National College of Ireland

BSc in Computing

2016/2017

Ciarán Byrne

13109740

ciaran.byrne2@student.ncirl.ie

Squad

Technical Report

National College of Ireland

Table of Contents

EXECUTIVE SUMMARY 6

1 INTRODUCTION 7

1.1 BACKGROUND 7
1.2 AIMS 8
1.3 TECHNOLOGIES 9

2 SYSTEM 10

2.1 REQUIREMENTS 10
2.1.1 FUNCTIONAL REQUIREMENTS 10
2.1.2 USE CASE DIAGRAM 11
2.1.3 REQUIREMENT 1: EDIT PLAYERS/ ADD PLAYERS 11
2.1.4 REQUIREMENT 2: PLAYER DROPS OUT 13
2.1.5 REQUIREMENT 4: MATCH MANAGEMENT 14
2.1.6 REQUIREMENT 6: REGISTER ACCOUNT 16
2.1.7 DATA REQUIREMENTS 18
2.1.8 USER REQUIREMENTS 18
2.1.9 ENVIRONMENTAL REQUIREMENTS 18
2.1.10 USABILITY REQUIREMENTS 19
2.2 NON-FUNCTIONAL REQUIREMENTS 19
2.2.1 PERFORMANCE/RESPONSE TIME REQUIREMENT 19
2.2.2 AVAILABILITY REQUIREMENT 19
2.2.3 RECOVERY REQUIREMENT 20
2.2.4 SECURITY REQUIREMENT 20
2.2.5 RELIABILITY REQUIREMENT 21
2.2.6 MAINTAINABILITY REQUIREMENT 21
2.2.7 PORTABILITY REQUIREMENT 21
2.2.8 EXTENDIBILITY REQUIREMENT 21
2.3 DESIGN AND ARCHITECTURE 21
2.4 IMPLEMENTATION 26

2.4.1 OUTLINE 26
2.4.2 PROJECT SET UP AND CONFIGURATION 27
2.4.3 AUTHENTICATION AND DATABASE SET UP 27
2.4.4 INPUT PHONE FRAGMENT 32
2.4.5 DISPLAY MATCH FRAGMENT 33
2.4.6 DATA MODEL 34
2.4.7 DATABASE STRUCTURE 35
2.4.8 CREATE MATCH 37
2.4.9 ADD PLAYERS 39
2.4.10 PUSH NOTIFICATIONS 46
2.5 TESTING 46
2.5.1 BLACK BOX TESTING 47
2.5.2 USER ACCEPTANCE TESTING 56
2.6 GRAPHICAL USER INTERFACE (GUI) 60
2.6.1 REGISTER 60
2.6.2 LOGIN 61
2.6.3 MAIN SCREEN 62
2.6.4 MATCH MANAGEMENT 63
2.6.5 EDIT PLAYERS 64
2.7 EVALUATION 65

3 CONCLUSIONS 67

4 FURTHER DEVELOPMENT 68

5 APPENDIX 69

5.1 PROJECT PROPOSAL 69
5.1.1 OBJECTIVES 69
5.1.2 BACKGROUND 70
5.1.3 TECHNICAL APPROACH 71
5.1.4 TECHNICAL DETAILS 72
5.1.5 EVALUATION 73
5.2 PROJECT PLAN 74

5.3 Monthly Journals 75
5.4 References 86

Executive Summary

The Squad Application allows people who play sports on a casual basis with friends and acquaintances to more easily organise matches amongst themselves through one convenient application. The app is designed to work on all modern Android devices and will be developed using the most up to date and innovative Android based technologies Firebase Server side management, Firebase Realtime Database, Firebase Authentication Management and Cloud Functions.

The project has been developed to demonstrate how a sports complex encouraging users to use this application can make the whole process of organising a match easier for the players. The application will provide the option for players to add users to their team, create regular matches, nominate regular players. The application will allow the administrator of a group to easily visually see if a user has accepted an invitation or not, with confirmed players highlighted in green on their list. Synchronicity of data is key to the functionality of Squad. When a player is added to an Admin's player list this data will synch over to that user's home page. Prompting the user to accept or decline the invitation. Once the invitation is accepted or declined this data synchs back to the player list on the Admin users device.

Businesses are always looking to take advantage of new convenient technology especially one that may separate them from the crowd. I believe the Squad app will provide a solution to a problem for regular users of a Five a side pitch/ facilities for Football and other sports. Managers and administration staff of sports complexes will seize the opportunity to push this application out to its users once it has been developed.

1 Introduction

1.1 Background

The idea for this application came to me when I noticed a common problem when trying to organise five a side football matches with friends. Every day thousands of people across the country have pitches booked to play on a regular or once off basis. They have their players ready and looking forward to a game with friends but at the last minute somebody drops out because they have to work late or some other reason. This can lead to a logistical problem for the person who is organising the game, people are depending on him/ her to ensure the numbers are there for the weekly match. Emails and texts are sent out to try and find a replacement but the they are busy with their own life and don't have the time to find a new player.

This problem is nobody's fault but can cause a lot hassle if you are the one organising the match. A simple fact is that life events get in the way and people will drop out on a very regular basis.

Squad aims to provide a solution to this problem, making it easier to find replacements when the inevitable happens and taking some of the micromanagement out of the equation. The application will prompt users with an invite and allow for the admin to easily add replacements onto the roster.

With the number of people playing Five a side football and other sports amongst friends on a casual basis there is great opportunity for an organization application such as Squad to become very popular. There are some applications that offer some of the features but none offer the whole package. Some attempts to solve the problem of players dropping out from sporting events have been attempted such as the 'LateCallUp' Twitter page, but this is quite labour intensive from the user's perspective and not personalized to the users group of friends meaning that there may be unwanted players joining the group. While this may suit some people, most groups playing five a side, tag rugby etc. want to exclusively play amongst their friends, work colleagues etc.

These solutions do not offer the ease of use for players and gyms/ sports complexes with all the features that Squad offers.

The plan for this application in the future would be to approach local sports complexes and encourage them and their users to adopt Squad as a system for their members to take advantage of. This would be attractive to them for the ease of use and it would set them apart from the other gyms and sporting complexes in the eyes of the player.

1.2 Aims

The main objective for this project are to provide an easy to use mobile and web application to end users that eases the organization process of sports events amongst friends.

The application will be a mobile first application and will aim to solve a common problem when trying to organise team based sporting events and will be mainly focused in the domain of Five a side football as a starting point. The reason for this being that five a side footballs huge popularity it makes for a logical starting point for the application's user base.

Users will sign up to the application in order to make the organization of matches easier amongst friends and team members. The application aims to mainly be a time saver for people involved in paying these matches and to remove a unnecessary needed level of communication when organising games. At the moment there is far too much back and forth between the organisers of a game and the participants whereas Squad will remove this and users will be able to easily see all the information needed. Admins will see a list of players, who has confirmed and who has not and players who have been invited to participate will see the details of the match and who has invited them, time and date etc. allowing them to make a decision on whether they can play or not. This functionality extends to the fact that users will be able rescind invitations removing them from the player list and users will be at any time able to update their status as to whether they are playing or not.

Although initially focused on five a side football the application can be used to organise any number of events e.g. Tag rugby, Badminton etc. Basically any sporting events between friends where you need to meet up and play but sometimes there

The application will provide a secure login and sign up, utilizing sign in features through social media and interaction with social media through the Google platform as well as allowing for secure login with the usual email and password functionality.

Consistent information being displayed on all user's devices is a priority, and the data will need to be consistently available and quick read and writes will be needed for this.

1.3 Technologies

The application will be a native Android application and will work on most smartphones based on the Android platform. It will be built using Android Studio and Java with XML for the front end, utilizing the latest Android technologies and practices with the Firebase suite in the back end.

Firebase Cloud Messaging will be utilized to handle the push notifications being sent out to the users. NodeJS will be used to handle the server side functionality of these push notifications using Firebase Cloud Functions. The regular backend will be expanded up on with Firebase Realtime database and Authentication used to manage multiple user's information vital to the app like login details and which team they are grouped in with as well as their credentials on the database. Using this technology means the app will be able to handle multiple users, and groups in a modern dynamic fashion and allow for the transfer of information and data to be processed more efficiently while maintaining information will be consistent amongst all users.

2 System

In this section I will outline the stages that were needed in order to build the Squad application. The requirements process will be explained below with descriptions on system architecture and the other required components that make up the system as a whole. Testing and the Graphical user Interface will be shown and elaborated on to give an idea of how the system looks and the lengths that were taken to test the systems robustness. I will outline how I utilised the Android Studio, Java and Firebase to fully realise the application.

2.1 Requirements

The requirements for my application have not been altered much from my original Requirements Document, the only change being that it is now being built in Android.

All requirements for this application should be verifiable to the requirements specified in this document. When the project is close to completion the application will be tested by a group of testers to evaluate whether the system is simple to use, free of errors, passes all the requirements listed in this document and accessible to the applications potential user base.

Following the system being demonstrated to the system testers a realistic test scenario of attempting to organise a match using the application will go ahead to demonstrate the application is capable of being used for this purpose free of errors. Once these tests have been made it will have demonstrated that it can be adopted by a sports facility or five a side pitch to be distributed to its members as a means to organise and book matches.

2.1.1 Functional requirements

Functional requirements give an indication of what is needed from the system and what goals it should achieve for the user specifically. Below I will outline these goals and expectations through use case diagrams and explanations.

2.1.2 Use Case Diagram

Below is a Use Case diagram of the entire system once the application has been loaded.

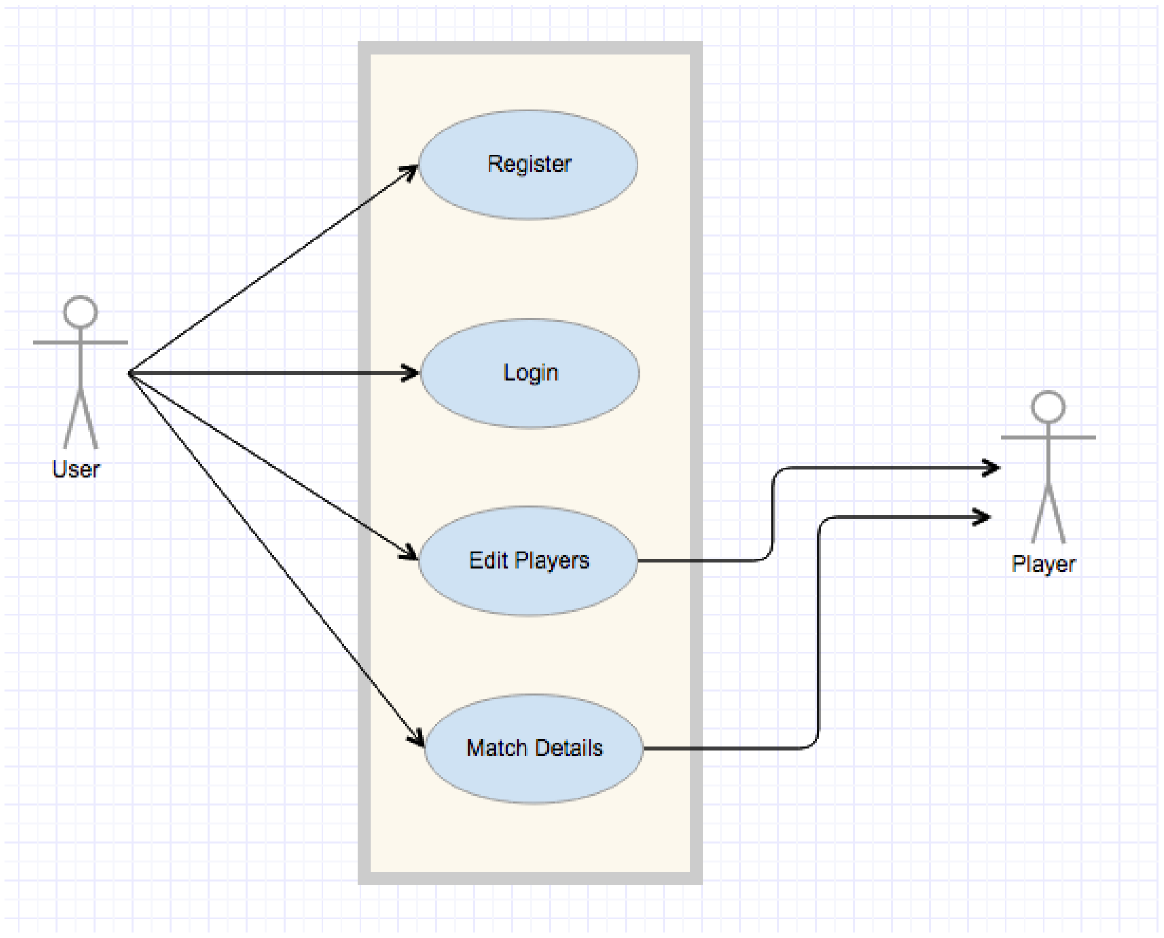


Figure 1 – Use Case Diagram of system

2.1.3 Requirement 1: Edit Players/ Add Players

2.1.3.1 Description & Priority

This use case occurs after the user has logged themselves in and they have chosen the ‘Edit Players option from the main menu. This is the most critical element to the application as without it the rest of the functionality would not be available to the user.

2.1.3.2 Use Case

Edit Players

Scope

The scope of this use case is to allow the user to add themselves and invite other users to a match.

Description

This use case describes the creation of a team within the application

Flow Description

Precondition

The system is launched and is running, the user has navigated to the Edit Players section.

Activation

This use case starts when a user selects 'Edit Players from the menu.

Main flow

Main flow

1. The system identifies the button has been pressed and generates the manage page for that user.
2. The User is presented with a form to complete with details for the squad.
3. The user selects players from the contact search picker which brings up the user's contact list
4. The user selects a contact to add
5. The system adds the name and number to a text view below
6. The user confirms this is the user to add and selects the add button
7. The system validates the form selection and it is added to the database with a unique identifier linked to this user
8. Squad details are sent to the database and users are added

Alternate flow

A1 : Unable to validate squad selection

1. The system recognises that the squad selection is invalid for following reasons. Not a valid contact, blank contact

2. The system notifies the user of the problem and asks to check input and re submit or abandon changes
3. The use case continues at position 4 of the main flow

Termination

The system shows the user the updated list

Post condition

The system goes into a wait state

2.1.4 Requirement 2: Player drops out

2.1.4.1 Description & Priority

This use case describes the process a nominated player goes through when they need to take themselves off of the player list for the next upcoming game. This is a critical use case as it demonstrates the main functionality of the application.

2.1.4.2 Use Case

Player drop out

Scope

The scope of this use case is to allow a player to drop out of the player list for the next upcoming match.

Description

The user should be able to remove themselves from the player list for the next upcoming match and the system should asynchronously update all data correctly

Flow Description

Precondition

The system has launched and the user is presented with a menu screen.

Activation

This use case starts when a User marked as a Player starts the application and the system checks they have been added to a match. The Display Match fragment appears prompting the user to confirm their attendance

Main flow

1. The system identifies the User has been added as a player on another user’s player list
2. The Player confirms they want to remove themselves from the player list for this match by changing the switch option and pressing update response button
3. The system removes this player from the list and updates the asynchronously across all devices

Alternate flow

A1 : All Subs decline invitation

1. The Sub declines the invitation by pressing decline button
2. The System marks this sub as declined across all devices
3. System awaits these inputs to be selected.

Termination

The system brings user back to the menu page.

Post condition

System notes time has gone passed scheduled game time and sets next game as next scheduled game. System enters wait state.

2.1.5 Requirement 4: Match Management

2.1.5.1 Description & Priority

This use case describes the process the user marked as ‘Admin goes through to manage a details for their squad. Updating times or day details. This is a high priority use case as the user should be able to edit these options after creation.

2.1.5.2 Use Case

Team management

Scope

The scope of this use case is to allow a user marked as admin to manage their Squad of players

Description

This use case describes the process the user goes through when managing details of a Squad

Flow Description

Precondition

The system has launched and the user is at the My Match Details menu.

Activation

This use case starts when an Admin user selects the My Match Details option

Main flow

1. The system identifies the ‘button has been pressed by the user and then displays the details for this match to the user
2. The User sees the details and edits them as needed (including day of match, etc.). The user presses enter
3. The system recognises the okay button has been pressed and displays a confirmation screen with details entered, system prompts user to okay if details are correct. (See A1: Input details are not okay)
4. These details are synched to all added players on match
5. The system recognises the okay button has been selected and sends the data with linked to the user’s Squad to the database as well as the Matches section of the application.

Termination

The system redirects the user to the main menu page.

Post condition

The system goes into a wait state

2.1.6 Requirement 6: Register account

2.1.6.1 Description & Priority

This use case describes the process the user goes through in order to register an account on the application. This requirement is of high priority as you will need an account in order to use the application.

2.1.6.2 Use Case

Register account

Scope

The scope of this use case is to allow a user to register an account

Description

This use case describes the user inputting their details into the system to register an account.

Actors

User

Flow Description

Precondition

The system has launched and the user is presented with a start-up main menu.

Activation

This use case starts when a User presses the Register button

Main flow

1. The system identifies the Register button has been selected and generates a new account menu.
2. The User is given the option to register via e-mail or via social media accounts (Twitter, Facebook) and selects e-mail.(See A1: User selects to register via social media)
3. The system generates the register new account page.
4. The user fills in the required fields such as username, email and password.
5. The user submits these details they have entered (See E1: Incorrect details entered)
6. The system validates the details and sends the registered account to the database.
7. The system redirects the user to the main menu

Alternate flow

A1 : User selects to register via social media

1. The User selects to register via social media
2. The system generates the option to register with Twitter or Facebook
3. The user selects their preferred social media
4. The system sends a request via API to the social media provider and validates the users account
5. The use case continues at position 5 of the main flow

Exceptional flow

E1 : Incorrect details entered

1. The user enters incorrect details such as an email address that already exists
2. The system validates the inputted data but cannot finish the process.
3. The system refreshes the display and highlights the errors to the user
4. User corrects the errors and submits

Termination

The system confirms the user is registered and redirects them to the main menu.

Post condition

The system goes into a wait state

2.1.7 Data requirements

The data requirements for Squad application will be outlined in this section. The application relies on interaction with the Firebase Realtime Database. Which will host the application's database and sore user and system data.

- **Data:** The data in the application will be structured in a NoSQL JSON format as this is that Firebase backend is most easily managed with. The structure will be broken down into users, players, groups, and matches.
- **Data storage:** The application must allow a user to add other users to their team, book a time and date for the match on the database, this will be stored in a NoSQL JSON structure much like the core structure of the application on Firebase.
- **Data Security:** Firebase implements a flexible and effective security solution that developers can take advantage of to ensure there are no unauthorized users and data maintains a high level of integrity. It is proven to be a highly secure solution and resistant to malicious attacks.

2.1.8 User requirements

The user will be required to be able to make use of the following in order to use the application:

- **Android Phone:** The user will need an Android phone to use and access the application

- **Internet Access:** The user will need to connect to internet either through a 3G/4G connection or Wi-Fi in order to make full use of the application.

2.1.9 Environmental requirements

For me to develop this application I will have to ensure that I have certain pieces of Software and Hardware to hand in order to complete the task such as an up to date laptop that is suitable for Android Development

- **Laptop**: An up to date laptop with sufficient memory and RAM

- **Android Phone**: To ensure the application is working correctly I will need to test it on an Android Phone

- **Android Studio:** To code and develop the app

2.1.10 Usability requirements

2.2 Non-Functional Requirements

2.2.1 Performance/Response time requirement

The system will be required to show a quick response performance/ response time and the user should not feel like there is a delay to their actions. The application will need access to the internet either via wireless connection or mobile data and this there should not be a delay in accessing the data required. The application should be launched immediately when the user selects it from the phones menu. The welcome message should be displayed for a maximum of 5 seconds while the application loads the system in the background and it should take no longer than 10 seconds for the application to start up from when it is launched.

The application will be built with a server taking care of the back end data, there will be stress testing performed in the final stages of development to ascertain the upper limit of users on the app at the same time and other speed/ capacity tests. Automated testing will also be used to stress and push the system to its limits. These methods will allow for some conclusions to be drawn and how best to adapt the server with high amounts of traffic.

2.2.2 Availability requirement

The main question that will arise when it comes to availability will be down to the server's functionality and internet connection from the client side.

In order to reduce any potential server down time regular updates will need to be applied to the server and performance stress testing. The updates should be applied at low usage times for the application so as to avoid downtime for customers. Server optimization may need to be applied in order to get the best from the servers.

If the internet is unavailable from the client's device a popup message will displayed to them warning them that an active internet connection is needed to synch the application.

When the internet is not available Firebase caches data writes via data time stamp that first writes to the users local storage and as soon as internet access is granted again the write takes place with the correct information and in the write order as the time stamp ensures consistency of all data.

2.2.3 Recovery requirement

The application will need to be tested against eventualities such as hardware faults and crashes, as well as connectivity faults. The application should be able to recover from hardware and software failures. Nearing the end of the development lifecycle testing will be carried out to determine how well the system can recover from such failures. The following tests will be carried out:

Start application and while starting up, restart the phone. Open the application and ensure that data integrity is maintained.

Submit or make changes to data to the database and while still running shut down connectivity, data integrity should be maintained.

2.2.4 Security requirement

As users will be logging in with their personal information it is imperative that security is made a priority. The passwords will be hashed and authenticated using recognised and trusted third party resources to ensure that security is not compromised. The social media login feature should also provide another level of security for users, as the social media logins are proven to have a high level of security. Database integrity will be maintained and monitored using current best practices.

Firebase security rules are highly customisable and robust and I have developed with best practices in mind for the Squad application, using the recommended guidelines.

2.2.5 Reliability requirement

Application standards in terms of reliability are of a very high standard in the modern marketplace, as a result any application that is developed must be reliable and dependable or users may become frustrated and will not use the app. Towards the end of the development lifecycle testing will be done to ensure that the application's code is reliable and does not crash or fail during any actions. The application will be optimised for a reliable user experience and in the event of an unlikely error it will be handled by the system, informing the user and redirecting them to the main page.

2.2.6 Maintainability requirement

The system will be coded using modern modular practices in order to ensure maintainability.

2.2.7 Portability requirement

The application will be for Android devices, aimed primarily at Android smart phones and should work on the majority of Android phones on the market. Some very old models (5 years plus) may have difficulty but it is not expected that many users will be using these older models as we are specifically targeting the 20 -35 demographic.

2.2.8 Extendibility requirement

With a potential for a high amount of users concurrently the system will need to be able to handle a large volume of users. Due to the nature of the application there may not be a large volume of data being used as for most users the app will only be used with a couple of interactions. Regardless automated testing and real world testers will be used to load the server side with requests. The servers will need to be extendible to deal with a potential growing user base

2.3 Design and Architecture

This section outlines the components and sub systems that will be used in building the application.

The system will use a Firebase server back end and database storing the data in the cloud. One advantage to this is that the Firebase API will provide security and the ability to sync data in real time allowing for very fast data reads and writes with all data maintaining consistency and accurate reads available just one second after a data write has taken place.

Firebase Realtime maintains all the data in cloud storage on Firebase servers, this data is accessed by registered clients and verified. The data is then read by the users phone as JSON file. Firebase listeners provide a means that when they are set up in the code the listener is automatically awaiting any changes on the database removing the need (as in SQL databases) to send a read request to the server, server responds, then server sends information. Firebase simply allows for the data to be asynchronously read by the client as soon as there is a changed made to the data on the server.

The Firebase backend suite is part of the Firebase Cloud Platform and is easily integrated into the Android ecosystem and are natural choices no matter the size of the app.

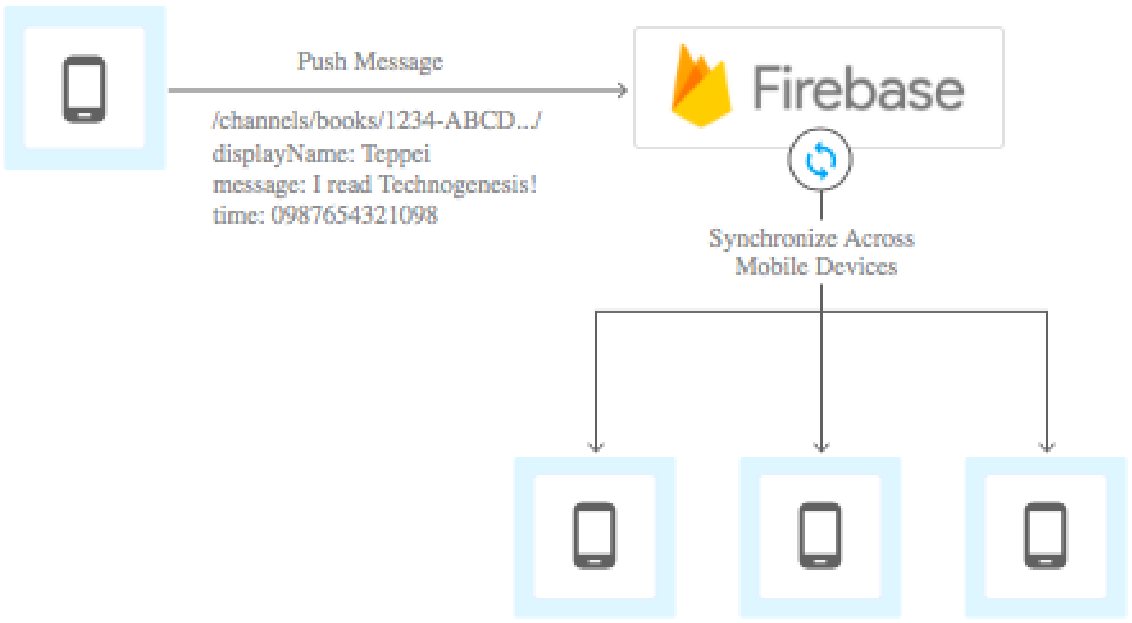
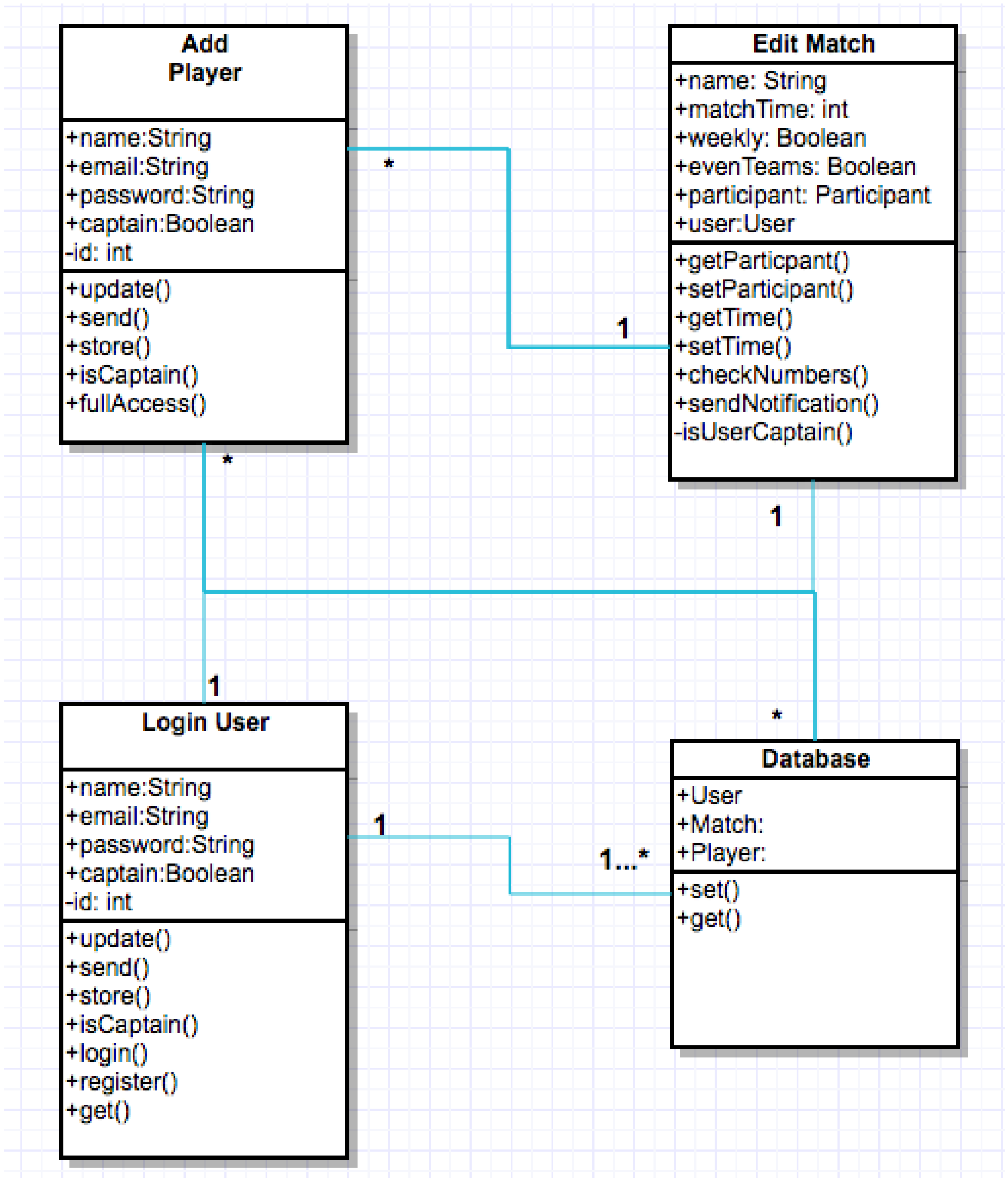

Push Message
/channels/books/1234-ABCD.../
displayName: Teppei
message: I read Technogenesis!
time: 0987654321098
Firebase
Synchronize Across Mobile Devices

Figure 2 – Firebase Architecture

Source: (Cloud.google.com, 2017)


Add Player
+name:String
+email:String
+password:String
+captain:Boolean
-id: int
+update()
+send()
+store()
+isCaptain()
+fullAccess()
Edit Match
+name: String
+matchTime: int
+weekly: Boolean
+evenTeams: Boolean
+participant: Participant
+user:User
+getParticpant()
+setParticipant()
+getTime()
+setTime()
+checkNumbers()
+sendNotification()
-isUserCaptain()
Login User
+name:String
+email:String
+password:String
+captain:Boolean
-id: int
+update()
+send()
+store()
+isCaptain()
+login()
+register()
+get()
Database
+User
+Match:
+Player:
+set()
+get()
*
1
*
1
1
*
1
1...*

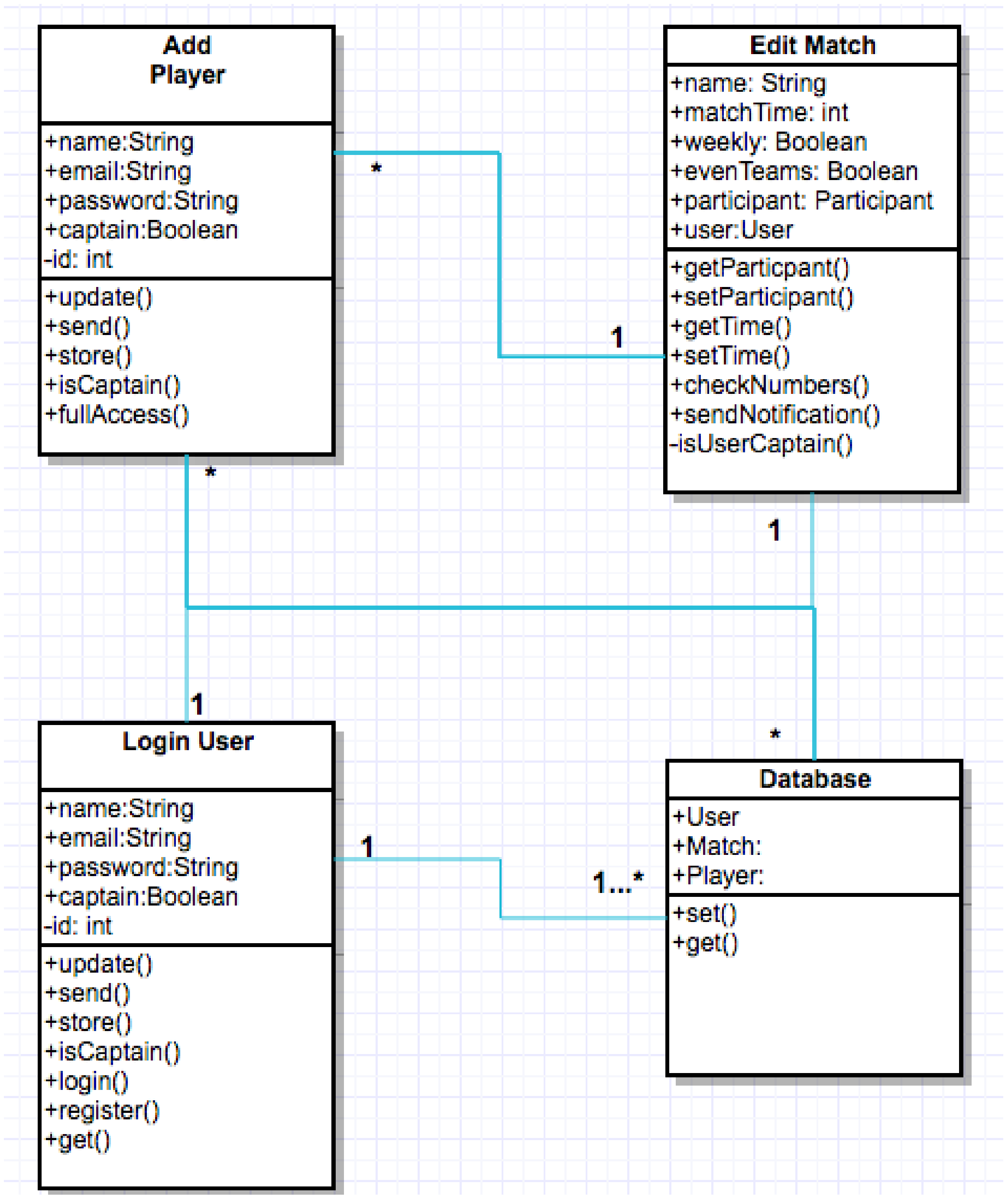


Figure 3 - Class Diagram

2.4 Implementation

2.4.1 Outline

In this section I will outline and describe how the project was developed, the main pieces of functionality will be described and the code will be presented and explained throughout.

The majority of the work for my project was done using Android Studio, the programming language used in Android development is a derivative of Java and all the Android APIs are called primarily using Java. All the functions, data structures and classes needed for the app to run were developed in Android Studio as well as the front end GUI.

For my backend I used Firebase Realtime Database as my database for the project. Firebase uses a NoSQL database which allows for very fast reads and writes to the database. It is an emerging technology and this was among the reasons I was attracted to using it for my project. It also allows for huge amounts of scalability and is proven to be up to the task in real world with major companies such as SkyScanner and Shazam relying on it for its backend technology.

Firebase database automatically synchs the data between devices as soon as changes are made, and users see these changes instantly and if there is no internet connection changes are stored locally until a connection is made, maintain consistency with time stamped push ids. For a real world example, you only need to look at SkyScanner where flight data is constantly changing (prices, times etc.) to see that there are major benefits to this type of database.

I used Firebase for my User Authentication as it integrates very well with the Realtime Database.

The same is true for the last piece of the puzzle for Squad, push notifications are needed to notify Users of various events triggered in the app. For this functionality I used Firebase Cloud Functions to handle the triggers and Firebase Push Notifications to send the notifications. In order to take advantage of the Cloud triggers I had to implement a Firebase Cloud Server using NodeJS.

Below I will go in to more detail on the above.

2.4.2 Project set up and configuration

When setting up my project I had to consider a couple of factors, such as the minimum version of Android required to run on a user’s phone. I chose SDK 21 as it runs on most Android devices but still has a lot of the most modern features in the current version 25. As I am using Firebase for many features I had to set up a Firebase account so I could use the Firebase console to manage my database and authentication etc.

2.4.3 Authentication and Database set up

For authentication as mentioned above I implemented the Firebase Authentication flow, the user is able to sign up/ sign in using either their standard e-mail address or use the Google authentication which automatically registers the user with their Gmail account.

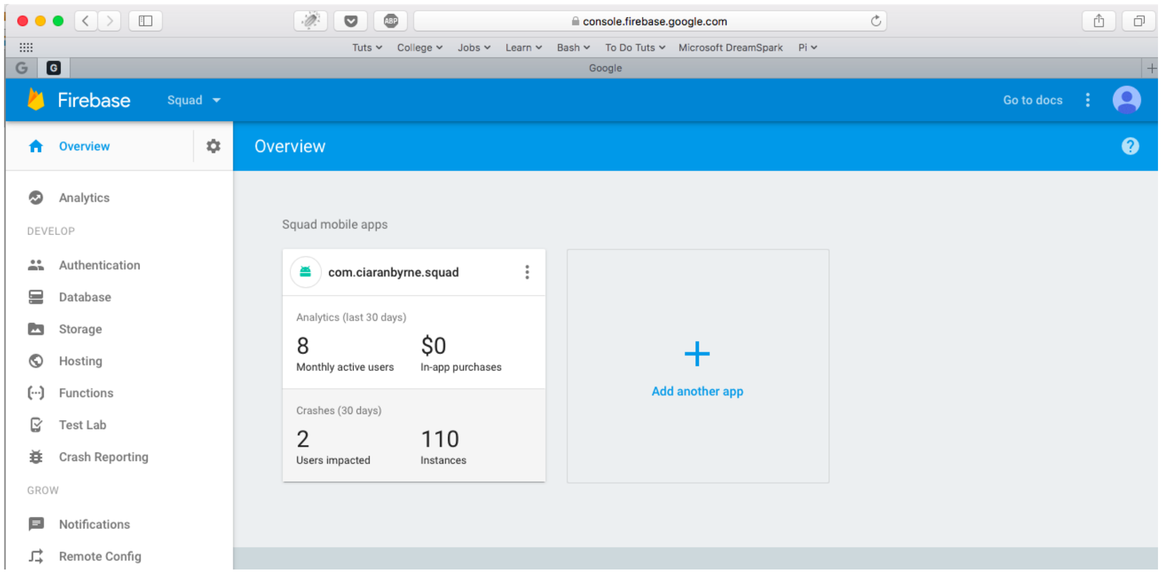
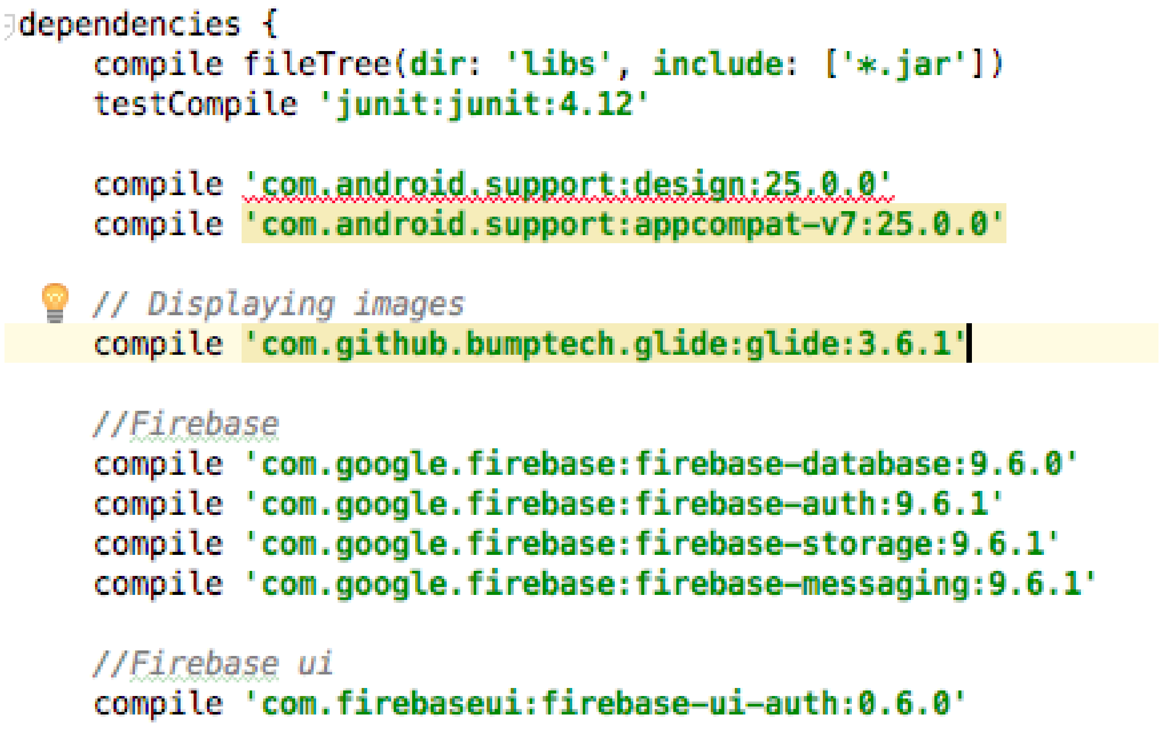


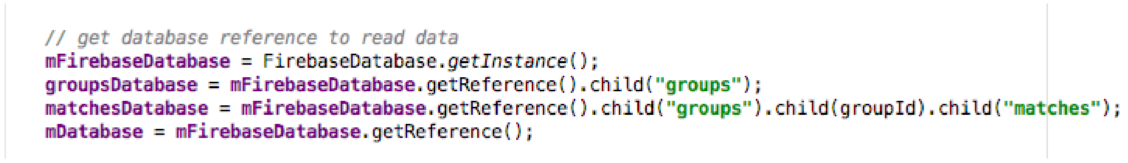
Figure 4 – Firebase console available at https://console.firebase.google.com/

Once I set up a Firebase account I enabled the Database and Authentication functionality as these are the two main areas I needed for the back end of my Application. Going back to Android Studio I then ensured I had the correct dependencies in my manifest file and gradle build files. I had to be careful here as each version of Firebase correlates to a specific version of Android and you need to input dependencies for each piece of functionality you are using for Firebase e.g. database, authentication etc.



```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'

    compile 'com.android.support:design:25.0.0'
    compile 'com.android.support:appcompat-v7:25.0.0'

    // Displaying images
    compile 'com.github.bumptech.glide:glide:3.6.1'

    //Firebase
    compile 'com.google.firebase:firebase-database:9.6.0'
    compile 'com.google.firebase:firebase-auth:9.6.1'
    compile 'com.google.firebase:firebase-storage:9.6.1'
    compile 'com.google.firebase:firebase-messaging:9.6.1'

    //Firebase ui
    compile 'com.firebaseui:firebase-ui-auth:0.6.0'
```

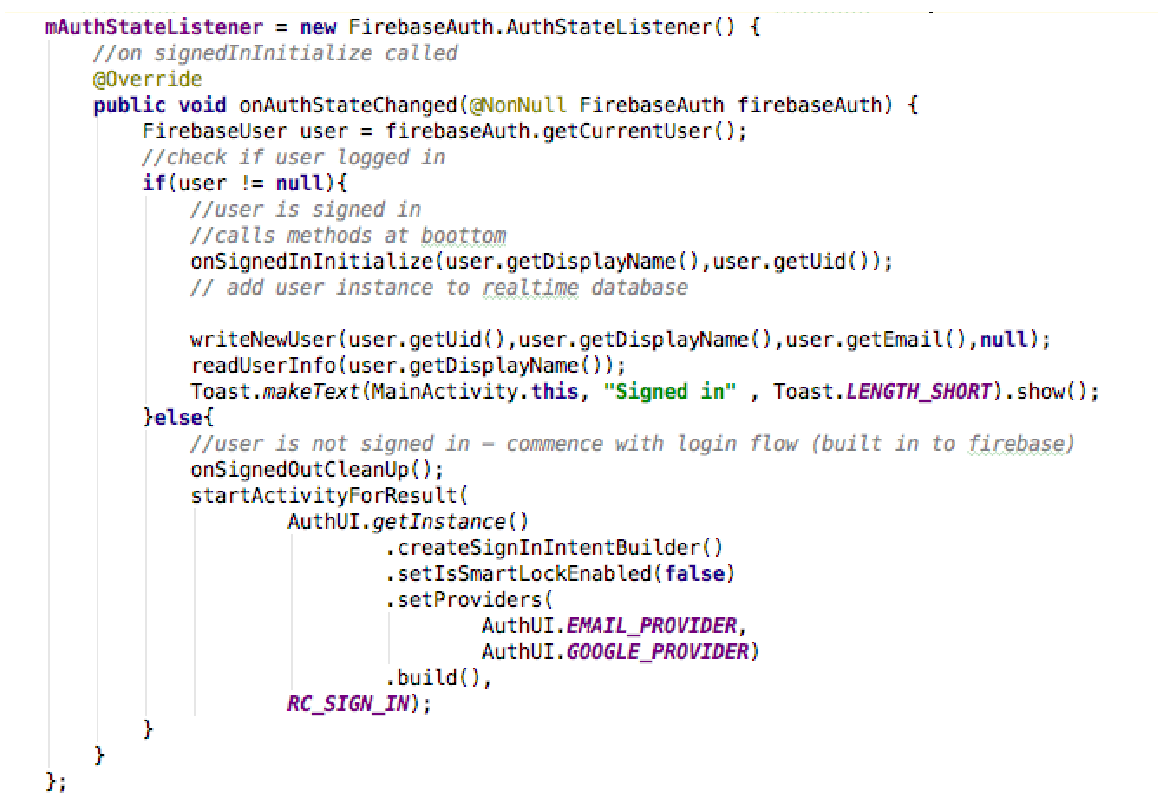
Figure 5 - Firebase dependencies in build.gradle



```
// get database reference to read data
mFirebaseDatabase = FirebaseDatabase.getInstance();
groupsDatabase = mFirebaseDatabase.getReference().child("groups");
matchesDatabase = mFirebaseDatabase.getReference().child("groups").child(groupId).child("matches");
mDatabase = mFirebaseDatabase.getReference();
```

Figure 6 - Firebase dependencies must be set in each activity

For Authentication I opened up my MainActivity.java file as this is the first Activity to be activated once the application starts and where the authentication process would start up. What the authentication process is essentially doing is providing once screen and set of actions to a user if they are logged in to the app (Main page and normal functionality) and another to a user who is not logged in (Register/ Sign in page).

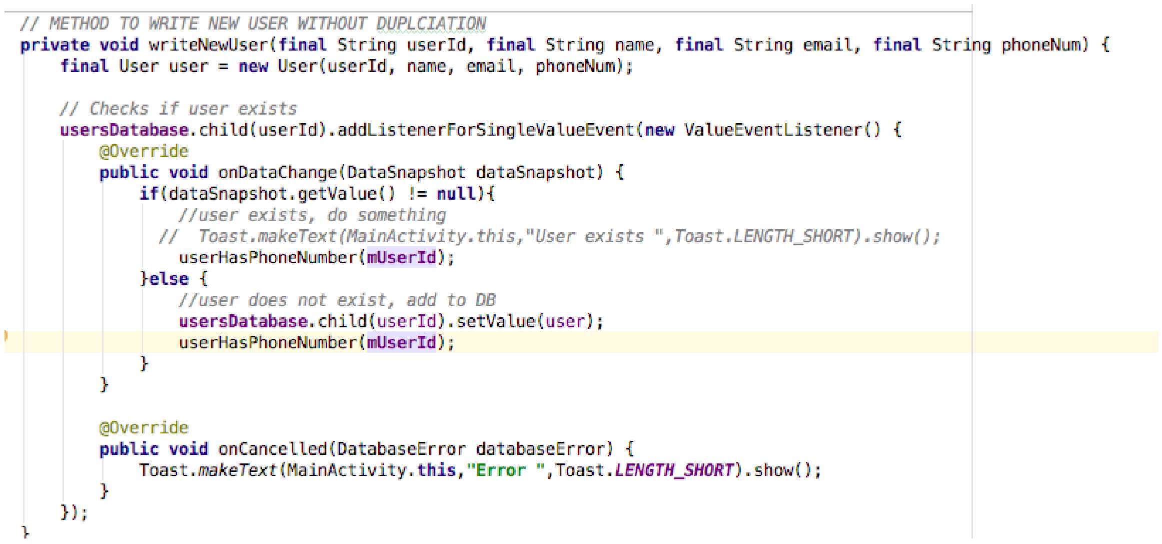


```java
mAuthStateListener = new FirebaseAuth.AuthStateListener() {
    //on signedInInitialize called
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser user = firebaseAuth.getCurrentUser();
        //check if user logged in
        if(user != null){
            //user is signed in
            //calls methods at boottom
            onSignedInInitialize(user.getDisplayName(),user.getUid());
            // add user instance to realtime database

            writeNewUser(user.getUid(),user.getDisplayName(),user.getEmail(),null);
            readUserInfo(user.getDisplayName());
            Toast.makeText(MainActivity.this, "Signed in" , Toast.LENGTH_SHORT).show();
        }else{
            //user is not signed in - commence with login flow (built in to firebase)
            onSignedOutCleanUp();
            startActivityForResult(
                    AuthUI.getInstance()
                            .createSignInIntentBuilder()
                            .setIsSmartLockEnabled(false)
                            .setProviders(
                                    AuthUI.EMAIL_PROVIDER,
                                    AuthUI.GOOGLE_PROVIDER)
                            .build(),
                    RC_SIGN_IN);
        }
    }
};
```

Figure 7 - Standard login and Authentication process as per Firebase documentation

As we can see in this code these two cases are looked after, the app checks to see if it is a logged in user and if so, starts the regular app flow if not it starts the sign in flow for the user. Providing the user with the option to login with standard email login or use the Google social media auto login.

We can also see there is a call to a method 'writeNewUser', this was developed as while Firebase keeps record of the users it is done so completely separate from the database and these values cannot be accessed beyond the sign in flow. A large part of Squad is about User management and Group management so I needed to write the user details separately to the database.



```java
// METHOD TO WRITE NEW USER WITHOUT DUPLCIATION
private void writeNewUser(final String userId, final String name, final String email, final String phoneNum) {
    final User user = new User(userId, name, email, phoneNum);

    // Checks if user exists
    usersDatabase.child(userId).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            if(dataSnapshot.getValue() != null){
                //user exists, do something
              //  Toast.makeText(MainActivity.this,"User exists ",Toast.LENGTH_SHORT).show();
                userHasPhoneNumber(mUserId);
            }else {
                //user does not exist, add to DB
                usersDatabase.child(userId).setValue(user);
                userHasPhoneNumber(mUserId);
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            Toast.makeText(MainActivity.this,"Error ",Toast.LENGTH_SHORT).show();
        }
    });
}
```

Figure 8 –MainActivity.java - writeNewUser()

The above method does a couple of things, first takes in the values for the User and of that User does not exist on the database it adds an entry for them. It also searches by phone number in the Players node of the database to see if this user has input their phone number yet. This is needed as in order to consolidate the Players and Users database I had to make it searchable by phone number. With Firebase it is not possible to get the Users phone number details which is needed in order to search to see if that user is a Player in another Users' Squad. Without this check it would not be possible for users to be notified about matches they may have been added to by other Users.

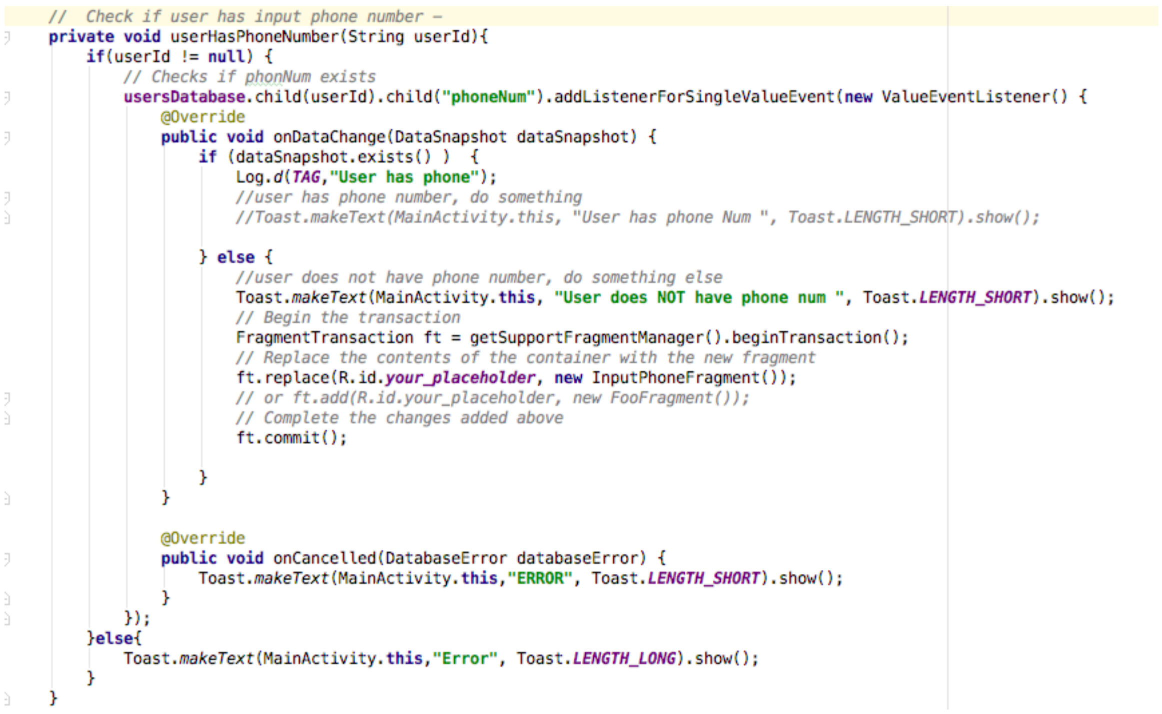

```
//  Check if user has input phone number -
private void userHasPhoneNumber(String userId){
    if(userId != null) {
        // Checks if phonNum exists
        usersDatabase.child(userId).child("phoneNum").addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists() )  {
                    Log.d(TAG,"User has phone");
                    //user has phone number, do something
                    //Toast.makeText(MainActivity.this, "User has phone Num ", Toast.LENGTH_SHORT).show();

                } else {
                    //user does not have phone number, do something else
                    Toast.makeText(MainActivity.this, "User does NOT have phone num ", Toast.LENGTH_SHORT).show();
                    // Begin the transaction
                    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
                    // Replace the contents of the container with the new fragment
                    ft.replace(R.id.your_placeholder, new InputPhoneFragment());
                    // or ft.add(R.id.your_placeholder, new FooFragment());
                    // Complete the changes added above
                    ft.commit();

                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                Toast.makeText(MainActivity.this,"ERROR", Toast.LENGTH_SHORT).show();
            }
        });
    }else{
        Toast.makeText(MainActivity.this,"Error", Toast.LENGTH_LONG).show();
    }
}
```

Figure 9 - MainActivity.java – userHasPhoneNumber()

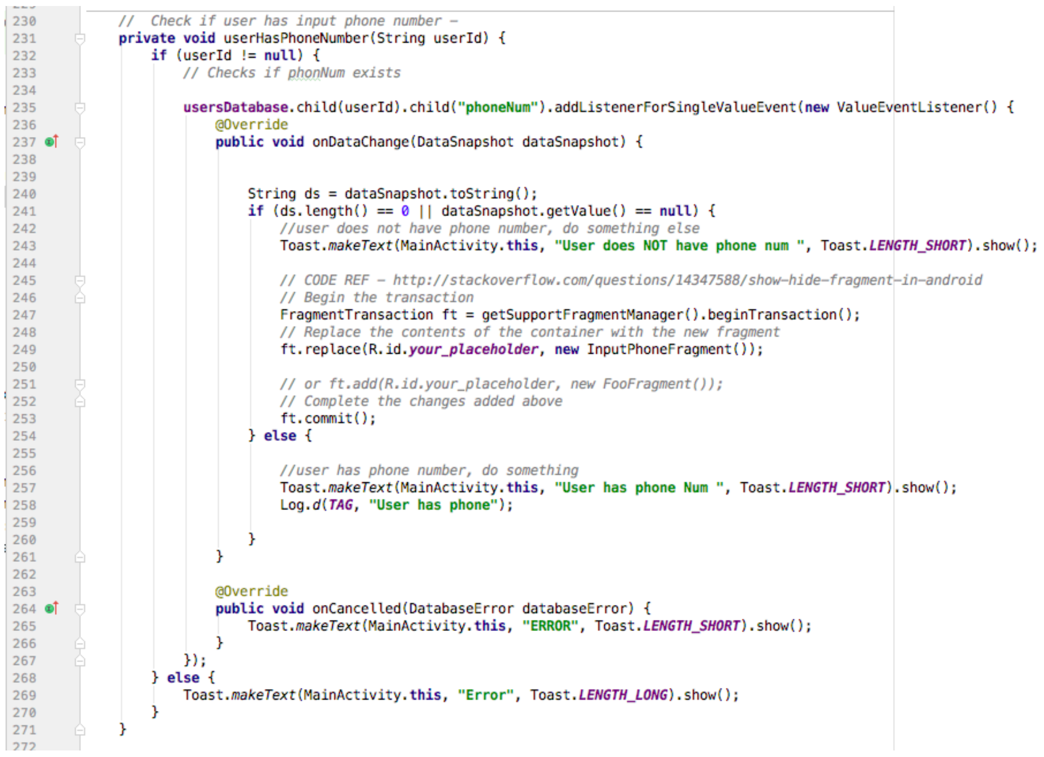
If the outcome of this method is that the User has not yet entered a phone number in the database they will be prompted with a Fragment to enter their phone number on the main page. Once entered the user will not see this again.

The last part of the sign in flow is to allow the user to sign out, this sign out functionality tears down the main app flow scaffold and brings the user back to the Sign in flow.

There is also an option on the sign in page for users that have forgotten their password which will notify them via email of their password.

2.4.4 Input Phone Fragment

In order to consolidate the data between the Players and Users nodes in the database we need a common way of referencing both. As Players are added from the contact picker on a user’s phone we will have their phone number which can be used as a reference. We cannot retrieve phone number details from the standard Firebase API so a check has been implemented that checks to see if a user’s phone number exists on the database when they login. If the phone number does not exist the InputPhoneNumFragment is called and prompts the user to enter in their phone details.



```java
    //  Check if user has input phone number -
    private void userHasPhoneNumber(String userId) {
        if (userId != null) {
            // Checks if phonNum exists

            usersDatabase.child(userId).child("phoneNum").addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {


                    String ds = dataSnapshot.toString();
                    if (ds.length() == 0 || dataSnapshot.getValue() == null) {
                        //user does not have phone number, do something else
                        Toast.makeText(MainActivity.this, "User does NOT have phone num ", Toast.LENGTH_SHORT).show();

                        // CODE REF - http://stackoverflow.com/questions/14347588/show-hide-fragment-in-android
                        // Begin the transaction
                        FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
                        // Replace the contents of the container with the new fragment
                        ft.replace(R.id.your_placeholder, new InputPhoneFragment());

                        // or ft.add(R.id.your_placeholder, new FooFragment());
                        // Complete the changes added above
                        ft.commit();
                    } else {

                        //user has phone number, do something
                        Toast.makeText(MainActivity.this, "User has phone Num ", Toast.LENGTH_SHORT).show();
                        Log.d(TAG, "User has phone");

                    }
                }

                @Override
                public void onCancelled(DatabaseError databaseError) {
                    Toast.makeText(MainActivity.this, "ERROR", Toast.LENGTH_SHORT).show();
                }
            });
        } else {
            Toast.makeText(MainActivity.this, "Error", Toast.LENGTH_LONG).show();
        }
    }
```

Figure 10 - MainActivity - check to see if user has input their phone number - Fragment transaction code ref - (Stack Overflow, 2017)

The phone fragment is called and covers the users main screen asking for them to input their phone number. Once the user inputs the number it is saved in the database, and a check takes place to see if the user’s phone number exists in the groups node. If so the match details are copied to the user’s node in the database to allow for consistent reads and writes to the database across all users.

2.4.5 Display Match Fragment

When the user logs in another check is done to see if that user has been added to any matches. If the user has been added as a player by another user then the details of this match appear on their home screen inviting the user to confirm their attendance. The invitation gives all the match details including who the admin of the group is, what time it is at and what day it is on. The code used is similar to the check used in Figure 10.

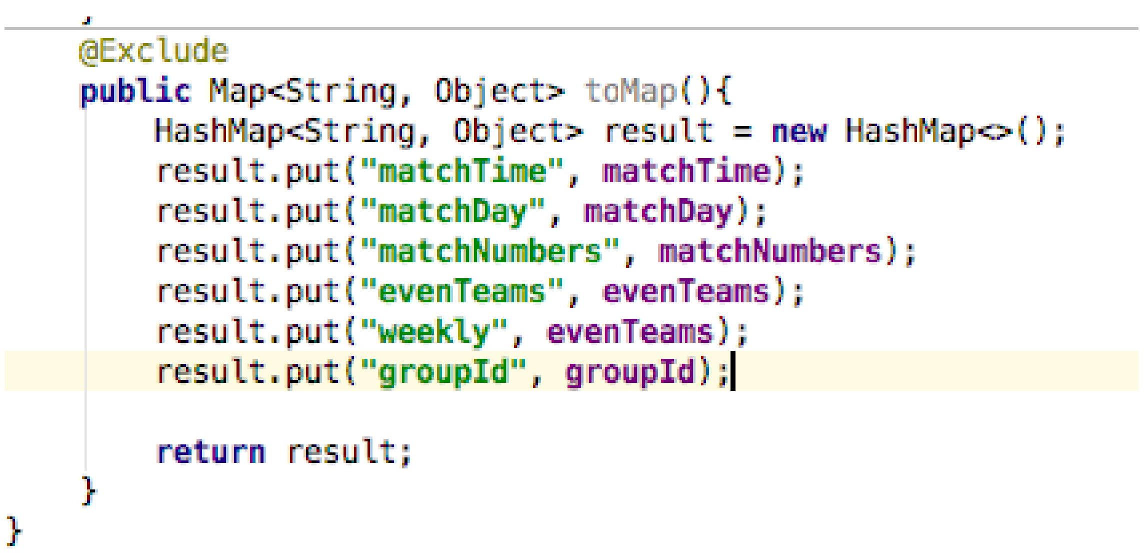
The user enters in their response and this asynchronously updates the administrator's players list as well as in the user's own database node ensuring consistency across the application. The user once confirmed updates to a green colour on the admin players list once they have confirmed their attendance.

2.4.6 Data Model

The Models used for the project necessitated that Java Classes be implemented for each, I needed one for the User, Group, Player, and Match. Below I will outline the relevant parts of these.

These classes told the object the data types needed for each object and provided a means to access them with Getter and Setter methods. These were called anytime an object was written or read from the database e.g. when a new User is written to the database.

The Match and the Player Classes were a little more complex as they require HashMaps to store data in lists and to allow for them to be updated dynamically.

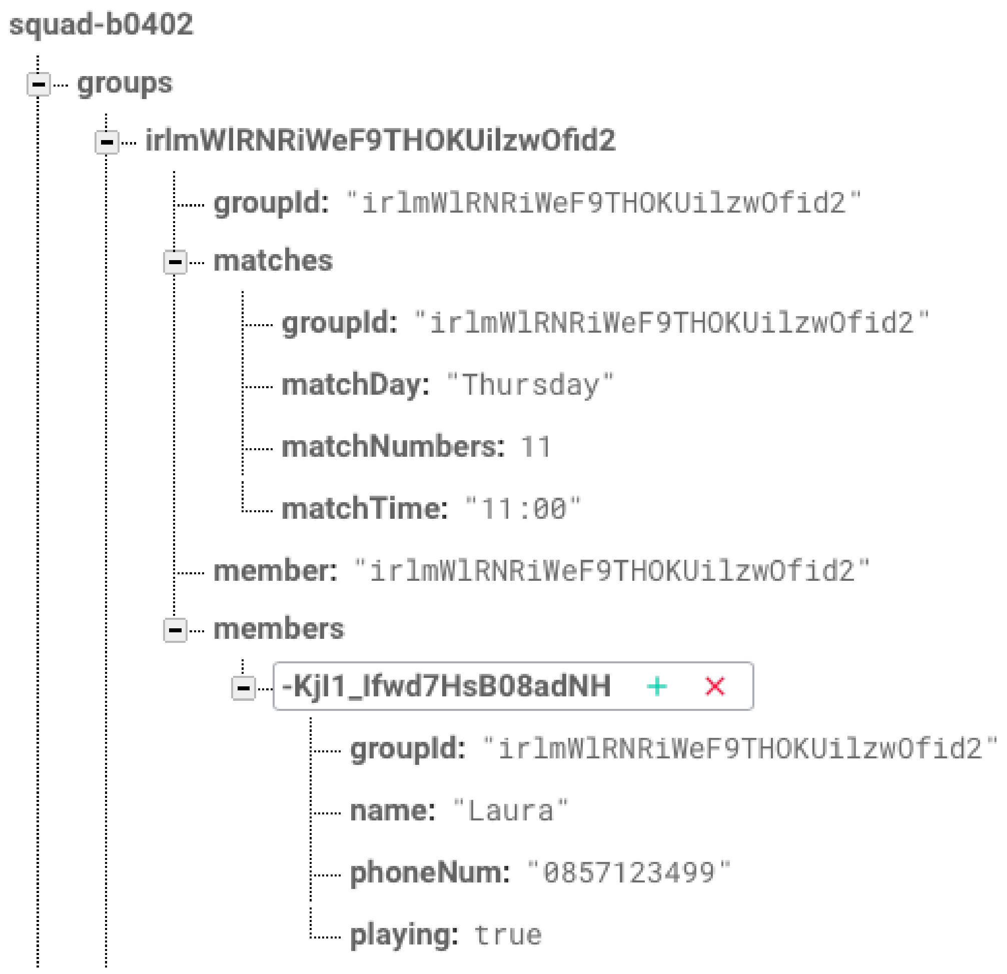


```
    @Exclude
    public Map<String, Object> toMap(){
        HashMap<String, Object> result = new HashMap<>();
        result.put("matchTime", matchTime);
        result.put("matchDay", matchDay);
        result.put("matchNumbers", matchNumbers);
        result.put("evenTeams", evenTeams);
        result.put("weekly", evenTeams);
        result.put("groupId", groupId);

        return result;
    }
}
```

Figure 11 - HashMap in Match.java

2.4.7 Database Structure

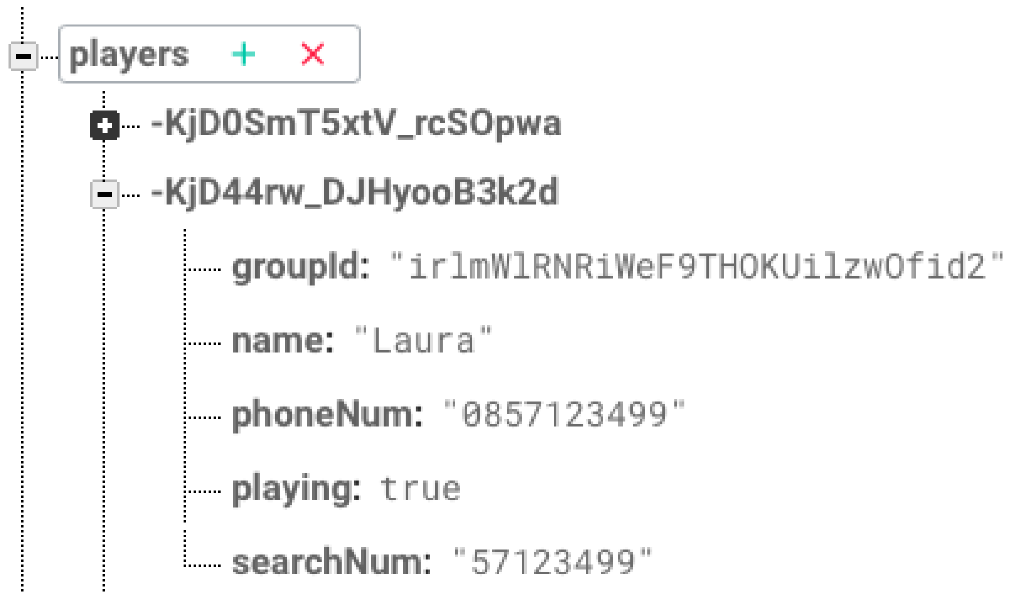
As I am using a NoSQL database the structure needs to be different to a regular relational database. In parts some duplications is needed to ensure quicker reads and writes and as per Firebase documentation a flattened data structure is preferable. This is because the database is essentially a large JSON file, and I have limited the amount of nodes that need to be read in order to access the requested node. To facilitate this some de-normalization of my data has taken place with as mentioned above duplications.



```
squad-b0402
  groups
    irlmWlRNRiWeF9THOKUilzwOfid2
      groupId: "irlmWlRNRiWeF9THOKUilzwOfid2"
      matches
        groupId: "irlmWlRNRiWeF9THOKUilzwOfid2"
        matchDay: "Thursday"
        matchNumbers: 11
        matchTime: "11:00"
      member: "irlmWlRNRiWeF9THOKUilzwOfid2"
      members
        -Kjl1_lfwd7HsB08adNH
          groupId: "irlmWlRNRiWeF9THOKUilzwOfid2"
          name: "Laura"
          phoneNum: "0857123499"
          playing: true
```

Figure 12 - Group structure, showing match and members details

Here we can see a group node with the id, match details and the members of this group.



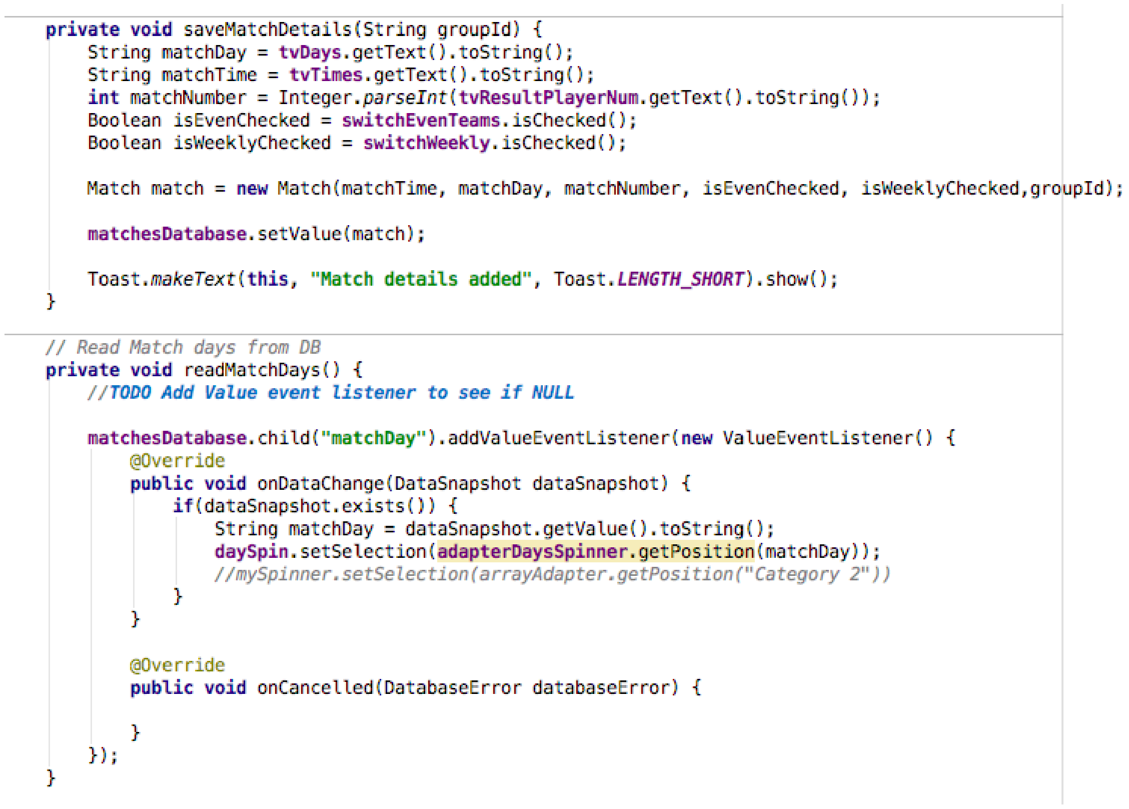
```
players
-KjD0SmT5xtV_rcSOpwa
-KjD44rw_DJHyooB3k2d
    groupId: "irlmWlRNRiWeF9THOKUilzwOfid2"
    name: "Laura"
    phoneNum: "0857123499"
    playing: true
    searchNum: "57123499"
```

Figure 13 - Players node with details

In the above we can see some duplication, this means that in order to check what group this user is a part of we do not need to go as deep in the tree structure of the JSON file instead of groups/group id/ members and the searching we can simply go to player's/ user id to see what group the player is associated with thus ensure faster access times.

2.4.8 Create match

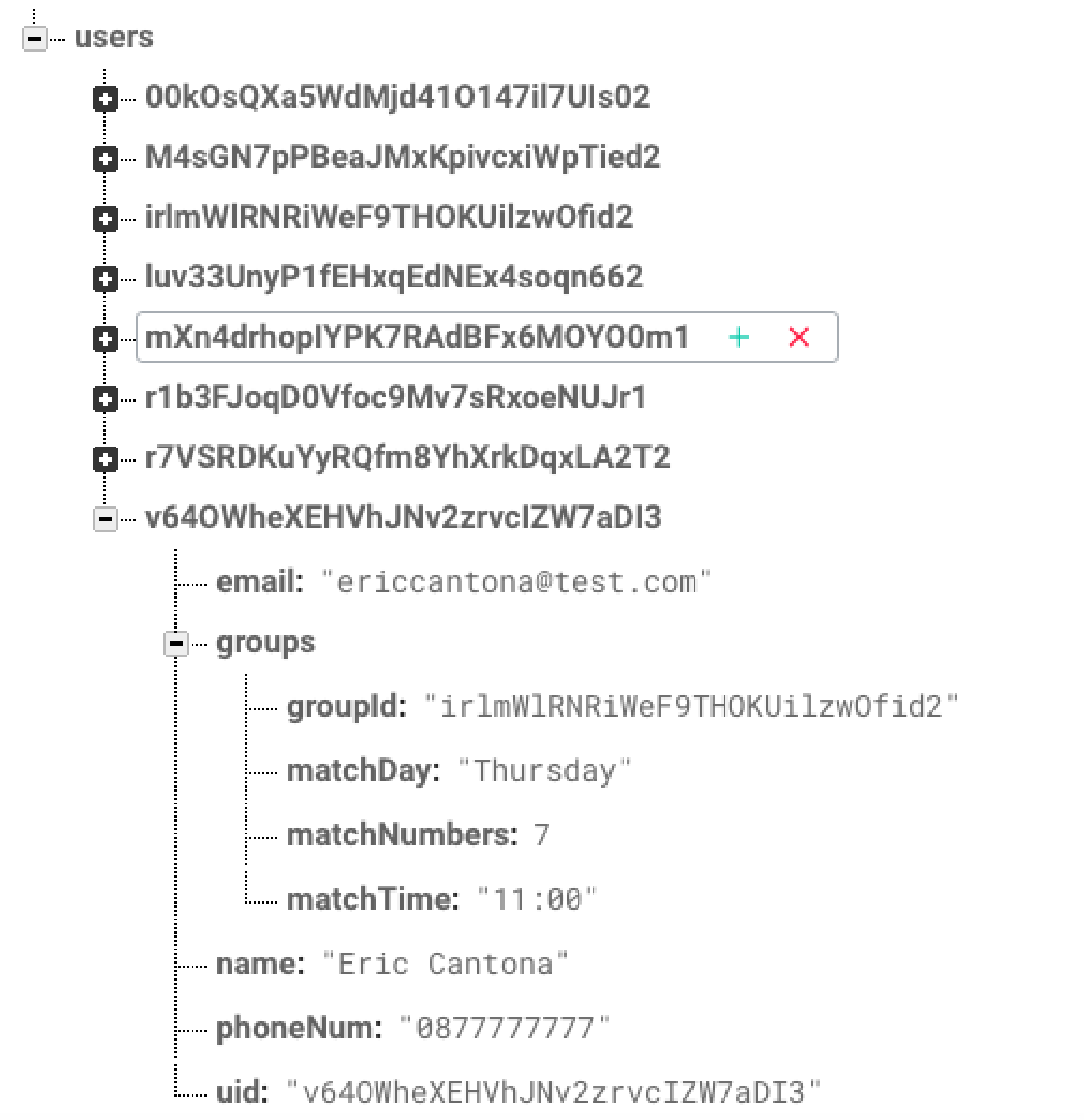
Here is where a logged in user creates and edits the match that they are an admin of, each user can be an admin of one match only. The admin sets the time, day, player numbers, weekly, even teams for the match here. Once saved the details are updated in the database for that Group that the user is assigned to.



```java
private void saveMatchDetails(String groupId) {
    String matchDay = tvDays.getText().toString();
    String matchTime = tvTimes.getText().toString();
    int matchNumber = Integer.parseInt(tvResultPlayerNum.getText().toString());
    Boolean isEvenChecked = switchEvenTeams.isChecked();
    Boolean isWeeklyChecked = switchWeekly.isChecked();

    Match match = new Match(matchTime, matchDay, matchNumber, isEvenChecked, isWeeklyChecked,groupId);

    matchesDatabase.setValue(match);

    Toast.makeText(this, "Match details added", Toast.LENGTH_SHORT).show();
}

// Read Match days from DB
private void readMatchDays() {
    //TODO Add Value event listener to see if NULL

    matchesDatabase.child("matchDay").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()) {
                String matchDay = dataSnapshot.getValue().toString();
                daySpin.setSelection(adapterDaysSpinner.getPosition(matchDay));
                //mySpinner.setSelection(arrayAdapter.getPosition("Category 2"))
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
}
```

Figure 14 - Methods save and read data for developed for each input

These details are written to the database and if a Player a part of this is found in the Users node the data is copied to that users node in the database.



```
users
  00kOsQXa5WdMjd410147il7UIs02
  M4sGN7pPBeaJMxKpivcxiWpTied2
  irlmWlRNRiWeF9THOKUilzwOfid2
  luv33UnyP1fEHxqEdNEx4soqn662
  mXn4drhoplYPK7RAdBFx6MOYO0m1
  r1b3FJoqD0Vfoc9Mv7sRxoeNUJr1
  r7VSRDKuYyRQfm8YhXrkDqxLA2T2
  v64OWheXEHVhJNv2zrvcIZW7aDI3
    email: "ericcantona@test.com"
    groups
      groupId: "irlmWlRNRiWeF9THOKUilzwOfid2"
      matchDay: "Thursday"
      matchNumbers: 7
      matchTime: "11:00"
    name: "Eric Cantona"
    phoneNum: "0877777777"
    uid: "v64OWheXEHVhJNv2zrvcIZW7aDI3"
```

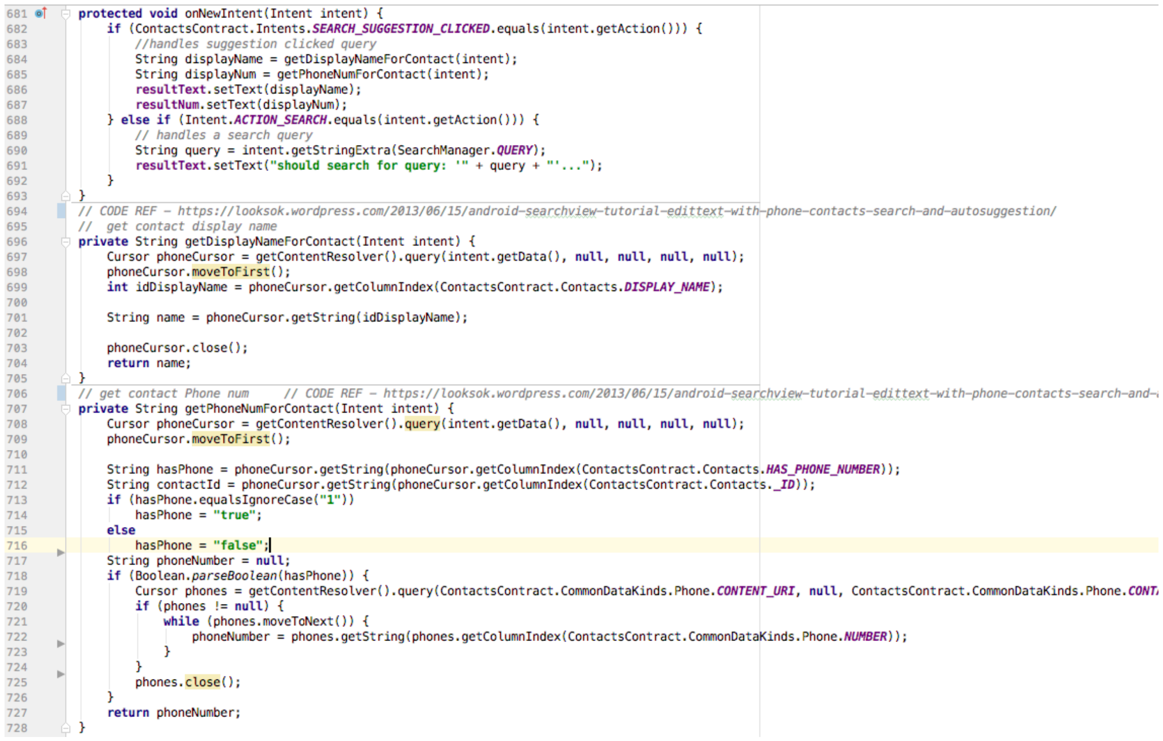
Figure 15 - Group copied to Users node

2.4.9 Add players

This Activity allows logged in users to add players to a match that they are an admin of, this match is in full control of the admin user and they can add / remove users at any time.

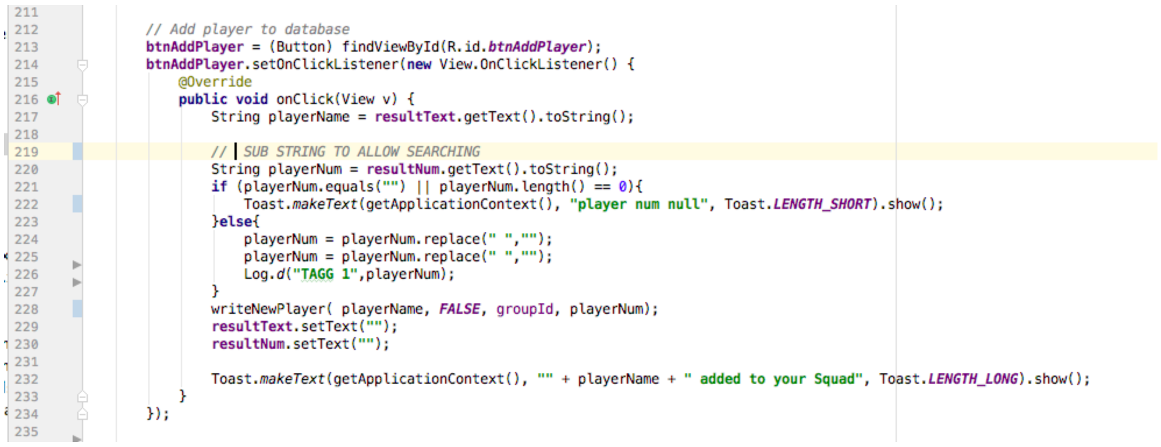
The Player class and PlayerAdapter are used to add Players, the app then checks to see if this Player is a User of Squad and if they are updates this Users home page with the details of the match and the option to opt in or out of playing. This is indicated in the database by the ‘playing’ field and updates the Admin on which players have currently confirmed they are playing or not. All players added are by default ‘playing’ unless they opt out.

When adding a player, the user is presented with a screen to select a contact from their phone, this was a challenging piece of the app to achieve and proved very difficult. I had to get permissions to access the SQL database on the phone and look through a user’s phone contact list. From here a user selects a Contact to add, it is displayed in a TextView with name and number and the user presses add. This then writes the Player and number to a TextView so the user can confirm the details are correct and then writes the data to both the Players node, and the Groups members node and also calls the function to search by phone number to see if a player exists in the user’s node of the database. If they do the group details are then copied to this user’s list and presented on his home screen.



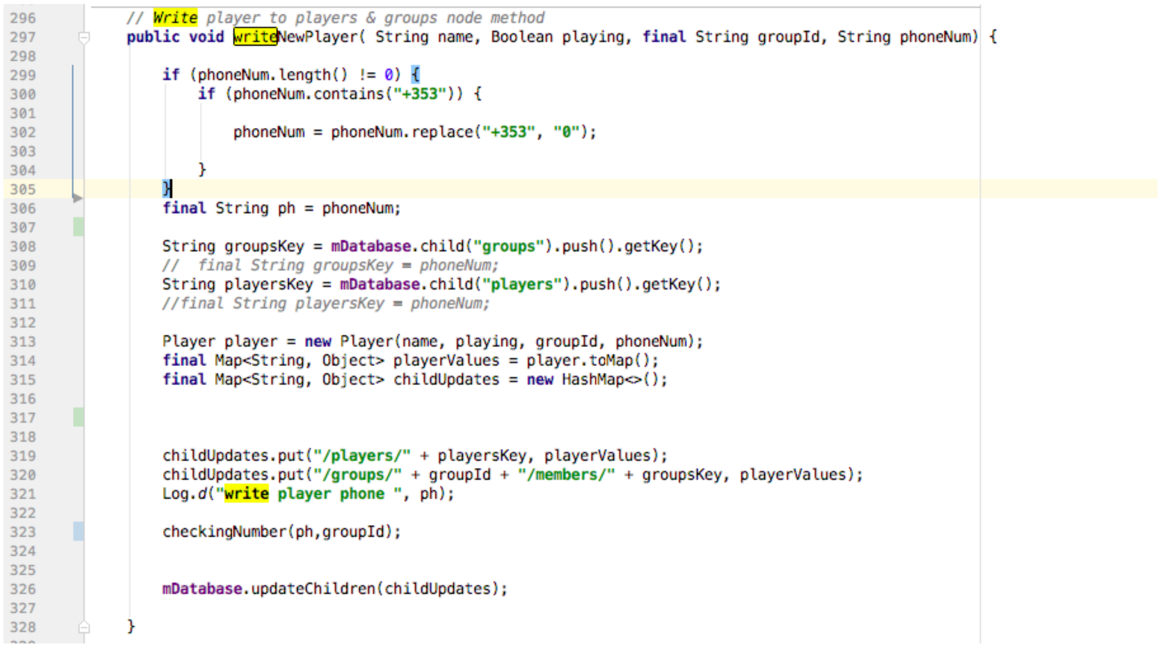
```java
protected void onNewIntent(Intent intent) {
    if (ContactsContract.Intents.SEARCH_SUGGESTION_CLICKED.equals(intent.getAction())) {
        //handles suggestion clicked query
        String displayName = getDisplayNameForContact(intent);
        String displayNum = getPhoneNumForContact(intent);
        resultText.setText(displayName);
        resultNum.setText(displayNum);
    } else if (Intent.ACTION_SEARCH.equals(intent.getAction())) {
        // handles a search query
        String query = intent.getStringExtra(SearchManager.QUERY);
        resultText.setText("should search for query: '" + query + "'...");
    }
}
// CODE REF - https://looksok.wordpress.com/2013/06/15/android-searchview-tutorial-edittext-with-phone-contacts-search-and-autosuggestion/
//  get contact display name
private String getDisplayNameForContact(Intent intent) {
    Cursor phoneCursor = getContentResolver().query(intent.getData(), null, null, null, null);
    phoneCursor.moveToFirst();
    int idDisplayName = phoneCursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);

    String name = phoneCursor.getString(idDisplayName);

    phoneCursor.close();
    return name;
}
// get contact Phone num     // CODE REF - https://looksok.wordpress.com/2013/06/15/android-searchview-tutorial-edittext-with-phone-contacts-search-and-
private String getPhoneNumForContact(Intent intent) {
    Cursor phoneCursor = getContentResolver().query(intent.getData(), null, null, null, null);
    phoneCursor.moveToFirst();

    String hasPhone = phoneCursor.getString(phoneCursor.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER));
    String contactId = phoneCursor.getString(phoneCursor.getColumnIndex(ContactsContract.Contacts._ID));
    if (hasPhone.equalsIgnoreCase("1"))
        hasPhone = "true";
    else
        hasPhone = "false";
    String phoneNumber = null;
    if (Boolean.parseBoolean(hasPhone)) {
        Cursor phones = getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null, ContactsContract.CommonDataKinds.Phone.CONT
        if (phones != null) {
            while (phones.moveToNext()) {
                phoneNumber = phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
            }
        }
        phones.close();
    }
    return phoneNumber;
}
```

Figure 16 – EditPlayersActivity.java Methods for Contact Picker - adapted from Code Ref - (Android SearchView Tutorial 2017)



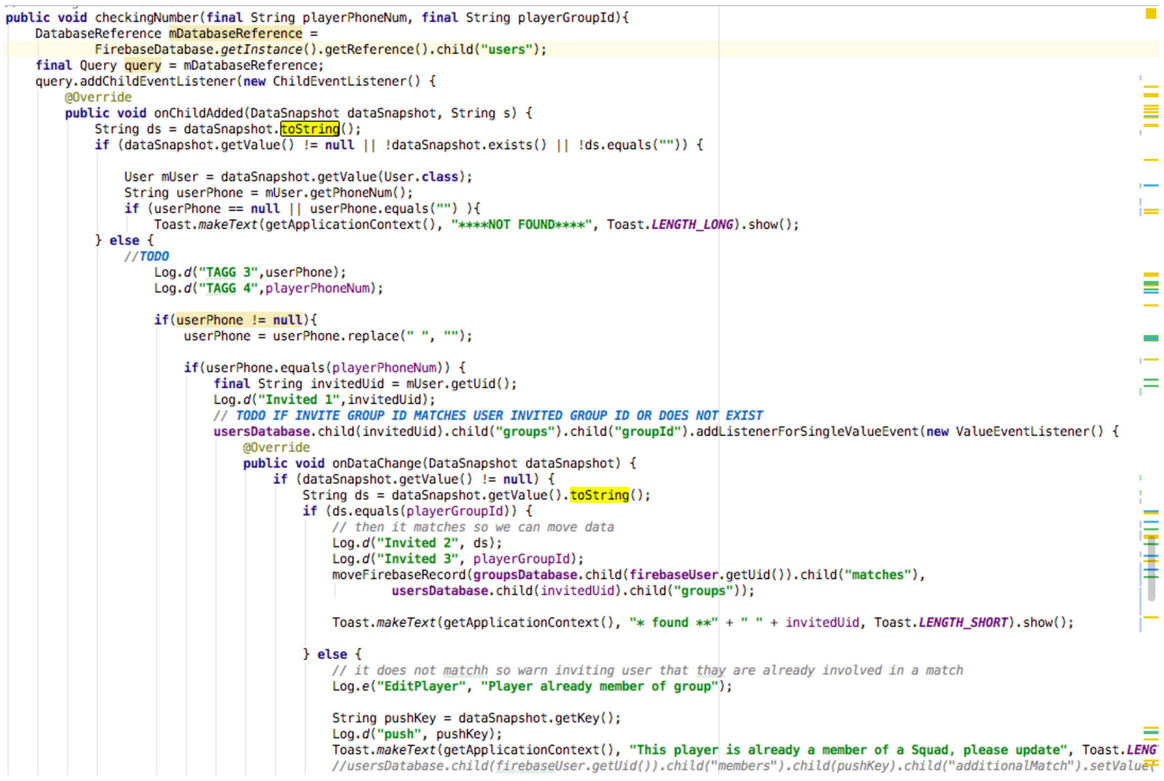
```java
        // Add player to database
        btnAddPlayer = (Button) findViewById(R.id.btnAddPlayer);
        btnAddPlayer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String playerName = resultText.getText().toString();

                // SUB STRING TO ALLOW SEARCHING
                String playerNum = resultNum.getText().toString();
                if (playerNum.equals("") || playerNum.length() == 0){
                    Toast.makeText(getApplicationContext(), "player num null", Toast.LENGTH_SHORT).show();
                }else{
                    playerNum = playerNum.replace(" ","");
                    playerNum = playerNum.replace(" ","");
                    Log.d("TAGG 1",playerNum);
                }
                writeNewPlayer( playerName, FALSE, groupId, playerNum);
                resultText.setText("");
                resultNum.setText("");

                Toast.makeText(getApplicationContext(), "" + playerName + " added to your Squad", Toast.LENGTH_LONG).show();
            }
        });
```

Figure 17 - EditPlayersActivity.java - on click listener to call method to write new player



```java
    // Write player to players & groups node method
    public void writeNewPlayer( String name, Boolean playing, final String groupId, String phoneNum) {

        if (phoneNum.length() != 0) {
            if (phoneNum.contains("+353")) {

                phoneNum = phoneNum.replace("+353", "0");

            }
        }
        final String ph = phoneNum;

        String groupsKey = mDatabase.child("groups").push().getKey();
        //  final String groupsKey = phoneNum;
        String playersKey = mDatabase.child("players").push().getKey();
        //final String playersKey = phoneNum;

        Player player = new Player(name, playing, groupId, phoneNum);
        final Map<String, Object> playerValues = player.toMap();
        final Map<String, Object> childUpdates = new HashMap<>();


        childUpdates.put("/players/" + playersKey, playerValues);
        childUpdates.put("/groups/" + groupId + "/members/" + groupsKey, playerValues);
        Log.d("write player phone ", ph);

        checkingNumber(ph,groupId);


        mDatabase.updateChildren(childUpdates);

    }
```

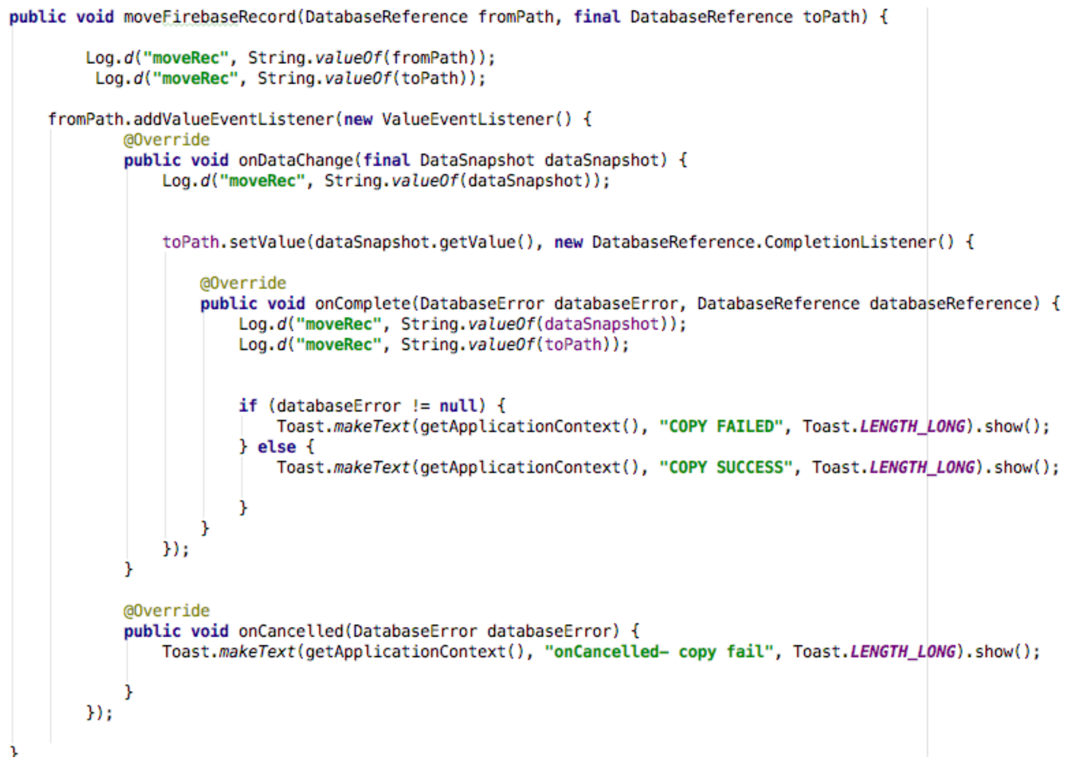
Figure 18 – EditPlayersActivity.java - Write player to players and groups node



```java
public void checkingNumber(final String playerPhoneNum, final String playerGroupId){
    DatabaseReference mDatabaseReference =
            FirebaseDatabase.getInstance().getReference().child("users");
    final Query query = mDatabaseReference;
    query.addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            String ds = dataSnapshot.toString();
            if (dataSnapshot.getValue() != null || !dataSnapshot.exists() || !ds.equals("")) {

                User mUser = dataSnapshot.getValue(User.class);
                String userPhone = mUser.getPhoneNum();
                if (userPhone == null || userPhone.equals("") ){
                    Toast.makeText(getApplicationContext(), "****NOT FOUND****", Toast.LENGTH_LONG).show();
            } else {
                //TODO
                    Log.d("TAGG 3",userPhone);
                    Log.d("TAGG 4",playerPhoneNum);

                    if(userPhone != null){
                        userPhone = userPhone.replace(" ", "");

                        if(userPhone.equals(playerPhoneNum)) {
                            final String invitedUid = mUser.getUid();
                            Log.d("Invited 1",invitedUid);
                            // TODO IF INVITE GROUP ID MATCHES USER INVITED GROUP ID OR DOES NOT EXIST
                            usersDatabase.child(invitedUid).child("groups").child("groupId").addListenerForSingleValueEvent(new ValueEventListener() {
                                @Override
                                public void onDataChange(DataSnapshot dataSnapshot) {
                                    if (dataSnapshot.getValue() != null) {
                                        String ds = dataSnapshot.getValue().toString();
                                        if (ds.equals(playerGroupId)) {
                                            // then it matches so we can move data
                                            Log.d("Invited 2", ds);
                                            Log.d("Invited 3", playerGroupId);
                                            moveFirebaseRecord(groupsDatabase.child(firebaseUser.getUid()).child("matches"),
                                                    usersDatabase.child(invitedUid).child("groups"));

                                            Toast.makeText(getApplicationContext(), "* found **" + " " + invitedUid, Toast.LENGTH_SHORT).show();

                                        } else {
                                            // it does not matchh so warn inviting user that thay are already involved in a match
                                            Log.e("EditPlayer", "Player already member of group");

                                            String pushKey = dataSnapshot.getKey();
                                            Log.d("push", pushKey);
                                            Toast.makeText(getApplicationContext(), "This player is already a member of a Squad, please update", Toast.LENG
                                            //usersDatabase.child(firebaseUser.getUid()).child("members").child(pushKey).child("additionalMatch").setValue(
```

Figure 19 – EditPlayersActivity.java - search by phone number to see if player is an existing user called from the writeNewUser method. Code modified from Ref – (Stack Overflow 2017)

The above method covers a lot of scenarios, and checks by phone number to see if the player added is a registered user on the app, it also checks to make sure that if the player is a registered user if they have been added to another users Squad. As mentioned earlier each user can be an administrator of one Squad and can be a member of only one other Squad. If the player is found to be a member of another users squad a message is displayed letting the user know. If the player is not a member of a Squad or is a member of the inviting users Squad already then the below method (moveFirebaseRecord()) is called.

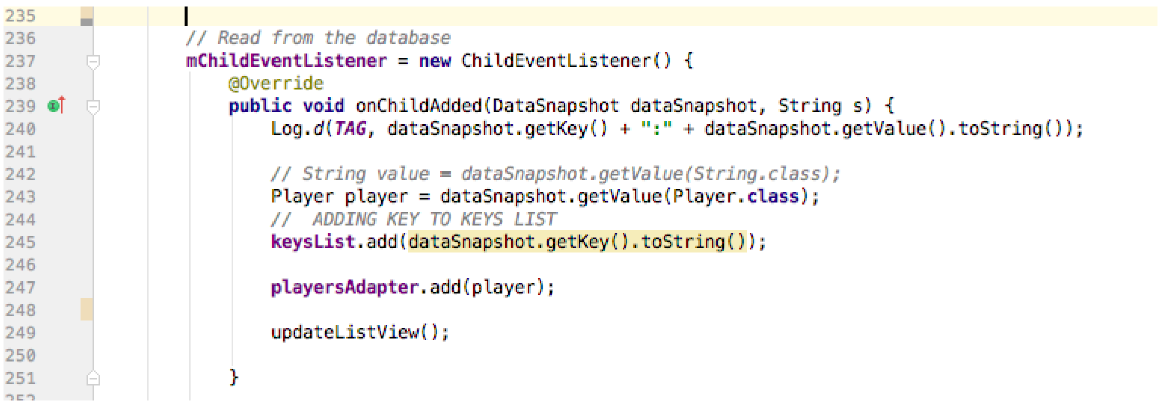


```java
public void moveFirebaseRecord(DatabaseReference fromPath, final DatabaseReference toPath) {

        Log.d("moveRec", String.valueOf(fromPath));
         Log.d("moveRec", String.valueOf(toPath));

    fromPath.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(final DataSnapshot dataSnapshot) {
            Log.d("moveRec", String.valueOf(dataSnapshot));


            toPath.setValue(dataSnapshot.getValue(), new DatabaseReference.CompletionListener() {

                @Override
                public void onComplete(DatabaseError databaseError, DatabaseReference databaseReference) {
                    Log.d("moveRec", String.valueOf(dataSnapshot));
                    Log.d("moveRec", String.valueOf(toPath));


                    if (databaseError != null) {
                        Toast.makeText(getApplicationContext(), "COPY FAILED", Toast.LENGTH_LONG).show();
                    } else {
                        Toast.makeText(getApplicationContext(), "COPY SUCCESS", Toast.LENGTH_LONG).show();

                    }
                }
            });
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            Toast.makeText(getApplicationContext(), "onCancelled- copy fail", Toast.LENGTH_LONG).show();

        }
    });

}
```

Figure 20 – EditPlayersActivity.java - If Player's phone number search is found in User node, this method copies match data to that User . Code modified from Ref – [GitHub Gist 2017)

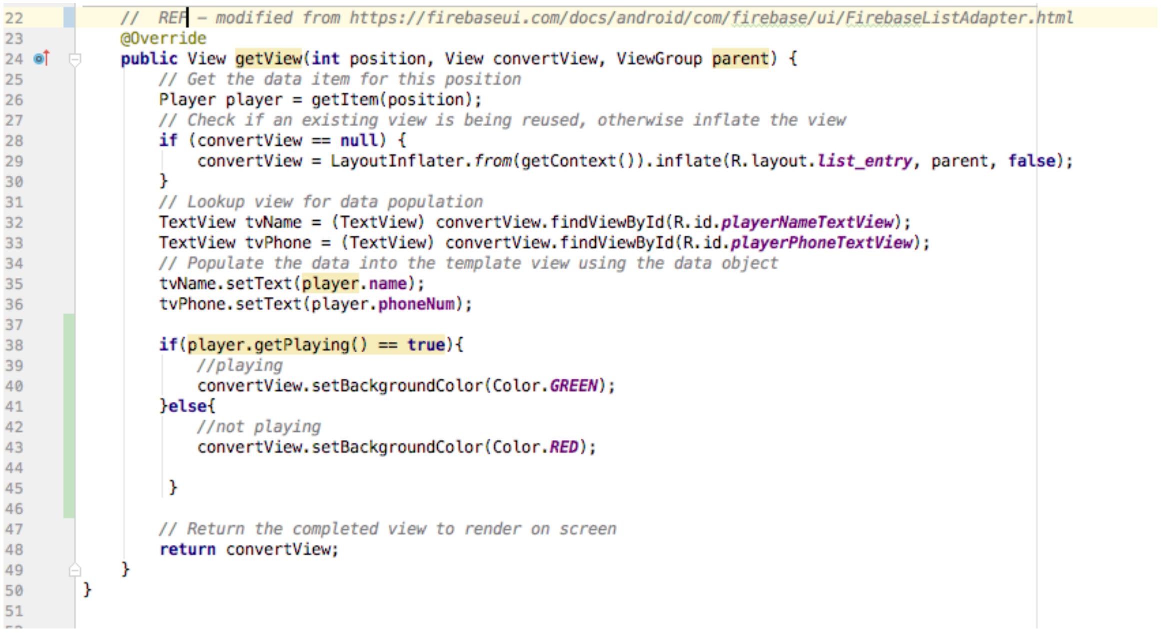
The above method is used to ensure that there is consistency between Players and Users. Keeping this data consistent was a priority as the data displayed on one users phone needed to be consistent with what was displaying on another user. Consolidating the players and the user’s nodes on the database proved to be a major challenge in keeping consistency and many checks such as the above had to be taken to ensure read and write consistency.

Once users have been added the recommended approach from the Firebase documentation is used to read the data and populate the list view.



```
        // Read from the database
        mChildEventListener = new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {
                Log.d(TAG, dataSnapshot.getKey() + ":" + dataSnapshot.getValue().toString());

                // String value = dataSnapshot.getValue(String.class);
                Player player = dataSnapshot.getValue(Player.class);
                //  ADDING KEY TO KEYS LIST
                keysList.add(dataSnapshot.getKey().toString());

                playersAdapter.add(player);

                updateListView();

            }
```

Figure 21 – EditPlayersActivity.java - onChildAdded listens on the appropriate node on the database for changes and populates the list view accordingly.

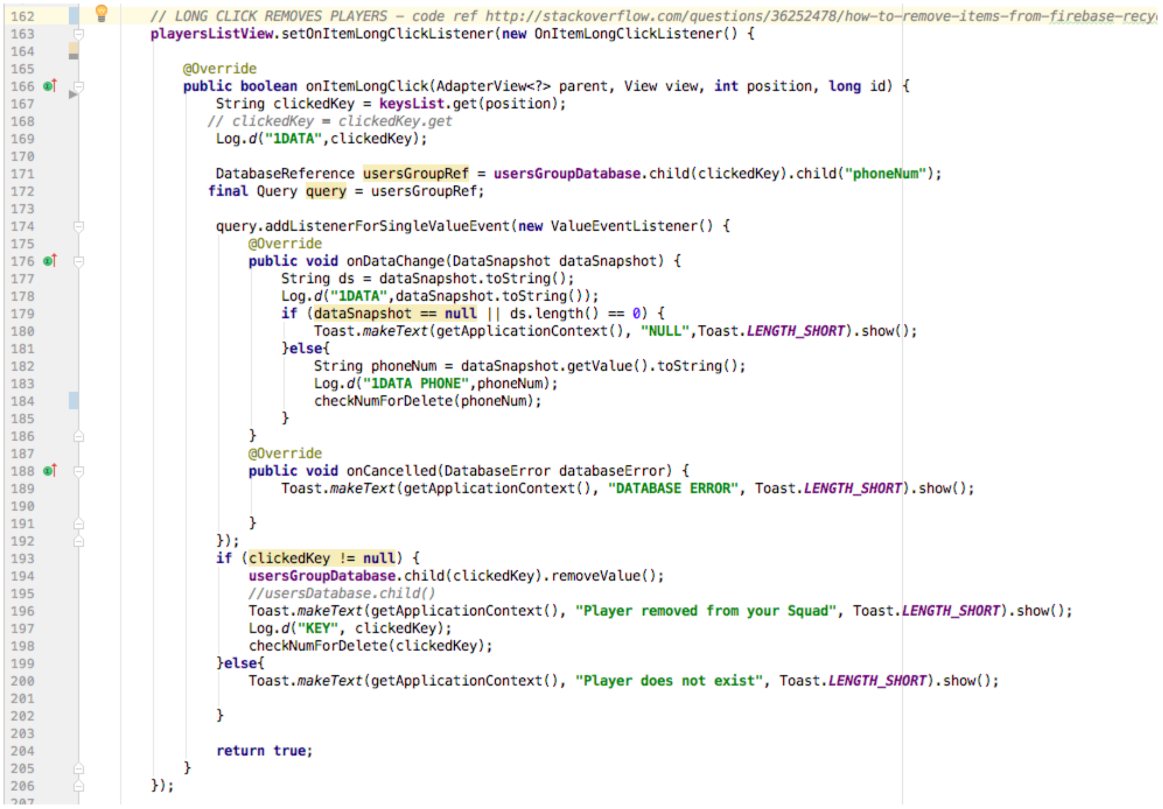


```
    //  REF - modified from https://firebaseui.com/docs/android/com/firebase/ui/FirebaseListAdapter.html
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // Get the data item for this position
        Player player = getItem(position);
        // Check if an existing view is being reused, otherwise inflate the view
        if (convertView == null) {
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.list_entry, parent, false);
        }
        // Lookup view for data population
        TextView tvName = (TextView) convertView.findViewById(R.id.playerNameTextView);
        TextView tvPhone = (TextView) convertView.findViewById(R.id.playerPhoneTextView);
        // Populate the data into the template view using the data object
        tvName.setText(player.name);
        tvPhone.setText(player.phoneNum);

        if(player.getPlaying() == true){
            //playing
            convertView.setBackgroundColor(Color.GREEN);
        }else{
            //not playing
            convertView.setBackgroundColor(Color.RED);

         }

        // Return the completed view to render on screen
        return convertView;
    }
}
```

Figure 22 - PlayerAdapter.java - modified from Code Ref (Firebase UI 2017)

The PlayerAdapter is a custom adapter that has been modified to show the text on the list item. It checks what the status of that player is at the time and marks it a different colour if the player has confirmed that they are playing. Squad uses an opt in strategy, when players are added to a group they are set as not confirmed on that group and thus show as red to the admin. Once they have confirmed they are playing the list will update to reflect this on the admins screen. This way the admin can keep a visual check on how many players have been invited and how many have confirmed their attendance. Players can change their status at any time through the app and the change will be reflected on the Admins screen.

With the above code users can add player's data consistently on all devices reading and writing from the database, below I will outline how deletions are handled. As with adding player data consistency is key with deletions and there are a number of checks that take place to ensure read and write consistency for all Squad users throughout.



```java
    // LONG CLICK REMOVES PLAYERS - code ref http://stackoverflow.com/questions/36252478/how-to-remove-items-from-firebase-recy
    playersListView.setOnItemLongClickListener(new OnItemLongClickListener() {

        @Override
        public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
            String clickedKey = keysList.get(position);
           // clickedKey = clickedKey.get
            Log.d("1DATA",clickedKey);

            DatabaseReference usersGroupRef = usersGroupDatabase.child(clickedKey).child("phoneNum");
           final Query query = usersGroupRef;

            query.addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    String ds = dataSnapshot.toString();
                    Log.d("1DATA",dataSnapshot.toString());
                    if (dataSnapshot == null || ds.length() == 0) {
                        Toast.makeText(getApplicationContext(), "NULL",Toast.LENGTH_SHORT).show();
                    }else{
                        String phoneNum = dataSnapshot.getValue().toString();
                        Log.d("1DATA PHONE",phoneNum);
                        checkNumForDelete(phoneNum);
                    }
                }
                @Override
                public void onCancelled(DatabaseError databaseError) {
                    Toast.makeText(getApplicationContext(), "DATABASE ERROR", Toast.LENGTH_SHORT).show();

                }
            });
            if (clickedKey != null) {
                usersGroupDatabase.child(clickedKey).removeValue();
                //usersDatabase.child()
                Toast.makeText(getApplicationContext(), "Player removed from your Squad", Toast.LENGTH_SHORT).show();
                Log.d("KEY", clickedKey);
                checkNumForDelete(clickedKey);
            }else{
                Toast.makeText(getApplicationContext(), "Player does not exist", Toast.LENGTH_SHORT).show();

            }

            return true;
        }
    });
```

Figure 23 - EditPlayersActivity - Long click listener allows data to be removed, modified from code ref – (Stack Overflow 2017)

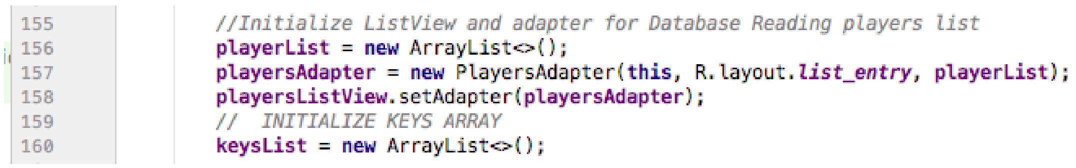

```
//Initialize ListView and adapter for Database Reading players list
playerList = new ArrayList<>();
playersAdapter = new PlayersAdapter(this, R.layout.list_entry, playerList);
playersListView.setAdapter(playersAdapter);
//  INITIALIZE KEYS ARRAY
keysList = new ArrayList<>();
```

Figure 24 - EditPlayersActivity - Adapter, ArrayList to store Player data in and ArrayList to store the key of the item clicked to facilitate deletion

The above code listens for a long click on the list view and using the keys ArrayList identifies the item that has been selected for deletion and this is passed through the event listener in to delete the data from the database. A check takes places searching by phone number to ensure that the match details data is asynchronously deleted from the deleted player's main screen so that it matches the data in the admins list view. The same method from figure 19 that searches for users via phone number has been adapted to facilitate this search.

2.4.10 Push Notifications

In order to take advantage of the push notifications when a user is added to a match I used Firebase Cloud Functions to listen to events on the Realtime database. I added this functionality using NodeJS and using NPM I installed the Firebase functions suite, tis allowed me to listen to events on the database. Once the trigger for a user being added, the cloud function fire a welcome email to that user. Notifications are also used in this same manner. When a new player is added to the database a push notification is sent to that user notifying them of the details of that match.

2.5 Testing

An integral part of developing any software application is the testing phase. It is a much vital stage that cannot be overlooked before releasing an application. Some of the main reasons and benefits for the testing phase are:

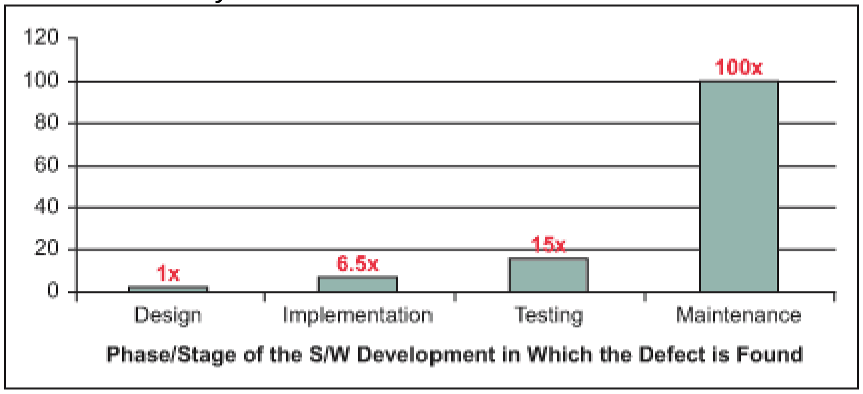
1. To evaluate that the system can do what it is supposed to
    a. It is very important that the system is evaluated to prove that it achieves the task that has been set out to achieve.
2. Find defects early
    a. It has been proven that the later an error is caught the higher the time and money cost to fix it.
    b. 

Figure 25 - Relative Costs to Fix Software Defects (Source: (Isixsigma.com, 2017)

l)

3. Ensure that users cannot harm system integrity
    a. Users should not be able to gain system admin access to the database or compromise the system in any way
4. Ensure that the system works for many users not just one
5. Find out how reliable the system is.
6. Ensure that the application provides a good user experience

2.5.1 Black Box Testing

Black Box Testing is a method of testing that takes a look at the functionality of the application from a user's perspective. Without analysing code, the tester is given a Scenario and follows the steps needed to achieve the goal in the scenario. The expected result is outlined and the actual result is taken note of and based on the outcome we can mark it as a pass or fail.

Black Box Test # 1	
Test Name:	Sign up new user with e-mail
Test Purpose:	To ensure that user can successfully sign up to the application with a normal e-mail address
Test System:	Android HTC One M8 Phone
Steps:	1. Start the application 2. User login screen should appear if no user logged in 3. Select sign in with e-mail 4. Enter e-mail address, name and password 5. Press enter
Expected Result:	User should have access to the application and be prompted with the first time user welcome message to enter in their phone number.
Actual Result:	As above - User successfully created and logged in.

Black Box Test # 2	
Test Name:	Sign up new user with Social media button for Google account
Test Purpose:	To ensure that a new user can successfully sign up for the application using their saved social media Google credentials
Test System:	Android HTC One M8 Phone
Steps:	Pre-requisite: user must have Google account and be signed into it on the device 1. Start the application 2. User login screen should appear if no user logged in 3. Select sign in with Google 4. Choose the account to associate with Squad

Expected Result:	User should be automatically signed in and brought to the main page where the user should have access to the application and be prompted with the first time user welcome message to enter in their phone number.
Actual Result:	As above - User successfully created and logged in

Black Box Test # 3	
Test Name:	Log in with e-mail
Test Purpose:	To ensure that a user who has signed up can use those credentials to sign in
Test System:	Android HTC One M8 Phone
Steps:	1. Start application 2. Select sign in with e-mail 3. Enter e-mail address 4. App should recognise that you are a registered user and prompt you to enter your password. 5. Enter your password and press enter
Expected Result:	User should have access to the application and be prompted with the first time user welcome message to enter in their phone number.
Actual Result:	As above - User successfully logged in.

Black Box Test # 4	
Test Name:	Login with social media Google account credentials
Test Purpose:	To ensure that a user who has registered via the Google sign up flow can use this account to sign in
Test System:	Android HTC One M8 Phone

Steps:	1. Start application 2. Select the Google sign in button 3. Choose the account to associate with Squad
Expected Result:	User should be automatically signed in and brought to the main page where the user should have access to the application and be prompted with the first time user welcome message to enter in their phone number.
Actual Result:	As above - User successfully logged in.

Black Box Test # 5	
Test Name:	New user login enter phone prompt
Test Purpose:	To ensure that a newly registered user with no phone details entered is prompted to enter their phone details on login, and to ensure that this user will not be asked for these details again.
Test System:	Android HTC One M8 Phone
Steps:	1. Log in as a newly created user with no phone number record saved to their account. 2. User should be prompted with message to enter their phone number details and save. 3. Enter a phone number and save
Expected Result:	The prompt message should disappear and a Toast message should appear for 5 seconds letting the user know their details have been saved. The users phone record should be saved against their profile.
Actual Result:	As above

Black Box Test # 6

Test Name:	Create a new match
Test Purpose:	To ensure that a user can create a new match and that this user alone is the administrator of this match.
Test System:	Android HTC One M8 Phone
Steps:	1. Login 2. Select ‘My Match Details’ 3. Input selected details 4. Save Details 5. Log out and log back in, return to the match screen to ensure details have been saved 6. Check database that the match details have been saved correctly to this users node and they are admin
Expected Result:	Input details should be saved, Toast message should briefly pop up letting the user know that the save was successful
Actual Result:	As above, match saved

Black Box Test # 7	
Test Name:	Add Player(s) to match
Test Purpose:	To ensure that a user can add players from their contacts to the group for their match.
Test System:	Android HTC One M8 Phone
Steps:	1. Navigate to Edit Players section (ensure that a match has been created by this user) 2. Press the Plus button with no details entered 3. Application should tell user to enter a valid contact 4. Select the magnifying glass icon, this should bring up the user’s keyboard 5. Type the first letter of a name, this should bring up the user’s contacts list 6. Select a user with no phone number saved

	7. Application should tell user to choose a valid contact with phone number 8. Select a user with a valid phone number and name 9. Selected contact details should appear in text view below 10. Use the plus button to add this user to the list
Expected Result:	As well as above points, user should be shown a Toast message confirming the player was added. The player should be added to the List View with a White background to indicate they have not confirmed they will be playing yet. Validate the counter is working correctly.
Actual Result:	As above all validated

Black Box Test # 8	
Test Name:	Remove player(s) from match
Test Purpose:	To ensure that a user can remove players at any time from their players list.
Test System:	Android HTC One M8 Phone
Steps:	1. Navigate to Edit Players screen 2. Use long click to remove player from the list pf players
Expected Result:	Player should be removed from the list and in the list in the database. A Toast message should appear to the user confirming the deletion has been successful. The player count should decrement by one.
Actual Result:	As above all validated

Black Box Test # 9	
Test Name:	Check invitation has synched to added players account

Test Purpose:	To ensure that when user A adds user B to their match list that the information is synched and an invitation appears on User B’s screen.
Test System:	Android HTC One M8 Phone
Steps:	1. Login as User A 2. Note match time and day 3. Navigate to Edit players 4. Add User B to your Group 5. Login as User B
Expected Result:	Once logged in as User B there should be an invitation from User A containing the match details(who is the invite from, day, time) and a switch to confirm if User B is playing.
Actual Result:	As above, all validated

Black Box Test # 10	
Test Name:	Invitation positive response check
Test Purpose:	Following from Test #9 , to ensure that when User B updates their response on the app the data synchs correctly to User A’s players list.
Test System:	Android HTC One M8 Phone
Steps:	1. Login as User B 2. As User A has added User B to their list, there should be an invite on the homepage. Update the status of this and press update to a positive response. 3. Sign out as User B 4. Log in as User A 5. Go to Edit Players 6. Observe the players list

Expected Result:	User B on players list should now be showing as green on User A’s list
Actual Result:	As above, validated

Black Box Test # 11	
Test Name:	Invitation negative response check
Test Purpose:	Following from Test #10 , to ensure that when User B updates their response on the app the data synchs correctly to User A’s players list.
Test System:	Android HTC One M8 Phone
Steps:	1. Login as User B 2. As User A has added User B to their list, there should be an invite on the homepage. Update the status of this and press update to a negative response. 3. Sign out as User B 4. Log in as User A 5. Go to Edit Players Observe the players list
Expected Result:	User B on players list should now be showing as white on User A’s list
Actual Result:	As above, validated

Black Box Test # 12	
Test Name:	Sign in forgot password flow
Test Purpose:	To ensure that if a user forgets their password they can get a reset email sent to their email address
Test System:	Android HTC One M8 Phone

Steps:	1. Start the application 2. Enter email for registered user 3. Select forgot password 4. Follow flow and send e-mail to reset password 5. Check email and verify
Expected Result:	User should receive an email sent to the registered email address that allows them to reset and choose a new password once they click the link in the e-mail
Actual Result:	As above, verified.

Black Box Test # 13	
Test Name:	Check sign out
Test Purpose:	To ensure that users once signed in can sign out of the app
Test System:	Android HTC One M8 Phone
Steps:	1. Login 2. Sign out on each page option is available 3. Observe results
Expected Result:	User should be able to sign out with no system issues.
Actual Result:	As above, verified

Black Box Test # 14	
Test Name:	Update match details data synch
Test Purpose:	To ensure that when User A updates the details of the match (time, day etc.), these details are synched to User B (assuming User B has been added to User A's list of players)
Test System:	Android HTC One M8 Phone

Steps:	1. Log in as User A 2. Ensure User B is on the players list 3. Navigate to Match Details 4. Update time and day, press save 5. Log in as User B 6. Observe details on invite details
Expected Result:	Details on User B's invite should be concurrent with details from User A's match details
Actual Result:	As above, verified

Black Box Test # 15	
Test Name:	Update players list data synch
Test Purpose:	To ensure that when there is a deletion of player made to User A's player list that the data synchs to User B(player being removed)
Test System:	Android HTC One M8 Phone
Steps:	1. Login as User A 2. Navigate to player list and delete User B 3. Log out 4. Log in as User B 5. Observe main screen
Expected Result:	Invite with User A's match details should no longer appear on User B's home screen as User B has been removed from User A's list
Actual Result:	As Above, verified

2.5.2 User Acceptance Testing

To provide User Acceptance Testing I approached a group of people who regular play five a side and approached them about using the application to arrange their next match. They were all Android users and agreed to take part, I provided them with a build version of the APK and they installed the app on their phone. Their next game was in 2 days' time. One of the users was assigned as the admin of the group and he went about arranging the match through the application. I monitored the UAT process and was available for any issues, I also used Firebase Analysis services to monitor any crashes that may have occurred throughout. I asked if the users could come back to me after their match and report their findings. Based on their reports User acceptance was derived using the below question sheer with the following criteria:

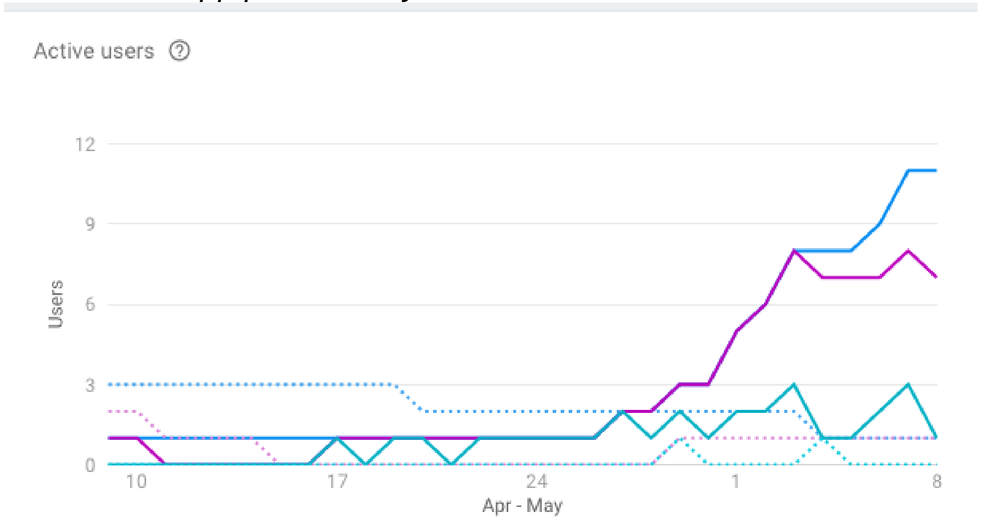
REF - https://www.tutorialspoint.com/software_testing_dictionary/use_acceptance_testing.htm

- Functional Correctness and Completeness
  - *Does every action behave in an expected manner?*
  - Users reported that all actions performed in an expected manner with no major faults to report. One user reported accepting an invite and this not updating correctly on the admins player list but was correct after next login. As this seems to have been a one off no action to be taken but I will be keeping an eye out for any further instances.

- Data Integrity
  - *Is data stored correctly?*
  - Other than the above temporary instance all data for the match was saved correctly. Users data synched correctly with no other issues reported.

- Usability
  - *Was the app easy to use and provide a pleasant user experience?*
  - All users reported that the app provided an intuitive experience with little to no guidance required as to how to use the app. The flow between screens worked correctly at all times and the menu systems provided clear and concise instructions where needed. Where there were no instructions users intuitively knew from using other technologies the expected behaviours required.

- Performance
  - *How did the app perform? System crashes? Timeouts?*



  -

Figure 26 - User graph from Squad Firebase analytics during UAT

  - Users reported that the application performed well and they were successfully able to manage people dropping out from the game and invite replacements to play. All data synched correctly and users reported that the application saved a lot of time wasting with messages etc. There were 2 issues with the application crashing when users log out immediately after logging in. This appears to have been to do with a null pointer exception issue with one of the Toast Messages which has now been handled. User reported that app performed correctly once they logged back in. All users reported that the user interface provided a clear and concise user experience

- Timeliness
  - *Was every action executed in a timely fashion?*
  - Users reported that all actions executed with no slow response times, and that every action responded in under 1 second. Only issue being the above mentioned crash which has now been fixed.

In summary the test group were able to successfully manage and arrange a Five a side football match using the application with little to no issues springing up in this early build. The general consensus from all users was that this was a useful app and they would be interested in using it going forward to arrange their Five a side matches. Points raised by the users were that even all of the test group were Android users there could be a scenario where a user could be an iPhone user and would not be able to user the app. This scenario will be handled in future releases.

2.6 Graphical User Interface (GUI)

2.6.1 Register

When the user starts the application it will automatically log the user in if they have logged in previously. Otherwise it will redirect the user to the register page where they are prompted to register a new account with the application. If the user does not have an account, they will not be able to access the application. The user will be presented a form with user details such as 'user name', 'e-mail' and 'password' to fill in. Once all fields have been completed the user will be allowed to press the register button to register a new account. There is an option for existing users to sign in to their account and also two buttons to register via social media (Twitter and Facebook). Figure 1 represents the Register page.

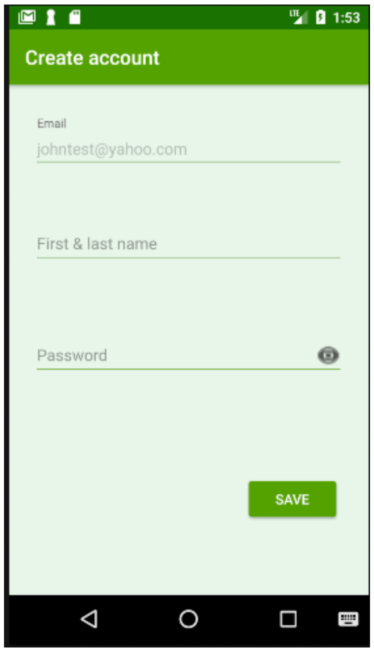


Figure 27 - Register details page

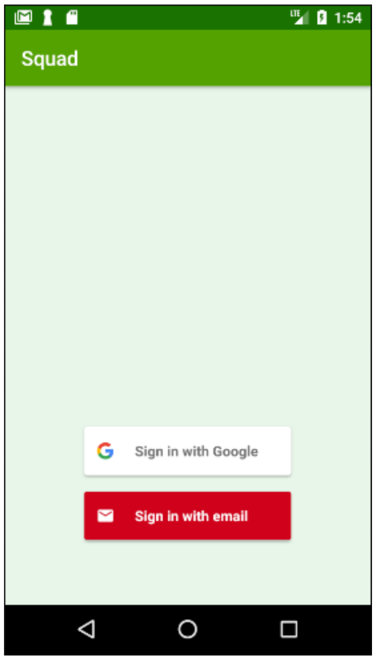
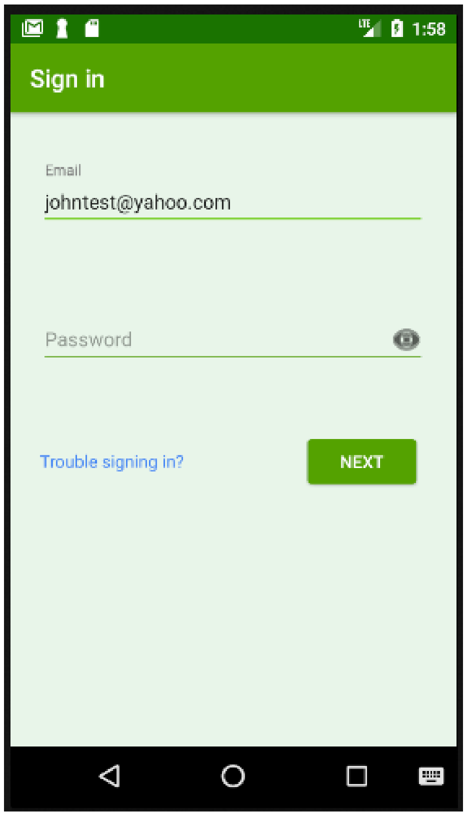


Figure 28 - Register splash page

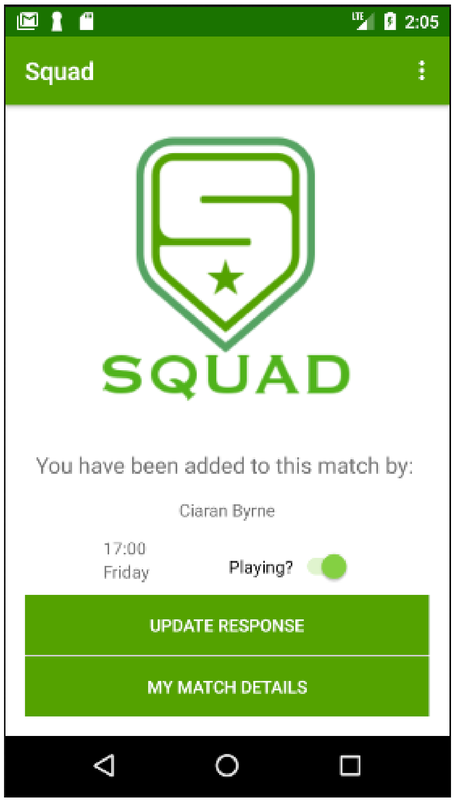
2.6.2 Login



The user will have the option to log in once the application has started. The menu asks for email and password. Figure 28 represents this screen, there is also an option to login with Social media the Google account button allowing an automatic sign in flow for registered Gmail users as seen in Figure 27

Figure 29 – email sign in page

2.6.3 Main Screen



You have been added to this match by:

Ciaran Byrne

17:00
Friday

Playing?

Figure 30 - Match invite on welcome screen

The main menu will welcome the user with their registered user name and will display to the user match invite confirm fragment if they are added to one. The display will show details of the next upcoming match with time, date etc shown. Any user invited to play a match will be prompted to respond to the invite as shown in Figure 30. The user will have the option to edit matches from the button below which will bring them to the 'My Match Details" screen.

2.6.4 Match Management

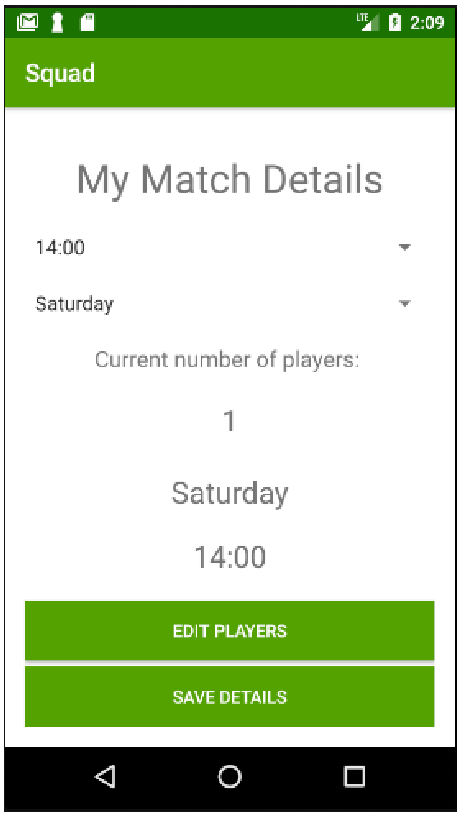


Figure 31 - Match details

If the user selects the edit match button the system will bring them to the Match Management menu screen. The user can edit details of an existing match by using the menu and can also create a new match where they will input the same details. If the user wishes to add participants, they choose the edit players button which will bring them to a new screen. The current player count is also represented on this screen

Match Management is represented in Figure 30

2.6.5 Edit Players

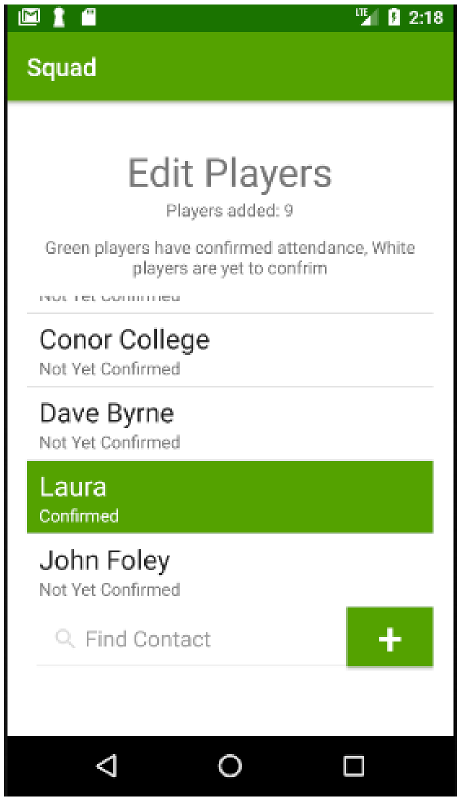


Figure 32 – Edit Players

The edit players screen is presented to the user when the edit participants button is pressed. This screen allows the user to edit existing participants or add new participants. The application validates the information before submitting the changes to the database. This screen also acts as a visual aid to see which players have confirmed their attendance. In this case we can see that 9 players are invited but only one has confirmed their attendance.

Edit players is represented in Figure 31

2.7 Evaluation

Below I will outline the steps take to provide a heuristic evaluation of the entire application. This evaluation is needed to ensure that there are not any oversights in the systems interface that may lead to some confusion for the user. This is an important step to undertake before going Live as when you are in the development process and become so involved with developing and designing an application something that may seem obvious to the developer may not be obvious to the end user. I used the same group of users from the UAT test group to provide feedback on the heuristics of Squad.

Using the Heuristic guidelines, it is necessary to ensure that your application passes on the following criteria.

- Visibility of system status
    - *The user should at all times know where they are in the system and the interface should reflect this.*
    - The GUI for Squad clearly labels the name of the page that the user is on and none of the users at any time were confused as where they were or where to go in the application.

- Match between system and real world
    - *System should use real world language the user will understand, no jargon*
    - There is not a lot of reading to be done with the application and any areas with text is clear and concise with users reporting this.

- Consistency and standards
    - *There should be a consistency with situations and languages and interface to avoid confusion*
    - Users reported that the GUI was very consistent through out

- Error prevention
    - *Errors should be prevented where possible, and handled with an error message when not. Users should not experience a crash with no explanation.*
    - As mentioned in the UAT testing there was one once off crash but all other errors observed are either avoided or handled with a message to the user.

- Recognition rather than recall

    - *All options, actions and objects should be visible to the users. The users should not have to remember where a button is hidden to access it for example.*
    - This has been covered in Squad with all menu items very clearly marked and their use clear to the user

- Aesthetic and minimalist design
    - *Extraneous information is a distraction and a slow down*
    - Squad's minimalist design and lack of distraction allow for a smooth user experience. The user is provided with just the information they need for the app to work for them and any first time users have shown that the intention for all menu items on the app is obvious without the need for labels.

3 Conclusions

This report outlines the process and technologies used to develop a fully operational application for user and group management of sporting events with friends – Squad. I have shown how the application was developed, the technologies used and practices put in place to ensure its success. I found this to be a very challenging endeavour and I learned a lot about Software development from start to finish and what it takes to build a fully functioning application. I learned many new technologies along the way and I have pushed the envelope in terms of implementing the latest technology by using NoSQL databases in the form of the Firebase system and the latest Android techniques.

There were many hurdles along the way, I got stuck very early on in the development process and could not progress with the application for a number of weeks as I needed to achieve this one area of the app in order for the rest of it to function. This set me back a number of weeks but it was a major milestone when I overcame it and it gave me the confidence to push on and complete the application.

Learning how to develop complex Android systems with equally complex databases that needed to be synched for a variety of scenarios in order to maintain consistency in the app posed a major challenge and pushed my abilities right to the end. I feel I have learned a lot of good programming skills from developing this app and while using Android / Java to develop the app was a major challenge as I had little to not experience with developing in Android I am happy I did now as it is more rewarding than if I chose a simpler route to take.

4 Further development

Future development for the application would be to build a supporting web application for non-Android users and eventually develop a version for iPhone users so as to have full market coverage.

An in built booking system could be integrated into the app to provide the facility to book pitches at the sport complex.

I plan to continue development of the application with the view to meeting these goals and also to approach some gyms and sport complexes about rolling the application out to their users.

5 Appendix

5.1 Project Proposal

5.1.1 Objectives

The main objective for this project are to provide an easy to use mobile and web application to end users that eases the organization of sports events amongst friends.

The application will be a mobile first application and will aim to solve a common problem when trying to organise team based sporting events and will be mainly focused in the domain of Five a side football as a starting point. The reason for this being that five a side footballs huge popularity it makes for a logical starting point for the application's user base.

Users will sign up to the application in order to make the organization of matches easier from both the side of the people playing and the people who run the pitches that are used.

Owners/ managers of gyms and football clubs offering the facilities will sign up to the application and will publish their timetables for the pitches through an easy to use interface. This booking system will update dynamically once teams ("Squads") have booked an available time slot, stopping anyone else from double booking.

The booking interface will be accessible to the group's captain once they have also signed up, joined Squads and nominated a captain. The captain will be the main point of contact with the pitch owners for organising times, payment etc. and is responsible for their group.

Another main feature is player management within squads, once you have a squad of your main 10 – 14 regular players you can nominate additional substitutes. If a player from within your main playing base cannot make it to the match on a particular day they can "Sub" themselves out. The application will then send a push notification to the substitutes and they one of them can nominate themselves to play if they are available.

Although initially focused on five a side football the application can be used to organise any number of events e.g. Tag rugby, Badminton etc. Basically any sporting events between friends where you need to meet up and play but sometimes there

The application will provide a secure login and sign up, utilizing sign in features through social media and interaction with social media e.g. Tweet about this game etc.

Push notifications and consistent but dynamic information will be a priority and it must display consistent information on all user's devices.

5.1.2 Background

The idea for this application came to me when I noticed a common problem when trying to organise five a side football matches with friends. Every day thousands of people across the country have pitches booked to play on a regular or once off basis. They have their players ready and looking forward to a game with friends but at the last minute somebody drops out because they have to work late or some other reason. This can lead to a logistical problem for the person who is organising the game, people are depending on him/ her to ensure the numbers are there for the weekly match. Emails and texts are sent out to try and find a replacement but the they are busy with their own life and don't have the time to find a new player.

This problem is nobody's fault but can cause a lot hassle if you are the one organising the match. A simple fact is that life events get in the way and people will drop out on a very regular basis.

Squad aims to provide a solution to this problem, making it easier to find replacements when the inevitable happens and taking some of the micromanagement out of the equation. The application will notify users and add replacements onto the roster once they have nominated themselves in.

Giving users the ability to book pitches and timeslots without calling the pitch owners or face to face interaction with them is another selling point of the application. The easy to use booking system allows for this feature and makes life easier for both the players and the manager of the pitches.

Other features will allow mini leagues or cups to be organized between other Squads.

With the number of people playing Five a side football and other sports amongst friends on a casual basis there is great opportunity for an organization application such as Squad to become very popular. There are some applications that offer some of the features but none offer the whole package. Some attempts to solve the problem of people dropping out are the ‘LateCallUp’ Twitter page.

This does not offer the ease of use for players and gyms/ sports complexes with all the features that Squad offers.

The plan for this application on roll out would be to approach local sports complexes and encourage them and their users to adopt Squad as a system for booking their pitches. This would be attractive to them for the ease of use and it would set them apart from the others in the eyes of the player.

5.1.3 Technical Approach

The project will be built in an iterative and incremental development approach along the lines of the Agile framework. The reason for this being that when designing and developing this project I will be learning many new technologies and concepts and it will be difficult to plan in advance for many scenarios and using this model will allow for some flexibility. Developing in this iterative manner will allow me to add and test new parts of the system bit by bit. This will mean that any problems that may be found in development can be solved early without too much of a detrimental effect on the system build as a whole.

There is much to learn on this project so the bulk of my research has been on the technologies I plan to use but I feel I am up to the task and have a good grounding in the technologies needed. I have started the process of learning how to build React Native applications through online courses and researching the React documentation.

Requirements capture will involve speaking to potential end users and asking them their opinions on what are must haves and other desirable features for the application. In addition to this measurable goals will involve a brainstorming session and continually asking why until the applications purpose is discovered. Using this methodology, I will discover these functional requirements and through the building of the application in an iterative manner I will optimize the non-functional requirements such as reliability, speed and ease of use.

Use case diagrams will be made for the major use cases as we as a UML diagrams/ Entity Relationship diagrams in the planning stage to help visualize and understand what is required from each of the distinct parts of the system and how they relate to each other functionally.

Once the planning stage is completed development will begin based on the iterative and incremental model mentioned above.

5.1.4 Technical Details

Implementation language and principal libraries

The application will be built a combination of technologies mainly comprising of a JavaScript Framework- React Native for the View portion of the application, which allows for UI rich native applications. Using React, JavaScript, HTML5, CSS3 among others you can develop an application that runs as well as the language it runs on i.e. Java on Android but is more flexible. Applications made in React run much better than an application built using a Cordova system which makes calls to APIs, React Native provides an app that runs as well as any natively built app.

React is heavily supported by the likes of Facebook and Instagram and is a proven technology in mobile applications and web applications. The modular nature of React Native will also fit well with the iterative approach I am taking.

Other technologies used in conjunction with React Native are; Mongo DB and Node to handle the server and database side of the application. All are technologies that are very friendly from the development point of view with JavaScript/ JSON.

Nightwatch JS will be used for automated systems testing.

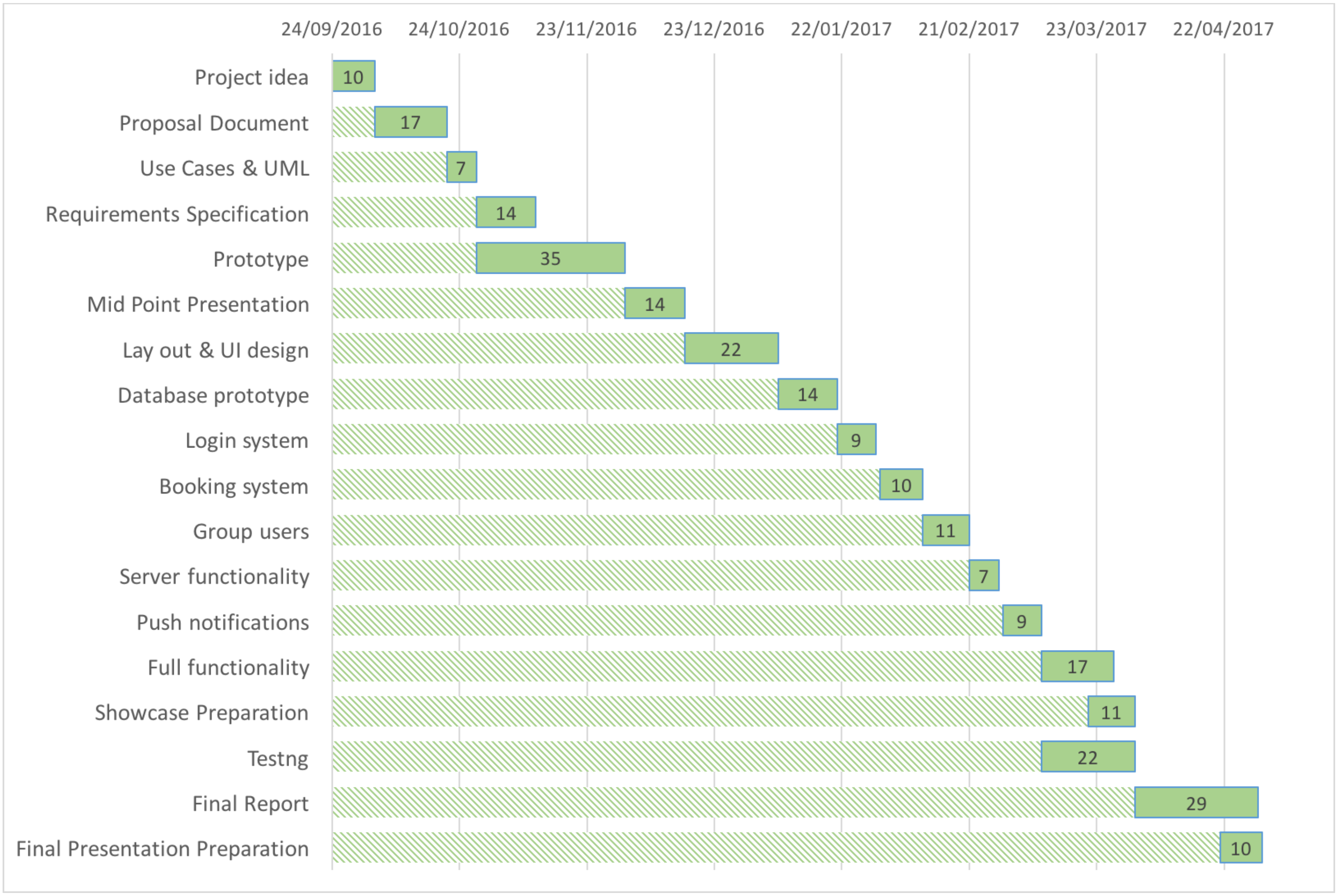
5.1.5 Evaluation

Describe how you will evaluate the system with real technical data using system tests, integration tests etc. In addition, where possible describe how you will evaluate the system with an end user.

To evaluate the project I will use isolated systems and unit testing as well as real world testing. I will get some volunteers to download and use the application and form a Squad, with me acting as an administrator of a pitch. I will provide timetable information and allow user to book via the app, Members of the squad will add themselves in and nominate players as subs, then swap in and out. Users will be allowed to perform general testing on the application at various stages.

Testing will also be performed based on the Use cases from start to finish and some automated systems testing will be performed using Nightwatch.js testing framework.

5.2 Project Plan



Task	**Start Date**	**Duration**	**End Date**
Project idea	24/09/2016	10	04/10/2016
Proposal Document	04/10/2016	17	21/10/2016
Use Cases & UML	21/10/2016	7	28/10/2016
Requirements Specification	28/10/2016	14	11/11/2016
Prototype	28/10/2016	35	02/12/2016
Mid Point Presentation	02/12/2016	14	16/12/2016
Lay out & UI design	16/12/2016	22	07/01/2017

Database prototype	07/01/2017	14	21/01/2017
Login system	21/01/2017	9	30/01/2017
Booking system	31/01/2017	10	10/02/2017
Group users	10/02/2017	11	21/02/2017
Server functionality	21/02/2017	7	28/02/2017
Push notifications	01/03/2017	9	10/03/2017
Full functionality	10/03/2017	17	27/03/2017
Showcase Preparation	21/03/2017	11	01/04/2017
Testing	10/03/2017	22	01/04/2017
Final Report	01/04/2017	29	30/04/2017
Final Presentation Preparation	21/04/2017	10	01/05/2017

5.3 Monthly Journals

Month: September '16

21/09/2016
We had our first Software project class with Eamon last night and he outlined what is expected of us for the project, looks like we have as tight a schedule as I thought (maybe tighter) as we need to have our proposal done by 7th October. I haven't got a definite idea for the project yet but have a few ideas that I need to research more and flesh out in general to see if they are big enough in scope and that I am capable of under taking them.

22/09/2016
After work I spent the evening brainstorming ideas for the project, I have a fair few ideas a lot are not very original or unique. I have narrowed down on a few and need to expand on these and see which are the best potential candidates.

27/09/2016
Following much brainstorming I have come up with an idea that i feel will be a good fit for my Final year project. As the lecturers have suggested I have kept it relatively simple and attempted to play to my strengths while doing something that I feel will be interesting to me and worthwhile.

Squad will be an application primarily to manage 5 & 7 a side football matches, basically to make it easier when the scenario arises that someone drops out of a match. The application will send a notification to the nominated back up players and they will sub themselves in. It will also handle time tables of available pitches and allow the user to create mini leagues & cups etc.
Now I just need to prepare it for the presentation to the judging panel to get approval.

30/09/2016
After work today I spent an hour sitting down and trying to write down the workflows involved for my application and just what would be involved. Looks like it could become a bit of a beast which is making me second guess my choice! With the level of the workload we already have I am not sure if I am to completing this project to a standard. I would be happy with.
Especially if I take into account that the technologies I have been looking at (React Native / MongoDB/ Node). I have done a little bit with React but nothing on the scale of what I am planning with this application. It would require a lot of hard work to learn a new technology along the way but on the other hand it may be the best way to learn new languages, to try and build something with it.

Achievements
This month, I was able to come up with a general plan on the project, including what the project will be, what it will be built with. I got it through the approval process and have fleshed out some initial ideas and have an idea of what is involved in building it. I just hope it is not too much with our considerable workload from other subjects.

Intended Changes

Next month, I will try to get the project proposal completed and further plan and manage the paperwork for the project. Further research in to React Native and Node is needed to see if this is the best set of technologies to use to complete the project for me, I am unsure on it as I would be learning a lot of it as I go along which may be time consuming. I need to also come up with a time line to give me an idea of when I should be hitting certain targets.

Month: October

04/10/2016

Pitch project day, I spent a couple hours preparing my slides and my pitch for the judging panel. I am up last which gives me some extra time…also means I will miss the beginning of our AI class but no big deal as long as I get accepted. Although Eamon tells us that the lectures projects are of quite a high standard so it may not be too bad either way. I am going in with a pretty relaxed attitude anyway. Not sure if that will work for me or against me!

07/10/2016
My project idea has been accepted so happy days on that front. I have been very busy the last few days, but my project has been accepted, all I have to do now is build it. I was researching and planning more on my final year project and there is a lot of work to be done. I still have not settled on what technologies I will use but I am still leaning towards the MERN stack (Mongo, Ember, React, Node) as I am comfortable with JavaScript and have used a little bit of React in the past, albeit not React Native which is the direction I would go for the app. But judging from my research it seems that they are very similar, and if you know one you know (or can learn very quickly) the other.

14/10/2016
I've begun work on my project proposal document, it’s actually a quite useful process that makes me think about the various tasks that lie ahead of as well as helping in the planning stage. We are extremely busy in college the last week and next, next week we have 2 deliverables, and 2 CAs which doesn't leave a lot of time for anything else. I am actually going to have to give classes a miss this week in order to get everything done.

21/10/2016
This week I finished the project proposal document and submitted it, I found the hardest part of this document was putting together a Gantt chart and planning the deliverable dates. Although it was a bit of a pain between this and putting together the technical approach it helped me cement in my mind more how the project is going to be built. The workload is making me feel a bit anxious especially as I have mentioned that I have to learn a new framework while building it. I emailed my project supervisor Padraig De Burca today. I know he is very good with the diagrams and planning stage of software engineering so I am hoping to get some useful tips for the next document the Requirements Specification.

28/10/2016
I have been mostly working on the requirements document, I have not got really stuck into the meat of it just yet but the template alone is 15 pages nearly. I have not heard

back from Padraig about meeting up but he would be useful on this. Reading week is coming up this week so I am planning to get really into it then. We have a number of other assignment and projects that I need to prepare for so it is going to be a busy time over reading week.

My Achievements

This month, I was able to finish my project proposal and get a start on the project requirements document.

My Reflection

I felt, it worked well to get the project proposal and plan done.

However, I was not successful in getting a meeting with my supervisor and getting started on the prototype.

Intended Changes

Next month, I will try to get a meeting with my supervisor

I realized that I need to get a start on my prototype and start learning the technologies I will be using to build the application.

Month: November

03/11/2016
I have begun research and started planning out y requirements spec document this week, looks like it will be a huge undertaking to get this finished to a high standard but there are a lot of marks going for it so it is worth it i think. I am going to try and get most of it done during my lunch breaks in work so it will hopefully minimize impact on other modules that I need to work on.

11/11/2016
I had my first meeting with my project supervisor Padraig De Burca last night, it was definitely productive and he gave some good insights in aspects of the project I may have overlooked. He has given me one problem though, he brought up the fact that what the app achieves could be achieved with a WhatsApp group. This is a bit of a spanner in the works for my idea in terms of originality and I fear it may go against me. I was determined to keep the application as simple as possible but I may have to expand upon it and add a booking system aspect so it stands out more and has more

functionality. I will have to research this more and I am just hoping it just means the scope of the project doesn't get too big for me to handle. It will also mean I will have to revise my Project Requirements document again which is a real pain, I thought I was done with it. It was a real slog to get through it all as well.
Other than all this just the usual pressure from modules, it seems sometimes with the amount of time they suggest spending on their subject that they think we are studying them in isolation.
I would definitely question the College's decision to change the idea that fourth year for part time students should be completed over 2 semesters and not 3 like previous years.

18/11/2016
The requirements spec document was due today, after much work and 5000 words it is finally done. I am a little bit fed up of all the paper work involved now and just want to get started in the coding. I added new functionality in the spec document as per Padraig and hopefully this will make it a more viable project now. I sent it onto him to get his opinion and to set up another meeting but no reply as of yet.

25/11/2016
So this was an interesting week to say the least outside of college, unfortunately I got knocked off of my bicycle by a car and have ended up with a broken wrist and sprained shoulder. I have to make some hospital visits and physio visits and can only type with one hand for the for seeable future which is going to make the assignments and coding quite a bit slower. On the plus side i have gotten a bit of time off work s it will allow me more time to slowly chip away at some of the assignments and project work.
We have our second CA in Android development tomorrow and i have been doing quite a bit of work with it, I've taken to it quite well so I am considering doing my prototype and main project using Android. It will mean I will have to update some of the documentation but this is a small price to pay, it makes more sense to use Android now that I am familiar with it than learn a brand new programming language. I am going to run this by my project supervisor and see what he thinks once we arrange a time to meet.

30/11/2016
I am falling behind on work as I have been instructed by the doctor to not use my left hand for typing, starting to get a bit stressed out. I don't think this semester is going to end well grade wise unfortunately, struggling to keep my head above the water.
i have managed to get started on the project prototype and have been using a 'Material Design' menu system on Android studio. We have not covered anything like this in class and i thought it was going well but have now hit a bit of a road block with it. Using intents seems to behave differently from what we have been taught so far. I am going to ask for help from our Android lecturer the next class we are in and hopefully he can point me in the right direction.
I have just emailed Padraig again about another meeting and will hopefully get time with him so i can get a better idea of what is expected in the prototype.

My Achievements

This month, I was able to get my prototype started.

My contributions to the projects included getting started on the prototype and more paperwork and reports.

My Reflection

I felt, it worked well to change over to Android to program the project, and I am happy to have made a start

Month: December 2016

My Achievements

This month, I was able to complete my prototype and make strides to developing the beginnings of the functionality of my project.

My contributions to the projects included the prototype and interface

07/12/2016

I have been working on my projects this week but finding it quite hard with just the one hand to type with, at last I got some time off work otherwise I don't think I would be able to keep my head above water at all. I'm already falling behind in a couple of modules like AI and Data Application Design. I have managed to keep up some work with my main project but I have neglected it recently in order to study and meet deadlines for other subjects. I am aiming to get material design working for my main prototype and will continue to work towards this. I have a base menu system done that I followed from a tutorial, it took a lot of work but it looks really professional and well done.

14/12/2016

This week I have been working on getting the prototype ready for the demonstration on the 19th. It's coming together okay but I am struggling with some parts like getting the screens interacting consistently. It's worth a bit to get this right and feeling the pressure with everything else due (as usual)

I met with my project mentor this week and he guided me towards what major points I should be hitting in the presentation and prototype. He actually said to not focus so much on the prototype as it is only worth 20% of the presentation. I am glad to have spoken to him about this as I may have been putting too much emphasis on it.

21/12/2016

Prototype presentation went very well and even though I got a bit nervous and skipped over some of the stuff I wanted to speak about but I think I gave a good account of the project and it's potential. The lecturers both seemed happy with the progress and direction I was taking and seemed to both really like the project idea. I got some feedback regarding it being an Android only project at the moment but I didn't really consider this to be anything beyond a college project to show my competencies as a coder as opposed to the potential to sell it on. Which I still think it has despite not having an IOS version. We are on Christmas break as of 23rd and I will submitting the last of my projects of the semester on that day and not looking at anything until the new year when I have to study for the exams

Supervisor Meetings

Date of Meeting: 15/12/16

Items discussed: Prototype and presentation

Action Items :Focus more on the presentation and not so much on the prototype, outlined items to hit in presentation.

Month: January

My Achievements

This month, was less productive than others, between studying for the exams and needing a bit of a break after an extremely hectic first semester. I did a couple of tutorials on some concepts I need to learn to complete my project; login authentication, Firebase database. I got though most of these tutorials on Udacity and felt I learned a lot from them but need to go back and learn more. I am a little worried at the level of work that needs to be done, and more so the amount I need to learn in order to finish the project. I have been looking at revising my project plan and will need to complete tasks as they come on a week by week basis.

My Reflection

It was good to take some time out from the project and all the work in general and will go back into semester 2 with a productive mind frame.

Month: February 2017

06/02/2017

This week I started back properly after the New Year break working on my project, I have researched and know what frameworks and tools I need to complete my Android project. This week was mainly doing the initial set up of the project in Android studio, setting up the XML files for displaying everything on screen. I have kept the design a more basic version of what I plan on finally plan on implementing as all the logic will be the same and I think it is better to get the basic functionality working and then focus on the aesthetics. I have set up all most of the Java files and am ready to start getting into it. My first step is to get the NoSQL Firebase database setup as well as Authentication (Login etc) , I have been following some tutorials and am ready to apply them to my own app.

13/02/2017

I have the authentication working with Google sign in and email, it was a bit more of a struggle than I anticipated to get it setup. Firebase is great when it is working as it should be but there is a large number of dependencies and files that are needed for it to work, I had some errors and being new to the whole thing it took a while to work out the issues. It seems to be working now after much trial and error.   I have a meeting with my project supervisor this week so will hopefully get some advice that I am on the right track.

27/02/2017

Meeting with my supervisor went ok, its good to check in and get some external feedback. He said I am on the right track and we have arranged another meeting for 2 weeks' time. I feel maybe I am not making as much progress as I need to be. It has been slow going, I have the guts of the database now set up but with Firebase I need to explicitly set up a user's "table" in the Realtime database in order to access users details. I am currently working on this but have found that the documentation from Firebase is not up to date in this area.

01/03/2017

I have been struggling a lot with getting the multiple writes of my User and Group object at the same time on one button press. I need to get this going for the rest of application to function correctly. From my reading I cannot see any way of doing the app without this functionality. This is the skeleton of the whole thing. In the meantime I have been doing more tutorials and work with Android and Firebase through Pluralsight and Udacity

10/03/2017

Just had a meeting with Padraig yesterday and outlined my concerns with getting the app finished. I am still stuck on this fundamental part and cannot see a way passed it. There does not seem to be any information online anywhere. Either on the docs for Firebase or anywhere else

25/03/2017

Had a meeting with my supervisor a few days ago and he suggested I simplify the application slightly if I am still having issues with getting it done. I think this may be a good idea and I discussed disposing of the booking system as a start. I am still worried though as I cannot get this one part working to get the synchronous writes to the database.
I was also made redundant from work and they are trying not to give me a severance package so outside of the app I have other issues to contend with. I will have to find a new job and try get the money from my old company. Padraig said this would be our last meeting.

05/04/2017

I have not had much of a chance to look at the app this week, I have been trying to sort out my redundancy, dole and a new job. The Workplace relations committee says I should have a case to get a severance package.

12/04/2017

Back on the app and still little to no progress made, I have had other college commitments to get finished so haven't had a chance to look much at it. Very concerned now as I only have a few weeks left.

24/04/2017

Finally cracked the issue with multiple writes to the database, it was not in the documentation for Android but I found it in the JavaScript one. I wish that I had looked here earlier. Now to crack on and get the rest done, even though I cracked this I have a lot more work to do.

01/05/2017

Made a lot of progress on the app today, have the guts of the player management done. It was very complex getting the data synch correctly across all devices, and I have had to do this multiple times . Seems every time I think I am nearly finished I find more scenarios not covered.

07/05/2017

Phew, the app is finished and just in the nick of time, I gave a build to a group of lads playing five a side at my gym to use to organise their match tomorrow and will see what they come back with. I gave them a quick UAT testing page to give me feedback. Now to complete the document.

5.4 References

Cloud.google.com. (2017). roadmap-overview-firebase-flexible-playchat-client. [online] Available at: https://cloud.google.com/solutions/mobile/images/firebase-flexible-playchat-client.svg [Accessed 9 May 2017].

Stack Overflow. (2017). Show hide fragment in android. [online] Stackoverflow.com. Available at: http://stackoverflow.com/questions/14347588/show-hide-fragment-in-android [Accessed 9 May 2017].

Android SearchView Tutorial (2017). Android SearchView tutorial: EditText with phone contacts search and autosuggestion (including source code). [online] Available at: https://looksok.wordpress.com/2013/06/15/android-searchview-tutorial-edittext-with-phone-contacts-search-and-autosuggestion/ [Accessed 9 May 2017].

Stack Overflow. (2017). Firebase check if child value in node A matches child value in node B. [online] Stackoverflow.com. Available at: http://stackoverflow.com/questions/43742366/firebase-check-if-child-value-in-node-a-matches-child-value-in-node-b [Accessed 9 May 2017].

Gist. (2017). Move or copy a Firebase path to a new location. [online] Available at: https://gist.github.com/katowulf/6099042 [Accessed 9 May 2017].

Firebaseui.com. (2017). FirebaseListAdapter. [online] Available at: https://firebaseui.com/docs/android/com/firebase/ui/FirebaseListAdapter.html [Accessed 9 May 2017].

Stack Overflow (2017). How to remove items from firebase RecyclerView. [online] Stackoverflow.com. Available at: http://stackoverflow.com/questions/36252478/how-to-remove-items-from-firebase-recyclerview [Accessed 9 May 2017].

Isixsigma.com. (2017). Defect Prevention: Reducing Costs and Enhancing Quality. [online] Available at: https://www.isixsigma.com/industries/software-it/defect-prevention-reducing-costs-and-enhancing-quality [Accessed 9 May 2017].

Firebase. (2017). Documentation | Firebase. [online] Available at: https://firebase.google.com/docs/ [Accessed 9 May 2017].