

Fantasy Premier League

Assistant Android

App

**John Grossi**

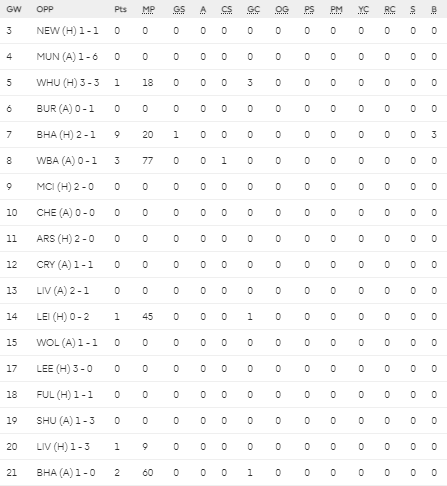
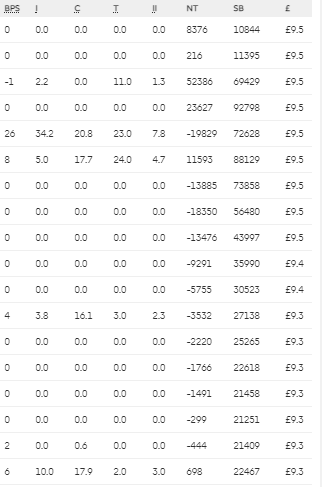
**AC40001 Honours Project**

**BSc (Hons) Computing Science**

**University of Dundee, 2021**

**Supervisor: Dr Daniel Rough**

**Abstract –** Fantasy Premier League (FPL) is a free to play game that lets anyone become a fantasy manager at the top level of English football. In the most basic form managers have a budget to pick players from teams then decide who plays each week, given one free transfer each week. It is a game of strategy in which players must predict who will get them the most points and have them in their team.



The aim of this project was to design and develop an Android app which will benefit players of FPL in making transfer decisions. The app includes features such as predicted line-up predictions for all fixtures, a player comparison tool and a best FPL team predictor for every week. The official app does not include these, but these features are what players of the game seek as known by user surveys I gave out and provide a positive impact on their decision making as known from my research into them. To ensure the app achieves the goal of helping FPL users, users were involved in the requirements gathering, prototyping stages and at the end of the project to evaluate the app to ensure all features are ones that the average FPL user is seeking and also that the whole UX of the app is fit for purpose.

# 1 Introduction

Every week millions of FPL players are left with the same tantalising question “who do I transfer in this week?”. The main objective of FPL is transferring in and out different players trying to get the best 11 for that week’s fixtures to score the highest number of points. But how do you know who to bring in? This is decided on several factors: who they play for, who they are playing against, form, numbers of goals, assists or clean sheets (not conceding a goal), if they are fit/available to play and several other factors. FPL gives you all this data and more as you can see in Figure 1. This is great for the people who have a great interest in stats and numbers but for the average FPL player this screen can be daunting. If you want to compare two players you must click on the player, look through the spreadsheet and then memorise the numbers you want to compare. You then must do the same for the other player you want to compare with. This is highly impractical given that comparing two or more players to decide which one to include is such a crucial element of gameplay. Having such a poor and intimidating user experience can be such a turn off to new and current players of the game and I feel this is crucial to address this issue.

Figure 1: Data that is shown when clicking on a player

What I propose is an Android app that will achieve two objectives. One is to include features that FPL players want and would benefit from that the official app/website does not provide to help make transfer decisions. The second being to improve on the way users compare players they may be trying to decide between, instead of the separate spreadsheet system FPL currently uses.

To achieve this, the following steps were required. Construct research into other FPL assistant app features and construct a survey to ask FPL players directly what features they want and what data they care about. While doing this I will do research into how to implement these features or why they are important or what impact they would have on a players team. For example, researching prediction algorithms that will produce the best results in fantasy sports or why it is important to know the starting line-up of a team from an FPL perspective. I will then create paper and high-fidelity prototypes to let users try to get feedback on the features and the overall UI of the app. Lastly, I will use an agile software development approach to create the application.

In summary, the primary goal of this project is to help FPL player’s overall user experience with fantasy football by having access to features that are not available from the official app and will have a positive impact on their decision making when it comes to player transfers.

# 2 RELATED WORK

In recent years FPL has seen a recent surge in popularity when it comes to content outside of the official site. From the many YouTube channels uploading videos about their team selections and tactics to apps designed to help players make their transfer decisions. To see what other apps are doing and sometimes more importantly what they are not doing I downloaded some of the most popular FPL apps.

## 2.1 Other FPL Apps

Five of the most popular FPL apps I found were: Fantasy Football Fix (100,000+ downloads), Fantasy Manager (50,000+ downloads), Fantasy Football Manager (1,000,000+ downloads), Fantasy Football Assistant (10,000+ downloads) and FFHub (10,000+ downloads). After downloading all of these and trying them I created a table to highlight their features [Appendix 1] and a full write up of the apps. Two of the apps I found did not offer much in terms of new features, they seemed to just be re-skinned versions of the official app. These did not offer much for me I wanted to gather the full range of different features offered by popular apps to use as prompts in my user research. The other 3 apps all offered different features that I decided to delve into further to see how much of an impact they might have on a user’s performance and the most effective ways to implement them.

## 2.2 Predicted Line-Ups

Knowing who will start in a premier league game is very important for different reasons. It is not as simple as more playing time means more chance of doing something (although this is important). It is due to the scoring system in FPL [Appendix 2]. Each player gets one point for playing in a game and a second for playing 60+ minutes. Let us look at goalkeepers and defenders. These types of players are defensive-minded so their goal in a game is to not concede goals as they get points for not letting it happen and lose points if they do. They only get clean sheet points from playing 60+ minutes and not conceding. So, it is important when picking defenders to make sure they are going to start so they are more likely to get these points as they cannot get this unless they then come on before the 30th minute. This is highly unlikely as first half substitutes are very rare, and defenders are the least substituted players (next to GKs) [1]. Therefore, it is critical to pick defenders who will start the game. Midfielders and forwards also benefit from starting games rather than coming on as a sub. Both types of players get most of their points from scoring goals.

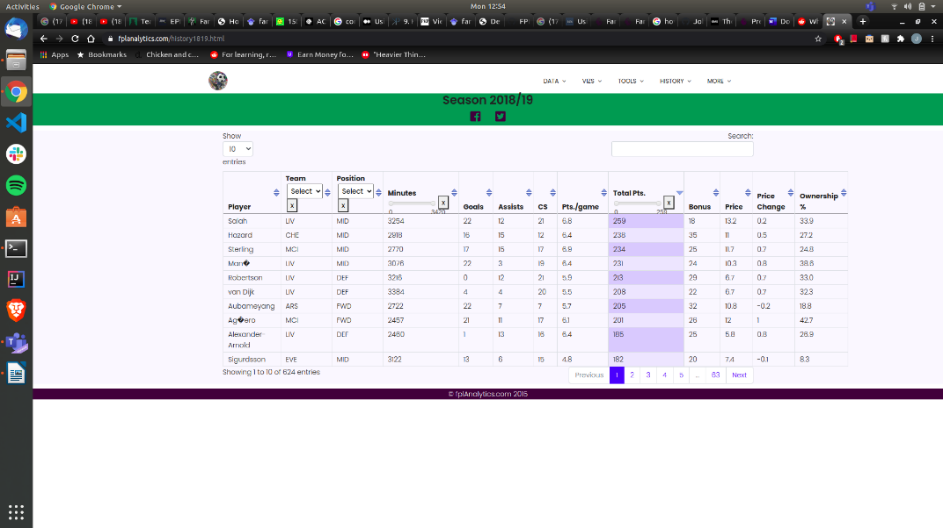
Here it shows players that play a full 90 minutes or even start the game but later get subbed off score more goals than players that get subbed on. So knowing who starts games can be vital to picking players for your team from defenders getting clean sheets and a more likely scenario of your forward players scoring goals. All of which results in you getting more points.

Figure 2: Table showing number of goals from players depending on time played [2]

## 2.3 Best FPL Team This Week Prediction

When choosing a structure for my algorithm I looked at other people’s attempts at predicting the best FPL team and general sports prediction algorithms. After searching I narrowed my focus to two algorithms that both showed promising results and had similar structures to what I wanted to use.

First was a model proposed by Bonomo et al [3]. This model was based on the Argentinian fantasy football (Gran DT) which is “run by a major Argentinian newspaper for the first division of the real Argentinian professional soccer league”. The scoring system and transfer system has the same idea as FPL with a few additions such as points awarded by the newspaper for MOTM. The model worked by averaging the past three weeks points for a player to then predict how many they would get in the upcoming. This number was then multiplied a number between 1.05 and 0.95 four times depending on the following things: playing at home (x1.05) or away (x0.95, league position (1 to 1.05 if in the bottom five of the table, 0.95 to 1 if in the top five) and a number between 1.05 and 0.95 depending on a scoring streak. Lastly it is multiplied by 1 or 0, 1 if they are expected to play and 0 if not. So if someone has a high expected points but isn’t going to start (say because of injury) their points are set to 0 as they won’t play. There is an exception for if they think they will be substituted on they get x1 instead of 0.

The second algorithm was a linear/lasso regression approach proposed by William Eilertsen et al [4]. Regression algorithms are common tools in sports prediction but for the purpose of this review I'm going to be looking at the specific one proposed by William. He splits the model into 3 steps: position, variable selection and then fit into a regression model. The variables are the realised points (actual points gained), team, position and home/away. There are then the numerical variables: each week’s previous points, price, transfers in/out, minutes played in each game, yellow/red cards, goals, assists, penalties missed/saved, saves and clean sheets. In the paper they recognise that each position gets different points for different things. They categorise them into each position before continuing with the regression. This is what lasso regression is, by not factoring in for example clean sheets for a striker it gives a more accurate point prediction as it does not matter in a points perspective if a striker’s team does not concede a goal. The model then takes in training data which is done to determine the accuracy and the value of error, this error is then accounted for.

## 2.4 Why Look at The Stats?

Why not look solely on real-life ability when making transfer decisions? One of the biggest mistakes that new or existing FPL players make is going for “big name” players. When they are given the choice of players, they will go for all the players that are heavily talked about in the media or their social circles. Although this is not the way to look at fantasy football. Fantasy football is a stats game, not a popularity game. Even though pundits will go on about how good some players are it does not always transfer over to FPL. Two examples of great players not transferring well into FPL are N'Golo Kanté and Virgil van Dijk, with Kanté winning the player of the year in the 16/17 season and Van Dijk winning the same honour in the 18/19 season. Both players had excellent seasons, coming in for big money moves, and were instrumental to their teams doing well that season with Kanté helping Chelsea win the league and Van Dijk steadying a leaky Liverpool defence.

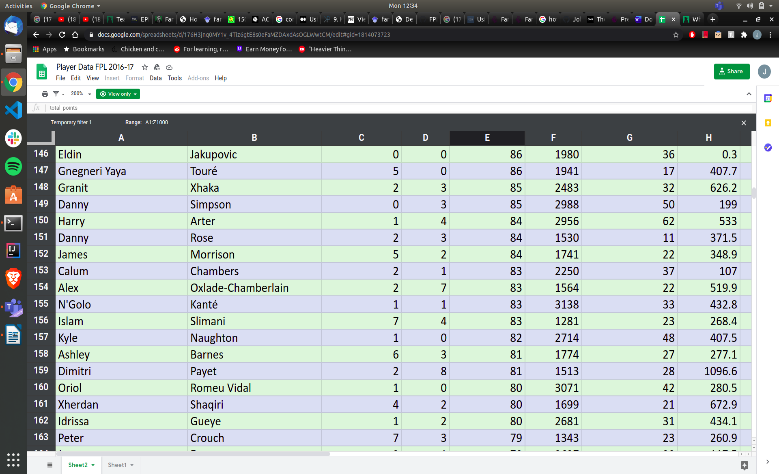


Figure 3: 16/17 FPL player season stats [5]

But as these figures show real-life ability does not always move over to FPL. The season that Kanté won the player of the year award he only came in 156th place for points, as seen in Figure 3. A drastic 181 points off the topmost points. The main reason he is not a good FPL option is due to his role as a player which is a defensive midfielder. A midfielder will only get one point for clean sheets. Therefore with this system Kanté does not get as many points as the defenders who get 4. Since Kanté does not offer much in the attacking sense of goals and assists and only gets one point for his defensive contribution he is not a good FPL option

Figure 4: 18/19 FPL player seasons stats [6]

Van Dijk suffers a different problem compared to Kanté. Van Dijk is a highly renounced player and he greatly improved Liverpool's defence which resulted in them winning more games [7]. When he is playing Liverpool have a 70.5% win rate and concede an average of 0.94 goals per match. When he is not playing, Liverpool’s win rate decreases to 42.9% and concede an average of 1.57 goals per match. Conclusively, without Van Dijk playing Liverpool concede more goals and lose more games. Conceding fewer goals mean more clean sheet points and if they’re winning more games they have to be scoring more goals. These statistics show that Van Dijk is fundamental to Liverpool playing well, keeping clean sheets and is highly praised, winning player of the year that season. However, Van Dijk was still not the highest-scoring defender, or even the highest scoring Liverpool defender for that matter in the season he won player of the year, as seen in Figure 4. Andy Robertson (Liverpool, left-back) has outscored Van Dijk every season for Liverpool as seen below.

Figure 5 : Robertson vs Van Dijk season stats [FPL website, in transfers]

The main reason for this is because Robertson is a full-back and Van Dijk is a centre-back. Centre-backs are traditionally more defensive and full-backs can be more attacking. Since they have this freedom to go further up the field, they are more likely to get assists more often. Where centre-backs are more likely to get goals from corners (due to their height) but far less often. In summary a player should not be picked solely on their on pitch ability but instead how likely they are to do things that result in gaining points and not losing points.

This stat-based approach is most commonly known as a ‘Moneyball’ approach. Instead of using scouts who use a subjective approach Billie Bean (an American baseball coach) used a stat-based approach when bringing in new players with their team’s small budget. This approach was later written into a book by Michael Lewis called ‘Moneyball: The Art of Winning an Unfair Game’ [8] and has been shown to work in football as Liverpool have adopted this approach with much success [9].

**3 Specification**

**3.1 Methodology**

When deciding on a methodology to use as my approach I looked at the positive and negatives of both to decide which would be a best fit for the project. After much consideration a mostly agile approach was decided upon. One of the reasons was because I had expected many obstacles to arise during the software development due to many reasons. I had never created a mobile application, it had been some time since I had programmed in Java and I had not had any previous experience using Firebase. Since an Agile approach allows for flexibility and quick restructuring which a Waterfall approach does not, it made the most sense to go with this approach. After my requirements were created I turned them into user stories and held all of them in my sprint backlog[#-clickup]. These could then be split up into individual sprints which would focus on different features. This is where I then decided to go for a highbred approach. Getting a review from FPL players at the end of every sprint was going to be difficult as I would need them to be available every week. Therefore I got user feedback multiple times before the software development, a survey, paper prototype focus group and a high fidelity prototype focus group, then a final user evaluation after it. Although my advisor and I held weekly meeting on Microsoft Teams for around an hour to discuss what I had done that week and what I would be doing in the upcoming week. This helped me keep motivated and was useful to get regular feedback on how I was progressing and what I could improve on.

**3.2 Tools & Technologies**

Throughout the whole project many tools and technologies have been utilised.

GitHub was vital to the development of the project as it was where I could hold all my files and have version control for all of them if something were to break while editing them. Having these stored remotely was also useful to know if my PC was to lose files in anyway there was a backup of all of them on GitHub. Being able to split all the different features into different branches helped keep thing encapsulated and made into smaller more manageable chunks. After a feature was created and tested, I knew I could then merge it into the main branch and have no problems. Some other alternatives were briefly considered such as GitLab and BitBucket but GitHub was what I had the most experience with and I knew it was fit for purpose for all the things I previously mentioned.

When it came to creating my Android application I went with the Android Studio IDE. Again I had no previous experience making mobile applications so I had to research what would be the best approach. Android studio is the official IDE for Android and has all the tools I would need to make an Android app. It seemed that it was what everyone who makes these applications use and no other IDE came close to the tools and convenience that Android Studio offered. Eclipse is the only other IDE that people seemed to use but to use it you would need to add a plugin to do Android development. Also, Android Studio offers an easy drag and drop feature to easily edit the xml files which Eclipse did not have as well as an emulator to see the application running. Having an Android phone helped with development as Android studio made it easy to just connect my phone by cable and have it run with no problems.

With version control and my Android IDE chosen I then had to choose how I would approach the back-end of the application. I wanted the back-end to do most of the heavy lifting to avoid the app being a large size to download and reduce loading times as much as possible. So, I wanted the back-end to: get the data I needed, store the data and manipulate it before the app requests it. For example every time somebody requests the best team for the upcoming week I want it to be in the database already instead of having to get every player every time and having to run an algorithm to decide who should be getting displayed to the user. After much consideration I decided to use Firebase. Firebase is a backend service which you can use for mobile and web applications. They provide NoSQL databases, cloud functions and an array of other tools. Firebase documentation is extremely extensive and they have a very helpfully YouTube channel which definitely made it very appealing to use. Also being owned by google Firebase was integrated into Android Studio so getting it set up and connected with the app was made much simpler. The cloud functions could be used to take off much pressure from the front-end by being able to time them to run at specific times on specific days to perform tasks. For example getting team news from external sites or running my prediction algorithm so the front-end would just need to get the players from the database and then display them. Another advantage was the amount of free storage, reads and writes Firebase offered which could not be rivalled by many other services. Because of all of these features previously mentioned Firebase was decided upon to be used. Another alternative was a XMPP server but compared to Firebase its documentation was not as extensive, difficult to set-up and would be more difficult to connect to my app than Firebase. The last option would be to create my own server to host my back-end but due to time constraints with the amount tasks I had to have done this didn’t seem to be the best course of action.

Lastly I need a place to manage my sprints and product backlog. When it came to this nothing fancy was needed so it was decided to use ClickUp. ClickUp was exactly what I was looking for where it had a template that made it easy to make a product backlog, where I put all my user stories, and sprints. Having useful features where you can see all user stories and which sprint they’re in, what stage they’re at in development, story points and an easy to use interface. There are many other project management tools I could have used here which would have achieved the same things I wanted but ClickUp was decided simply as it was intuitive to pick up so I didn’t need to spend time learning how to use another and had all the tools I wanted to help manage all my user stories for each sprint.

**3.3 Diary and Meeting Minutes**

Throughout the whole project notes were taken to keep track of what was getting done and to have a better idea if deadline were going to be met. Daily notes were taken, Appendix #, until the software development started as my sprints were keeping a record of what was getting done that week. Meeting minutes, Appendix #, were kept with my advisor to keep a record of what was discussed every week. This helped to be able to look back on and make sure everything that had been mentioned in the meetings were being addressed that week.

**4 Design**

**4.1 Requirements gathering**

To gather my requirements/features the best way was to go straight to the people who would use the app and ask them what they want. The survey included questions to find what features they would most like to see in an app, what data they use to make decisions and what other medias they used to help them make transfer decisions. Thanks to this I was able to see what feature people were most passionate about and what data they wanted to be shown to compare players to make their transfer decisions. Some of the most popular answers included the current weeks fixtures, a player comparison tool, predicted line-ups and a best FPL team predictor. Thanks to the answers I got I also knew what data to show the user when they wanted to compare two players. The survey was created using JISC and was answered by twenty-four FPL players. The full survey and responses can be found in Appendix #.

**4.2 Paper prototypes**

After consulting my background research and my user surveys I decided on the following features:

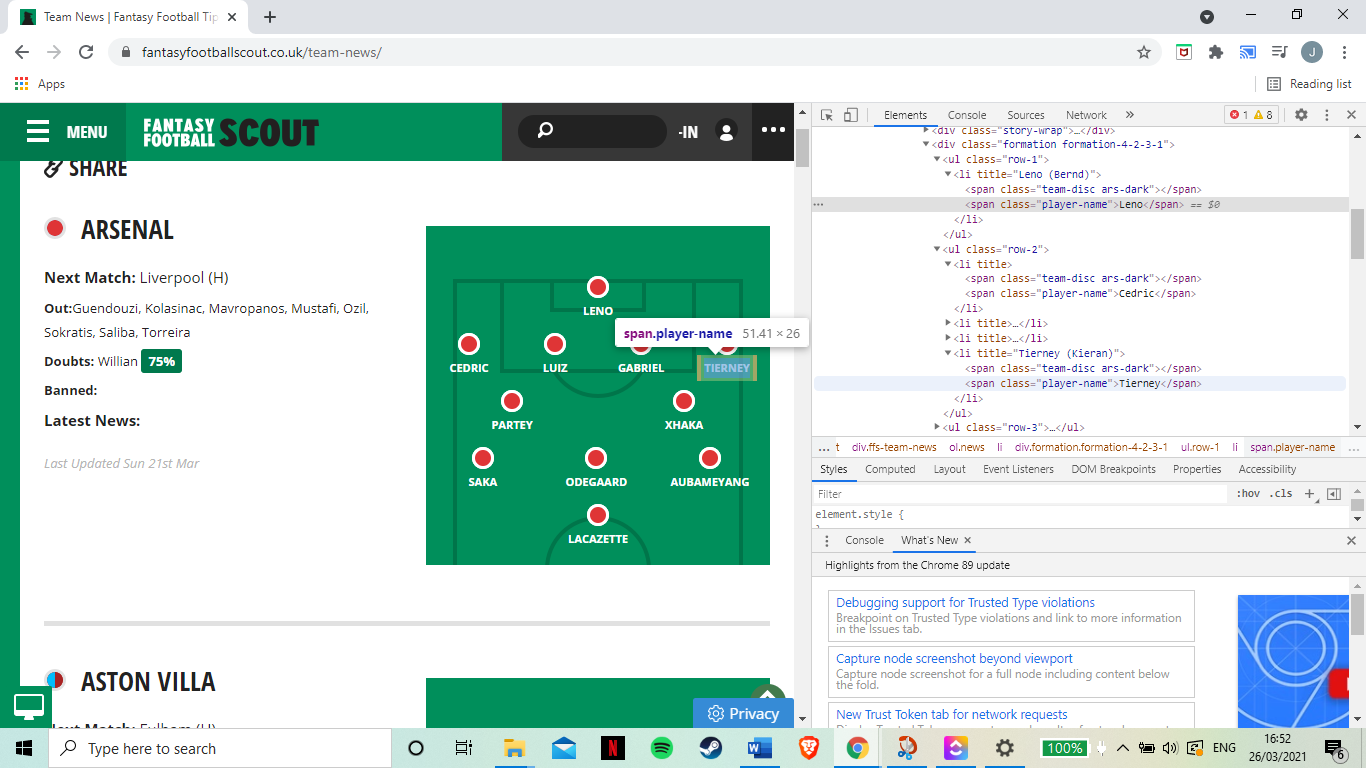
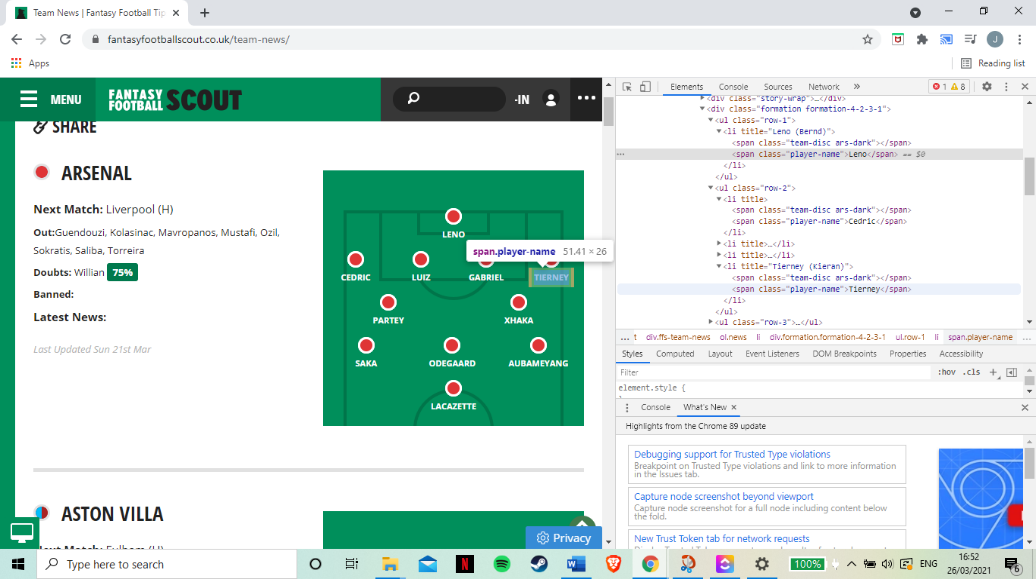
* This week’s fixtures
* Line-up predictor for each fixture
* Player comparison tool
* Highest scoring team predictor

The background research helped me realise that knowing who starts games often result in more points, so this is important for people to know if they want to improve their scores. A player comparison tool was one of the most popular results from the survey and is not a feature offered by the official app or even many other apps currently on the market. Finally, prediction algorithms of the best FPL team that week have great results according to my research into them, was a popular option in the survey and not many other FPL assistant apps offered this. Overall, all my features had a combination of a positive reception from the survey and from my background research but also was offering something that that other apps on the Play Store weren’t..

With these features in mind, I created a paper prototype to encompass them [Appendix 4]. These were then put into Marvel which lets you create your paper prototypes into more interactive ones, where clicking the drawn buttons will move you to the next screen. You can try it yourself here: <https://marvelapp.com/prototype/27a4d81g> change to appendix?). This was sent to three FPL users and a focus group was constructed. The interviewees were asked what they liked and did not like about the prototype. The array of features and clear design were the highlights but some things, for example the predicted line up not being clear that it wasn’t the official one, were brought up as a negative. Feedback was recorded and was applied to my high-fidelity prototype [Appendix 5].

**4.3 High Fidelity Adobe XD Prototypes (expand now I have space)**

All the feedback from the paper prototype was taken into consideration and applied to the production of the high-fidelity prototype. This was created using Adobe XD and can be seen in the appendix [Appendix 6]. This also was then presented to a focus group of the same three people. The interviewees again were asked what they liked and did not like about the prototype. The group really like the inclusion of the player images being used but found that some of the buttons, for

example the budget on or off button, was too small and not very clear. Feedback was again recorded was will be used when creating the final product [Appendix 7].

**5 Implementation**

When it came to my software development cycle as previously mentioned it was decided to take it on with a mostly agile approach. There were originally only eight sprints but by the end there was a total of ten, which was expected as I knew things would take longer than expected as I was unexperienced in many things such as the langue, tools and environments. Each sprint was a week long lasting from Monday to Sunday and each had a focus of a different feature.

**5.1 Sprint 1 14/01 - 17/01**

For the first sprint it was a bit shorter than the preceding sprints. This sprint lasted four days and the main focus way getting myself familiar with the technologies I was using and setting myself up for the rest of the software development. During this sprint I was able to set up and become familiar with android studio and get an app to move between 3 screens which would hold my three main features. After this I wanted to set up my back end and become familiar with how it would worked. I created a Firestore database on Firebase. This is a NoSQL database which is flexible for mobile development. It keeps data in sync across apps but also offers offline support so the app is still responsive regardless of internet connectivity. When this was set up I was then able to connect my Android Studio project to the back end Firestore. Being all owned by Google made this fairly simple. Now that all of this had been done I spend the rest of the sprint becoming familiar with the FPL API. This is the official API but has no documentation at all. After some research a Medium article was found [#] which contained all the end points and the data it would retrieve. All these end points were recorded for future use.

**5.2 Sprint 2 18/01 – 24/01**

Sprint two was the first full sprint and the focus was on the predicted line-up fixture. The idea with this function was that a user would click an upcoming fixture and it would show the user the two teams predicted elevens of player who will play that fixture. Firebase offers cloud functions which can be ran at certain times. So with these the idea was to write a cloud/firebase function that would predict the eleven players for each team then write them to the database. How the function worked was it would go to three different websites and web-scrape the eleven player for each team.

Figure # : FPL Scout

Each site required a different approach to scrape their data but for an example Figure # shows one of the sites and the HTML of the page. What the scraper would do is then shown in the figure bellow, Figure #

Figure # : FPL Scout

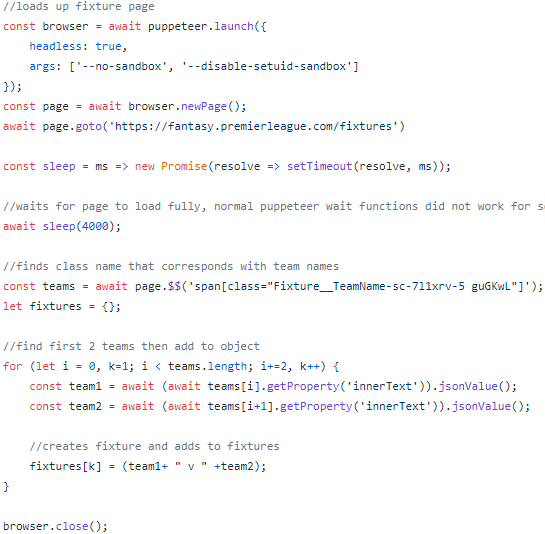
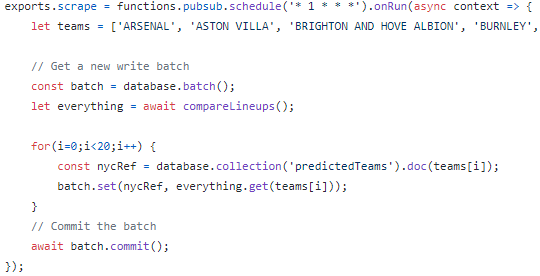
What the function does is it requests the webpage that contains all the team line-ups and loads the HTML. With the HTML it then goes through each player name under the “.player-name” tag, gets their last name and then adds it to a player array. Once it has done this it then sets a JavaScript object to have a nested object which is the first team’s name, this object is then set to hold the eleven player predicted to start. This loops until all 20 teams have been added with their predicted elevens. E.g teamANDpalyers {Arsenal { 1:Leno, 2:Holding ….} Aston Villa {1.Martinez… This nested JavaScript object is then returned to method which is responsible for then comparing this Object with the two other gotten from the other two sites. The code snippet is too large to add as a figure but what it does is work on a 2:1 ratio. If two sites think for example Mendy will start in goal for Chelsea and the other thinks Caballero will the code will add Mendy since that was the majority. This is done for all the positions for each team which results in an Object with the same structure of holding teams which hold players but are the players that have been predicted most likely to start and were at least 2:1 with the websites.

Figure # : Code

This data is then written to the database with a path of ‘predictedTeams/TeamName’ where predictedTeams is the collection, TeamName is the document and then all the player names are set to the documents fields.

**5.3 Sprint 3 25/01 – 31/01**

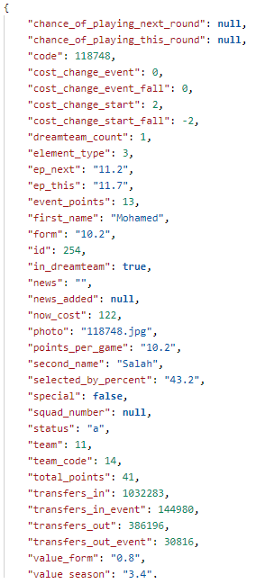
Sprint three was the first sprint that a roadblock was hit. The purpose of sprint three was to do the back-end of the fixtures feature. This feature would show the user the week’s upcoming fixtures which they could then click on to see their predicted starting elevens. The plan was to use the FPL API as one of the end point was to do with all the game’s fixtures. Unfortunately the end point seems to be broken. For the first attempt a simple request was done but no matter how it was tried the endpoint would not return a readable response. When this didn’t work a GitHub repo/library was tried. This library also didn’t work so it was concluded that the end was broken as myself or external libraries were unable to get a response. Since the API wouldn’t work another approach was attempted, web scraping. I was going to use the same approach as last time using cheerio to get the HTML but it was noticed early this wasn’t possible. This was due to the page using JavaScript to load the fixtures so when cheerio loaded the HTML the JavaScript wouldn’t run to show the fixtures. So to get around this problem puppeteer was used instead. Puppeteer simulates a chrome window instead of just getting the HTML, this lets the JavaScript run and the teams then are in the HTML code. Normally this isn’t available if using an external server as you cant make them open chrome tabs but with Firestore they have something called ‘headless chrome’ which allows you to do this with puppeteer, Once it was loaded the teams were found in the html under a tag and put into an array where I could then build the fixtures in another array which is returned at the end of the function.

Figure # : Code

**5.4 Sprint 4 01/02 – 07/02**

Sprint four was a far more productive sprint with there being less problems encountered and far more work was able to get done in this sprint as expected. The story points I gave each task in this sprint did not take as long as expected so I was able to get two features done in this sprint, opposed to just one, being the front-end for the fixtures and predicted line ups fragments.

With the back-end all set up all that had to be done on the front end was the UI, making database queries and presenting the data in the desired way. I set up my database in a way that may have not utilized normalisation but instead to make my queries as simple as possible, this was recommended in the Firebase documentation [#].

To display the fixtures recyclerView was used. This allowed the fixtures to be displayed dynamically as it only made it as large as the number of fixtures received. Each fixture was set to a button title and when it was pressed it brought the user to a new activity which shows the home and away team predicted line-ups. These can be seen in Figure #. When the activity is loaded up the two teams from the fixture which was clicked is passed through using intents. With the team names they are then used to query the database to get their predicted line-ups. In Figure # you can see the database so to query it the path ‘predictedTeams/team-name/’ is used where the team names from the fixture is just passed in to get the home and away team players. After this a tab layout was made to switch between the home and away fragments and a back button was added to users could go back to the fixtures fragment.

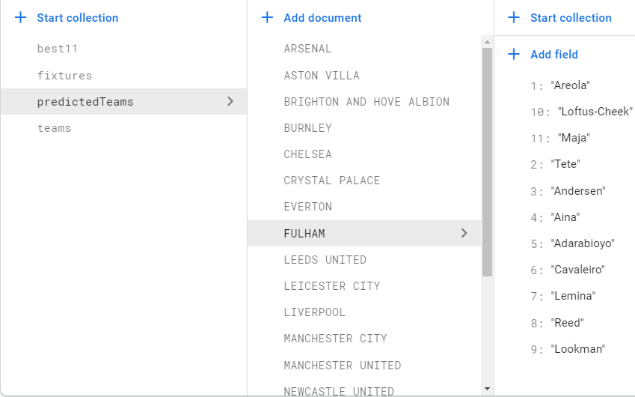
From my user feedback on my high fidelity prototypes it was brought up that the screen would look better to have the player images above their names instead of just having a circle. To avoid having the app being larger than need the player images were saved to Firebase storage where the images could be requested when needed. To get these a python script was written to scrape the English Premier League website where it would load a chrome window and keep scrolling to the bottom to load all the players. Once it had done this all the hyperlinks to the player pages were saved to an array where it then looped through them to get the players images from them. A problem was noticed here as the images were just saved as the players name .png but when there were repeats of names they would just override each other, for example there are multiple players called Fernandez who play for different teams. To avoid this the images were then saved as ‘playernameTEAMNAME’ so the Fernandez from Manchester United and Newcastle would not override each other. These images were then requested from storage and displayed above their names.

Figure # : Code

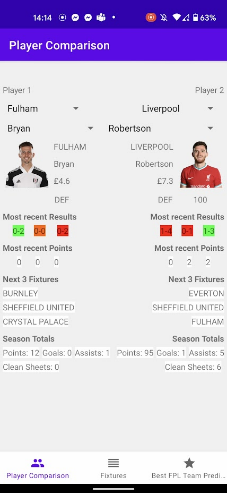
**5.5 Sprint 5 08/02 – 14/02**

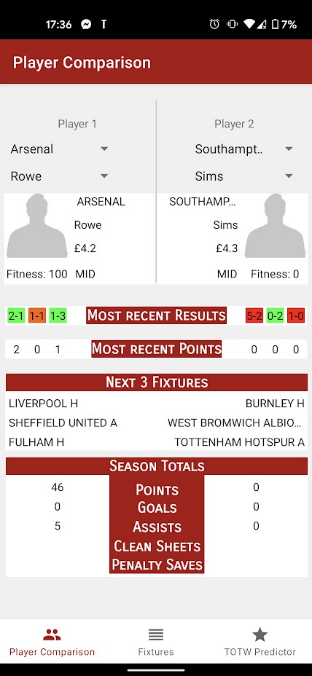
Sprint fives’s main focus was to develop the back-end for the player comparison screen. There were multiple things to be done in this sprint such as getting the data from the FPL API, manipulating the data into a more readable or beneficial form and populating the database. Thankfully the FPL API end points for player data was working to I was able to request the players data I needed. For gathering the data the first thing was done was requesting the bootstrap endpoint that return every player’s basic data as seen in figure #.

Figure # : Salah’s seasons stats

Once this is requested the function would then iterate through each player that was received through this request. First the second name is set to just be the last word as if a player has many names it sets their last to just be whatever the last one is, as footballers are always referred to by their last names. This due to consistency across the app and for when the app requests the image for the player it has to match the image name. For example João Pedro Cavaco Cancelo. The second name from the API is Pedro Cavaco Cancelo but in the predicted line-ups and images his second name is just Cancelo so all the words previous have to be removed. After this all accents are removed for the same reasons as the image names don’t have accents. This had to be done across many parts of the project. In the predicted eleven function sometimes it would not get eleven as some sites would use accents and others wouldn’t. This meant when comparing it sometimes wouldn’t work as id one site had Ag**ü**ero and another had Ag**u**ero it wouldn’t say it was a match meaning he wouldn’t be added to the prediction. So to avoid this when all the names were scraped all accents were removed so the error would not occur. Now that that had all been done the ellementSummary endpoint can be used. This requires the players id which can be found in the bootstap return, e.g elementSummary/id. This returns all the players data how they did in each previous game week and who they still have to play. Lots of the data collected at this stage has to be changed to be more readable. In the API teams and player positions are always referred to by an ID so these have to be changed, score lines are saved as 2 variables with it being the home team score and the away team score so these have to be written clearer and figure out if the player plays for the home or away team. All of this data is saved to an object with the same structure as the predicted eleven one mentioned previously, then set to the database. The team data is also set here, being the team strength, home and away attack and the home and away defence.

**5.6 Sprint 6 15/02 – 21/02**

Sprint fives’s main focus was to develop the front-end for the player comparison screen. Having the experience of getting data from the database in the previous two features made it faster to implement at this stage. The biggest challenge was filtering the data. The player comparison screen was split in half for the two players to be seen side-by-side as you can see in Figure #.

Figure # : Early development of player comparison screen

Code was written to change the colour of their results to show if they won or lost, with green being a win, orange being a draw and red being a loss. The most problematic part of this feature was definitely the drop down filters (spinners on android studio). The first dropdown would give the user the option to filter by team, picking Fulham lets say, and then the dropdown bellow would then be populated by all the Leicester players. Clicking a Fulham player, Bryan for example, then displays all the player’s data to that side of the screen as seen in Figure #. All the data on the screen are the most popular results from my user survey of what data they looked at for each position when making a transfer but also what data results in more points for their position. For example it makes sense to show clean sheets for defenders as that is their main income for points but forward players get very little or no points for keeping clean sheets. Different positions show different stats but everything down to ‘season totals’ in Figure # is the same for every position. Goalkeeper has total points, clean sheets and penalty saves, defenders have total points, goals, assists and clean sheets and midfielders and forwards have total points, goals and assists. The code works as the first dropdown displays the team and when the user selects one the code makes a database call with the team name in the path. This returns all the players for that team and when the user selects a player it then makes another database call this time with the team and the player’s name returning all their stats. Making all four dropdowns (two for each player) work was troublesome. Instead of having four almost identical methods for each I was able to make it work in one onClick method but have two if statement with the first being for the team spinners and the second being for the player name spinners. This meant the code had to get the spinners id of which had been clicked then change the relevant text views so the data was shown on the correct side.

When the user selects the player the code grabs the players data but it then gets their player image from Firebase storage. Although every player doesn’t have an image, this is normally due to them joining the team late and not having an official image on the FPL site. So if this was to be the case or there was an error requesting the image a default image was then used as a placeholder so the user wouldn’t wait for the player image to appear as that player did not have one. This was also implemented into the other feature screens and can be seen in Figure #.

Figure # : blah

**5.7 Sprint 7 22/02 – 28/02**

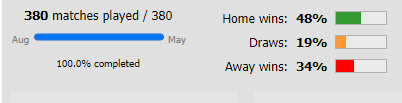
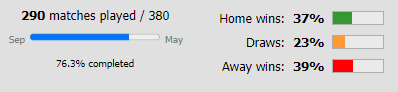
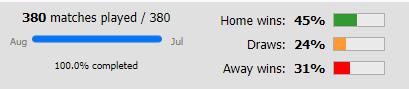
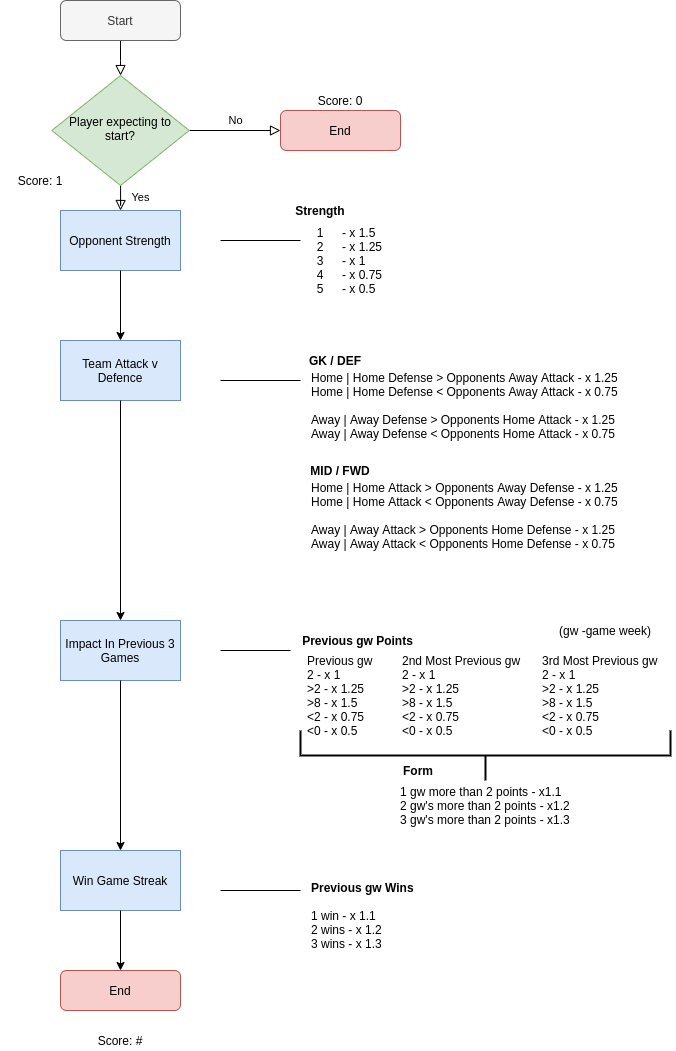
Sprint seven was going to be the start of the development of the best team that week prediction algorithm. The algorithm would be responsible for going through every player and giving them a score depending on a number of factor such as past results and likelihood of them doing well in the upcoming week. Unfortunate I was unwell this week and not as much work as expected was able to get done this sprint. Fortunately due to the amount I had managed to get done in the previous weeks and accounting for things going wrong this did not hamper development greatly as it could have. What did get done this sprint was the plan for how the algorithm would get developed, Figure #.

Figure # : blah

The first step of the algorithm was check if the player is expected to start. Due to the background research mention previously it was important to know who stats as this normally reflects in expected points. If they were not expected to start they were given a score of zero and then the algorithm would more to the next player. If they were expected to start they were given a starting score of zero. Depending on a number of factors after this the score would be multiplied by depending on their stats, their teams stats and who they were coming up against. This was a hybrid of the two algorithms from my research. Taking inspiration from Bonomo’s idea of looking at past stats to predict the future and having the system of multiplying the players score depending on the stat. Then William’s technique of looking more at areas like players positions and home/away dependencies to be more specific of where the points are coming from, for example in Figure # ‘Team attack v Defence’ looks at the players position and if they’re home or away then compares it against the opposite. So a forward at home is compared against the oppositions defence away. It was chosen to not solely look at home and away advantage as that has nor been as prominent this year. With no crowds at the grounds since Covid-19 the crowd has less of a chance at inspiring the team when at home, resulting in the home advantage argument not being relevant. This can be seen in Figure # with the 18/19 season being the last full season with crowds, 19/20 having crowds for half the season and then finally the 20/21 having none whatsoever.

Figure # : EPL 18/19, 19/20, 20/21 season stats [#] https://www.soccerstats.com/latest.asp?league=england

So instead of giving a team a positive multiply if they’re away it was instead decided to look at a team’s defence/attack at home/away vs an opponent’s attack/defence at away/home.

**5.8 Sprint 8 01/03 – 07/03**

With sprint seven being responsible for the planning of the algorithm sprint eight was where the development began. Following my flowchart, Figure #, the first step was to check if the player was starting. This was able to be done quickly as I didn’t need to iterate through every player, all that had to be done was get the predicted eleven from the database. These players scores were set to one. Next was to get their data relating to them. With this the code could then give the player its score. Each square in the flow chart was made into it’s own function which would return a multiplier which is then used to give the player their final score. GO DOWN EACH ONE.

**5.9 Sprint 9 08/03 – 14/03**

Sprint nine was responsible for solving my constraint satisfaction problem, more specifically it’s called a Multidimensional Knapsack Problem (MKP). The idea of the Knapsack Problem is that you have a bag and you need to fill it up with the combination that would hold the most value but not go over the weight limit of the bag. The MKP is the same but with more constraints than weight, sizes for example. So in my situation the bag is my eleven player slots and the constraints are as follows:

1. there must be eleven players

2. there must be at least one goalkeeper, three defenders, two midfielders and one forward

3. there must not be more than one goalkeeper, five defenders, five midfielders and three forwards

4. there can be no more that three players from each premier league team

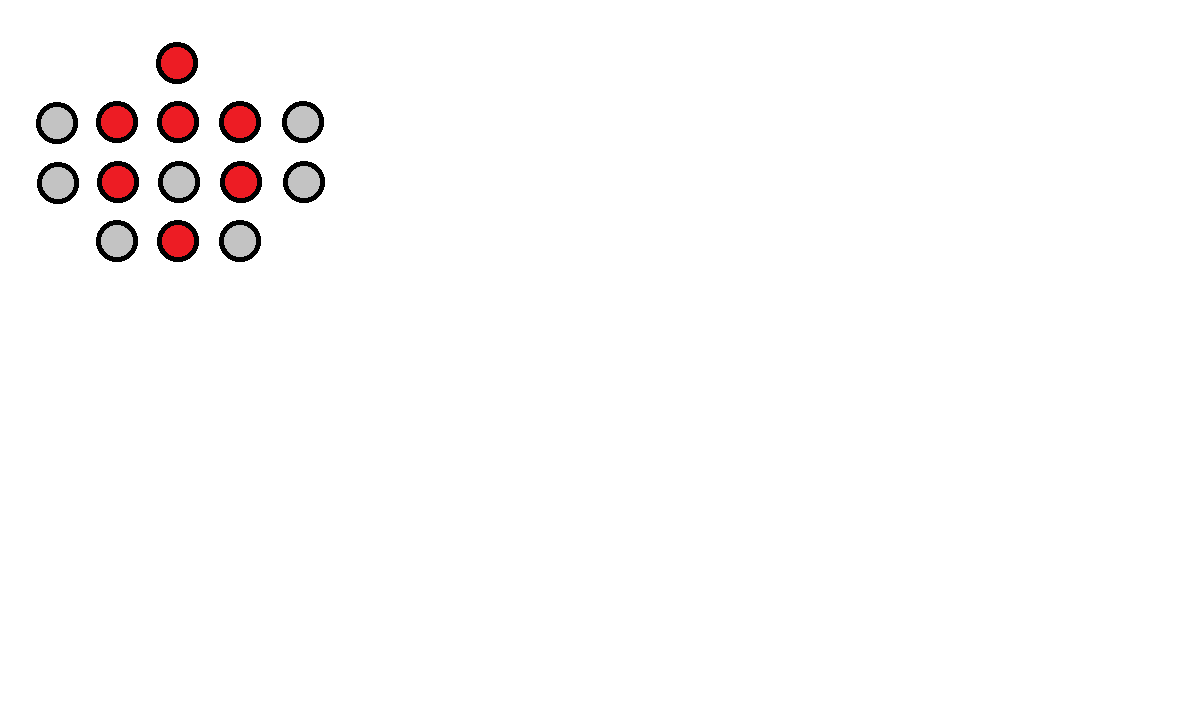
So it’s not as simple as just grabbing the top eleven players as that would not satisfy the constraints. One idea was to get the top three players from every team (means it wouldn’t go above the three player limit) and then make the eleven from that picking the best player for each position. The problem with this approach is you’re not guaranteed to get a player for every position. If no team has a goalkeeper in their top three then it does not satisfy constraint two. The next approach was to work my way down the formation to fill up the minimum formation, Figure #, then fill the rest of the spots with the next highest scoring players and skipping them if there’s already too many from that team.

Figure # : blah

This would satisfy the first four constraints but there’s still a problem. This doesn’t guarantee the highest scoring combination. I simple example would be to say the first three positions (gk, first two defenders) were all Chelsea players with a score of one, two and three. The code would then not let any more Chelsea players in the team, matching the constraints. The problem is if when it gets to the forward position its possible the highest scoring player could be a Chelsea player with a score of four. Meaning the highest scoring team would not be made. After much consideration of other methods an algorithm had to be made to solve this problem of picking the starting eleven.

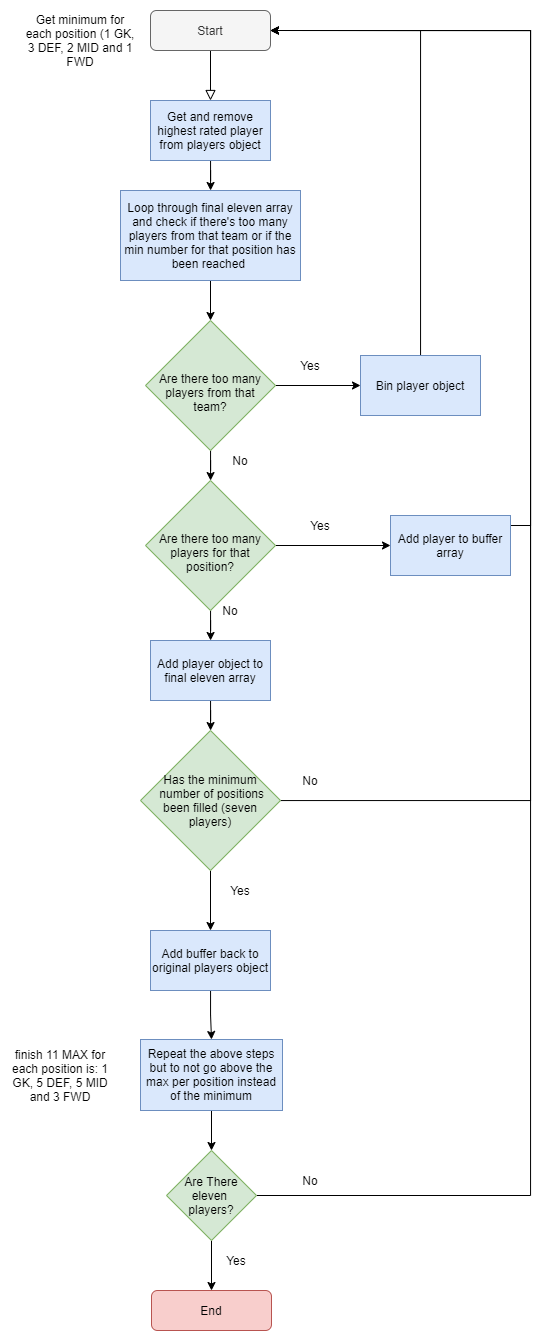


Figure # : blah

Above, in Figure #, is a diagram of how the problem was solved to meet all the constraints and get the highest scoring predicted team. The first goal is to fill up the minimum number of positions, seen in Figure <#, but in order of highest scorer first instead of going down each position in order of goalkeeper to forward. As the coded gets the highest scorer from the nested object it removes it so it doesn’t just get found every time. The array that holds the best eleven is then iterated through to check if it holds too many from the position or the team. If there is too many from that team the player is just deleted. If there is too many for the minimum position already it’s put into a buffer array. The reason why keeping these players is because if there is too many for the position at the moment when the minimum is met for every position the player could then be added to fill up the remaining spots. Although if there is already three from that team that’s never going to change so it would be redundant to keep the player somewhere as there is no situation where they will be used. So once the minimum is met for each position the buffer array is put back into the nested object holding the players. Then the same process is repeated but now each position can’t go past its maximum and will stop once it hits eleven players. This way it’s always a valid formation and is the combination that results in the highest scoring team, solving the knapsack problem I had encountered. This step of the software development was far more difficult than previously expected. It was thought that the algorithm to give each player would be the most difficult to achieve then getting the 11 would be simple. In reality it was the opposite. With so much research before hand the algorithm was easy to code as I was well informed and prepared on how it would work but as stated getting the eleven, matching all the constraints and getting the best combination of players proved challenging but was achieved. [add more code here?].

Once this was achieved the front end was developed very quickly as it was very simple to implement. The best eleven had been written to the database [talk about cronc time] so it was a simple database call to get the players then similar to the other screens of displaying them with their corresponding pictures.

**5.10 Sprint 10 15/03 – 21/03**

Sprint ten was the final sprint and was dedicated to improving the UI and fixing bugs I had noted through development. During this time I had also preformed a final user evaluation focus group but also a few informal user testing. This was just letting a few friends and family try out the app. All of this resulted in some bugs being found and UI improvements that were implemented in this sprint. Using ClickUp was useful during development and here was useful as below the backlog I was able to keep a note of any bugs noticed during development to ensure none were unattended.

**6 Evaluation**

**6.1 Focus group**

As previously mentioned, during sprint ten a focus group was constructed to review the final product. The Android APK was sent to all the participants the day before after all the appropriate forms had been read and signed. If the APK did not work for whatever reason for some of the participants a live demo was prepared to show them how it worked and what they could do in the app and if they wanted to try it they could take control of the screen and try it themselves on the emulator. The APK worked for everyone so this wasn’t necessary. Questions were prepared prior to the online meeting and each participant were cooperative. All the feedback was summarised, put into one page and shown to the participants to confirm it was the overall thoughts of the group. This resulted in much positive feedback such as the overall look and the app and the array of features but also constructive feedback with bugs they had noticed and things mentioned in the high fidelity prototypes that hadn’t been added to the app. One of the things brought up was that the app did work with no internet connection to show the fixtures and the player names but was unstable and would crash another being that I had not added to the predicted line up screen when the players were updated, which I had mentioned in my high fidelity prototypes. The focus group members were all friends of mine and all knew each other well. This made the meeting run smoothly as everyone was comfortable each other and didn’t need to be prompted to contribute as everyone knew each other so were not nervous. They were also more comfortable to criticise the product as they knew me and didn’t need to try and be polite to me with their criticism. Although I am aware since I knew the participants that this would add a level of bias as it could be argued since they knew me they would be more positive with their reviews. Overall the focus group provided much information into what had worked and things to consider for my future work.

**6.2 Starting 11 compare too [have to wait for these 2]**

**6.3 Algorithm**

**7 Description of the Final Product**

**8 Conclusion / Future Work**