Dublin Institute of Technology

Institiúid Teicneolaíochta Bhaile Átha Cliath

Football Manager Companion
Final Year Project Report

DT228

BSc in Computer Science

Scott Lynch

Diana Carvalho E Ferreira

School of Computing

Dublin Institute of Technology

25-03-2017

Abstract

When you’re playing a sport for a team it's always entertaining to record statistics and compare results between one another. This project aims to provide the manager of a football team with the necessary functionality to just this. By allowing a manager create a team, create profiles for his players, record statistics and alter their player’s ratings the application can be integrated with ease into a team.

Regarding the player’s interaction with the system, the players can view their season statistics in a clean design using lists and graphs. Players will also be able to enter a league against their teammates by picking three players off their team. Fantasy football is a feature where players are allocated points for different activities related to their position. Once a player has selected their three players their total score is then displayed in a league table which is available to view by all the squad members.

Declaration

Acknowledgements

First, I would like to thank my parents for their continued support over the last few years. Their encouragement has made this opportunity possible as at times I felt like quitting and felt I wouldn’t be able to complete this course. Secondly, I would also like to thanks my friends for their continued support over the years and without all of them, this wouldn’t be possible. Finally, I would like to thank my supervisor Diana Carvalho E Ferreira for taking time out of her weeks to help me through the year by offering her advice and guidance as I worked on my project.

Contents

Table of Figures

Chapter 1 – Introduction

This project involves a web application and an android application. The web application allows a user to act as the manager of a football team by firstly registering and creating a team. Once a manager has created a team, players can then be added to that team. When a team has been created, the user can then access his account through the android application which will allow either the player or manager login.

When the manager logs in he can monitor, and alter his player's statistics which will then be displayed to the player. These statistics include attributes such as a player's defensive and attacking ratings out of 100.

The manager will act as the primary user where they are required to input the data which is displayed to the players when they log in.

The Android Application includes a fantasy football feature which in brief is a points system based on different activities that occur during a match. These activities for now include goals and assists which have different values related to a players position. This will be discussed in further detail in Chapter 1, Section 1.

1.1 Project Overview

Football is without question one of the most played sports around the world and is the most followed with an estimated 3.5 billion fans across Europe, Africa, Asia and America [1]. According to FIFA, there are 265 million people playing football with teams. This number was recorded in the last big count in 2007 [2] and has certainly risen since then as has the technology industry with most people now having their own smartphone which they have access to the internet and applications through. The figure below which was recorded in 2014 illustrates the astronomical rise of smartphones.

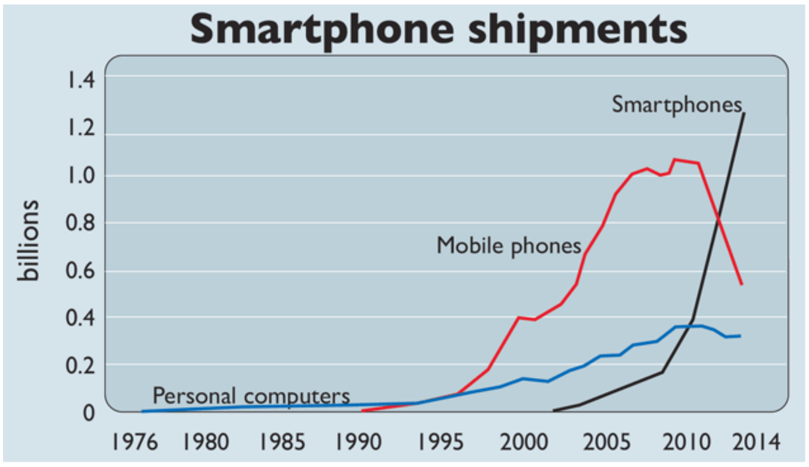


Figure 1.1: Smartphone shipments rise [3]

In figure 1.1, it shows that smartphone shipments rose dramatically between 2002 and 2013 with a dramatic increase between the years 2008 and 2013 where shipment sales rose from approximately 200 million to 1.3 billion. The correlation between the number of people that play football and the rise in smartphone shipments indicates that the smartphone users have the ability to download Football Manager Companion and the number of people playing football provides a large target market.

Powerleague is an operator of small sided football artificial pitches with approximately 45 football centers across the UK and Ireland. From my own experience, the rise in the number of people playing football in Dublin has grown massively in recent years with more and more people concerned about their physique and health which playing sport certainly helps. This has been quite visible as in previous years as pitches were cheaper and wouldn't require a booking. According to Powerleague, they now have over 10 million people every year use their service with 30,000 regular players participating in 5-a-side leagues and 30,000 players playing on a weekly basis [4]. Powerleague have now grown to become a community that will continue to grow. Powerleague provide the opportunity for business' and social players to form teams and compete in their leagues or if they are not interested in the competitive aspect pitches can be rented to play amongst each other.

In relation to the development of Football Manager Companion, an application with features more suited to accommodate the entertainment levels of the players there would certainly be interest from teams competing in leagues and as Powerleague are a strongly based community club there would be a large audience to promote towards.

Fantasy football is also another massive factor behind the application which would be one of the priority features to draw interest. Fantasy football is a website based on player's statistics from matches they've played in the Premier League, in England, with 4.2 million players across the UK and Ireland taking part [5]. Within the application, there will is a feature in place where players are awarded points based on a points system after matches have been played. The points system in place will make a calculation related to the player's position and the activities they may have achieved during the duration of the match. Currently, there are two activities implemented into the system. The first activity is a goal. The points system will allocate more points to a defender scoring a goal than to a striker scoring a goal as they are less likely to score goals on a regular basis. The second activity is an assist. The points system will once again allocate a certain amount of points to a player based on the position they play for this activity.

Introducing such a feature to an application and introducing it to a team increases competition and a player's motivation to achieve greater totals.

From my own experience of playing on football teams, tracking statistics and events that occur during matches would have been hugely beneficial and entertaining. Previously, I've created excel spreadsheets and input data accordingly after games had concluded. Appendix A I've attached an excel spreadsheet which indicates as to how these statistics were recorded in the past. Personally one of the factors in developing this application is a personal interest and I feel that it would be a lot of fun to have on any team and would only require one member of the team acting as the manager to input the data which can be done easily during a match.

1.2 Project Objectives

The goal of this project was to create a web application where a user could act as the manager of a team, create a team and manage that team from an Android Application.
The aim of the Android Application was to design an application from the perspective of two different users with the first perspective being the manager of a team and the second perspective being a player on the team.

The idea is that the application would primarily be used on the day of a match that would allow a manager to select his team for the day and when fitting, allocate different activities to different players during a match. This data is then displayed back to the player when they log in through the use of graphs and lists.

When a player logs into the application they have the ability to view the overall squad and each of their related statistics including their defensive and attacking statistics along with their position and also their total goals and assists to date for the season.

The player also has the ability to create a mini team of his own where he can act as the manager by selecting three players from the entire squad. When the player has selected these players he is entered into a league against his teammates where each of their three players totals is combined for their total score. This data is then displayed to the player where they can visibly see which player in the league has the most combined total from their three players and which three players are on their team.

1.3 Project Challenges

Regarding a project of this magnitude, there were many challenges associated with the development of Football Manager Companion which I will discuss in no specific order.

1.3.1 Setting up a Web Server

Setting up a web server that would allow a MySQL database communicate with both the Web Application and Android Application provided a great deal of difficulty throughout this project. Initially, tutorials were followed on YouTube and on Amazon to create an EC2 instance, however, as I had little experience regarding the technology and services that were being provided it proved to be very difficult.

Moving away from Amazons services and creating an Ubuntu Server on Okeanos also came with its challenges however after much trial and error resulted in success.

1.3.2 Node.js - RESTful API

Getting the RESTful API to communicate with the Android Application proved to be difficult initially. Originally after implementing the API, it was required that I start it every time by typing the command 'npm start' which communicated with a localhost MySQL database. Attempting to migrate the API to the server provided a lot of difficulties when attempting to make a request such as 'Error: ECONNREFUSED' relating to port 3306 and 'mysql_native_password packets out of order'. Once solutions to these errors were implemented responses from the API returned NULL. An adjustment to how the database connection was established was made which led to the correct responses from the API. The issue now was that the API had to be manually started every time. To resolve this there is a feature provided by node with the purpose of keeping a child process running continuously and to automatically restart it when it exits unexpectedly.

Chapter 2 – Research

2.1 Alternative Existing Solutions

In this section, I will discuss applications with a similar concept related to managing a football team.

2.1.1 Teamsnap

Teamsnap is an application predominantly based off manager input as is my proposed application. The idea behind this application is to make life easier for coaches, players, and parents [6]. This application allows managers to upload team sheets prior to matches, schedule events on a calendar which syncs to user's own calendars, it provides a group chat feature and allows members of a team to confirm their availability for events. In my own opinion, this application would be great for a children's team where parents are in charge and don't have the means to contact each other whereas with my own application members of the team already have a group chat through WhatsApp which would make the group chat feature quite redundant in my case and in many others.

2.1.2 Teamer

Teamer is an online management system created and designed to eliminate the hassle managing a sports team [7]. However, as I have used teamer in the past with a football team and this application never took off. It was very difficult to navigate through the different features and required a lot of signups for the players and then to navigate to the team and join their team. This inconvenience discouraged a lot of people and although the application provides wonderful features it has never quite become the success it could have been.

Applications such as Teamer and Teamsnap are directed more towards the management side of things, such as seasonal finances, availability for events or storing team documents whereas I plan to implement a fantasy football feature and player statistics which will hopefully add more of an interest from the player's side of things.

2.1.3 ClubFantasy

ClubFantasy is a Web Application where a manager can set up a team and invite players from their squad to a league by issuing them with a code to join. ClubFantasy allows a player to create their own team where their player's totals will be compared to other members of the league's totals. ClubFantasy is currently released as a beta with the aim of improving regarding feedback from users.
ClubFantasy is a similar application to this project that allows team members to compete against one another in a league created by the manager. [8]

2.2 Technologies –

In this section, I will discuss the different technologies that have been considered for this project and the reasons for selecting one technology over the other. This section also includes the selection of mobile operating system and the programming languages used.

2.2.1 PHP

PHP is predominantly a server sided a scripting language. PHP pages contain HTML embedded with code to perform tasks [9]. PHP code is executed on the server which generates the HTML which in turn is then sent to the client. PHP provides a long list of features aimed at beginners to advanced programmers. One of the most significant features in PHP is the support it provides for a variety of databases. The features PHP provides are beneficial as this project is further developed such as handling sessions, HTTP authentication, connection handling and PHP also provides basic error handling, therefore it was used in this project to develop the Web Application.

2.2.2 CSS

CSS is a design language which describes how HTML elements are to be displayed on screen [10]. CSS allows you to modify design features such as font size, the ability to use different fonts, alter the colours of different aspects and virtually most things that the user is presented while also providing the ability to style each page individually with the option to also use the same style sheet for multiple pages, therefore CSS was the language of choice for styling the Web Application.

2.2.3 SQL vs NoSQL

SQL is the language of choice for the development of this project, to gain access to the database. After researching and comparing the use of SQL and NoSQL, the features provided by SQL were more suited to the requirements.

SQL stores related data in tables whereas NoSQL stores related data in JSON style name-value documents. SQL conforms to a fixed schema meaning that for each record columns must be defined before any data is entered and to amend the database it would involve going offline for a period although with NoSQL schemas are dynamic and information can be added at any time [11]. This may be beneficial to a larger company where change is necessary on a regular basis but for this project, SQL provides more than sufficient needs to store and retrieve data as there won't be a need to modify tables regularly.

2.2.4 MySQL

For this project, it was decided a MySQL database system would be used. *MySQL is an open source relational database management system. Information in a MySQL database is stored in the form of related tables [12].* MySQL was installed through Ubuntu during LAMP installation. *LAMP stack is a group of open source software used to get web servers up and running. The acronym stands for Linux, Apache, MySQL, and PHP [13].*

The installation of LAMP also provided phpMyAdmin which provided sufficient means to maintain the database.

2.2.5 AngularJS

AngularJS is a JavaScript framework that extends HTML attributes with directives and binds data to HTML with expressions [14]. AngularJS allows you to build an application one way and provides the ability to reuse your code across your destined targets, for example, the web, mobile or desktop applications. A key benefit for using AngularJS is that Angular implements MVC by asking you to split your application into MVC components and Angular does the rest whereas most frameworks that implement MVC ask you to split your application into MVC components and then you are required to write the code to bring them all together.

Angular is on its own in the frontend. It accesses all the data it needs through the Node API. Node hits the database and returns JSON information to Angular based on the RESTful routing. This way, you can separate the frontend application from the actual API. If you want to extend the API, you can always build more routes and functions into it without affecting the frontend Angular application. This way you can eventually build different apps on different platforms since you just must hit the API.

2.2.6 IDE Selection

Android studio is an Integrated development environment (IDE) for Android platform development which was announced in 2013 by Google, which led to a gradual decline in Eclipses market share which became outdated within one year [15].

Eclipse is a Java-based open source platform that allows a developer to create a customized development environment (IDE) from plug-in components built by Eclipse members [16]. Eclipse was a popular IDE used to develop Android Applications using the Android Developer Tools (ADT) plugin, however, the eclipse ADT plugin is no longer supported by Android [17]. Benefits associated with using Android Studio is that it is designed particularly for Android development to accelerate the development of your project and in general making it simpler is the aim of the IDE and beneficial features such as Code Completion to assist the developer. Figure 2.1 below highlights the decline of iOS devices in the past 10 years and the rise of Android which is one of the deciding factors in my decision to use Android studio over Eclipse and why I have decided to develop an Android application as opposed to an iOS application.

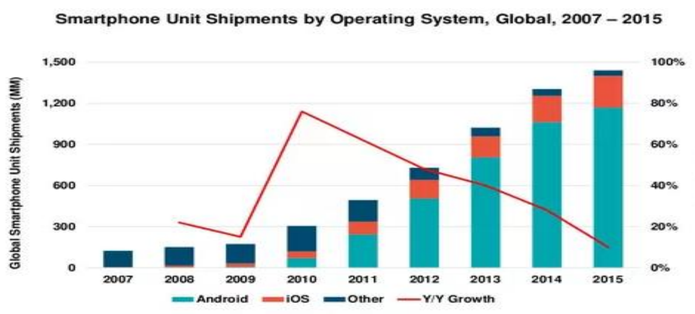
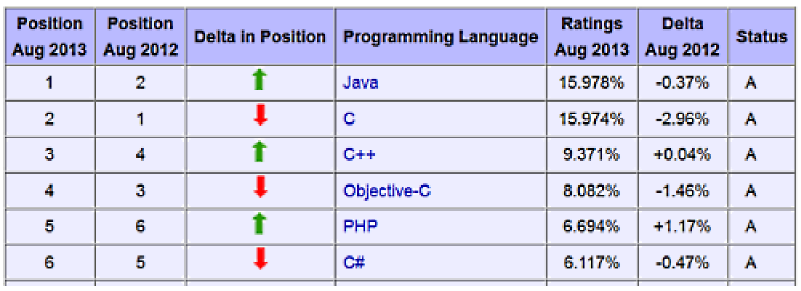


Figure 2.1: Rise of Android 1

2.2.7 Java

Java was the predominant programming language used throughout the development of the Android application using Android Studio. "Java is the heart of our digital lifestyle" with over 15 billion devices running Java, 10 million Java developers worldwide and is the number 1 platform for development in the cloud [19]. Java has proven to be a success with major companies adopting their practices with "Twitter migrating their core infrastructure to Java and now powers more than 400 million tweets per day" and "Netflix powers through 2 billion content request per day with Java-driven architecture" [20]. Java also provides a great amount of support online due to its popularity.

as I have previous experience using it from modules in the past and another factor in my decision is simply that Oracle provides great support for Java.
The figure below illustrates how as of august 2013 Java had risen to the top programming language.



Position Aug 2013	Position Aug 2012	Delta in Position	Programming Language	Ratings Aug 2013	Delta Aug 2012	Status
1	2	↑	Java	15.978%	-0.37%	A
2	1	↓	C	15.974%	-2.96%	A
3	4	↑	C++	9.371%	+0.04%	A
4	3	↓	Objective-C	8.082%	-1.46%	A
5	6	↑	PHP	6.694%	+1.17%	A
6	5	↓	C#	6.117%	-0.47%	A

Figure 2.2: Java Popularity [20]

2.2.8 JUnit

Android studio is designed is such a way that allows you to implement testing with greater ease. Android tests are based on JUnit which you can run as local unit tests on the virtual machine or as instrumented tests on an Android device [21].

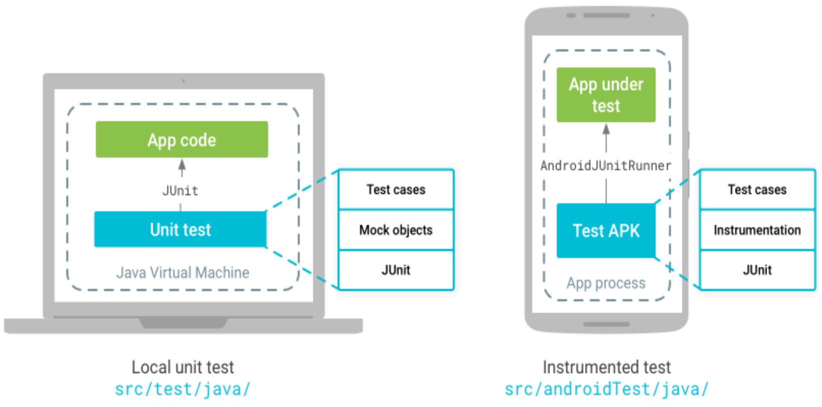


Figure 2.3: JUnit Testing [22]

Figure 2.3 illustrates how unit testing can be implemented. The local unit test runs locally on the virtual machine. These tests are used to minimize execution time when your testing has no android framework dependencies.

Instrumented unit tests are those that run on an android device or on an emulator. This form of testing has access to Instrumentation information which refers a class that may be instantiated for you before any coding begins, allowing you to monitor the interaction between the application and the system. [22].

Tutorials are provided online with the sufficient steps to implement and begin uniting testing through JUnit.

2.2.9 Server Research

Googles App Engine and Amazons EC2 are the two most common server hosting platforms that provide periods of free usage.

Googles App Engine seemed to provide an ideal platform for building a scalable web application with a database service to manage and maintain a MySQL database however it is restricted in what technologies you can use and for the development of this project it would be necessary to have freedom in the decisions of what technologies would be used.

The application, however, is now hosted through Okeanos which runs an Ubuntu virtual machine with an apache server. Okeanos provides great integration with GitHub and using the command line, changes that have been made to the project locally are pushed through to GitHub where through Okeanos, a repository in the virtual machine will have access to pull from that git repository with the project's folders and the changes will come into effect. Through Ubuntu's virtual machine there are two folders.

For the frontend, which the Web Application will use, there is a folder which includes the HTML and PHP files and the services that connect with and query the database.

For the backend Node was used which defined the structure which the Android Application will use. There is a folder which includes the API, which creates a connection to the database. The API folder contains two folders, the ManagerService and the PlayerService which contain their appropriate functions/services required to communicate with web server. These services are then called from the app which contains the route that will be added to the server's URL.

2.2.10 MPAndroidChart

MPAndroidChart is a powerful android chart view/graph view library used in an Android Application. To use MPAndroidCharts you must first import the repository and add it to your project. [23] Once the package has been added to your project you can then access a wide range of different chart types.

In this project the use of Bar charts and PieCharts were used to display different aspects of the player's statistics which are then presented to the player in an elegant design.

2.2.11 Android Volley vs AsyncTask

Androids Volley is an HTTP library to make networking for Android apps easier and faster [24]. This new framework was introduced to reduce the complexity involved in fetching data over a network.

AsyncTask enables the proper and easy use of the UI thread. This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers [25].

Through Android Volley there is no need to write the code for accessing the network. Volley utilizes the power of Android to access the network in the best possible way. Each time a network request is created, with volley, a background thread is spawned for network processing whereas AsyncTask calls the onPostExecute on the UI thread. Throughout the project when making post requests, such as inserts into the database, Volley was implemented which adds your request to a queue. Volley includes RequestQueue classes such as String requests and JSON requests which allow extending the URL's string with a JSON result which can then be processed by the API communicating with the database.

2.2.11 User-Centred Design:

The concept of user-centred design is that "approaches assume that the participation of relevant stakeholders will ensure an appropriate design outcome" [26, p1], where in relation to this application, as it is not technology driven and is requirements driven it was important to ensure an appropriate design and to meet requirements on time.

Chapter 3 – Design

This section describes the methodology, use case diagrams, sequence diagrams and database design related to this systems design. This section will also discuss the decisions related to the user interface design components.

3.1 Design Methodology

With regards to this project, it was important to implement a design methodology that would allow for flexibility and expansion. The approach which I decided to adopt was a combination of both the Agile development model and the Iterative model. Agile development seeks alternatives to traditional project management and can create and respond to change to succeed in an uncertain environment [27]. Through adopting a generic agile approach and introducing the Iterative model it allowed the application to be developed as a series of prototypes. Each prototype introduces new requirements, which will be designed, developed and tested.

Agile allows for flexibility as it assumes the first idea is not necessarily the best and final one and that learning happens continuously throughout the process [28]. Throughout development and implementation, this was an important feature as by planning for each prototype it allowed for the testing of each feature independently and to ensure it was functional before integrating it with the application. "The agile principles enunciated in the agile manifesto1 motivated and empowered software developers – relying on technical excellence and simple designs – create business value by delivering working software to users at regular short intervals" [29, p3]. As discussed in section 2.2.11, to combine the user-centred design with the agile methodology in iterations and alternatively produce a visually pleasing design as opposed to a simple design was the goal.

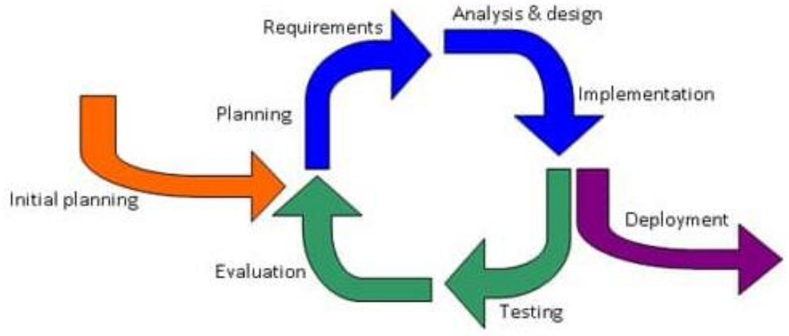


Figure 3.1: Iterative-Model [30]

The iterative lifecycle model does not require you to begin your project with a complete list of requirements. Development may begin with specifying and implementing a small part of the software which after competition of this cycle you can then review what you have achieved to identify further requirements. As illustrated in figure 3.1, the requirements are identified which are then divided into smaller multiple builds. Within each iteration, the requirements for each build are required for the design, implementation, and testing. Each iteration will gain more functionality than the previous build which will continue in this manner for the duration of the project. Using a Gantt chart requirements for each prototype were identified and given a time frame to be completed.

3.2 Design Components

This section will discuss the design components related to the user interface and the decisions as to why components were designed in such a way.

3.2.1 User Interface Design –

Initially during the development of the Android Application when incorporating lists the simple list item was used. The simple list item is an inbuilt layout file which can be attached to a listview with the help of an adapter. Figure 3.2 indicates the initial view which was implemented to display a team.

"Thumbnail images improve text-only lists because they look appealing, help identify items, and establish a generous height for the list items." [31, p459].

Relating to this information, an image view was introduced to the list which involved creating a custom list adapter. For each listview item, a custom row layout is assigned to it in the adapter class as opposed setting the layout to the simple_list_item_1. Creating the list adapter allows for customisation and flexibility to enhance the users experience.

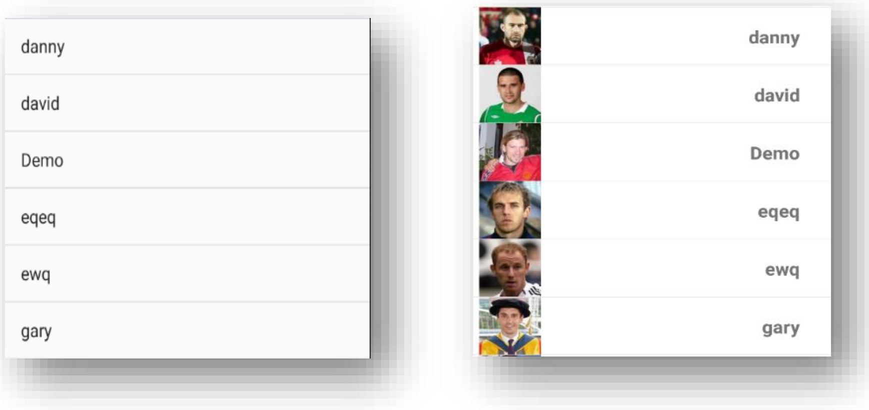


Figure 3.2: Thumbnail-and-Text List

Figure 3.3 indicates the use of an AutoCompleteTextView. “*Another flavor of field is one that offers auto-completion, to help users supply a value without typing in the whole text. That is provided in Android as the AutoCompleteTextView widget” [32, p85].* The suggestions are extracted from a collection of strings which have been retrieved from the database. To improve the user’s interaction the AutoCompleteTextView is implemented which is set to display names of players based on the first two letters as can be seen below.

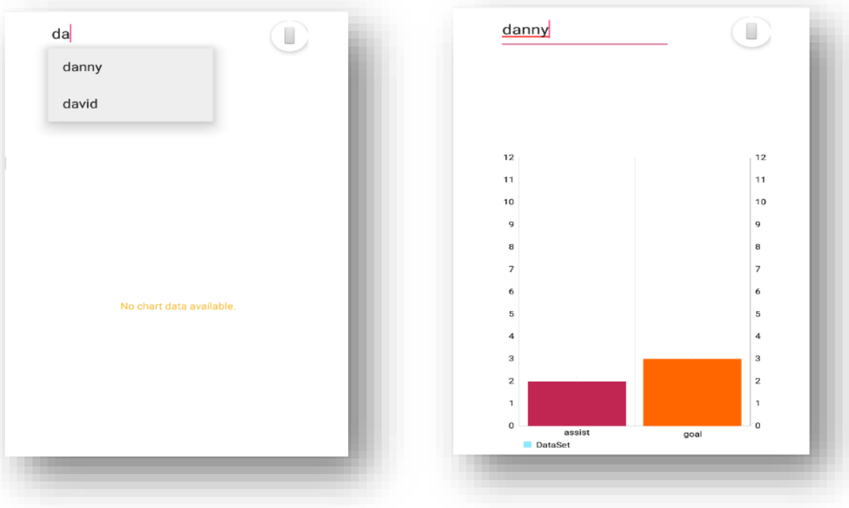


Figure 3.3: Searching for a Players stats

Originally when the user logged in the were greeted with a row of buttons where they could navigate between activities as illustrated in figure 3.4. According to, UNITiD Interaction Designers, when implementing navigation patterns, the drill down navigation can be used to improve the user interaction by organizing information in a list which will open the next level

[33]. Implementing a navigation drawer allows a low level of detail within a list which directs the user to the required information.

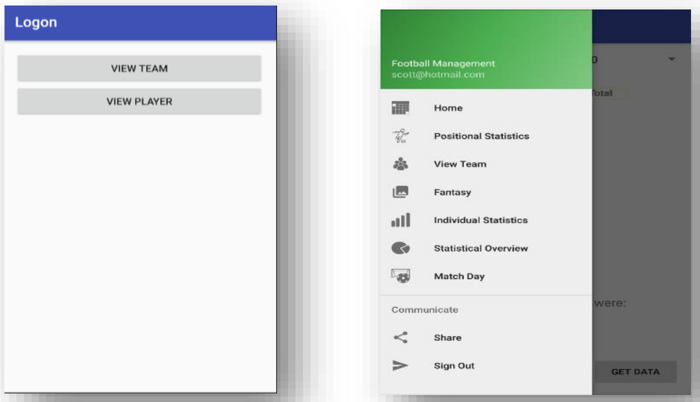


Figure 3.4: Navigation Patterns

The Vertical Stack Pattern is *"A mobile application needs a way to show its top-level navigational structure. A persistent toolbar across the top or bottom of each app page is one standard way to organize a mobile interface; tabs and full-page menus are two other common ways"* [31, p448].

Regarding this information, there are three types of positions a player can have within this project. There are defenders, midfielders, and strikers. Implementing a tab pattern for this situation seemed fitting to allow the manager of a team scroll between his players sorted by position as illustrated below in figure 3.5. Navigation between tabs can be accomplished by pressing the required tab or scrolling left or right between fragments.

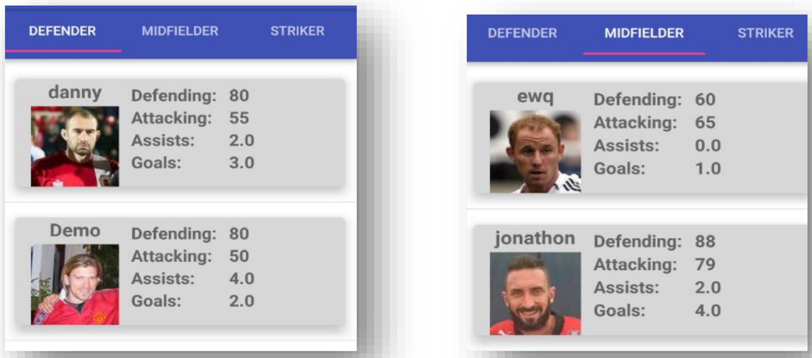


Figure 3.5: Tab Pattern

The grid of equals is content arranged in a grid where each item follows a common template. It is said to use the Grid of Equals *"when a page contains many content items that have similar style and importance"* [31, p149]. With this information, when a match day arrives and the manager navigates to the Match Day tab from the navigation drawer there is a list of his players. As each player is of equal importance the players are displayed in grid form with their image and username.

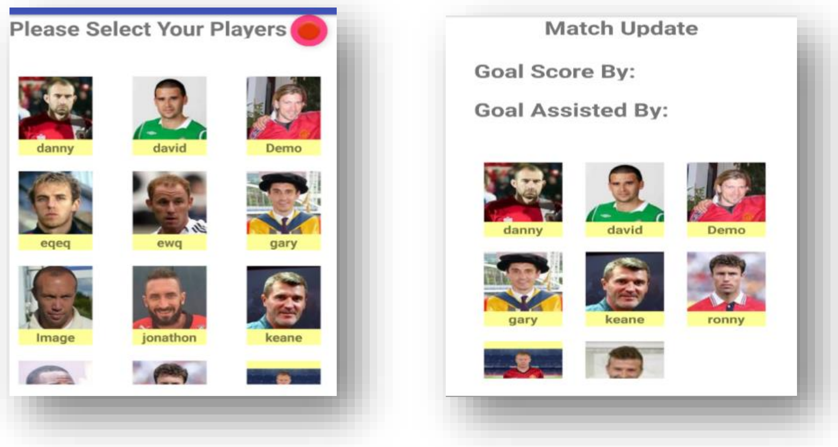
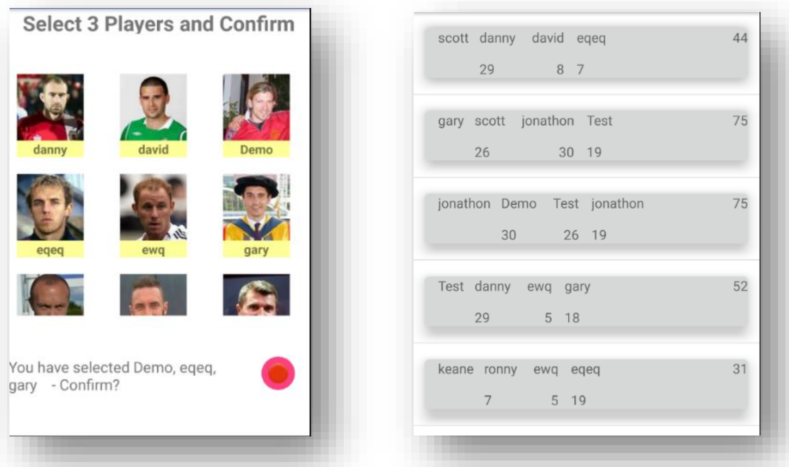


Figure 3.6: Match Day / Fantasy Football Input

When a player has logged in they will compete in a league against their own teammates. Figure 3.7 shows the initial screen where they player can choose 3 players from his squad. Once a player has selected his three players he can act as the manager of that team where the points from his selected three players are combined into one total where the players can see who has earned the greatest score.



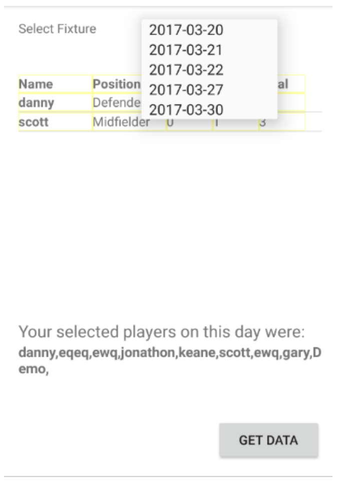
You have selected Demo, eqeq, gary - Confirm?

```
scott  danny  david  eqeq        44
       29            8  7
gary  scott  jonathon  Test      75
      26               30  19
jonathon  Demo  Test  jonathon   75
          30          26  19
Test  danny  ewq  gary           52
      29          5  18
keane  ronny  ewq  eqeq          31
       7           5  19
```

Figure 3.7: Player Fantasy League

When getting input from a user, the spinner can be used if a user must select an item from a predefined list. A spinner can save valuable screen estate and help users to make a choice "*Spinners are useful for quick input of accurate date*" (Michael Martinez) [33].

Regarding this project, a spinner seemed fitting to allow the user select a date from the drop-down menu as illustrated in figure 3.8 which in turn is used to query the database.



Your selected players on this day were:
danny,eqeq,ewq,jonathon,keane,scott,ewq,gary,Demo,

Figure 3.8: Spinner

3.2 UML Specification

This section of the document describes the use cases for this project. Through use case diagrams, a visualization of how the user interacts with the system is explained.

3.2.1 Login Use Case

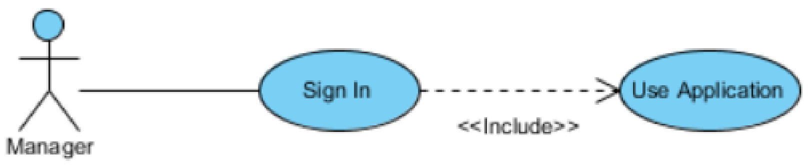


Figure 3.9: Login Use Case

Figure 3.9 illustrates a use case diagram that explains the flow of events when a registered manager enters their details into the application. A manager cannot use the application unless they are signed into their account.

3.2.2 Manager Interaction Use Case

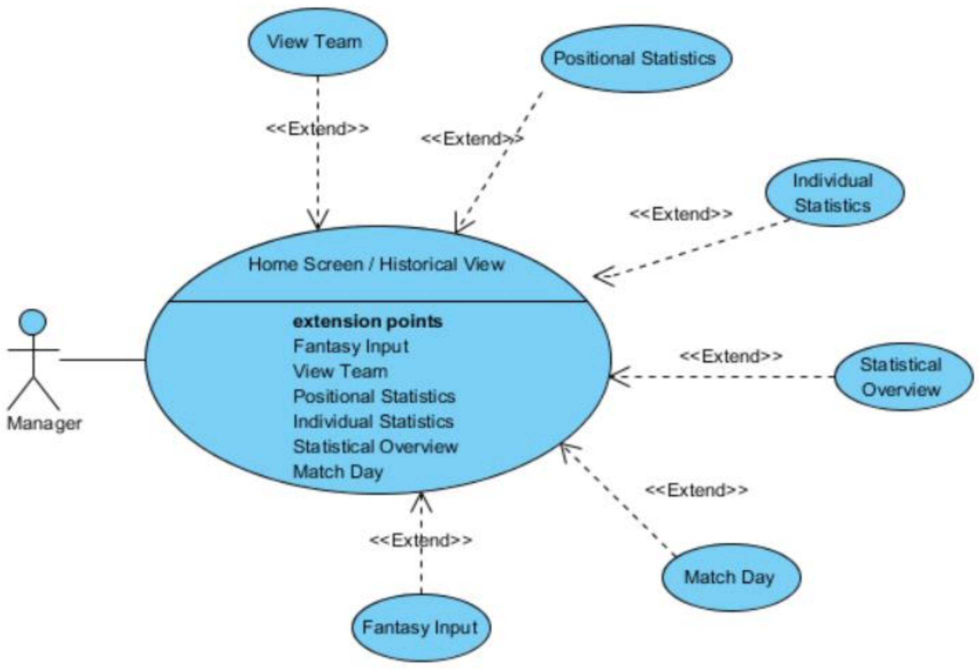


Figure 3.10: Manager Interaction

Figure 3.10 illustrates a use case diagram that explains the flow of events when a registered manager has signed into the application.

3.2.3 Manager Add Player

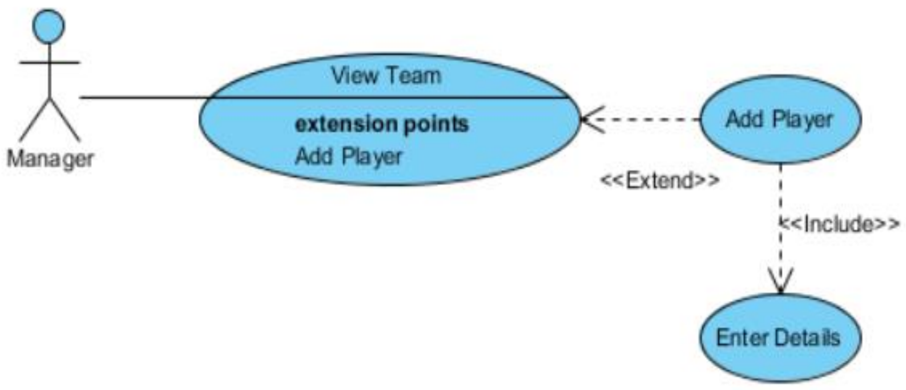


Figure 3.11: Add player

Figure 3.11 illustrates a use case diagram that explains the flow of events when a registered manager has signed into the application and wishes to add a player.

3.2.4 Manager Browse Fixture

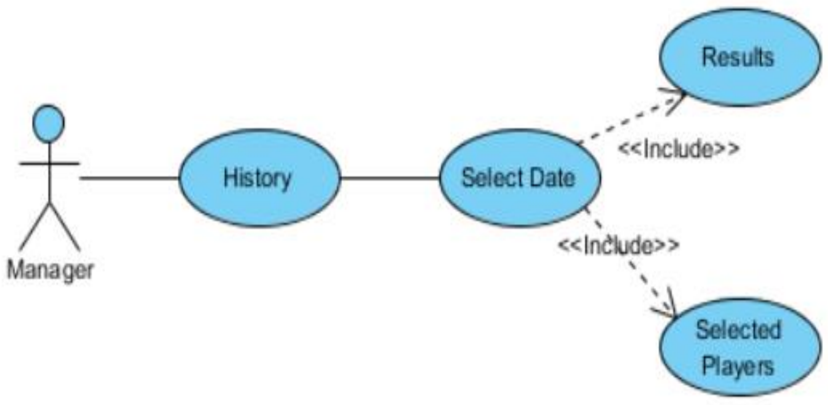


Figure 3.12: Browse History

Figure 3.12 illustrates a use case diagram that explains the flow of events when a registered manager has signed into the application and wishes to browse events that occurred on a date.

3.3 Sequence Diagrams

3.3.1 Manager Add Player

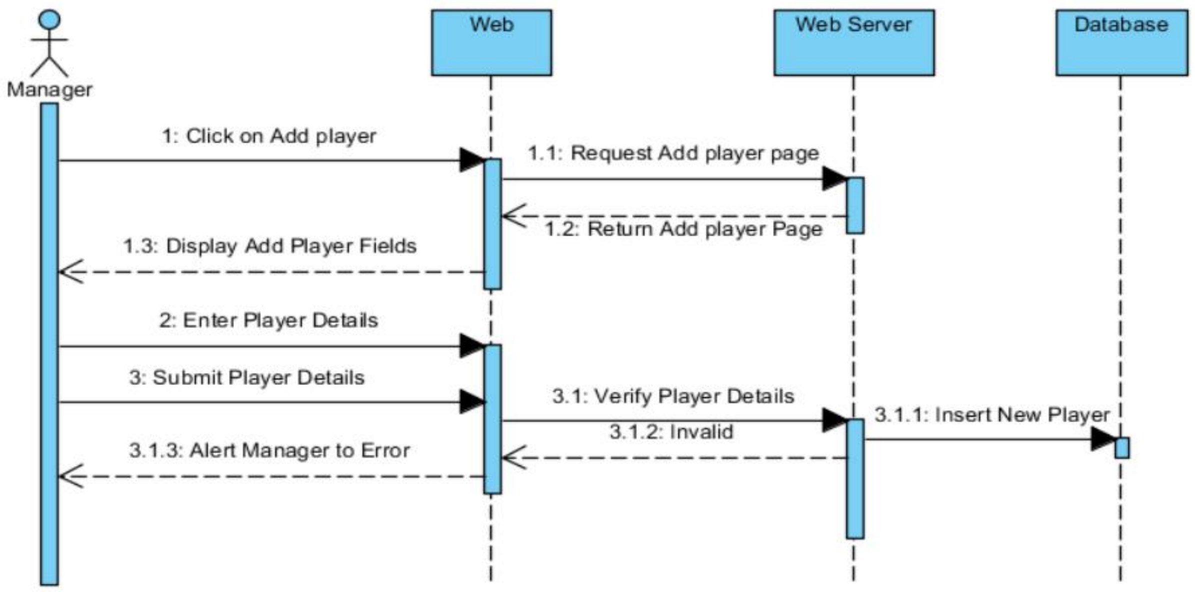

Manager
Web
Web Server
Database
1: Click on Add player
1.1: Request Add player page
1.2: Return Add player Page
1.3: Display Add Player Fields
2: Enter Player Details
3: Submit Player Details
3.1: Verify Player Details
3.1.1: Insert New Player
3.1.2: Invalid
3.1.3: Alert Manager to Error

Figure 3.13: Add Player Sequence Diagram

Figure 3.13 illustrates a sequence diagram that explains the flow of events when a registered manager has signed into the application and wishes to add a player to his team.

3.3.2 Match Day Dates Request Sequence Diagram

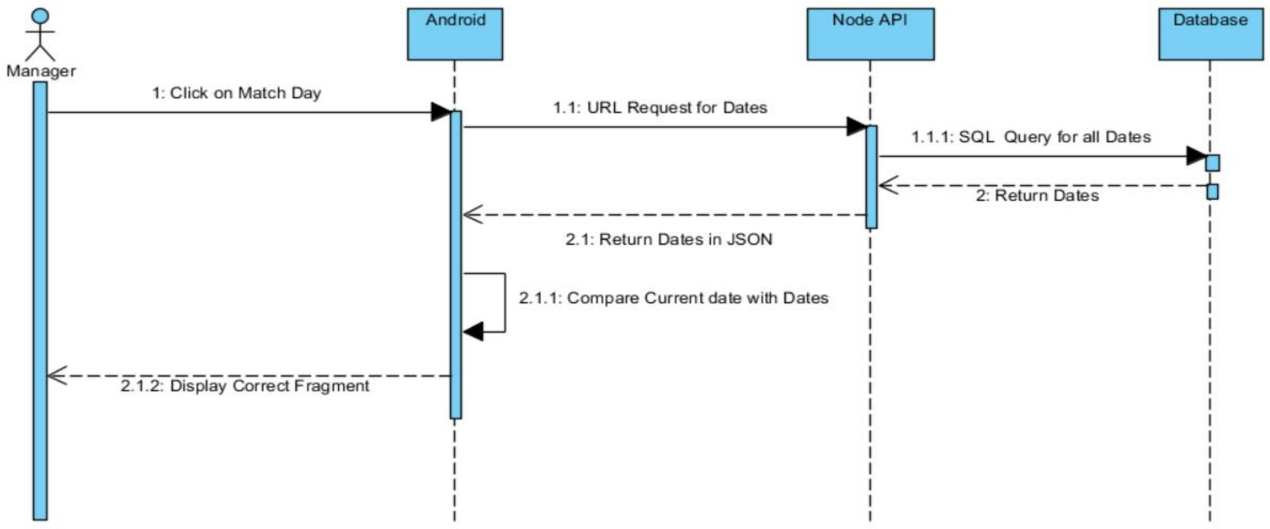

Manager
Android
Node API
Database
1: Click on Match Day
1.1: URL Request for Dates
1.1.1: SQL Query for all Dates
2: Return Dates
2.1: Return Dates in JSON
2.1.1: Compare Current date with Dates
2.1.2: Display Correct Fragment

Figure 3.14: Match Day Fragment Sequence Diagram

Figure 3.14 illustrates a sequence diagram that explains the flow of events when a registered manager has signed into the application and clicks on the match day fragment.

When the manager navigates to the match day tab the system will make a request as to whether a squad has been submitted for the current date. If no squad has been submitted for the current date the manager is prompted to select his players. Once a squad has been submitted the manager can assign the activities to players.

3.4 Activity Diagrams

3.4.1 Manager Submit Squad

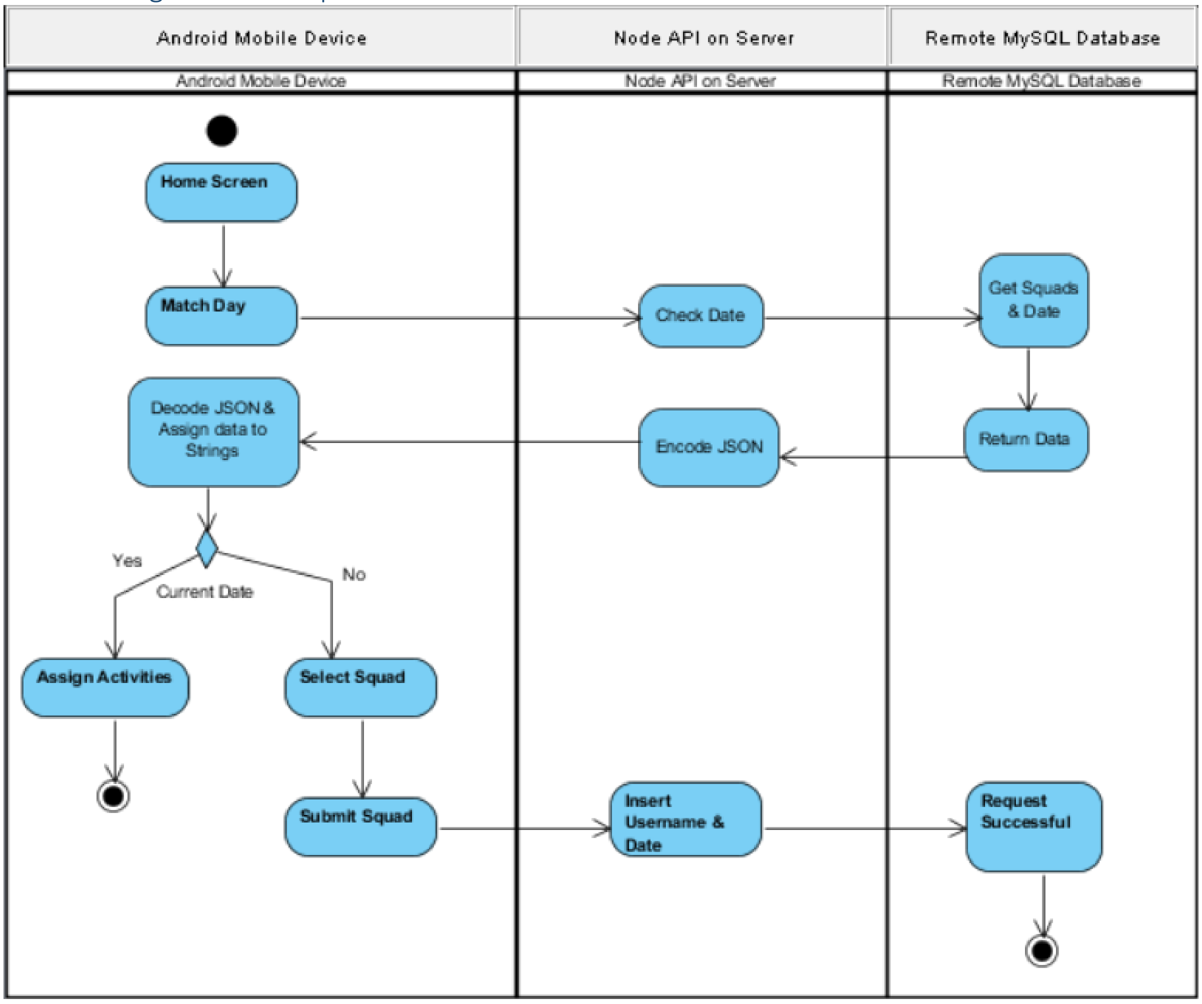


Figure 3.15: Squad submitted activity diagram

Figure 3.15 illustrates exactly what happens when a manager navigates to the match day tab from the Android Application. When the Application initially loads, a request is made to the

API with the current date which in turn is used to query the database. If there is data in the database related to the current date the squad for that date is returned to the Application in JSON. Once the JSON has been processed by the Application it is presented to the manager is a clean design where he can assign activities for the match to his players.
If there is no date in the database related to the current date the manager is prompted to select his group of players which are then sent to the database with their username and date once the manager has confirmed.

3.4.2 Player View League Standings

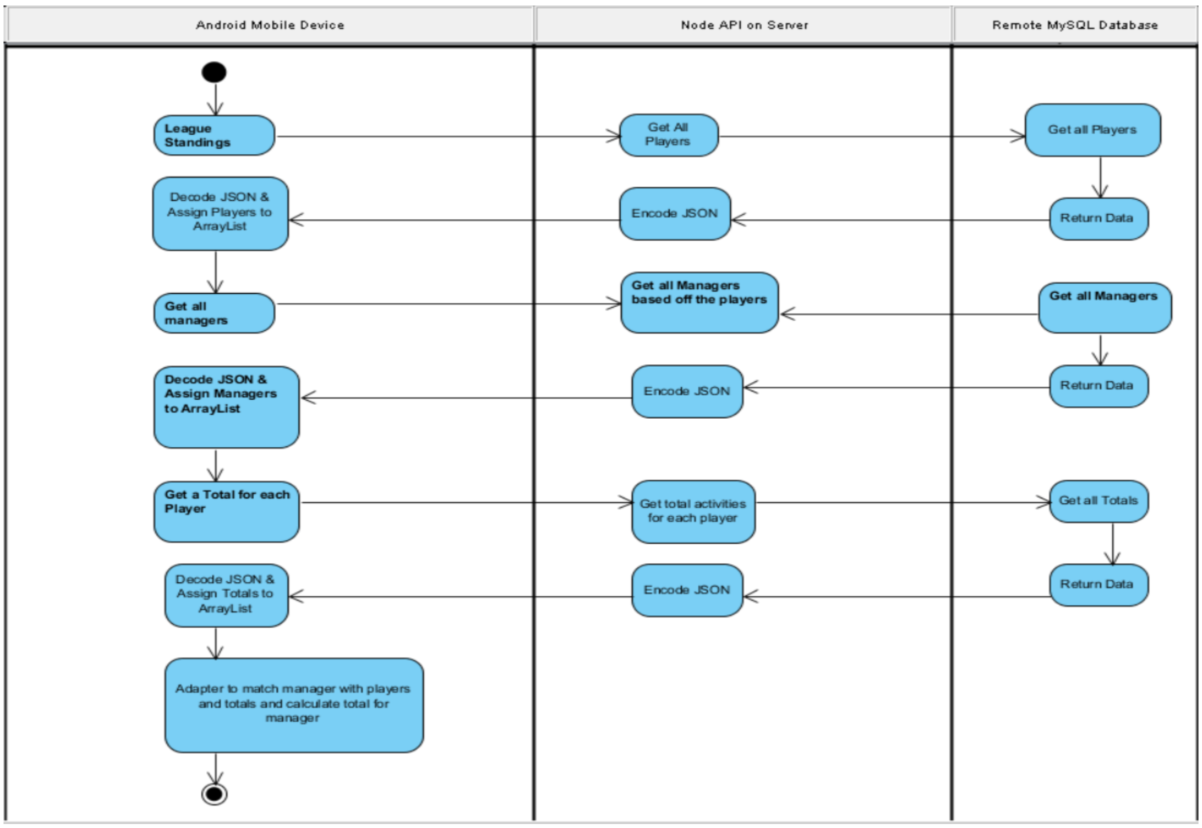

Android Mobile Device
Node API on Server
Remote MySQL Database
League Standings
Get All Players
Get all Players
Decode JSON & Assign Players to ArrayList
Encode JSON
Return Data
Get all managers
Get all Managers based off the players
Get all Managers
Decode JSON & Assign Managers to ArrayList
Encode JSON
Return Data
Get a Total for each Player
Get total activities for each player
Get all Totals
Decode JSON & Assign Totals to ArrayList
Encode JSON
Return Data
Adapter to match manager with players and totals and calculate total for manager

Figure 3.16: View League Standings

Figure 3.16 illustrates the process the Android Application undergoes when a player wishes to view their Fantasy Football League standings. When the fragment loads, all the players are retrieved from pleague table in the database where from the API is returned to the Application in JSON where the players are assigned to an ArrayList. A similar process is performed to retrieve the managers of the teams. When this has been done the totals for each player in the players ArrayList are retrieved. The reasoning behind this is that as the manager is a player and retrieving the totals all at once when the fragment initially loads creates an issue with the manager’s total being added to the total score. The process I have indicated above identifies this solution where the data can be manipulated from Android Studio through JSON and Java.

3.6 Entity Relationship Diagram

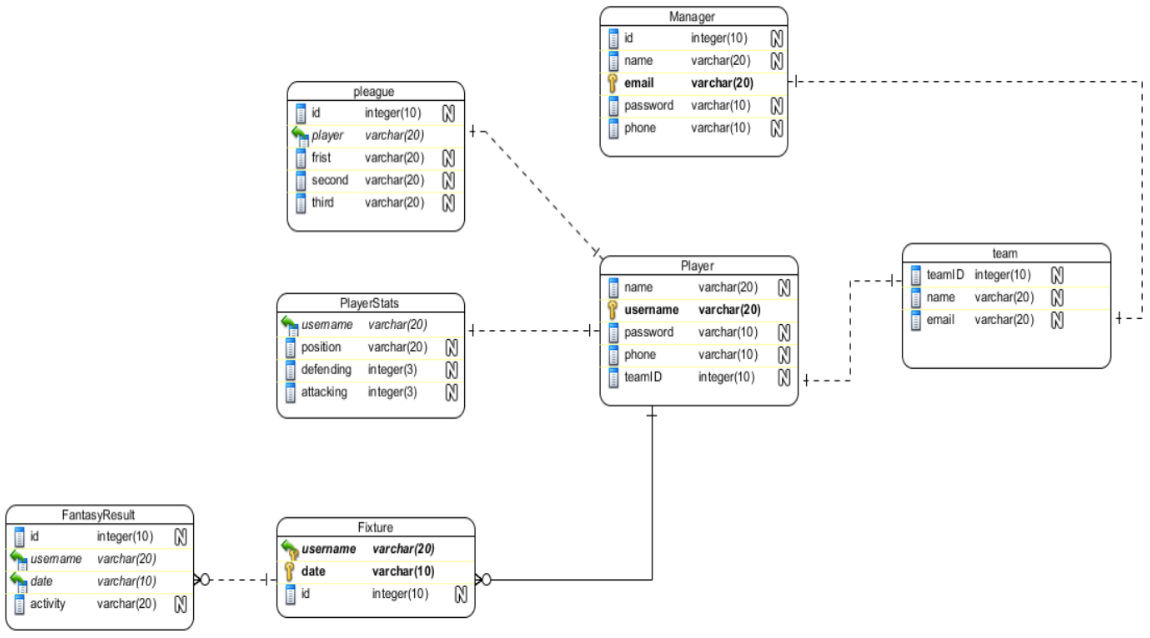


Figure 3.18: ERD

Figure 3.18 illustrates the database design for this project which will be discussed in more detail below.

Manager:

This is where the manager's details are stored when they create an account. The primary key email is unique and the ID auto increments for each new account.

Team:

This is where the team is stored. Email is the primary key which relates to the email of the manager. The teamID is auto incremented.

Player:

This is where the player's details are stored when they create an account. The primary key username is unique. TeamID is populated with the teamID that relates to the manager who created the player.

PlayerStats:

This is where statistics for a player are stored. Username relates to the player's username. Statistics for each player are stored in this table along with the position of the player.

Fixture:

This is where the players who took part in a fixture are stored with the date.

FantasyResult:

This is where the player’s username and the date are stored with the activity they registered within a match.

Pleague:

This is where the players’ names are stored as they enter a league against one another. Each field relates to the username field in the player's table.

3.6.2 Database Trigger

Within the database structure, there is a trigger that when a player is added to a team they are assigned default values for their statistics and a default position which the manager can later alter through the Android Application. See appendix B.

Chapter 4 - Architecture & Development

4.1 System Architecture

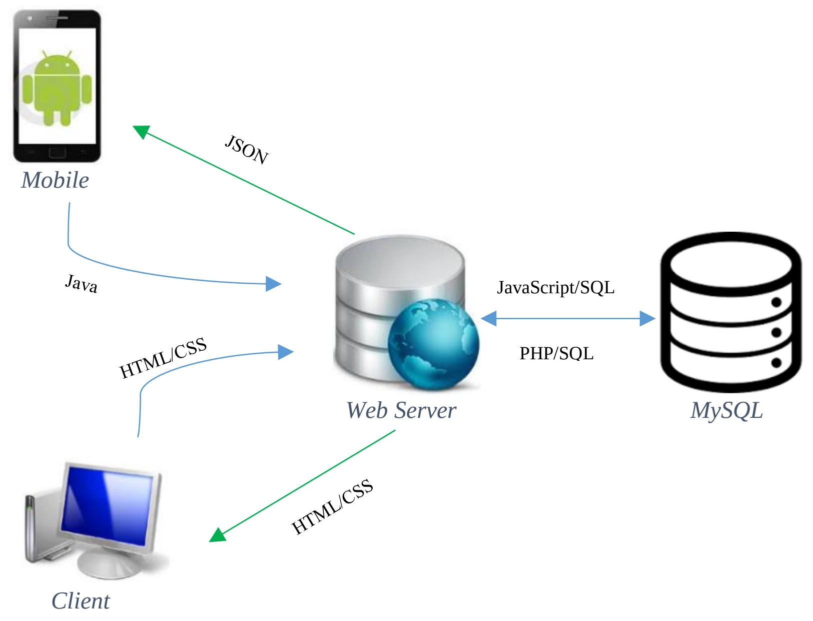


Figure 4.1: Technical Architecture Overview

As illustrated above in figure 4.1, the architecture that was used in this project is Three Tier Architecture. The Client, the Server (Application layer) and the Database. The user interface is stored in the client which will only display the GUI on the front end and has no part to play in producing results. The application logic is stored in a RESTful API on the server which controls and operates the requested resources by calling on the database layer. The data is stored in the database layer which in turn provides the server layer with the requested data.

For the Web application, the client and the server communicate through PHP. The client makes a request to the server which in turn makes a request to the database. Through PHP the server creates a connection to the database where either the ManagerService, PlayerService or TeamService is called. Once one of these services is called they contain the relevant code to perform the request to the database. If the request is successful a response is sent to the server which displays the relevant information regarding the request.

For the Android Application, the user will make a request through the application which will communicate with the server where there is a RESTful API. Once the user's request reaches the server the RESTful API begins to communicate with the database. In the API, there are multiple services, the ManagerService and the PlayerService. Depending on the request that is made the app.js file will get the necessary function related to the URL request. Once the request has been processed, the response from the database will return the result to the server. Once the server has received a result it is then converted to JSON format which is sent to the Android application. Once the Android Application has received the response from the server the data is then displayed to the user.

4.2 Technology Requirements

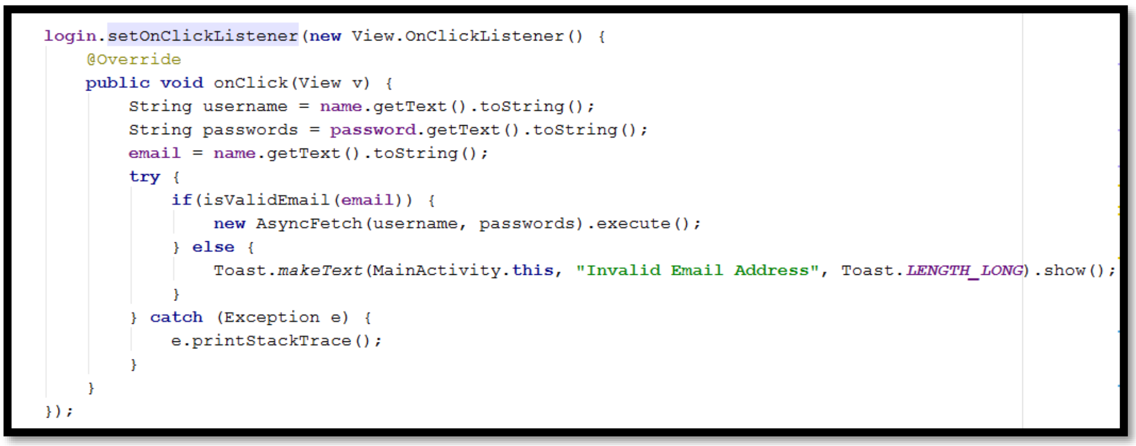
- A computer
- Android Studio
- Android SDK
- Android Mobile Device
- Okeanos Web Server
- PHP Scripts
- MySQL database
- JSON, XML
- JavaScript

4.3 Project Demonstration & source code layout

4.3.1 Android Development

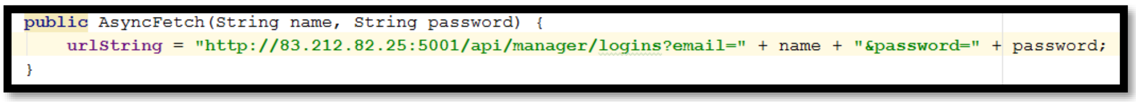
4.3.1.1 Login Activity

When a user enters their login information from the Android Application and clicks the login button a new AsyncTask is instantiated which will send a POST request with the email and password to the API.



```java
login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String username = name.getText().toString();
        String passwords = password.getText().toString();
        email = name.getText().toString();
        try {
            if(isValidEmail(email)) {
                new AsyncFetch(username, passwords).execute();
            } else {
                Toast.makeText(MainActivity.this, "Invalid Email Address", Toast.LENGTH_LONG).show();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
```

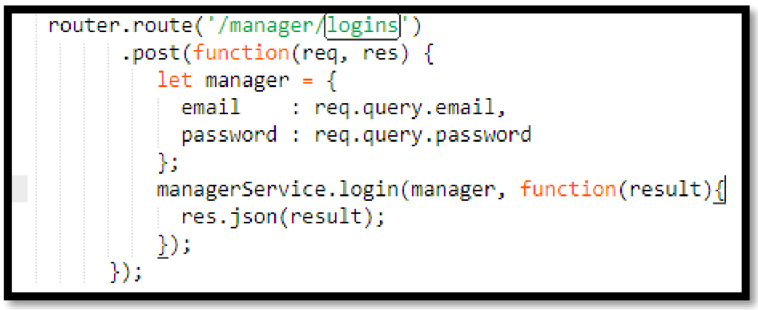
Code snippet 1 – Login Button



```java
public AsyncFetch(String name, String password) {
    urlString = "http://83.212.82.25:5001/api/manager/logins?email=" + name + "&password=" + password;
}
```

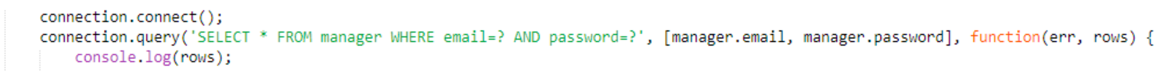
Code snippet 2 – URL Request

When the request arrives at the app’s route it is directed to the manager service with the email that has been entered and the password where a query is executed.



```javascript
router.route('/manager/logins')
    .post(function(req, res) {
        let manager = {
            email    : req.query.email,
            password : req.query.password
        };
        managerService.login(manager, function(result){
            res.json(result);
        });
    });
```

Code snippet 3 – App route



```javascript
connection.connect();
connection.query('SELECT * FROM manager WHERE email=? AND password=?', [manager.email, manager.password], function(err, rows) {
    console.log(rows);
```

Code snippet 4 – Login Query

If the result is a success and the email and password are correct the user is directed to a new activity.

4.3.1.2 Navigation Drawer

When a manager gains access to their account they can navigate between different fragments through the navigation drawer. The navigation drawer consists of multiple XML layout files including a header file which displays the "Football Management" and the email address of the user logged in.

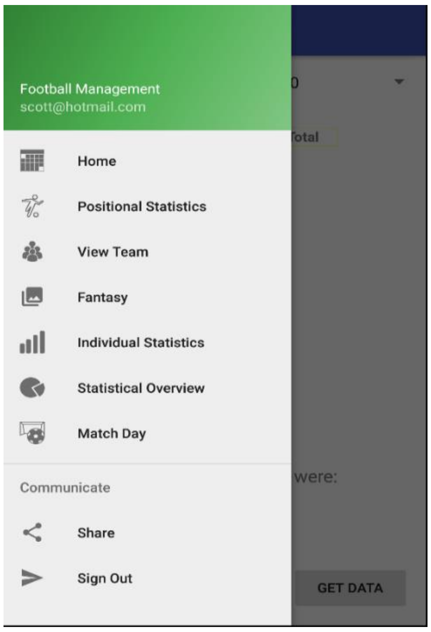
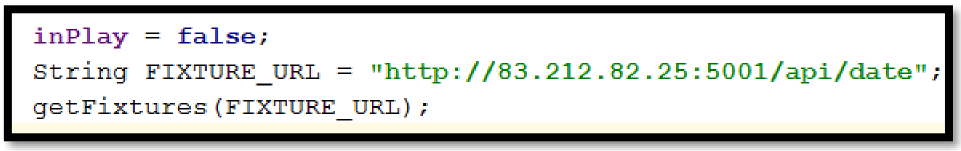


Figure 4.1: Navigation Drawer

The nav_drawer_menu xml file consists of the icon and title of the drawer item. An ID is also assigned to the item. From the MainActivity this ID can be accessed which allows the user to switch between fragments.

4.3.1.3 Match Day

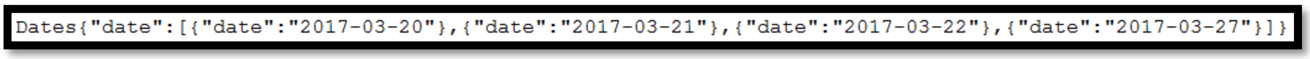
When the manager originally logs in a request is made to the database to return all the dates that have been submitted to the database in the onCreate method. A Boolean inPlay is also declared as false meaning that there is no match currently in progress.



```
inPlay = false;
String FIXTURE_URL = "http://83.212.82.25:5001/api/date";
getFixtures(FIXTURE_URL);
```

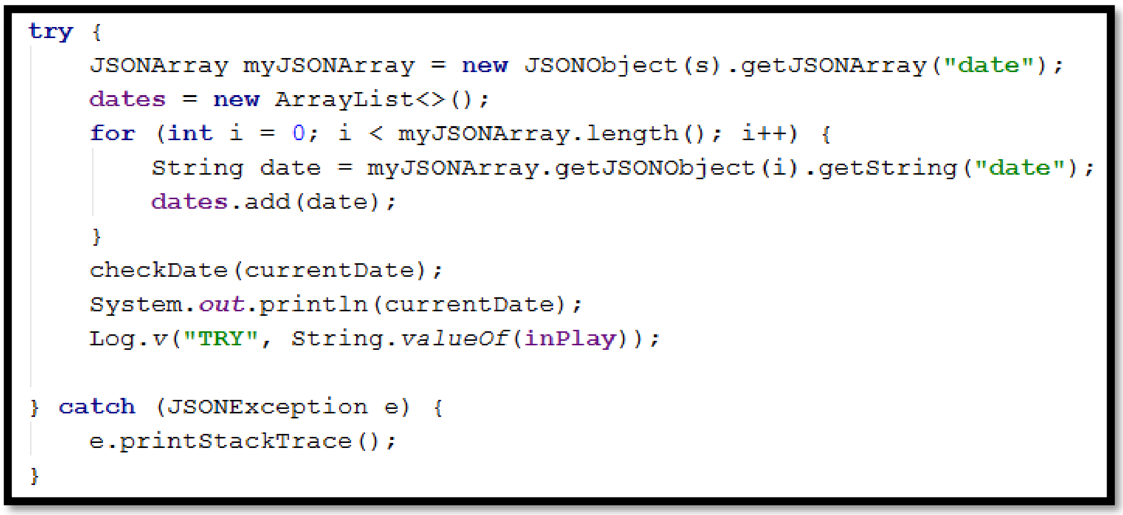
Code snippet 5 – Get Fixtures

Once the request has been made the response is returned in JSON format where the dates are added to an ArrayList dates.



```
Dates{"date":[{"date":"2017-03-20"},{"date":"2017-03-21"},{"date":"2017-03-22"},{"date":"2017-03-27"}]}
```

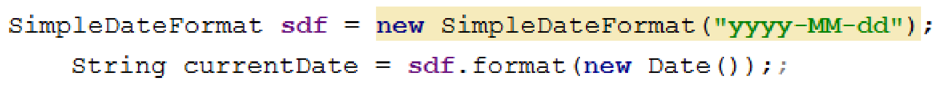
Code snippet 6 – JSON response



```
try {
    JSONArray myJSONArray = new JSONObject(s).getJSONArray("date");
    dates = new ArrayList<>();
    for (int i = 0; i < myJSONArray.length(); i++) {
        String date = myJSONArray.getJSONObject(i).getString("date");
        dates.add(date);
    }
    checkDate(currentDate);
    System.out.println(currentDate);
    Log.v("TRY", String.valueOf(inPlay));

} catch (JSONException e) {
    e.printStackTrace();
}
```

Code snippet 7 – Get Fixtures

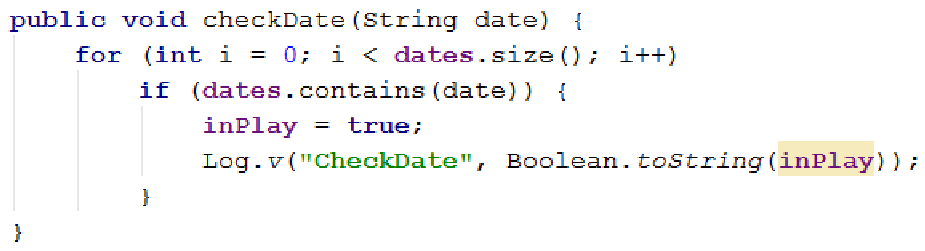
To ensure that the date format in Android studio matches the date format in android studio you must define the structure.



```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    String currentDate = sdf.format(new Date());;
```

Code snippet 8 – Date Format

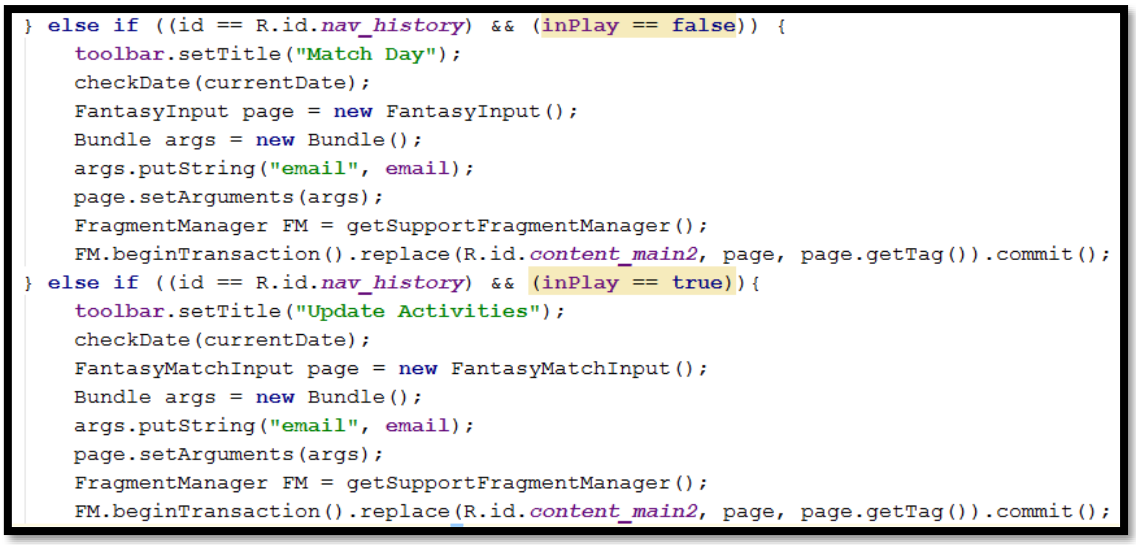
A function called checkDate is called from the navigation drawer which sets the Boolean inPlay to true if there is a date added for the current date.



```
public void checkDate(String date) {
    for (int i = 0; i < dates.size(); i++)
        if (dates.contains(date)) {
            inPlay = true;
            Log.v("CheckDate", Boolean.toString(inPlay));
        }
}
```

Code Snippet 9 – checkDate

As indicated below in code snippet 10, When the manager clicks on the menu item with the ID “nav_history” which is the match day tab it will call the checkDate function which searches the dates ArrayList for the current date. If the Boolean inPlay is set to false it will direct the manager to a fragment where the manager can then select a group of players for the match. If the Boolean inPlay has been set to true it will direct the manager to the activities input fragment where activities can be assigned to players which will be demonstrated in section 4.3.1.4.



```
} else if ((id == R.id.nav_history) && (inPlay == false)) {
    toolbar.setTitle("Match Day");
    checkDate(currentDate);
    FantasyInput page = new FantasyInput();
    Bundle args = new Bundle();
    args.putString("email", email);
    page.setArguments(args);
    FragmentManager FM = getSupportFragmentManager();
    FM.beginTransaction().replace(R.id.content_main2, page, page.getTag()).commit();
} else if ((id == R.id.nav_history) && (inPlay == true)){
    toolbar.setTitle("Update Activities");
    checkDate(currentDate);
    FantasyMatchInput page = new FantasyMatchInput();
    Bundle args = new Bundle();
    args.putString("email", email);
    page.setArguments(args);
    FragmentManager FM = getSupportFragmentManager();
    FM.beginTransaction().replace(R.id.content_main2, page, page.getTag()).commit();
```

Code Snippet 10 – Nav Drawer inPlay

4.3.1.4 Squad Selection

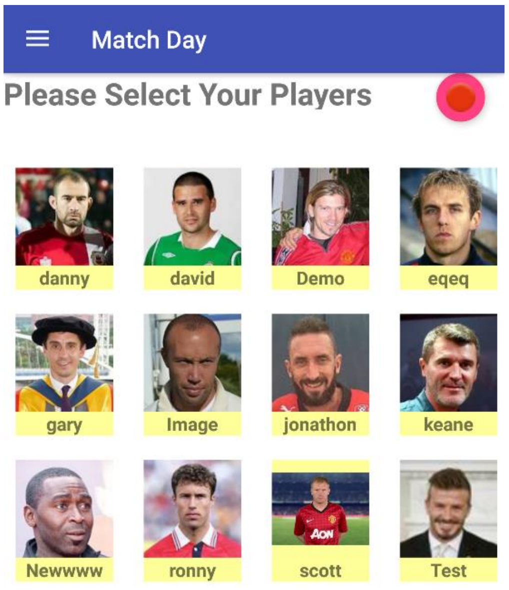
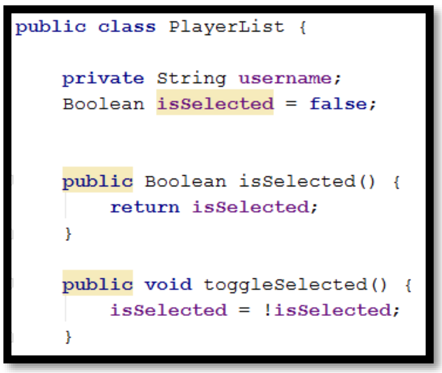


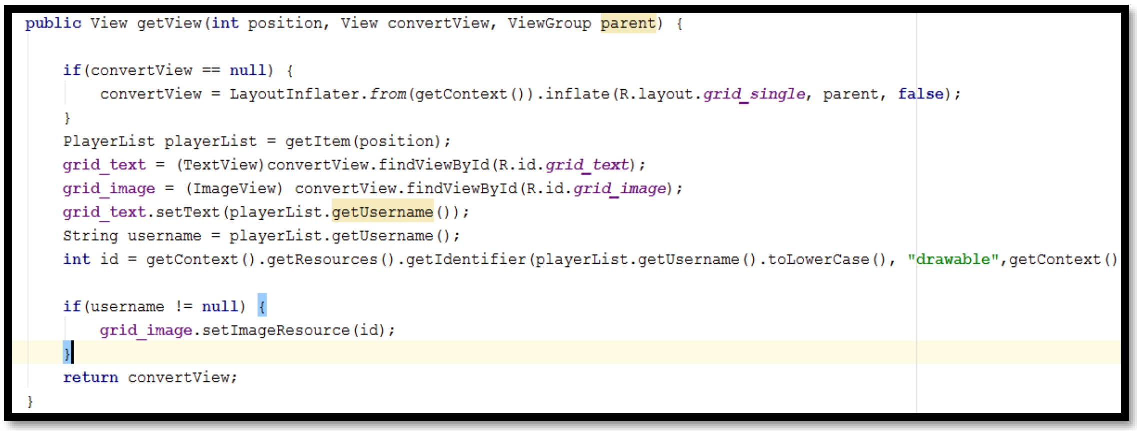
Figure 4.2: Match Day Squad Selection

As discussed previously in section 4.3.1.3, if the Boolean inPlay is false the manager is directed to a screen with a grid view of his players. Each player is originally assigned a Boolean isSelected which is set to false. When the manager clicks on a player in his squad it toggles the Boolean isSelected to true.



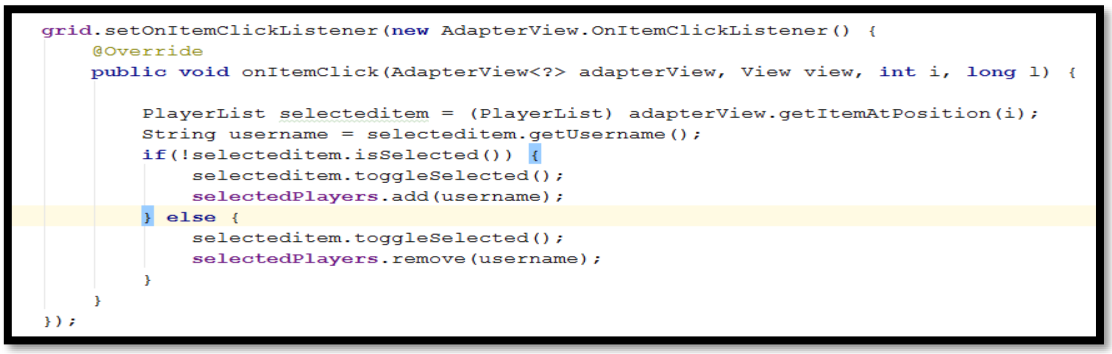
```
public class PlayerList {

    private String username;
    Boolean isSelected = false;


    public Boolean isSelected() {
        return isSelected;
    }

    public void toggleSelected() {
        isSelected = !isSelected;
    }
```

Code snippet 11 – isSelected



```
public View getView(int position, View convertView, ViewGroup parent) {

    if(convertView == null) {
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.grid_single, parent, false);
    }
    PlayerList playerList = getItem(position);
    grid_text = (TextView)convertView.findViewById(R.id.grid_text);
    grid_image = (ImageView) convertView.findViewById(R.id.grid_image);
    grid_text.setText(playerList.getUsername());
    String username = playerList.getUsername();
    int id = getContext().getResources().getIdentifier(playerList.getUsername().toLowerCase(), "drawable",getContext()

    if(username != null) {
        grid_image.setImageResource(id);
    }
    return convertView;
}
```

Code Snippet 12 – Custom Grid Adapter

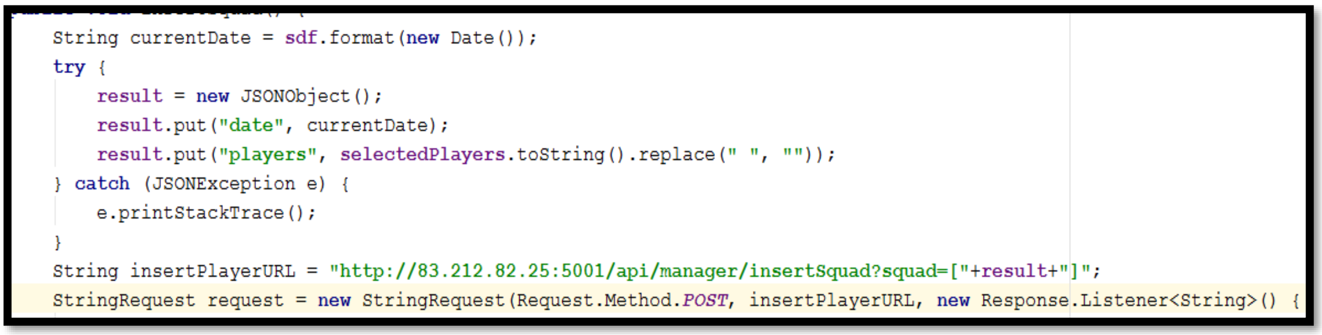


```
grid.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

        PlayerList selecteditem = (PlayerList) adapterView.getItemAtPosition(i);
        String username = selecteditem.getUsername();
        if(!selecteditem.isSelected()) {
            selecteditem.toggleSelected();
            selectedPlayers.add(username);
        } else {
            selecteditem.toggleSelected();
            selectedPlayers.remove(username);
        }
    }
});
```

Code Snippet 13 – Grid onClick

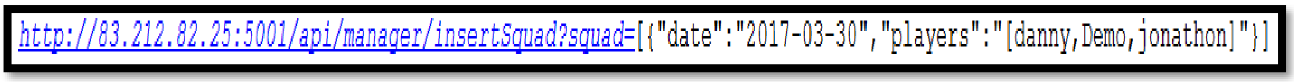
Code snippet 12 indicates how the username for the player can be accessed after the custom grid adapter is applied to the list of players.

Once each grid has been populated with an image and a username the manager can then toggle the selected players by clicking on the image which is displayed in code snippet 13. When the manager has selected his players for the match he can submit his team by clicking the floating action button.



```
    String currentDate = sdf.format(new Date());
    try {
        result = new JSONObject();
        result.put("date", currentDate);
        result.put("players", selectedPlayers.toString().replace(" ", ""));
    } catch (JSONException e) {
        e.printStackTrace();
    }
    String insertPlayerURL = "http://83.212.82.25:5001/api/manager/insertSquad?squad=["+result+"]";
    StringRequest request = new StringRequest(Request.Method.POST, insertPlayerURL, new Response.Listener<String>() {
```

Code snippet 14 – Insert Squad

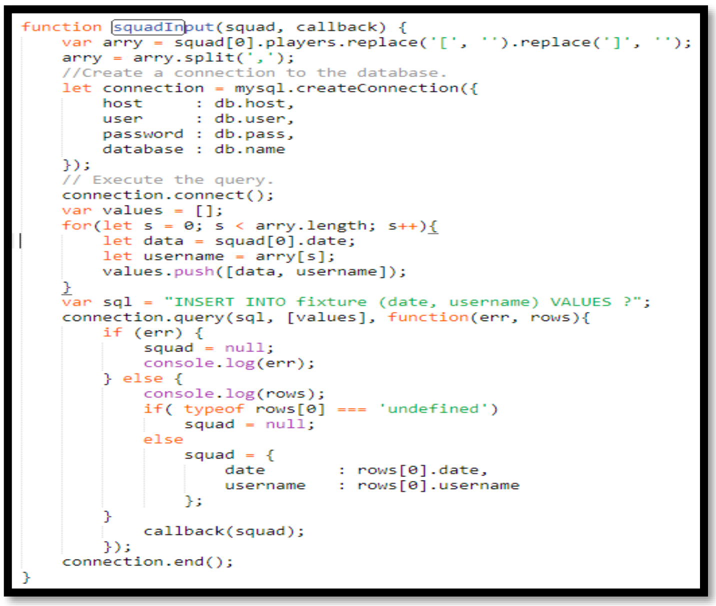


```
http://83.212.82.25:5001/api/manager/insertSquad?squad=[{"date":"2017-03-30","players":"[danny,Demo,jonathon]"}]
```

Code snippet 15 – Example Insert Squad URL

When a request is posted to the server it is processed by the API which will insert the players into the database along with the date.

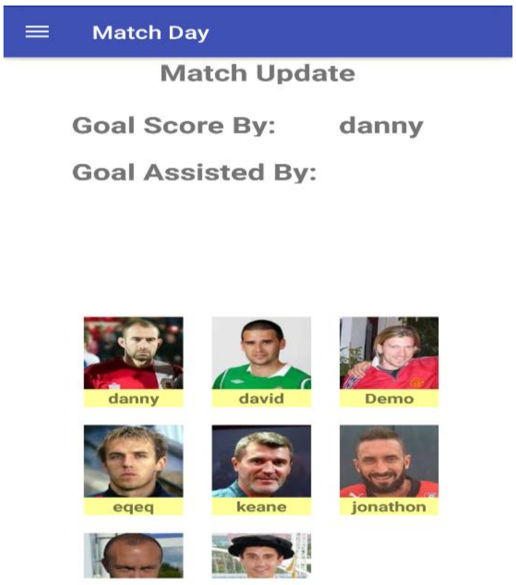
Code snippet 16 indicates how each player is inserted into the database with a date. When the request is processed, an array is created removing the brackets.
A new array of values is created which contain a date and a username. Each username is separated by a "," and from this the array 'arry' can be looped through to extract the usernames which get added to the values array.



```
function squadInput(squad, callback) {
    var arry = squad[0].players.replace('[', '').replace(']', '');
    arry = arry.split(',');
    //Create a connection to the database.
    let connection = mysql.createConnection({
        host     : db.host,
        user     : db.user,
        password : db.pass,
        database : db.name
    });
    // Execute the query.
    connection.connect();
    var values = [];
    for(let s = 0; s < arry.length; s++){
        let data = squad[0].date;
        let username = arry[s];
        values.push([data, username]);
    }
    var sql = "INSERT INTO fixture (date, username) VALUES ?";
    connection.query(sql, [values], function(err, rows){
        if (err) {
            squad = null;
            console.log(err);
        } else {
            console.log(rows);
            if( typeof rows[0] === 'undefined')
                squad = null;
            else
                squad = {
                    date        : rows[0].date,
                    username    : rows[0].username
                };
        }
            callback(squad);
        });
    connection.end();
}
```

Code Snippet 16 – Squad input API

4.3.1.5 Activity Input


Match Day

Match Update

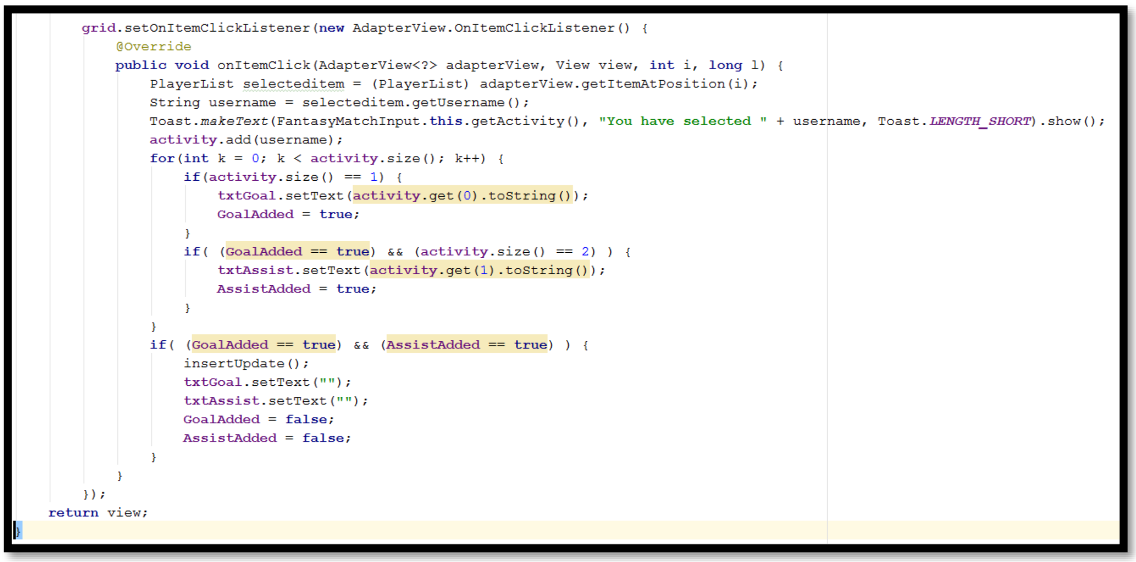
Goal Score By: danny

Goal Assisted By:

Figure 4.3: Match Input

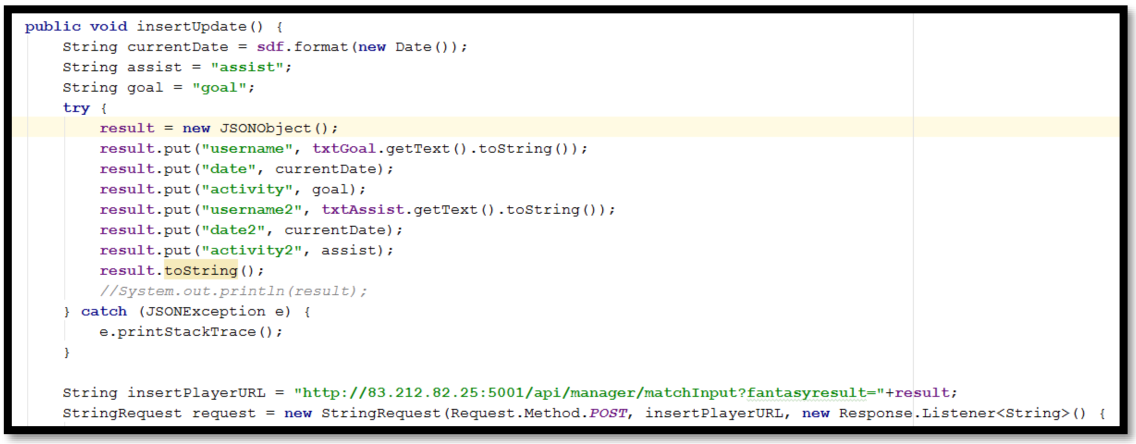
As discussed previously in section 4.3.1.3, if the Boolean inPlay is true and the manager has selected a team for the current date a list of the selected players will appear.

The first player the manager clicks on will assign that player a ‘goal’. The second player the manager clicks on will assign that player an ‘assist’.


```
        grid.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
                PlayerList selecteditem = (PlayerList) adapterView.getItemAtPosition(i);
                String username = selecteditem.getUsername();
                Toast.makeText(FantasyMatchInput.this.getActivity(), "You have selected " + username, Toast.LENGTH_SHORT).show();
                activity.add(username);
                for(int k = 0; k < activity.size(); k++) {
                    if(activity.size() == 1) {
                        txtGoal.setText(activity.get(0).toString());
                        GoalAdded = true;
                    }
                    if( (GoalAdded == true) && (activity.size() == 2) ) {
                        txtAssist.setText(activity.get(1).toString());
                        AssistAdded = true;
                    }
                }
                if( (GoalAdded == true) && (AssistAdded == true) ) {
                    insertUpdate();
                    txtGoal.setText("");
                    txtAssist.setText("");
                    GoalAdded = false;
                    AssistAdded = false;
                }
            }
        });
    return view;
}
```

Code Snippet 17 – Insert Activity

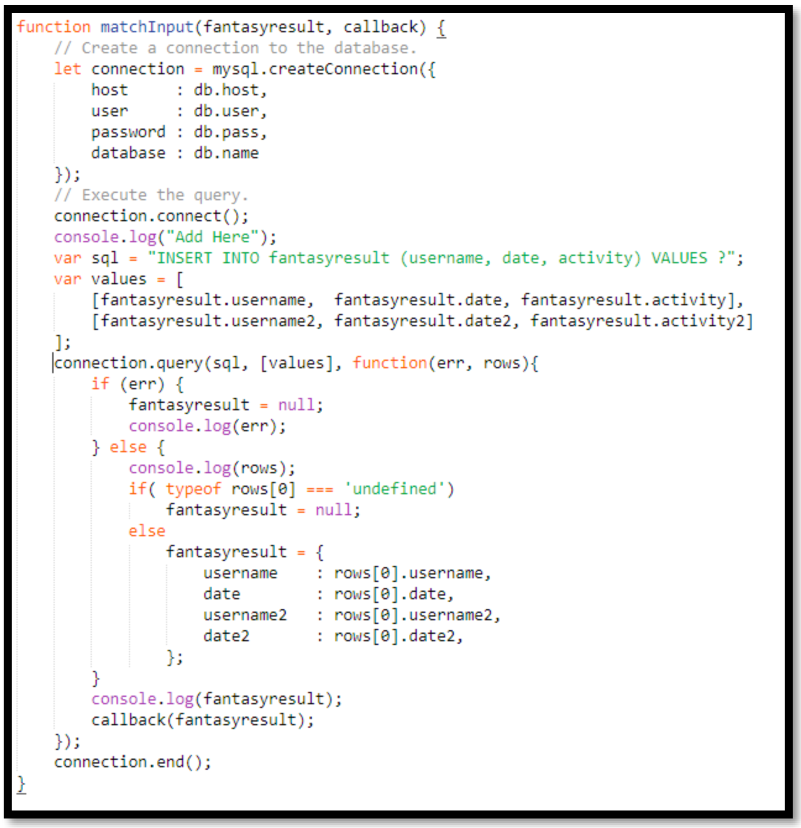
As shown in code snippet 17 there are two Booleans, goalAdded and assistAdded initialized to false in the onCreate section of the fragment. An ArrayList called activity which adds the players' names each time they are selected.



```
public void insertUpdate() {
    String currentDate = sdf.format(new Date());
    String assist = "assist";
    String goal = "goal";
    try {
        result = new JSONObject();
        result.put("username", txtGoal.getText().toString());
        result.put("date", currentDate);
        result.put("activity", goal);
        result.put("username2", txtAssist.getText().toString());
        result.put("date2", currentDate);
        result.put("activity2", assist);
        result.toString();
        //System.out.println(result);
    } catch (JSONException e) {
        e.printStackTrace();
    }

    String insertPlayerURL = "http://83.212.82.25:5001/api/manager/matchInput?fantasyresult="+result;
    StringRequest request = new StringRequest(Request.Method.POST, insertPlayerURL, new Response.Listener<String>() {
```

Code snippet 18 – InsertUpdate

When goalAdded and assistAdded are both true insertUpdate is called, which posts a request to the API which processes the request and inserts the data into the database.



```
function matchInput(fantasyresult, callback) {
    // Create a connection to the database.
    let connection = mysql.createConnection({
        host     : db.host,
        user     : db.user,
        password : db.pass,
        database : db.name
    });
    // Execute the query.
    connection.connect();
    console.log("Add Here");
    var sql = "INSERT INTO fantasyresult (username, date, activity) VALUES ?";
    var values = [
        [fantasyresult.username,  fantasyresult.date, fantasyresult.activity],
        [fantasyresult.username2, fantasyresult.date2, fantasyresult.activity2]
    ];
    connection.query(sql, [values], function(err, rows){
        if (err) {
            fantasyresult = null;
            console.log(err);
        } else {
            console.log(rows);
            if( typeof rows[0] === 'undefined')
                fantasyresult = null;
            else
                fantasyresult = {
                    username    : rows[0].username,
                    date        : rows[0].date,
                    username2   : rows[0].username2,
                    date2       : rows[0].date2,
                };
        }
        console.log(fantasyresult);
        callback(fantasyresult);
    });
    connection.end();
}
```

` Code Snippet 19 – matchInput API

When the request arrives at the API and a connection is established the API will process two queries. The first query will insert the goal scorer along with the date and the activity. The second query will insert the player who got the assist along with the date and the activity.

4.3.1.6 Individual Statistics

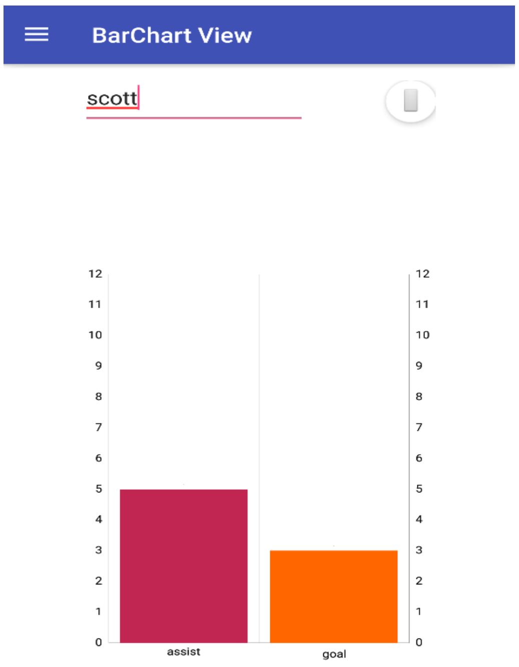
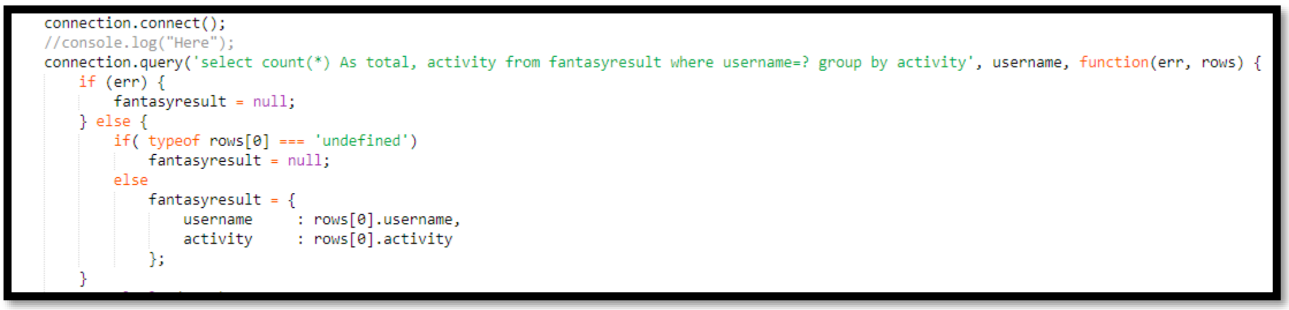


Figure 4.4: Statistical Overview

The Statistical Overview fragment allows the manager to enter one of his players' names which returns a BarChart with the total goals and assists that player has registered.

When a name is entered the AutoCompleteTextView will suggest player's names based on the first two letters. Once a name has been entered and the action button is pressed the text from the AutoCompleteTextView is assigned to a String and a request is posted to the server.



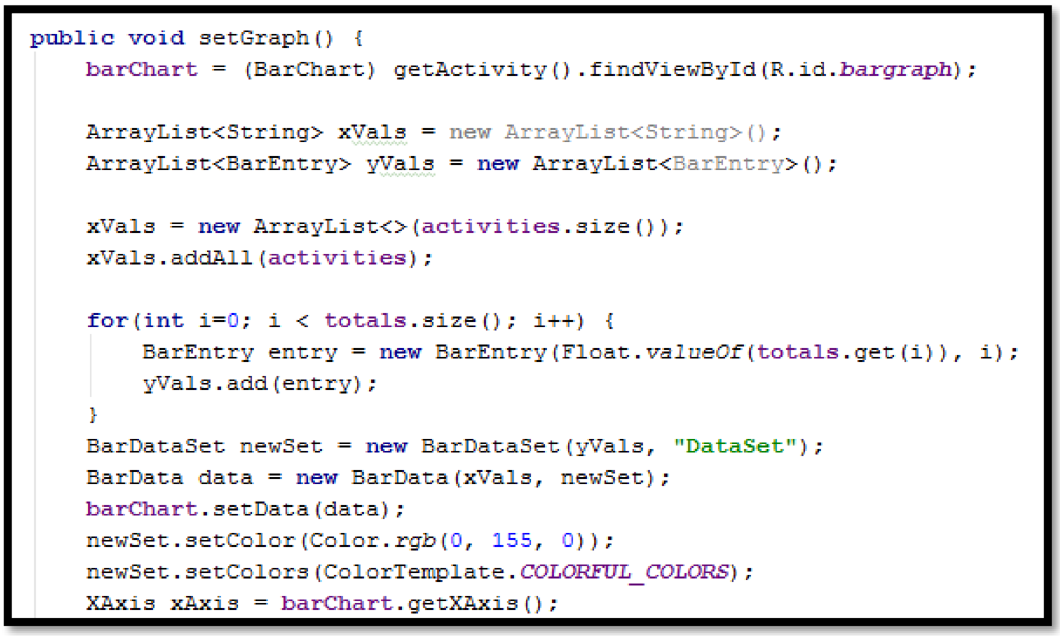
```
connection.connect();
//console.log("Here");
connection.query('select count(*) As total, activity from fantasyresult where username=? group by activity', username, function(err, rows) {
    if (err) {
        fantasyresult = null;
    } else {
        if( typeof rows[0] === 'undefined')
            fantasyresult = null;
        else
            fantasyresult = {
                username      : rows[0].username,
                activity      : rows[0].activity
            };
    }
```

Code Snippet 20 – Count activities

The API returns the result to Android Studio in JSON where it is processed. The activities, Assists, and goals are added to an ArrayList names activities and the totals are added to a new ArrayList totals.

```
Fantasy Result[{"total":5,"activity":"assist"},{"total":3,"activity":"goal"}]
```

Code Snippet 21 - JSON result1



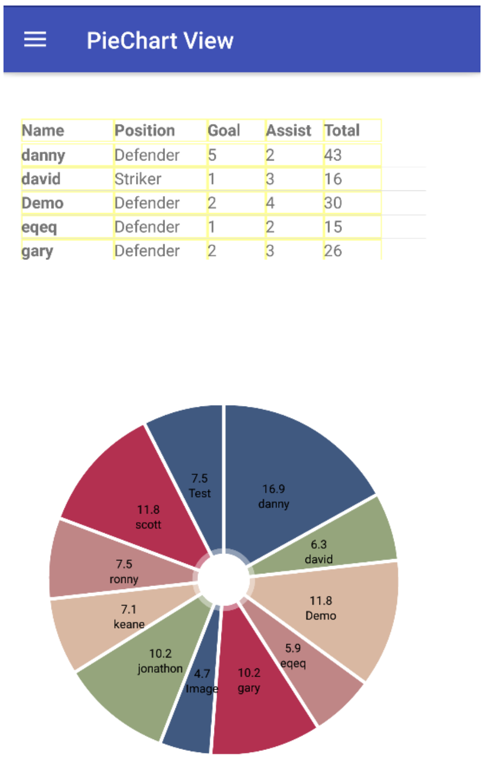
```
public void setGraph() {
    barChart = (BarChart) getActivity().findViewById(R.id.bargraph);

    ArrayList<String> xVals = new ArrayList<String>();
    ArrayList<BarEntry> yVals = new ArrayList<BarEntry>();

    xVals = new ArrayList<>(activities.size());
    xVals.addAll(activities);

    for(int i=0; i < totals.size(); i++) {
        BarEntry entry = new BarEntry(Float.valueOf(totals.get(i)), i);
        yVals.add(entry);
    }
    BarDataSet newSet = new BarDataSet(yVals, "DataSet");
    BarData data = new BarData(xVals, newSet);
    barChart.setData(data);
    newSet.setColor(Color.rgb(0, 155, 0));
    newSet.setColors(ColorTemplate.COLORFUL_COLORS);
    XAxis xAxis = barChart.getXAxis();
```

Code Snippet 22 – Set BarChart

The BarChart is imported from MPAndroidCharts. The X values which appear at the bottom of the chart are introduced from the activities ArrayList created in the previous method. These activities appear at the bottom of the chart.

Each entry into the Y values is taken from the totals ArrayList previously created.
Once the chart has the BarData which is the data taken from the totals ArrayList and the X values which are the activities the data can be combined which are then set to the barChart.

4.3.1.7 Statistical Overview



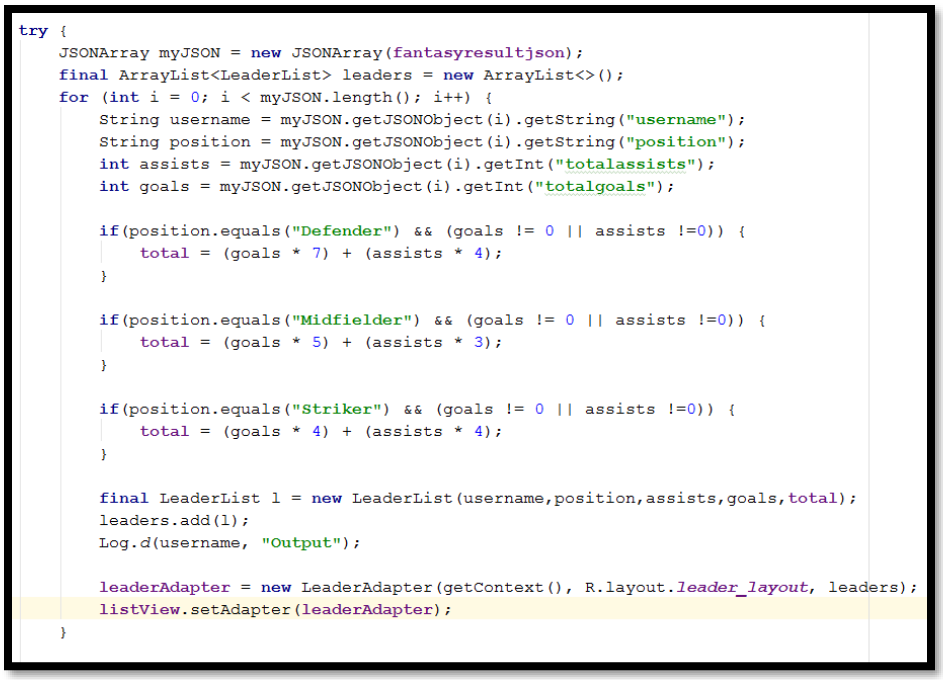
Name	Position	Goal	Assist	Total
danny	Defender	5	2	43
david	Striker	1	3	16
Demo	Defender	2	4	30
eqeq	Defender	1	2	15
gary	Defender	2	3	26

Figure 4.5: Overall Totals

The statistical overview fragment returns the overall activities for each player in the manager's squad. At the top of the screen is a ListView with a table row placed directly above it. The rows are assigned to the player's name, position, total goals, total assists and their total score.

When the fragment has first created the manager's squad is loaded, the player's respective positions earn them different points for different activities. Code snippet 23 indicates how the totals are calculated for each position. For example, if a player is a defender they will earn 7 points for a goal and 4 for an assist.

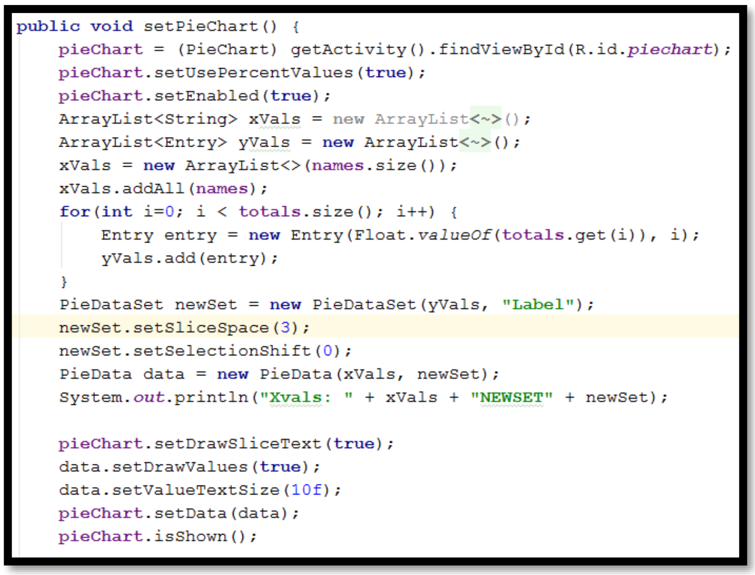
To calculate how many goals and assists each player has when a request is made to the database the sum function is used. For each time a goal appears with the player's name 1 is assigned to the activity and for each time assist appears next to the player's name 1 is also assigned.



```java
try {
    JSONArray myJSON = new JSONArray(fantasyresultjson);
    final ArrayList<LeaderList> leaders = new ArrayList<>();
    for (int i = 0; i < myJSON.length(); i++) {
        String username = myJSON.getJSONObject(i).getString("username");
        String position = myJSON.getJSONObject(i).getString("position");
        int assists = myJSON.getJSONObject(i).getInt("totalassists");
        int goals = myJSON.getJSONObject(i).getInt("totalgoals");

        if(position.equals("Defender") && (goals != 0 || assists !=0)) {
            total = (goals * 7) + (assists * 4);
        }

        if(position.equals("Midfielder") && (goals != 0 || assists !=0)) {
            total = (goals * 5) + (assists * 3);
        }

        if(position.equals("Striker") && (goals != 0 || assists !=0)) {
            total = (goals * 4) + (assists * 4);
        }

        final LeaderList l = new LeaderList(username,position,assists,goals,total);
        leaders.add(l);
        Log.d(username, "Output");

        leaderAdapter = new LeaderAdapter(getContext(), R.layout.leader_layout, leaders);
        listView.setAdapter(leaderAdapter);
    }
```

Code Snippet 23 – Calculate Totals

Once the JSON has been assigned to different Strings, the names and the totals are added to a names ArrayList and a totals ArrayList.

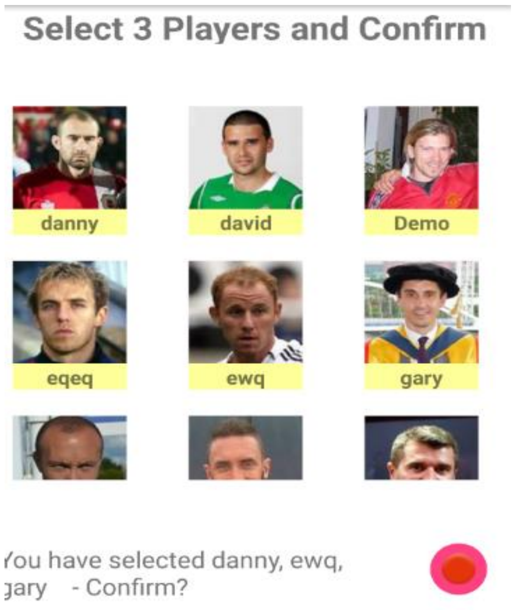


```java
public void setPieChart() {
    pieChart = (PieChart) getActivity().findViewById(R.id.piechart);
    pieChart.setUsePercentValues(true);
    pieChart.setEnabled(true);
    ArrayList<String> xVals = new ArrayList<~>();
    ArrayList<Entry> yVals = new ArrayList<~>();
    xVals = new ArrayList<>(names.size());
    xVals.addAll(names);
    for(int i=0; i < totals.size(); i++) {
        Entry entry = new Entry(Float.valueOf(totals.get(i)), i);
        yVals.add(entry);
    }
    PieDataSet newSet = new PieDataSet(yVals, "Label");
    newSet.setSliceSpace(3);
    newSet.setSelectionShift(0);
    PieData data = new PieData(xVals, newSet);
    System.out.println("Xvals: " + xVals + "NEWSET" + newSet);

    pieChart.setDrawSliceText(true);
    data.setDrawValues(true);
    data.setValueTextSize(10f);
    pieChart.setData(data);
    pieChart.isShown();
```

Code Snippet 24 – Set PieChart

Once the players' names and their totals have been added to their respective ArrayLists the X values are assigned to the names and the Y values are assigned to the totals which are displayed to the user through a PieChart.

4.3.1.8 Player Fantasy League

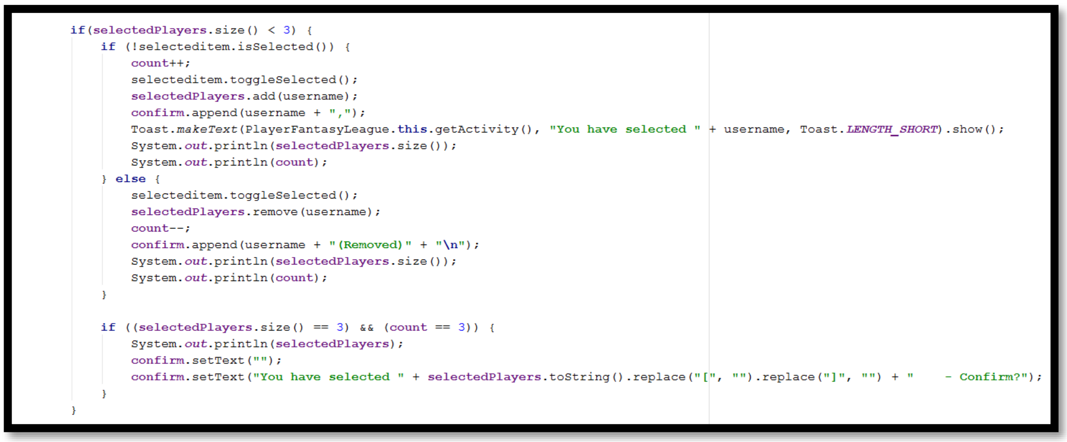

Select 3 Players and Confirm

You have selected danny, ewq, gary - Confirm?

Figure 4.6: Select players

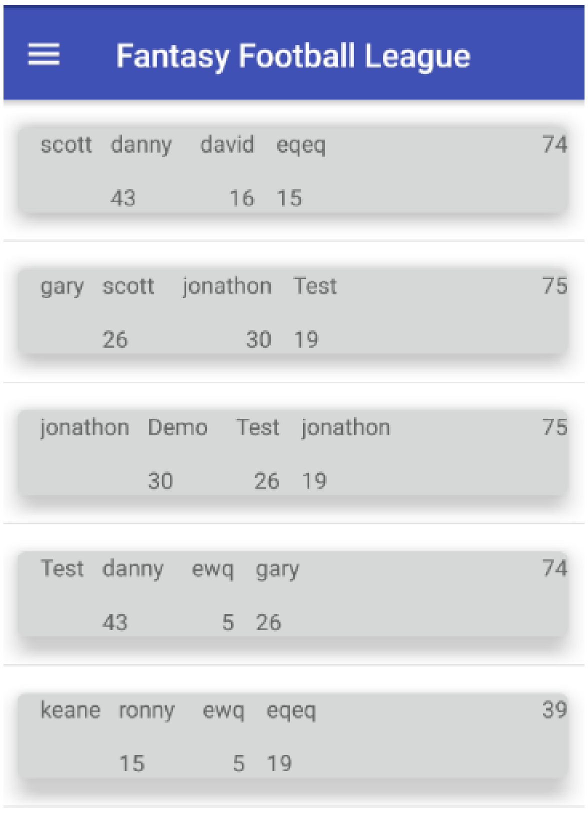
The fantasy football feature allows a player to select three of their teammates where they are assigned as the manager of those three players. Each player on the team accumulates points during matches that are added to the managers overall total. These totals are displayed in a list where the players can compare totals with one another.

Code snippet 25 indicates how the players are selected by the user.
An ArrayList called selectedPlayers is initialized when the fragment is first created. For each player, the manager selects, that player is added to the ArrayList. If the manager clicks on the player again that player is removed from the ArrayList. When the size of the selectedPlayers ArrayList reaches 3 the manager is asked if they wish to confirm their squad.


```java
if(selectedPlayers.size() < 3) {
    if (!selecteditem.isSelected()) {
        count++;
        selecteditem.toggleSelected();
        selectedPlayers.add(username);
        confirm.append(username + ",");
        Toast.makeText(PlayerFantasyLeague.this.getActivity(), "You have selected " + username, Toast.LENGTH_SHORT).show();
        System.out.println(selectedPlayers.size());
        System.out.println(count);
    } else {
        selecteditem.toggleSelected();
        selectedPlayers.remove(username);
        count--;
        confirm.append(username + "(Removed)" + "\n");
        System.out.println(selectedPlayers.size());
        System.out.println(count);
    }

    if ((selectedPlayers.size() == 3) && (count == 3)) {
        System.out.println(selectedPlayers);
        confirm.setText("");
        confirm.setText("You have selected " + selectedPlayers.toString().replace("[", "").replace("]", "") + "    - Confirm?");
    }
}
```

Code Snippet 25

4.3.1.9 Fantasy Player League Standings

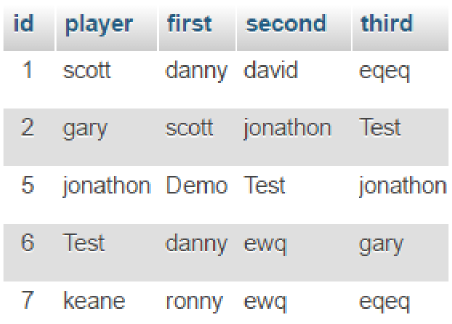


scott	danny	david	eqeq	74
	43	16	15	
gary	scott	jonathon	Test	75
	26	30	19	
jonathon	Demo	Test	jonathon	75
	30	26	19	
Test	danny	ewq	gary	74
	43	5	26	
keane	ronny	ewq	eqeq	39
	15	5	19	

Figure 4.7: League standings

If a player has submitted a team into the fantasy football league their team will appear in a grid with his three players, their totals and the manager of the players combined total.
For example in the first grid, the manager of the team is Scott. His three players, Danny, David and Eqeq all have individual totals which are combined for Scott's total.

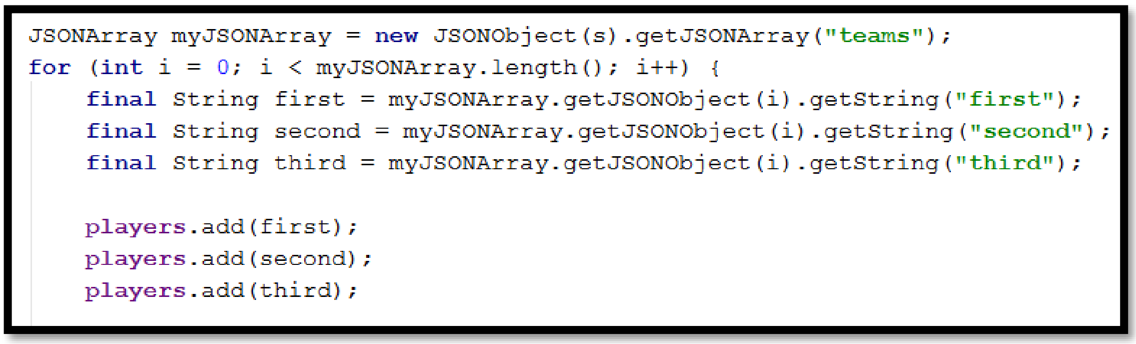
Due to the player on the team acting as the manager, this led to issues where the manager of the teams total was being incorporated into the end total as he is also a player. To rectify this issue when the fragment initially loads, all the players and managers from the league are loaded.



id	player	first	second	third
1	scott	danny	david	eqeq
2	gary	scott	jonathon	Test
5	jonathon	Demo	Test	jonathon
6	Test	danny	ewq	gary
7	keane	ronny	ewq	eqeq

Figure 4.8: Pleague Database Table

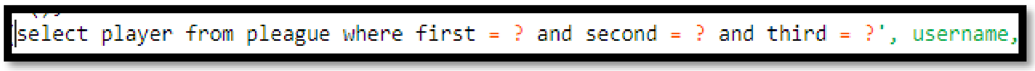
When the teams have been retrieved from JSON they are then added to an ArrayList where the players are related to the fields first, second and third.



```
JSONArray myJSONArray = new JSONObject(s).getJSONArray("teams");
for (int i = 0; i < myJSONArray.length(); i++) {
    final String first = myJSONArray.getJSONObject(i).getString("first");
    final String second = myJSONArray.getJSONObject(i).getString("second");
    final String third = myJSONArray.getJSONObject(i).getString("third");

    players.add(first);
    players.add(second);
    players.add(third);
```

Code Snippet 26 – Retrieve Players

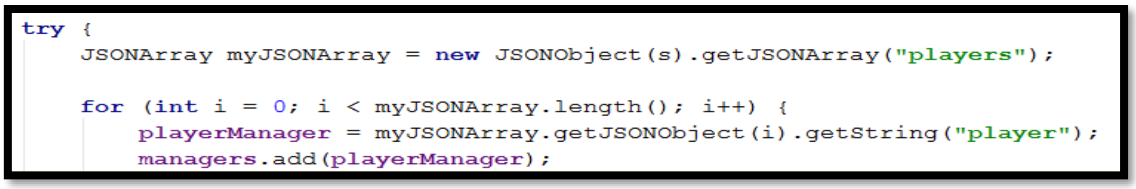
To retrieve the manager the query in code snippet 27 was used. This query is performed each time there is a row of three players.



```
select player from pleague where first = ? and second = ? and third = ?', username,
```

Code Snippet 27 – Get team manager

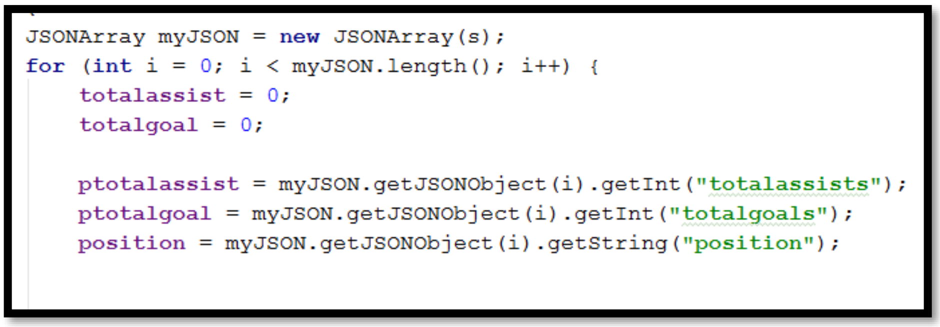
When the managers have been loaded, they are added to a manager's ArrayList as shown below.



```
try {
    JSONArray myJSONArray = new JSONObject(s).getJSONArray("players");

    for (int i = 0; i < myJSONArray.length(); i++) {
        playerManager = myJSONArray.getJSONObject(i).getString("player");
        managers.add(playerManager);
```

Code Snippet 28 – Retrieve managers

Once the players loading and the managers loading have completed, there are two Booleans that set to true. Once both loads have completed a request is sent for every set of three players in the players ArrayList. Once the response has arrived the player's position is used to calculate the total points for the specific player.



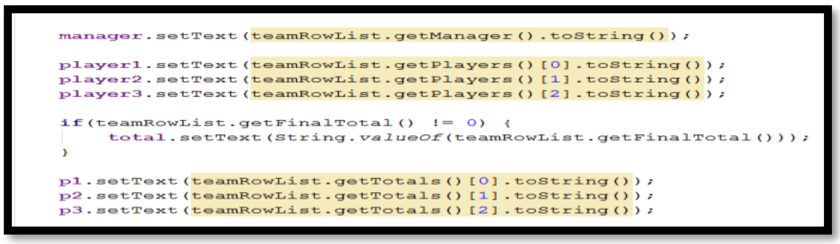
```
JSONArray myJSON = new JSONArray(s);
for (int i = 0; i < myJSON.length(); i++) {
    totalassist = 0;
    totalgoal = 0;

    ptotalassist = myJSON.getJSONObject(i).getInt("totalassists");
    ptotalgoal = myJSON.getJSONObject(i).getInt("totalgoals");
    position = myJSON.getJSONObject(i).getString("position");
```

Code Snippet 29 – Retrieve totals

To conclude, When the fragment initially loads a request is made to get all the players on league teams. Once the players ArrayList has been loaded a request is made to retrieve the manager associated with the three players and the managers are added to a managers ArrayList. Once these two requests have completed a request is made to retrieve the totals and position where the names = the first three names in the players ArrayList.

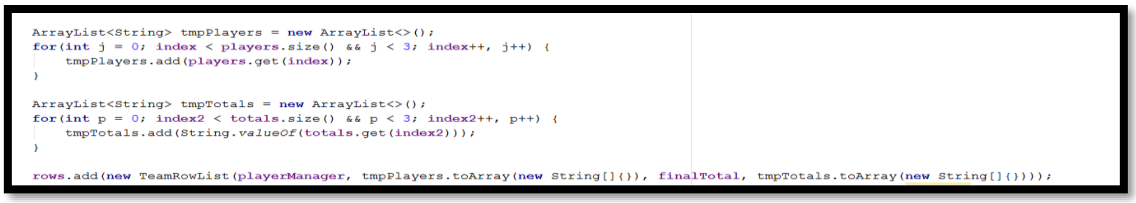
In a separate class named TeamRowList there is a constructor composed of a manager, an array of players, a total and an array of totals as can be seen on the final line in code snippet 30.
In an adapter class TeamRowAdapter, the values are retrieved and assigned to textFields in the grid.

Code snippet 31 indicates how each of the totals ArrayList and the players ArrayList and looped through every three values which are added as a row in the rows ArrayList.



```
manager.setText(teamRowList.getManager().toString());

player1.setText(teamRowList.getPlayers()[0].toString());
player2.setText(teamRowList.getPlayers()[1].toString());
player3.setText(teamRowList.getPlayers()[2].toString());

if(teamRowList.getFinalTotal() != 0) {
    total.setText(String.valueOf(teamRowList.getFinalTotal()));
}

p1.setText(teamRowList.getTotals()[0].toString());
p2.setText(teamRowList.getTotals()[1].toString());
p3.setText(teamRowList.getTotals()[2].toString());
```

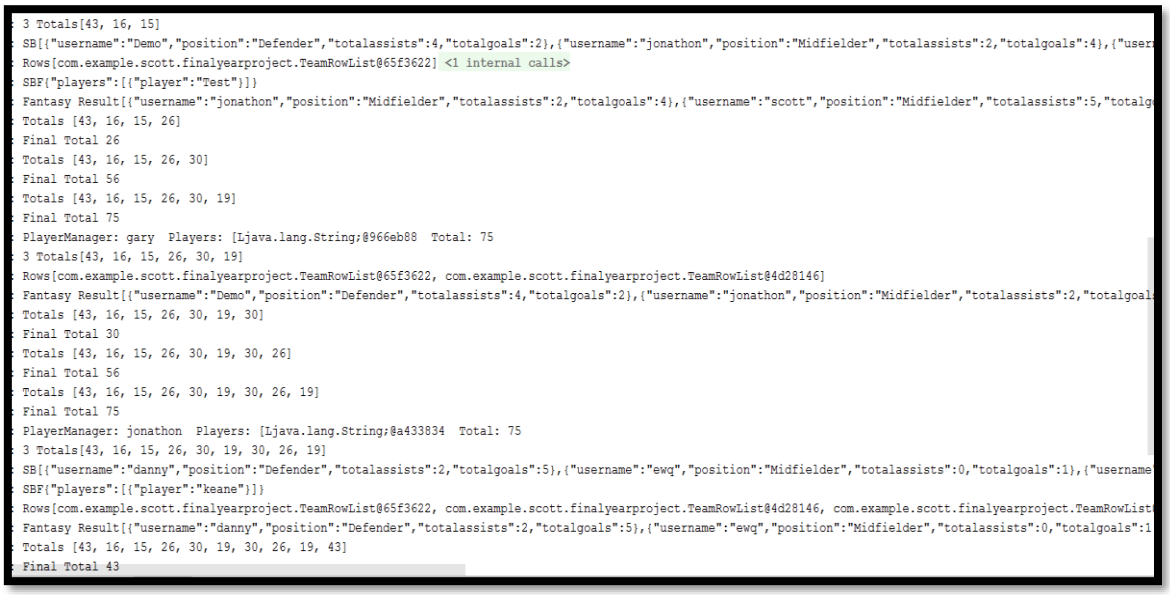
Code Snippet 30



```
ArrayList<String> tmpPlayers = new ArrayList<>();
for(int j = 0; index < players.size() && j < 3; index++, j++) {
    tmpPlayers.add(players.get(index));
}

ArrayList<String> tmpTotals = new ArrayList<>();
for(int p = 0; index2 < totals.size() && p < 3; index2++, p++) {
    tmpTotals.add(String.valueOf(totals.get(index2)));
}

rows.add(new TeamRowList(playerManager, tmpPlayers.toArray(new String[]{}), finalTotal, tmpTotals.toArray(new String[]{})));
```

Code Snippet 31 – TeamRow

Figure 4.9, displays the process the system is undergoing. Once the totals have been calculated each of the values are assigned to the row.



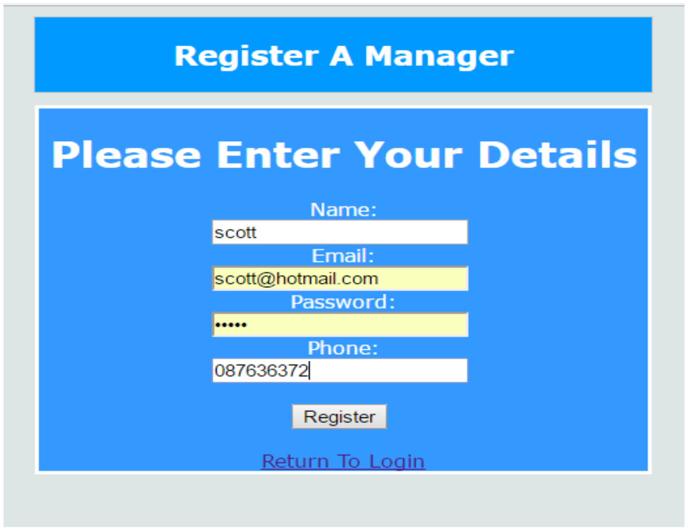
```
3 Totals[43, 16, 15]
SB[{"username":"Demo","position":"Defender","totalassists":4,"totalgoals":2},{"username":"jonathon","position":"Midfielder","totalassists":2,"totalgoals":4},{"user
Rows[com.example.scott.finalyearproject.TeamRowList@65f3622] <1 internal calls>
SBF{"players":[{"player":"Test"}]}
Fantasy Result[{"username":"jonathon","position":"Midfielder","totalassists":2,"totalgoals":4},{"username":"scott","position":"Midfielder","totalassists":5,"totalg
Totals [43, 16, 15, 26]
Final Total 26
Totals [43, 16, 15, 26, 30]
Final Total 56
Totals [43, 16, 15, 26, 30, 19]
Final Total 75
PlayerManager: gary  Players: [Ljava.lang.String;@966eb88  Total: 75
3 Totals[43, 16, 15, 26, 30, 19]
Rows[com.example.scott.finalyearproject.TeamRowList@65f3622, com.example.scott.finalyearproject.TeamRowList@4d28146]
Fantasy Result[{"username":"Demo","position":"Defender","totalassists":4,"totalgoals":2},{"username":"jonathon","position":"Midfielder","totalassists":2,"totalgoal
Totals [43, 16, 15, 26, 30, 19, 30]
Final Total 30
Totals [43, 16, 15, 26, 30, 19, 30, 26]
Final Total 56
Totals [43, 16, 15, 26, 30, 19, 30, 26, 19]
Final Total 75
PlayerManager: jonathon  Players: [Ljava.lang.String;@a433834  Total: 75
3 Totals[43, 16, 15, 26, 30, 19, 30, 26, 19]
SB[{"username":"danny","position":"Defender","totalassists":2,"totalgoals":5},{"username":"ewq","position":"Midfielder","totalassists":0,"totalgoals":1},{"username
SBF{"players":[{"player":"keane"}]}
Rows[com.example.scott.finalyearproject.TeamRowList@65f3622, com.example.scott.finalyearproject.TeamRowList@4d28146, com.example.scott.finalyearproject.TeamRowList
Fantasy Result[{"username":"danny","position":"Defender","totalassists":2,"totalgoals":5},{"username":"ewq","position":"Midfielder","totalassists":0,"totalgoals":1
Totals [43, 16, 15, 26, 30, 19, 30, 26, 19, 43]
Final Total 43
```

Figure 4.9: Console view

4.3.2 Web Application

4.3.2.1 Registration

A user will register their account through the URL http://83.212.82.25/registration.html. The user is asked for the name, email, password and phone which are posted to the database upon submission. The email address field is unique and will not allow multiple users enter the same email address.



Register A Manager

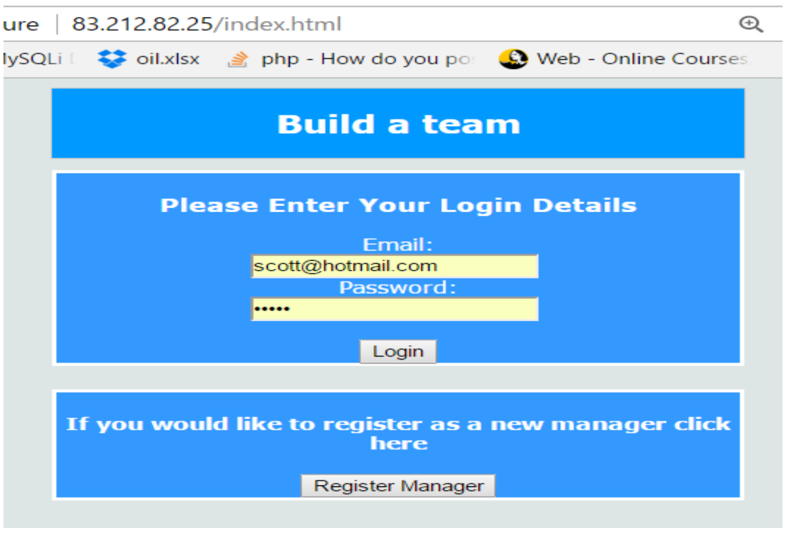
Please Enter Your Details

Name: scott
Email: scott@hotmail.com
Password: •••••
Phone: 087636372

Figure 4.10: Register

4.3.2.2 Login

Once a user has registered they can then access their account by entering their email and password.


ure | 83.212.82.25/index.html

lySQLi | oil.xlsx | php - How do you po | Web - Online Courses

Build a team

Please Enter Your Login Details

Email: scott@hotmail.com

Password: •••••

Login

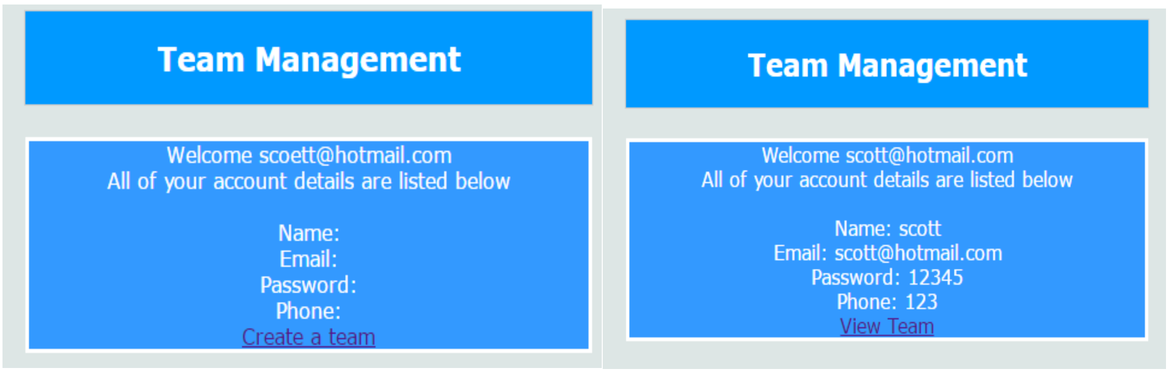
If you would like to register as a new manager click here

Register Manager

Figure 4.11: Index

4.3.2.3 Welcome

When a manager has logged in if they haven't already created a team they are prompted to do so. If the manager has created a team with a group of players the option to view team will appear. Code snippet 41 indicates how, from the welcome page, a request is made to the ManagerService which queries the database using the manager's email as to whether they have a team.


Team Management

Welcome scoett@hotmail.com
All of your account details are listed below

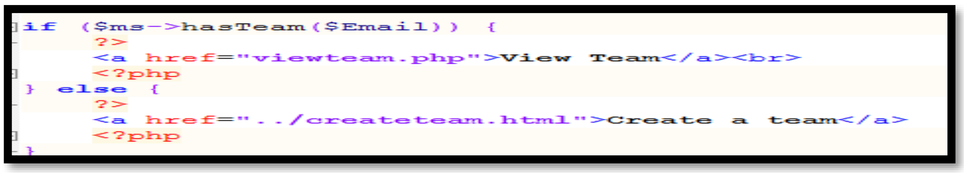
Name:
Email:
Password:
Phone:
Create a team

Team Management

Welcome scott@hotmail.com
All of your account details are listed below

Name: scott
Email: scott@hotmail.com
Password: 12345
Phone: 123
View Team

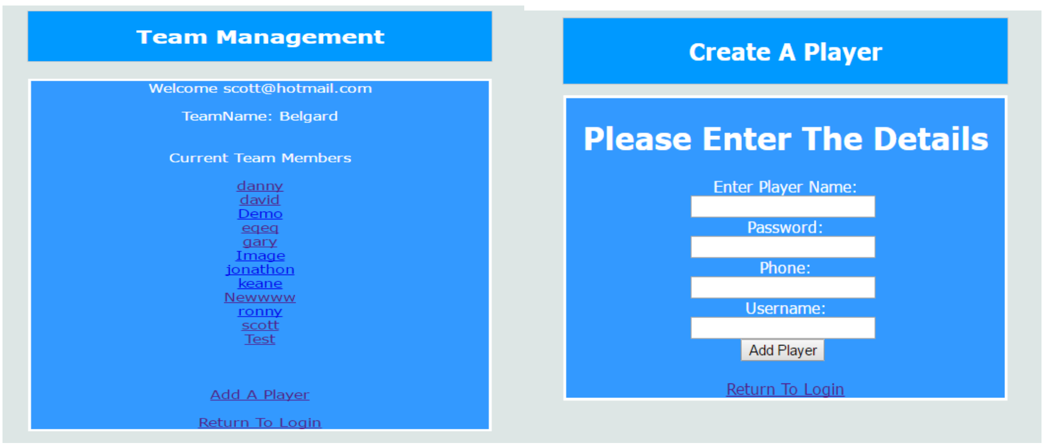
Figure 4.12: Welcome



```
if  ($ms->hasTeam($Email))  {
    ?>
    <a href="viewteam.php">View Team</a><br>
    <?php
} else  {
    ?>
    <a href="../createteam.html">Create a team</a>
    <?php
}
```

Code Snippet 32 – Check for team

4.3.2.4 ViewTeam & AddPlayer



Team Management

Welcome scott@hotmail.com

TeamName: Belgard

Current Team Members

danny
david
Demo
eqeq
gary
Image
jonathon
keane
Newwww
ronny
scott
Test

Add A Player

Return To Login

Create A Player

Please Enter The Details

Enter Player Name:

Password:

Phone:

Username:

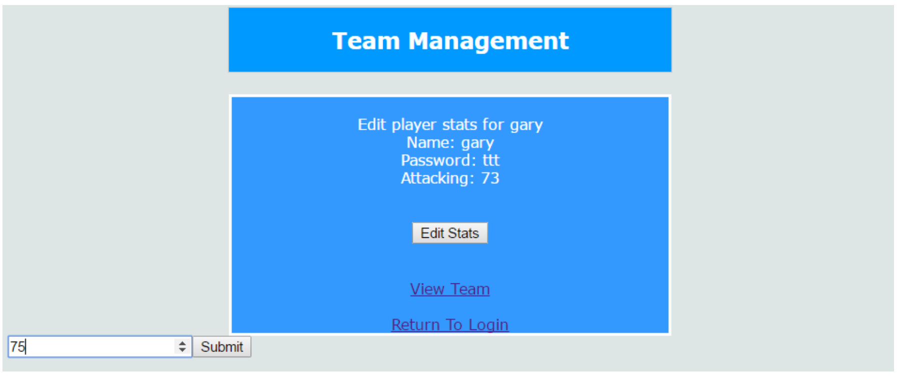
Add Player

Return To Login

Figure 4.13: View Team & Add Player

The view team feature will display a list of the manager’s players with the option to add a player to his squad

4.3.2.5 View Player



Team Management

Edit player stats for gary
Name: gary
Password: ttt
Attacking: 73

Edit Stats

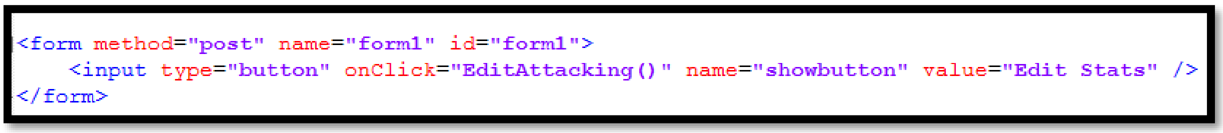
View Team

Return To Login

75 Submit

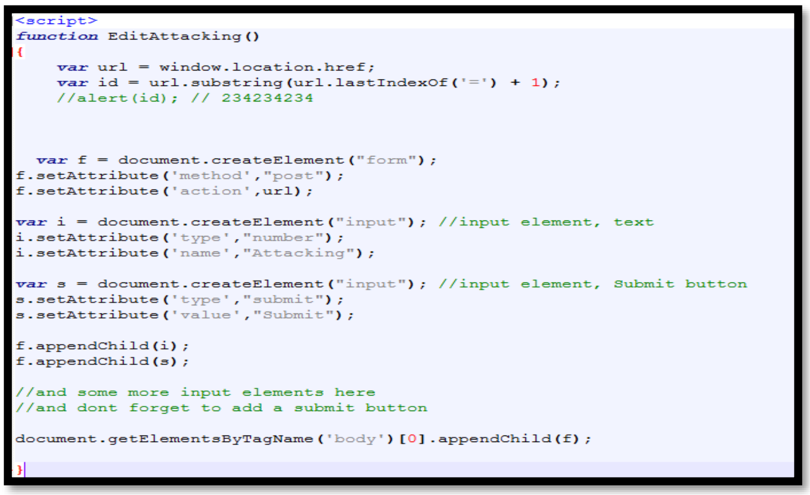
Figure 4.14: ViewPlayer

When a manager clicks on a player, the view player screen is displayed. The manager from here can alter the Attacking rating for that specific player. When the edit stats button has pressed the textbox, and submit button will appear.



```
<form method="post" name="form1" id="form1">
    <input type="button" onClick="EditAttacking()" name="showbutton" value="Edit Stats" />
</form>
```

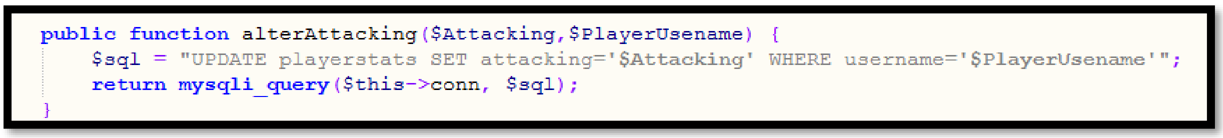
Code Snippet 33



```
<script>
function EditAttacking()
{
    var url = window.location.href;
    var id = url.substring(url.lastIndexOf('=') + 1);
    //alert(id); // 234234234


  var f = document.createElement("form");
f.setAttribute('method',"post");
f.setAttribute('action',url);

var i = document.createElement("input"); //input element, text
i.setAttribute('type',"number");
i.setAttribute('name',"Attacking");

var s = document.createElement("input"); //input element, Submit button
s.setAttribute('type',"submit");
s.setAttribute('value',"Submit");

f.appendChild(i);
f.appendChild(s);

//and some more input elements here
//and dont forget to add a submit button

document.getElementsByTagName('body')[0].appendChild(f);

}
```

Code Snippet 34

When the manager enters a new rating the previous rating is updated in the database.



```
public function alterAttacking($Attacking,$PlayerUsename) {
    $sql = "UPDATE playerstats SET attacking='$Attacking' WHERE username='$PlayerUsename'";
    return mysqli_query($this->conn, $sql);
}
```

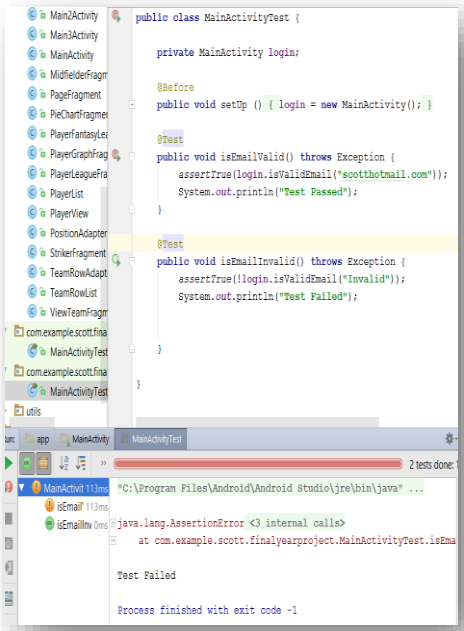
Code Snippet 35

Chapter 5 – System Evaluation

In this section, I will discuss the different testing techniques carried out throughout the development of this project.

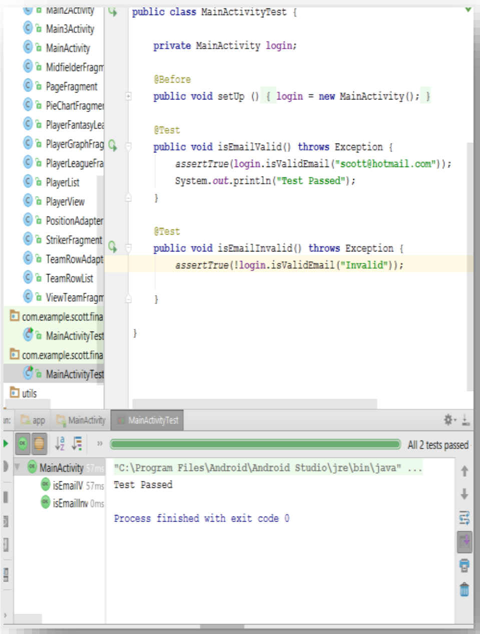
5.1 Unit Test

Unit testing is the concept of testing aspects of our code, which provides us with the ability to verify that these aspects run accordingly [35].



```
public class MainActivityTest {

    private MainActivity login;

    @Before
    public void setUp () { login = new MainActivity(); }

    @Test
    public void isEmailValid() throws Exception {
        assertTrue(login.isValidEmail("scotthotmail.com"));
        System.out.println("Test Passed");
    }

    @Test
    public void isEmailInvalid() throws Exception {
        assertTrue(!login.isValidEmail("Invalid"));
        System.out.println("Test Failed");


    }

}
```

```
"C:\Program Files\Android\Android Studio\jre\bin\java" ...

java.lang.AssertionError <3 internal calls>
    at com.example.scott.finalyearproject.MainActivityTest.isEma

Test Failed

Process finished with exit code -1
```

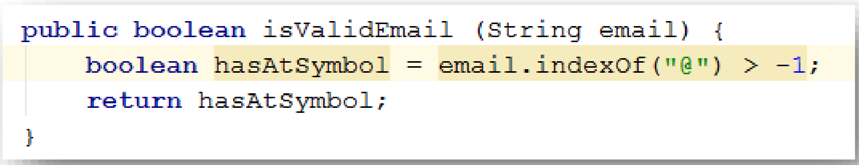


```
public class MainActivityTest {

    private MainActivity login;

    @Before
    public void setUp () { login = new MainActivity(); }

    @Test
    public void isEmailValid() throws Exception {
        assertTrue(login.isValidEmail("scott@hotmail.com"));
        System.out.println("Test Passed");
    }

    @Test
    public void isEmailInvalid() throws Exception {
        assertTrue(!login.isValidEmail("Invalid"));

    }

}
```

```
"C:\Program Files\Android\Android Studio\jre\bin\java" ...
Test Passed

Process finished with exit code 0
```

Figure 5.1: Email Validation

Figure 5.1 is a local test named MainAcvitityTest which was created to ensure a valid email address has been entered. Two local tests were run. The first is a function called isEmailValid() that has been defined in the MainActivity where a user's login credentials will be checked. In the MainActivity where an email is entered, there is a Boolean that checks if the string contains the symbol "@" as shown below in figure 5.2.



```
public boolean isValidEmail (String email) {
    boolean hasAtSymbol = email.indexOf("@") > -1;
    return hasAtSymbol;
}
```

Figure 5.2: Boolean hasAtSymbol

In figure 5.1 the image on the left displays a test with the login address “scotthotmail.com” which doesn’t contain the “@” symbol. In the console, an error is displayed which indicated that the test has failed and the email is not valid.

The image on the right displays the email address scott@hotmail.com with the “@” symbol and in the console, it prints that the test has passed.

This test upon success was then implemented into the user’s login where if the email address is valid an AsyncTask will be made to ensure the correct credentials have been entered.

5.2 Think Aloud Protocol

“In a thinking aloud test, you ask test participants to use the system while continuously thinking out loud — that is, simply verbalizing their thoughts as they move through the user interface.” [36, p1]

For this test, I asked a friend of mine to create a new account as a manager, to create a team and to add players to his team without any guidelines and recorded his thoughts on the 01/04/2017.

No.	Process	Response
1	Register an account	“Straight forward”
2	Create a team	“Which email is this for”
3	Add Players	“Nice and simple”

Once my friend had completed this I asked him to sign into the android application and assign an activity to a player without any guidelines.

No.	Process	Response
1	Sign in	“Straight forward”
2	Home screen	“The design looks well”
3	Match day fragment (Select Squad)	“Difficult to tell which players are selected”
4	Assign Activities	“Very straight forward”

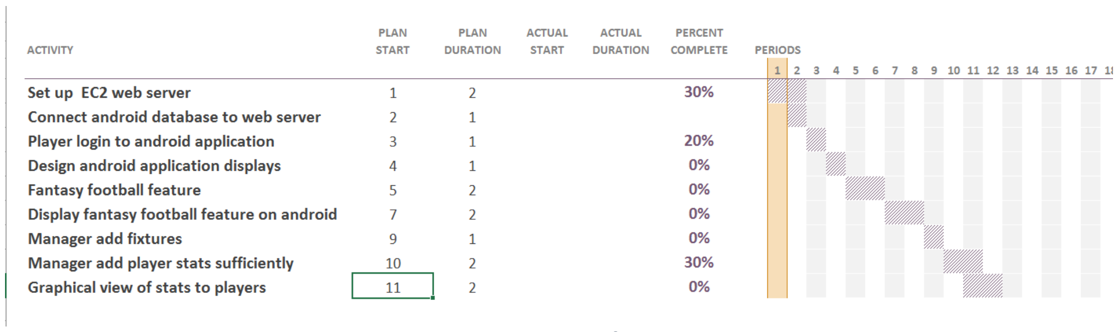
Regarding this feedback, it is a priority to use this information to improve the applications. Overall his experience was positive and with a few minor adjustments, these issues could be fixed.

5.3 Black Box Testing

Black box testing was carried out using a questionnaire which I’ve attached in appendix C.

Chapter 6 – Project Plan

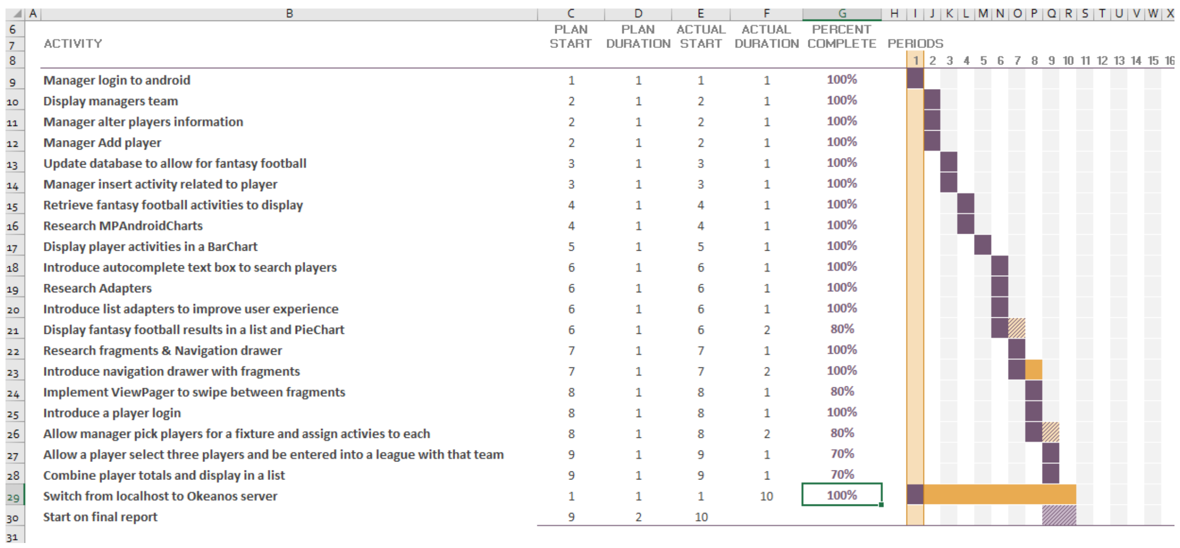
6.1 Plan Analysis



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Set up EC2 web server	1	2			30%
Connect android database to web server	2	1			
Player login to android application	3	1			20%
Design android application displays	4	1			0%
Fantasy football feature	5	2			0%
Display fantasy football feature on android	7	2			0%
Manager add fixtures	9	1			0%
Manager add player stats sufficiently	10	2			30%
Graphical view of stats to players	11	2			0%

Figure 6.1: Initial plan for Semester 2

Figure 6.1 illustrates the proposed plan which was created in semester 1, prior to the interim report presentations. The 'plan start' section indicates the week numbers of semester 2 and the 'planned duration' indicates the number of weeks proposed to spend on a feature. However, as I was unhappy with how the presentation went and felt the project involved no complexity once exams had been completed I began to implement the Web Application through Angular2 believing that this would add complexity which I later found wasn't. Because of this, the Web Application made no progress since Christmas.



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Manager login to android	1	1	1	1	100%
Display managers team	2	1	2	1	100%
Manager alter players information	2	1	2	1	100%
Manager Add player	2	1	2	1	100%
Update database to allow for fantasy football	3	1	3	1	100%
Manager insert activity related to player	3	1	3	1	100%
Retrieve fantasy football activities to display	4	1	4	1	100%
Research MPAndroidCharts	4	1	4	1	100%
Display player activities in a BarChart	5	1	5	1	100%
Introduce autocomplete text box to search players	6	1	6	1	100%
Research Adapters	6	1	6	1	100%
Introduce list adapters to improve user experience	6	1	6	1	100%
Display fantasy football results in a list and PieChart	6	1	6	2	80%
Research fragments & Navigation drawer	7	1	7	1	100%
Introduce navigation drawer with fragments	7	1	7	2	100%
Implement ViewPager to swipe between fragments	8	1	8	1	80%
Introduce a player login	8	1	8	1	100%
Allow manager pick players for a fixture and assign activies to each	8	1	8	2	80%
Allow a player select three players and be entered into a league with that team	9	1	9	1	70%
Combine player totals and display in a list	9	1	9	1	70%
Switch from localhost to Okeanos server	1	1	1	10	100%
Start on final report	9	2	10		

Figure 6.2: Semester 2 progress

Figure 6.2 indicates in much more detail as to what was implemented during semester 2 and the time frame for each feature. As the Web Application came to a halt the entire focus was on the Android Application as development hadn't begun. Due to the little complexity involved in the project, it was decided to ensure the user's experience was positive and that the design was clean and elegant.

6.2 Future Work

My aim is to introduce Football Manager Companion to teams I participate in which will involve introducing an abundance of new features.

Android Hardware Camera API: The aim here is to allow a player upload their own photo to their account.

Google Maps API: Implementing the google maps API that would provide a player with directions to an away pitch would be an interesting feature to implement however it would not be a priority.

Create a password to join a fantasy football league: Adding a password set by the manager that allows a player join a league would be a priority. Currently, players can enter a league and that its.

Fantasy Football Transfers: Assigning a value to a player and providing the manager of a team with a budget to choose his players would be a feature that would be implemented to improve competitiveness within the fantasy football league.

Adding Activities: Currently, in the application, there are only two activities a player can be assigned and they are both input at the same time. To produce an application that would be sufficient for a team to use there would need to be 3 or 4 more activities that could be assigned to a player.

Adding Statistics: Currently, in the application, there are only two statistics per player. To produce an application that would be sufficient for a team to use there would need to be more statistics per player.

Recommend a Team: Implementing a feature that can recommend a manager his best team for each formation based on a player's statistics is a feature that will be implemented.

Improving the Web Application: As I have partially implemented the Web Application through Angular2 I would like to continue with that and improve the functionality associated with the Web Application.

Chapter 7 – Conclusion

The primary aim of this project was to develop a Web Application and an Android Application that would allow the manager of a football team record activities during matches and alter player's statistics throughout the season.

One of the major challenges associated with this project for me was that originally I was unsure as to what I would be able to achieve which led to poor planning. Because of this, I spent a lot of time implementing features that I would later alter to improve the user's interaction and how the data is manipulated to and from the database. As the project developed, my understanding of the system grew and I felt more confident in what I was doing.

Another challenge for me was the complexity issue. Prior to submission of the interim report and the presentation, I had been developing the web application using MySQL and PHP which was running on an Okeanos server. In the aftermath of the interim report and the presentation, I began to develop an entirely new system using Angular2 and Node. At the time, I believed introducing languages and techniques that I was unfamiliar with was adding complexity when it wasn't and it's only now that I fully understand this. This evidently took time that I didn't have and slowed progress.

I feel that because of my poor planning and my lack of knowledge, in the beginning, I didn't fully achieve what I may have could. My plan was to implement a system where a manager could access a web application and create a team and add his players. I've achieved this and from the android application, I've also met my goals where a manager can input data related to his players and players can view this data in graphical form.

To conclude this project, I do feel that the knowledge I gained from working on this project will be very beneficial to me and how I can learn from the mistakes I made.

If I were to offer advice to someone who may be doing a similar project in the future, I would recommend researching and learning in a lot more detail about the current and emerging technologies and making use of the most efficient ways.

Throughout the duration of this project, I found myself becoming increasingly frustrated when dealing with material I was unfamiliar with. A prime example of this was Amazon's EC2 Web Service. Without researching the underlying architecture, and beginning on tutorials I was creating services and following videos but I was unaware as to what exactly was going on. I did successfully deploy the web application to Amazons Elastic Beanstalk, however, there was an overwhelming amount of services that I was unfamiliar with which I didn't feel comfortable about.

Being unaware of how to maintain the database efficiently or how to stop and start the server was an indication to me that availing of the services was negative which is when I found it could be a lot more convenient to create an Ubuntu server through Okeanos where I manually configure everything through LAMP.

A second example of this would be when I first began to develop through Android Studio I was appending URL's with multiple strings each time I made a request whereas once I began to gain a better understanding of the system and began to understand how services were communicating with one another I learnt that I could attach objects or arrays through JSON which could be processed through JavaScript in the backend.

As discussed in section 6.2 in future work I have highlighted key features that I would like to implement once I have time and to refer to my own advice and research and fully understand what's happening before attempting to develop anything.

Bibliography

[1] World's Most Popular Sports by Fans [Online]. [cited 2016 November 10]. Available from: http://www.topendsports.com/world/lists/popular-sport/fans.htm

[2] Fifa Magazine BigCount [Online]. [cited 2016 November 26]. Available from: http://www.fifa.com/mm/document/fifafacts/bcoffsurv/emaga_9384_10704.pdf

[3] Chart of the week: The relentless rise of the smartphone [Internet]. MoneyWeek. 2015 [cited 2016 November 11]. Available from:
http://moneyweek.com/chart-of-the-week-the- http://moneyweek.com/chart-of-the-week-the-relentless-rise-of-the-smartphone/

[4] Campbell R. What are Powerleague football clubs [Internet]. [cited 2016 November 25]. Available from: https://www.powerleague.co.uk/about-us

[5] Fantasy Premier League, Official Fantasy Football Game of the Premier League [Internet]. [cited 2016 November 25]. Available from: https://fantasy.premierleague.com/a/home

[6] Everything You Need to Know About TeamSnap [Internet]. [cited 2016 November 23]. Available from: https://www.teamsnap.com/about

[7] Teamer: Sports Team Management [Internet]. Teamer. [cited 2016 November 21]. Available from: http://teamer.net/demo1/

[8] Club Fantasy Football | Create your own Fantasy Football League [Internet]. [cited 2017 Apr 5]. Available from: http://www.clubfantasyfootball.co.uk/

[9] PHP: What is PHP? - Manual [Internet]. [cited 2016 October 15]. Available from: http://php.net/manual/en/intro-whatis.php

[10] CSS Introduction [Internet]. [cited 2016 October 11]. Available from: http://www.w3schools.com/css/css_intro.asp

[11] Buckler C. SQL vs NoSQL: How to Choose [Internet]. SitePoint. 2015 [cited 2016 November 1]. Available from: https://www.sitepoint.com/sql-vs-nosql-choose/

[12] MySQL database | Motive Glossary [Internet]. [cited 2017 Apr 5]. Available from: http://www.motive.co.nz/glossary/mysql.php?ref

[13] How to Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu [Internet]. DigitalOcean.[cited 2017 Apr 5]. Available from:
https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu

[14] Lau D. 10 Reasons Why You Should Use AngularJS [Internet]. SitePoint. 2013 [cited 2016 November 17]. Available from: https://www.sitepoint.com/10-reasons-use-angularjs/

[15] Mehul R. Why Android Studio Is Better for Android Developers Instead of Eclipse - DZone Mobile [Internet]. dzone.com. [cited 2016 November 26]. Available from: https://dzone.com/articles/why-android-studio-better

[16] What is Eclipse? - Definition from WhatIs.com [Internet]. SearchMicroservices. [cited 2017 Apr 5]. Available from: http://searchmicroservices.techtarget.com/definition/Eclipse

[17] ADT Plugin (DEPRECATED) | Android Studio [Internet]. [cited 2017 Apr 5]. Available from: https://developer.android.com/studio/tools/sdk/eclipse-adt.html#notes

[18] The decline of apple, Rise of android [Internet]. 2016 [cited 2016 November 16]. Available from: http://www.instantflashnews.com/en/2016/06/02/6-2-xiaomi-bought-1500-patents-from-microsoft-in-next-2-years-lenovo-wants-to-sell-phone-at-product-costs-phone-panels-supply-decline-is-the-reason-of-rising-prices-etc/

[19] Java Resources for Students, Hobbyists and More | go.Java | Oracle [Internet]. [cited 2016 November 20]. Available from: https://go.java/index.html

[20] Java: To Learn or Not to Learn? [Internet]. Effective Agile Software Development and IT Outsourcing. [cited 2016 November 11]. Available from: http://www.acceptic.com/blog/java-development-to-learn-or-not-to-learn.html

[21] Instrumentation | Android Developers [Internet]. [cited 2016 November 1]. Available from: https://developer.android.com/reference/android/app/Instrumentation.html

[22] Getting Started with Testing | Android Developers [Internet]. [cited 2017 Apr 5]. Available from: https://developer.android.com/training/testing/start/index.html

[23] PhilJay/MPAndroidChart [Internet]. GitHub. [cited 2017 Apr 5]. Available from: https://github.com/PhilJay/MPAndroidChart

[24] Transmitting Network Data Using Volley | Android Developers [Internet]. [cited 2017 Apr 5]. Available from: https://developer.android.com/training/volley/index.html

[25] AsyncTask | Android Developers [Internet]. [cited 2017 Apr 5]. Available from: https://developer.android.com/reference/android/os/AsyncTask.html

[26] Gasson, S. (1999) THE REALITY OF USER-CENTERED DESIGN.

[27] Should we choose Iterative or Agile? [Internet]. Nomad8. 2012 [cited 2017 Apr 5]. Available from: https://nomad8.com/should-we-choose-agile-or-iterative/

[28] What is Agile Software Development? [Internet]. Agile Alliance. 2015 [cited 2017 Apr 5]. Available from: https://www.agilealliance.org/agile101/

[29] A decade of agile methodologies: Towards explaining agile software development / The Journal of Systems and Software 85 (2012) 1213–1221

[30] Iterative Model in Software Development and Testing [Internet]. Testing Excellence. 2008 [cited 2017 Apr 5]. Available from: http://www.testingexcellence.com/iterative-model/

[31] Tidwell, J. (2011). Designing interfaces. 1st ed. Sebastopol, CA: O'Reilly.

[32] Murphy, M. (2011) The Busy Coders Guide to Android Development

[33] Drill down navigation | Android Interaction Design Patterns | [Internet]. [cited 2017 Apr 5]. Available from: https://unitid.nl/androidpatterns/uap_pattern/drill-down-navigation

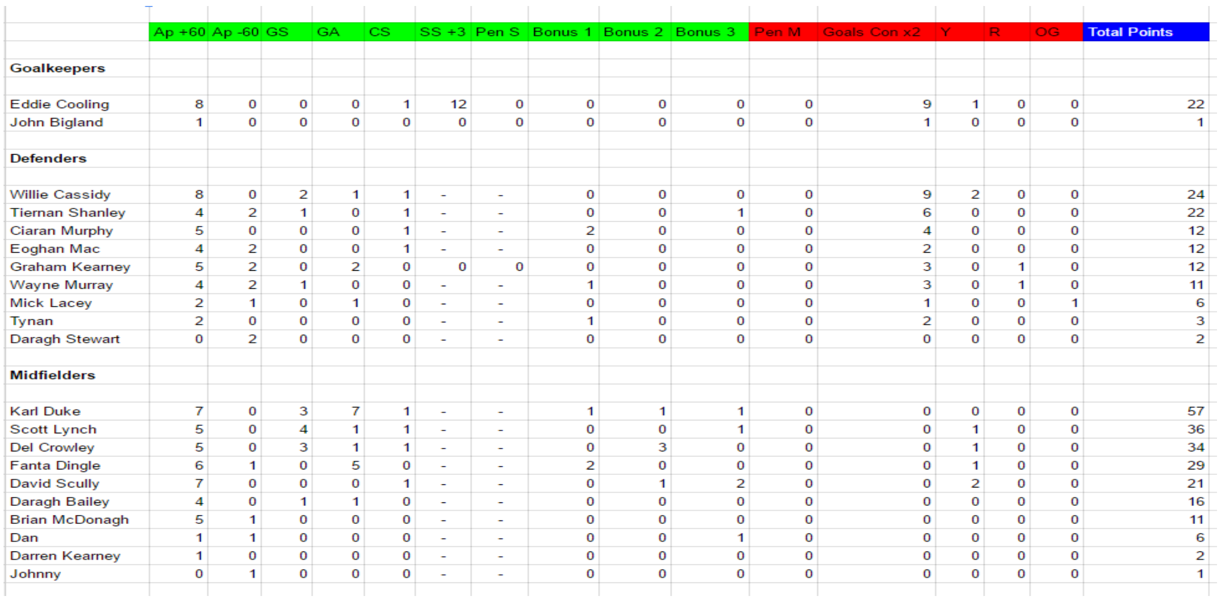
[34] Spinner | Android Interaction Design Patterns | [Internet]. [cited 2017 Apr 5]. Available from: https://unitid.nl/androidpatterns/uap_pattern/spinner

[35] McFarlin T. The Beginner’s Guide to Unit Testing: What Is Unit Testing? [Internet]. [cited 2016 November 17]. Available from: https://code.tutsplus.com/articles/the-beginners-guide-to-unit-testing-what-is-unit-testing--wp-25728

[36] Nielsen, J. (2009). Usability engineering. 1st ed. Amsterdam [u.a.]: Kaufmann.

Appendix

A) Fantasy football league table through Excel



	Ap +60	Ap -60	GS	GA	CS	SS +3	Pen S	Bonus 1	Bonus 2	Bonus 3	Pen M	Goals Con x2	Y	R	OG	Total Points
Goalkeepers																
Eddie Cooling	8	0	0	0	1	12	0	0	0	0	0	9	1	0	0	22
John Bigland	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
Defenders																
Willie Cassidy	8	0	2	1	1	-	-	0	0	0	0	9	2	0	0	24
Tiernan Shanley	4	2	1	0	1	-	-	0	0	1	0	6	0	0	0	22
Ciaran Murphy	5	0	0	0	1	-	-	2	0	0	0	4	0	0	0	12
Eoghan Mac	4	2	0	0	1	-	-	0	0	0	0	2	0	0	0	12
Graham Kearney	5	2	0	2	0	0	0	0	0	0	0	3	0	1	0	12
Wayne Murray	4	2	1	0	0	-	-	1	0	0	0	3	0	1	0	11
Mick Lacey	2	1	0	1	0	-	-	0	0	0	0	1	0	0	1	6
Tynan	2	0	0	0	0	-	-	1	0	0	0	2	0	0	0	3
Daragh Stewart	0	2	0	0	0	-	-	0	0	0	0	0	0	0	0	2
Midfielders																
Karl Duke	7	0	3	7	1	-	-	1	1	1	0	0	0	0	0	57
Scott Lynch	5	0	4	1	1	-	-	0	0	1	0	0	1	0	0	36
Del Crowley	5	0	3	1	1	-	-	0	3	0	0	0	1	0	0	34
Fanta Dingle	6	1	0	5	0	-	-	2	0	0	0	0	1	0	0	29
David Scully	7	0	0	0	1	-	-	0	1	2	0	0	2	0	0	21
Daragh Bailey	4	0	1	1	0	-	-	0	0	0	0	0	0	0	0	16
Brian McDonagh	5	1	0	0	0	-	-	0	0	0	0	0	0	0	0	11
Dan	1	1	0	0	0	-	-	0	0	1	0	0	0	0	0	6
Darren Kearney	1	0	0	0	0	-	-	0	0	0	0	0	0	0	0	2
Johnny	0	1	0	0	0	-	-	0	0	0	0	0	0	0	0	1

B) Trigger

```sql
CREATE TRIGGER new_player
AFTER INSERT ON player
FOR EACH ROW
  INSERT INTO playerstats (username, position, defending, attacking)
  VALUES (NEW.username, 'Midfielder', 50, 50);
```

C) Questionnaire

Question	User	Answer	Comment
Did the web application launch successfully?	User A	Yes	
	User B	Yes	
	User C	Yes	

Did you encounter any issues logging in?	User A	No	
	User B	No	
	User C	No	
Were you able to create a team?	User A	Yes	
	User B	Yes	
	User C	Yes	
Were you able to create a player and did they appear for you?	User A	Yes	
	User B	Yes	
	User C	Yes	
Did any players that you didn't register appear in your team?	User A	No	
	User B	No	
	User C	No	
Were you able to log into the Android Application with your details?	User A	Yes	
	User B	Yes	
	User C	Yes	
Were you able to navigate around the application without guidance?	User A	Yes	
	User B	No	Going back through certain fragments kept logging the user out of the application.
	User C	No	
Rate the web application design out of 10	User A	5/10	
	User B	5/10	

	User C	6/10	
Rate the android application design out of 10	User A	8/10	
	User B	9/10	
	User C	8/10	
Do you think the application has potential to gather an interest?	User A	Yes	With extra features the application could be considered by a team.
	User B	Yes	
	User C	Yes	