questionnaire which can be found in the appendix section of the report. The feedback gotten from users is useful as it creates a good structure for identifying areas of improvement for future work (section 8.3) concerning this project and any future projects.

7.2.1 User Acceptance

Feedback gotten from users was in all positive. Users found the integration of a fitness application alongside a chatting application quite intriguing and found it genuine based on other applications they've used. Users also found the brightness in the user interface to be suitable although a few users found it not suitable and preferred a darker team or the option to choose their own colour layout for the application given that feature a mixed reaction.

Going on to the registration and login aspect users found this to be ok and working properly however would have liked another extra level of authentication when signing up to the application. The sending and receiving messages features according to users worked fine along with editing their Regarding the changing of password feature, some users were concerned about the amount of time it takes to authenticate via a confirmation to the user's email he/she signed up with, however, some users felt fine with this and understood it was just another layer of authentication.
A handful of users stated that it was difficult to access the drop-down list to edit a group. This raises concerns about the accessibility of the application covered in the risk analysis.
Regarding the fitness aspect of the application, users found the training functionality summary of given a star out of 5 based on the strength of a training session to be appealing and possibly could lead to motivating users to strive while using the training feature in order to obtain higher stars. Users also found the displayed summary listed in days, months or years as an important feature when monitoring goals set by a user.

7.2.2 User Acceptance Test Questions

Question 1
How did you find the application?
The majority of users who took part in this gave positive feedback on the application. A high percentage of users said they found the application to be simple however contain all the features required for a successful application
Question 2
Would you consider using this application on a daily basis?
The answers gotten from this section were also mixed. Most users said they would rather have two separate applications while others said they could consider using it.

Question 3
How smooth did the application feel when running?
Most Users said the application was smooth enough when ran with little to no lag involved.

Question 4
Did all application features work to your satisfaction?
Users noted that all features of the application were working except for the walking modes activity which has been identified to be non-working earlier on in the testing section.

7.3 Conclusion

In conclusion, This section gave a good understanding of how the system built functioned in a real-world scenario. The testing of requirements was overall successful. Some low-level features in the application were not completed on time such as the walking mode feature due to the time constraints and focusing on completing the higher level of functionalities first. The argument could be made that these tasks should have been properly accounted for in the Gantt chart, this, as well as other room for improvements regarding the project, will be discussed in more depth in the evaluation section.

Section 8 - Project Evaluation and Recommendations

8.1 Introduction

This section of the projects aims to provide an evaluation of the success criteria regarding the project and the overall success of the project. The academic input I gained during the span of the project will also be highlighted as well as that this section of the report will also look into recommending future improvements to the project by highlighting any problems that could be fixed concerning this project such as improvements to the scope or developmental aspects of the project.

8.2 Evaluation against project objectives

This section aims to identify if the objectives identified at the start of the project were achieved. These objectives were set in the introduction section of the report and range from my own improvements in coding to create a feasible application.

Objective/Success Criteria	Description	Was it Achieved	Comments/ from objective
Objective 1	To create an application suitable for users in a real-world scenario	Yes	This objective was completed however I would consider it to be the most difficult as I had to learn a lot of new features in android studio in order

			to complete it
Objective 2	To conduct research into firebase in order to apply it to my application	Yes	I enjoyed the research aspect of this as I was pushed out of my comfort zone in just programming and had to learn and read up on other topics such as firebase and how its implemented
Objective 3	To conduct research into how designing a system can help user approval	Yes	Similarly to objective 2 researching, this gave me a clear idea of the dos and don'ts in designing a UI in order to maximise customer output
Objective 4	Integrating fitness features into the chat application presentable to users	Yes	This objective was completed as the fitness aspect of the application was properly integrated with the chatting system.
Objective 5	To learn new features in android studio such as firebase authentication, learning more complex forms of services, receivers and fragments and integrating these features into the coding aspect in android studio	Yes	This objective was possibly the most important as the things I learnt from the project will forever stay with me
Objective 6	Setting out a project management plan in order to have all application deliverables ready on schedule	Yes	This objective was completed by using my skills in Microsoft project to come up with a project plan to

			complete the project at a good time.
Success criteria 1	Providing a working application suitable for use by the target population	Yes	This success criterion was met however the application is still in a prototype stage and might not be suitable in a real world scenario due to security concerns surrounding the application highlighted in the risk assessment
Success criteria 2	Improving my skills in Android development to implement features into my application thus strengthening my coding while giving me more insight into android development.	Yes	This success criterion was met as my coding skills have now been improved
Success criteria 3	The user acceptance testing feedback gotten from users	Yes	This success criterion was met as feedback gotten from the user acceptance test came back as positive in its entirety.
Success criteria 4	The testing of functional requirements being met	Yes	This success criterion was completed as the testing phase of requirements showed that all core features of the application were working
Success criteria 5	Finishing the project on time and within the scope and quality level of coding practice I've been taught at the University of Surrey	Yes	This success criterion was completed successfully

8.3 Future Work

The current version of the system can be seen as just an In the case of advances
To the project in the future the following advancements will be considered for implementation to the application:

1. Making the application exclusive to the University of Surrey: Currently, the system is available to all email types. Making the application exclusive to the University of Surrey would require permissions from the university to access student usernames details similar to the University of Surrey email. This would be a good move however I feel the system is currently not secure enough to go through with that.
2. Adding more settings to enable users to personalise their home screens
3. Video/voice calling - video/voice call is used by many applications as a means of communication. Implementing this feature will increase the variety of features available to users however this would also require extra permissions as users personal phone numbers are being stored.
4. Sending files/videos through messages - this is another feature which would increase the variety of features accessible to users in the application. If this were to be implemented there would be a maximum data size for files being sent across by users. This feature would also require revisiting the risk section of the report as hackers may become attracted to files sent along the channels.

8.4 Areas of Improvement

While this project was a successful one in its entirety there are still some parts which could be improved. This section will identify those aspects of the project which could be improved after consulting feedback from all the previous phases of the project such as feedback from the survey, requirement testing and user acceptance testing.

1. Authentication could be improved - The registration part of the application currently doesn't have two-factor authentication in place. This part of the application would be improved if the application was in a real-life scenario for security purposes
2. User Interface - The current user interface as suggested by users could be improved by adding a personalisation effect allowing users an option to edit the colour and theme of the application.
3. Security - The security of the application is currently defined as public on firebase meaning anyone can steal, modify, or delete data in your database. This can easily be resolved by changing the settings on firebase however I don't see it as required as the app currently is just a demo.

8.5 Academic Input and Personal Experience Gained

To gain some level of experience from the project was one of the main objectives stated in the introduction section. I also gained experience learning how to use MS project to create and manage a Gantt chart. This was helpful as my project allowed me to plan properly the timeline of features in the application.

Information Security Management - Applying my skills learnt in information security management was pivotal in completing the risk analysis section. Using these skills I learnt I was able to understand what was required for my application to remain confidential, of good integrity and availability and hence uphold the CIA triad. I also learnt how to identify risks and grade them based on its severity and likelihood and provide a method for control for the risk identified.

Firebase and cloud storage - The project thought me a lot about how firebase is I also got a chance to practice using a cloud firestore instead of the real-time database the application is currently using.

Time Management - Time management is something often overlooked by people. At the start of the project, I fell victim to this and initially overlooked having a clear time structure. However later on in the project when activities started to pile up I had to adapt and learn how to manage my time appropriately. This The project also helped me improve my time management skills as I was able to manage the bulk load of work and complete the project within the initial requirements identified in the introduction section. This skill could also be useful outside university and in any future projects, I partake in.

8.6 Conclusion

In conclusion, the project was developed at the level. Using android studio again was a good experience for me. Although the application was successful there is still room for improving the project before it can be utilized in a real-world situation. There are not many fitness chat applications out there in the market currently however I strongly feel this will become more of a thing in the future based on how much users liked my application.

Appendix

Anthony Awobasivwe 6442385

AP.1 Survey Questions

1. Would you consider this app suitable for university students?
    - ❑ Yes
    - ❑ No
2. Could you please in a few words explain your reason behind your answer for question 1

3. What features would you like to be added to the application?

4. Have you ever used any applications similar to this one?
    - ❑ Yes
    - ❑ No

5. If yes how do they differ from this application

6. Do you think this application could be financially successful in a real-world market
    - ❑ Yes
    - ❑ No

7. Could you explain the reasoning behind your answer for question 6
8. What Current features do you like in the application

AP.2 References

Uersnap.com, 2019. USER ACCEPTANCE TESTING – HOW TO DO IT RIGHT!
[Online] Available at: https://usersnap.com/blog/user-acceptance-testing-right/ [Accessed 03/04/2019].

surveygizmo.com, 2019. Defining the purpose of a survey
[Online] Available at https://www.surveygizmo.com/resources/blog/purpose-of-survey/
[Accessed 22/03/2019].

businessanalystfaq.com, 2019. Importance of Use Case diagram for Business Analyst
[Online] Available at
https://businessanalystfaq.com/2017/07/30/importance-of-use-case-diagram-for-business-analyst/
[Accessed 24/03/2019].

Macworld.co.uk , Android vs iphone market .share [Online] Available at:
https://www.macworld.co.uk/feature/iphone/iphone-vs-android-market-share-3691861/
[Accessed 04/03/2019]

Hackernoon.com , introduction to firebase [Online] Available at:
https://hackernoon.com/introduction-to-firebase-218a23186cd7
[Accessed 20/02/2019].

Qualitrics.com, building effective surveys [Online] Available at
https://www.qualtrics.com/blog/10-tips-for-building-effective-surveys/
[Accessed 03/20/2019].

UXplanet, Bright colors in ui design [Online] Available at:
https://uxplanet.org/bright-colors-in-ui-design-benefits-and-drawbacks-433680f0a1c7
[Accessed 05/04/2019].

Android, Guide to app architecture [Online] Available at
https://developer.android.com/jetpack/docs/guide
[Accessed 15/04/2019].

Agriya.com, Growth of messaging apps [Online] Available at
https://www.agriya.com/blog/growth-potential-of-instant-messaging-app-market-why-build-chat-app/
[Accessed 02/04/2019]

NCBI, Prevalence of Overweight/Obesity and It's Associated Factors among University Students from 22 Countries [Online] Available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4113885/
[Accessed 22/03/2019]