**Developing a Sports Analysis System**

**for the amateur American Football Scene**

****

Batetambe Tanyi

Supervisor: Iain Phillips

Module Code: COC251

Department of Computer Science

Loughborough University

A thesis submitted for the degree of

Master of Computer Science COUM01

**April 2019**

**Acknowledgments**

**Personal**

The work produced for this project would have not been possible without the help and support of my parents and members of the American Football community for suggestion of ideas in the project. I would like to give special thanks to my Project supervisor Iain Phillips for his guidance during the project.

**Abstract**

“The US National Football League (NFL) has the highest attendance of any professional sports league in the world by some distance. This Sunday, the Super Bowl will likely smash national TV ratings, while also drawing in an international audience of millions. Now, entering the internet of things (IoT) age, the amount of information coming out of a game has dramatically increased. Everything is getting sensored and ‘smarter’ and Super Bowl sci-fi is becoming a reality.”

https://www.siliconrepublic.com/machines/american-football-nfl-iot-tech

The game of American Football is gaining popularity in the UK at a rapid rate and so is participation of the sport.  There are currently 62 teams who compete in 12 divisions across three levels of football. The 10 teams who contest both the Premier Division North and South compete to reach the [Britbowl](https://en.wikipedia.org/wiki/Britbowl" \o "Britbowl). (BAFA National Leagues, Wikipedia, 2019)

The aim of this project is to improve player development by collecting sports data from players. This data can be used to analyse weakness in the team’s tactics and players performance.

Contents

[1.Project description 8](#_Toc7471336)

[2. Literature Review 11](#_Toc7471337)

[2.1 Problem domain 11](#_Toc7471338)

[2.2.1 Importance of Analytics in Sports 11](#_Toc7471339)

[2.2 Problem solution 11](#_Toc7471340)

[3. Gap Analysis 12](#_Toc7471341)

[3.1 Hudl 12](#_Toc7471342)

[3.2 Pro Football Focus 14](#_Toc7471343)

[3.3 Youtube 15](#_Toc7471344)

[3.4 Conclusion 16](#_Toc7471345)

[4 Goals, Challenges and Operating Constraints 17](#_Toc7471346)

[4.1 User interface 17](#_Toc7471347)

[4.2 Database structure 17](#_Toc7471348)

[5.Requirements 18](#_Toc7471349)

[5.1 Login Screen 18](#_Toc7471350)

[5.2 Track Player Statistics 21](#_Toc7471351)

[5.3 Games 24](#_Toc7471352)

[5.4 Player Menu 25](#_Toc7471353)

[5.5 coaches’ menu 26](#_Toc7471354)

[6.Implementation 27](#_Toc7471355)

[6.1 Database 29](#_Toc7471356)

[6.1.1 Firebase Cloud Services 30](#_Toc7471357)

[6.1.2. database 30](#_Toc7471358)

[6.1.3. Data Synchronization 32](#_Toc7471359)

[6.2 Database Schema 32](#_Toc7471360)

[6.2.1 Users 32](#_Toc7471361)

[6.2.2 Players Statistics 32](#_Toc7471362)

[6.2.3 Coaches 32](#_Toc7471363)

[6.2.4 Game Day Player Table 32](#_Toc7471364)

[6.2.5 Game Day Team Table 32](#_Toc7471365)

[6.3 Database Querying 33](#_Toc7471366)

[6.3 Sign in screens 36](#_Toc7471367)

[6.4 Main Menu 37](#_Toc7471368)

[6.4 Structure, performance and scalability 39](#_Toc7471369)

[Use of Intent 41](#_Toc7471370)

[6.5 Players Section 43](#_Toc7471371)

[6.5.1 Add players 44](#_Toc7471372)

[6.5.2 Retrieve players 46](#_Toc7471373)

[6.5.3 Select a player 47](#_Toc7471374)

[6.6 Player Statistics 49](#_Toc7471375)

[6.6.1 Page 0 Season Summary 50](#_Toc7471376)

[6.6.2 Page 1 Week-by-Week Analysis 50](#_Toc7471377)

[6.6.3 Page 2 Graphical view of players rating 51](#_Toc7471378)

[6.6.4 Page 3 53](#_Toc7471379)

[6.7 Mid Project Changes 54](#_Toc7471380)

[6.8 Games Section 56](#_Toc7471381)

[6.9 List of Games 56](#_Toc7471382)

[6.9.1 Adding a new game 57](#_Toc7471383)

[6.9.2 Starting a game 57](#_Toc7471384)

[6.10 Record Game 58](#_Toc7471385)

[6.10.1 Kick off Fragment 59](#_Toc7471386)

[6.10.2 Offence fragment 60](#_Toc7471387)

[6.10.3 Defence 62](#_Toc7471388)

[6.10.4 Field Goal 63](#_Toc7471389)

[6.10.5 PAT 64](#_Toc7471390)

[6.10.6 2pt conversion 65](#_Toc7471391)

[6.10.7 punt 66](#_Toc7471392)

[6.10.8 Kick return 67](#_Toc7471393)

[6.10.9 Ratings 68](#_Toc7471394)

[6.10.10 End Game 69](#_Toc7471395)

[6.11 Coaches 70](#_Toc7471396)

[6.12 Users 71](#_Toc7471397)

[7. Testing 72](#_Toc7471398)

[7.1 Splash screen 72](#_Toc7471399)

[7.2 Login 72](#_Toc7471400)

[7.3 Registration 73](#_Toc7471401)

[7.4 Players Section 74](#_Toc7471402)

[7.5 Player Statistics Section 75](#_Toc7471403)

[7.6 Games section 77](#_Toc7471404)

[7.7 Record Game 78](#_Toc7471405)

[7.8 Coaches Section 83](#_Toc7471406)

[7.9 Users Section 84](#_Toc7471407)

[8.Conclusion 85](#_Toc7471408)

[9. References 87](#_Toc7471409)

# 1.Project description

Sports Analysis System based on American Football

Introduction to the sport

A team is split into offense and defence and points are scored by advancing the ball into opponent territory/end zone (6 points) like rugby or by kicking the ball from the playing field through the posts (3 points). An offence has 4 chances to gain 10 yards of forward progress from the line of scrimmage until they reach the end zone.

An offence can advance by

* Throwing the ball forward once
* Running the ball
* Laterally or passing the backwards like rugby

A defence may interrupt the forward progress of the offence by

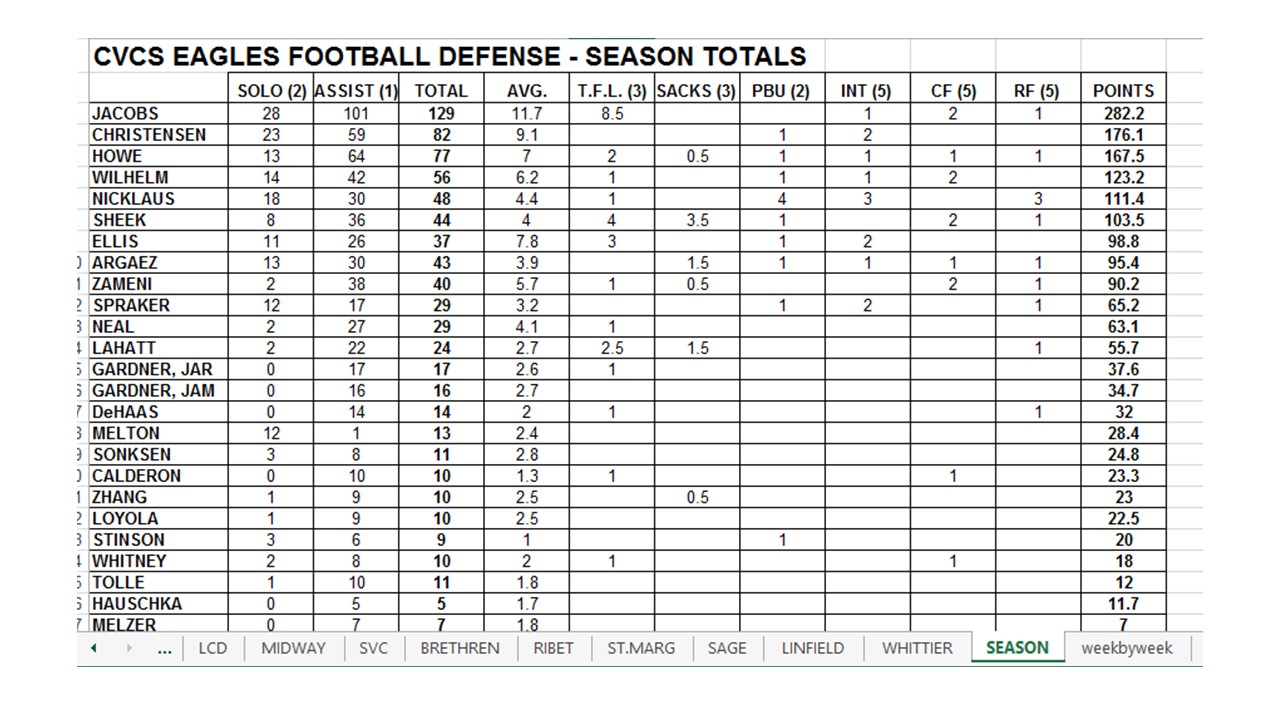
* Tackling the ball carrier and f the ball carrier contacts the ground play stops
* Causing a turnover by intercepting the ball or forcing a fumble

The game of American football may seem very aggressive and brute but requires the same intellect and strategic approach in a game such as chess so to improve performance of players coaches normally would record stats such as completed forward passes, tackles made, interceptions, fumbles and analysis trends in those stats as the season goes on in combination of game film.

In the professional game stats like this are recorded with the aid of specialist software which is possible because of funding available and access to computers inside an arena or stadium.

On the other hand, this is not possible in the amateur game this is not possible because of the lack of access to specialist software and computers as games are normally played outdoors on Sunday League Football style pitches.

So, what my project is going to deliver is a professional level of stat scripting and analysis system in the form of an iPad app to amateur level American Football team.



(Coachfore n.d.)

Specification

* Record stats produced from all different positions and store them on a database
* To be cost effective so it can be affordable to teams of any level
* The app can be used by any team
* Compare performance across teams in the same league who also use the app
* Coaches can set seasonal milestones and weekly targets for players which can be done by having accounts for each team with different levels of access
* Graphical display of how the stats of the whole team and individual are improving or declining
* Then the main part of the Analysis system which is how players and teams will be analysed.

1. I will collect data from Teams at different levels
2. Data from all the different positions
3. Data from players at different levels of experience
4. Data from teams with different play schemes
5. Data from teams with a different geographical location as weather condition such as rain or snow will affect a stat like pass completion
6. Scores of 100 will be reserved for players that break records

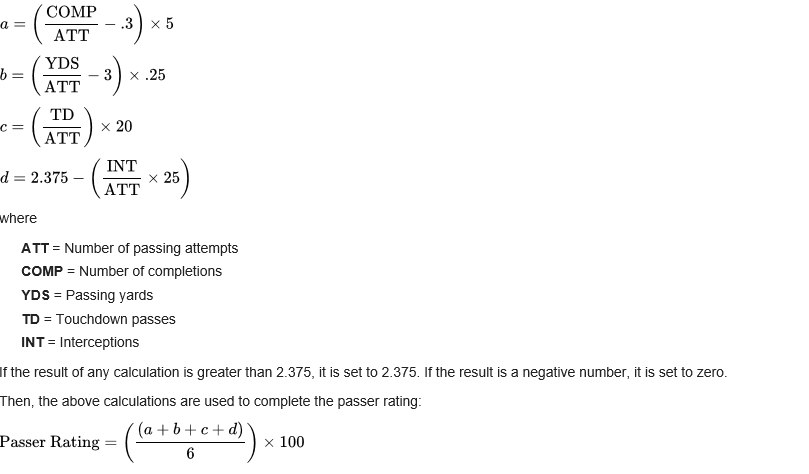
* I will use this data to first overall create a set of algorithms to compute overall season performance which will be rated out of 100 and then also performance increase and decline which will be shown graphically these first two sets of results will not be sensitive to factors such as experience and play scheme so amateur level player should expect to scores such as 20/100 but in a steady increase in the graphical analysis of performance increase which will look like y= mx +c type graph but then there will be a set of results displayed based on algorithms which will be sensitive to the factors I mentioned above which will give a realistic indication of performance of players of lower levels which will mean elite amateur level player can still achieve scores in 90s

**Example of Passer rating is calculated in NFL I will be using this formula in my project**

“The NFL passer rating formula includes four variables: completion percentage, yards per attempt, touchdowns per attempt, and interceptions per attempt. Each of those variables is scaled to a value between 0 and 2.375, with 1.0 being statistically average (based on league data between 1960–1970). When the formula was first created, a 66.7 rating indicated an average performance, and a 100+ rating indicated an excellent performance.[[3]](https://en.wikipedia.org/wiki/Passer_rating#cite_note-a-3) However, passing performance has improved steadily since then and in 2017 the league average rating was 88.6.[[4]](https://en.wikipedia.org/wiki/Passer_rating#cite_note-4)

The four separate calculations can be expressed in the following equations:

a = ( COMP ATT − .3 ) × 5 {\displaystyle a=\left({{\text{COMP}} \over {\text{ATT}}}-.3\right)\times 5}

b = ( YDS ATT − 3 ) × .25 {\displaystyle b=\left({{\text{YDS}} \over {\text{ATT}}}-3\right)\times .25}

|  |  |
| --- | --- |
| **A perfect passer rating (158.3) requires at least:**[**[1]**](https://en.wikipedia.org/wiki/Passer_rating#cite_note-NFLQBRating-1) | **A minimum rating (0.0) requires at best:** |
| 77.5% completion percentage 12.5 yards per attempt 11.875% TD/ATT (1 TD/8.421ATT) No interceptions | 30.0% completion percentage 3.0 yards per attempt 0.0% TD/ATT 9.5% INT/ATT (1INT/10.526ATT)  (Passer Rating, Wikipedia, 2019) |

c = ( TD ATT ) × 20 {\displaystyle c=\left({{\text{TD}} \over {\text{ATT}}}\right)\times 20}

Passer Rating = ( ( a + b + c + d ) 6 ) × 100 {\displaystyle {\text{Passer Rating}}=\left({(a+b+c+d) \over 6}\right)\times 100}

# 2. Literature Review

## 2.1 Problem domain

This project aims to improve the process of player development in the amateur level of American Football. There have already been attempts to include technology in the process of player development such as Hudl which allows players and coaches to view game film, to allow critique. This process has seen a lot of success, but I believe that there is a lack of automation. Coaches normally critique players using what they see of game film, but this process can be improved and extended by using the data produced by players in the game which is normally recorded at the lower level.

Players need to normally approach coaches for advice whereas a system could give them critique points just based on their seasonal stats. Coaches manually record stats midgame using a printed stat template. Recording stats that way can be very inefficient as during high pace sequence of consecutive plays players actions may be missed. In a windy or humid environment, it would be difficult to record data using pen and paper.

For player development to be effective coaches need to be flexible, so they will need some form of communication between players which is available in HUDL but is not directly available to the player being analyzed.

For players to improve they continually need to be challenged and have their limits pushed. At the lower level this is difficult to do as communication can be challenging as players and coaches are limited at times on when they communicate as most players and coaches see American football as a hobby and have full time jobs alongside of playing. (Competitive Advantage: Mental Toughness, n.d.)

<https://www.competitivedge.com/special-what-makes-good-coach>

### 2.2.1 Importance of Analytics in Sports

One of the most famous success stories is the story of Oakland Athletics Baseball team which went from being a losing team to winning the World Series. The driving-force behind this was the general manager being inspired by an economics graduate to use Sabermetrics (empirical analysis of [baseball](https://en.wikipedia.org/wiki/Baseball), especially [baseball statistics](https://en.wikipedia.org/wiki/Baseball_statistics) that measure in-game activity).

## 2.2 Problem solution

The project will attempt to reduce the impact or solve completely the issues I have mentioned above.

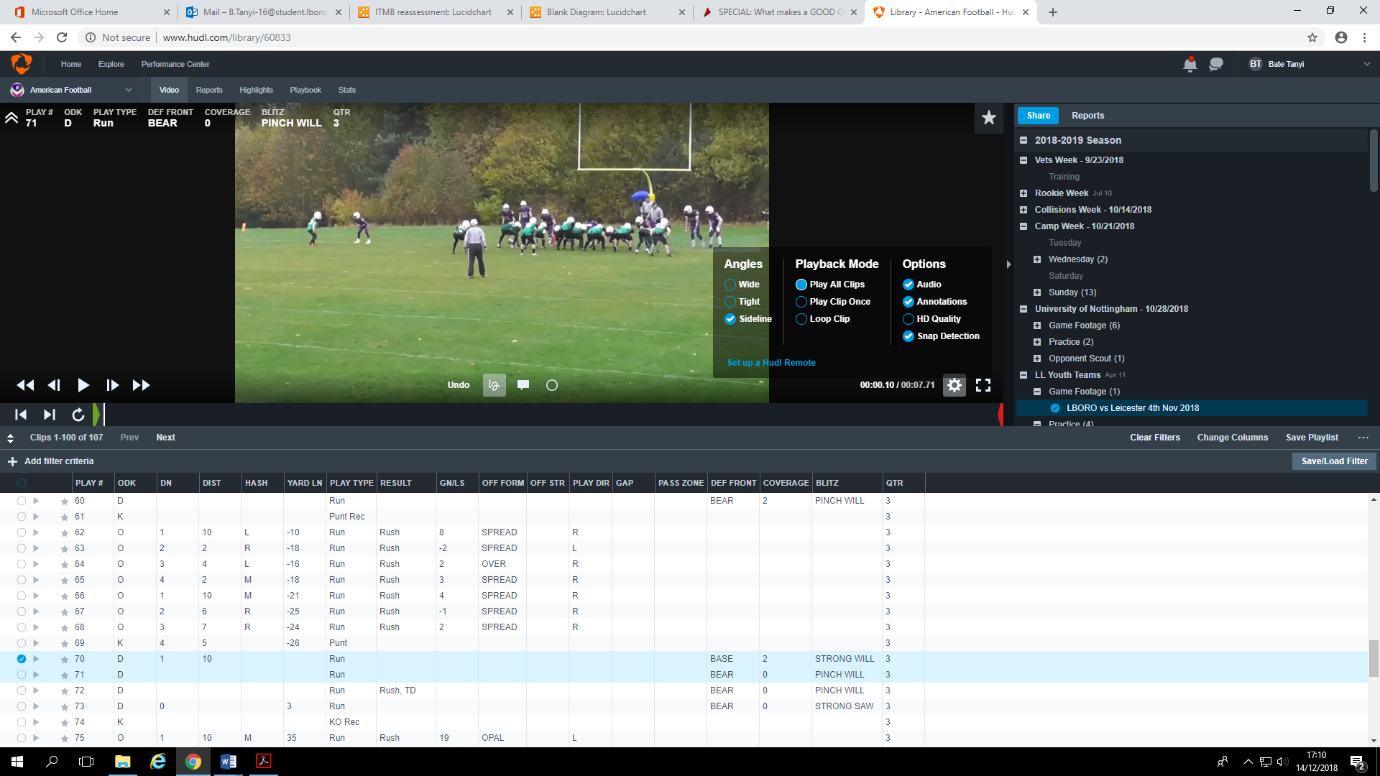
Efficiency is improved by speeding up the time it takes to record the stats mid game by implementing a Graphical User interface to record stats.

Coaches will be more flexible with communication with players as they will be able to leave players notes and goals based on their current progress which will be automatically calculated using algorithms.

# 3. Gap Analysis

In this section I will compare competing applications to decide whether my proposed application will compare to its marketplace. This section will explore if there potentially will be a gap in the industry for my application.

## 3.1 Hudl



Hudl is video platform which allows coaches and players to view to record their games and training sessions and break them to down into separate plays. Coaches can leave notes on certain plays to give advice to their players and players can add plays to their highlight tape to be viewed by scouts. Breaking the game down in this format allows key plays to be found quicker. This allows coaches Create a presentation for team review or pull together clips to illustrate what a player needs to improve.

Advantages

* It saves time on film: It reduces the time to make game film available to players as its all stored on one centralized platform
* Provides the ability to do analytics
* Player involvement in film

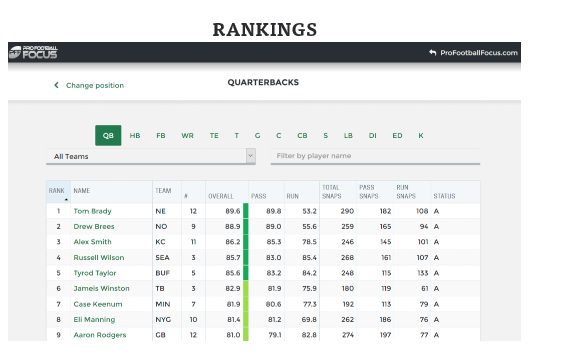
Disadvantages

* Highlights are chunky
* You must manually export stats by copying and pasting the data instead of it being readily available to be exported

(Brablc, 2017)

The strengths of Hudl far outweigh the weaknesses as the weaknesses are nit-picks from a power user category that I doubt many would fall into.

## 3.2 Pro Football Focus



Pro Football Focus is a website that focuses on thorough analysis of the National Football League and NCAA Division-I football in the United States. It is a database driven website and rates players with scores of 100 and ranks payers by position. My app will be very similar to this.

**Advantages**

* all information can be found in one place increasing simplicity
* database driven graphic user interface is used making it easy to navigate around data

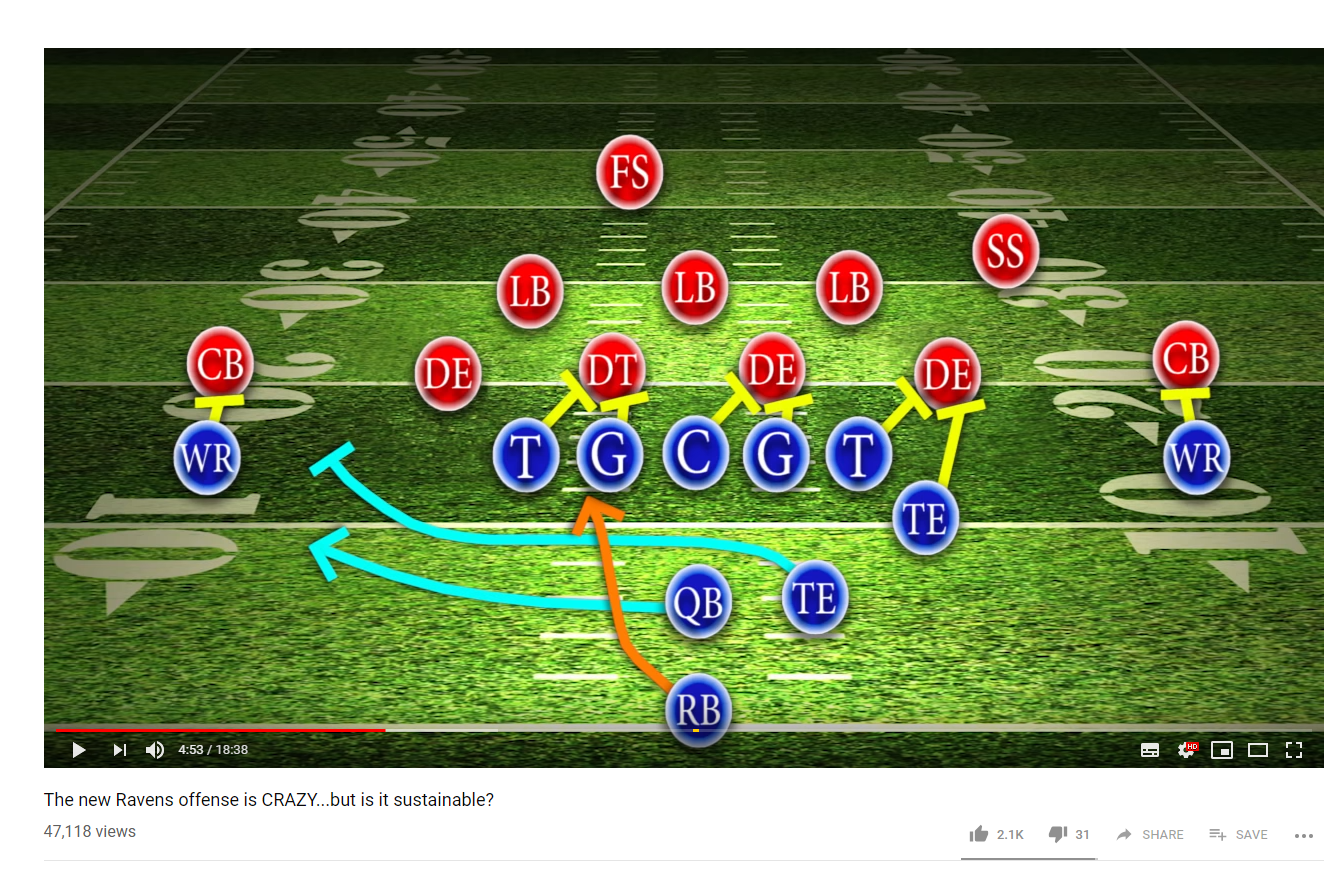
**Disadvantages**

PFF uses qualitative and opinion-based grading as the root of its 0-100 Player Grades -- not its advanced statistics. As such, the 0-100 Player Grades are not truly quantitative and could be prone to bias, poor sample sizing, or other issues.

Pff.En.wikipedia.org. (n.d.)

https://en.wikipedia.org/wiki/Pro\_Football\_Focus

## 3.3 Youtube

****

YouTube will always be one of the greatest learning tools and has wide range of educational videos when it comes to American Football. In this example video a breakdown of a run play is shown. The weak points and strengths of this formation vs a typical defensive formation is discussed. Both offensive and defensive players can take away great points from the video to overall improve their game.

**Advantages**

* Videos are the best way to present information as they can combine text, images and audio commentary to deliver the information
* Users can learn at their own pace as they can pause, slow down or repeat videos
* Flexibility is needed for player development and YouTube coaching videos can be accessed by players whenever they want
* Videos are engaging so players will find it easier to focus on their learning

(Ispringsolutions.com, n.d.)

https://www.ispringsolutions.com/articles/key-advantages-of-video-lectures.html

**Disadvantages**

* Does not replace the one to one coaching available from a player’s team coach
* Video learning is individualistic might improve individual player performance but not team performance

## 3.4 Conclusion

There are very similar applications to my app, but they are not in direct competition as most of the apps are not aimed at small time users.

Within the UK there are very few direct competitors as the sport of American football is still blooming in the UK compared to sports like Football and Rugby. All do ones that currently do exist like Hudl will not provide some of the functions my app can provide.

The reason for the lack of competition is because of the low number current players of the sport in the UK no software development companies have seen an interest to develop a large-scale app based around American football as it could potentially be unprofitable.

I believe that there is space in the market for my app as my app targets the small market. My app will incorporate a lot of functions by professional level budget applications at an affordable price. It will combine multiple different unique function of certain apps e.g. Hudl’s stat analysis and PFF player grading to create unique experience for coaches and players.

# 4 Goals, Challenges and Operating Constraints

This application is a database driven distributed system which will mean that there will be certain challenges in the creation with this kind of system. It is also important to consider factors caused by the system environment which it will run under. Often an app which is available to be used by the general public crashes when it is put under pressure crashes in a more stressful environment.

In this chapter I will discuss the areas of the applications that would be a focus in development and how the meet the challenges this type of application presents

## 4.1 User interface

The user interface should be a GUI and be very simplistic but with a premium look. The application will be recording high-speed real-time data, so the user interface should very easy fast and easy to navigate to allow all data to be captured. The application will be adopted by multiple different teams, with a diverse set of branding. It is important that the application does not clash with a team’s style e.g. colours and can be adapted to teams of different sizes.

## 4.2 Database structure

The database to be structured in such a way that all data required for a process are available in one read of the database as due to the asynchronous nature of the database system I will be using the scenario of using multiple reads will not be plausible. This is due to the fact that data loaded asynchrously does not follow the order it was requested in which could lead to data being displayed in the wrong order

# 5.Requirements

## 5.1 Login Screen

**System Feature 1**

**Create new coach account.**

**Description and Priority**

The user should be able to create a coach access level account. **High priority**

**Stimulus/Response Sequences**

Interface should allow user to enter account details.

Go to options screen on sign in.

**Functional Requirements**

Locally stored user data or database connection.

**System Feature 2**

**Create new player account.**

**Description and Priority**

The user should be able to create a player access level account. **High priority**

**Stimulus/Response Sequences**

Interface should allow user to enter account details.

Go to options screen on sign in.

**Functional Requirements**

Locally stored user data or database connection.

**System Feature 3**

**Coach sign in.**

**Description and Priority**

A coach should be able to sign in with email and password. **High priority**

**Stimulus/Response Sequences**

Interface should allow user to enter account details.

Go to options screen on sign in.

**Functional Requirements**

Locally stored user data or database connection.

**System Feature 4**

**Player sign in.**

**Description and Priority**

A player should be able to sign in with email and password. **High priority**

**Stimulus/Response Sequences**

Interface should allow user to enter account details.

Go to options screen on sign in.

**Functional Requirements**

Locally stored user data or database connection.

**System Feature 5**

**Validate details.**

**Description and Priority**

Refuse duplicate account credentials, or incorrectly formatted email and password when trying to create an account. **High priority**

**Stimulus/Response Sequences**

User is informed account already exists or email/password is invalid.

**Functional Requirements**

None

**System Feature 6**

***Reject log in attempt with incorrect details.***

**Description and Priority**

User cannot log in with incorrect account details. **High priority**

**Stimulus/Response Sequences**

User enters incorrect account details.

Application refuses log in request.

**Functional Requirements**

None.

## 5.2 Track Player Statistics

**System Feature 1**

***View player stats***

**Description and Priority**

* Select a player from the list of all players (only applicable if the user signed in is a coach)
* View an intricate selection of the players seasonal stats that’s relevant to the players position
* View all their seasonal stats
* View their positional statistics of every single they’ve played in
* Data presented in graphical format so it can be easily understood
* **High priority**

**Stimulus/Response Sequences**

When a player is selected all his statistics should be displayed

**Functional Requirements**

Locally stored user data or database connection.

**System Feature 2**

***Add player***

**Description and Priority**

Add a player to the database of players by entering their name, height, number, weight, position. **High priority**

**Stimulus/Response Sequences**

When a player is created he should appear in the database of players.

**Functional Requirements**

Locally stored user data or database connection.

**System Feature 3**

***Remove a player***

**Description and Priority**

Remove a player from the database of players. **Low priority**

**Stimulus/Response Sequences**

Should be removed from the database of players

**Functional Requirements**

Locally stored user data or database connection.

**System Feature 4**

***Validate details.***

**Description and Priority**

Refuse duplicate player credentials, **High priority.**

**Stimulus/Response Sequences**

User is informed when user already exist.

**Functional Requirements**

None

**System Feature 5**

***Edit player stats***

**Description and Priority**

To allow coach user account to be able to go on a player’s statistics and modify a player’s stats in a scenario where mistakes were made, **Medium priority.**

**Stimulus/Response Sequences**

Players stats are updated in database

**Functional Requirements**

Locally stored user data or database connection.

## 5.3 Games

**System Feature 1**

***Record stats of a game in game***

**Description and Priority**

* Stats should be recorded quickly as the game moves at a high speed
* Should be recorded using a GUI for simplicity
* Select a game from list of fixtures to record statistics for
* **High priority**

**Stimulus/Response Sequences**

When recording in game action it should be updated immediately in the database

**Functional Requirements**

Locally stored user data or database connection.

**System Feature 2**

***View game statistics***

**Description and Priority**

Select a game from a list of games and view the stats of the game such as score, opponent, turnover differential. **Medium priority**

**Stimulus/Response Sequences**

None

**Functional Requirements**

None

**System Feature 3**

***Add a new game to the list of fixtures***

**Description and Priority**

Add a game to the list of fixtures. **High priority**

**Stimulus/Response Sequences**

When the details of a game are submitted they should appear in the database

**Functional Requirements**

Locally stored user data or database connection.

## 5.4 Player Menu

**System Feature 1**

***View his own stats***

**Description and Priority**

Select a game from a list of games and view the stats of the game such as score, opponent, and his personal statistics and the coaches rating. **High priority**

**Stimulus/Response Sequences**

**Functional Requirements**

**None**

## 5.5 coaches’ menu

**System Feature 1**

***Set players ratings***

**Description and Priority**

Select a game from a list of games and give a player a rating from a score out of 5 based on their performance. **High priority**

**Stimulus/Response Sequences**

**Functional Requirements**

None

# 6.Implementation

The implementation of the sports analysis will be covered in this section. I have decided to use cover the UI design, database design and application logic together in this chapter rather than separating them into different section as they work together in cohesion, one cannot function without the other

**Objected Oriented Programming Principles**

The application will be heavily relying on n the principles of OOP here is an example of a class called ‘player’ which will be used to set the values in the database and create player objects to fetch a player’s data.

*// Instance Variables*String **name**;  
String **position**;  
**int number**;  
**int weight**;  
**int feet**;  
**int inches**;  
**float forty**;  
**int year**;  
**int month**;  
**int day**;  
  
*//receiving***int passesCaught**;  
**int longestReception**;  
**int receivingTD**;  
  
*//passing***int passesAttempted**;  
**int passesCompleted**;  
**int interceptions**;  
**float passingPercentage**;  
**int passingYards**;  
**float yardsPerAttempt**;  
**int passingTD**;  
**float yardsPerGame**;  
  
*//rushing***int carries**;  
**int rushingYards**;  
**float yardsPerCarry**;  
**int fumbles**;  
**int fumblesRecoverd**;  
**int fumblesLost**;  
**int rushingTD**;  
  
*//defense***int tackles**;  
**int assistedTackles**;  
**int tacklesForLoss**;  
**int forcedFumbles**;  
**int dFumblesRecoverd**;  
**int sacks**;  
**int dInterceptions**;  
**float tacklesPerGame**;  
**int dTD**;  
**int passesDefended**;  
  
*//kicking***int puntYardsMax**;  
**int puntTouchBacks**;  
**int puntSetback**;  
**int kickoffYardMax**;  
**int kickoffTouchback**;  
**int kickoffSetback**;  
**int fgConverted**;  
**int fgAttempted**;  
**int fgLong**;  
  
*//returning***int biggestreturn**;  
**int returnTD**;  
  
*//*

Advantages of using OOP

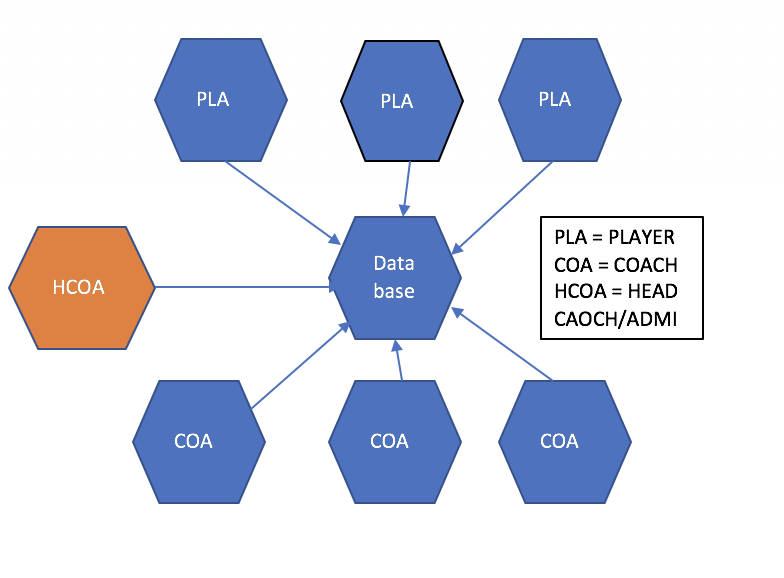
* It provides clear modular structure for programs which makes it easy for defining abstract data types. The application will be built in an iterative process so using a modular structure will allow me to follow an iterative approach in completing the project
* It provides a good framework for code libraries where the supplied software components can be easily modified which allows me to cater for scalability as in the future I plan to have the application being used worldwide.
* Faster development as code is reusable so will allow me to meet the deadline of the project.

**GSON Library**

Since the structure of my database is based on JSON Notation I will be importing and exporting JSON Data to and from my database. To handle this JSON data in an effective manner I’ve decided to use the GSON library to convert JSON Objects from my database into user defined objects such as the Player class I mentioned earlier.

* Gson can handle collections, generic types and nested classes (including inner classes, this cannot be done by default though)
* When deserializing, Gson navigates the type tree of the object being deserialized. This results in ignoring extra fields present in the JSON input.
* User can write a custom serializer and/or deserializer so that they can control the whole process and even (de)serialize instances of classes for which the source code is not accessible.

## 6.1 Database



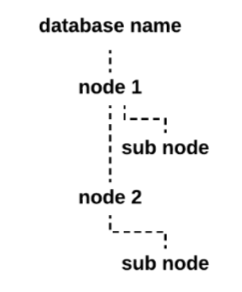
The application that has been developed for the final year project is a centralized system where there will be one admin per team and he will be the only one who will have access on all levels and responsible recording the stats in game. The system must deal with multiple users accessing it at the same time. It requires the ability to store data and access it in real time as new data created in the system will be affecting potentially new data within the next second. Nay changes to the data must be visible to all the users in real time.

Therefore, I have chosen the cloud to be the core of the system. The database being hosted in the cloud allows there to be a single point on interaction between all the users. Any changes to the database will be instantly available to all the users of the system as they are all accessing the same data set.

### 6.1.1 Firebase Cloud Services

The cloud infrastructure that was chosen for this project was Firebase by google and the application to hold the database is called Firestore.

### 6.1.2. database

****

Firebase offers a NoSQL real-time database application called Firestore. The database is cloud hosted and data is stored in a similar notation like JSON objects.

Firestore offers Flexibility

* ‘The Cloud Firestore data model supports flexible, hierarchical data structures. Store your data in documents, organized into collections. Documents can contain complex nested objects in addition to subcollections.’ This will allow me to create objects dynamically in the system and then store and retrieve them in the same format which reduce the amount time spent on creating data strcutures fo

Expressive querying

* ‘In Cloud Firestore, you can use queries to retrieve individual, specific documents or to retrieve all the documents in a collection that match your query parameters. Your queries can include multiple, chained filters and combine filtering and sorting. They're also indexed by default, so query performance is proportional to the size of your result set, not your data set. ‘The system will be gathering a large amount of data but only relevant data should be displayed.

Real time updates

* ‘Like Realtime Database, Cloud Firestore uses data synchronization to update data on any connected device. However, it's also designed to make simple, one-time fetch queries efficiently.’ This will allow the process of recording statistics midgame to be effective.

Offline support

* ‘Cloud Firestore caches data that your app is actively using, so the app can write, read, listen to, and query data even if the device is offline. When the device comes back online, Cloud Firestore synchronizes any local changes back to Cloud Firestore.’ This is an important function as the system has been designed to be used at an amateur level and some amateur teams play in rural areas where a strong mobile connection isn’t readily available.

Scalability

* Cloud Firestore brings you the best of Google Cloud Platform's powerful infrastructure: automatic multi-region data replication, strong consistency guarantees, atomic batch operations, and real transaction support. The app in the future should be considered to be used on a global scale

### 6.1.3. Data Synchronization

Optimal data synchronization is of the upmost importance for this project. Without it delays in the process of recording statistics it would lead to incorrect statistics being recorded.

Firebase provides data synchronization through the implementation of database listeners. A database listener is attached to an event such a completed action by the player or whilst retrieving a player’s or teams statistics from the database within the database by the user. Any changes to the players or team’s stats, such as update, is returned to the user in real time.

## 6.2 Database Schema

I will be discussing the tables used and their functions

### 6.2.1 Users

This table hold all the registered users of the system with their password and email address and level of access. User accounts of type player will also contain their shirt number

### 6.2.2 Players Statistics

This table holds all the players details and their statistics of the whole season

### 6.2.3 Coaches

A table which contains all coaches

### 6.2.4 Game Day Player Table

This table holds the exact same content as the previous table but only specifically for one game. Firebase Firestore has been optimized for denormalized table as it only loads the required data so there is no impact on performance

### 6.2.5 Game Day Team Table

Holds statistics which are relevant to the team as a whole e.g. turnovers, score etc

## 6.3 Database Querying

**Querying the database**

*To read a whole collection*

*db*.collection(**"users"**)  
 .get()  
 .addOnCompleteListener(**new** OnCompleteListener<QuerySnapshot>() {  
 @Override  
 **public void** onComplete(@NonNull Task<QuerySnapshot> task) {  
 **if** (task.isSuccessful()) {  
 **for** (QueryDocumentSnapshot document : task.getResult()) {  
   
   
 }  
 } **else** {  
  
 }  
 }  
 });

In this example the collection that contains all the users is queried. The result is stored in a Task object. QueryDocumentSnapshot document : task.getResult() allows you to iterate through all the documents and then use function such as task.get() to get the value at a certain field.

*To read an individual document*

**public void** passcompleteted() {  
 **doc** = RecordGame.*database*.collection(**"players"**).document(**passer**.getText().toString());  
 **doc**.get().addOnCompleteListener(**new** OnCompleteListener<DocumentSnapshot>() {  
 @Override  
 **public void** onComplete(@NonNull Task<DocumentSnapshot> task) {  
 **if** (task.isSuccessful()) {  
 DocumentSnapshot document = task.getResult();  
 **if** (document.exists()) {  
 Player passerInPlay = document.toObject(Player.**class**);  
 passerInPlay.setPassesCompleted(passerInPlay.getPassesCompleted()+1);  
 passerInPlay.setPassesAttempted(passerInPlay.getPassesAttempted()+1);  
 passerInPlay.setPassingYards(passerInPlay.getPassingYards() + Integer.*valueOf*(**passingyardsgained**.getText().toString()));  
 RecordGame.*database*.collection(**"players"**).document(**passer**.getText().toString()).set(passerInPlay);  
 RecordGame.*database*.collection(**"weeks"**).document(**passer**.getText().toString()).update(RecordGame.*weekselected* + **".passesCompleted"**, FieldValue.*increment*(1));  
 RecordGame.*database*.collection(**"weeks"**).document(**passer**.getText().toString()).update(RecordGame.*weekselected* + **".passesAttempted"**, FieldValue.*increment*(1));  
 RecordGame.*database*.collection(**"weeks"**).document(**passer**.getText().toString()).update(RecordGame.*weekselected* + **".passingYards"**, FieldValue.*increment*(Integer.*valueOf*(**passingyardsgained**.getText().toString())));  
  
  
  
 } **else** {  
 *//* }  
 } **else** {  
 *//* }  
 }  
 });  
  
}

*“database*.collection(**"players"**).document(**passer**.getText().toString());” This allows you to read a specific document in a collection then that document can be directly queried for the required field.

*Writing data*

RecordGame.*database*.collection(**"weeks"**).document(**passer**.getText().toString()).update(RecordGame.*weekselected* + **".passesAttempted"**, FieldValue.*increment*(1));

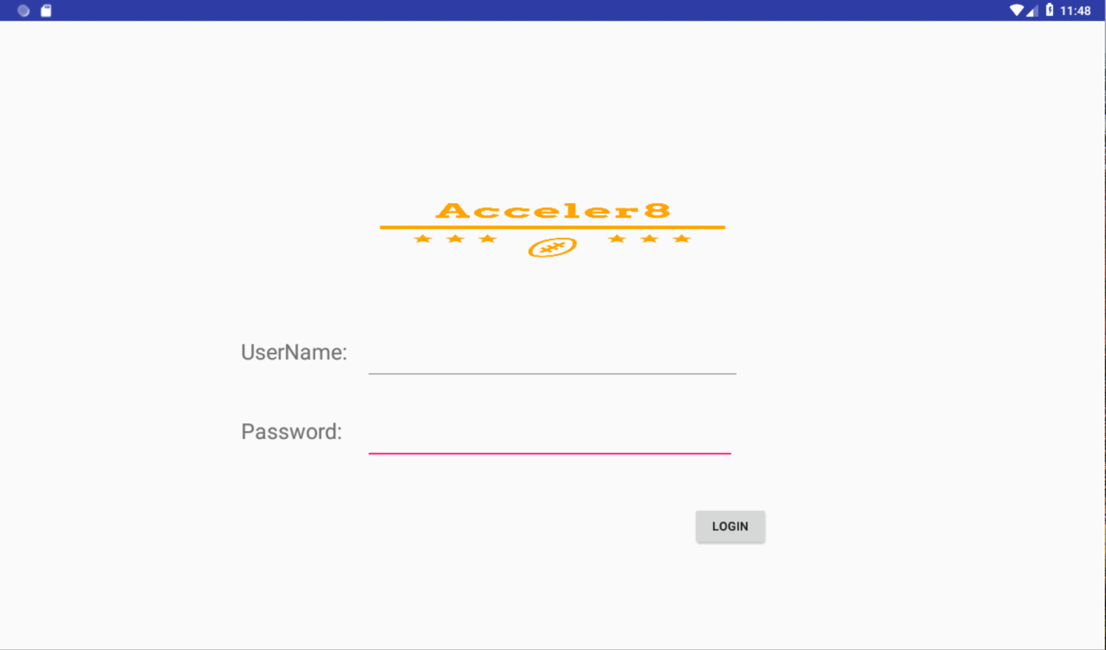
This allows you to directly write to a field in a document, by using FieldValue.increment() the value of the field can be incremented by the user entered value.

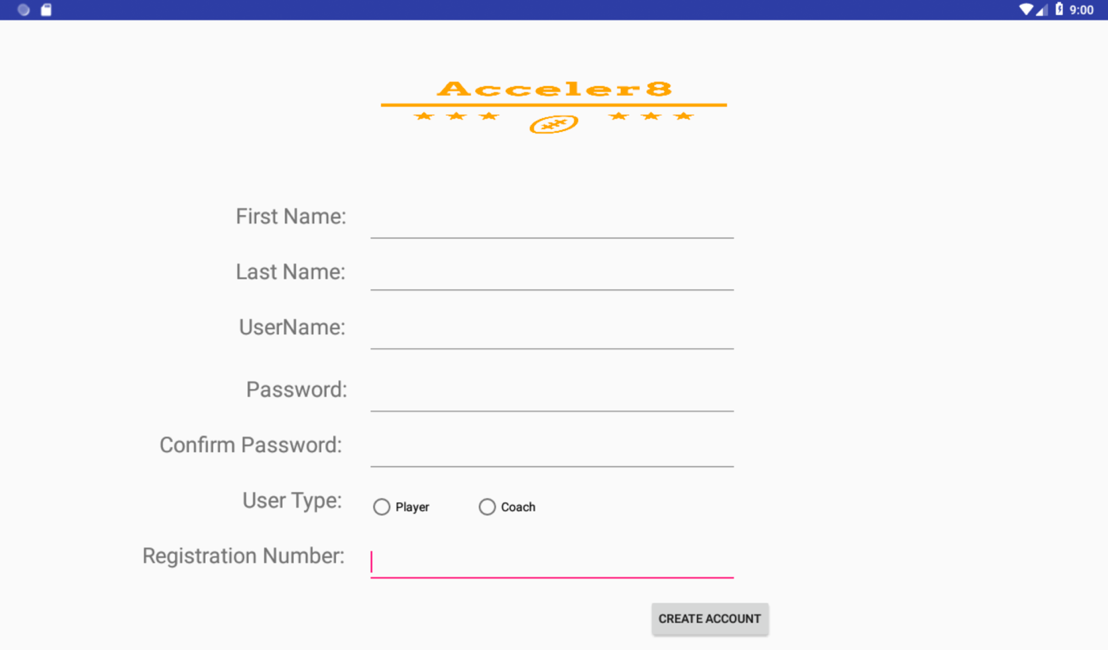
Player passerInPlay = new Player();

RecordGame.*database*.collection(**"players"**).document(**passer**)**.**set(passerInPlay);

A document in a collection can be created using an object.

## 6.3 Sign in screens

****

****

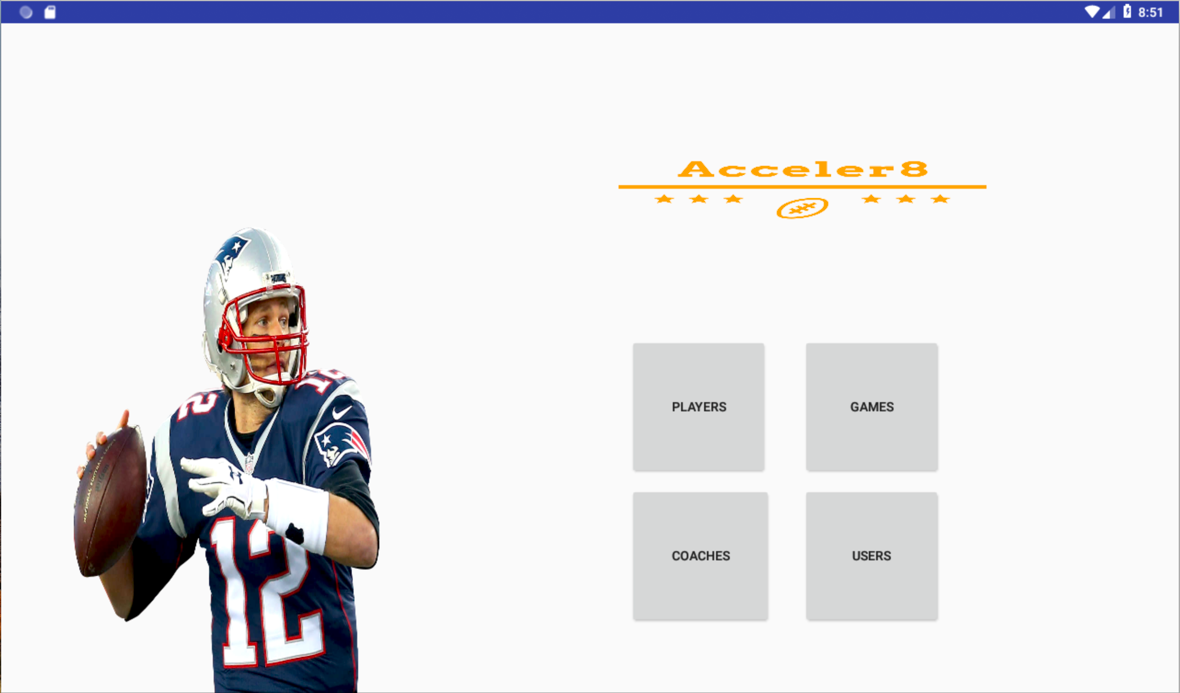
Because of the fact that the database contains sensitive information, access to the database should only be possible for authorised accounts. Since the system is a centralised system where an admin he should have the capability of authorising accounts. This achieved by preauthorisation meaning where only members who are already in the team database can create an account. Every member of the team is assigned a unique number for which they will be required to enter to be able to create an account.

## 6.4 Main Menu

The main menu will be a simple option screen where different level users will have different levs of access to separate sections of the app

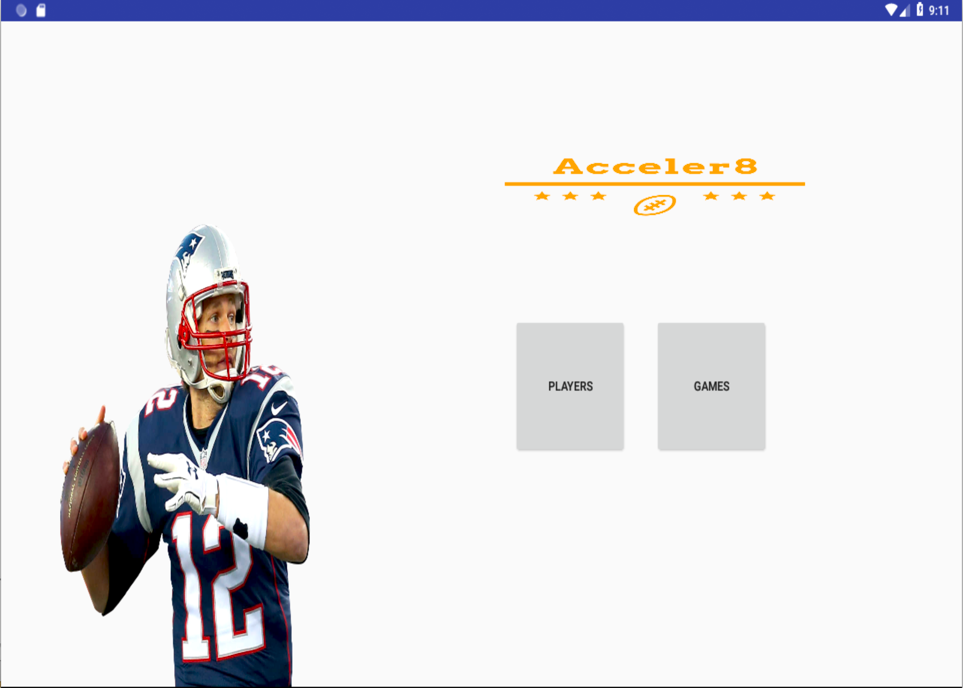
The admin will be able to:

* View the players section which allows to view all the players including their statistics, add new players, remove players and assign a registration code to players.
* Coaches section which will allow the admin to add new coaches and remove coaches from the team and assign a registration code to coaches
* Games section where the admin can add games to be played and then be giving the option to collect statistics in game. In addition, the admin will be able to view an in-depth analysis of all the games
* User section where they will be able view all registered user of the system; change their level of access; change user names and passwords and retrieve them according to a user’s name

****

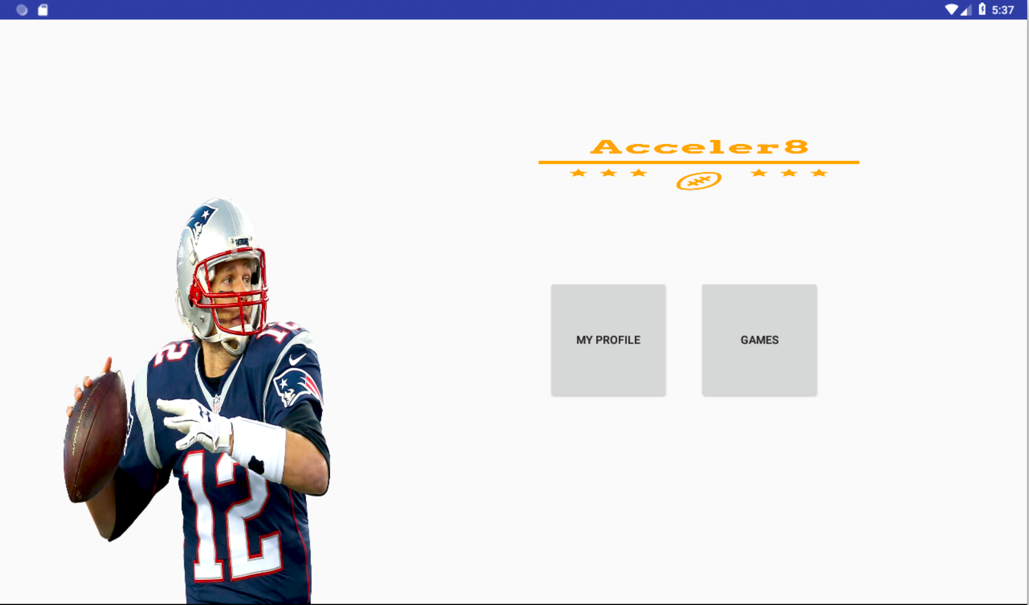
A coach will be able to:

* View the players section which allows to view all the players including their statistics,
* will be able to view an in-depth analysis of all the games in the games section



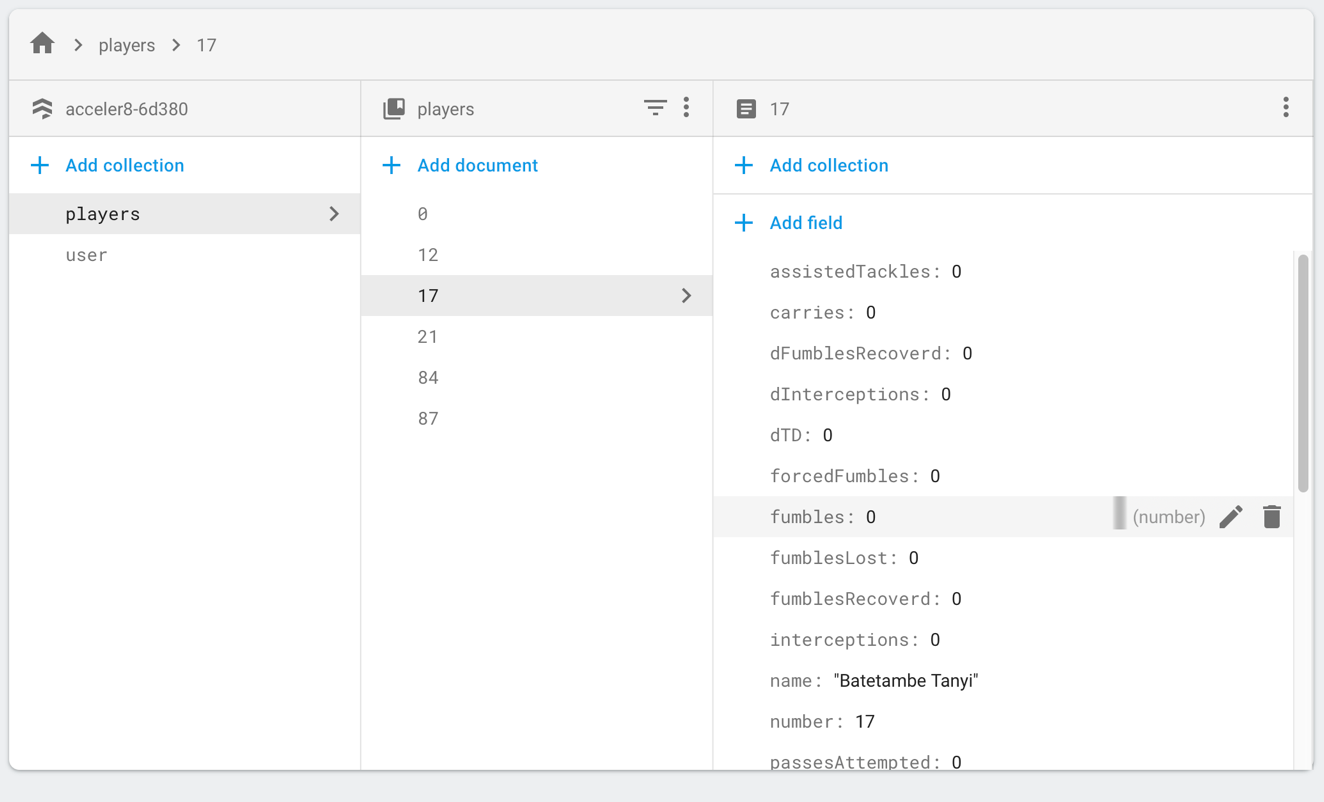
A player will be able to:

* Go to the My profile section and view an in-depth analysis of his season
* will be able to view an in-depth analysis of all the games in the games section



## 6.4 Structure, performance and scalability

As Firebase Firestore database stores data in a similar fashion to json objects so to create a table you will be required to create a collection of documents. The players table essentially is a collection of player object documents.



Scalability, operating constraints and performance have been an important consideration in the design of this system. If this application were to exist in industry the backend database would be placed under heavy stress during high usage periods such as Sunday afternoons where the majority of games will be happening. Multiple teams’ players and coaches will be requesting team and layer data and new data will be created in the process of taking down statistics mid game.

An activity is a screen that the user interacts with and has its own backend logic and UI. They are normally referred to as pages in an app.

**Fragments**

My application will be heavily relying on an Android Studio Java Class called Fragments which can be described as windows inside a certain page of an application. These windows are held inside a container in the activity. The windows can be interchanged on based on a constraint or the completion of an action. Each of these windows has a different function.

Current page

Current fragment being displayed

**Fragments that can be readily interchanged with the current fragment being displayed**

The reason behind the use of fragments in android application development is to improve performance and reduce the need of memory of the application. This is because instead of creating a new page with similar content Only the part of the page with changing content should be loaded.

This in terms of the application will allow a more responsive GUI.

**Intents**

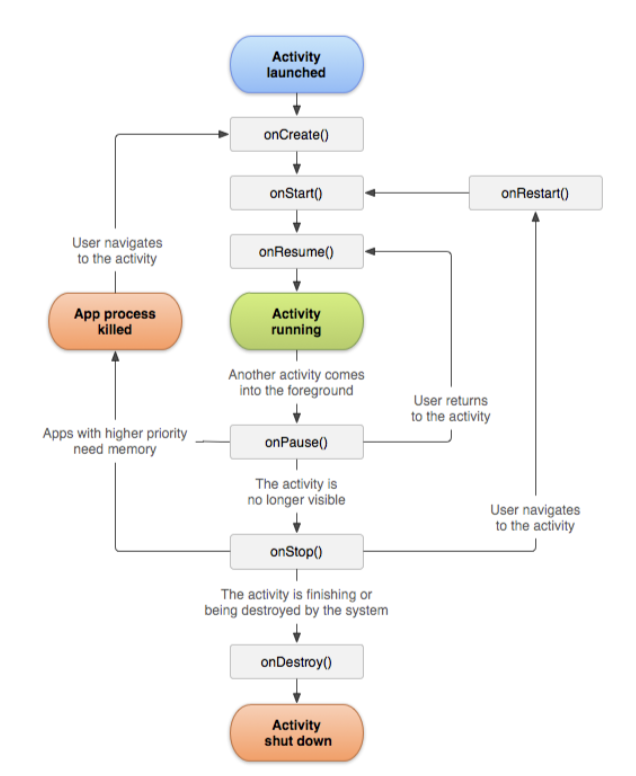
An Intent is a simple message object that is used to communicate between [android](https://acadgild.com/web-development/android-development-training-certification) components such as activities, content providers, broadcast receivers and services. Intents are also used to transfer data between activities.

Intents are used generally for starting a new activity using startActivity().

## Use of Intent

1. For Launching an Activity e.g going from the login activity(page) to the main menu activity(page)
2. To start a New Service
3. For Broadcasting Messages
4. To Display a list of contacts in ListView

**Activity-lifecycle concept**

(developer.android.com)

For the application to function correctly the activity-lifecycle has to be used correctly to allow the individual activities to function correctly. If any database data needs to be displayed on start-up. The function that loads that data should belong in the OnCreate() section of the activity. On the other hand, it is important that only the required functions are placed into the on create section as having too much data to be processed in the OnCreate() section could lead to the activity being throttled.

**Firebase and Denormalization**

‘’Firebase essentially has two ways to query for data: by path and by priority. This is more limited than SQL, and there's a very good reason for that — our API is carefully designed to only allow operations we can guarantee to be fast. Firebase is a real-time and scalable backend, and we want to enable you to build great apps that can serve millions of users without compromising on responsiveness.’’ (developer.android.com)



The above example is how data would be structure in an asynchronous database in an SQL world only one table would be needed so the duplicating of data is counter-intuitive to many developers. However, in order to build scalable applications optimizing data reads is a must. Consider that disk space is cheap but a user’s time is not.

## 6.5 Players Section

**Database Structure**

**Players: 12: passesAttempted : 2**

**passesCompleted: 1**

**….**

**13: passesAttempted: 2**

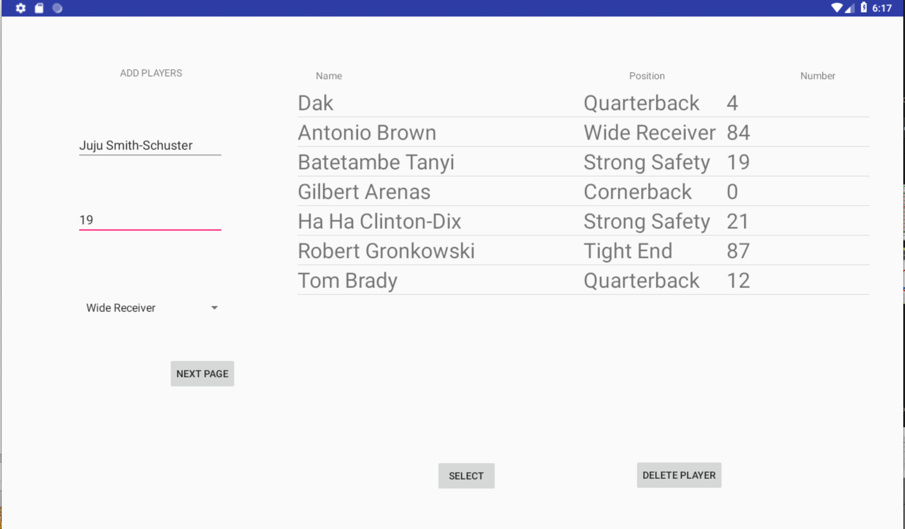
**passesCompleted: 1**

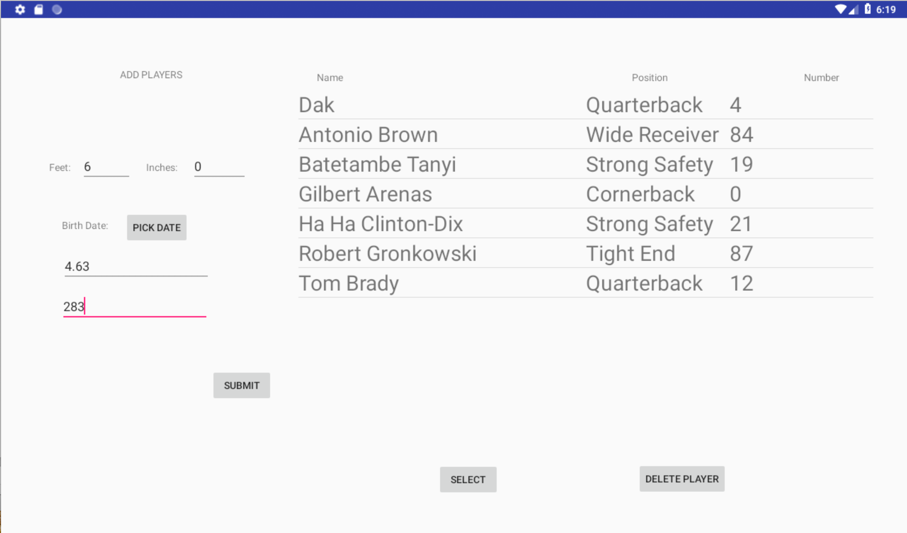
**….**

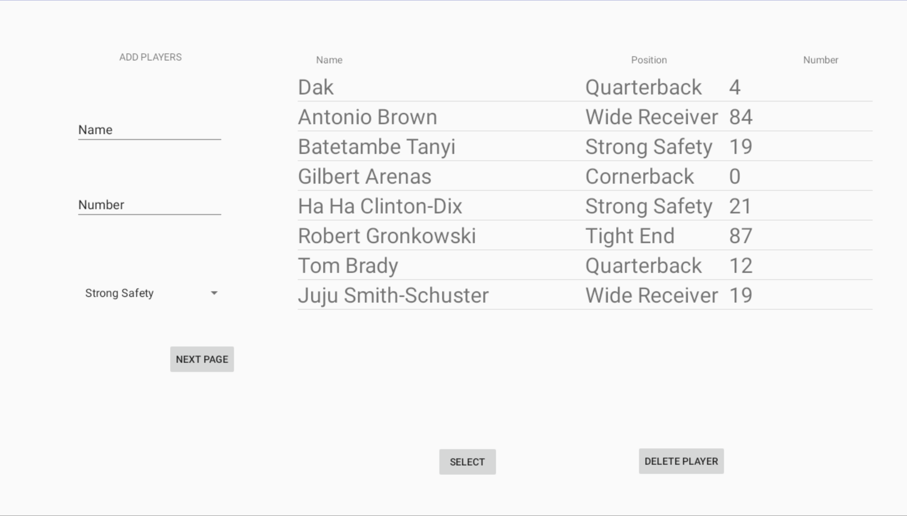
Players is the name of the collection and the players jersey number is the document id

An overview of each player is shown in this activity with the following fields:

* The players name
* The players position
* The players jersey number



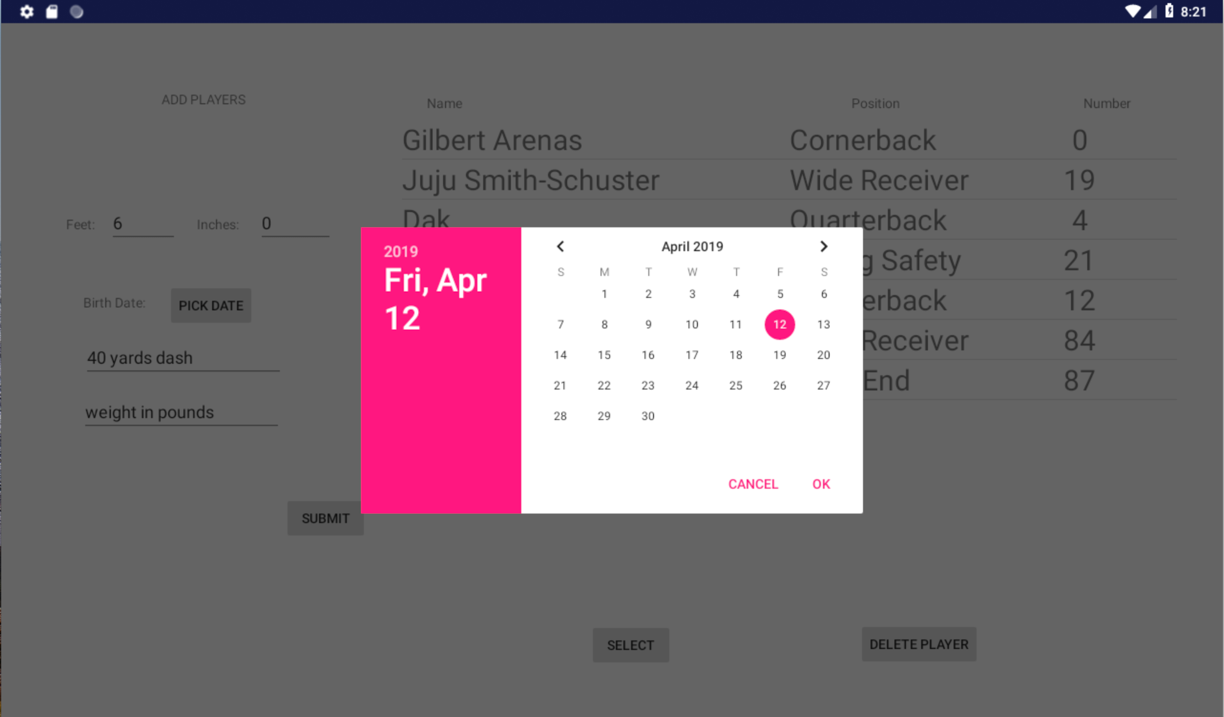


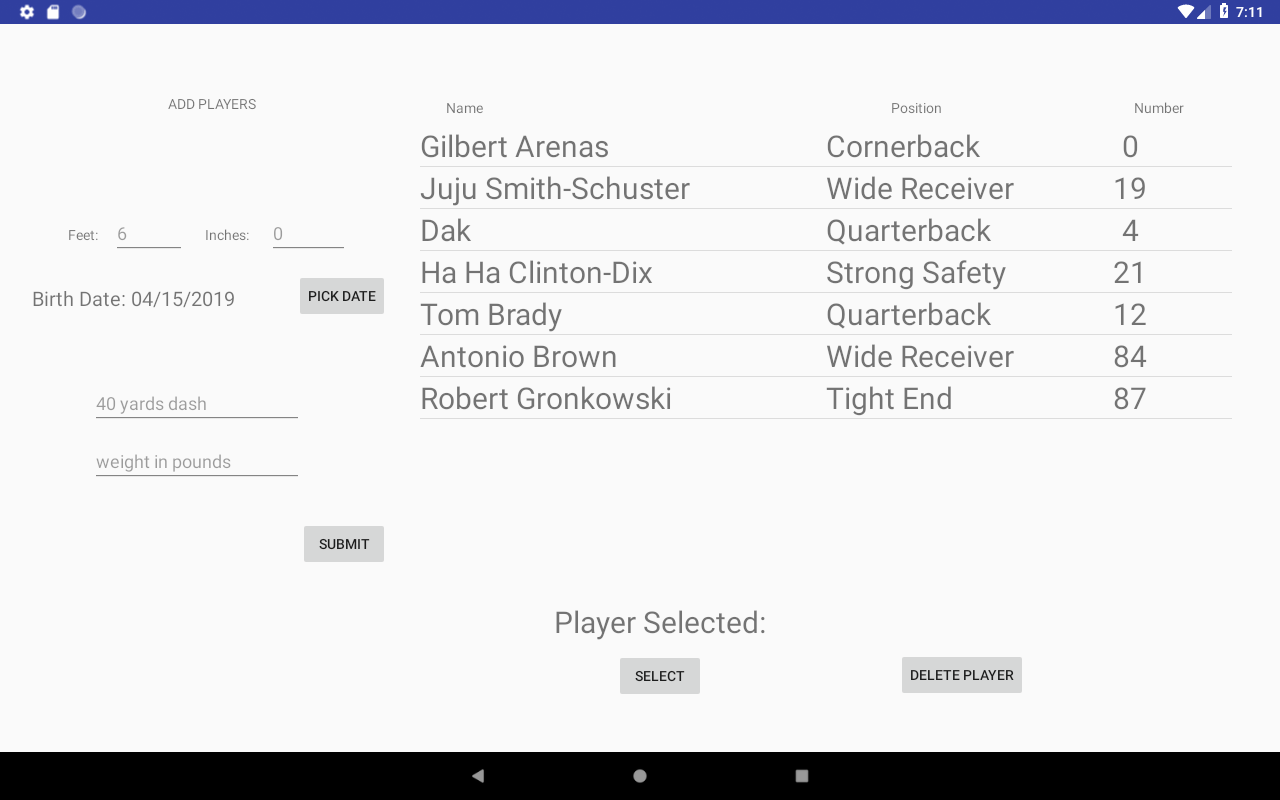


The highlighted boxes are the fragments which allow the user to add another player to the database.

### 6.5.1 Add players

The first fragment and second fragments Tet fields data are stored in variables which will be later then used in setter methods to create a player object.





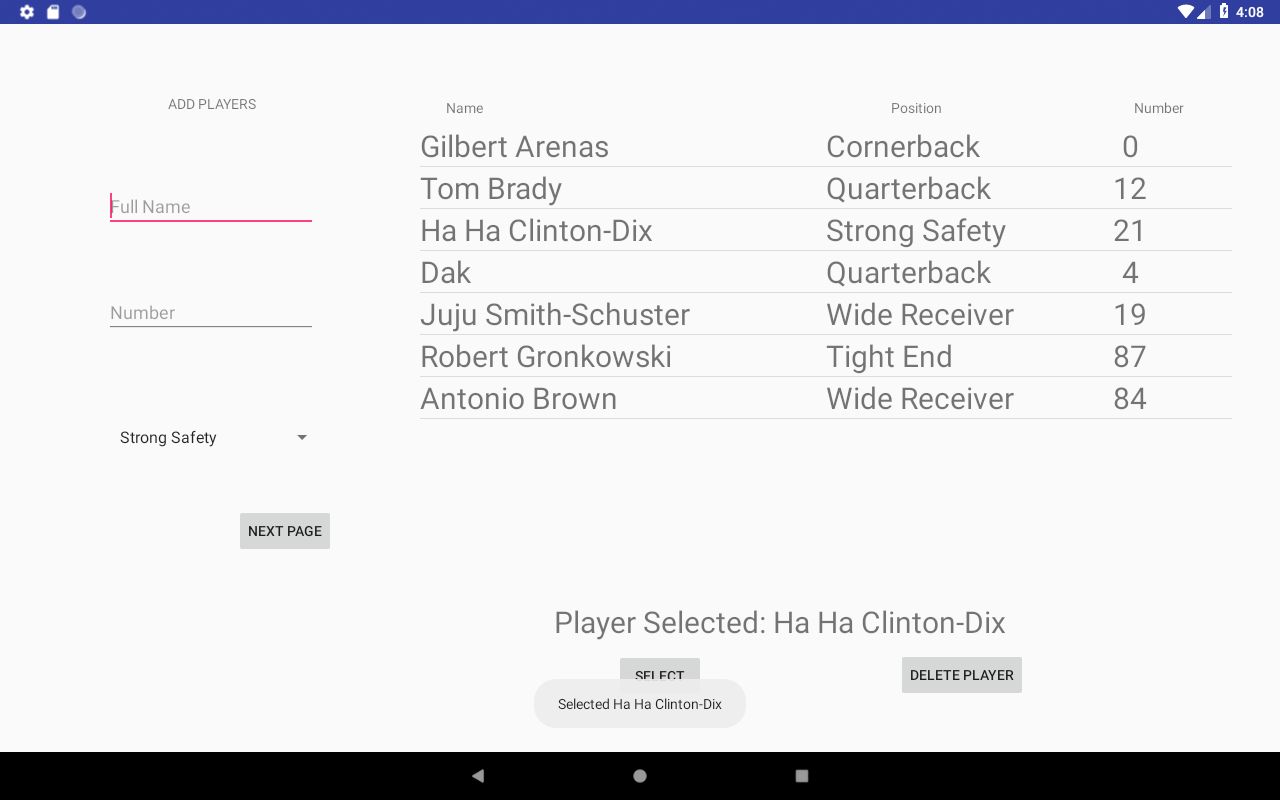
GUI is used to select the players date of birth as there are multiple date formats which can cause ambiguity. Data can only be entered in one format. The month, day and year and then stored in variables again.

### 6.5.2 Retrieve players

*db*.collection(**"players"**)  
 .get()  
 .addOnCompleteListener(**new** OnCompleteListener<QuerySnapshot>() {  
 @Override  
 **public void** onComplete(@NonNull Task<QuerySnapshot> task) {  
 **if** (task.isSuccessful()) {  
 **for** (QueryDocumentSnapshot document : task.getResult()) {  
 Player upload = document.toObject(Player.**class**);  
 HashMap<String, String> hashmap4 = **new** HashMap<String, String>();  
 hashmap4.put(***FIRST\_COLUMN***, upload.getName());  
 hashmap4.put(***SECOND\_COLUMN***, upload.getPosition());  
 hashmap4.put(***THIRD\_COLUMN***, Integer.*toString*(upload.getNumber()));  
 hashmap4.put(***FOURTH\_COLUMN***, Integer.*toString*(upload.getUniqueID()));  
 *list*.add(hashmap4);  
 }  
 } **else** {  
  
 }  
 }  
 });

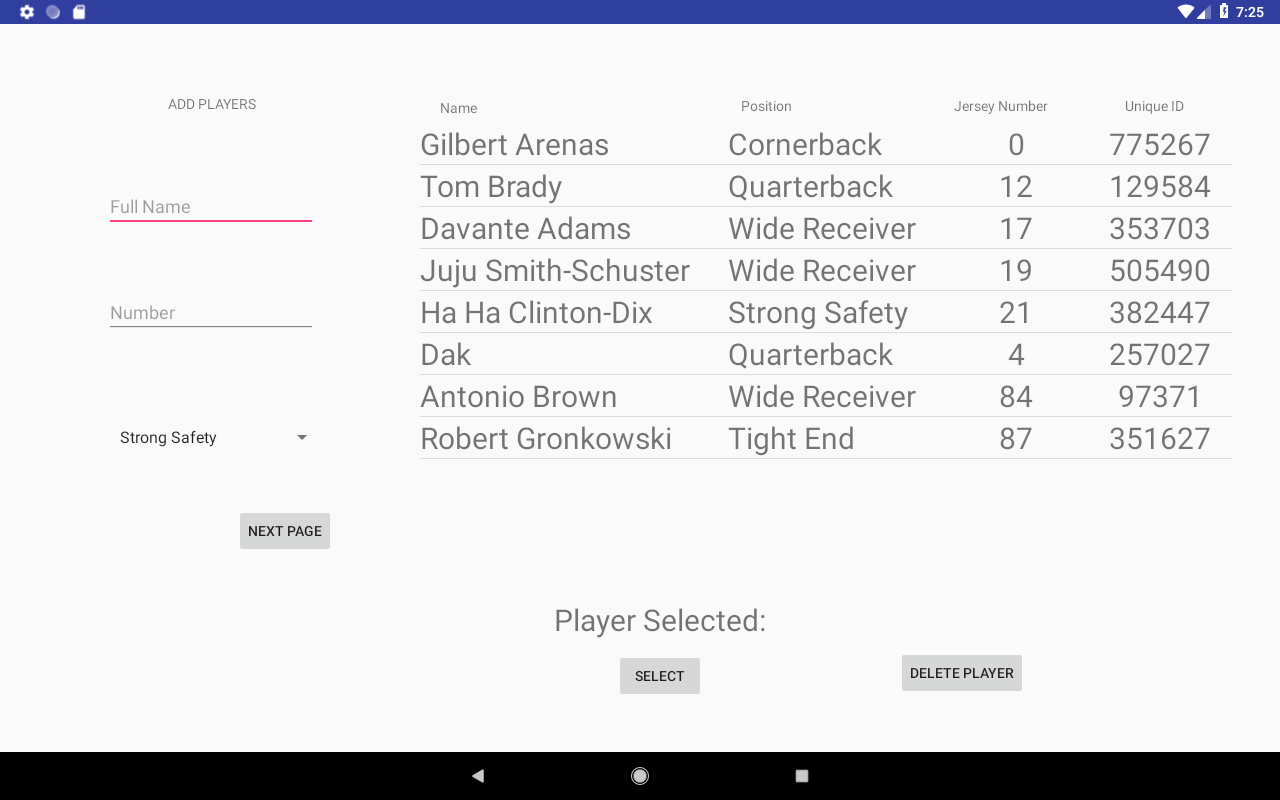
When the players page is launched the page will make a request for all players in the players which has been implemented using a for each loop which iterates through all players in players table and then processes each player one by one by converting the players data into a player object. Then getter methods are applied to the object to get the players name, position and number which are then added to the list of players. The format of the list was generated using an adapter. The adapters responsibility is to create a new row item and each row item contains three text fields.

### 6.5.3 Select a player

****

To be able to select a player you will have to click on the row of the player you want to select and then a little pop up on the screen is produced to notify the user of which layer they have just selected. The name of the player is in the first textbox in each row so the text of the textbox is requested and then stored in a String object.

The player will now have the option to either delete the player from the database or view the players statistics; So that a user can keep sight of a player they have selected



**UniqueID**

**int uniqueID** = **new** Random().nextInt(999999);

Every player has a unique id generated when they are added to the system. This unique id is generated randomly it is required for that player to be able to register to the system.

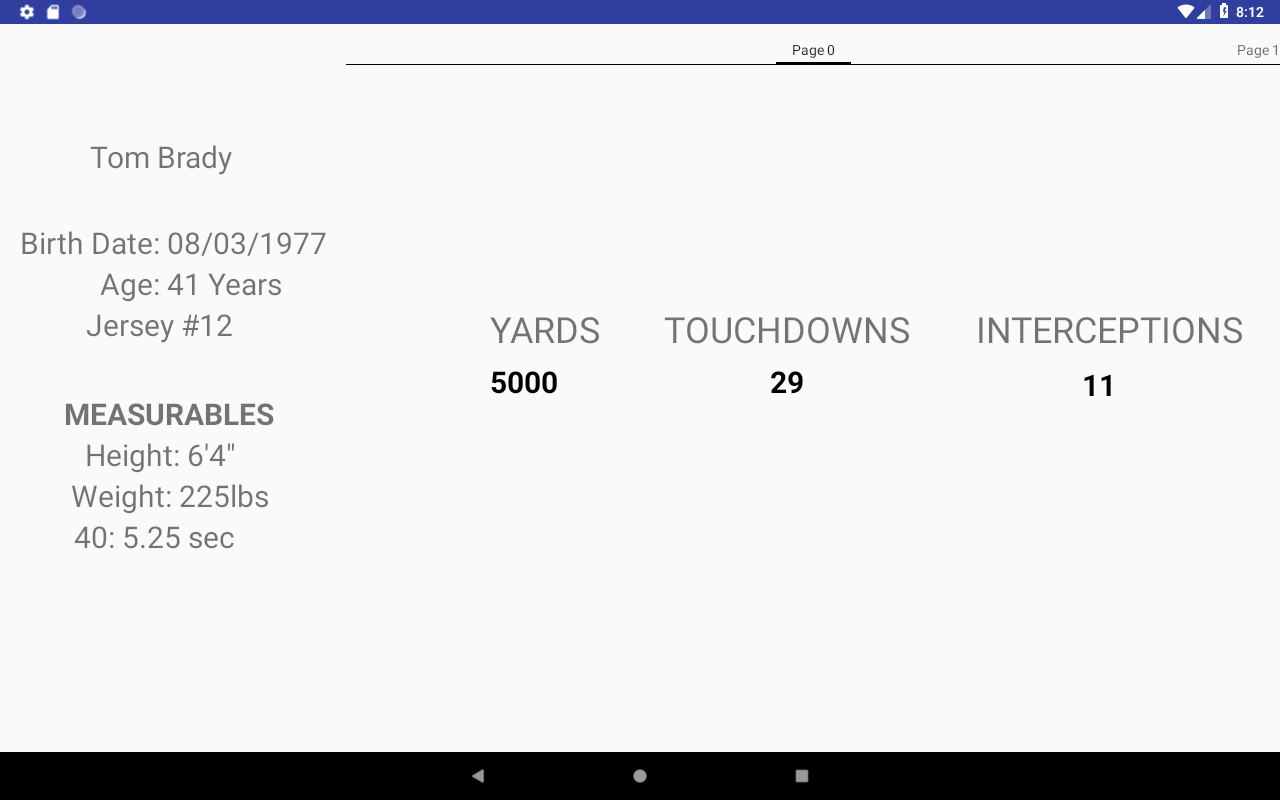
The reasoning behind using a unique ID is that only members of the team can actually use the system because in the scenario a player or coach from an opposing team was able to register and use the system it would give the opposing team an advantage.

**Choice in Design**

* Using a scrollable list to show all the layers in the team allows the user to easily select a player from all the player in the team
* Adding a player and selecting a player being on the same page as viewing all the players reduces the amount of navigation a user has to do find a function

## 6.6 Player Statistics

Once a player has been selected a new activity will be started which will be the player statistics page. The activity is started using an intent with the selected players number stored in the intent.

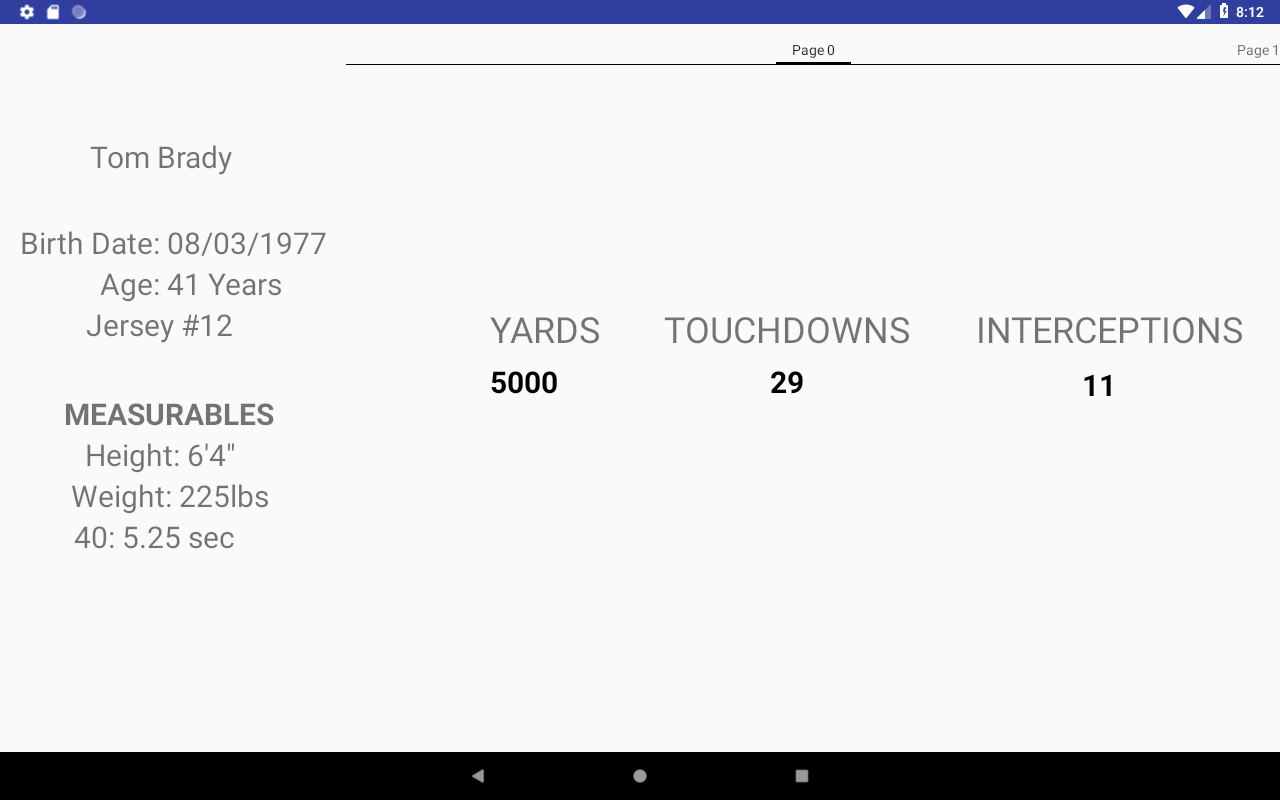


On the left-hand side of the page you can look at the players profile with the details being loaded from the players table in the database. The players number is extracted from the intent on activity tart which allows the correct player to be found in the database. On the right-hand side there is container which will be holding 4 different fragments. Fragments can be switched by swipe action. With the

The player statistics activity allows the user to slide between four different fragments which will be displaying a player’s stats in different formats and different level of depths.

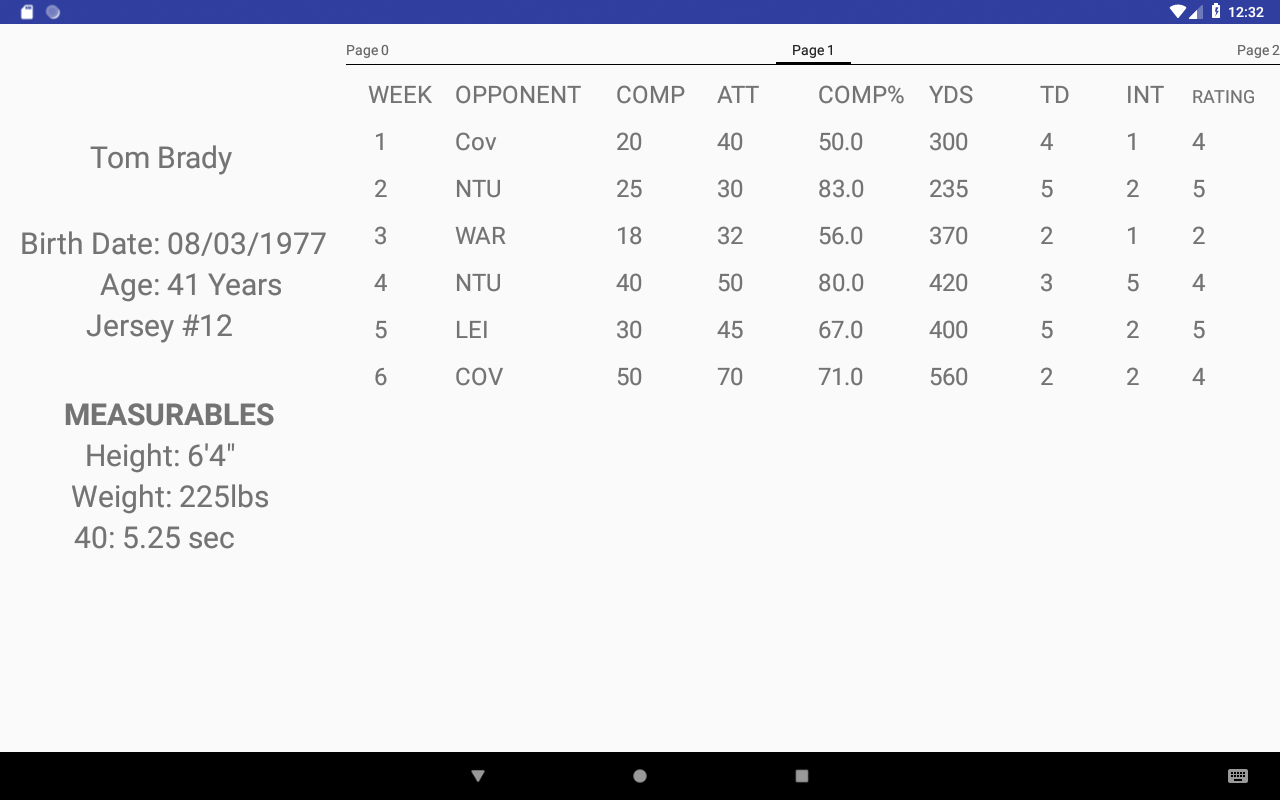
1. The first fragment will be displaying a selection of a player statistics for the whole season stats which are relevant to their position
2. A selection of statistics of all their games played
3. All of a player’s total season statistics.
4. A graphical interpretation of how the player performed week by week

### 6.6.1 Page 0 Season Summary



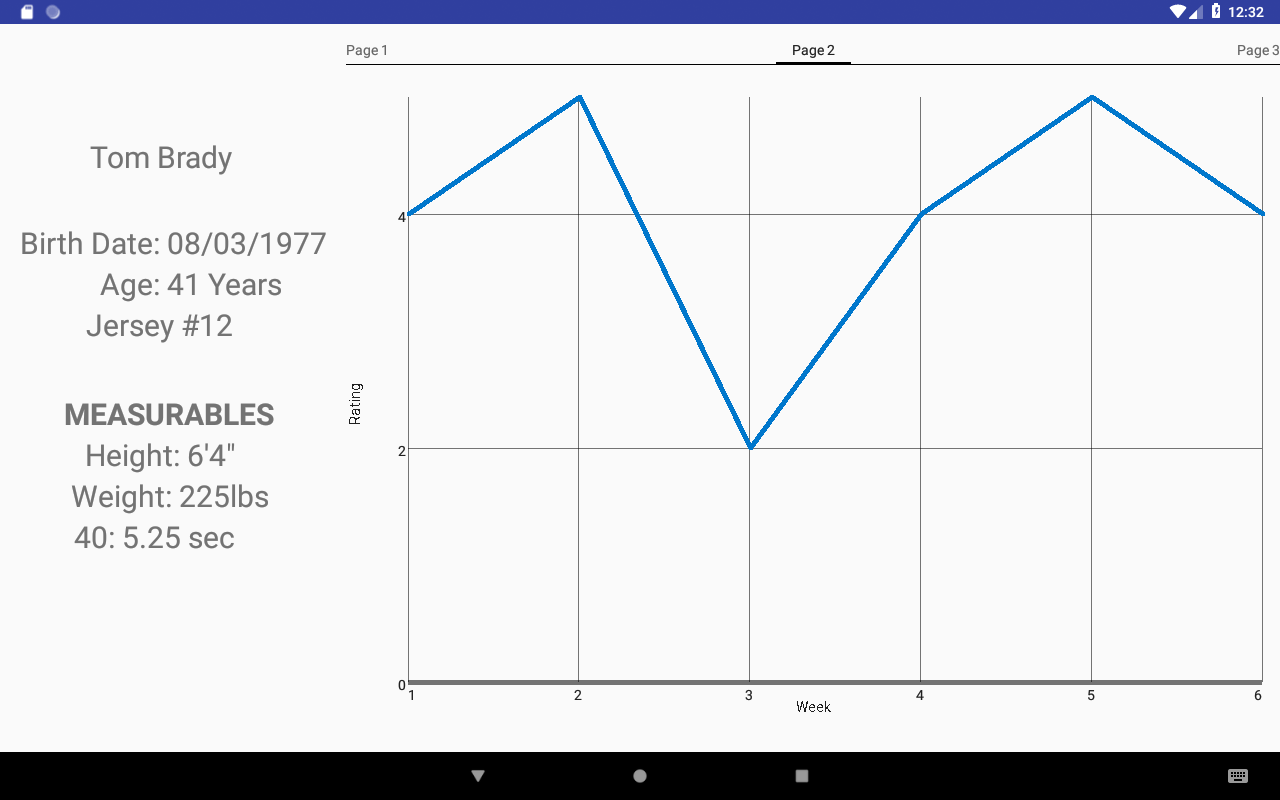
A summary of a player season by displaying a selection of a player’s statistics over a whole season which are believed to be their most important for their position. Page 0 will display the statistics based on the player position. The player position is obtained together with the players detail for their profile. This was implemented by creating a function which loads the required data from the database and placing it in the

### 6.6.2 Page 1 Week-by-Week Analysis



The second fragment displays a week by week of payer statistics with the most important statistics being displayed. The statistics being displayed are relevant to their position so will change if a player of a different position is selected. The table has been implemented by iterating through all the document which holds the data for the player for each game week and then converting the players data into a new type of ‘Player’ Object which extends the original player object but contains the opponent and the rating of the player performance for that game which is assigned by the coach.

### 6.6.3 Page 2 Graphical view of players rating



The third fragment is a graphical display of the players rating over the weeks which allows players and coaches to recognise trends in a player’s seasonal performance

* Coaches can get an understanding of which players perform well against certain teams so that they can adjust the roster and stating line up accordingly
* Recognise if a player is consistently underperforming so a coach can possibly introduce extra training for that player.

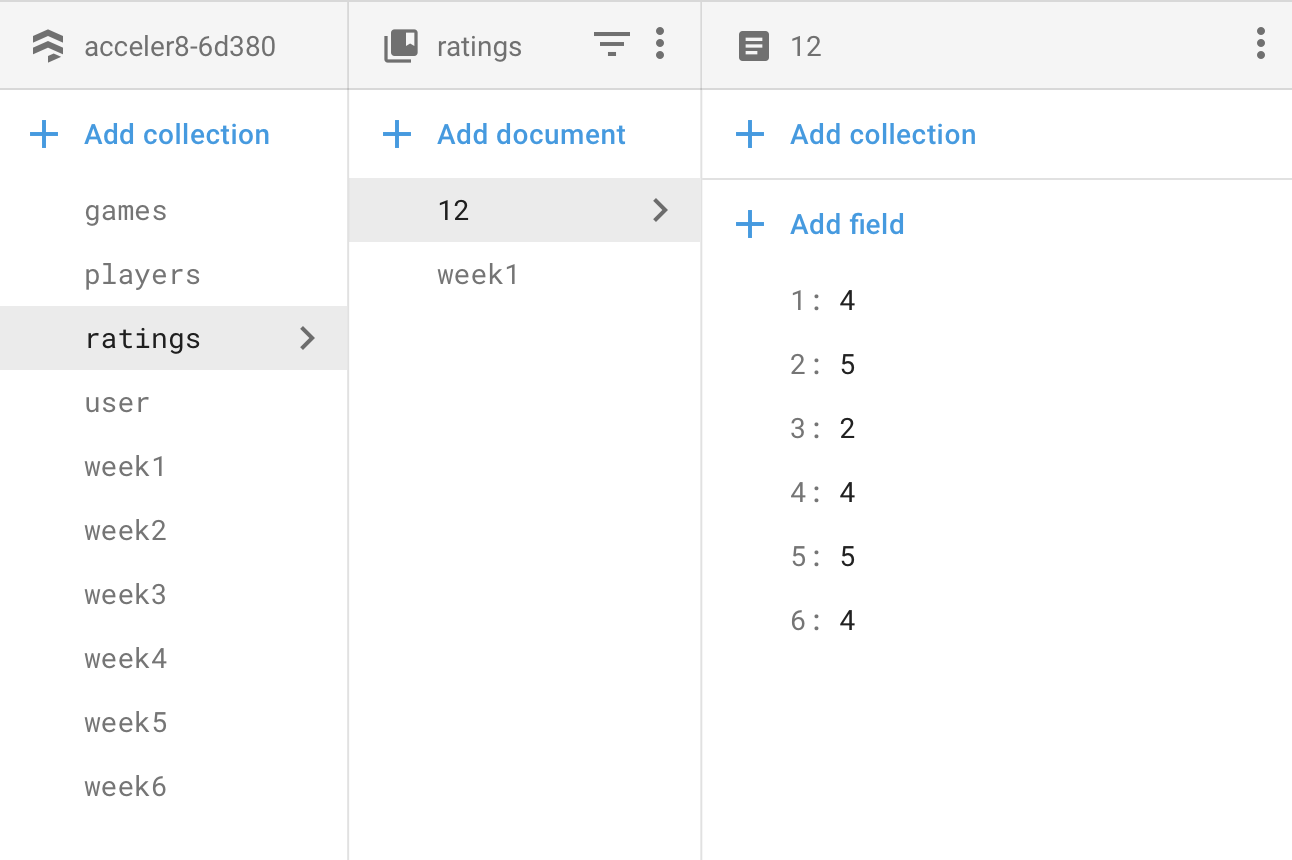
This was implemented using a plug-in called GraphView. As Firebase is synchronous and the plugin being relatively unstable it did cause some errors such as the graph just being blank. For the plugin GraphView to be able to work with the asynchronous nature of Firebase, data in the database had to be denormalized. Firebase cannot work with multiple collections of documents so a collection just containing the players rating had to be created with a document containing the rating of a player week by week. The documents id is the players number. With the number of the week being the name of the field.

**series**.appendData(**new** DataPoint(i,y), **false**, **numberofgames**);  
  
GraphView graph = view.findViewById(R.id.***graph***);  
  
graph.addSeries(**series**);

This code allows to add a new point to the series as the graph is a line graph.

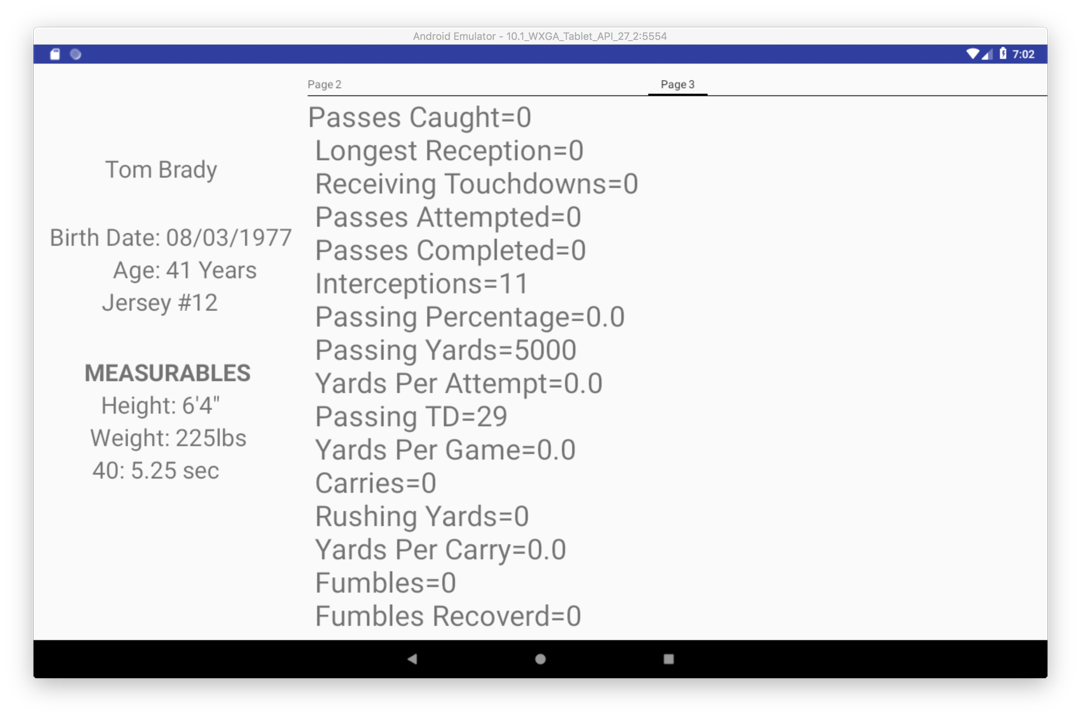
Reasons behind using a graph to present information

* Helps the user grasp the information visually which is seen as more engaging
* Easier to compare data



The document is iterated through to create the x and y coordinates of the graph with the x coordinates being the week and y coordinates being the corresponding rating of the player for that week.

### 6.6.4 Page 3

****

Page 3 the last page displays all players statistics there is no filtering done according to their position. A request for the players document is made from the players document collection. Where all the players data is converted into a player object and then the instance variables are concatenated into one Strings to be displayed. This fragment/page is scrollable to allow the user to scroll through all the players statistics. This fragment is required as it allows coaches to be able to complete analyse player season as in the amateur level of the sport players play multiple positions so the coach can also view the statistics for the secondary position.

## 6.7 Mid Project Changes

Mid-project I realised that the way I structured my database was ineffective and was not compatible with the asynchronous nature of Firebase. In my original structure there was a lot of redundant data.

My original structure

The week number is the collections name and then the players name would be a document and then with fields such as passesAttempted

**Week 1: 12: passesAttempted : 2**

**passesCompleted: 1**

**….**

**13: passesAttempted: 2**

**passesCompleted: 1**

**….**

**Week 2: 12: passesAttempted: 2**

**passesCompleted: 1**

**….**

**13: passesAttempted: 2**

**passesCompleted: 1**

**….**

**Week 3: 12: passesAttempted: 2**

**passesCompleted: 1**

**….**

**13: passesAttempted: 2**

**passesCompleted: 1**

This is my improved structure created using nested objects

Only one collection with the players id (jersey number) being the name of documents and then the players statistics for each week are converted into an object with the object then being nested into to the document.

**Weeks: 12: Week1:**

**passesAttempted:2**

**passesCompleted:1**

**…**

**Week2:**

**passesAttempted:2**

**passesCompleted:1**

**…**

**13: Week1:**

**passesAttempted:2**

**passesCompleted:1**

**…**

**Week2:**

**passesAttempted:2**

**passesCompleted:1**

My new database structure which made use of nested objects in documents is more optimal than the previous database structure as it eliminates the need of having a separated document with a player rating week by week. It functions with the asynchronous nature of firebase. In the original structure to be able to display Page 1 which is the week by week display of statistics of player there you would have to loop all the collections which would cause the weeks to be displayed in an incorrect order in the application. This is because the content inside the loop which are the database listeners run asynchrously meaning it would move on the next iteration of the loop before loading the required document. In the second structure it would load the document first. To populate the table in the application it would make subqueries into the nested objects to populate the table. In addition, it reduces the number of collections making the database easier to manage. The application is futureproofed using the new structure as it allows the application to be used over multiple seasons as for example a new collection called Weeks2020 can be added which would contain all the player data for the 2020 season.

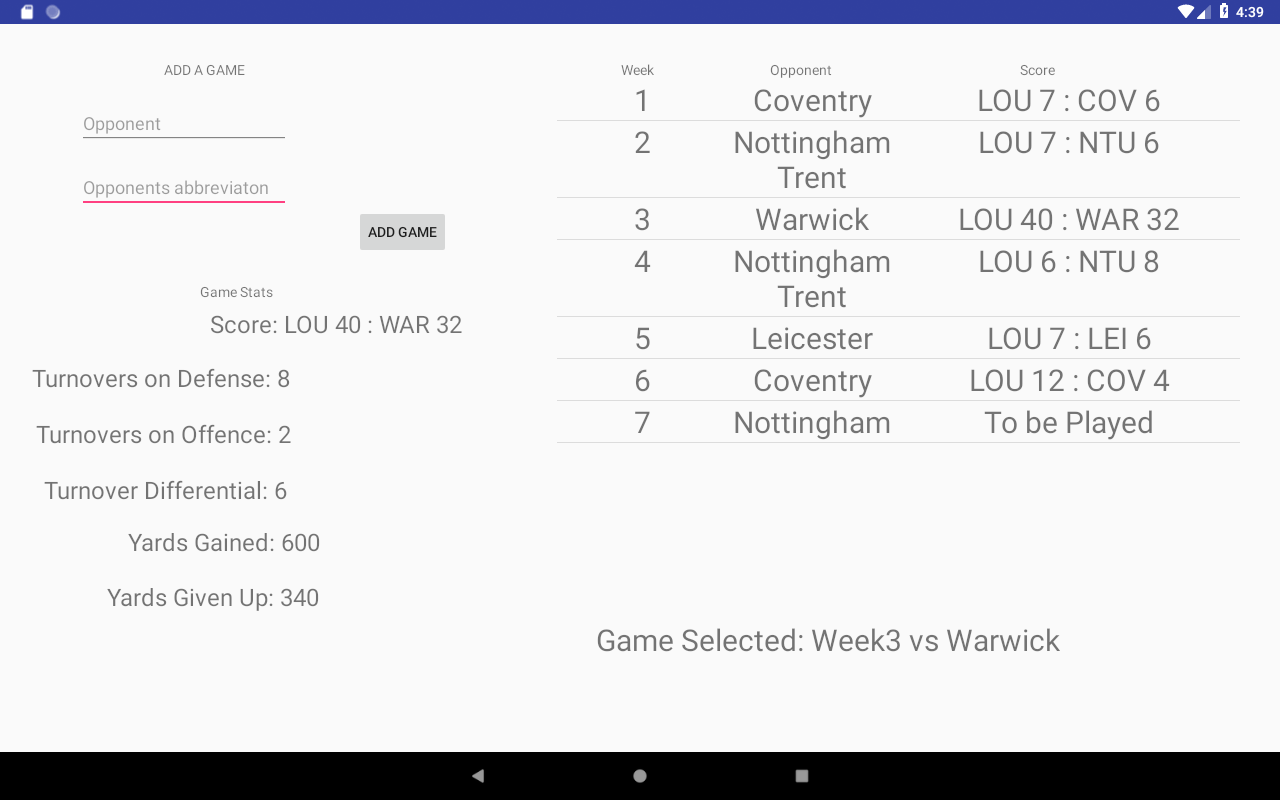
## 6.8 Games Section

**Game Class**

**int defensiveturnovers**, **offensiveturnovers**,**yardsgained**,**yardsgivenup**,**teamscore**,**opponentscore**;  
String **opponent**,**opponentabreviation**;  
**boolean played** = **false**;

Above shown are the instance variable of the game class which will be used to create a new game and fetch the data from the database and transform it into a game object to populate the list of games.

## 6.9 List of Games

****

This is the activity where the user gets to view the performance against a team. When a user selects a game statistic such the score, turnovers etc are displayed in the bottom left. The list of games has been implemented similarly to the Players activity.

**Weeks: Games: Week1:**

**Opponent: Coventry**

**Score: LOU 7:6 COV**

**…**

**Week2:**

**Opponent: Nottingham Trent**

**Score: LOU 7:6 NTU 6**

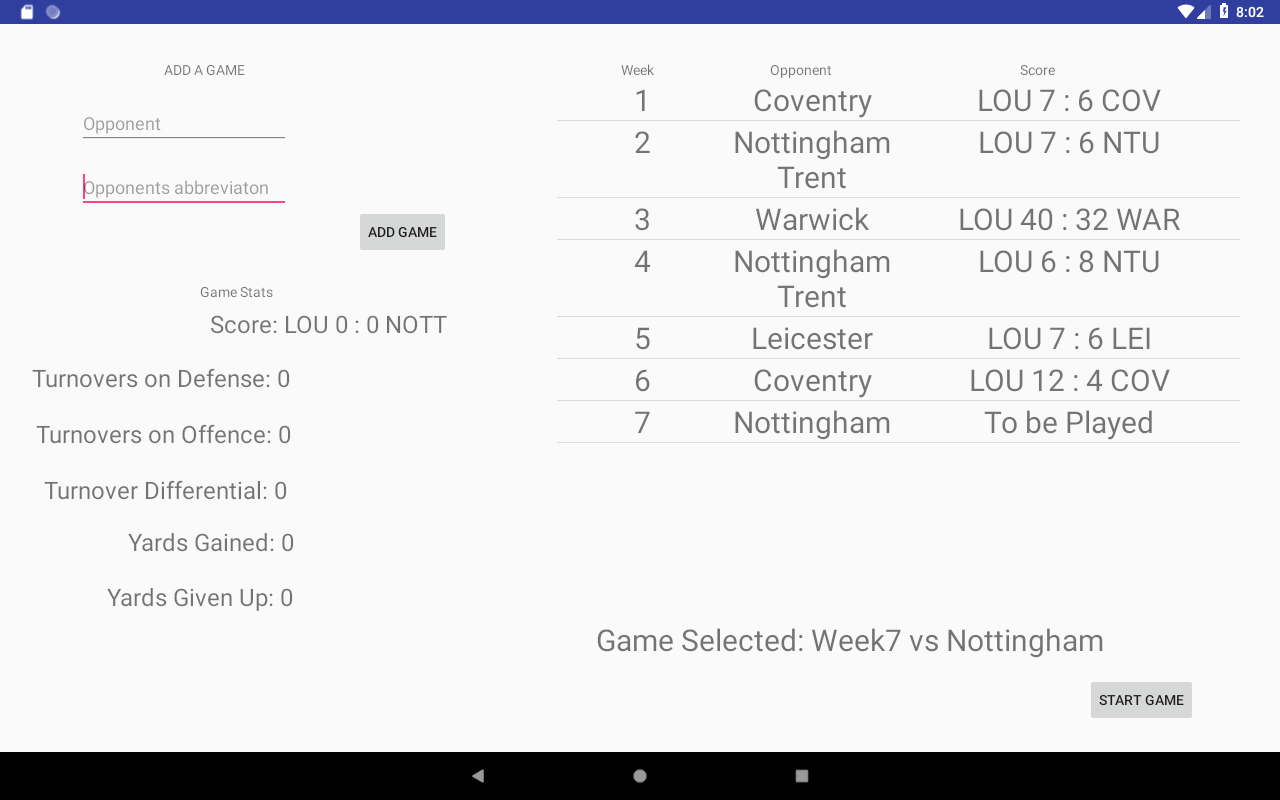
**…**

All the required data to populate the list is found in the Games document in the weeks collection. Once the document is loaded and then is iterated through each week. The data of each week is stored as a nested object so the required data used in the list is simply retrieved using getter functions.

### 6.9.1 Adding a new game

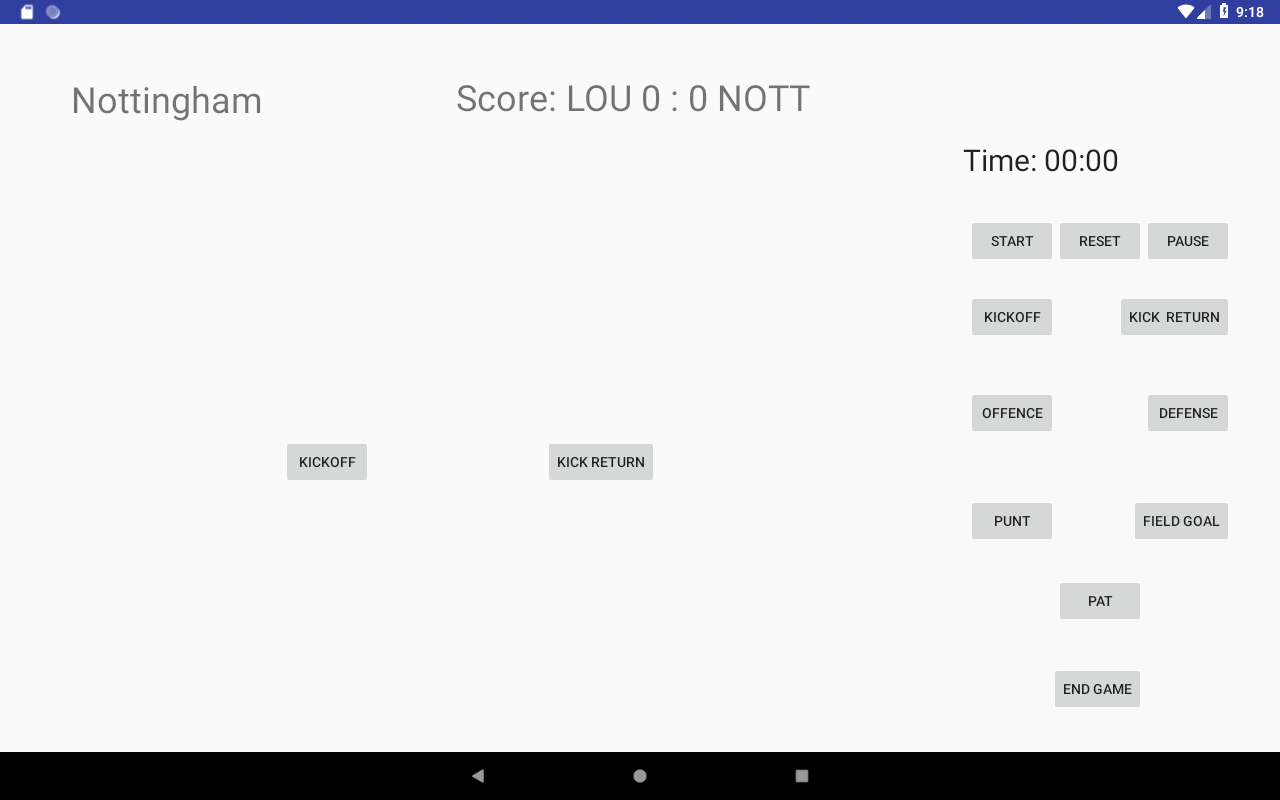
New games can be added by simply entering name of the opponent and the desired abbreviation for the opponent’s name. The week number is automatically assigned by obtaining the total number of games and incrementing that number by +1. As the match has not been played yet in the score column it will display “To be Played”.

### 6.9.2 Starting a game

****

If a game that has not been played is selected the START GAME button will appear which will allow the user to get to the activity which allows them to note down statistics produced in game.

## 6.10 Record Game



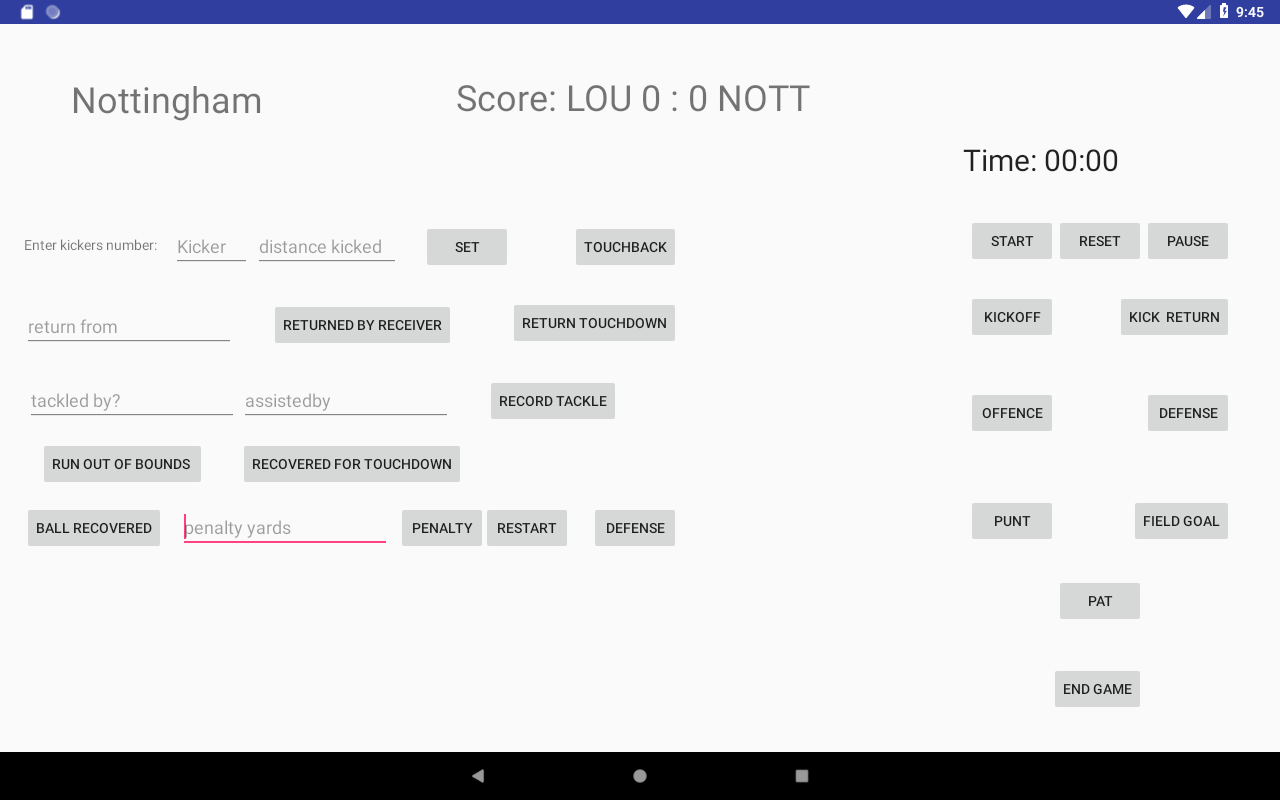
This is the activity which allows you to capture statistics in game. The game of American football has multiple different scenarios. The buttons on the right-hand side which allow the user to switch between game scenarios. In the bottom middle of the screen which has been highlighted will display the different fragments.

To record a players, action his number is entered to a textbox and then the action e.g. complete pass will be a button that is responsible updating the number of passes the player has completed. Once the action has been completed the user is informed of what action has been performed and updated in the database using a small pop up that appears at the bottom of the screen.

Choice in design

* Simplistic so that the user can navigate between fragments in instance so that person who is the script will not miss a play.
* Timer is there for game management as being aware of how long a game has been played for is important
* Score is in the top middle so that score is easily noticeable.

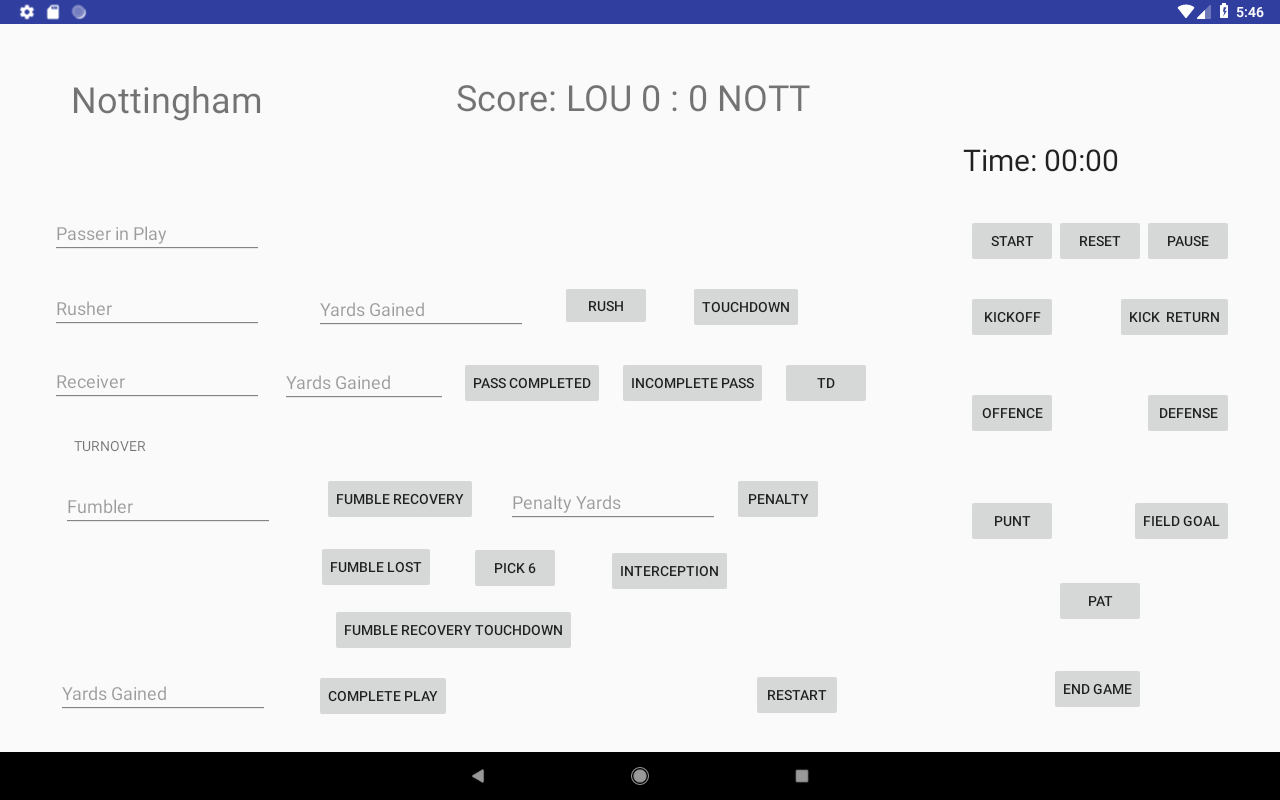
### 6.10.1 Kick off Fragment

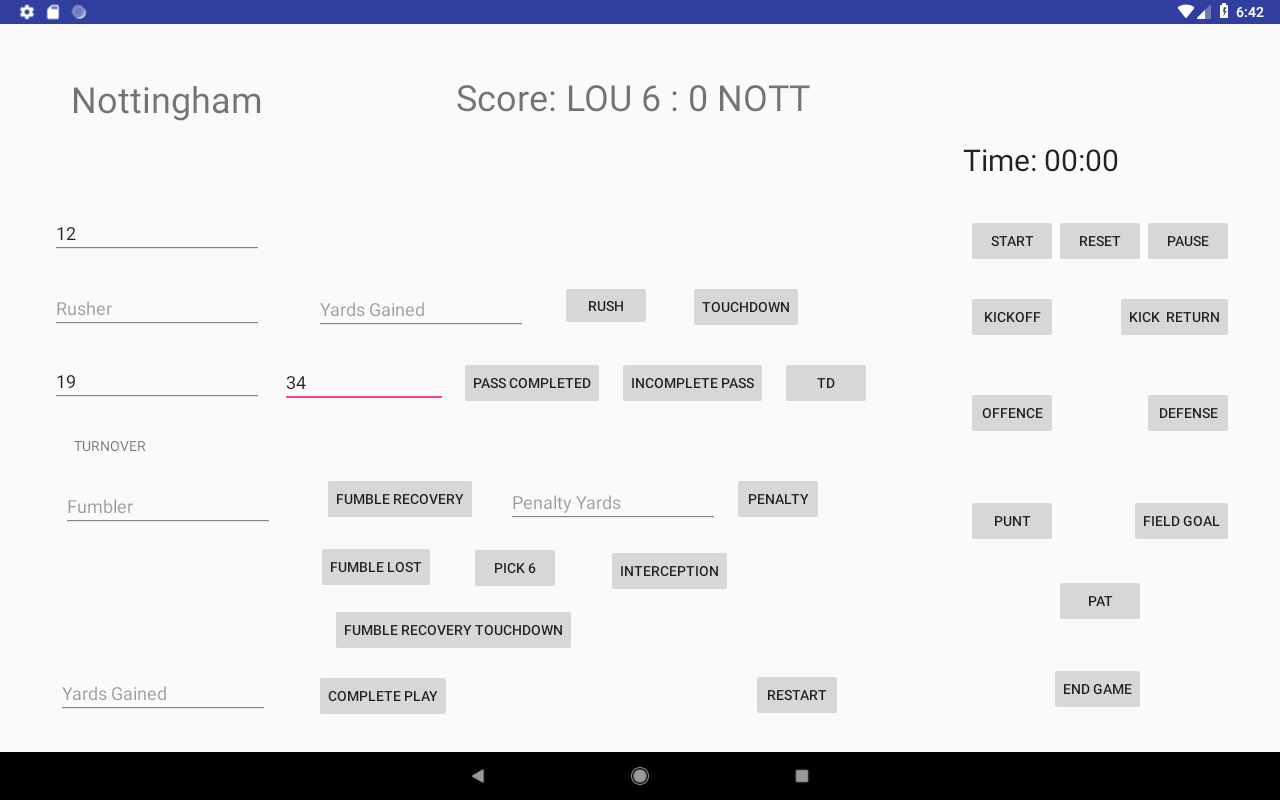
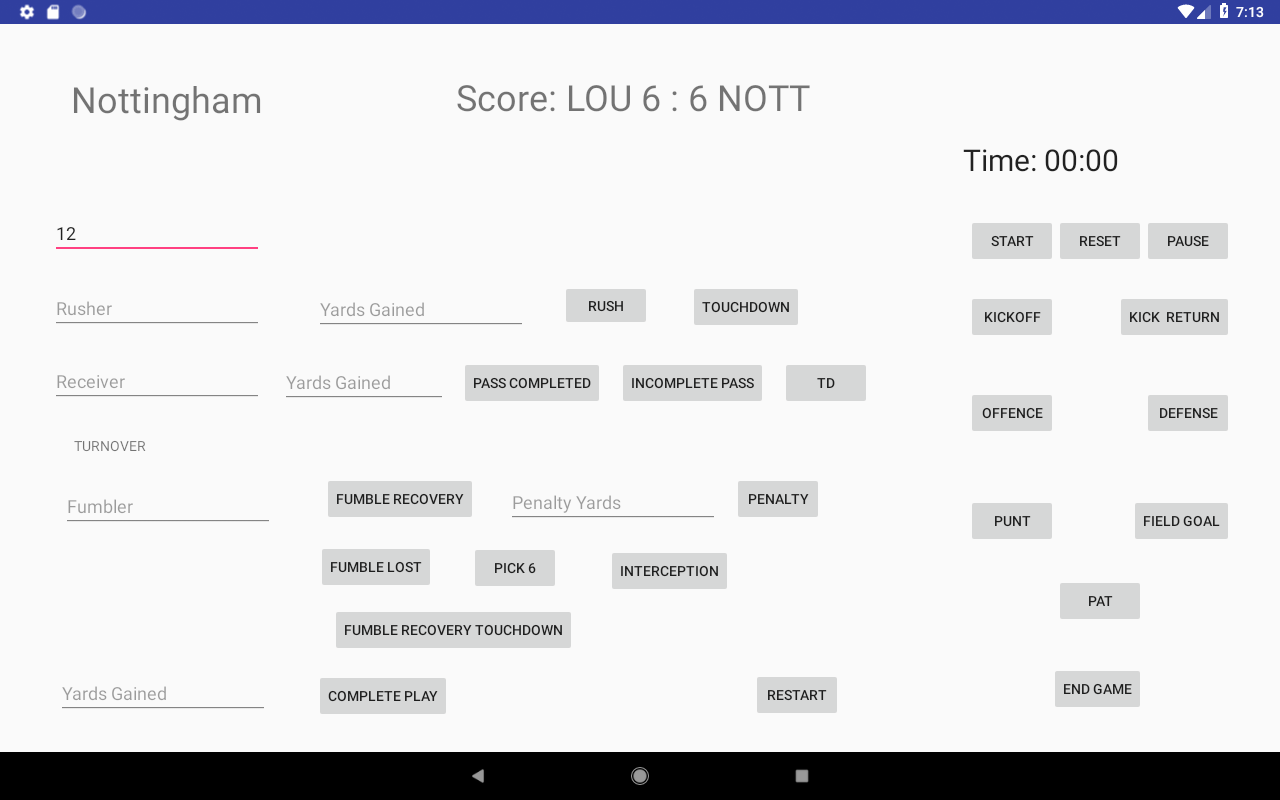
****

This one of the fragment which is used in the scenario of there being a Kick-off.

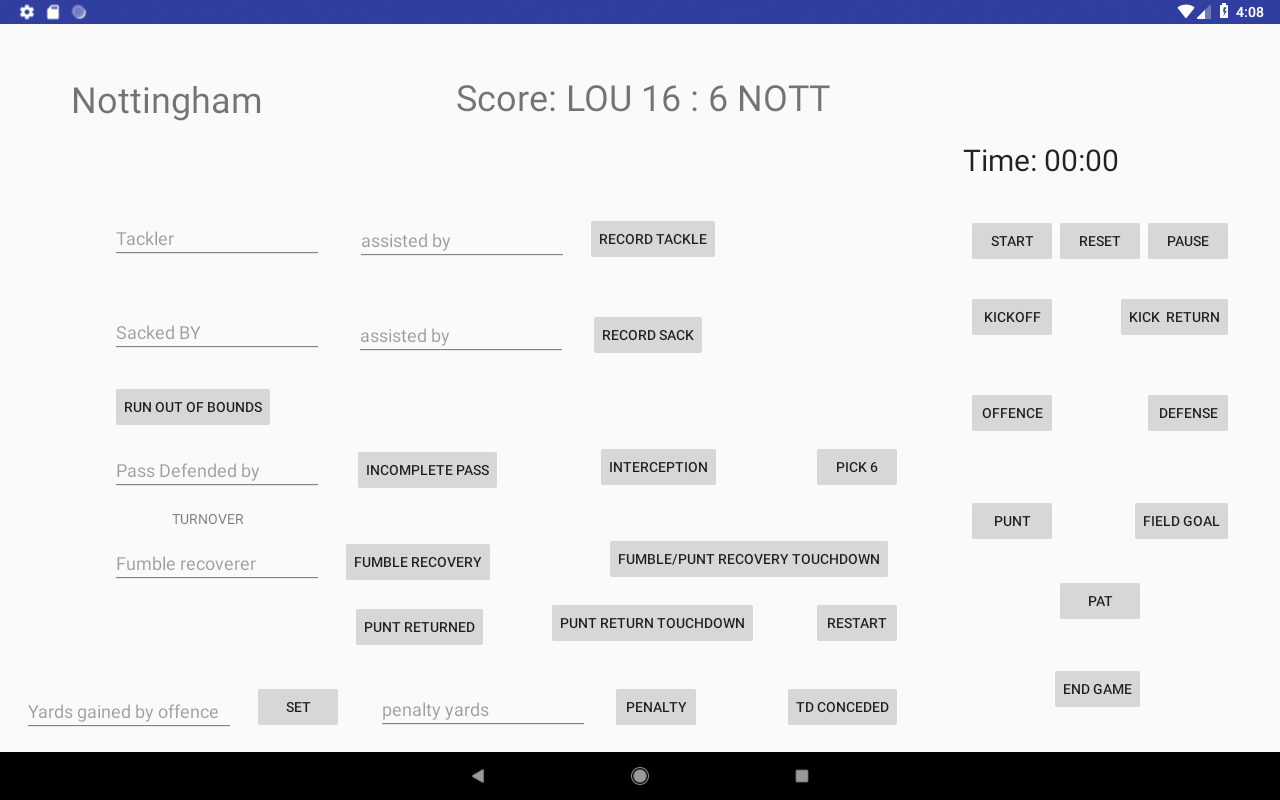
1. The kickers number is entered into the kicker EditText which is basically a textbox. The input type has been limited to just numbers so the numerical keyboard will appear. The function of the set Button is to be able to set the longest Distance a kicker has kicked over a game and season. It checks the original value and if the value entered in the distance kicked EditText is bigger the longest distance kicked will be updated with the value entered.
2. By the rules of American Football if a ball is kicked into or out of the end zone it is called a touchback. If the Touchback button is clicked it automatically increments the players Touchbacks for the season and the game
3. A player on the opposing team can return the ball if the kick has not been declared to be a touchback if the opponent player returns the ball into the team’s own end zone it will count as touchdown and the top middle where the score is displayed the score is displayed the score will increase by 6 and will automatically update in the database too.
4. When an opponent tries to return the ball players on the team are allowed to tackle the player. The first person to make the contact will be noted down to be the tackler. In the process of tackling a player can assist his team mate with the tackle. If the record tackle button is clicked on whatever player that has been entered in to the ‘tacked by?’ and ‘assistedby’ textboxes

### 6.10.2 Offence fragment



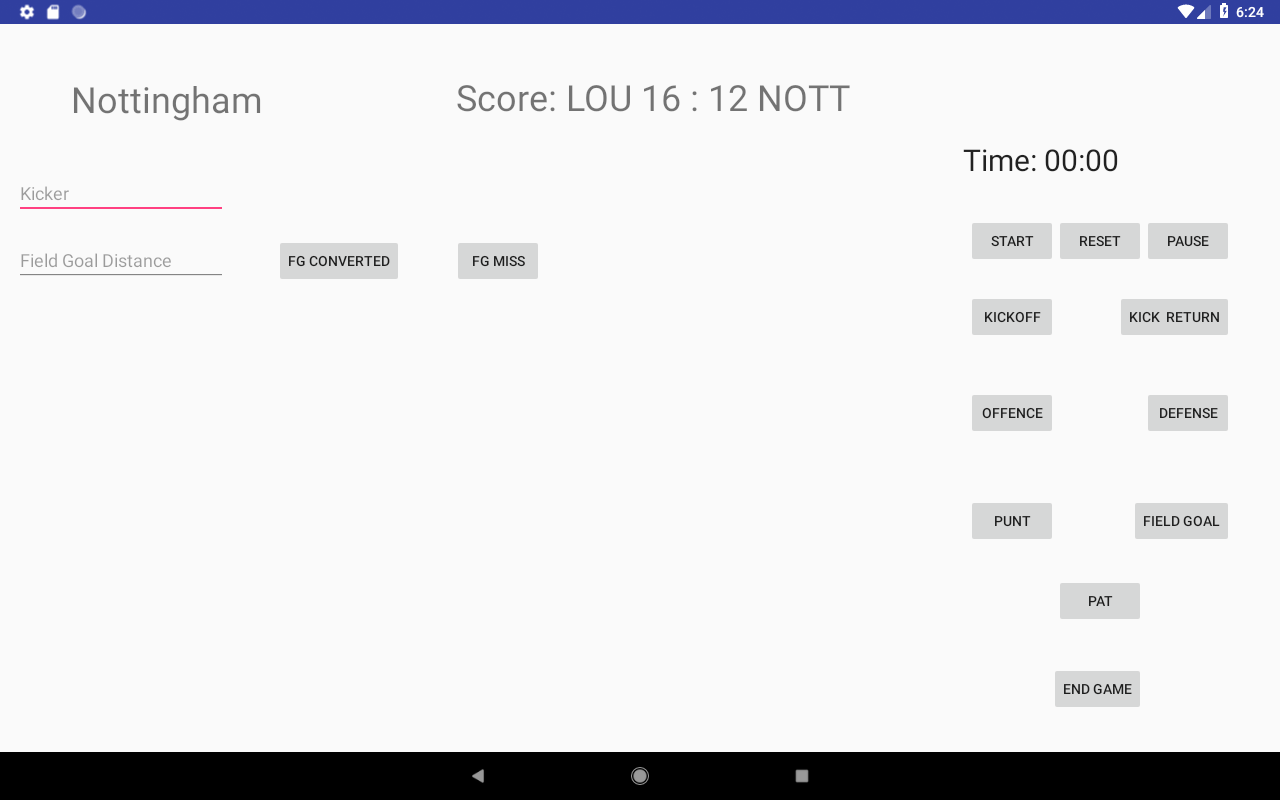
1. When the team is on offence a player can pass the ball forwards to another player for the passers and receiver passes caught and passes completed to be recorded the pass complete button has to be clicked on. If the yards gained through the completion of a pass the longest reception of that receiver for that’s season and game is compared to the value that has been entered. If the value that has been entered is higher than the current value it is replaced with the value entered. If the user clicks on the ‘TD’ button it does the same action as the ‘pass completed’ button but it will increase the score for the user’s team by 6 and increase the number of passing touchdowns of the receiver and passer by 1
2. A player can also be run to record the distance a player run with the ball. The player running with the ball should be entered in the ‘rusher’ textbox and also the distance he gained running with the ball. The data entered will be updated in the database using the ‘rush’ button. The ‘touchdown’ button works exactly same as the ‘td’ button but rushing touchdowns will be incremented by 1 instead.
3. On offence the ball can be turnover an example of a turnover is if the passer in that play throws the ball and it is intercepted by an opponent player. If the opponent player runs the ball into offence end zone without getting tackled the opponents team score will increase by 6. This action can be done using the ‘pick 6’ button

### 6.10.3 Defence

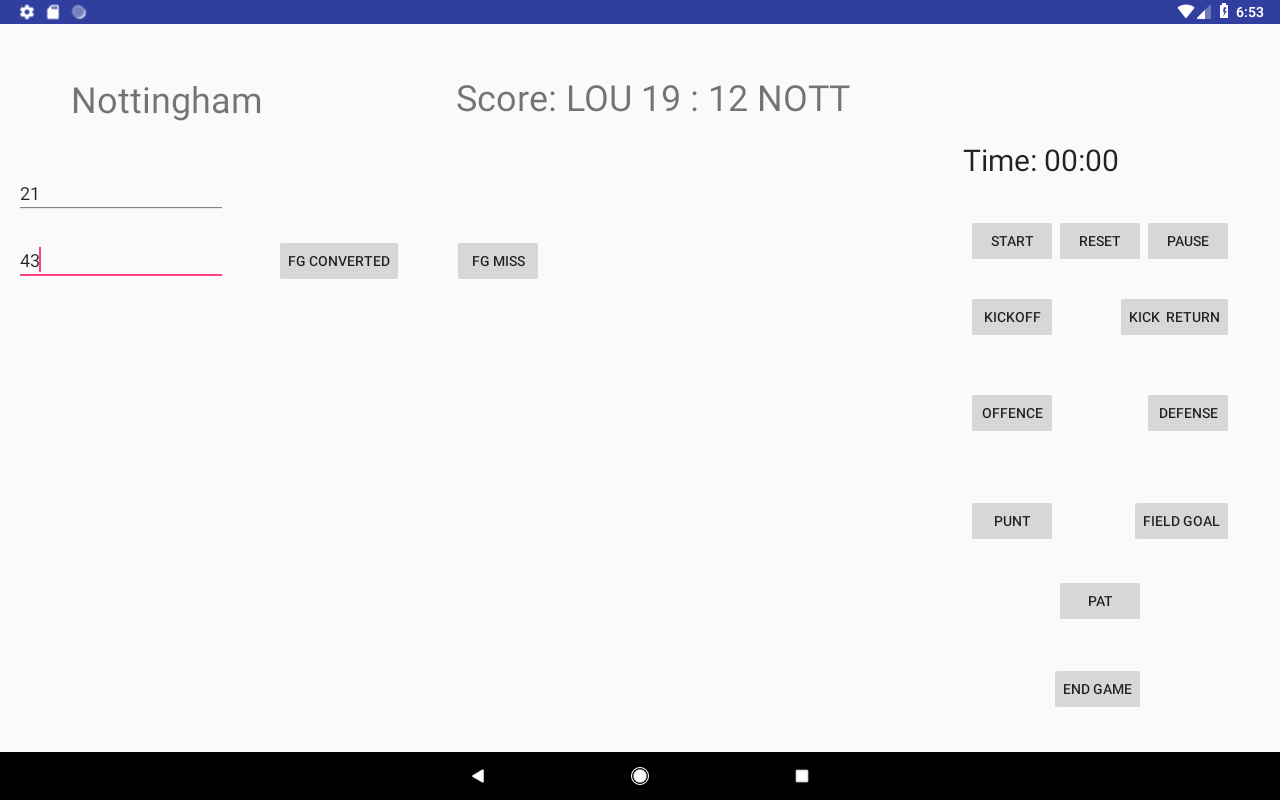


1. Similar to the kick-off fragment where a player can make a tackle so the ‘record tackle’ button will just increment the number of tackles of a player by 1.
2. Another button called ‘record sack’ which in simplified terms is a tackle on the Quarterback on the opponent team will increment the number of sack of player by 1
3. A player on defence can intercept the ball when the ball has been thrown. If the ball is intercepted the player that has been noted down in the textbox with ‘pass defended by’ his number of interceptions for that game and the whole season are incremented by 1. If the user clicks on the ‘pick 6’ button the number of touchdowns and interceptions is incremented by 1 and the users team score is increased by 6
4. If the user clicks on td conceded the opponents score increases by 6
5. The ‘yards gained by offence’ is used to set the number of yards the offence gained on that play. The value entered will increase the number of yards the team has given up in total for that game.

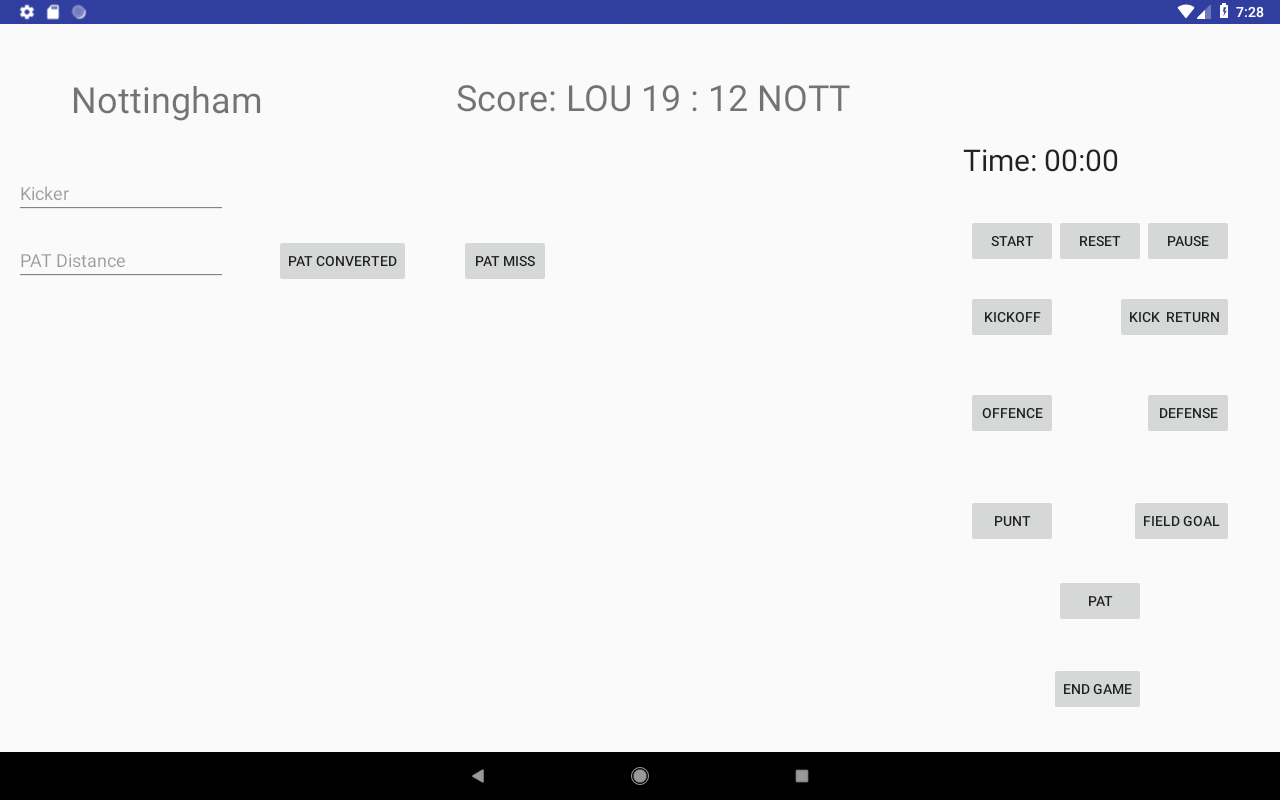
### 6.10.4 Field Goal



1. Team can go for a field goal instead of getting the ball into the end zone. A field goal is the successful completion of kicking the ball through the goalposts after a touchdown has been score. It is rewarded with 3 points.
2. The kicker and the distance has to be entered if the field goal is converted the number of completed field goals and number of attempts for the player is incremented by 1 and score is increased by 3 if the player misses number of attempts are increased. The ‘fg converted’ button also triggers a function which compares the longest distance a player has attempted a field goal from with the value the user has entered to keep the longest field goal distance of a player up to date.



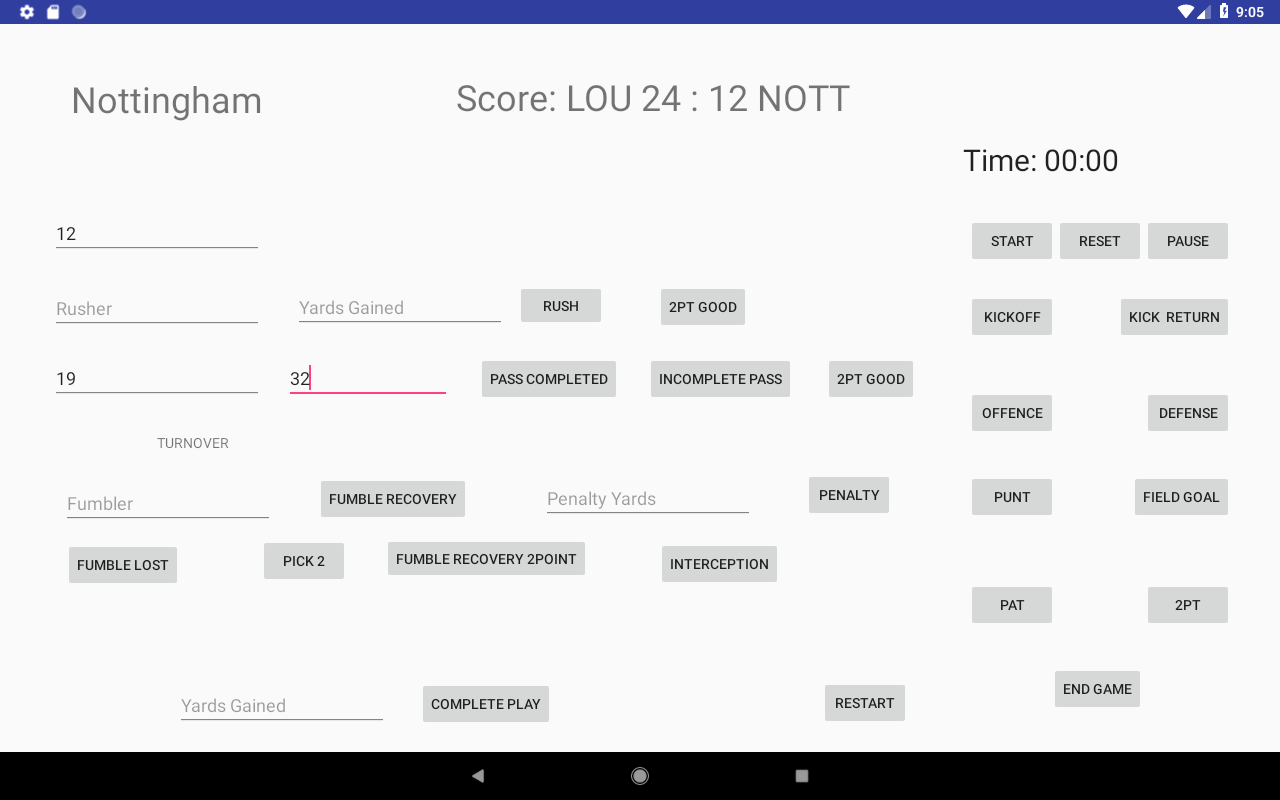
### 6.10.5 PAT



Very similar to a field goal but a PAT which is short for Point After Touchdown is a method of scoring after a touchdown has been scored as in American Football similar to rugby after a team score by getting into the opponents end zone the team has a chance to go for extra points. A PAT gives the team a chance to score an extra point by successfully kicking the ball through the goalposts.

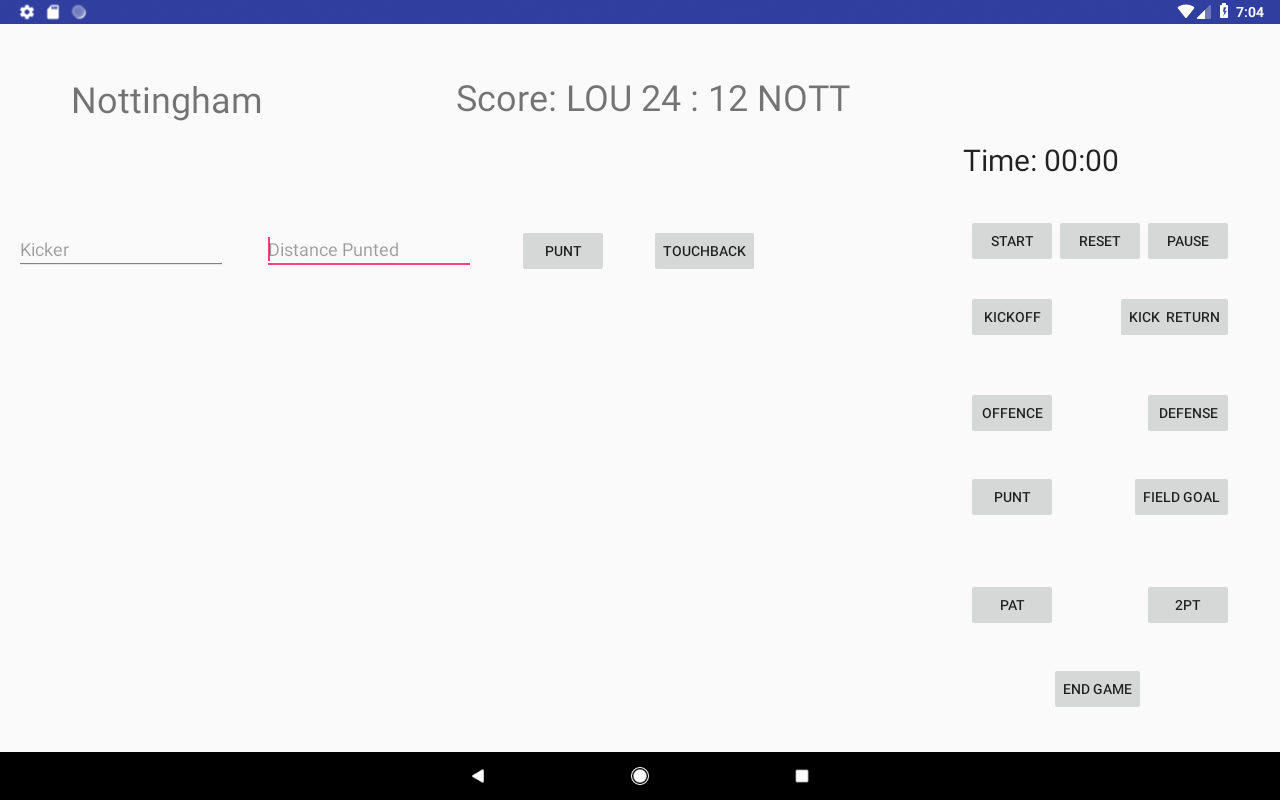
1. Kickers number is entered with distance the attempt to kick the ball from too if kick is converted users team score is increased by 1.

### 6.10.6 2pt conversion



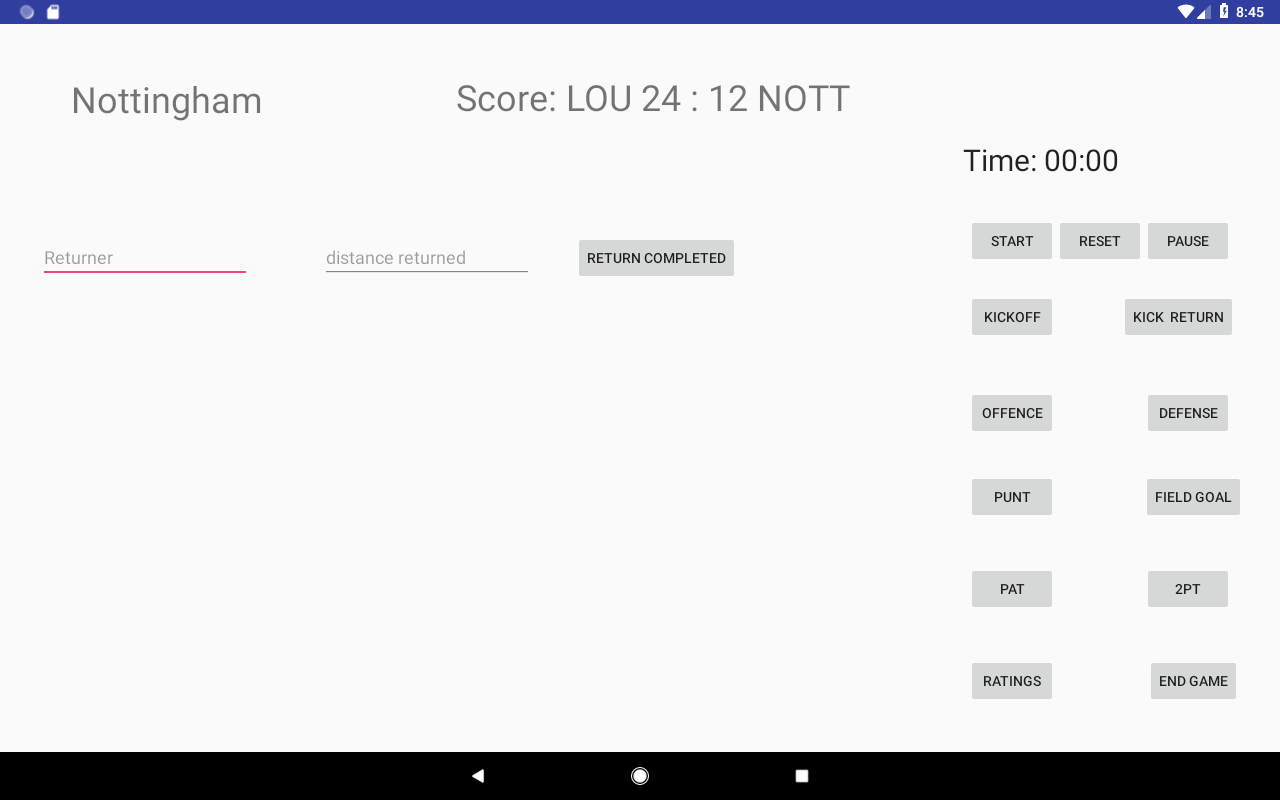
Similar to the Offence fragment but it is used in the scenario when a team scores and they are taking the option of going for a two-point conversion instead of a PAT. A team only gets one attempt and to get the two extra points they will be required to get the ball into the end zone. So, all the functions used in this fragment work exactly the same as the one in the offence fragment. The only difference is that when the team gets the ball into the end zone it will only be two extra points.

### 6.10.7 punt

****

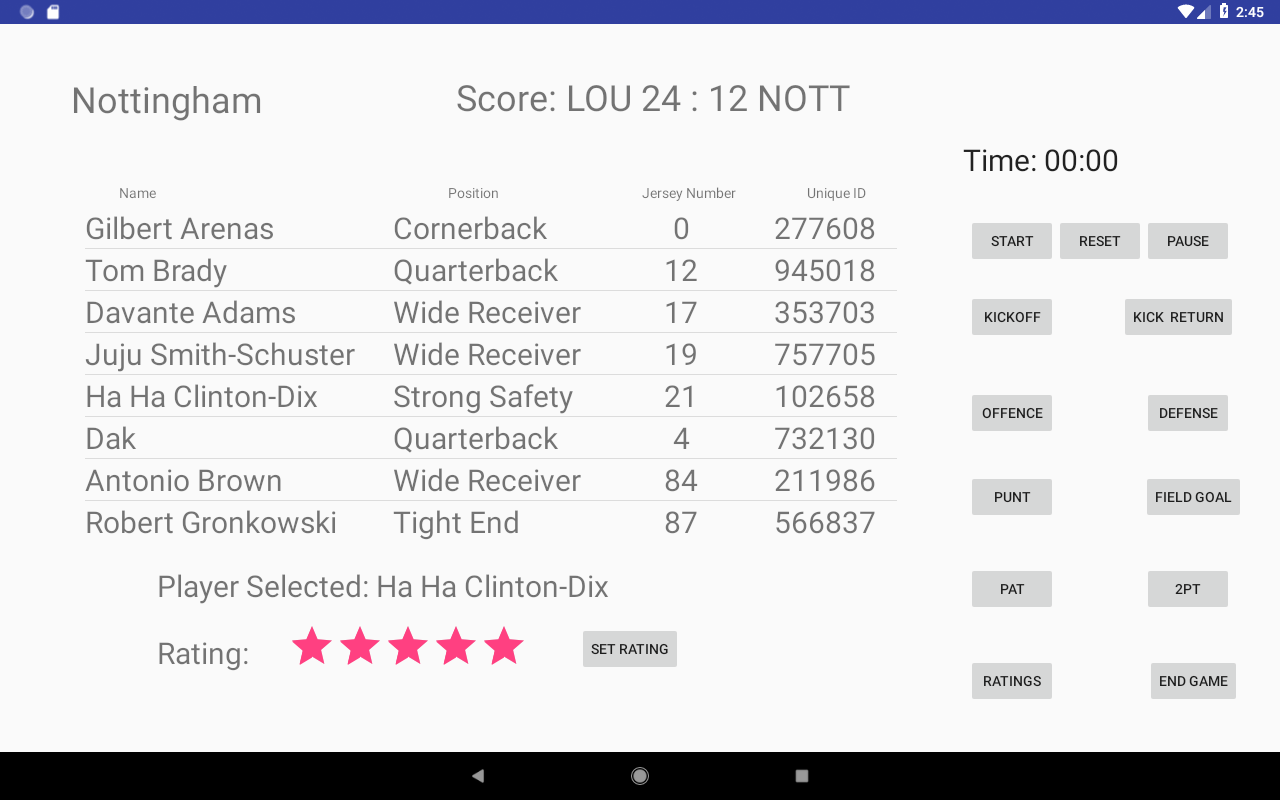
1. If ‘punt’ button is clicked it will record the distance of the punt and the player that punted the ball if the distance entered is currently the biggest distance they have punted this season and the game the distance of the longest punt will be updated with the entered value.
2. Clicking the ‘touchback’ button will do the same as mentioned previously but will also increase their number of punt touchbacks by 1.

### 6.10.8 Kick return

****

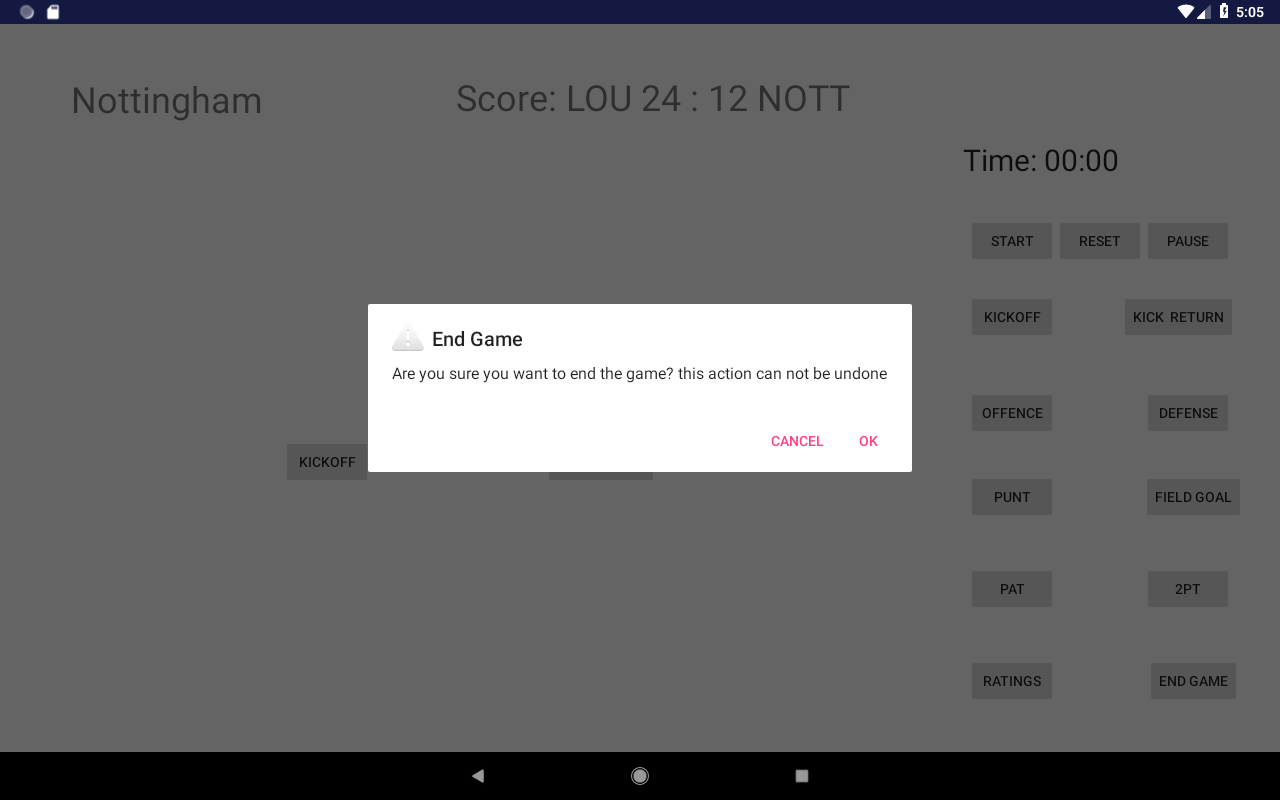
The returner and the distance the ball was returned is entered by the user. When ‘return completed’ is clicked it will query the database and compare the longest distance a player has returned the ball this season and for the current week

### 6.10.9 Ratings

****

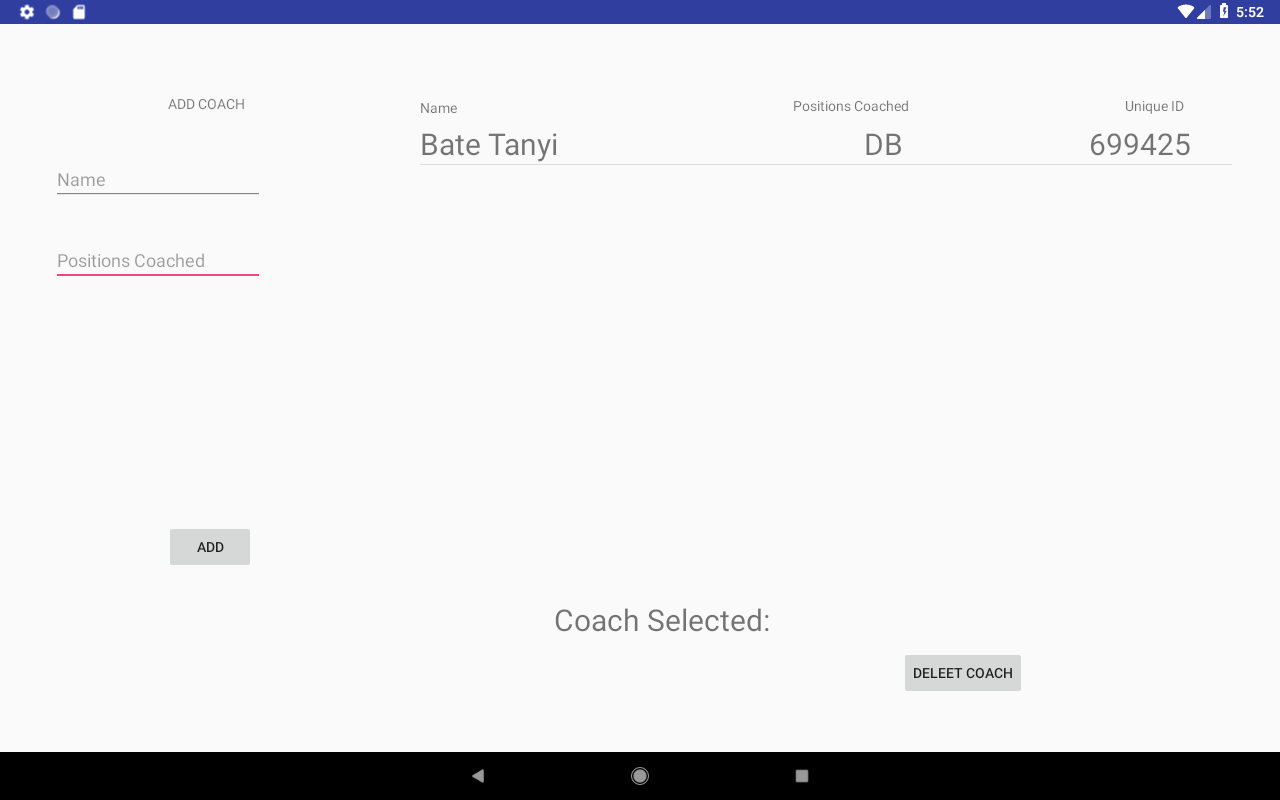
After a game is complete the admin can set a players rating based on coaches’ feedback. There is a list of players which the coach can scroll through the list of players and set their rating for that wee by selecting a number of stars. The players rating for that week will be updated with the number of stars the user has selected.

### 6.10.10 End Game

s

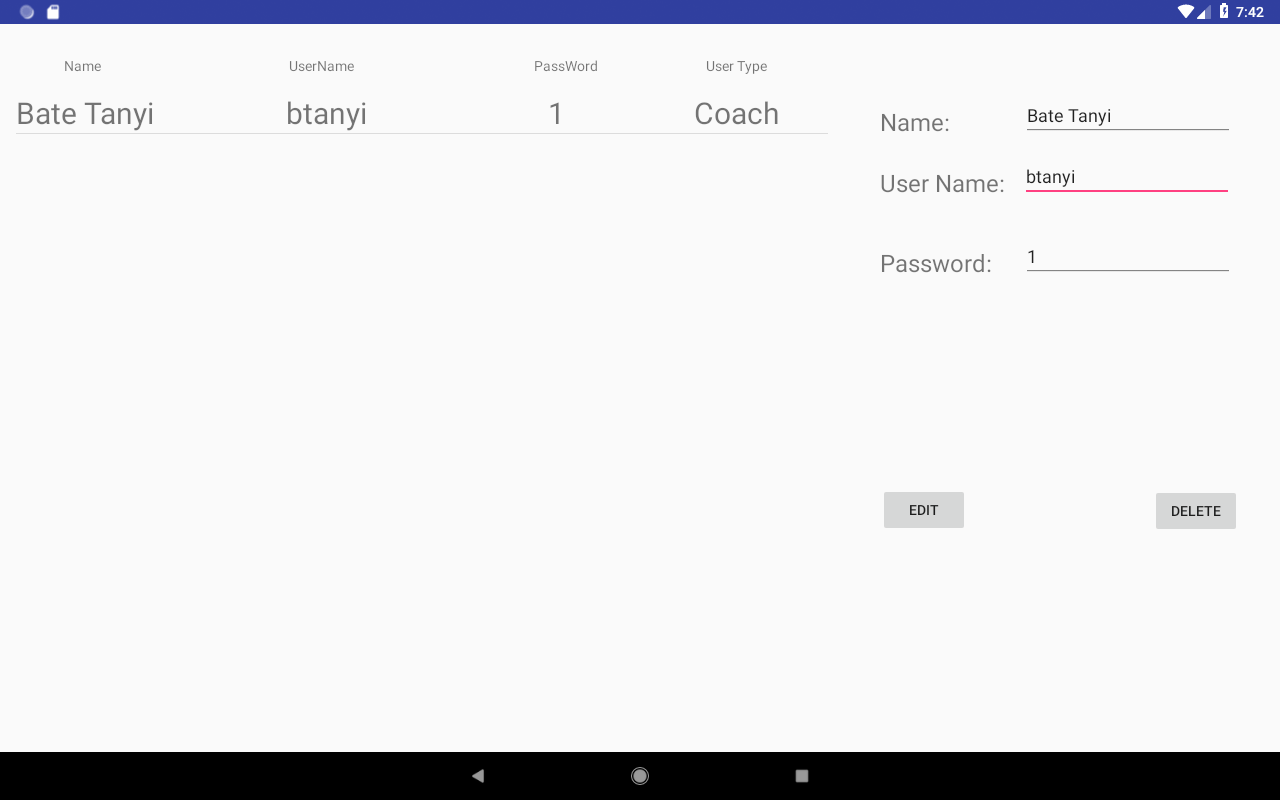
If the game is finished and all players have been given a rating user can click on the ‘End Game’ button which will change the status of the game from ‘To be Played’ and display the score of the game. After this action has be done it cannot be undone so the user is challenged to confirm the action using a dialog. If the user clicks on OK inside the dialog it will change the status of the game to played and take you back the list of games.

## 6.11 Coaches

****

Similar to the ‘Players’ activity in concept but when a new coach is added a Unique ID is randomly generated. This unique ID is given out to the coach so that he can register and use the system too. The reasoning behind using a unique ID is that only members of the team can actually use the system because in the scenario a player or coach from an opposing team was able to register and use the system it would give the opposing team an advantage.

## 6.12 Users



Allows the admin user to edit the username and password of the user or delete a user that does not belong to the team anymore. The list of users is loaded at the start of the activity. When user in the ls clicked on the fields on the right-hand side of the screen will be populated with the details of the user. These fields are textboxes which the admin can use to edit the user’s details. Clicking on the ‘edit’ button will cause the changes in the database to be triggered. The list of users is retrieved with a similar function to retrieve the list of players in the ‘Players’ Activity.

# 7. Testing

## 7.1 Splash screen

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1. Start the app** | Splash screen should be displayed for 5 seconds |
| **2.Click the back button whilst on the login activity** | Should not bring you back to the splash screen |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Start the app | Passed  27/04/2019 | N/A |
| **Action 2** | Click the back button whilst on the login activity | Passed  27/04/2019 | N/A |

## 7.2 Login

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1. Enter wrong credentials** | No log in |
| **2. Enter correct log in credentials** | Log in |
| **3. Enter credentials of a user with admin level access** | Takes you to admin homepage |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Enter wrong credentials | Passed  27/04/2019 | N/A |
| **Action 2** | Enter correct log in credentials | Failed  Returned null object error as no data found in database | Initialize the Database |
| **Action 2 retest** | Enter correct log in credentials | Passed  27/04/2019 | N/A |
| **Action 3** | Enter credentials of a user with admin level access | Passed  27/04/2019 | N/A |

## 7.3 Registration

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1. Enter invalid unique ID** | No registration |
| **2. Don’t enter jersey number if a player is registering** | No registration and challenge to enter the jersey number |
| **3. Don’t fill in all fields** | No registration |
| **4. Fill in form correctly** | Registration complete and brought to sign in page |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Enter invalid unique ID | Passed  27/04/2019 | N/A |
| **Action 2** | Don’t enter jersey number if a player is registering | Passed  27/04/2019 | N/A |
| **Action 3** | Don’t fill in all fields | Passed  27/04/2019 | N/A |
| **Action 4** | Fill in form correctly | Failed  No new user appeared in database and did not take user to sign in activity | The link was incorrectly linked to the wrong collection so was correctly linked |
| **Action 4 retest** | Fill in form correctly | Passed  27/04/2019 | N/A |

## 7.4 Players Section

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1.Add player already in database** | Notification given that player is already in database |
| **2. Add player** | Player appear in the list of players |
| **3. select player from list** | Selected player shown in ‘selected player’ box |
| **4. Load the Players page/activity** | List of players is populated |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Add player already in the database | Passed  17/04/2019 | N/A |
| **Action 2** | Add a player | Passed  17/04/2019 | N/A |
| **Action 3** | Select a player from the list | Passed  17/04/2019 | N/A |
| **Action 4** | Load the Players page/activity | Passed  17/04/2019 | N/A |

## 7.5 Player Statistics Section

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1.Load the players statistics page/activity** | Player profile correctly populated with the data of the player selected from the players Activity |
| **1.1 Load the players statistics page/activity** | Fragments should be in the order of  Page 0 – Summary  Page 1 – Week by week statistics  Page 2 – Graphical display of players rating week by week  Page 3 – All of a player’s statistic |
| **1.2 Load the players statistics page/activity** | Correct data from the database is displayed and correctly in a format which can be easily read |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Load the players statistics page/activity | Passed  17/04/2019 | N/A |
| **Action 1.1** | Load the players statistics page/activity | Passed  17/04/2019 | N/A |
| **Action 1.2** | Load the players statistics page/activity | Failed  Graph is blank not displaying any values  In week by week analysis the weeks are displayed in incorrect order  17/04/2019 | GraphView plugin errors and data is loaded asynchrously so it does not adhere to the for loop implemented to load the weeks in correct order |
| **Action 1.2 retest** | Load the Players page/activity | Failed  Graph has been populated with correct coordinates but x labels are missing  In week by week analysis the weeks are displayed in incorrect order  17/04/2019 | GraphView plugin errors and data is loaded asynchrously so it does not adhere to the for loop implemented to load the weeks in correct order  To fix the graph view plugin errors I enabled hardware acceleration and to fix the data not being displayed in the correct order I changed the database structure by having nested objects inside a document so no looping through document is required |
| **Action 1.2 retest 2** | Load the Players page/activity | Passed  18/04/2019 | N/A |

## 7.6 Games section

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1.Load the games section** | The list of games to be populated correctly |
| **2.Add a game** | Game to appear in the list of games column, in the score column it should display “To be Played” |
| **3. Click on a game in the list** | Team Statistics should be displayed in the bottom left corner and if the score column the score says “To be Displayed” |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Load games section | Passed  19/04/2019 | N/A |
| **Action 2** | Add a game | Passed  17/04/2019 | N/A |
| **Action 3** | Click on a game in the list | Passed  17/04/2019 | N/A |

## 7.7 Record Game

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1.load the section** | Score displayed should be 0-0 and correct opponent in should be displayed in top left-hand corner after selecting a game |
| **2. Navigating between fragments** | Correct fragment displayed when user clicks on it |
| **3. Clicking on a button inside the fragments** | Correct function is carried out |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Load games section | Passed  20/04/2019 | N/A |
| **Action 2** | Add a game | Passed  20/04/2019 | N/A |
| **Action 3** | Click on a game in the list | Passed  20/04/2019 | N/A |

**Testing for the individual fragments**

As all the fragments in this section follow a similar design and reuse some of the code for example the function to record that a player has made a tackle can be found in both the ‘Kick Off’ and ‘Defence’ fragment so I will be testing all the unique functions used throughout the fragments to avoid repeating test. The functions will increment both the statistics for the game week and the season for the player entered in the relevant Textbox.

|  |  |
| --- | --- |
| **Function to be tested** | **Expected result** |
| **Tackle()** | Number of a player’s tackles for the game week is increased by 1 and for the total season by 1 |
| **Touchback()** | Touchbacks +1 |
| **PassCompleted()** | Passes Completed + 1 and Passes Attempted +1 and Passing Yards increased by the amount of passing yards gained on that player which the user entered |
| **IncompletePass()** | Passes Attempted +1 |
| **InterceptionThrown()** | Interceptions + 1and Passes Attempted +1. Increase turnovers on offence by 1 |
| **PassTD()** | Passes Completed + 1 and Passes Attempted +1 and Passing Yards increased by the amount of passing yards gained on that player which the user entered. Passing Touchdowns +1 and score increase by 6 |
| **Pick6()** | Interceptions + 1and Passes Attempted +1 and opponents score increase by 6. Increase turnovers on offence by 1 |
| **PassReceived()** | Passes Caught +1 and if the passing yards gained specified by the user is currently higher than the longest reception for the receiver the original value should be replaced with user given value |
| **PassReceivedTD()** | Passes Caught +1 and if the passing yards gained specified by the user is currently higher than the longest reception for the receiver the original value should be replaced with user given value. Receiving Touchdowns +1 |
| **Rush()** | Carries +1 and Rushing yards increased by the value the user entered |
| **RushTD()** | Carries +1 and Rushing yards increased by the value the user entered and Rushing touchdowns +1 |
| **FumbleLost()** | Fumbles +1 and turnovers on offence +1 |
| **FumbleRecovered()** | Fumbles +1 |
| **Interception()** | Defensive Interceptions +1 and Turnovers on Defence +1 |
| **PassesDefended()** | Passes Defended +1 |
| **Pick6()** | Defensive Interceptions +1 and Turnovers on Defence +1 and score for user’s team is increased by 6 |
| **Yardsgained()** | Increment the value of yards gained for the team by the value entered by the user |
| **Yardsgivenup()** | Increment the value of yards given up by the value the value entered by the user, SHOULD NOT ACCEPT NEGATIVE VALUES |
| **FieldGoalConverted()** | Field Goals Attempted +1, Score is increased by 3 and Field Goals Converted+1 if distance entered is bigger than then players longest field goal converted is replaced with the value entered. SHOULD NOT ACCEPT NEGATIVE VALUES |
| **FieldGoalMissed()** | Field Goal Attempts +1 |
| **PointGood()** | 2-point attempt is converted |
| **PATConverted()** | PAT Attempted +1, Score is increased by 1 and PAT Converted +1 |
| **PATMissed()** | PAT Attempted +1 |
| **Punt()** | if distance entered is bigger than then players longest punt it is replaced with the value |
| **PuntTouchback()** | if distance entered is bigger than then players longest punt it is replaced with the value and Touchbacks +1 |
| **DistanceReturned()** | To update longest distance of return |
| **ReturnTouchdowns()** | To update longest distance of return and increase return touchdown by 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function tested** | **Input values(Players and/or Yards)** | **Expected Result Met?** | **Remedial action** |
| **Tackle()** | Player: 21 | Passed  22/04/2019 | N/A |
| **Touchback()** | Player: 21 | Passed  22/04/2019 | N/A |
| **PassCompleted()** | Player: 12  Passing Yards: 65 | Failed  Total Passing Yards did no increase when function was run  22/04/2019 | Create a function that will increase total passing yards |
| **PassCompleted()**  **Retest** | Player: 12  Passing Yards: 65 | Passed  22/04/2019 | N/A |
| **IncompletePass()** | Player: 12 | Passed  22/04/2019 | N/A |
| **InterceptionThrown()** | Player: 12 | Passed  22/04/2019 | N/A |
| **PassTD()** | Player: 12  Passing Yards: 65 | Failed  Total Passing Yards did no increase when function was run  22/04/2019 | Create a function that will increase total passing yards |
| **PassTD()**  **Retest** | Player: 12  Passing Yards: 65 | Passed  22/04/2019 | N/A |
| **Pick6()** | Player: 12 | Passed  22/04/2019 | N/A |
| **PassReceived()** | Player: 19  Passing Yards:65 | Failed  Longest passing reception for that game was 40 yards so should’ve changed to 65  22/04/2019 | correct the if statement in the function that is responsible for keeping the longest reception of the receiver up to date |
| **PassReceived()**  **Retest** | Player: 19  Passing Yards: 65 | Passed  22/04/2019 | N/A |
| **PassReceivedTD()** | Player: 19  Passing Yards: 70 | Failed  Longest passing reception for that game was 40 yards so should’ve changed to 65  22/04/2019 | correct the if statement in the function that is responsible for keeping the longest reception of the receiver up to date |
| **PassReceivedTD()**  **Retest** | Player: 19  Passing Yards: 65 | Passed  22/04/2019 | N/A |
| **Rush()** | Player: 19  Rushing Yards: 65 | Passed  22/04/2019 | N/A |
| **RushTD()** | Player: 19  Rushing Yards: 65 | Passed  22/04/2019 | N/A |
| **FumbleLost()** | Player: 19 | Passed  22/04/2019 | N/A |
| **FumbleRecovered()** | Player: 19 | Passed  22/04/2019 | N/A |
| **Interception()** | Player: 21 | Passed  22/04/2019 | N/A |
| **PassesDefended()** | Player: 21 | Passed  22/04/2019 | N/A |
| **Pick6()** | Player: 21 | Passed  22/04/2019 | N/A |
| **Yardsgained()** | Yards gained:30 | Passed  22/04/2019 | N/A |
| **Yardsgivenup()** | Yards given up:-2 | Failed  Should not take a negative value | Add validation inside the function that prevents user from entering a negative value for yards given up |
| **Yardsgivenup() retest** | Yards gained:-2  Yards given up:30 | Passed gave the user  22/04/2019 | N/A |
| **FieldGoalConverted()** | Player: 21  Field Goal Distance:53 Yards | Passed  22/04/2019 | N/A |
| **FieldGoalMissed()** | Player: 21 | Passed  22/04/2019 | N/A |
| **PointGood()** | N/A | Passed  22/04/2019 | N/A |
| **PATConverted()** | Player: 21 | Passed  22/04/2019 | N/A |
| **PATMissed()** | Player: 21 | Passed  22/04/2019 | N/A |
| **Punt()** | Player: 21  Yards Punted:60 | Passed  22/04/2019 | N/A |
| **PuntTouchback()** | Player: 21  Yards Punted:60 | Passed  22/04/2019 | N/A |
| **DistanceReturned()** | Player: 21  Yards returned: 50 | Passed  22/04/2019 | N/A |
| **ReturnTouchdowns()** | Player: 21  Yards returned: 50 | Passed  22/04/2019 | N/A |

## 7.8 Coaches Section

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1.Add a coach already in database** | Notification given that player is already in database |
| **2. Add a coach** | Player appear in the list of players |
| **3. select a coach from the list** | Selected player shown in ‘selected player’ box |
| **4. Load the Coaches page/activity** | List of coaches is populated |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Add a coach already in the database | Passed  17/04/2019 | N/A |
| **Action 2** | Add a coach | Passed  17/04/2019 | N/A |
| **Action 3** | Select a coach from the list | Passed  17/04/2019 | N/A |
| **Action 4** | Load the Coaches page/activity | Passed  17/04/2019 | N/A |

## 7.9 Users Section

|  |  |
| --- | --- |
| **Action** | **Expected Result** |
| **1. Load the users Activity** | List of users should be populated |
| **2. click on a user in the list** | The users detail should populate the text fields |
| **3. Edit the users password and username** | Username and password updated |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Plan Action** | **Methodology** | **Expected Result Met?** | **Remedial Action?** |
| **Action 1** | Load the users Activity | Passed  29/04/2019 | N/A |
| **Action 2** | click on a user in the list | Passed  29/04/2019 | N/A |
| **Action 3** | Edit the users password and username | Passed  29/04/2019 | N/A |

# 8.Conclusion

With the application complete and fully functional this chapter will discuss what was achieved during the project.

The scope of the project was large and due to time constraints and unfortunate circumstances concessions had to be made to the design and features dropped, but more positively areas were focused on and developed in more depth.

A lot of focus went into the backend architecture of the system. A lot of time was spent on researching the most suitable type of database system that would be able to deal with the challenges of system that is scheduled to have multiple users and have the ability to be scalable.

Another area that was important was the quick reads and writes to the database as the new data created is immediately required for future read and writes. In addition, it was important to structure the database such that the required data can be fetched in the least number of reads.

Another area of focus on is in the simplicity of the design of the system as the user should be note down players actions mid-game. The design of the system should have a low degree of ambiguity meaning the action a button carries out should be clear to the user as the user will be using the system in high-paced pressured environment. This will impact his decision making so layout of the button and textboxes was designed to aid the user as much as possible.

In Sports analysis or nay analysis in general the representation of data is important as it has to be clear and precise to be understood and analysed to create a hypothesis. In similar systems/applications data was displayed using tables and short summaries and it’s a common tend in analysis system to make use of graphs. Both these ways of data representation were implemented in the system.

The applications as it stands could function in the industry setting. It is a unique application that is targeted at a niche market. It incorporates functions of its competitors but to a smaller extent. This give the average user to the access to the premium tools offered by the competitors of the application.

I have already mentioned that the scope of the project was too large to fully complete the project to its full potential. I had to make some compromises. In addition, there are some improvements that would have given the application a more premium look and operate at a higher standard.

One of these improvements would be more functions to analyse player performance by giving a graphical analysis of a player vs player analysis.

A feature that a competing application used is the rating of player using algorithms which I did intend to also do but realised this is more of a Big Data problem. In the future I will use the player data in my database and the ratings given by coaches to create my own algorithms.

A further improvement would be including more technologies inside my applications such as video analysis as that is a big part in sports analysis. A coach could highlight certain events in the game and leave notes for players.

As I have already database with the correct ‘tables’ setup I could use the data in my database to set up a function where a coach could set goals and target for players.

To allow future scalability I could make the use of database sharding. A bucket could be a table, schema or different physical database. Firebase can support up to 100000 concurrent read and writes at once. Considering the constant device to device communication happening all the time throughout this app, 100000 database operations would be quickly exceeded. There are two solutions to this problem one is to shard by database as well as tables. So, for instance you could have a database for each area of the UK and place teams into the database by the area they fall into.

The final improvement I would have is in the design of the application as I focused my time mainly on including functionality in the system even though the Design is simple and effective it does not compare in the premium look of competing applications.

# 9. References

Ispringsolutions.com. (n.d.). *Top 8 Advantages of Using Video Presentations in Teaching*. [online] Available at: https://www.ispringsolutions.com/articles/key-advantages-of-video-lectures.html

Pff.En.wikipedia.org. (n.d.). *Pro Football Focus*. [online] Available at: <https://en.wikipedia.org/wiki/Pro_Football_Focus>.

Brablc, C. (2017). *A Coach's Unfiltered Review on Basketball for Hudl | Hudl Blog*. [online] Hudl. Available at: <https://www.hudl.com/blog/hudl-review-for-basketball-coaches>.

Competitive Advantage: Mental Toughness. (n.d.). *SPECIAL: What makes a GOOD COACH?*. [online] Available at: <https://www.competitivedge.com/special-what-makes-good-coach>.

Wikipedia. (2019). *Passer rating*. [online] Available at: <https://en.wikipedia.org/wiki/Passer_rating>.

Wikipedia. (2018). *BAFA National Leagues*. [online] Available at: <https://en.wikipedia.org/wiki/BAFA_National_Leagues>.

UK Government. (2018). *Data protection*. [online] Available at: https://www.gov.uk/data-protection [Accessed 8 Jan. 2019].

Developer.android.com

Coach fore (n.d.) http://coachfore.org/2016/08/24/free-stat-templates/