M Title Page

Abstract

Technology and exercise are two topics of which in the past could not sound more further apart. However, as technology evolves, more innovative and creative forms of use are being explored to provide a new world of possibilities. Could the use of mobile phone technology be successfully utilised to create a platform that promotes social interaction and exercise? This report outlines the steps taken through the development of an android application with the intention to improve the method in which people meet and plan for sports-based events. Starting with thorough research into related topics ranging from efficient and effective graphical user interfaces to the statistics of technology use today. What was discovered is transformed into a tactical and inventive solution. Ideas are analysed and debated until an appropriate system is decided and designed. The implementation is then developed and tested to assign value to the success. MatchUp allows for people of any ability to join a community of like minded peers to support each other in developing their skills. This is possible due to several user-friendly features backed up by research including an instantaneous live chat, revised social matching method and easy event creation.

Acknowledgements

I would like to thank Dr. Frederic Stahl for his continued academic guidance as well as moral support throughout the duration of this project.

**Table of Contents**

[Glossary of Terms and Abbreviations 6](#_30j0zll)

[1.](#_1fob9te) Introduction 7

[2.](#_3znysh7) Problem Articulation 8

[2.1.](#_2et92p0) Problem Statement 8

[2.2.](#_tyjcwt) Technical Specification 9

[2.3.](#_3dy6vkm) Stakeholders 9

[2.4.](#_1t3h5sf) Project Motivation 10

[2.5.](#_4d34og8) Constraints 11

[2.5.1.](#_2s8eyo1) Project constraints 11

[2.5.2.](#_17dp8vu) Technical constraints 11

[2.6.](#_3rdcrjn) User Acceptance Criteria 11

[3.](#_26in1rg) Literature Review 13

[3.1.](#_lnxbz9) Operating system 13

[3.2.](#_1ksv4uv) Programming language 13

[3.3.](#_44sinio) Integrated development environment 14

[3.4.](#_2jxsxqh) Firebase 15

[3.4.1.](#_z337ya) Real-time database 15

[3.4.2.](#_3j2qqm3) Authentication 16

[3.5.](#_1y810tw) Existing technologies 16

[3.5.1.](#_4i7ojhp) Eventbrite 16

[3.5.2.](#_2xcytpi) Snafflz 17

[3.5.3.](#_1ci93xb) Sporty 17

[3.5.4.](#_3whwml4) Tinder 18

[3.5.5.](#_2bn6wsx) Teamer 18

[3.6.](#_qsh70q) Human Computer Interaction 19

[4.](#_3as4poj) The Solution Approach 21

[4.1.](#_1pxezwc) Potential Solutions 21

[4.2.](#_49x2ik5) Solution Discussion 22

[4.3.](#_2p2csry) Solution Definition 24

[4.3.1.](#_147n2zr) Chosen solution 24

[4.3.2.](#_3o7alnk) Justification 24

[4.3.3.](#_23ckvvd) Definition 25

[5.](#_ihv636) Design 26

[5.1.](#_32hioqz) User interface design 26

[5.1.1.](#_1hmsyys) Start page 26

[5.1.2.](#_41mghml) Login page 27

[5.1.3.](#_2grqrue) Registration page 27

[5.1.4.](#_vx1227) Main page 28

[5.1.5.](#_3fwokq0) Settings page 29

[5.1.6.](#_1v1yuxt) Matches page 30

[5.1.7.](#_4f1mdlm) Live chat page 31

[5.1.8.](#_2u6wntf) Calendar page 31

[5.2.](#_19c6y18) Database design 32

[6.](#_3tbugp1) Implementation 33

[6.1.](#_28h4qwu) Swipe cards 33

[6.2.](#_nmf14n) Firebase 34

[6.3.](#_37m2jsg) Login and registration 34

[6.4.](#_1mrcu09) User matching 36

[6.5.](#_46r0co2) Live chat 39

[6.6.](#_2lwamvv) Event creation/management 40

[6.7.](#_111kx3o) User profile settings 41

[7.](#_3l18frh) Testing: Verification & Validation 43

[7.1.](#_206ipza) Verification 43

[7.2.](#_4k668n3) Validation 43

[7.3.](#_2zbgiuw) Functionality testing 43

[7.4.](#_1egqt2p) Usability testing 46

[7.5.](#_3ygebqi) Testing limitations 48

[8.](#_2dlolyb) Discussion 49

[8.1.](#_sqyw64) Results of functionality testing 49

[8.2.](#_3cqmetx) Results of usability testing 50

[8.3.](#_1rvwp1q) Hindsight 50

[9.](#_4bvk7pj) Reflection 52

[10.](#_2r0uhxc) Social, legal, health & safety and ethical implications 54

[11.](#_1664s55) Future Improvements 55

[12.](#_3q5sasy) Conclusion 56

[13.](#_25b2l0r) References 57

[14.](#_kgcv8k) Appendices 60

[Appendix 1 – Project Initiation Document 60](#_34g0dwd)

[Appendix 2 – Application Java Code (functionality) 76](#_1jlao46)

[2.1. LoginOrRegistrationActivity 76](#_43ky6rz)

[2.2. LoginActivity 77](#_2iq8gzs)

[2.3. Registration Activity 78](#_xvir7l)

[2.4. MainActivity 81](#_3hv69ve)

[2.5. SettingsActivity 85](#_1x0gk37)

[2.6. Cards 89](#_4h042r0)

[2.7. ArrayAdapter 90](#_2w5ecyt)

[2.8. Match Package 91](#_1baon6m)

[2.9. Messaging Package 96](#_3vac5uf)

[Appendix 3 – Application XML Code (design) 104](#_2afmg28)

[3.1. Activity\_calendar.xml 104](#_pkwqa1)

[3.2. Activity\_chat.xml 104](#_39kk8xu)

[3.3. Activity\_login.xml 105](#_1opuj5n)

[3.4. Activity\_login\_or\_register.xml 106](#_48pi1tg)

[3.5. Activity\_main.xml 106](#_2nusc19)

[3.6. Activity\_matches.xml 107](#_1302m92)

[3.7. Activity\_registration.xml 108](#_3mzq4wv)

[3.8. Activity\_settings.xml 110](#_2250f4o)

[3.9. Item.xml 112](#_haapch)

[3.10. Item\_chat.xml 113](#_319y80a)

[3.11. Item\_match.xml 113](#_1gf8i83)

[Appendix 4 – failed registration 115](#_40ew0vw)

# Glossary of Terms and Abbreviations

PID – Project initiation document

IDE – Integrated development environment

GU – Graphical user interface

CS – Computer science

IT – Information technology

MatchUp – Name of the developed application

DB – Database

# Introduction

The main aim of this project is to improve methods of finding peers and organising events in terms of pleasurable sports with the aid of current technology. Techniques currently on the market are insufficient to fulfil this aim. For example, several current methods exist as websites which work as administration management, disregarding the need to be able to meet people and plan events. As well as this, websites are not portable enough for the intended function. Some methods have been implemented on mobile devices however only available on a single operating system. Therefore, ignoring half of the world’s current market. These are only a small dose of the current problems which realise the need for a more efficient and appropriate solution.

The motivation from this project derived from ongoing conversations with a family member who was constantly disgruntled by the fact that he could never find someone to play badminton with. His current circle of friends either did not enjoy the sport of were busy when he was not. The idea then stemmed that if he was able to be exposed to a larger group of people who had similar interests, it would be easier to find a match partner. This had the potential to benefit many people as it was possible several people experienced the same problem too. In future, the development of this project could benefit group sports such as football, allowing several people to meet and communicate in a single, organised way. There is also potential to incorporate the use of an activity tracker owned by the user, further promoting the project and the idea of staying health.

To create an effective solution to the discovered problem, there is much to be considered. For example, all methods currently in use as well as user requirements and general objectives. The produced system will allow a user to find others to play the same sport that they are interested in. The system will also allow for communication and event management. If the end product does not contain these features and meet these requirements, the project will be classified as a failure. Otherwise it is a success.

To be able to develop a system that meets the intended purpose, thorough investigation will be completed into all current technologies available to help produce the system. There will also be research into current systems on the market, so that the produced system can make up for wherever they fall short. Offering the used a system that is entirely functional and useful to them. Use of the PID (see appendix 1) will guide through the initial research for the problem specification. Allowing for a basis to start the development from.

This report will guide the reader through the development of the project, starting with articulating the problem and creating a specification. A literature review will be included as a form of research into the subject and surrounding topics. A justified solution will then be proposed, followed by the design, implementation and testing of the system. There will then be a discussion of the progress of the project and reflection on contribution as well as discussion about the health and safety implications of the project. Finally, the future implications and improvements of this project and produced system will be considered before concluding the entire project.

# Problem Articulation

Before any solution can be built, the situation as is must be analysed. This allows for an accurate diagnosis of the problem which leads to the development of different ways to solve the problem. Many factors must be considered, such as the potential requirements necessary to solve the problem, anyone that is involved with the project or system, and the benefits and constraints of any given solution. After all of these factors have been discussed and determined, it is possible to see a devised solution forming. This will expectantly solve the problem.

## Problem Statement

As technology advances, humans have created a habit to evolve around it. This in most cases such as medicine and transport for example, is extraordinary. With continually growing potential to improve the current quality of life. However, in terms of the development of portable technology such as mobile phones and tablets, it is debated that the development of such technology has stunted the social growth of the world. Instead of communicating verbally people easily text each other. Making it much harder to meet new people.   
As well as this lack of social interaction, with the development of technology in the entertainment industry, people are more satisfied sitting and using their devices to play games and watch movies rather than being satisfied by exercise (Rock, 2018).

In a study conducted by a group of academic individuals, it was quickly found that increased times spent on viewing television and using computers were associated with the increased prevalence of obesity amongst adolescents (Kautiainen et al., 2005). Figure 1 represents the ratio of overwight boys and girls by which their time was spent using technology. It shows for example that viewing television for at least 4 h per day was positively associated with overweight in all age groups in girls (OR 1.93, 95% CI 1.09–3.42 in 14-; OR 2.91, 95% CI 1.40–6.04 in 16-; and OR 2.13, 95% CI 1.09– 4.14 in 18-y-olds). Therefore, proving that technology was demoting exercise as it is much easier to gain enjoyment from playing a computer game for example than a sport.

However, what if the two (technology and exercise) were combined in a way that made meeting people easier. Motivating people to play the sport they liked more, as they have someone to play it with. This would promote exercise across the board and have the potential to decrease statistics shown in Figure 2, resulting in a healthier and happier society.

## Technical Specification

The application will have the ability to be hosted on a mobile phone or tablet device with android as the operating system. There are some specific features necessary for the application to be a success. Such as a feature to allow for communication, event planning and lastly a way for users to know that other people exist and are interested in the same sport.

Whilst adhering to the PID, there are a number of technically related objectives that the result of the project should satisfy to be deemed a success. The following objectives have been taken from the PID and revised, so that they align with the developed scope of the project.

These requirements have been categorised into ‘Must haves’, ‘Should haves’ and ‘Could haves’ to delegate the priority of the objectives.

* Must have requirements

These objectives are those that are completely necessary for the project to be a success. These requirements include:

* 1. This project will produce an android application.
  2. The application will be able to run on a portable platform.
  3. The application will pair the user with others that want to play the same sport as them.
  4. The application will allow the users to create an event that will be saved in their phone calendar
  5. The application will allow users to communicate with each other
  6. The application will make use of a database to save all user data
  7. The application will allow for the user to change the chosen sport
* Should have requirements  
  These are the requirements that are not completely necessary for the project to be a success, however would be highly beneficial to the system if included. These requirements consist of:
  1. The application will utilise a GUI that is comfortable for the user
  2. The application possesses user profiles
  3. The application will allow image sharing
  4. The application will pair users with others within a specified area radius.
* Could have requirements  
  These are the requirements that are not completely necessary to the project, and therefore are low priority. However, if it is possible, these requirements would also benefit the system. These include:
  1. The application will allow for the user to edit the personal information which other users can see.

The highest priority requirements are the ‘Must haves’. Next are the ‘Should haves’ and lastly the ‘Could haves’.

## Stakeholders

The stakeholders in any project are crucial to a successful end product. Each group that contribute to the project must be carefully considered whilst creating a solution to ensure all needs are satisfied and the project is a success. The key stakeholders are listed below.

1. **Developer (Zamena Jaffer)**

The developer is concerned with ensuring that the solution is developed to a standard that satisfies the technical specification in section 2.2 as well as the user acceptance criteria in section 2.7.

They must ensure that the project runs without error and meets all provided timescales and objectives. As well as this, they must be able to deal with any situations where problems are encountered, or deadlines are missed, making informed decision to still ensure high level quality of the solution being developed.

1. **Project supervisor (Frederic Stahl)**

The project supervisor is reliable for overseeing the project that is undertaken by the developer and offer guidance and support throughout the entire duration of the project. The project supervisor will evaluate the success and/or failure of the project on completion.

1. **User**

The user is debatably the most valuable stakeholder as they will be utilising the solution that is developed by the developer. The user models the user acceptance criteria in section 2.7 and their satisfaction is a major part in assessing the success of the project as a whole.  
The user will also play a big part in section 7 when testing the devised solution to ensure it works to a standard that is acceptable to them.

## Project Motivation

The motivation for this project stemmed from a conversation shared with the Developer’s brother in which they discussed the problem that they wanted to play a certain sport, however, could not find someone who was also interested to play with them. The conversation then progressed to talk about the idea that so many people are so used to using their phone devices to communicate with the friends that they already have, so it was almost impossible to meet someone new unless people were forced.  
After that conversation the developer realised that if they incorporated the concept of meeting people, sport and mobile phones, then the problem could potentially be solved. An application that could be downloaded from an app store which allowed users to meet likeminded peers to pursue their interests with. This would make meeting people much more comfortable for those used to the smartphone life as well as promote a healthy lifestyle with exercise and social interaction.

Initially, a PID was proposed describing the application with more complex features such as user ratings and the incorporation of an activity tracker. However, after reviewing the time limit and the level of competence possessed by the developer, these features were deemed unobtainable. Therefore, only the most necessary features that allow the solution to function in a way that is effective and efficient have been brought forward. The left-over features will be revisited given there is time after developing the solution to implement them appropriately.

This project has high potential for future development as if the unimplemented features from the PID are still relevant, then they can be further improved before being implemented. For example, instead of just having a activity tracking feature, it could be possible to pair an activity tracking device to the application for more accurate readings of the users progress. As well as this, with much further research it could be found possible to improve application performance altogether.

## Constraints

These are all the factors that will prevent the project from continuing as desired. These constraints can lead to a solution that does not meet the needs of the stakeholders. Therefore, they must be identified and discussed so that they can be either avoided or overcome.

### Project constraints

These constraints will influence the development of the project overall and therefore must be considered.

One of the most obvious and inconvenient restraints is the schedule. The solution only allows for around 9 months from the start of the academic term to the complete development of a system that solves the problem. Though this is feasible, it will require high levels of project and time management to ensure delivery of a solution that meets all requirements and functions as desired.

Another constraint is the competence level of the developer in working with the chosen programming language and IDE. Though there has been some experience in these departments, the developer still holds a novice status and is nervous. However, due to the nature of this constraint it is possible to overcome. This is due to the fact that there are many different resources available either via the university library, teaching staff or the internet which provide much support in understanding and learning to work with java and android studio. These resources will be heavily utilised to result in a solution that works for its intended purpose.

A possible constraint is the strike that occurred in the spring term of the academic year. Where a number of university staff refused to work for 3 weeks of the term. Though the project supervisor was not one of the staff members who partook in the strike, others that did could have provided useful help in an important time of the project life cycle. Therefore, resources were limited for a certain amount of time. And though the project was not majorly affected by this event, it is still a contributing factor to the development of the project.

### Technical constraints

This is a list of constraints that must be considered as they will affect the development and implementation of the system that is created.

1. The performance of the application is limited by the hardware device
2. The application will be built for one specific operating system
3. The cache of the device may result in small amounts of storage space

## User Acceptance Criteria

These following requirements are related to the user of the system. They are one of the most important stakeholders as they will be the ones utilising the system. Therefore, the application must satisfy their standards. The application must meet these requirements before the project can be deemed a success.

1. The user can run the application on their mobile device
2. The user can choose a sport they are interested in
3. The user can see other users on the system that want to play the same sport
4. The use can upload an image to their profile
5. The user can change their personal information
6. The user can change their chosen sport
7. The user can communicate with other users
8. The user can create events

# Literature Review

Before starting the development of the solution, it is important to familiarise oneself with all related subject matter. Any relevant literature must be reviewed to develop understanding of all related concepts. This will allow for more informed decision making within the development life cycle process. Here, research will be conducted into any associated concepts and conclusions will be made from findings.

## Operating system

There are several operating systems available in which the project solution could be developed. In most cases it would be sensible to develop the solution to be available in all operating systems. Therefore, allowing it to be accessed by all users that use different operating systems. However, due to the limited amount of time and lack of knowledge on a system that would allow this, it would be much more sensible (considering the context) to develop the solution in one operating system.

According to IDC, the most popular operating system is Android. Dominating the market hare for worldwide smartphone operating systems between 2014 and 2017 by over 80% as shown in figure 2 (Chau, Reith and Nagamine, 2018). This makes the choice clear that Android would be an appropriate operating system because it is used by a large majority of the market. Meaning that if the application is released on android only, it is likely to be used by more people.

There are many benefits to using android operating system to develop an application. For example, Android is an opensource platform therefore all means necessary to develop with it are easily available to developers. As well as this, it allows for development of a comfortable user interface which is crucial to an application of this nature. As the user must be comfortable with the GUI to happily use the application (creativethoughts, 2018).

After reviewing literature related to the concept, it has been decided that android is an appropriate choice for an operating system and no further research into the field id necessary.

## Programming language

In terms of programming languages, there are several available in which the solution could be developed in. Each with their own benefits. For example, in a forum created on the Intel website, it was discussed what the best language for the development of mobile applications is. After much debate a user states that quite obviously it depends on what the application requires (Kostrov, 2012). If the application only requires high level performance, then a language such as C or C++ would be appropriate. These are compiled languages; therefore they are close to the machine hardware in nature, resulting in fast performance. As well as this, it is very mature and has many resources available for support Gribble, 2012).  
If the user interface is the focus of the application, then a more high-level language is necessary such as Visual basic or Java. Visual basic allows for cross platform development of an application. Allowing it to run on several platforms all at once (Msdn.microsoft.com, 2015). However, since it has already been decided that the application will be built to run on android, this is unnecessary.

Performance is a key factor to developing a good application however the nature of this solution calls for a comfortable and efficient user interface. Therefore, Java is an option that is appropriate. Java is a commonly known and widely popular programming language due to its many features that make it extremely useful. For example, the probability feature, also known as ‘Write once, use anywhere’ allows the application to be deployed on any hardware as many times without having to change any of the code. As well as this, multithreading allows for multiple users to utilise the application at the same time, which is a priority for an application that is available to a huge market. Multithreading also results in fast response time, higher performance and fewer bugs or glitches (Kozlov, 2017).

Java is also a highly flexible object orientated programming language which allows updates to be done easily and changed to be made quickly. This will be important when discussing the future development of the solution and how it will be upkept.

After conducting research, it has been found that Java is the most appropriate language to use to develop an android application that works optimally well.

## Integrated development environment

Java can be use with many IDE’s. The most popular of which include Visual studio, Eclipse and android studio (Krill, 2017).

Visual studio is the most popular IDE and can be used for app development as described on the Visual Studio website. VS allows for cross platform mobile development tools by aiding the development of hybrid apps, utilising 2 different types of code or more. As well as this it provides an emulator to run apps on and provides a wide array of tools that can be utilised in development (Webster, 2018).  
The advantages of using visual studio are plentiful, for example it is efficient in integrating a database into the application that is being created. This is useful in this context as the solution will utilise a database to store all user data. As well as this, because visual studio is the most popular IDE, there are many resources available in learning how to develop with it. This is beneficial as the developer for this project is only highly experienced in android studio.  
However, visual studio also has challenges to work with. Compared to other environments, this IDE runs a lot slower than many others. As well as this, there are memory constraints as using visual studio to code in takes up much memory when first installed onto the device (David, 2015).

Eclipse is the next most popular IDE and it is more of a highly extensible platform rather than a tool for a specific task. Eclipse defines the mechanisms and allows tools to be built on top of them by providing plug-ins that are utilised by the developer to create the application (Cinar, 2012).  
There are benefits of working with Eclipse, for example, it uses a high level performance debugging tool that allows for a step by step breakdown of each like of code. This makes it easier to narrow down the problem and results in a faster resolution time for the problem. As well as this, it allows for global refactoring, making small changes in the code easy to complete.  
There are also disadvantages with using eclipse. It is a very complex IDE to get used to and contains a lot to learn if there has been no experience with it before. The developer of this project has tad little experience with this IDE therefore working with it may be hard to start with.

Android studio being the third most popular IDE, is still the most promising to be used in this context. For obvious reasons such as the fact that the chosen operating system is android, as it is specifically developed for android development.  
Android studio also contains a separate design feature, allowing the developer to manually design the user interface, making use of code and interaction to create a perfect GUI for the intended purpose (Rajput, 2015).

The developer is most comfortable and trained in using Android studio therefore this is the IDE that will be utilised to develop the solution. However, it has also been defined and clearly understood, the different benefits of other IDE’s that exist on the market today.

## Firebase

Firebase is a set of tools offered by Google to aid in the development of scalable applications, based on the cloud. Providing features such as hosting, performance monitoring, analytics and more. It can be used with the development of both Android and IOS applications and most importantly, can be used with Android studio, which is the decided IDE that will be used to develop the solution.

Though it is debated that firebase is not the most efficient middleware for application development due to the fact that it is still new and has not been used by many enterprises. Therefore, it is not guaranteed to be successful. The benefits heavily outweigh the challenges. For example, it allows for real time updates to the database and datastore, requires minimal experience to understand and set-up, and provides a massive amount of storage to applications whose userbase is likely to grow a substantial amount (Cleveroad Inc.-Web & App development company, 2018).

There are two main features of Firebase that will definitely be utilised in the development of the solution which are as follows;

### Real-time database

A cloud-hosted database which allows major flexibility for the developer as it syncs all data across all platforms in real time, meaning the sync does not need to be triggered manually or periodically. This is possible because the database uses data synchronisation instead of queries and requests. Therefore, any time any data changes, it is automatically synced

This feature will be useful with the proposed solution as it will have users registering to the application constantly in high volumes, therefore the data needs to be synced constantly and immediately to allow for accurate representation for the users, of who else is available on the app for them to play sports with.   
As well as this, the application will allow users to edit personal data whenever desired, meaning certain values in the database will need to be changed when the user saves these profile changes. Real time database synchronisation will be benefit this situation.

The database is also available to the Developer through any device with an internet connection to be edited or managed. Making the job for the developer easier in any time of emergency.

Data is also stored locally on the user’s device so that the return time of information is quick and available even when Firebase goes offline (Cleveroad Inc.-Web & App development company, 2018).

### Authentication

Firebase also provides aid in allowing users to be authenticated when registering to the application. Giving the option to allow log in with specified social media accounts or just the users email address. This service takes the users registration information (email and password) and stores it in an encrypted data store which only the developer can access and edit. Ensuring that the data is secure and cannot be infiltrated.

Firebase allows walkthroughs of how to integrate their code with the services of firebase, by just including a single API reference. Making the work for the Developer much less complicated, handing off data storage to Firebase. This allows the Developer to focus on more complicated tasks within the code (Cleveroad Inc.-Web & App development company, 2018).

## Existing technologies

It is useful to review other similar applications on the market at the moment. This will allow for an informed decision of what the proposed solution must include, therefore making up for anything that current technologies or applications do not include. This will ensure the solution highly satisfies the requirements of the problem.

### Eventbrite

(Eventbrite.co.uk, 2018)

Eventbrite is an event planning application that asks users for the genres that they are interested in when they register. The application will take that information and filter through all the registered events on the system and only show the relevant ones to the sure. These events range from theatre to dinner. The application will only show events within a specified radius and allows users to select from options listed ‘interested’ or going. Users can also invite other people to these events (generating a sharable link). This application allows users to buy tickets for the events too.   
This application is relevant to the project as it specialises in event planning. It consists of features such as a locating based search and communication.  
Eventbrite is available on both Android and IOS operating systems and can also be used as an online website. Therefore, it is easily and readily available to many users.  
Though this application exists, the proposed solution is still relevant as this system specialises in event planning and management in all sectors. Whereas the proposed solution is only related to sports.   
It is possible to take inspiration from this application to carry into the proposed solution. For example, this system allows image sharing for a better understanding of the situation. As well as this, there is a separate ‘page’ called “Messenger”. Indicating that all communicational aspects of the application are kept on their own separate page. This is a good idea as it keeps the GUI of the app uncluttered and comfortable to use.

### Snafflz

(Snafflz.com, 2018)

Snafflz is an event creating and management application, specifically developed to manage a guestlist at large scale events. It is available on both IOS and Android operating systems as well as Windows as it is available as an application software on computers as well as phones and tablets. It provides many features to the user such as real time data updates, email management, registration, even QR scanning. This application is utilised by large enterprises when organising large events with many people attending.   
This application is not directly linked to the project theme. However, has many useful features that can easily inspire the design of the proposed solution. For example, real time data update will be extremely useful when registering large amounts of users at the same time. As well as this, email management is a feature that would be useful when the user is successfully registered to the system.

### Sporty

(Sporty, 2018)

Sporty is a mobile application with a premise very similar to the one proposed. It helps users find others to play a certain sport with. The application allows the user to pick a sport at the home screen that they want to play. The application will then take the user to a list of people that also want to play the same sport in a proximity to them. The user can then talk to this user.   
The Sporty application is only available on IOS whereas the proposed solution will be available on Android, filling that gap in the market. As well as this, there are similar to the types of features that are required for the proposed solution, however there should also be a feature to allow for event planning.  
The use of image gives this application a clean and neat interface, making it extremely easy to interact with. This is something to be considered when designing the proposed solution in section 5.   
This application also allows for a profile as shown in the bottom right hand corner of Figure 5. This is a feature that the proposed solution is required to have, allow the user a profile to edit personal information at any time.  
One other thing to be considered is the fact that the user can choose which sport they want to play from the home screen. In the proposed solution it was initially thought that the user could chose a single sport when registering but had the ability to change the sport in the settings of their user profile. However, this option portrayed in Figure 5 may be much more user friendly. This will be seriously considered in section 5 when designing the proposed solution.

### Tinder

(Tinder, 2018)

Tinder is a popular dating application that matches users on certain preferences such as location, age, gender and sexual orientation. This application is available on android and IOS, across platforms. It makes it extremely easy for users to meet people by using a simple swipe or button feature. If the user likes the profile on the screen, they can either swipe the profile to the right of the screen or press the green heart as shown in Figure 6. Otherwise they will swipe left or select the red cross. The users are able to upload images and a small profile with personal information about them to make them more appealing to other users.   
Though the goal of this application is not related to the proposed solution, the general theme of meeting people is. However, instead of matching users on so many different preferences, the solution will only match them based on the sport that they want to play.   
One feature that could be adapted in the design stage of this project is the swipe cards. This feature is used in many applications and makes navigation simple and effortless. It is almost universally known that to pass on something, a user must swipe left but for the opposite, the user will swipe right. 

### Teamer

(Teamer.net, 2018)

Teamer is an application that allows a user to manage a sports team. It creates groups that other users can join (if for example they are part of a football club) and can access certain features. For example, users can pay subscription fees to be part of a football club, they can also view who is playing a match when and where. As well as this the users are able to share albums of images taken whilst playing a match between the group.  
This application does not allow live chat between all users within a group, but it does provide contact information of all members in the group, making it easy to contact them if necessary.  
This application has slight links to the proposed solution as it focuses on event management, however it is developed more for group events.   
The application is available on android, IOS and as a desktop application, therefore is accessible by many.

## Human Computer Interaction

HCI is crucial to be considered when developing any solution to do with technology hardware. It is the study and practice of human interaction with computers. Including the extent to which computers are or are not developed for successful interaction with humans. When designing a mobile application this must be considered as a mobile device is still a type of computer that each user will interact with (Carroll, 2013).

It is discussed that mobile devices pose five main issues when considering HCI in designing an application (Dunlop and Brewster, 2002). These are as follows:

1. Designing for mobility  
   whatever application or system being designed must consider that the user is on the move. This is one of the main assumptions when designing a mobile application as it is the main implication of using a portable hardware device. Therefore, any props or notes necessary must be considered in context to the fact that the users working environment is likely to constantly change.
2. Designing for a widespread population  
   Users will not have any training or support in the application, therefore it must be easily understood by most of the population it is being used by. No technical jargon must be used, and universal language and symbols are a necessity.
3. Designing for limited input/output

Only a limited amount can be displayed on a small screen such that belong to portable devices. Image quality, sound quality and information quality will all be lower than when working with a computer and keyboards are limited in size. Therefore, an application must consider the fact that asking for too much input or delivering too much output will be strenuous on the user. Single press buttons or small text fields would be most appropriate.

1. Designing for varying context information

Mobile phones have the ability to be aware of their situation through different sensors such as GPS. This provides the potential to gain much information but also the problem of being inaccurate or unreliable. Therefore, is used, they must be used correctly or with the knowledge of skewed data.

1. Designing for users multitasking at levels unfamiliar to most desktop users  
   All users have different levels of skill; therefore, all applications should be developed with this in consideration. So that when the application is complete, any level of competence user is able to work it.

These five features must be heavily considered in section 5 when designing the application.

# The Solution Approach

After conducting thorough research in the previous section (section 3), I am now fully aware of the different technologies, methodologies and statistics that the solution will be developed upon. Leading me to make informed decisions on the type of solution that should be proposed. This section is where some potential solutions will be outlined with descriptions of their features and why they would be relevant to satisfy the problem. The solutions will then be discussed to decide on the most appropriate one. And finally, the chosen solution will be clearly defined. This will allow for a distinct understanding of the solution to be developed.

## Potential Solutions

Following on from the conclusions in section 3, It was decided that the solution would be most efficient if developed as an Android mobile application. Therefore, the following proposals for solutions will focus on the features and GUI that the application will possess.

1. **Application focussed on event planning and management**

The android application could be developed with a similar premise to that studied in section 3.5.1. Inspired by the theme of Eventbrite, the application focuses mainly on the event aspect of the problem.

In terms of features, when the user logs into their account, the first screen that they are shown is a list of events occurring in their local area of which they are able to sign up to and attend. All users have the ability to create events and state information such as where, when and a description of the event.  
However, contrary to the Eventbrite application, this solution will not allow the exchange of money as this application is purely for leisure purposes, allowing people to connect. It would be unethical to allow users to charge each other to peruse a hobby of which they could easily do for free.  
Though adding and communicating with users will be important within this solution, there will be no real need for user profiles as the main focus is essentially advertising different sports related events that can be attended.  
Users will be allowed access to communicate with others only if they are attending the same event. This also shows potential for group live chats to be utilised if the event requires more than one person to play.  
In reference to the type of events shown, the user will be presented with all sports events registered with the application and will have the option to filter through the lists of events to show them only a certain sport, location or time.

1. **Application focussed on social interaction and event creation**

This android application is loosely inspired by Tinder, an application already on the market that was analysed in section 3.5.4. The application would adapt the focus on connecting different types of users based on the chosen preference (this being the sport that they want to play).

The main feature of this application when the user logs into their account is the swipe card mechanism. Each other user that is registered to the application and interested in the same sport will be displayed in a stack of cards. The card will contain information such as username, a short description (preferably user experience level) and an image. The user can then either swipe right to agree to communicate with this person or left to not. If the other user also swipes the card right, the users are matched and are able to communicate in a live chat. Here is where an event creation feature will be introduced, so that if after the users are met and comfortable, they are able to create an event to meet and play the chosen sport. This event is saved in the user’s phone calendar and reminded to the user until it happens.   
Due to the nature of this application, user profiles will be necessary as the social interaction is a major part of the application. This therefore means that there will also be a feature where the user can edit their personal information including description and profile image for example.

When the user registers to the system, they will be asked which particular sports they are interested in. They will then select each that they prefer and once the user is registered/ signs in, they will be presented with users interested in the same sports. The user will have the ability to change these chosen preferences the same way in which they change their profile information. There will be a settings page in which they can edit all of this information.

## Solution Discussion

Now that there are some potential solutions that have been proposed, they must be discussed and compared to previous research such as the technical specification, to determine which solution would be most effective in meeting these requirements and fully solving the problem at hand.

Within section 2.2, a technical specification was devised where the requirements for this project to be a success were categorised into 3 main classifications. Must haves, should have and could haves. Of these three classes, the highest priority requirements are the Must haves. Therefore, the proposed solution must be able to satisfy the 7 key requirements proposed in that section.

Requirements 1 and 2 were as follows:

1. The project will produce an android application
2. This application will be able to run on a portable platform

These two requirements have already been discussed in section 3. More specifically in sections 3.1 to 3.3. Therefore, no further acknowledgement is needed as it is already determined that the application will be developed to meet these requirements before this section. So, both proposed solutions satisfy these.

1. The application will pair the user with others that want to play the same sport as them

Both potential solutions A and B are able to satisfy this key requirement however, they are able to do so in very different ways.

Solution A presents the user with a list of all currently ongoing events in which they are able to register with, they will then have access to see all the different users who are also interested in them and can interact with them from this point onwards.  
This solution does not match the users specifically on their preferred sport, however lists all of the sports with the option to filter the events. Resulting in a single chosen sport to be presented to the user.

Whereas solution B asks the user for the sports they are interested in when they register to the application. They will then take this data and compare it with other users already on the application system. This allows the user to be accurately matched with only the other users on the system that are interested in the same sports. The user then chooses if they want to interact with the people presented to them.

Whilst solution A does present the user with other people on the application who are interested in the same sport, there is a small process necessary to get to this. Whereas solution B immediately shows the user everyone that is registered to the application and interested in the same sport as soon as the log in/register to the application.

After analysis, it is understood that solution B satisfies requirement 3 much more effectively than solution A is able to.

1. The application will allow the users to create an event that will be saved on their phone calendar

As solution A is focussed more on event management and planning, it is instantly obvious that this proposed solution is more likely to satisfy the requirement. Users are able to create events on the application which other users can register to attend. Initially, this event is not saved in the phone calendar but on the application as it would take up too much memory on the device hardware. However, when a user chooses to attend the event, a reminder is automatically (with user permissions) saved in the user’s device calendar so that it is reiterated to them closer to the time.

This being said, solution B also has the ability to meet this requirement. When the user has matched with and chosen to communicate with another user on the application, they are able to create a calendar event that is saved directly onto the user’s device calendar.

Though both solutions are able to satisfy the requirement, solution A is able to meet the standard much more effectively. Saving the event automatically to the user’s device is much easier and quicker for the user than having to create and save the event manually (as required in solution B).

1. The application will allow users to communicate with each other

Solution A allows communication between users if they are attending the same event. To access this, the user will register to attend an event, they will then have access to see everyone else who is attending the event and can communicate with them from there onwards. Therefore, solution A is able to meet requirement 5

Solution B also allows communication between users. To allow this, two users must match. This forms a mutual agreement that both users consent to interacting. Once the users match they can go to a page that lists all users that have matched with and can start a live chat. So, solution B also satisfies requirement 5.

Both solutions are able to meet the fifth requirement, however solution B allows for a safer process of communication as both users must agree to interact before communication is allowed, whereas solution A allows communication with anyone provided they are attending the same event, no filters included. Therefore, though they both meet the minimum standard to fulfil this requirement, Solution A is much more effective and safer in doing so.

1. The application will make use of a database to save all user data

Solutions A and B both make use of a database as both applications will require the user to register to the system, providing some details that will be saved to the database. This ensures that users are safely registered, and all their data is saved so that there is no confusion if two users want to use the same log in identification. The user that registers first will receive their desired identification to log in with and no one else can log into their account unless they have their log in details.

Therefore, both solutions A and B satisfy requirement 6 equally.

1. The application will allow for the sure to change the chosen sport.

Solution A partly satisfies this requirement. This is because the user does not have to choose a specific sport to use the application, they are shown all events from all sports and can choose from these events. However, the application does allow the user to filter the events in terms of sports. Therefore, in this context, the application partly satisfied the requirement. As they can change the sport by changing the specified filter.

Whereas solution B asks the user to choose the preferred sports when they register to the system and allows the user to change their profile information/settings. Therefore, letting the user change the sports they are interested in. This will then change the group of people that they are presented with when they use the application. So, this solution fully satisfies requirement 7.

As a result of this analysis, it is decided that solution B is the most appropriate solution in reference to fulfilling requirement 7.

## Solution Definition

After analysing the potential solutions and comparing them with the technical specification, an informed decision can be made on which proposed solution will be chosen to develop, based on which is able to more appropriately solve the problem.

### Chosen solution

**Solution B - Application focussed on social interaction and event creation**

Proposed solution B is the chosen solution which will be further developed through the rest of this report, resulting in a functioning android application which meets the technical specification and user acceptance criteria. Tin turn, solving the problem.

### Justification

Whilst solution A showed much promise in aiding to fix the specified problem in section 2.1. When compared against the technical criteria, solution B was more able to effectively fulfil each and every requirement. There were none that were partly or not fulfilled. And in most cases, solution B was the most effective in satisfying each requirement individually. Whereas solution A fell short of fully meeting requirement 7.

As well as this, reviewing the solutions in comparison to the technical skill of the developer, solution B is seen as more attainable. Especially when factoring in the time sale by which the solution must be completed for and comparing this with the time needed by the developer to train in the already required technologies. Therefore, solution B is a much more realistic option.

### Definition

The android application developed throughout this project as discussed above in sections 4.1 through to 4.3.2. will be a mobile device application that allows users to register and log on to a platform that aids them in interacting with other users who possess the same interest as them. As the user registers to the application, they will provide basic information such as their name, email address, a small description about themselves. As well as this, they will be asked to select all of the sports that they are interested in (presented to them in a list of the most popular sports). Once the user creates the account, this data will be stored in a Firebase database. The application will then search the database for all other users who have selected the same sports and display this data to the user in the form of a stack of cards. The user will then have the choice to swipe either right or left to decide on if they want to interact with these users. If two users both swipe right on each other’s profile, they have matched and are now able to communicate with each other.

Communication is allowed on a separate page to the main page where there is a list of people that the user has matched with. The user can then select any of these people to begin a live chat with.

Once they select a person to chat with, they are taken to a new screen where there is a message box and an area in which send messages are seen. At this point, the users can choose to create an event by clicking a button next to the keyboard. This will then take them to a screen where they can pick a date for the event, enter an event name location and description, and save this information. This data is then saved onto the user’s device in their calendar. This allows them to be reminded that they have an upcoming event.

# Design

This section of the report will take what was discussed in all previous sections of the report to create a well thought out plan for the development of the entire solution. Consisting of UI design, application design and database design. Each section is crucial to the development of the application and everything discussed in sections 1 to 4 will be considered and put to use.

## User interface design

This requires designing the GUI of the application. Which refers to the part of the solution that the user interacts with. This is extremely important because if an application has an inappropriate GUI, users are less comfortable in using it and therefore will not use the application. As discussed in section 3.1. there are 5 key challenges that must be taken into account when designing a system that requires human computer interaction. These will all be heavily considered when creating the plans for each page of the application.

### Start page

This is the first view of the whole application the user has when they first download or open the app. Therefore, it must create a good impression. It should be clean and uncluttered with universal language instructing the user on what to do. The main components needed on this page is the logo of the application, and two buttons. One which leads the user to a log-in page and one that leads the user to a page where they register. 

Figure 8 shows a detailed representation of the desired design of this page. The page will consist of two buttons and one image. The image will be of the application logo. This will ensure a comfortable aesthetic as well as acknowledgement for the user that they are on this application. The two buttons consist of ‘Login’ which will be selected if the user has already registered to the application, taking them to the log on page. The other button will be ‘Register’ where the user can sign up to the application.

Figure 9 shows the actual design of the application after development. The design is close to the desired, however a label has been added which reads “Welcome to MatchUp!”. Making it a more personal experience for the user as they are greeted.

### Login page

Here is where the user returns to after they have registered. This page must again contain universal language when asking for the user’s identification and password to grant access to the users account.

Figure 10 represents an annotated illustration of the desired design of this log in page. It consists of 6 main components. A label which reads ‘Welcome to MatchUp’, greeting the user when they log in. A label that states ‘email address’ which implied the user should input the email address that they registered to the application with in the text field below. As well as this there is a label that says ‘Password’, prompting the user to enter their associated password in the text field below to gain access to their account.

The actual design of the login screen is much simpler compared to the plan as shown in figure 11. Firstly, the welcome label was moved to the start screen as this is constantly the first screen that all users see when they open the application, therefore this made more sense. As well as this the email and password labels were disregarded, and hidden text fields were used so that when the user first encounters the email text field for example, it states that the users email address is required. When the user selects the text field, the text that says ‘email address’ will disappear and the user is free to type their information. This makes the screen much more neat and cluttered. The button component was kept the same.

### Registration page

This page is important as it is the users first real interaction with the application. It is crucial that the application is able to obtain all of the necessary information from the user without overwhelming them by asking for too much. Here is where the user returns to after they have registered. This page must again contain universal language when asking for the user’s identification and password to grant access to the users account.

As shown in figure 12, the registration page holds many components to obtain certain information from the user. There is a label and corresponding text field for the user to ender their name, email address, password and bio (a short description about themselves). This stage also asks for the user’s gender in a radio group consisting of two radio buttons as well as a radio group with 10 radio buttons which asks for the sport the user was interested in. Lastly there is a button labelled ‘REGISTER’ which saves the users information in the database and takes them to the main application screen. 

Figure 13 displays the actual outcome of the registration screen. As we can see a majority of the formatting stayed the same, however, instead of including text fields and labels, each text field contains text that disappears when selected as a prompt for what the user should enter in each field.

### Main page

At this page, the main functions occur. The cards with user information are displayed to the user. The developer must be careful to ensure that all the necessary information is displayed but at the same time, the user is not overwhelmed by huge amounts of information cluttering the screen.

This page should also have links to the settings and matches pages as well as the option to log out of the application when desired.

Figure 14 shows the desired design of this main page. At the top there is a navigation bar consisting of three buttons that when selected will direct the user to the corresponding page. In the middle there is a ‘swipe card’. This is a card that can be moved around the screen and is swiped right to interact with the user and left to ignore. This card consists of another user’s name, bio and sport they are interested in. Giving enough information to the current user to make a decision about whether or not they will attempt to interact. 

Figure 15 displays the actual outcome of the main page with a depiction of the swipe card to the right, and the navigation bar in the left image. When the application is run, the two views are combined with the swipe card underneath the navigation bar. A minor change was made in the process of development where the user sport was removed from the card. This is because the user is being paired with only users who are interested in the same sport therefore it would be futile to leave this on the card. As a consequence of this, the image has been enlarged to make up the space.

### Settings page

Here is where the user can change any of their personal information such as profile image, name, and description. There should also allow for the ability to change the type of sport the user is interested in. This page should also include a save button to ensure user that their information has in fact been updated.

Figure 16 shows the planned design for the settings page with an image view of the user’s profile image at the top that can be changed if selected. Then there are 3 text fields and labels asking for the users changed name, contact number and bio. Next there is a radiogroup of sports that the user can select to change their current interest. Lastly there are 2 buttons. One labeled ‘SAVE’ which when selected will change the old information to the new information in the database. The other is labeled ‘BACK’ which will take the user back to the main screen.

As shown by figure 17, there were no major changes when developing this screen aside from the ones following the same theme of the previous sections.

### Matches page

This page lists all the people that have matched with the user. And should include 2 main pieces of information. The persons profile image and their name. 

As described by figure 18, the matches page contains a list of customised buttons containing the other users profile image, name, short description about themselves and the last message sent in the conversation between them. This screen contains a simple layout so as to not confuse the user.

The actual design outcome of this page is exactly as desired. With functioning buttons to the live chat feature between two matched users.

### Live chat page

This is a simple page with a text field, keyboard and an initially blank screen. This will fill up when users send messages to each other. This page also includes a button that takes the user to the calendar. 

As shown by figure 19, the illustrated design for the live chat screen is simple, consisting of a text view each time a user sends a message. The user’s device keyboard is also displayed as it will be used to type messages into the text field. Also included on this screen is a button that will direct the user to the calendar screen. This is where the user can create events.

The actual outcome from development of this screen is exactly as desired. No changes were necessary.

### Calendar page

Here the user is shown a calendar and 3 main text fields. Event name, location and description. The user will fill these out and press save so that the information is saved in their phone calendar.

This screen has been designed as desired with three text fields and labels for the user to input the event title, location and a short description as shown by figure 3. The user can then choose a date to hold the event using the calendar view. After all details have been input, the user can save the information by selecting the ‘SAVE’ button. This will save the details into the user's phone calendar.

Figure 21 shows that the actual outcome is the same as the desired in figure 20. No changes were made as no changes were necessary. The illustrated design specified everything that needed to be included in detail.

## Database design

FIRST NORMAL FORM STUFF

# Implementation

Here is where the process of creating the application in android studio will be discussed. Key challenges and successes will be deliberated, and each component will be outlined in an attempt to explain the route taken and why.

## Swipe cards

To begin the technical development of the application, a basic android studio project was created with a single basic activity (to start with). This would be the starting point for the entire application named the Main activity. On creating this project, the first feature developed would be the swipe card effect. This was previously discussed in section 3.5.4 when analysing current technologies and realising that the swipe feature on Tinder makes interaction with the application easy and comfortable for users. As well as this, the fact that it is now becoming universally know that to accept something a user should swipe right and to reject they should swipe left meant that this would make using the application easier for the user.

Therefore, the first thing done after creating the project was implementing a specific library that allows this to happen. A library was chosen to be used instead of being developed from scratch because this meant there was more time available to further develop the application. As well as this, this library is known for doing what it does well. Therefore, including it would improve the quality of the application overall. The library was accessed via GitHub from user ‘Diolor’ (Lorenzo, 2016).



To implement this library into the application, first some specific dependencies were taken from the code available and implemented into the “app gradle” as depicted by Figure 22. Then the remaining code to allow for the swipe card effect was implemented into the main activity and edited to apply to the context of this project (see Appendix 2.4).

The original code for the swipe card effect showed the user swiping through a set of cards with different types of language on them. However, after importing the code to the new application, the developer changed this drastically. 

First, item.xml (see appendix 3.9) was created to allow for the swipe cards to represent a string whenever a new card was shown (see figure 23). This is necessary to ensure that when developed further, instead of including just a word, the card will represent a user with an image, name and description.

Along with the creation of this main activity, another java class is created named ‘cards’ (see appendix 2.6). This class is necessary as it is developed to include all the code that populates the swipecard as seen in figure 23. Here is where 4 strings are created, userId, name, profileImageUrl and bio. These strings are populated using methods created such as ‘getUserId()’ which will filter through the user database (as specified in section 6.2) and find an eligible user (a user that has the same sport listed as the one who is logged in). The method ‘setUserId()’ is then called upon set each value with the correct information.

As well as this, a class named ‘arrayAdapter’ whose purpose is to take the information collected in the ‘cards’ class and display it on a separate card for each user. This is how the user is able to see different information as they swipe through the stack of users. By doing this, the developed application will now look as depicted in figure 24 as opposed to the simple original as shown in figure 23. However, at this point in the main activity, the only thing that would happen when the cards were swiped was that a message would appear on the screen corresponding to whichever way the user swiped. To assign an action to the swipe event, the original code was edited to create what is shown in figure 25.

The snippet of code shown in figure 25 shows a method that when called upon (so when a user swipes a card right) a certain event happens. The method gets the user ID that has been assigned to that card and updates the database under the database node ‘pass’. This is to show that the user has decided to not interact with the other user. The other user’s user ID is then saved under the specified node in the databse.

## Firebase

The next step of development was to get the application along with Firebase ready to sync and work together. To allow for this, first a file had to be downloaded from the Firebase server and input into the application ‘app’ folder. This was necessary in allowing communication between the application and Firebase. After this had been done, some dependencies had to be implemented into the application gradle to ensure that any code using Firebase functions would be accepted and not cause error. After all dependencies had been implemented, the google plug-in had to be included. This plug-in is crucial to the application development if a google product is in use. It is what allows the code to work together with firebase.

Once these have been implemented within the code, the developer is safe to start programming the application whilst using the Firebase services.

## Login and registration

This feature has been implemented in a series of 3 java classes. This is because the ‘log in’ and ‘registration’ both perform very different actions. As well as this another class is necessary to allow the user to pick between the two options from the start screen.

First the class ‘LoginOrRegisterActivity’ is created with two buttons in the corresponding .xml file (see appendix 3.4). Then within the code view of android studio, each button is assigned by id to an indicator. For example, the Login button is assigned to ‘mLogin’. Next an ‘setOnClickListener’ method is created for each button with the code shown in figure 26. The code states that when the login button is selected (onClick), then a new intent is created. This intent takes the user from the current page (LoginOrRegister.this) to the login page (LoginActivity.this). A similar event happens when the user selects the register button, however, the user is taken to the registration page.

When a user chooses to log in to the application, they are taken to the ‘LoginActivity’. This activity has been created with one button which the user will select to attempt to gain access into the application. There are also two text fields that are to be edited which are shown in the activity\_login.xml file (see appendix 3.3). Here, the application will make use of the Firebase authentication services and set a listener to analyse the information that the user inputs (email address and password), compare it to the Firebase authentication data store and decide if the user should be granted access. The user can only access the application if the exact same email address and corresponding password are in the data store. This is all shown in figure 27. 

Here (figure 27), when the user inputs the data and selects the login button, there is a ‘setOnClickListener’ which collects the data that has been entered into the email and password fields and then uses the ‘mAuth’ function to check them against the datastore. A validation check has also been implemented stating that is the task is not successful (the data does not match the store) then an error message appears on the user’s screen.

Programming the registration class took much trial and error, as the user inputs much information that must be read and used by the application. The .xml file for this activity (see appendix 3.7) required implementation of many text fields, and two radio groups. The text fields were simple to read information from. For example, the users name required a piece of code as shown in figure 28. Where the name text field has been assigned to ‘mName’ and the public methods ‘.getText()’ and ‘.toString()’ were used to read the data input and assign it to a sting called ‘name’. The same method is used to get the users email and password. However this information is used to create a new Firebase authentication (see appendix 3.7). 

The radio group items were much more challenging to read as they do not use the same simple function. After having no idea in reference to a solution, much searching on the popular programming forum stack overflow lead the Developer to ask their own question and receive an answer that helped (Mamo, 2018). Therefore, development continued and the solution was implemented (figure 29). To gain the information from a radio group, a method had to be implemented to get the selected radio button and convert that to text to populate a string. After this was one, the information must be saved into the Firebase database using a method implemented called ‘mAuth.createUserWithEmailAndPassword’. This stored those elements in the data store but next a database reference was created to gain connection to the database. The information is all then saved in the database under ‘Users’ by calling upon ‘userInfo.put’ which assigned values to the strings. And ‘currentUserDb.updateChildren(userInfo)’. This command instructs the application to update the database with the new information as shown in figure 30.

This concludes all significant note points necessary to explain the implementation of the login and registration features of the application.

## User matching

The social matching feature of the application is one of the most crucial in making it unique for its intended purpose. The most critical events in terms of code for this feature occurred in the main activity (see appendix 2.4). However, much background work was done in its own package called ‘Messaging (see appendix 2.8).

When the user registers to the application, they must choose a sport that they are interested in which is saved in the database. Then when taken to the home screen, the idea is that the user is shown more users who are interested in the same sport. When first developing the system, instead of a radio group in the registration activity a group of checkboxes were implemented. This was to allow the user to select several sports they were interested in and meet people with all interests. However, when it came to search through the user database to find the selected sports, this proved to be much more complicated than intended. Initially when the user would register, a node would be added to their user ID called ‘Sports’ and under that would be each of the 10 sports that the user could choose from. If the user had checked that box, then the sport would hold the value ‘true’ if not, the value would be ‘false’ (see appendix 4). However, when it came to search the database for the selected sports, it was found that Firebase did not allow for nested searches of a database node. It was too complicated. Therefore, a solution was needed.

After much deliberation it was found that the original idea of using a radio group where the user could pick one sport (and have the option to change it) was much more efficient. As well as this, it was much less confusing for the user. If the user had chosen more than one sport that they were interested in, they would be shown many users and have no way of indicating what their interests are. Therefore, this method is much more comfortable. So, the search through the database has been implemented as shown in figure 31.

The above snippet of code represents the method that was chosen to search the database for the current users chosen sport. The application checks the authentication datastore to get the information of the current user. A database reference is then made to access use usersDb and view the current users saved information. It is then checked if the value for sport exists. If it does then a switch is entered instructing the code that if (for example) the users chosen sport is ‘Boxing’, then a string called ‘matchSport’ is populated stating that the corresponding sport (or matching sport) is ‘Boxing’.

Next, the application will find all of the users in the database with the same chosen sport (see figure 32). 

The figure above clearly shows the method implemented to find users with the same chosen sport. First a child event listener is added to the user database (usersDB) and a data snapshot is taken of the nodes. The application then checks if the current users of the database have a value assigned to the child node ‘sport’. If they do, then a lengthy check is done. This like of code checks if the data snapshot exists, and if the user is not a passed connection of the current user, and if the user is not a played connection of the current user, and if the value assigned to the users child node ‘sport’ is valid in terms of the ‘checkChosenSport()’ method (as discussed above). If the user passes this check then they can be displayed to the current user.

Referring back to section 6.1, after the swipe cards have been implemented and assigned functions for whichever direction a card is swiped. The application must now be able to find out if two users are a match for each other. As discussed previously (section 6.1), whenever a user swipes right on another, then that other users ID is stored in the current user’s database. However, what happens if two users swipe right on each other? A method is required to search through the user database and check if two users are stored under each other’s ‘play’ nodes. Therefore, the method ‘isConnectionMatch()’ is implemented (figure 33).

As shown in figure 33, to check the database, a database reference is created to filter through all users that the current user has swiped right to (under the ‘play’ node). This method goes on to create a live chat if the two users are matched, however this will be discussed in section 6.5 and is not necessary now.

The Match package consists of four classes all linked to the item\_match.xml file (see appendix 3.11) which states that any use that has matched with the current one will be listed on this page with the user’s profile image, name, user ID and bio. This entire package is more concerned with the formatting and presentation of the matches that are displayed to the user in the form of a list than the functionality. First is the ‘MatchViewHolders’ class (see appendix 2.8.4) will assign an object to each text and image view that is present in item\_match.xml. Then ‘MatchActivity’ (see appendix 2.8.1) will access the matched user and check if they exist in the database. If they do, then the class will obtain all of the relevant information necessary to populate the objects specified in MatchViewHolders. ‘MatchObject’ (see appendix 2.8.3) will then populate the designated area on the matches screen with the objects gained from ‘MatchActivity’ using the same methods that were discussed in section 6.1. Lastly ‘MatchAdapter’ (see appendix 2.8.2) will add the information one at a time to the list on the matches screen, handling the number of users shown to the current user.

## Live chat

The live chat feature of the application I developed similarly to the matches package. It includes a ‘Messaging’ package (see appendix 2.8) with a ‘ChatActivity’ class (see appendix 2.9.1.), a ‘ChatAdapter’ class (see appendix 2.9.2), a ‘ChatObject’ class (see appendix 2.8.3) and a ‘ChatViewHolders’ class (see appendix 2.9.4). all representing ‘item\_chat.xml’ (see appendix 3.10).

The ‘item\_chat.xml’ file essentially contains the foundation of a list which is used as the basis of the messaging feature. The classes in the ‘messaging’ package update this activity the same way in which the ‘match’ package populates the ‘item\_match.xml’ activity (see section 6.4). However, there is one significant difference within this package.

There are two methods within the ‘ChatActivity’ class. The first is called ‘sendMessage()’ and allows for the user to send a message to the other user they want to talk to.

Figure 34 represents a snippet of code that takes the message that the user wants to sent to the other and saves it to the database under a node called ‘Chat’. This code takes the user id of the user sending the message (the current user) and the message that the user wants to send. Then the ‘ChatObject’ and ‘ChatViewHolders’ classes will populate the conversation between the two users with the messages as simple text views.

The next method introduced in the ‘ChatActivity’ class is ‘getChatMessages()’. This method, as shown in figure 35 is used whenever a user enters a conversation they have previously been having with another user. The method allows the application to search through the Chat database to get the previous messages sent between the users stored under ‘chat’ and retrieve them from the database to populate the conversation screen. The application is allowed to find the message sent which is assigned to ‘text’ and who sent it which is assigned to ‘createdByUser’. This information is then utilised by the ‘ChatAdapter’ to update the list of messages the user sees on their screen.

## Event creation/management

This feature exists within the chat package (see appendix 2.9.5). It was placed there as it is accessible to the user via the live chat feature, therefore is related. This single feature allows the user to create an event and save it to the user’s phone. It is linked to the ‘Activity\_calendar.xml’ file which consists of 3 text fields that the user must populate with information, a calendar view which takes the date chosen by the user and a button to save this information (see appendix 3.1).

As we can see in figure 36, the class collects the user information including the event title, description and location in as strings. However, the calendar view is broken up and taken as integers for the selected day, month and year. This is essentially to allow the application to understand the values individually instead a format that it cannot comprehend. 

Then a new intent is created to take the strings of information and populate them in the device calendar using ‘CalendarContract’. As well as this, the date is inserted into the device using the public class gorgian calendar. This code is all written within an on click listener so that when the save button is pressed, this information is all saved in the phone instead of the database. This is because the event is local information that the user will want to check up on and remind themselves is upcoming. If this information was saved in the database, it would be more complicated for the user to view. It is also easier for the user to just enter their device calendar which already exists, instead of having to enter the MatchUp application and then view events.

## User profile settings

This feature of the application has been developed similarly to the ‘RegistartionActivity’ class (see appendix 2.5). However, it is also able to change values that are already saved in the database as well as add a profile image.

As shown in figure 37 to the left. This activity is associated with the ‘Activity\_settigns.xml’ file (see appendix 3.8) which asks the user for a large sum of information. The application must then read in all this information. It is able to do this by creating a data snapshot of the database and finding all children which hold a value. It was then developed to use if statements to get the current value of the database node. For example, to replace the name of a user in the database, the application will check if the value is present. If it is then ‘map.get’ will be used to retrieve the current value to string and then ‘.setText’ will be used to populate the database with the new value as read in from the user as normal.

The next method is called ‘saveUserInformation()’ and is responsible for updating the database with the changes read in figure 37. As shown in figure 38, this method takes all the new information collected and converts it to sting. Then it is written to use ‘userInfo.put’ to change them in the database. Lastly ‘.updateChildren(userInfo) is to save the database information with the new data. This is only done if and when the user selects the ‘save’ button.

Lastly, the settings activity allows for the user to upload or change their profile image. Figure 39 shows that the code written will check the database to ensure that there is no current value for it. Then if the value does not equal null, the application goes into a switch statement. This is where it states that if the value of the ‘profileImageUrl’ is set to default, then a default image that has been created will be used for the profile image.

Whereas if the user chooses to change the image by selecting it, the code from figure 40 is called. This is where the user can select an image from their gallery for their profile image. If they choose to do so then a URL is generated for the image which is saved in the database. So that any time the users profile image needs to be seen, it is called upon and loaded from the database.

# Testing: Verification & Validation

This is a critical phase in the software project. It is where everything that has been developed in sections 5 and 6 are assessed to ensure that the developed solution meets requirements. The requirements specified in the problem articulation section will be used to test each component of the system along with a test plan to ensure the problem is solved.

## Verification

Verification of a project is done to ensure that the application is being developed in accordance to the specified technical requirements as devised in section 2.2. Throughout this project so far, each developmental stage has contributed to the verification of the application. Each stage counts as planning to ensure the needs of the project are met such as section 5 which specified how the application will be designed to meet the technical specification. Another example is in section 3 where the literature review informed the development of the project to ensure decisions being made would be only to support the application in satisfying the technical requirements.

The functionality testing conducted in this section of the report will assess the application to ensure that everything that has been done up until this point (from the problem articulation to implementation) have followed the path in supporting the requirements of the project and lead it to ultimate success.

## Validation

So that the application being developed is correct, tests are to be conducted in reference to the requirements created in the PID which were revised and officially stated in section 2.6. The application will be tested in terms of its usability. The results from which will be carefully analysed to ensure no further action is needed to meet the requirements, and the entire process of testing will be further discussed in section 8.

## Functionality testing

Each feature that the application possessed will be assessed against the technical specification (Stephens, 2015), devised in section 2.2. All requirements will be used as standards that the application must meet to pass the test. If the system is able to meet each requirement, the functionality of the project application will be deemed a success. However, if the application fails against any of the requirements then action must be taken to modify this.

| **ID** | **Requirement** | **Description** | **Expected result** | | **Actual result** | | **Necessary action** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Must have requirements** | | | | | | | |
| **1.1** | This project will produce an android application. | The product developed runs on an Android operating system | Application will run on any android mobile device without hesitation | | Application successfully deploys on android mobile devices | | None needed |
| **1.2** | The application will be able to run on a portable platform. | The application must be used on a mobile device for ease of use. So that it can be used on the go | The application will work on portable platform devices such as a mobile phone | | Application successfully deploys on portable mobile devices | | None needed |
| **1.3** | The application will pair the user with others that want to play the same sport as them. | The application will read the sport that the user is interested in from the DB and match the user with others who have the same sport listed | When the user logs on to the system, they will be shown other users who have the same sport interest listed in their database | | The user is presented with any users registered on the database with the same sport listed in their information in the form of a swipe card | | None needed |
| **1.4** | The application will allow the users to create an event that will be saved in their phone calendar | The application will be tested to ensure that the event creation feature exists, and successfully saves the event to a user’s calendar | When an event is created and saved, it will be seen in the user’s device calendar | | The event that is created within the MatchUp application can be seen in the device calendar | | None needed |
| **1.5** | The application will allow users to communicate with each other | Test to ensure that the messaging feature successfully sends messages to the correct user in real time without hesitation | A user will send a message to another and it will be almost immediately received in the format that it was sent in | | Outcome is as expected where the message is received instantaneously in a format that matches the message that was sent | | None needed |
| **1.6** | The application will make use of a database to save all user data | Each user registers with the application, and the information should be submitted to the Firebase database and correctly saved under their own user ID | When the user registers, the database node ‘Users’ will be updated with the new user information | | The database node ‘Users’ is updated with the submitted information, including Name, Profile image, bio, gender and preferred sport | | None needed |
| **1.7** | The application will allow for the user to change the chosen sport | A settings page has been implemented to allow users to change the sport they have chosen. This must be tested to ensure that when changes, new corresponding people are shown | When the user selects a new sport and saves this, different users who have the same sport chosen will be shown to them as opposed to those interested in different sports | | As the user submits the ne selection, the database node ‘sport’ is updated. Therefore, the code analyses the new update in the database and shows the user people also interested in their newly chosen sport | | None needed |
| **Should have requirements** | | | | | | | |
| **2.1** | The application will utilise a GUI that is comfortable for the user | Check section 7.4 for the result of this requirement being met. (successful) | | | | | |
| **2.2** | The application possesses user profiles | User profiles are implemented in two ways. The profile that the user does not see but other people do (a), and within the settings page where the user information can be edited (b) | **(a)** The user profile that other users see will contain the user name, image and bio | | **(a)** The user name, image and bio are seen by other users on the system successfully | | None needed |
| **(b)** Correct information will be displayed to the user when they access their settings page | | **(b)** Correct information including the user profile image, name, bio and selected sport are displayed in the settings page | | None needed |
| **2.3** | The application will allow image sharing | The application allows the user to display an image in some shape or form. | The user will be able to upload a profile image to their profile via the settings page which can be seen by all other users of the application. | | The profile image that is uploaded to the user’s profile can be seen by others whilst searching through other users and can be changed in the settings page | | None needed |
| **2.4** | The application will pair users with others within a specified area radius | A location-based search will be implemented to match users with others in their local area. This must be tested to ensure it is successful | The users that are displayed will be a local specified radius, chosen by the current user of the application | | This requirement has not been met as this feature was not implemented in the application. | | Discussed in section 6 |
| **Could have requirements** | | | | | | | |
| **3.1** | The application will allow for the user to edit the personal information which other users can see. | A settings page has been implemented to allow users to update their personal information. This must be tested to ensure the database (a) and what other users see (b) is updated | **(a)**When the user submits any changes within the settings page, the new data will be updated in the database | **(a)** The data under the ‘user; node is changed to the new data that was input in the settings page | | None needed | |
| **(b)** The data displayed to other users on the system is updated will be changed. So that other users will see the new data | **(b)** The data has changes and the information that the other users see is new | | None needed | |

## Usability testing

This testing is necessary to determine whether or not the application is easy to use for the user (Stephens, 2015). The application will be assessed against the user acceptance criteria created in section 2.6. Should the application meet all the requirements specified, the project is classed as a success in this aspect. However, if any of the requirements are not met, modifications will be made to ensure that any problems are rectified.

| **ID** | **Requirement** | **Description** | **Expected result** | **Actual result** | **Necessary action** |
| --- | --- | --- | --- | --- | --- |
| **1** | The user can run the application on their mobile device | The user should have no problem opening the application on their mobile device | User will select the application from their menu and it will automatically deploy | Application deploys without hesitation | None needed |
| **2** | The user can choose a sport they are interested in | The user will select a sport when they register to the system. This must be clear and easy to do | User will have the option to select a sport | User can select a sport they are most interested in | None needed |
| **3** | The user can see other users on the system that want to play the same sport | When the user logs on to the application, they must be presented with other users who are also registered and interested in the same topic. | User will be presented with a stack of other users who are interested in the same sport | Outcome as expected, user can see several others to match with | None needed |
| **4** | The use can upload an image to their profile | The system allows for users to have profile pictures. The user must be able to upload their picture somewhere | The user will have the ability to upload a profile picture to their user profile via the settings page. | User is able to easily change their profile image | None needed |
| **5** | The user can change their personal information | The system requires certain information from the user. If at any point this information changes, the user must be able to update this information some how | The user will have the ability to change their information on their user profile via the settings page | User can easily access the settings page and effortlessly change information | None needed |
| **6** | The user can change their chosen sport | In the case that the user wants to participate in a different sport, they need to be presented with different users who are interested in the new sport. There must be a feature to allow the user to update this data | The user will have the ability to change the sport that they previously selected via the settings page | User can pick a new sport via the settings page from a list of the available sports to be paired with on the system | None needed |
| **7** | The user can communicate with other users | There must be a feature to allow users to communicate in some form. This must be easy for the user | Communication will be easily accessible when a user has matched with another | Communication is effortless and in real time | None needed |
| **8** | The user can create events | The user needs a way to initiate planning for an event. This must be done in a way that is easy to understand. | Event creation will be easily accessible via the chat pages when the users have matched | Event creation is seamless, asking for only brief details so as not to confuse the user | None needed |

## Testing limitations

It is worth taking into consideration that there were some conditional restraints that meant that the entire application could not be fully tested to investigate each and every problem that exists within the solution.

The first limitation to this testing process is that all tests were conducted on a single android mobile phone device that contained android lollipop. Though this is a newer version of the operating system, it is highly likely that the potential users of the application are likely to use hardware devices that contain different versions of the android operating system. Therefore, it is very possible that the application would perform differently under these conditions.

Another limitation for this process was the restricted time limit. This prevented the Developer from going into much more depth whilst testing the abilities of the application. Therefore, some problems have been left unexplored. As well as this, had there been more time left in the budget for the project development, the application could have largely benefitted from beta testing.

Beta testing is the external pilot test of an application. Completed after any form of testing against specified requirements. Beta testing requires real life potential users of the application to test the application by using it to complete a set of specified tasks. The users will specify how they were able to carry out these tasks, defining the level of ease or complication along with other feedback (Dolan and Matthews, 1993). This form of testing would have been highly beneficial as certain problems that the Developer would have missed could have been pointed out. It would also provide real feedback on the application and ways in which it could have been improved. This form of testing would have ensured that the level of quality and usability required by the users was truly being met and if not, the beta testers would have specified what they would prefer. As well as this, any bugs left in the system could have been discovered much faster and more easily.

These limitations have had a significant impact on the testing process however did not stop testing from happening overall. Therefore, an appropriate amount of testing has occurred to validate the significance of the application. Though the project may have benefitted more had these limitations been surpassed.

# Discussion

After the events of section 7, the application has been fully developed and assessed. Therefore, the outcomes of the testing processes can be discussed. This is important as and problems that occurred in the testing phase must be debated and rectified. It also gives an opportunity to reflect on the decisions made throughout the duration of the project and understand what could have been done better or what has been done well.

## Results of functionality testing

Functionality testing was necessary to the project to ensure the applications functionality could be confirmed to fulfil the requirements as specified in section 2.2. As described by the table below, a majority of the requirements that the application had been tested against had been fully met. However, there was a single requirement that was not.

The requirements had been grouped into certain groups as described by ‘must have’, ‘should have’ and ‘could have’. This was considered necessary as it allowed to provide a level of priority to each requirement, making it clearer to understand which requirements needed to be met so that the project could be deemed a success. Ideally, every single requirement should be met. However due to the restrictions as specified in section 2.5, it was foreseen that there was a possibility that not all of the requirements could be met. Therefore, the level of priority assigned to each requirement meant that even if low or medium level requirements were not met, and each of the high-level requirements were met, the project could still be classed as a success.

| **Test ID** | **Priority** | **Requirement met?** |
| --- | --- | --- |
| **1.1** | High | Yes |
| **1.2** | High | Yes |
| **1.3** | High | Yes |
| **1.4** | High | Yes |
| **1.5** | High | Yes |
| **1.6** | High | Yes |
| **1.7** | High | Yes |
| **2.1** | Medium | Yes |
| **2.2** | Medium | Yes |
| **2.3** | Medium | Yes |
| **2.4** | Medium | No |
| **3.1** | Low | Yes |

In terms of the low-level priority requirements, there was only one (test ID 3.1). This requirement had been fully satisfied. Though this was only a low-level requirement, it is still beneficial that that it has been met as this means that the application is more efficient in satisfying the problem than if the requirement was not met.

In reference to the medium-level priority requirements, over 75% of them have been satisfied. This supports the application in solving the problem a significant amount. However, one requirement (test ID 2.4) was not met. This was the requirement to include a location-based search to ensure users met those who were in proximity to them. This would have been a useful feature as it meant it was easier to meet people who were closer in location, removing the issue of extensive travel. Unfortunately (as discussed in section 6.5) this feature was unable to be implemented due to the lack of knowledge possessed by the Developer, paired with the restricted time limit preventing them from the opportunity to research this concept more. Though this requirement was not met, because it is only a medium-level one the project is still on track to being deemed a success. However, it is still understood that this feature would have been beneficial to the application and will further be discussed in section 11.

With regards to the high-level priority requirements, as displayed by the table above, each requirement was shown to be satisfied when tested on the application. There was no need for any further action or modification as the application was able to fulfil the standards and meet requirements. This is satisfactory in terms of the project as it implies that the entire project is en-route to be a success. Especially considering these requirements were classed as high-level priority, and a necessity for the application to be successful.

## Results of usability testing

Usability testing is extremely crucial in the development of a project. It is necessary that the application fully satisfies the requirements of the users to be deemed a success. This method of testing ensures that the user will be fully satisfied with the application when they come to use it because it assesses the application against the needs of the user as specified in section 2.6.

| **Test ID** | **Priority** | **Requirement met?** |
| --- | --- | --- |
| **1** | High | Yes |
| **2** | High | Yes |
| **3** | High | Yes |
| **4** | High | Yes |
| **5** | High | Yes |
| **6** | High | Yes |
| **7** | High | Yes |
| **8** | High | Yes |

After conducting usability testing it is clear to see that each of the requirements assessed in section 7.4 have been effectively satisfied, as displayed by the table to the left. This proves that the application as a success in relevance to the user acceptance criteria. The standards of the user have been optimally met and the projects even more on track to be a success. In fact, the results of this testing technique paired with that of the results of functionality testing prove that according to almost all requirements created at the start of this project, whilst also considering the constraints considered in section 2.5, this project can quite confidently be deemed a success.

In summary, the usability testing technique was beneficial to the project in proving that the development of the application and project as a whole has been successful.

## Hindsight

After analysing the results of testing the application, some observations can be made that if considered beforehand would have been valuable to include in the development of this project. However, realising them now is still beneficial as they can be considered the next time a project is underway.

One aspect that would have been beneficial as discussed in section 7.5, would have been beta testing. This would have allowed a better understanding of how well the standards of the potential users of the system would have been satisfied. This would have led to a much more accurate debugging process as their separate users could have realised problems that a single developer on their own may have overlooked.

As well as this, more appropriate and efficient allocation of time would have been valuable as it would have meant that the Developer would have had more time to learn and understand the technical requirements to implement a location-based search within the application. Though this did not prevent the project from becoming a success as it was a medium-level priority requirement, including this feature would have satisfied the problem more than it currently does. Therefore, this could have been useful.

Looking back on how the requirements of the project have been assessed, it is now recognised that breaking down each specific requirement into more detailed conditions could have been useful. For example, taking one requirement such as ‘The user can choose a sport they are interested in’ and breaking this down into more detailed conditions such as;

* The user has a list of different sports to choose from.
* The sports that are listed are a compilation of the world's most popular, to ensure there is an option to statistically satisfy any user.
* The user can select one of the listed sports.
* The list is clear and visible with no complication.

This would have allowed for a more in-depth analysis when testing each feature. Therefore, resulting in more problems being addressed so that the application would be much more effective in solving the overall project problem.

# Reflection

Throughout the course of this project, there have been many experiences that developed my skills and prompted much personal growth over the past academic year. In the initial stage of proposing a PID, eagerness and excitement clouded the reality, being that my lack in any technical knowledge would make this project a significant amount harder than ever imagined. However, instead of withdrawing the proposal, I felt a challenge rise (as well as the feeling of fear and nervousness). Changing the project topic to one that was considered ‘easier’ for a student with an Information Technology background (as opposed to computer science) would benefit no one. A potentially easier grade was likely however not guaranteed. So, the original proposal was sustained, and I began research on application development.

Beginning the final year project was daunting, leaving me nervous to start when the academic year rolled around. Especially with my lack of knowledge in programming with only basic understanding of java. I found myself constantly scared to start and avoiding the programming aspect altogether, focussing on the design and planning of the project for a long time. This obviously begun to skew my Gantt chart that was devised in the PID form before the project begun. I was starting to lose time due to irrational fears. It wasn’t until my fifth meeting with my supervisor that I began to code. He was quick to pick up on my fear and reassured me that he was here to help and that I needed to start something before it could fail. So, after that meeting I took my first step at attempting to program.

I believe that being assigned supervisors was an extremely useful aspect of the project. Not only for academic help but also for moral support as Fred was able to understand that firstly, I was not the strongest programmer and secondly that it was hard to fit final project work around current assignments whilst still pushing me to work to the best of my ability. Therefore, making meetings feel like a more comfortable environment where I could ask for academic and moral support and constantly be provided with it. I really valued the help of a project supervisor throughout this project.

Programming the application took much trial and error at many points, as discussed in section 6. There were times where I would be stuck for days on the same problem. This lead me to appreciate the amount of resources available through the University of Reading. From the staff, to past projects available allowing me to gain inspiration. As well as this, I also realised the importance of using the internet when stuck on a programming problem. More specifically the website StackOverflow.com. Where I was able to communicate with people who were having or have had the same problems as me. Who were happily ready to provide aid and support.

The biggest problem I encountered when developing the application was a technical issue with part of the code. I created a method to allow a user to choose which sports that were interested. These would be collected as strings and saved in the database. Then when the user logged into their account, other users who had chosen the same sport would be displayed to them to allow them to communicate. However, I was unable to query the database to return the chosen sports saved in the database. After some digging I was able to find a solution on the popular programming forum Stack-overflow. I learnt that the Firebase database was unable to allow for nested queries. Therefore, I restructured the database and the code to solve this problem. This resulted in the user only being able to pick one sport they were interested in when they signed up to the application, however allowed them to change the chosen sport in the application settings. This method showed to be much more beneficial as showing users with all different types of interest would be confusing and overwhelming. Whereas showing one at a time with the option to change is much more comfortable and less cluttered.

Through facing and solving this problem, I was able to develop my understanding of Java, Android Studio and Firebase as well as structure my system in a much more user-friendly way. I enjoyed working through this as solving the problem gave me a sense of achievement and satisfaction. And though this project is coming to an end and I have completed the programming necessary for the application. I still find myself trying to improve or update the code and finding ways to improve the application. For example, I am still interested in implementing a location-based search and plan to continue attempting to program this into the current application.

Another aspect of the project that I appreciated was the requirement of a logbook that had to be signed off each meeting to ensure good time and project management was being kept. This also helped me organise my thoughts and ideas more efficiently as they were all kept in one place, where I could easily look back on them. It also meant meetings were much easier to keep track of as notes were made each time in the log book. This definitely helped develop my time management and project management skills which will be very useful when I am working in a career that values these skills.

Though I did enjoy my experience in working on this final year project, one thing that I did not enjoy was the amount of assignments due that did not count to the final grade of the project. For example, I understand that many of these minor assignments were to ensure out project and time management skills were being utilised. However, assignments such as the project poster could easily have contributed to 5% of the final grade and would have taken much of the pressure off. Instead of having a heavier workload that did not count for anything, I would have been much happier completing these assignments knowing they were contributing to something, instead of being an extra burden for no real reason.

As well as this, I found it slightly unfair that most of the staff proposed projects were tailed more towards CS students. It is understandable that CS students take up a majority of the department, however IT students do not have as much technical knowledge as CS, nor do they get the option to do as many technical subjects when choosing modules. Therefore, it was slightly more of a struggle keeping up with the final year project. However, if there were more IT related projects, the playing field would be more level for IT students.

Aside from this, I found the final year project very beneficial to my final year at university as it allowed me to gain slight experience in working in a career where I would have to manage a project or assignment individually. It has strengthened my technical and analytical skills as well as improved my ability to manage time. I can definitely say that this project largely supported my development to become a competent and confident Information Technology student.

# Social, legal, health & safety and ethical implications

One major social and safety issue that is raised by the development of this project is the safety of the users when they go to meet the people that they interact with in the application. Because of the nature of the application, it is similar to that or a dating application. Therefore, some users may arrange to meet others with the intent of harming or abusing them. This is quite obviously also a major health issue as it could result in someone being badly hurt.

An ethical issue raised is that some users may be using this application in place of a dating app. This first of all is not the intended purpose of the application, which is in fact to promote healthy social interaction and exercise. And secondly, this brings a bad stigma about the application which demotes its entire purpose and intended positive impact.

As well as this, another social issue caused with the creation of this application is that of ‘stranger danger’. Many people may be uncomfortable with the idea that the application is matching them with strangers. this is seen as dangerous to many people and they are therefore less likely to use the application. Therefore, something should be implemented to ensure that users are being monitored and they are safe to be using the application. This will be discussed in section 11, when debating further improvements for the application.

In terms of health, there is one positive implication associated and that is that the application can be seen to promote exercise and a healthy lifestyle. As well as this, the application also promotes social interaction and communication. Therefore, raising a positive social implication. Exercise and social interaction are both something that many people see as healthy and necessary for a happy lifestyle. Therefore, the application is benefitting the users in that aspect.

There are no negative legal implications associated with the development or existence of this application.

# Future Improvements

Though the development of this project is coming to an end, technology is constantly evolving and changing, bringing more potential to aid society in improving itself. Therefore, the potential that the developed solution has to improve in the future cannot be ignored and will now be discussed.

As mentioned in section 10, something must be implemented into the application to ensure that users are safe to meet these strangers that they interact with on the application. One way in which this can be supported is by including the option to block and report users. Therefore, any users who pose any form of threat or discomfort can be flagged to the moderators or developers of the application. The user will then be immediately investigated and removed from the application. These improvements are all features that should have been implemented sooner had there been more time and a higher level of competence in terms of the developer.

User rating are another way of ensuring safety for the users of the application. Each user is able to rate another user when matched with them on the application. The user being rated does not know their rating however, everyone else on the application can see it. This will indicate that lower rated users are to be avoided to promote safety. And in the case that a user has a rating that is lower than a certain amount, they are investigated by the developer of the application. As well as this, user ratings allow users to find others whose sporting ability is matched to theirs. Allowing them to find a more appropriate match to play a sport with. This allows for a more satisfying game where players are more accurately matched in terms of ability.

Another useful improvement that could be implemented in the future would be group chats. This would allow for better organisation of group sports such as football or basketball. Group chats will allow up to a certain number of users to be added to the chat. So that there is more effective communication between all of the people trying to plan an event. This is more efficient than two people trying to organise a group of 11 when the original two people are the only ones who could communicate with each other.

Invitations are another feature that could be beneficial to this system. For example, instead of communicating with a matched user that they are planning an event, and both having to individually put it in their calendar. One of the users will create an event and send an invite to the other. Once the invite is accepted, the information is automatically input into the user’s device calendar. This makes the event planning feature much easier for all users.

One last major improvement that would be highly beneficial to users of the application would be the integration of an activity tracker. This would work in two ways. Either the user keeps their phone on their person when participating in a sport and the sensors on the phone take in information whilst they are exercising, analysing their heart rate, amount of time they are active, etc. which they can view after they are finished.  
Or the user is able to pair an activity tracker device to the application which connects via Bluetooth. So, the activity tracker is reading the data from the user and then sending it to the phone application for the user to view.   
This improvement is highly in theme with the nature of the application and would benefit the users greatly as they are able to have fun exercising and see the results of them doing so afterwards.

# Conclusion

The research and development conducted within this project has been deemed a success by the ‘Must have’ technical requirements as well as the user acceptance requirements as specified in section 2. A functioning android application has been created that works for its intended purpose.

MatchUp is a product of this report that was created from nothing, now serving as a social interaction and event management aid. This is due to the organised development of this project, where each stage was necessary in the lead up to the result. Starting with with the problem articulation in section 2, where the problem at hand was described and discussed. And all requirements were devised. Moving on to the literature review where all related content and literature were analysed to ensure that from then onwards, informed decisions were made. The solutions approach was where everything that was learnt in section 3 was utilised to propose 2 potential solutions which were debated and assessed against requirements to choose the most appropriate one. The design stage was where the output from section 4 was used to plan out the entire application which was then created in section 6, Implementation. At this stage, the process of creation was tracked and monitored, discussing each success and failure of the project so far. Testing was the last major stage of technical development where the developed application was analysed and assessed against the technical specification and user acceptance criteria to be deemed a success.

Though the location-based search was not a success due to the restricted time limit and the fact that the Developer was unable to implement this feature because of their lack of knowledge of the subsect. The project can still be classed as a success as this was not one of the ‘Must have’ requirements. Therefore, it was not a high-level priority feature. However, this feature has been discussed in section 11 and there are plans to implement it soon.

After all the technical development of the project, there were still discussions about the progression of the project and its findings, reflections on the experience as well as debates on the social, legal, ethical and health & safety implications of the project. Ending with deliberation about the future and improvements of the application and how they would benefit the users.

As a result of this project, must development has been made in terms of ability and understanding of mobile application development, GUI design, Java programming language and project management. Matchup is a completed application that meets its intended purpose, however shows much room for further development and refinement as stated in section 11. Though, currently MatchUp is a useful application in an easily accessible and usable form that promotes many positive implications health and social wise (section 10). With much potential to benefit many users.

# References

**Carroll, J. (2013).** Human Computer Interaction - brief intro. [online] The Interaction Design Foundation. Available at: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro [Accessed 9 Mar. 2018].

**Chau, M., Reith, R. and Nagamine, K. (2018).** IDC: Smartphone OS Market Share. [online] IDC: The premier global market intelligence company. Available at: https://www.idc.com/promo/smartphone-market-share/os [Accessed 9 Mar. 2018].

**Cinar, O. (2012).** Android Apps with Eclipse. Dordrecht: Springer.

**Cleveroad Inc. - Web & App development company. (2018).** Why Choose Firebase for Your Business. [online] Available at: https://www.cleveroad.com/blog/a-story-of-firebase-or-your-next-favourite-cloud-based-service [Accessed 9 Mar. 2018].

**creativethoughts (2018).** Benefits of android app development for your business. [online] Slideshare.net. Available at: https://www.slideshare.net/creativethoughts/benefits-of-android-app-development-for-your-business-86511397?qid=41cf7a02-a8e7-424e-a713-12235b3b38e0&v=&b=&from\_search=1 [Accessed 9 Mar. 2018].

**David, M. (2015).** Visual Studio IDE offers many advantages for developers. [online] SearchSoftwareQuality. Available at: https://searchsoftwarequality.techtarget.com/feature/Visual-Studio-IDE-offers-many-advantages-for-developers [Accessed 9 Mar. 2018].

**Dolan, R. and Matthews, J. (1993).** Maximizing the Utility of Customer Product Testing: Beta Test Design and Management. Journal of Product Innovation Management, 10(4), pp.318-330.

**Dunlop, M. and Brewster, S. (2002).** The Challenge of Mobile Devices for Human Computer Interaction. Personal and Ubiquitous Computing, [online] 6(4), pp.235-236. Available at: http://delivery.acm.org/10.1145/600000/591427/20060235.pdf?ip=134.225.31.109&id=591427&acc=ACTIVE%20SERVICE&key=3B621857D79A46E2%2E3B621857D79A46E2%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&\_\_acm\_\_=1523295177\_60cb416291476b0fde79b0c8f001fb31.

**Eventbrite.co.uk. (2018).** Event Registration Online | Eventbrite. [online] Available at: https://www.eventbrite.co.uk/l/registration-online-b/?ref=sem0brd0ggl0uk0ppca0eventbrite&kw=event%20brite%20ex&cr=245948710063&mkwid=sApuuzLox\_dc&pcrid=245948710063&pkw=event%20brite&pmt=e&plc=&\_bk=event%20brite&\_bt=245948710063&\_bm=e&\_bn=g&gclid=Cj0KCQjwnqzWBRC\_ARIsABSMVTOWPDbOFdi0mgyrCKOi209FTd5ZXQeOQKwKMk-M9d4uXNu5AAqvMDsaAhleEALw\_wcB [Accessed 9 Mar. 2018].

**Gribble, P. (2012).** Why Program in C?. [online] Gribblelab.org. Available at: https://www.gribblelab.org/CBootCamp/1\_Why\_Program\_In\_C.html [Accessed 9 Mar. 2018].

**Kautiainen, S., Koivusilta, L., Lintonen, T., Virtanen, S. and Rimpelä, A. (2005).** Use of information and communication technology and prevalence of overweight and obesity among adolescents. International Journal of Obesity, [online] 29(8), pp.925-933. Available at: https://www.nature.com/articles/0802994.pdf [Accessed 6 Mar. 2018].

**Kostrov, S. (2012).** What is the best programming language for mobile apps?. [online] Software.intel.com. Available at: https://software.intel.com/en-us/forums/mobile-software-development/topic/288140 [Accessed 9 Mar. 2018].

**Kozlov, S. (2017).** When and Why Java is Suitable for Your Application | Romexsoft. [online] Romexsoft. Available at: https://www.romexsoft.com/blog/why-java/ [Accessed 9 Mar. 2018].

**Krill, P. (2017).** The most popular IDEs? Visual Studio and Eclipse. [online] InfoWorld. Available at: https://www.infoworld.com/article/3217008/development-tools/the-most-popular-ides-visual-studio-and-eclipse.html [Accessed 9 Mar. 2018].

**Lorenzo, D. (2016).** Swipecards. [online] GitHub. Available at: https://github.com/Diolor/Swipecards [Accessed 12 Mar. 2018].

**Mamo, A. (2018).** How to get value from Radiogroup and change in database. [online] Stackoverflow.com. Available at: https://stackoverflow.com/questions/49308241/how-to-get-value-from-radiogroup-and-change-in-database [Accessed 12 Mar. 2018].

**Msdn.microsoft.com. (2015).** Cross-Platform Mobile Development in Visual Studio. [online] Available at: https://msdn.microsoft.com/en-gb/library/dn771552.aspx [Accessed 9 Mar. 2018].

**Rajput, M. (2015).** Why Android Studio Is Better For Android Developers Instead Of Eclipse - DZone Mobile. [online] dzone.com. Available at: https://dzone.com/articles/why-android-studio-better [Accessed 9 Mar. 2018].

**Rock, M. (2018).** Is digital technology destroying human interaction? - Real Business. [online] Real Business. Available at: https://realbusiness.co.uk/tech-and-innovation/2013/02/04/is-digital-technology-destroying-human-interaction/ [Accessed 9 Mar. 2018].

**Snafflz.com. (2018).** #1 Event Check-in Guest List Manager. [online] Available at: https://snafflz.com/ [Accessed 9 Mar. 2018].

**Sporty. (2018).** Play Sports with People Nearby. [online] Available at: https://www.sportyapp.com [Accessed 9 Mar. 2018].

**Stephens, R. (2015).** Beginning software engineering. Indianapolis, Indiana: Wrox, pp.173 - 202.

**Teamer.net. (2018).** Teamer - The Free Sports Team Management App. [online] Available at: https://teamer.net/ [Accessed 9 Mar. 2018].

**Tinder. (2018).** Tinder | Swipe. Match. Chat.. [online] Available at: https://tinder.com/?lang=en-GB [Accessed 9 Mar. 2018].

**Webster, L. (2018).** Mobile App Development | Visual Studio. [online] Visual Studio. Available at: https://www.visualstudio.com/vs/features/mobile-app-development/ [Accessed 9 Mar. 2018].

# Appendices

## Appendix 1 – Project Initiation Document

**Individual Project (CS3IP16)**

**Department of Computer Science**

**University of Reading**

**Project Initiation Document**

PID Sign-Off

| **Student No.** | **24001764** |
| --- | --- |
| **Student Name** | **Zamena Jaffer** |
| **Email** | **Wk001764@student.reading.ac.uk** |
| **Degree programme** (BSc CS/BSc IT) | **BSc IT** |
|  |  |
| **Supervisor Name** | **Dr. Frederic Stahl** |
| **Supervisor Signature** |  |
| **Date** | **29/09/2017** |

SECTION 1 – General Information

Project Identification

| **1.1** | **Project ID**  (as in handbook) |
| --- | --- |
|  | 100 |
| **1.2** | **Project Title** |
|  | Event Management Assistant |
| **1.3** | **Briefly describe the main purpose of the project in no more than 25 words** |
|  | To develop an app that helps users find people in their local area who want to play the same sport, connect and meet to play. |

Student Identification

| **1.4** | **Student Name(s), Course, Email address(s)**  e.g. Anne Other, BSc CS, a.other@student.reading.ac.uk |
| --- | --- |
|  | Zamena, BSc IT, Wk001764@student.reading.ac.uk |

Supervisor Identification

| **1.5** | **Primary Supervisor Name, Email address**  e.g. Prof Anne Other, a.other@reading.ac.uk |
| --- | --- |
|  | Frederic Stahl, f.t.stahl@reading.ac.uk |
| **1.6** | **Secondary Supervisor Name, Email address**  Only fill in this section if a secondary supervisor has been assigned to your project |
|  |  |

Company Partner (only complete if there is a company involved)

| **1.7** | **Company Name** |
| --- | --- |
|  |  |
| **1.8** | **Company Address** |
|  |  |
| **1.9** | **Name, email and phone number of Company Supervisor or Primary Contact** |
|  |  |

SECTION 2 – Project Description

| **2.1** | **Summarise the background research for the project in about 400 words. You must include references in this section but don’t count them in the word count.** |
| --- | --- |
|  | When initially considering a sports based event management app, I began by researching what applications already exist on the market currently and assessing the need for the system. I found that the most popular related apps were such that allowed sports clubs and teams to organise when they would plan to meet next. For example, Teamer is a web and mobile application that allows an already existing team or club to communicate and organise when they would meet to play. The team would have a profile that all members could access to view updates and images when desired and there was also a feature to allow users to donate to the sports club [1]. This application focuses solely on already existing teams and clubs, whereas I am considering an app that allows any individual to find other people to play with. Other software’s such as ActiveNetwork only allowed an already existing to establish a network between members to communicate and recruit more members via the network. This network focusses more on the admin aspect of the event management as it holds features such as merchandising, financial management and analysis of the team’s stats [2]. This network bases itself around the inner workings of a sports team. I could use Active Network as a template when creating my app, however the product I aim to make also allows a social feature where users can find other users in their local area and connect to plan where and when to meet.  I also researched the top general event management apps using the following site [3], it compared applications such as:  Eventbrite- This application contains information on currently occurring events in the user’s local area, allowing them to filter the results in terms of a radius of distance. Holding information such as when, where and includes a feature to allow users to buy tickets. I could consider the feature of displaying currently occurring events when developing my project further.  Snafflz- This app focuses more on the social aspect where users can check into an event and leave reviews and recommendations. This feature could be something to take forward, for example, users being able to review each other.  Bizzabo- This app has many good reviews on the user interface. It uses a very minimalistic design leaving little room for confusion. The design is something to be highly considered as this is the aspect that the user interacts with most, therefore it should be something that they can easily work with.  Taking this all into consideration, I found that there aren’t many applications similar to the one I would like to develop.  **References:**  "Teamerrails5." Teamer.net. N.p., 2017. Web. 3 July 2017.  "Sports Software - Online Sports Management Software, Entry System And Registration Software By Active Network." Activenetwork.co.uk. N.p., 2017. Web. 4 July 2017.  "Best Event Management Software - 2017 Reviews & Pricing." Softwareadvice.com. N.p., 2017. Web. 3 July 2017. |
| **2.2** | **Summarise the project objectives and outputs in about 400 words.** These objectives and outputs should appear as tasks, milestones and deliverables in your project plan. In general, an objective is something you can do and an output is something you produce – one leads to the other.  Objective -> Outpot |
|  | One objective is to allow users to find people to play a given sport with. The output of this objective would be that each user has their own personalised profile that will display which sports grounds they are closest to, what sorts they enjoy as well as other details about the user.  Another objective is to implement a social matching algorithm and the output of this would be that the app will match users based on the sport they want to play and how close they are from each other as well as several other factors determined by the user’s profiles.  Planning a sport event is a main objective of this project therefore the desired output for this would be that the application will allow users to communicate with a basic messaging platform so that they are able to plan a time and place to meet. As well as this, the objective will also produce the output is that the app will sync to the user’s calendar on their phone (with user permission) as a reminder of the event.  Another objective is to find users in the local area therefore the output of this would be that the app will use locational services to place the user and find other users close to them. However, this will not give away any personal details of any users (e.g. address).  Improve the users playing experience is one objective. This is possible if the user is playing with someone who is equally matched and provides a challenge. Therefor the output would be matching users based on their ability which will be provided by the user when creating their profile.  One related objected for this sport based event management app would be informing the user of currently occurring sports based events in their local area. The output would be a feature within the app that displays all events within a set radius of the user’s location. This feature would list all current events that are going to happen within the upcoming week with information such as when, where, and how to join.  A last objective would be to inform the user of when someone would like to connect with them. Therefore, the output would be to allow push notifications on the user’s device. |
| **2.3** | **Initial project specification - list key features and functions of your finished project.** Remember that a specification should not usually propose the solution. For example, your project may require open source datasets so add that to the specification but don’t state how that data-link will be achieved – that comes later. |
|  | **Must have requirements**  Social Matching Algorithm- Allowing users to find others on the app to play a desired sport with  Communication- There will be an inbuilt messaging system which will allow users to contact each other. This will also make use of push notifications on the user’s device.  Database- A database containing information on any area or centre that offers the facilities to allow users to play a sport including their location, facilities, amenities and availability will be linked to the app.  Event creation- The app will allow the users to plan the event and it will be shown in their calendars and they will be reminded of the event.  **Should have requirements**  User Rating- Each user can be rated after the event has been played to allow them to build up their profile and find more appropriate matched for them specifically in the future.  User friendly- The user interface of the application will be easy to navigate and will be usable by people of all abilities.  Individual profiles- Each user has a profile they can personalise so that the sorting algorithm can use their details to find them matches to play with.  **Could have requirements**  Fitness Tracker feature- if there is enough time within the project timeline, it may be useful to include a fitness tracker, in theme with the app so that the user can view and track their progress within the app.  Image sharing- The profiles could have a space to share photos of the user playing sports so that it is easier for other users to identify them. |
| **2.4** | **Describe the social, legal and ethical issues that apply to your project. Does your project require ethical approval?** |
|  | One ethical issue posed by the proposal of this project is that the final product could easily be used in place of a dating app such as Tinder or Plenty of Fish. Though this issue cannot be completely avoided, it can be restricted by limiting the amount of personal information displayed on a user profiles as well as having a report and/or block user feature available within the app.  As well as this, to avoid certain ethical issues, dummy data will be used instead of real user data when testing so that no test users are at risk in terms of data protection. As well as this, an evaluation plan will be implemented after testing. This will specify the methods of evaluation of the product, for example:  A group of users who differ in ability will use the product for a set amount of time to complete certain activities, and then fill out a questioner giving feedback on their experience.  Self-evaluation will be completed where the creator will go through and assess the app compared to the initial requirements and state how it has met or fallen short of the specification.  In terms of social issues, the project will be developed considering users of all different abilities so that the product can be accessible and understandable by all users so that no one is limited. The application will be adjustable within phone setting so that features such as brightness and text size can be altered to be comfortable for users with any visual impairment.  The project will abide by the Data protection act (1998) to ensure personal data is protected and only shared rightfully. As well as all other laws and acts in relation to data and technology so that the product is made within the requirements of the law and causes no issues. |
| **2.5** | **Identify and lists the items you expect to need to purchase for your project. Specify the cost (include VAT and shipping if known) of each item as well as the supplier.** e.g. item 1 name, supplier, cost |
|  | None necessary. |
| **2.6** | **State whether you need access to specific resources within the department or the University e.g. special devices and workshop** |
|  | None necessary. |

SECTION 3 – Project Plan

| **3.1** | **Project Plan**  Split your project work into sections/categories/phases and add tasks for each of these sections. It is likely that the high-level objectives you identified in section 2.2 become sections here. The outputs from section 2.2 should appear in the Outputs column here. Remember to include tasks for your project presentation, project demos, producing your poster, and writing up your report. | | |
| --- | --- | --- | --- |
|  | | | |
| **Task No.** | **Task description** | **Effort**  **(weeks)** | **Outputs** |
| **1** | **Background Research** |  |  |
| 1.1 | Project idea development | 3 | A project |
| 1.2 | Market research | 2 | Idea of market |
| 1.3 | Project management planning | 1 | Idea of project plan |
| 1.4 | Risk Analysis | 1 | Idea of risks |
| 1.5 | Collect stagnant information for database | 1 | Datasets |
| 1.6 | Stakeholder interviews | 1 | Idea of basic requirements |
| **2** | **Analysis and design** |  |  |
| 2.1 | Requirement analysis | 2 | Requirements |
| 2.2 | Design app | 3 | App design |
| 2.3 | Design database | 3 | Database plan |
| **3** | **Develop prototype** |  |  |
| 3.1 | Develop database | 4 | Functional database |
| 3.2 | Develop app | 8 | Functional app |
| **4** | **Testing, evaluation/validation** |  |  |
| 4.1 | Unit testing | 1 | Test results |
| 4.2 | Functional testing | 1 | Test results |
| 4.3 | Usability testing | 1 | Test results |
| 4.4 | Maintenance/Fixes | 2 | Updated apps reflecting testing |
| 4.5 | Evaluation of product | 1 | Evaluation |
| **5** | **Assessments** |  |  |
| 5.1 | Write-up project report | 3 | Project Report |
| 5.2 | Produce poster | 0.5 | Poster |
| 5.3 | Demo | 0.5 | Demo |
| 5.4 | Presentation | 0.5 | Completed |
| **TOTAL** | **Sum of total effort in weeks** | **39** | **Completed project** |

| **SECTION 4 - Time Plan for the proposed Project work** | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| For each task identified in 3.1, please *shade* the weeks when you’ll be working on that task. You should also mark target milestones, outputs and key decision points. To shade a cell in MS Word, move the mouse to the top left of cell until the curser becomes an arrow pointing up, left click to select the cell and then right click and select ‘borders and shading’. Under the shading tab pick an appropriate grey colour and click ok. | | | | | | | | | | | | | |
| **Project stage** | **START DATE: 21/06/2017   Project Weeks** | | | | | | | | | | | | |
| 0-3 | 3-6 | 6-9 | 9-12 | 12-15 | 15-18 | 18-21 | 21-24 | 24-27 | 27-30 | 30-33 | 33-36 | 36-39 |
| 1 Background Research |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1.1. Project idea development** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1.2. Market research**  **.** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1.3. Project management planning** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1.4. Risk analysis** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1.5. Collect info for database** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1.6. Stakeholder interviews** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 Analysis/Design |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2.1. Requirement analysis** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2.2. Design app** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2.3. Design database** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 Develop prototype. |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3.1. Develop database** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3.2. Develop app** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 Testing, evaluation/validation |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4.1. Unit testing** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4.2. Functional Testing** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4.3. Usability testing** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4.4. Maintenance/fixes** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4.5. Evaluation** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 Assessments |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5.1. Write-up** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5.2. Project poster** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5.3. Demo** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5.4. Presentation** |  |  |  |  |  |  |  |  |  |  |  |  |  |

PROJECT WRITE UP WILL BE CONTINUED THROUGH THE COURSE OF THE PROJECT.

**RISK ASSESSMENT FORM**

| **Assessment Reference No.** |  | **Area or activity assessed:** |  |
| --- | --- | --- | --- |
| **Assessment date** |  |
| **Persons who may be affected by the activity (i.e. are at risk)** |  |

**SECTION 1: Identify Hazards -** *Consider the activity or work area and identify if any of the hazards listed below are significant (tick the boxes that apply).*

|  | Fall of person (from work at height) | ✔ |  | Lighting levels | ✔ |  | Use of portable tools / equipment | ✔ |  | Vehicles / driving at work |  |  | Hazardous fumes,  chemicals, dust |  |  | Occupational stress | ✔ |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Fall of objects | ✔ |  | Heating & ventilation | ✔ |  | Fixed machinery or lifting equipment |  |  | Outdoor work / extreme weather |  |  | Hazardous biological agent |  |  | Violence to staff / verbal assault |  |
|  | Slips, Trips & Housekeeping | ✔ |  | Layout , storage, space, obstructions | ✔ |  | Pressure vessels |  |  | Fieldtrips / field work |  |  | Confined space / asphyxiation risk |  |  | Work with animals |  |
|  | Manual handling operations |  |  | Welfare facilities | ✔ |  | Noise or Vibration |  |  | Radiation sources |  |  | Condition of Buildings & glazing |  |  | Lone working / work out of hours | ✔ |
| **55** | Display screen equipment | ✔ |  | Electrical Equipment | ✔ |  | Fire hazards & flammable material | ✔ |  | Work with lasers |  |  | Food preparation |  |  | Other(s) - specify |  |

**SECTION 2: Risk Controls** *- For each hazard identified in Section 1, complete Section 2.*

| **Hazard No**. | Hazard Description | Existing controls to reduce risk | **Risk Level** (tick one) | | | Further action needed to reduce risks |
| --- | --- | --- | --- | --- | --- | --- |
|  | High | Med | Low | *(provide timescales and initials of person responsible)* |
| 1 | Fall of a person | Any requires resources are generally at ground level so there is no need to reach or climb for anything. Area kept tidy with no obstacles. |  | ✔ |  | Constantly ensure the existing area is clean and uncluttered with no obstacles on the ground (e.g. loose wires) |
| 2 | Fall of an object | Any objects placed on a ledge or table are pushed back to reduce the risk of them falling off the edge. Ensure any important devices (e.g. tablets, laptops or hard drives) have protective casing for extra measure. |  |  | ✔ | Ensure all around keep good grip of any equipment being handled and are not careless. |
| 3 | Slips, Trips and house keeping | Ensure no food or drink are consumed in the area so that no computers or devices are at risk and no people are at risk of slipping on anything that has spilled. |  | ✔ |  | Ensure area is kept clean on an every day basis to ensure no one is at risk of tripping and no food or drink are consumed around computers. And if they are/ if anything spills, clean it up immediately to avoid the risk of someone slipping in it. |
| **Name of Assessor(s)** | |  | **SIGNED** | | | |
| **Review date** | |  |

| **Health and Safety Risk Assessments** – continuation sheet | **Assessment Reference No** |  |
| --- | --- | --- |
|  | **Continuation sheet number:** | **1** |

**SECTION 2 continued: Risk Controls**

| **Hazard No**. | Hazard Description | | Existing controls to reduce risk | **Risk Level** (tick one) | | | | | Further action needed to reduce risks | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | High | | | Med | Low | *(provide timescales and initials of person responsible for action)* | | |
| 5 | Display screen equipment | | Make sure the screens are set at a brightness level that does not impair the users vision after any amount of time. Stay a good distance away from the computer screen so that there is less risk of eye (and back) strain |  | | |  | ✔ | If possible, get an anti-reflective screen protector to reduce glare on monitors and reduce risk of eye strain. Get regular eye checks to keep up to date. Take breaks from looking at the screen | | |
| 6 | Lighting levels | | Ensure the lighting in the room is at a comfortable level so that users are not straining their eye sight to work in the room. |  | | |  | ✔ | None necessary | | |
| 7 | Heating and ventilation | | Heating is regulated depending on the temperature and is able to be changed by any one in the room. Windows are able to be opened if necessary to allow ventilation. |  | | |  | ✔ | None necessary | | |
| **Name of Assessor(s)** | |  | | | **SIGNED** | | | | | | |
| **Review date** | |  | | |
| **Health and Safety Risk Assessments** – continuation sheet | | | | | | **Assessment Reference No** | | | |  |
|  | | | | | | **Continuation sheet number:** | | | | **2** |

**SECTION 2 continued: Risk Controls**

| **Hazard No**. | Hazard Description | | Existing controls to reduce risk | **Risk Level** (tick one) | | | | | Further action needed to reduce risks | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | High | | | Med | Low | *(provide timescales and initials of person responsible for action)* | | |
| 8 | Layout, storage, space and obstructions | | The room is not cluttered or over crowded so it is not difficult to move around and there are no obstructions to move over or around |  | | |  | ✔ | Ensure no obstructions or obstacles are placed and room is kept clear for easy movement to reduce risk of harm. | | |
| 9 | Welfare facilities | | Breaks are taken regularly to visit restrooms, eat or stretch so that the user does not over work themselves or strain themselves. There is constant access to a bathroom. |  | | |  | ✔ | None necessary | | |
| 10 | Electrical equipment | | No wires being used to charge any devices are frayed or broken. |  | | | ✔ |  | If any wires or cables are frayed they should quickly be replaced to reduce the risk of them affecting any of their devices or starting a fire. | | |
| **Name of Assessor(s)** | |  | | | **SIGNED** | | | | | | |
| **Health and Safety Risk Assessments** – continuation sheet | | | | | | **Assessment Reference No** | | | |  |
|  | | | | | | **Continuation sheet number:** | | | | **3** |

**SECTION 2 continued: Risk Controls**

| **Hazard No**. | Hazard Description | | Existing controls to reduce risk | **Risk Level** (tick one) | | | | | Further action needed to reduce risks | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | High | | | Med | Low | *(provide timescales and initials of person responsible for action)* | | |
| 11 | Use of portable tools/ equipment | | Ensure all portable products have cases that protect them from falls. If necessary always bring the charger for any portable devices being used. |  | | | ✔ |  | Invest in an external hard drive to keep all data backed up so that if any portable devices are lost stolen or broken, there is a spare copy. | | |
| 15 | Fire hazards and flammable equipment | | Ensure all electrical equipment is maintained and in good condition. Put all papers or flammable equipment out of the way to reduce risk of fires spreading. Restrict the use of fire hazards in the working area.  All smoke alarms are working. |  | | | ✔ |  | Invest in fire preventing/stopping equipment such as fire blankets or extinguishers. Carry out a fire safety check within the working area. Regularly test fire alarms. | | |
| 26 | Occupational stress | | Do not work for extended periods of time. Take regular breaks. Create checklists and milestones to organise workload and reduce stress. | ✔ | | |  |  | Ensure regular breaks are taken at appropriate intervals. | | |
| **Name of Assessor(s)** | |  | | | **SIGNED** | | | | | | |
| **Health and Safety Risk Assessments** – continuation sheet | | | | | | **Assessment Reference No** | | | |  |
|  | | | | | | **Continuation sheet number:** | | | | **4** |

**SECTION 2 continued: Risk Controls**

| **Hazard No**. | Hazard Description | | Existing controls to reduce risk | **Risk Level** (tick one) | | | | Further action needed to reduce risks |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | High | | Med | Low | *(provide timescales and initials of person responsible for action)* |
| 29 | Lone working/ working out of hours | | Stick to the given time scales and use checklists to ensure that the given amount of work is completed in a given amount of time to reduce the need to work outside of hours. | ✔ | |  |  | None necessary |
| 30 | Lack of knowledge | | Access to many different forms of information to learn anything necessary (e.g. internet or library). Take online tutorials on any software or application tools that aren’t known. |  | |  | ✔ | None necessary |
| 30 | Loss of data | | Back up data to different forms of storage device. E.g. work is backed up onto google drive after every time it is updated. |  | |  | ✔ | Get an external hard drive as another means of storing data in a more efficient way. |
| **Name of Assessor(s)** | | Dr. Frederic Stahl | | | **SIGNED** | | | |

## Appendix 2 – Application Java Code (functionality)

### 2.1. LoginOrRegistrationActivity

**package** com.example.wk001764.finalproject;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
  
**public class** LoginOrRegisterActivity **extends** AppCompatActivity {  
  
 **private** Button **mLogin**, **mRegister**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_login\_or\_register);  
  
 **mLogin** = (Button) findViewById(R.id.login);  
 **mRegister** = (Button) findViewById(R.id.register);  
  
 **mLogin**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(LoginOrRegisterActivity.**this**, LoginActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 });  
  
 **mRegister**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(LoginOrRegisterActivity.**this**, RegistrationActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 });  
 }  
}

### 2.2. LoginActivity

**package** com.example.wk001764.finalproject;  
  
**import** android.content.Intent;  
**import** android.support.annotation.NonNull;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** com.google.android.gms.tasks.OnCompleteListener;  
**import** com.google.android.gms.tasks.Task;  
**import** com.google.firebase.auth.AuthResult;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.auth.FirebaseUser;  
  
**public class** LoginActivity **extends** AppCompatActivity {  
  
 **private** Button **mLogin**;  
 **private** EditText **mEmail**, **mPassword**;  
  
 **private** FirebaseAuth **mAuth**;  
 **private** FirebaseAuth.AuthStateListener **firebaseAuthStateListener**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_login***);  
  
 **mAuth** = FirebaseAuth.*getInstance*();  
 **firebaseAuthStateListener** = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
 **final** FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();  
 **if** (user !=**null**){  
 Intent intent = **new** Intent(LoginActivity.**this**, MainActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 }  
 };  
  
  
 **mLogin** = (Button) findViewById(R.id.***login***);  
  
 **mEmail** = (EditText) findViewById(R.id.***email***);  
 **mPassword** = (EditText) findViewById(R.id.***password***);  
  
 **mLogin**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **final** String email = **mEmail**.getText().toString();  
 **final** String password = **mPassword**.getText().toString();  
 **mAuth**.signInWithEmailAndPassword(email, password).addOnCompleteListener(LoginActivity.**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if**(!task.isSuccessful()){  
 Toast.*makeText*(LoginActivity.**this**, **"sign in error"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
 }  
 });  
 }  
  
 @Override  
 **protected void** onStart() {  
 **super**.onStart();  
 **mAuth**.addAuthStateListener(**firebaseAuthStateListener**);  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
 **mAuth**.removeAuthStateListener(**firebaseAuthStateListener**);  
 }  
}

### 2.3. Registration Activity

**package** com.example.wk001764.finalproject;  
  
**import** android.content.Intent;  
**import** android.support.annotation.NonNull;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.RadioButton;  
**import** android.widget.RadioGroup;  
**import** android.widget.Toast;  
  
**import** com.google.android.gms.tasks.OnCompleteListener;  
**import** com.google.android.gms.tasks.Task;  
**import** com.google.firebase.auth.AuthResult;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.auth.FirebaseUser;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
  
**import** java.util.HashMap;  
**import** java.util.Map;  
  
**public class** RegistrationActivity **extends** AppCompatActivity {  
  
  
 **private** Button **mRegister**;  
 **private** EditText **mEmail**, **mPassword**, **mName**, **mBio**;  
 **private** RadioGroup **mGender**, **mSport**;  
  
 **private** FirebaseAuth **mAuth**;  
 **private** FirebaseAuth.AuthStateListener **firebaseAuthStateListener**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_registration***);  
  
 **mAuth** = FirebaseAuth.*getInstance*();  
 **firebaseAuthStateListener** = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
 **final** FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();  
 **if** (user !=**null**){  
 Intent intent = **new** Intent(RegistrationActivity.**this**, MainActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 }  
 };  
  
 **mRegister** = (Button) findViewById(R.id.***register***);  
 **mEmail** = (EditText) findViewById(R.id.***email***);  
 **mPassword** = (EditText) findViewById(R.id.***password***);  
 **mName** = (EditText) findViewById(R.id.***name***);  
 **mBio** = (EditText) findViewById(R.id.***bio***);  
 **mGender** = (RadioGroup) findViewById(R.id.***gender***);  
 **mSport** = (RadioGroup) findViewById(R.id.***sport***);  
  
 **mRegister**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 **int** selectGender = **mGender**.getCheckedRadioButtonId();  
 **final** RadioButton genderButton = (RadioButton) findViewById(selectGender);  
 **if**(genderButton.getText() == **null**){  
 **return**;  
 }  
  
 **int** selectSport = **mSport**.getCheckedRadioButtonId();  
 **final** RadioButton sportButton = (RadioButton) findViewById(selectSport);  
 **if**(sportButton.getText() == **null**) {  
 **return**;  
 }  
  
 **final** String email = **mEmail**.getText().toString();  
 **final** String password = **mPassword**.getText().toString();  
 **final** String name = **mName**.getText().toString();  
 **final** String bio = **mBio**.getText().toString();  
  
 **mAuth**.createUserWithEmailAndPassword(email, password).addOnCompleteListener(RegistrationActivity.**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if**(!task.isSuccessful()){  
 Toast.*makeText*(RegistrationActivity.**this**, **"sign up error"**, Toast.***LENGTH\_SHORT***).show();  
 }**else**{  
 String userId = **mAuth**.getCurrentUser().getUid();  
 DatabaseReference currentUserDb = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(userId);  
 Map userInfo = **new** HashMap<>();  
 userInfo.put(**"name"**, name);  
 userInfo.put(**"sex"**, genderButton.getText().toString());  
 userInfo.put(**"profileImageUrl"**, **"default"**);  
 userInfo.put(**"sport"**, sportButton.getText().toString());  
 userInfo.put(**"bio"**, bio);  
 currentUserDb.updateChildren(userInfo);  
 sendVerification();  
 }  
 }  
 });  
 }  
 });  
 }  
  
 **private void** sendVerification() {  
 FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();  
 **if**(user!=**null**){  
 user.sendEmailVerification().addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if**(task.isSuccessful()){  
 Toast.*makeText*(RegistrationActivity.**this**, **"Please Check your Email for Confirmation and Verification"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
 }  
 }  
  
 ;  
  
 @Override  
 **protected void** onStart() {  
 **super**.onStart();  
 **mAuth**.addAuthStateListener(**firebaseAuthStateListener**);  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
 **mAuth**.removeAuthStateListener(**firebaseAuthStateListener**);  
 }  
}

### 2.4. MainActivity

**package** com.example.wk001764.finalproject;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
  
**import** com.example.wk001764.finalproject.Match.MatchActivity;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.auth.FirebaseUser;  
**import** com.google.firebase.database.ChildEventListener;  
**import** com.google.firebase.database.DataSnapshot;  
**import** com.google.firebase.database.DatabaseError;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
**import** com.google.firebase.database.ValueEventListener;  
**import** com.lorentzos.flingswipe.SwipeFlingAdapterView;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 **private** cards **cards\_data**[];  
 **private** arrayAdapter **arrayAdapter**;  
 **private int i**;  
 **private** FirebaseAuth **mAuth**;  
 **private** String **currentUId**;  
 **private** DatabaseReference **usersDb**;  
 **private** String **chosenSport**;  
 **private** String **matchSport**;  
  
 ListView **listView**;  
 List<cards> **rowItems**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **usersDb** = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**);  
 **mAuth** = FirebaseAuth.*getInstance*();  
 **currentUId** = **mAuth**.getCurrentUser().getUid();  
  
 checkChosenSport();  
 **rowItems** = **new** ArrayList<cards>();  
  
 **arrayAdapter** = **new** arrayAdapter(**this**, R.layout.***item***, **rowItems** );  
  
 SwipeFlingAdapterView flingContainer = (SwipeFlingAdapterView) findViewById(R.id.***frame***);  
  
 flingContainer.setAdapter(**arrayAdapter**);  
 flingContainer.setFlingListener(**new** SwipeFlingAdapterView.onFlingListener() {  
 @Override  
 **public void** removeFirstObjectInAdapter() {  
 Log.*d*(**"LIST"**, **"removed object!"**);  
 **rowItems**.remove(0);  
 **arrayAdapter**.notifyDataSetChanged();  
 }  
  
 @Override  
 **public void** onLeftCardExit(Object dataObject) {  
  
 cards obj = (cards) dataObject;  
 String userId = obj.getUserId();  
 **usersDb**.child(userId).child(**"connections"**).child(**"pass"**).child(**currentUId**).setValue(**true**);  
 Toast.*makeText*(MainActivity.**this**, **"Passed"**, Toast.***LENGTH\_SHORT***).show();  
 }  
  
 @Override  
 **public void** onRightCardExit(Object dataObject) {  
 cards obj = (cards) dataObject;  
 String userId = obj.getUserId();  
 **usersDb**.child(userId).child(**"connections"**).child(**"play"**).child(**currentUId**).setValue(**true**);  
 isConnectionMatch(userId);  
 Toast.*makeText*(MainActivity.**this**, **"Played"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 @Override  
 **public void** onAdapterAboutToEmpty(**int** itemsInAdapter) {  
 }  
 @Override  
 **public void** onScroll(**float** scrollProgressPercent) {  
 }  
 });  
  
 *// Optionally add an OnItemClickListener* flingContainer.setOnItemClickListener(**new** SwipeFlingAdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClicked(**int** itemPosition, Object dataObject) {  
 Toast.*makeText*(MainActivity.**this**,**"swipe right to play or left to pass!"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 }  
  
 **private void** isConnectionMatch(String userId) {  
 DatabaseReference currentUserConnectionsDb = **usersDb**.child(**currentUId**).child(**"connections"**).child(**"play"**).child(userId);  
 currentUserConnectionsDb.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
 Toast.*makeText*(MainActivity.**this**, **"new Match!!"**, Toast.***LENGTH\_LONG***).show();  
  
 String key = FirebaseDatabase.*getInstance*().getReference().child(**"Chat"**).push().getKey();  
  
 **usersDb**.child(dataSnapshot.getKey()).child(**"connections"**).child(**"matches"**).child(**currentUId**).child(**"ChatId"**).setValue(key);  
 **usersDb**.child(**currentUId**).child(**"connections"**).child(**"matches"**).child(dataSnapshot.getKey()).child(**"ChatId"**).setValue(key);  
 }  
 }  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 }  
  
 **public void** checkChosenSport(){  
 **final** FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();  
 DatabaseReference userDb = **usersDb**.child(user.getUid());  
 userDb.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
 **if** (dataSnapshot.child(**"sport"**).getValue() != **null**){  
 **chosenSport** = dataSnapshot.child(**"sport"**).getValue().toString();  
 **switch** (**chosenSport**){  
 **case "Badminton"**:  
 **matchSport** = **"Badminton"**;  
 **break**;  
 **case "Basketball"**:  
 **matchSport** = **"Basketball"**;  
 **break**;  
 **case "Boxing"**:  
 **matchSport** = **"Boxing"**;  
 **break**;  
 **case "Cricket"**:  
 **matchSport** = **"Cricket"**;  
 **break**;  
 **case "Football"**:  
 **matchSport** = **"Football"**;  
 **break**;  
 **case "Golf"**:  
 **matchSport** = **"Golf"**;  
 **break**;  
 **case "Hockey"**:  
 **matchSport** = **"Hockey"**;  
 **break**;  
 **case "Rugby"**:  
 **matchSport** = **"Rugby"**;  
 **break**;  
 **case "Swimming"**:  
 **matchSport** = **"Swimming"**;  
 **break**;  
 **case "Tennis"**:  
 **matchSport** = **"Tennis"**;  
 **break**;  
 }  
 getMatchSportUsers();  
 }  
 }  
 }  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 **public void** getMatchSportUsers(){  
 **usersDb**.addChildEventListener(**new** ChildEventListener() {  
 @Override  
 **public void** onChildAdded(DataSnapshot dataSnapshot, String s) {  
 **if** (dataSnapshot.child(**"sport"**).getValue() != **null**) {  
 **if** (dataSnapshot.exists() && !dataSnapshot.child(**"connections"**).child(**"pass"**).hasChild(**currentUId**) && !dataSnapshot.child(**"connections"**).child(**"play"**).hasChild(**currentUId**) && dataSnapshot.child(**"sport"**).getValue().toString().equals(**matchSport**)) {  
 String profileImageUrl = **"default"**;  
 **if** (!dataSnapshot.child(**"profileImageUrl"**).getValue().equals(**"default"**)) {  
 profileImageUrl = dataSnapshot.child(**"profileImageUrl"**).getValue().toString();  
 }  
 cards item = **new** cards(dataSnapshot.getKey(),dataSnapshot.child(**"name"**).getValue().toString(), profileImageUrl, dataSnapshot.child(**"bio"**).getValue().toString());  
 **rowItems**.add(item);  
 **arrayAdapter**.notifyDataSetChanged();  
 }  
 }  
 }  
 @Override  
 **public void** onChildChanged(DataSnapshot dataSnapshot, String s) {  
 }  
 @Override  
 **public void** onChildRemoved(DataSnapshot dataSnapshot) {  
 }  
  
 @Override  
 **public void** onChildMoved(DataSnapshot dataSnapshot, String s) {  
 }  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 }  
  
  
 **public void** logoutUser(View view) {  
 **mAuth**.signOut();  
 Intent intent = **new** Intent(MainActivity.**this**, LoginOrRegisterActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
  
 **public void** settings(View view) {  
 Intent intent = **new** Intent(MainActivity.**this**, SettingsActivity.**class**);  
 startActivity(intent);  
 **return**;  
 }  
  
 **public void** matches(View view) {  
 Intent intent = **new** Intent(MainActivity.**this**, MatchActivity.**class**);  
 startActivity(intent);  
 **return**;  
 }

### 2.5. SettingsActivity

**package** com.example.wk001764.finalproject;  
  
**import** android.app.Activity;  
**import** android.content.Intent;  
**import** android.graphics.Bitmap;  
**import** android.net.Uri;  
**import** android.os.Build;  
**import** android.provider.MediaStore;  
**import** android.support.annotation.NonNull;  
**import** android.support.annotation.RequiresApi;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.view.autofill.AutofillValue;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.ImageView;  
**import** android.widget.RadioButton;  
**import** android.widget.RadioGroup;  
  
**import** com.bumptech.glide.Glide;  
**import** com.google.android.gms.tasks.OnFailureListener;  
**import** com.google.android.gms.tasks.OnSuccessListener;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.database.DataSnapshot;  
**import** com.google.firebase.database.DatabaseError;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
**import** com.google.firebase.database.ValueEventListener;  
**import** com.google.firebase.storage.FirebaseStorage;  
**import** com.google.firebase.storage.StorageReference;  
**import** com.google.firebase.storage.UploadTask;  
  
**import** java.io.ByteArrayOutputStream;  
**import** java.io.IOException;  
**import** java.util.HashMap;  
**import** java.util.Map;  
  
**public class** SettingsActivity **extends** AppCompatActivity {  
  
 **private** EditText **mNameField**, **mPhoneField**, **mBio**;  
 **private** Button **mBack**, **mConfirm**;  
 **private** RadioGroup **mSport**;  
 **private** ImageView **mProfileImage**;  
 **private** FirebaseAuth **mAuth**;  
 **private** DatabaseReference **mUserDatabase**;  
 **private** String **userId**, **name**, **phone**, **bio**, **sport**, **profileImageUrl**, **userSex**;  
 **private** Uri **resultUri**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_settings***);  
  
 **mNameField** = (EditText) findViewById(R.id.***name***);  
 **mPhoneField** = (EditText) findViewById(R.id.***phone***);  
 **mBio** = (EditText) findViewById(R.id.***bio***);  
 **mSport** = (RadioGroup) findViewById(R.id.***sport***);  
 **mProfileImage** = (ImageView) findViewById(R.id.***profileImage***);  
 **mBack** = (Button) findViewById(R.id.***back***);  
 **mConfirm** = (Button) findViewById(R.id.***confirm***);  
 **mAuth** = FirebaseAuth.*getInstance*();  
 **userId** = **mAuth**.getCurrentUser().getUid();  
 **mUserDatabase** = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**userId**);  
 getUserInfo();  
  
 **mProfileImage**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(Intent.***ACTION\_PICK***);  
 intent.setType(**"image/\*"**);  
 startActivityForResult(intent, 1);  
 }  
 });  
 **mConfirm**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 saveUserInformation();  
 }  
 });  
 **mBack**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 saveUserInformation();  
 finish();  
 **return**;  
 }  
 });  
 }  
  
  
  
 **private void** getUserInfo() {  
 **mUserDatabase**.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.***O***)  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
  
  
 **int** selectSport = **mSport**.getCheckedRadioButtonId();  
 **final** RadioButton sportButton = (RadioButton) findViewById(selectSport);  
  
 **if**(dataSnapshot.exists() && dataSnapshot.getChildrenCount()>0){  
 Map<String, Object> map = (Map<String, Object>) dataSnapshot.getValue();  
  
 **if**(map.get(**"name"**)!=**null**){  
 **name** = map.get(**"name"**).toString();  
 **mNameField**.setText(**name**);  
 }  
 **if**(map.get(**"phone"**)!=**null**){  
 **phone** = map.get(**"phone"**).toString();  
 **mPhoneField**.setText(**phone**);  
 }  
 **if**(map.get(**"bio"**)!=**null**){  
 **bio** = map.get(**"bio"**).toString();  
 **mBio**.setText(**bio**);  
 }  
 **if**(map.get(**"sex"**)!=**null**){  
 **userSex** = map.get(**"sex"**).toString();  
 }  
 **if**(map.get(**"sport"**) !=**null**){  
 */\* sport = map.get("sport").toString();  
 sportButton.setText(sport);\*/* }  
  
 Glide.*clear*(**mProfileImage**);  
 **if**(map.get(**"profileImageUrl"**)!=**null**){  
 **profileImageUrl** = map.get(**"profileImageUrl"**).toString();  
 **switch**(**profileImageUrl**){  
 **case "default"**:  
 Glide.*with*(getApplication()).load(R.mipmap.default\_app\_image).into(mProfileImage);  
 **break**;  
 **default**:  
 Glide.with(getApplication()).load(profileImageUrl).into(mProfileImage);  
 **break**;  
 }  
 }  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 }  
  
 **private void** saveUserInformation() {  
 name = mNameField.getText().toString();  
 phone = mPhoneField.getText().toString();  
 bio = mBio.getText().toString();  
  
 **int** selectSport = mSport.getCheckedRadioButtonId();  
 View radioButton = mSport.findViewById(selectSport);  
 **int** chosen = mSport.indexOfChild(radioButton);  
  
 RadioButton rb = (RadioButton) mSport.getChildAt(chosen);  
 String selectedText = rb.getText().toString();  
  
 sport = selectedText;  
  
 Map userInfo = **new** HashMap();  
 userInfo.put(**"name"**, name);  
 userInfo.put(**"phone"**, phone);  
 userInfo.put(**"bio"**, bio);  
 userInfo.put(**"sport"**,sport);  
  
 mUserDatabase.updateChildren(userInfo);  
 **if**(resultUri != **null**){  
 StorageReference filepath = FirebaseStorage.getInstance().getReference().child(**"profileImages"**).child(userId);  
 Bitmap bitmap = **null**;  
  
 **try** {  
 bitmap = MediaStore.Images.Media.getBitmap(getApplication().getContentResolver(), resultUri);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 ByteArrayOutputStream baos = **new** ByteArrayOutputStream();  
 bitmap.compress(Bitmap.CompressFormat.JPEG, 20, baos);  
 **byte**[] data = baos.toByteArray();  
 UploadTask uploadTask = filepath.putBytes(data);  
 uploadTask.addOnFailureListener(**new** OnFailureListener() {  
 @Override  
 **public void** onFailure(@NonNull Exception e) {  
 finish();  
 }  
 });  
 uploadTask.addOnSuccessListener(**new** OnSuccessListener<UploadTask.TaskSnapshot>() {  
 @Override  
 **public void** onSuccess(UploadTask.TaskSnapshot taskSnapshot) {  
 Uri downloadUrl = taskSnapshot.getDownloadUrl();  
  
 Map userInfo = **new** HashMap();  
 userInfo.put(**"profileImageUrl"**, downloadUrl.toString());  
 mUserDatabase.updateChildren(userInfo);  
  
 finish();  
 **return**;  
 }  
 });  
 }**else**{  
 finish();  
 }  
 }  
  
 @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 **super**.onActivityResult(requestCode, resultCode, data);  
 **if**(requestCode == 1 && resultCode == Activity.RESULT\_OK){  
 **final** Uri imageUri = data.getData();  
 resultUri = imageUri;  
 mProfileImage.setImageURI(resultUri);  
 }  
 }  
}

### 2.6. Cards

**package** com.example.wk001764.finalproject;  
  
**public class** cards {  
 **private** String **userId**;  
 **private** String **name**;  
 **private** String **profileImageUrl**;  
 **private** String **bio**;  
  
 **public** cards(String userId, String name, String profileImageUrl, String bio) {  
 **this**.**userId** = userId;  
 **this**.**name** = name;  
 **this**.**profileImageUrl** = profileImageUrl;  
 **this**.**bio** = bio;  
 }  
  
 **public** String getUserId() {  
 **return userId**;  
 }  
  
 **public void** setUserID(String userID) {  
 **this**.**userId** = **userId**;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** String getProfileImageUrl() {  
 **return profileImageUrl**;  
 }  
  
 **public void** setProfileImageUrl(String profileImageUrl) {  
 **this**.**profileImageUrl** = profileImageUrl;  
 }  
  
 **public** String getBio() {  
 **return bio**;  
 }  
  
 **public void** setBio(String bio) {  
 **this**.**bio** = bio;  
 }  
}

### 2.7. ArrayAdapter

**package** com.example.wk001764.finalproject;  
  
**import** android.annotation.SuppressLint;  
**import** android.content.Context;  
**import** android.provider.ContactsContract;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ImageView;  
**import** android.widget.TextView;  
**import** com.bumptech.glide.Glide;  
  
**import** java.util.List;  
  
**public class** arrayAdapter **extends** ArrayAdapter{  
  
  
 Context **context**;  
  
 **public** arrayAdapter(Context context, **int** resourceId, List<cards> items){  
 **super**(context, resourceId, items);  
 }  
 **public** View getView(**int** position, View convertView, ViewGroup parent){  
 cards card\_item = (cards) getItem(position);  
  
 **if** (convertView == **null**){  
 convertView = LayoutInflater.*from*(getContext()).inflate(R.layout.***item***, parent, **false**);  
 }  
  
 TextView name = (TextView) convertView.findViewById(R.id.***name***);  
 ImageView image = (ImageView) convertView.findViewById(R.id.***image***);  
 TextView bio = (TextView) convertView.findViewById(R.id.***bio***);  
  
 name.setText(card\_item.getName());  
 bio.setText(card\_item.getBio());  
 **switch**(card\_item.getProfileImageUrl()){  
 **case "default"**:  
 Glide.*with*(convertView.getContext()).load(R.mipmap.***default\_app\_image***).into(image);  
 **break**;  
 **default**:  
 Glide.*clear*(image);  
 Glide.*with*(convertView.getContext()).load(card\_item.getProfileImageUrl()).into(image);  
 **break**;  
 }  
 **return** convertView;  
 }  
}

### 2.8. Match Package

#### 2.8.1. MatchActivity

**package** com.example.wk001764.finalproject.Match;  
  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.support.v7.widget.LinearLayoutManager;  
**import** android.support.v7.widget.RecyclerView;  
  
**import** com.example.wk001764.finalproject.R;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.database.DataSnapshot;  
**import** com.google.firebase.database.DatabaseError;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
**import** com.google.firebase.database.ValueEventListener;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** MatchActivity **extends** AppCompatActivity {  
 **private** RecyclerView **mRecyclerView**;  
 **private** RecyclerView.Adapter **mMatchesAdapter**;  
 **private** RecyclerView.LayoutManager **mMatchesLayoutManager**;  
 **private** String **currentUserID**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_matches***);  
  
 **currentUserID** = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
  
 **mRecyclerView** = (RecyclerView) findViewById(R.id.***recyclerView***);  
 **mRecyclerView**.setNestedScrollingEnabled(**false**);  
 **mRecyclerView**.setHasFixedSize(**true**);  
 **mMatchesLayoutManager** = **new** LinearLayoutManager(MatchActivity.**this**);  
 **mRecyclerView**.setLayoutManager(**mMatchesLayoutManager**);  
 **mMatchesAdapter** = **new** MatchAdapter(getDataSetMatches(), MatchActivity.**this**);  
 **mRecyclerView**.setAdapter(**mMatchesAdapter**);  
 getUserMatchId();  
 }  
  
 **private void** getUserMatchId() {  
  
 DatabaseReference matchDb = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**currentUserID**).child(**"connections"**).child(**"matches"**);  
 matchDb.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
 **for**(DataSnapshot match : dataSnapshot.getChildren()){  
 FetchMatchInformation(match.getKey());  
 }  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 **private void** FetchMatchInformation(String key) {  
 DatabaseReference userDb = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(key);  
 userDb.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
 String userId = dataSnapshot.getKey();  
 String name = **""**;  
 String profileImageUrl = **""**;  
 String bio = **""**;  
 **if**(dataSnapshot.child(**"name"**).getValue()!=**null**){  
 name = dataSnapshot.child(**"name"**).getValue().toString();  
 }  
 **if**(dataSnapshot.child(**"profileImageUrl"**).getValue()!=**null**){  
 profileImageUrl = dataSnapshot.child(**"profileImageUrl"**).getValue().toString();  
 }  
 **if** (dataSnapshot.child(**"bio"**).getValue()!=**null**){  
 bio = dataSnapshot.child(**"bio"**).getValue().toString();  
 }  
  
 MatchObject obj = **new** MatchObject(userId, name, profileImageUrl, bio);  
 **resultsMatches**.add(obj);  
 **mMatchesAdapter**.notifyDataSetChanged();  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
  
 }  
  
 **private** ArrayList<MatchObject> **resultsMatches** = **new** ArrayList<MatchObject>();  
 **private** List<MatchObject> getDataSetMatches() {  
 **return resultsMatches**;  
 }  
  
}

#### 2.8.2. MatchAdapter

**package** com.example.wk001764.finalproject.Match;  
  
**import** android.content.Context;  
**import** android.support.v4.content.ContextCompat;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
  
**import** com.bumptech.glide.Glide;  
**import** com.example.wk001764.finalproject.R;  
  
**import** java.util.List;  
  
**public class** MatchAdapter **extends** RecyclerView.Adapter<MatchViewHolders>{  
 **private** List<MatchObject> **matchesList**;  
 **private** Context **context**;  
  
  
 **public** MatchAdapter(List<MatchObject> matchesList, Context context){  
 **this**.**matchesList** = matchesList;  
 **this**.**context** = context;  
 }  
  
 @Override  
 **public** MatchViewHolders onCreateViewHolder(ViewGroup parent, **int** viewType) {  
  
 View layoutView = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.***item\_match***, **null**, **false**);  
 RecyclerView.LayoutParams lp = **new** RecyclerView.LayoutParams(ViewGroup.LayoutParams.***MATCH\_PARENT***, ViewGroup.LayoutParams.***WRAP\_CONTENT***);  
 layoutView.setLayoutParams(lp);  
 MatchViewHolders rcv = **new** MatchViewHolders(layoutView);  
  
 **return** rcv;  
 }  
  
 @Override  
 **public void** onBindViewHolder(MatchViewHolders holder, **int** position) {  
 holder.**mMatchId**.setText(**matchesList**.get(position).getUserId());  
 holder.**mMatchBio**.setText(**matchesList**.get(position).getBio());  
 holder.**mMatchName**.setText(**matchesList**.get(position).getName());  
 **if**(!**matchesList**.get(position).getProfileImageUrl().equals(**"default"**)){  
 Glide.*with*(**context**).load(**matchesList**.get(position).getProfileImageUrl()).into(holder.**mMatchImage**);  
 }  
 }  
  
 @Override  
 **public int** getItemCount() {  
 **return this**.**matchesList**.size();  
 }  
}

#### 2.8.3. MatchObject

**package** com.example.wk001764.finalproject.Match;  
  
**public class** MatchObject {  
 **private** String **userId**;  
 **private** String **name**;  
 **private** String **profileImageUrl**;  
 **private** String **bio**;  
 **public** MatchObject (String userId, String name, String profileImageUrl, String bio){  
 **this**.**userId** = userId;  
 **this**.**name** = name;  
 **this**.**profileImageUrl** = profileImageUrl;  
 **this**.**bio** = bio;  
 }  
  
 **public** String getUserId(){  
 **return userId**;  
 }  
 **public void** setUserID(String userID){  
 **this**.**userId** = **userId**;  
 }  
  
 **public** String getName(){  
 **return name**;  
 }  
 **public void** setName(String name){  
 **this**.**name** = name;  
 }  
  
 **public** String getProfileImageUrl(){  
 **return profileImageUrl**;  
 }  
 **public void** setProfileImageUrl(String profileImageUrl){  
 **this**.**profileImageUrl** = profileImageUrl;  
 }  
  
 **public** String getBio(){  
 **return bio**;  
 }  
 **public void** setBio(String bio){  
 **this**.**bio** = bio;  
 }  
}

#### 2.8.4. MatchViewHolders

**package** com.example.wk001764.finalproject.Match;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.view.View;  
**import** android.widget.ImageView;  
**import** android.widget.TextView;  
  
**import** com.example.wk001764.finalproject.Messaging.ChatActivity;  
**import** com.example.wk001764.finalproject.R;  
  
**public class** MatchViewHolders **extends** RecyclerView.ViewHolder **implements** View.OnClickListener{  
 **public** TextView **mMatchId**, **mMatchBio**, **mMatchName**;  
 **public** ImageView **mMatchImage**;  
 **public** MatchViewHolders(View itemView) {  
 **super**(itemView);  
 itemView.setOnClickListener(**this**);  
  
 **mMatchId** = (TextView) itemView.findViewById(R.id.***matchId***);  
 **mMatchBio** = (TextView) itemView.findViewById(R.id.***matchBio***);  
 **mMatchName** = (TextView) itemView.findViewById(R.id.***matchName***);  
 **mMatchImage** = (ImageView) itemView.findViewById(R.id.***matchImage***);  
 }  
  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(view.getContext(), ChatActivity.**class**);  
 Bundle b = **new** Bundle();  
 b.putString(**"matchId"**, **mMatchId**.getText().toString());  
 intent.putExtras(b);  
 view.getContext().startActivity(intent);  
 }  
}

### 2.9. Messaging Package

#### 2.9.1. ChatActivity

**package** com.example.wk001764.finalproject.Messaging;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.support.v7.widget.LinearLayoutManager;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.ImageButton;  
  
**import** com.example.wk001764.finalproject.R;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.database.ChildEventListener;  
**import** com.google.firebase.database.DataSnapshot;  
**import** com.google.firebase.database.DatabaseError;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
**import** com.google.firebase.database.ValueEventListener;  
  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** java.util.Map;  
  
**public class** ChatActivity **extends** AppCompatActivity {  
 **private** RecyclerView **mRecyclerView**;  
 **private** RecyclerView.Adapter **mChatAdapter**;  
 **private** RecyclerView.LayoutManager **mChatLayoutManager**;  
  
 **private** EditText **mSendEditText**;  
 **private** Button **mSendButton**;  
 **private** ImageButton **mCreateEvent**;  
 **private** String **currentUserID**, **matchId**, **chatId**;  
  
 DatabaseReference **mDatabaseUser**, **mDatabaseChat**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_chat***);  
 **matchId** = getIntent().getExtras().getString(**"matchId"**);  
 **currentUserID** = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 **mDatabaseUser** = FirebaseDatabase.*getInstance*().getReference().child(**"Users"**).child(**currentUserID**).child(**"connections"**).child(**"matches"**).child(**matchId**).child(**"ChatId"**);  
 **mDatabaseChat** = FirebaseDatabase.*getInstance*().getReference().child(**"Chat"**);  
 getChatId();  
 **mRecyclerView** = (RecyclerView) findViewById(R.id.***recyclerView***);  
 **mRecyclerView**.setNestedScrollingEnabled(**false**);  
 **mRecyclerView**.setHasFixedSize(**false**);  
 **mChatLayoutManager** = **new** LinearLayoutManager(ChatActivity.**this**);  
 **mRecyclerView**.setLayoutManager(**mChatLayoutManager**);  
 **mChatAdapter** = **new** ChatAdapter(getDataSetChat(), ChatActivity.**this**);  
 **mRecyclerView**.setAdapter(**mChatAdapter**);  
 **mSendEditText** = findViewById(R.id.***message***);  
 **mSendButton** = findViewById(R.id.***send***);  
 **mCreateEvent** = findViewById(R.id.***createEvent***);  
  
 **mCreateEvent**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(ChatActivity.**this**, CalendarActivity.**class**);  
 startActivity(intent);  
 finish();  
 **return**;  
 }  
 });  
  
 **mSendButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 sendMessage();  
 }  
 });  
 }  
  
  
 **private void** sendMessage() {  
 String sendMessageText = **mSendEditText**.getText().toString();  
  
 **if**(!sendMessageText.isEmpty()){  
 DatabaseReference newMessageDb = **mDatabaseChat**.push();  
  
 Map newMessage = **new** HashMap();  
 newMessage.put(**"createdByUser"**, **currentUserID**);  
 newMessage.put(**"text"**, sendMessageText);  
  
 newMessageDb.setValue(newMessage);  
 }  
 **mSendEditText**.setText(**null**);  
 }  
  
 **private void** getChatId(){  
 **mDatabaseUser**.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.exists()){  
 **chatId** = dataSnapshot.getValue().toString();  
 **mDatabaseChat** = **mDatabaseChat**.child(**chatId**);  
 getChatMessages();  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 **private void** getChatMessages() {  
 **mDatabaseChat**.addChildEventListener(**new** ChildEventListener() {  
 @Override  
 **public void** onChildAdded(DataSnapshot dataSnapshot, String s) {  
 **if**(dataSnapshot.exists()){  
 String message = **null**;  
 String createdByUser = **null**;  
  
 **if**(dataSnapshot.child(**"text"**).getValue()!=**null**){  
 message = dataSnapshot.child(**"text"**).getValue().toString();  
 }  
 **if**(dataSnapshot.child(**"createdByUser"**).getValue()!=**null**){  
 createdByUser = dataSnapshot.child(**"createdByUser"**).getValue().toString();  
 }  
  
 **if**(message!=**null** && createdByUser!=**null**){  
 Boolean currentUserBoolean = **false**;  
 **if**(createdByUser.equals(**currentUserID**)){  
 currentUserBoolean = **true**;  
 }  
 ChatObject newMessage = **new** ChatObject(message, currentUserBoolean);  
 **resultsChat**.add(newMessage);  
 **mChatAdapter**.notifyDataSetChanged();  
 }  
 }  
  
 }  
 @Override  
 **public void** onChildChanged(DataSnapshot dataSnapshot, String s) {  
 }  
 @Override  
 **public void** onChildRemoved(DataSnapshot dataSnapshot) {  
 }  
 @Override  
 **public void** onChildMoved(DataSnapshot dataSnapshot, String s) {  
 }  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 }  
  
  
 **private** ArrayList<ChatObject> **resultsChat** = **new** ArrayList<ChatObject>();  
 **private** List<ChatObject> getDataSetChat() {  
 **return resultsChat**;  
 }  
}

#### 2.9.2. ChatAdapter

**package** com.example.wk001764.finalproject.Messaging;  
  
**import** android.content.Context;  
**import** android.graphics.Color;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.view.Gravity;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
  
**import** com.bumptech.glide.Glide;  
**import** com.example.wk001764.finalproject.Match.MatchObject;  
**import** com.example.wk001764.finalproject.Match.MatchViewHolders;  
**import** com.example.wk001764.finalproject.R;  
  
**import** java.util.List;  
  
*/\*\*  
 \* Created by wk001764 on 07/02/2018.  
 \*/***public class** ChatAdapter **extends** RecyclerView.Adapter<ChatViewHolders>{  
 **private** List<ChatObject> **chatList**;  
 **private** Context **context**;  
  
  
 **public** ChatAdapter(List<ChatObject> matchesList, Context context){  
 **this**.**chatList** = matchesList;  
 **this**.**context** = context;  
 }  
  
 @Override  
 **public** ChatViewHolders onCreateViewHolder(ViewGroup parent, **int** viewType) {  
  
 View layoutView = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.***item\_chat***, **null**, **false**);  
 RecyclerView.LayoutParams lp = **new** RecyclerView.LayoutParams(ViewGroup.LayoutParams.***MATCH\_PARENT***, ViewGroup.LayoutParams.***WRAP\_CONTENT***);  
 layoutView.setLayoutParams(lp);  
 ChatViewHolders rcv = **new** ChatViewHolders(layoutView);  
  
 **return** rcv;  
 }  
  
 @Override  
 **public void** onBindViewHolder(ChatViewHolders holder, **int** position) {  
 holder.**mMessage**.setText(**chatList**.get(position).getMessage());  
 **if**(**chatList**.get(position).getCurrentUser()){  
 holder.**mMessage**.setGravity(Gravity.***END***);  
 holder.**mMessage**.setTextColor(Color.*parseColor*(**"#404040"**));  
 holder.**mContainer**.setBackgroundColor(Color.*parseColor*(**"#F4F4F4"**));  
 }**else**{  
 holder.**mMessage**.setGravity(Gravity.***START***);  
 holder.**mMessage**.setTextColor(Color.*parseColor*(**"#FFFFFF"**));  
 holder.**mContainer**.setBackgroundColor(Color.*parseColor*(**"#2DB4C8"**));  
 }  
  
 }  
  
 @Override  
 **public int** getItemCount() {  
 **return this**.**chatList**.size();  
 }  
}

#### 2.9.3. ChatObject

**package** com.example.wk001764.finalproject.Messaging;  
  
*/\*\*  
 \* Created by wk001764 on 07/02/2018.  
 \*/***public class** ChatObject {  
 **private** String **message**;  
 **private** Boolean **currentUser**;  
  
 **public** ChatObject(String message, Boolean currentUser){  
 **this**.**message** = message;  
 **this**.**currentUser** = currentUser;  
 }  
  
 **public** String getMessage(){  
 **return message**;  
 }  
 **public void** setMessage(String userID){  
 **this**.**message** = **message**;  
 }  
  
 **public** Boolean getCurrentUser(){  
 **return currentUser**;  
 }  
 **public void** setCurrentUser(Boolean currentUser){  
 **this**.**currentUser** = currentUser;  
 }  
}

#### 2.9.4. ChatViewHolders

**package** com.example.wk001764.finalproject.Messaging;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.view.View;  
**import** android.widget.ImageView;  
**import** android.widget.LinearLayout;  
**import** android.widget.TextView;  
  
**import** com.example.wk001764.finalproject.R;  
  
*/\*\*  
 \* Created by wk001764 on 07/02/2018.  
 \*/***public class** ChatViewHolders **extends** RecyclerView.ViewHolder **implements** View.OnClickListener{  
 **public** TextView **mMessage**;  
 **public** LinearLayout **mContainer**;  
 **public** ChatViewHolders(View itemView) {  
 **super**(itemView);  
 itemView.setOnClickListener(**this**);  
  
 **mMessage** = itemView.findViewById(R.id.***message***);  
 **mContainer** = itemView.findViewById(R.id.***container***);  
 }  
  
 @Override  
 **public void** onClick(View view) {  
 }  
}

#### 2.9.5. CalendarActivity

**package** com.example.wk001764.finalproject.Messaging;  
  
**import** android.content.Intent;  
**import** android.provider.CalendarContract;  
**import** android.support.annotation.NonNull;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.CalendarView;  
**import** android.widget.DatePicker;  
**import** android.widget.EditText;  
  
**import** com.example.wk001764.finalproject.R;  
  
**import** java.util.Calendar;  
**import** java.util.GregorianCalendar;  
  
**public class** CalendarActivity **extends** AppCompatActivity {  
  
 **private** Button **mSave**;  
 **private** EditText **mTitle**, **mDescription**, **mLocation**;  
 **private** CalendarView **mDate**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_calendar***);  
  
 **mSave** = (Button) findViewById(R.id.***save***);  
 **mTitle** = (EditText) findViewById(R.id.***title***);  
 **mDescription** = (EditText) findViewById(R.id.***description***);  
 **mLocation** = (EditText) findViewById(R.id.***location***);  
 **mDate** = (CalendarView) findViewById(R.id.***date***);  
  
 **mSave**.setOnClickListener(**new** View.OnClickListener(){  
 @Override  
  
 **public void** onClick(View view) {  
  
 **final** String title = **mTitle**.getText().toString();  
 **final** String description = **mDescription**.getText().toString();  
 **final** String location = **mLocation**.getText().toString();  
 Calendar date = Calendar.*getInstance*();  
 **int** year = date.get(Calendar.***YEAR***);  
 **int** month = date.get(Calendar.***MONTH***);  
 **int** day = date.get(Calendar.***DAY\_OF\_MONTH***);  
  
 Intent calIntent = **new** Intent(Intent.***ACTION\_INSERT***);  
 calIntent.setData(CalendarContract.Events.***CONTENT\_URI***);  
 calIntent.setType(**"vnd.android.cursor.item/event"**);  
 calIntent.putExtra(CalendarContract.Events.***TITLE***, title);  
 calIntent.putExtra(CalendarContract.Events.***EVENT\_LOCATION***, location);  
 calIntent.putExtra(CalendarContract.Events.***DESCRIPTION***, description);  
  
   
 GregorianCalendar calDate = **new** GregorianCalendar(year,month,day);  
 calIntent.putExtra(CalendarContract.***EXTRA\_EVENT\_ALL\_DAY***, **true**);  
 calIntent.putExtra(CalendarContract.***EXTRA\_EVENT\_BEGIN\_TIME***,  
 calDate.getTimeInMillis());  
 calIntent.putExtra(CalendarContract.***EXTRA\_EVENT\_END\_TIME***,  
 calDate.getTimeInMillis());  
  
 calIntent.putExtra(CalendarContract.Events.***ACCESS\_LEVEL***, CalendarContract.Events.***ACCESS\_PRIVATE***);  
 calIntent.putExtra(CalendarContract.Events.***AVAILABILITY***, CalendarContract.Events.***AVAILABILITY\_BUSY***);  
 startActivity(calIntent);  
  
 }  
  
 });  
 }  
}

## Appendix 3 – Application XML Code (design)

### 3.1. Activity\_calendar.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/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"  
 tools:context="com.example.wk001764.finalproject.Messaging.CalendarActivity"  
 android:orientation="vertical"  
 android:background="#ff92cdc3"**>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Event Title"  
 android:id="@+id/title"  
 android:textColor="#ffffff"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Event Location"  
 android:id="@+id/location"  
 android:textColor="#ffffff"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Event Description"  
 android:id="@+id/description"  
 android:textColor="#ffffff"**/>  
  
 <**CalendarView  
 android:id="@+id/date"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:selectedWeekBackgroundColor="#44BB99"  
 android:weekNumberColor="#44BB99"** /> =/>  
  
 <**Button  
 android:id="@+id/save"  
 android:layout\_width="195dp"  
 android:layout\_height="wrap\_content"  
 android:background="#ff92cdc3"  
 android:text="Save"  
 android:textColor="#ffffff"  
 android:layout\_gravity="center"**/>  
</**LinearLayout**>

### 3.2. Activity\_chat.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout  
 xmlns:android="http://schemas.android.com/apk/res/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"  
 tools:context="com.example.wk001764.finalproject.Messaging.ChatActivity"  
 android:background="#ff92cdc3"**>  
 <**android.support.v4.widget.NestedScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_above="@+id/sendLayout"**>  
 <**android.support.v7.widget.RecyclerView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/recyclerView"  
 android:scrollbars="vertical"**>  
 </**android.support.v7.widget.RecyclerView**>  
 </**android.support.v4.widget.NestedScrollView**>  
 <**LinearLayout  
 android:id="@+id/sendLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:orientation="horizontal"**>  
 <**EditText  
 android:layout\_weight="0.8"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/message"  
 android:hint="Start typing..."  
 android:textColor="#ffffff"**/>  
  
 <**Button  
 android:id="@+id/send"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.1"  
 android:text="Send"  
 android:textColor="#ffffff"  
 android:background="#ff92cdc3"**/>  
  
 <**ImageButton  
 android:id="@+id/createEvent"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.1"  
 android:background="#ff92cdc3"  
 app:srcCompat="@android:drawable/ic\_menu\_my\_calendar"** />  
  
 </**LinearLayout**>  
</**RelativeLayout**>

### 3.3. Activity\_login.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/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"  
 tools:context="com.example.wk001764.finalproject.RegistrationActivity"  
 android:orientation="vertical"  
 android:background="#ff92cdc3"**>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Email Address"  
 android:id="@+id/email"  
 android:textColor="#ffffff"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Password"  
 android:id="@+id/password"  
 android:textColor="#ffffff"**/>  
  
 <**Button  
 android:id="@+id/login"  
 android:layout\_width="193dp"  
 android:layout\_height="wrap\_content"  
 android:text="Login"  
 android:layout\_gravity="center"  
 android:background="#ff92cdc3"  
 android:textColor="#ffffff"**/>  
</**LinearLayout**>

### 3.4. Activity\_login\_or\_register.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/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"  
 tools:context="com.example.wk001764.finalproject.SettingsActivity"  
 android:background="#92cdc3"  
 android:orientation="vertical"  
 android:gravity="center"**>  
  
  
 <**TextView  
 android:id="@+id/welcome"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Welcome to MatchUp!"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1"  
 android:background="#92CDC3"  
 android:textColor="#ffffff"**/>  
  
 <**ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="134dp"  
 android:src="@mipmap/ic\_logo1"  
 android:background="#92CDC3"**/>  
  
 <**Button  
 android:id="@+id/register"  
 android:layout\_width="149dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:background="#92CDC3"  
 android:textColor="#ffffff"  
 android:text="Register"** />  
  
 <**Button  
 android:id="@+id/login"  
 android:layout\_width="149dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:background="#92CDC3"  
 android:textColor="#ffffff"  
 android:text="Login"** />  
  
</**LinearLayout**>

### 3.5. Activity\_main.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/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"  
 tools:context="com.example.wk001764.finalproject.MainActivity"  
 android:orientation="vertical"**>  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"**>  
 <**Button  
 android:layout\_weight="1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Logout"  
 android:onClick="logoutUser"  
 android:background="#ff92cdc3"  
 android:textColor="#ffffff"**/>  
 <**Button  
 android:layout\_weight="1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Matches"  
 android:onClick="matches"  
 android:background="#ff92cdc3"  
 android:textColor="#ffffff"**/>  
 <**Button  
 android:layout\_weight="1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Settings"  
 android:onClick="settings"  
 android:background="#ff92cdc3"  
 android:textColor="#ffffff"**/>  
 </**LinearLayout**>  
 <**com.lorentzos.flingswipe.SwipeFlingAdapterView  
 android:id="@+id/frame"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:rotation\_degrees="15.5"  
 tools:context=".MainActivity"  
 android:background="#ff92cdc3"** />  
  
</**LinearLayout**>

### 3.6. Activity\_matches.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/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"  
 tools:context="com.example.wk001764.finalproject.Match.MatchActivity"  
 android:background="#ff92cdc3"**>  
 <**android.support.v4.widget.NestedScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
 <**android.support.v7.widget.RecyclerView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/recyclerView"  
 android:scrollbars="vertical"**>  
 </**android.support.v7.widget.RecyclerView**>  
 </**android.support.v4.widget.NestedScrollView**>  
</**LinearLayout**>

### 3.7. Activity\_registration.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**ScrollView  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"**>  
<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/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"  
 tools:context="com.example.wk001764.finalproject.RegistrationActivity"  
 android:orientation="vertical"  
 android:background="#ff92cdc3"**>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Full Name"  
 android:id="@+id/name"  
 android:textColor="#ffffff"**/>  
 <**RadioGroup  
 android:id="@+id/gender"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:orientation="horizontal"**>  
 <**RadioButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Male"  
 android:textColor="#ffffff"**/>  
 <**RadioButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Female"  
 android:textColor="#ffffff"** />  
 </**RadioGroup**>  
  
 <**EditText  
 android:id="@+id/email"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Email Address"  
 android:inputType="textEmailAddress"  
 android:textColor="#ffffff"**/>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Password (Must be longer than 6 characters)"  
 android:id="@+id/password"  
 android:inputType="textPassword"  
 android:textColor="#ffffff"**/>  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Please Select the sport you're interested in:"  
 android:textColor="#ffffff"**/>  
  
 <**RadioGroup  
 android:id="@+id/sport"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:orientation="vertical"**>  
  
 <**RadioButton  
 android:id="@+id/badminton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Badminton"  
 android:textColor="#ffffff"** />  
  
 <**RadioButton  
 android:id="@+id/basketball"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Basketball"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/boxing"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Boxing"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/cricket"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Cricket"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/football"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Football"  
 android:textColor="#ffffff"** />  
  
 <**RadioButton  
 android:id="@+id/golf"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Golf"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/hockey"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hockey"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/rugby"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Rugby"  
 android:textColor="#ffffff"** />  
  
 <**RadioButton  
 android:id="@+id/swimming"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Swimming"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/tennis"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Tennis"  
 android:textColor="#ffffff"** />  
 </**RadioGroup**>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter a short description about yourself!"  
 android:id="@+id/bio"  
 android:textColor="#ffffff"**/>  
 <**Button  
 android:id="@+id/register"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Register"  
 android:background="#FF80CBC4"  
 android:textColor="#ffffff"**/>  
</**LinearLayout**>  
</**ScrollView**>

### 3.8. Activity\_settings.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**ScrollView  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"**>  
<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/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"  
 tools:context="com.example.wk001764.finalproject.SettingsActivity"  
 android:background="#ff92cdc3"  
 android:orientation="vertical"**>  
  
 <**ImageView  
 android:id="@+id/profileImage"  
 android:layout\_width="200sp"  
 android:layout\_height="200sp"  
 android:layout\_gravity="center"  
 android:layout\_marginBottom="20sp"  
 android:src="@mipmap/defaultuserprofile"  
 android:textColor="#29C9BC"** />  
  
 <**EditText  
 android:id="@+id/name"  
 android:layout\_width="398dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="20sp"  
 android:background="@null"  
 android:hint="Name"** />  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/phone"  
 android:background="@null"  
 android:hint="Phone Number"  
 android:layout\_marginBottom="20sp"  
 android:inputType="phone"**/>  
 <**EditText  
 android:id="@+id/bio"  
 android:layout\_width="398dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="20sp"  
 android:background="@null"  
 android:hint="Bio"** />  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Please Select the sport you want to change to:"  
 android:textColor="#ffffff"**/>  
  
 <**RadioGroup  
 android:id="@+id/sport"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:orientation="vertical"**>  
  
 <**RadioButton  
 android:id="@+id/badminton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Badminton"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/basketball"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Basketball"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/boxing"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Boxing"  
 android:textColor="#ffffff"** />  
  
 <**RadioButton  
 android:id="@+id/cricket"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Cricket"  
 android:textColor="#ffffff"** />  
  
 <**RadioButton  
 android:id="@+id/football"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Football"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/golf"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Golf"  
 android:textColor="#ffffff"** />  
  
 <**RadioButton  
 android:id="@+id/hockey"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Hockey"  
 android:textColor="#ffffff"** />  
  
 <**RadioButton  
 android:id="@+id/rugby"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Rugby"  
 android:textColor="#ffffff"**/>  
  
 <**RadioButton  
 android:id="@+id/swimming"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Swimming"  
 android:textColor="#ffffff"** />  
  
 <**RadioButton  
 android:id="@+id/tennis"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:text="Tennis"  
 android:textColor="#ffffff"**/>  
 </**RadioGroup**>  
 <**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/confirm"  
 android:text="Save"**/>  
 <**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/back"  
 android:text="Back"**/>  
</**LinearLayout**>  
</**ScrollView**>

### 3.9. Item.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:background="#ff92cdc3"  
 android:layout\_gravity="center"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:paddingLeft="40sp"  
 android:paddingRight="40sp"  
 android:paddingTop="20sp"  
 android:paddingBottom="20sp"  
 android:outlineProvider="bounds"  
 android:clipToPadding="false"**>  
  
 <**android.support.v7.widget.CardView  
 android:id="@+id/cardView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:elevation="2dp"  
 app:cardCornerRadius="4dp"**>  
  
 <**LinearLayout  
 android:layout\_width="300dp"  
 android:layout\_height="350dp"  
 android:layout\_gravity="center"  
 android:orientation="vertical"**>  
 <**ImageView  
 android:id="@+id/image"  
 android:layout\_width="match\_parent"  
 android:layout\_height="267dp"  
 android:scaleType="centerCrop"** />  
 <**TextView  
 android:id="@+id/name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:paddingLeft="20sp"  
 android:textColor="@android:color/black"  
 android:textSize="30sp"  
 tools:text="hello"** />  
 <**TextView  
 android:id="@+id/bio"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:paddingLeft="20sp"  
 android:textColor="@android:color/black"  
 android:textSize="15sp"  
 tools:text="description"** />  
 </**LinearLayout**>  
 </**android.support.v7.widget.CardView**>  
</**LinearLayout**>

### 3.10. Item\_chat.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#ff92cdc3"**>  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="20sp"  
 android:id="@+id/container"**>  
 <**TextView  
 android:paddingBottom="20sp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Message"  
 android:textColor="#ffffff"  
 android:id="@+id/message"**/>  
 </**LinearLayout**>  
</**LinearLayout**>

### 3.11. Item\_match.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal"  
 android:padding="20sp"  
 android:background="#ff92cdc3"**>  
 <**ImageView  
 android:layout\_width="100sp"  
 android:layout\_height="100sp"  
 android:src="@mipmap/defaultuserprofile"  
 android:id="@+id/matchImage"**/>  
  
 <**LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:orientation="vertical"  
 android:paddingLeft="20sp"**>  
  
 <**TextView  
 android:id="@+id/matchName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:paddingBottom="20sp"  
 android:text="Name"** />  
  
 <**TextView  
 android:id="@+id/matchId"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:text="Match Id"** />  
  
 <**TextView  
 android:id="@+id/matchBio"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:text="bio"** />  
 </**LinearLayout**>  
</**LinearLayout**>

## Appendix 4 – failed registration

mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(RegistrationActivity.this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if(!task.isSuccessful()){

Toast.makeText(RegistrationActivity.this, "sign up error", Toast.LENGTH\_SHORT).show();

}else{

String userId = mAuth.getCurrentUser().getUid();

DatabaseReference currentUserDb = FirebaseDatabase.getInstance().getReference().child("Users").child(userId);

Map userInfo = new HashMap<>();

userInfo.put("name", name);

userInfo.put("sex", radioButton.getText().toString());

userInfo.put("profileImageUrl", "default");

currentUserDb.updateChildren(userInfo);

}

if(!task.isSuccessful()){

Toast.makeText(RegistrationActivity.this, "sign up error", Toast.LENGTH\_SHORT).show();

}else {

String userId = mAuth.getCurrentUser().getUid();

DatabaseReference currentUserDb = FirebaseDatabase.getInstance().getReference().child("Users").child(userId).child("Sport");

Map userInfo = new HashMap<>();

if(mBadminton.isChecked()) {

userInfo.put("badminton",true);

}else{

userInfo.put("badminton",false);

}

if(mBasketball.isChecked()) {

userInfo.put("basketball",true);

}else{

userInfo.put("basketball",false);

}

if(mBoxing.isChecked()) {

userInfo.put("boxing",true);

}else{

userInfo.put("boxing",false);

}

if(mCricket.isChecked()) {

userInfo.put("cricket",true);

}else{

userInfo.put("cricket",false);

}

if(mFootball.isChecked()) {

userInfo.put("football",true);

}else{

userInfo.put("football",false);

}

if(mGolf.isChecked()) {

userInfo.put("golf",true);

}else{

userInfo.put("golf",false);

}

if(mHockey.isChecked()) {

userInfo.put("hockey",true);

}else{

userInfo.put("hockey",false);

}

if(mRugby.isChecked()) {

userInfo.put("ruby",true);

}else{

userInfo.put("rugby",false);

}

if(mSwimming.isChecked()) {

userInfo.put("swimming",true);

}else{

userInfo.put("swimming",false);

}

if(mTennis.isChecked()) {

userInfo.put("tennis",true);

}else{

userInfo.put("tennis",false);

}

currentUserDb.updateChildren(userInfo);

}

}

});

}

});

}