

Fantasy Premier League

Assistant Android

App

**John Grossi**

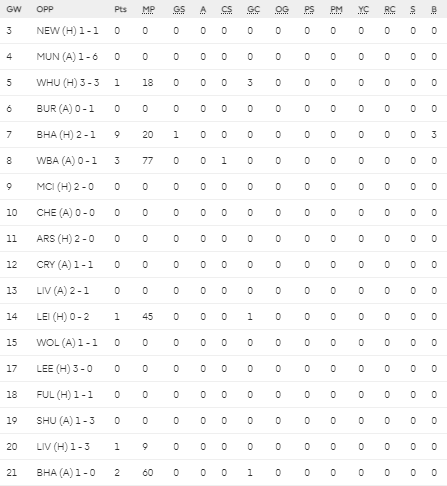
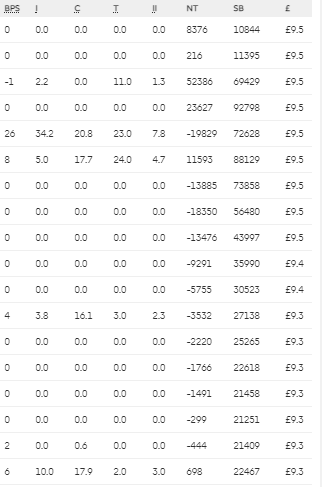
**AC40001 Honours Project**

**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.

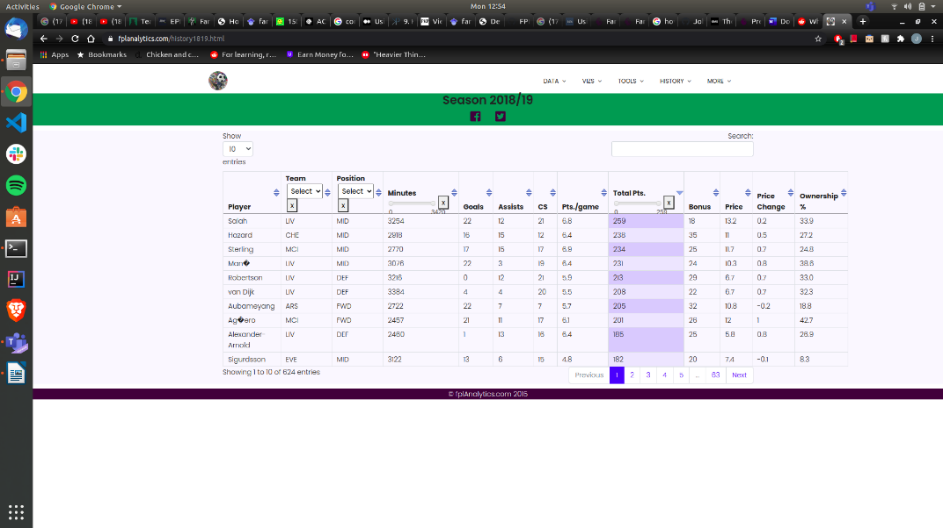
Here it shows players that play a full 90 minutes or even start the game but later get subbed off score more goals than players that get subbed on. So knowing who starts games can be vital to picking players for your team from defenders getting clean sheets and a more likely scenario of your forward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

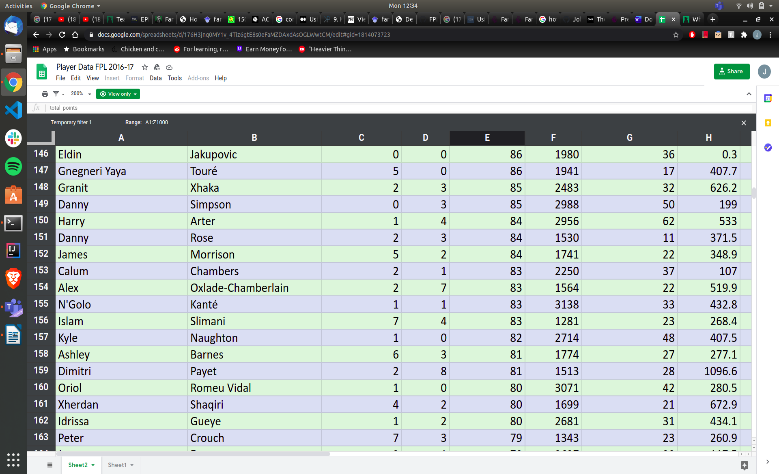


Figure 3: 16/17 FPL player season stats [5]

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 to the project. After much consideration an 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 doesn’t it made the most sense to go with it. After my requirements were created I turned them into user stories and held all of them in my sprint backlog[#]. 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. This is why I got user feedback multiple times before the software development 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 was 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 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. 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 chose 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 request 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 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 ti 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 time 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 couldn’t 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 it’s documentation was not as extensive, difficult 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 tracking**

Daily notes, meeting notes? (find a place to reference these

**4 Design**

**4.1 Requirements gathering(spec maybe)**

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:

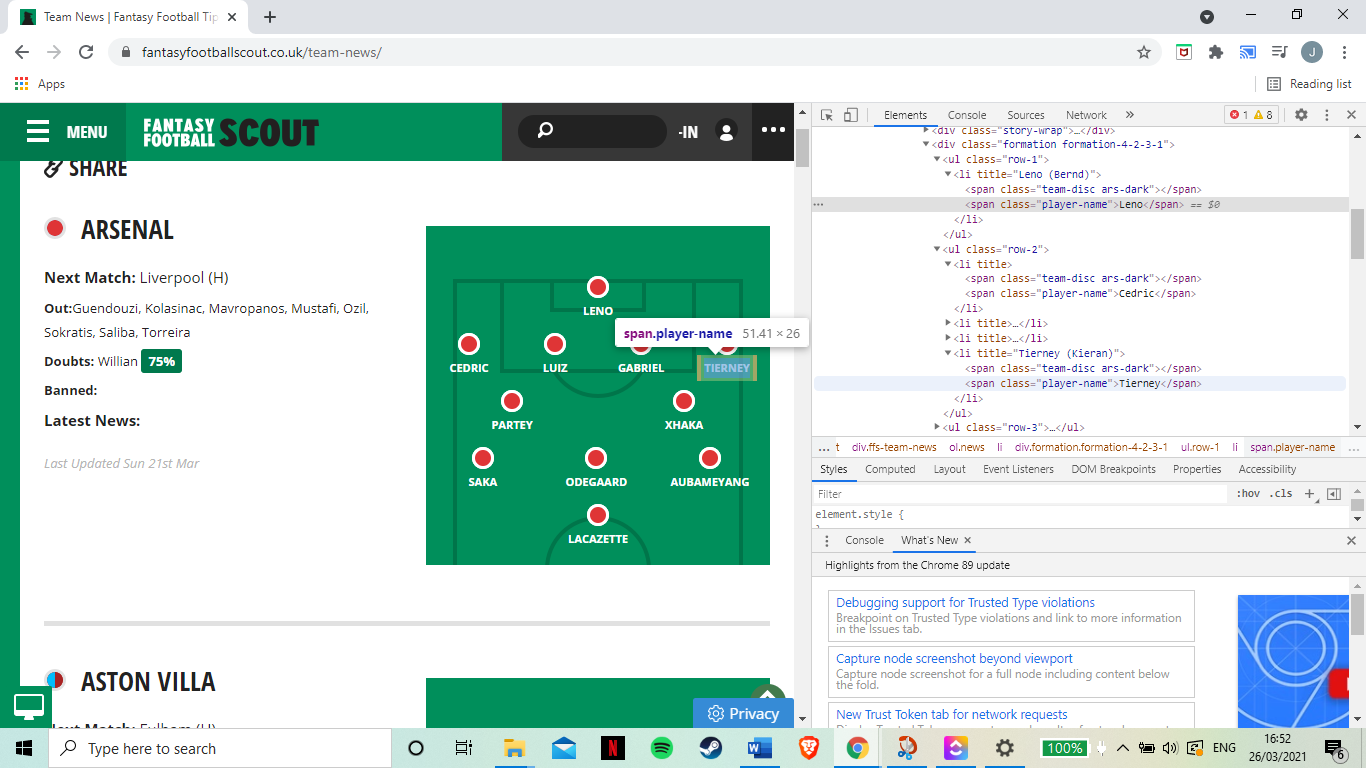
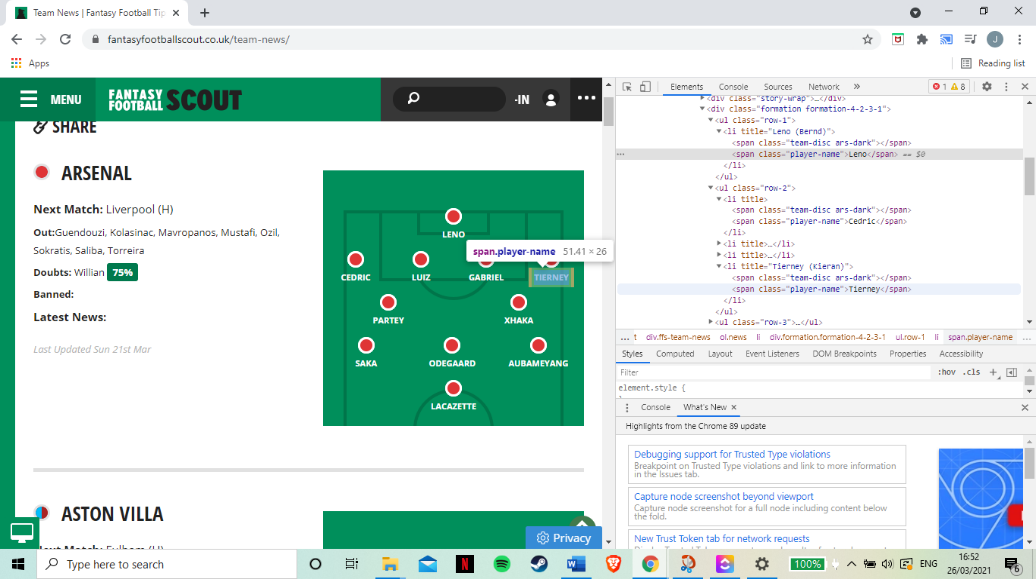
* This week’s fixtures
* Line-up predictor for each fixture
* Player comparison tool
* Highest scoring team predictor

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 Fedelity 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

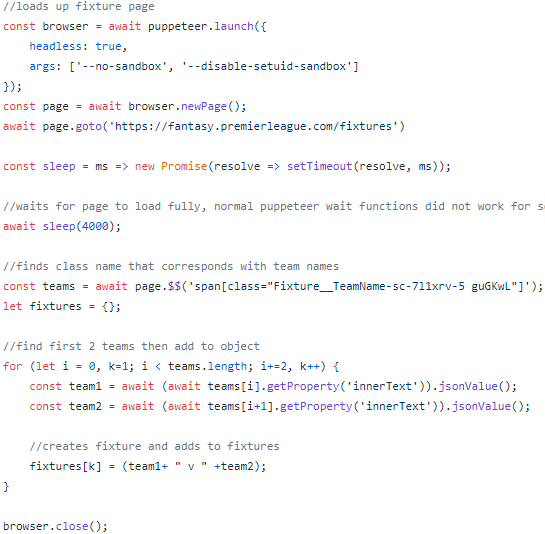
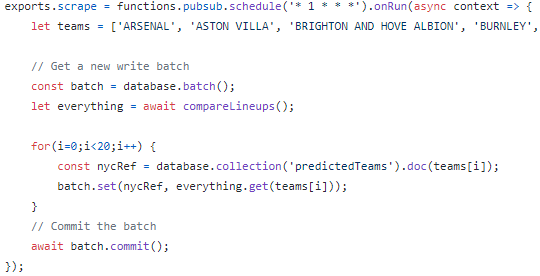
What the function does is it requests the webpage that contains all the team line-ups and loads the HTML. With the HTML it then goes through each player name under the “.player-name” tag, gets their last name and then adds it to a player array. Once it has done this it then sets a JavaScript object to 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. Unfortunate 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, front-end for the fixtures and predicted line ups fragments.

**5.5 Sprint 5**

**5.6 Sprint 6**

**5.7 Sprint 7**

**5.8 Sprint 8**

**5.9 Sprint 9**

**5.10 Sprint 10**

**6 Evaluation**

**6.1 Focus group**

**6.2 Starting 11 compare too**

**6.3 algorithm**

**7 Description of the final product**

**8 Conclusion / Future Work**