

# National College of Ireland

BSc(Hons) in Computing

2022

Ross Buckley

18501723@student.ncirl.ie

## Fantasy Futsal Application

## Technical Report

# Acknowledgements

I'd like to acknowledge and thank some people for the help they've gave me, over the past year and past four years.

Firstly, I'd like to say a big thanks to Frances Sheridan for being my supervisor for the final year project, meeting with me weekly and guiding me in the right direction any way she could. Frances helped me develop a suitable idea from my suggestions and helped by answering all the many questions I had throughout the year.

I would like to thank all the lecturers of NCI that have taught me over the last four years. Over the past few years, I've gained lots of experience and knowledge in all different areas of computing, thanks to the help of all these lecturers.

Finally, I'd like to thank all my classmates for making college life easy, offering support in group chats, and providing information on deadlines etc when needed.

## Contents

Executive Summary .....	4
1.0 Introduction .....	4
1.1. Background .....	4
1.2. Aims.....	4
1.3. Technology .....	5
1.4. Structure .....	6
2.0 System.....	7
2.1. Requirements.....	7
2.1.1. Functional Requirements.....	7
2.1.1.1. Requirement 1 User Registration.....	8
2.1.1.2. Description & Priority.....	8
2.1.1.3. Use Case .....	8
2.1.1.4. Requirement 2 User login .....	10
2.1.1.5. Description & Priority.....	10
2.1.1.6. Use Case .....	10
2.1.1.7. Requirement 3 User should be able to pick a team.....	12
2.1.1.8. Description & Priority.....	12
2.1.1.9. Use Case .....	12
2.1.1.10. Requirement 4 The user should be able to create a private league.....	14
2.1.1.11. Description & Priority.....	14
2.1.1.12. Use Case .....	14
2.1.1.13. Requirement 5 User should be able to view leagues .....	16
2.1.1.14. Description & Priority.....	16
2.1.1.15. Use Case .....	16
2.1.2. Data Requirements .....	17
2.1.3. User Requirements .....	18
2.1.4. Environmental Requirements .....	18
2.1.5 Non-Functional requirements.....	18
2.1.5.1 Performance .....	18
2.1.5.2 Reliability.....	18
2.1.5.3 Back-ups .....	18
2.2. Design & Architecture .....	19
2.2.1 System Diagram .....	19
2.2.2 Database design .....	19
2.3. Implementation .....	23

2.3.1 Project set up .....	23
2.3.2 Backend.....	26
2.3.2.1 Node modules and node.js set up .....	26
2.3.2.2 Rapid API's and requests.....	26
2.3.2.4 Data collecting .....	34
2.3.2.5 Machine learning model .....	37
2.3.3 Frontend.....	39
2.3.3.1 Firebase and authorisation .....	40
2.3.3.2 REACT components.....	40
2.4. Graphical User Interface (GUI).....	51
2.4.1 Component wireframes .....	51
2.4.2 GUI pages .....	54
2.5. Testing.....	60
2.6. Evaluation .....	60
3.0 Conclusions .....	61
4.0 Further Development or Research .....	61
5.0 References .....	62
6.0 Appendices.....	62
6.1 Objectives.....	64
6.2 Background .....	65
State of the Art.....	65
6.3 Technical Approach.....	66
6.4 Technical Details .....	66
6.5 Special Resources Required .....	67
6.6 Project Plan .....	67
6.7 Testing.....	69
6.8 Ethics Approval Application (only if required) .....	70
6.9 Reflective Journals .....	71
6.9.1 October .....	71
6.9.2 November .....	73
6.9.3 December.....	75
6.9.4 January .....	77
6.9.5 February .....	79
6.9.6 March .....	81
6.9.7 April.....	83

## Executive Summary

The Premier League is the most watched football league in the world. Having played football and been a fan since I'm 6 years of age, I find it interesting that there aren't many quality applications/websites that offer fun football games for all ages free of charge.

The main type of sports fun punters like to play is gambling. Obviously, this isn't suitable for all ages and has a negative effect on people who will lose their money and become addicted to the punt. My thinking was that there needs to be a suitable application that keeps users engaged for the full season of the Premier League where friends, family, work colleagues can all chip in and compete their football knowledge against each other's. An application that will keep users engaged for the full season from start to finish.

The best fun football website in the market right now would have to be Barclays Premier League Fantasy football. Without a doubt every year my work colleagues, football teammates, family and friends, all create private leagues for competition. And without a doubt every year after 4 or 5 weeks 90% of the league members have given up or forgotten about the league and its left with no activity. I believe this is down to some bad features implemented into the app, like strange prices for players which don't fall into users budgets and only having one free transfer a week when you have a squad of 14 players are just two examples.

The aim of Fantasy Futsal is to exploit the bad features of its competitor's apps/website and create a high quality site that will keep users of all ages engaged for the full Premier League season.

## 1.0 Introduction

The objective of this document is to highlight the reasoning, requirements, planning and implementation for the Fantasy Futsal application.

### 1.1. Background

After deciding to undertake a project which involved the football industry, I then chose to identify a problem in the industry and come up with a solution which I could implement for my project. The problem I found involves football players and teams being robbed of money from agents in the industry. In today's era, every footballer has an agent who negotiates transfer fees and contracts for the player. Time after time we've seen players transferring for crazy amounts of money all due to agents bumping up transfer fees for a cut of the fee. We also see players being given incredibly high salaries when they don't deserve near the sum. I want to create a machine learning model that predicts people's transfer worth and salaries using their statistics on the pitch. I believe this will be implemented at a higher scale in the future and will put an end to unnecessary fees in the football industry.

### 1.2. Aims

The main aim of the project is to create a fully functioning fun football game for people of all ages to use and enjoy. The project will contain many functions and offer security for the safety of user's data.

The aims are:

- Contain a login/register system with validations
- Be connected to a database with numerous tables
- Be responsive
- Look appealing and be styled
- Allow users to pick a new team of five players every game week of the Premier League
- Users will automatically be entered into public leagues
- Users should be able to create a private league
- Users should be able to view the standings of all the leagues they are in
- Users should be able to logout
- Predict accurate transfer prices/salaries for football players
- Have validations in place so users can't go over their transfer/salary budget
- Make asynchronous requests to API every week for updated stats
- Cookies implemented

### 1.3. Technology

Technologies	Description and use
IDE – Visual studio code	VS Code is a personal favourite for myself as it has lots of features that speed up the process of coding and it supports all the languages implemented throughout the project
Frontend - REACT JS	REACT is a frontend JavaScript library. This will make the website a single page that is made up of different components. The components will make up the different pages on the website.
Backend – Node.js	Node.js is a backend JavaScript library. Node will take care of all requests to API's and handle the server's requests from the database.
Database - Firebase	Firebase is a gem of a tool owned by Google. Firebase handles authentication from users registering/logging in, contain user, team and league tables in a database and host the website.
Machine Learning – Tensor Flow	Tensor flow will be the tool used to get predictions of transfer value and salaries of football players for the website.

Styling - Bootstrap	Bootstrap will be used to make the website responsive so it can be accessed on any device while looking good quality. Bootstrap will also be used for the forms needed in the website and navbar etc.
API's	Numerous API'S from Rapid API's will be used for many different things across the website. The API's will be used show the 20 premier league teams, search players, search squads, get stats etc.
CSV's	A CSV is used to gather all the transfers from the last year for all of the top football leagues of the World. The CSV 's is found on a GitHub Account and will be referenced later in the document.
Data handling - Python	Some Python is used to create data frames and then convert the data frames to CSV's as the machine learning model needs. Pythons panda library is used to carry out these functions.
AXIOS	AXIOS is the connection between frontend and backend, this allows me to send http requests with no interference.
Docker	Software platform that provides everything needed to run project, used due to numerous issues occurring with no other solution working.

#### 1.4. Structure

Section 1 of the report starts off with an introduction to the project ahead. This tells us a brief bit why I chose to implement my idea and how the idea of the project came about. Then, some of the aims of the project are outlined before going into detail about what technology and tools will be used to develop the project.

Section 2 will take up most of the document going into detail about the requirements, design, and implementation of the project. The requirements for the project are outlined, firstly the functional requirements with the aid of use case diagrams before data, user, environmental and usability requirements are discussed.

The design and architecture sub-section will be filled with a description of the design implemented, the architecture that will be developed with the aid of an architecture diagram and the algorithms that will be used in the project.

The implementation section will talk about the languages, functions, and algorithms. Some code snippets from the workspace will be used to explain better, how it will work. A lot will be discussed about the many problems I ran into throughout the development of the project which resulted in the project plan being pushed back numerous times.

The graphical user interface will show screenshots of all the pages/components of the website , and wireframes that were drawn up while planning in earlier stages of the project.

The testing section will explain how the project will be tested in various ways to secure it has no bugs and is working as should. This will show evidence of the project being built securely and show the right steps been taking to prove a phase of the project is complete, so that the next phase can be started.

The final part of the system section will be evaluation of the project. Evaluations of performance in terms of, is the project doing what it should? What is working? What didn't work? Is the project scalable? What would I do differently if I was to start the project again? All these questions will be answered with facts and figures in the evaluation section.

Section 3 will be the conclusion. Here the advantages and disadvantages of the project will be outlined. The strengths of the project over other similar applications will be made clear, along with the limitations of the project.

Section 4 covers further development and research, a section that can show where the project can develop in the future if there was extra time and resources on the project. Before creating my project plan, there was so many unique features and plans that could have been added to application but with limited time and resources, serious thought had to go into the structure of the plan to make sure there was time to develop the base of the project which is what is in the project plan now. The project has serious potential if there was no time limit and had more people working on it which will be discussed in this section of the report.

The Appendices section will be made up of a past project proposal report that was written before the project was started, any project ethic forms needed and monthly reflective journals on what work was done on the project previously.

## 2.0 System

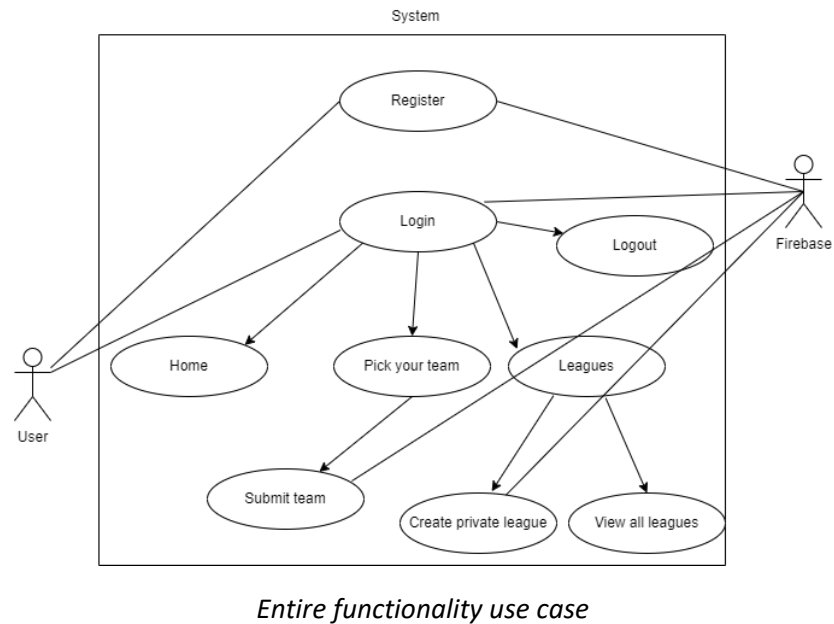
### 2.1. Requirements

Fantasy futsal is a fun application for everyone to use. All requirements that are specified in the next few sections should be quick and easy to use. An non-experienced controller should be able to guide their way through the app with little or no training required. The system will be designed so non-experienced users have little or no trouble using the site.

#### 2.1.1. Functional Requirements

- 1.The user must register first to be able to login to access the application.
- 2.The user must login to use the application.
- 3.The user should be able to pick a team.
- 4.The user should be able to create a private league.
- 5.The user should be able to view leagues.





#### 2.1.1.1. Requirement 1 User Registration

#### 2.1.1.2. Description & Priority

This use case describes the steps that should be taking by the user to register for the Fantasy futsal application and where there data is sent to be stored.

#### 2.1.1.3. Use Case

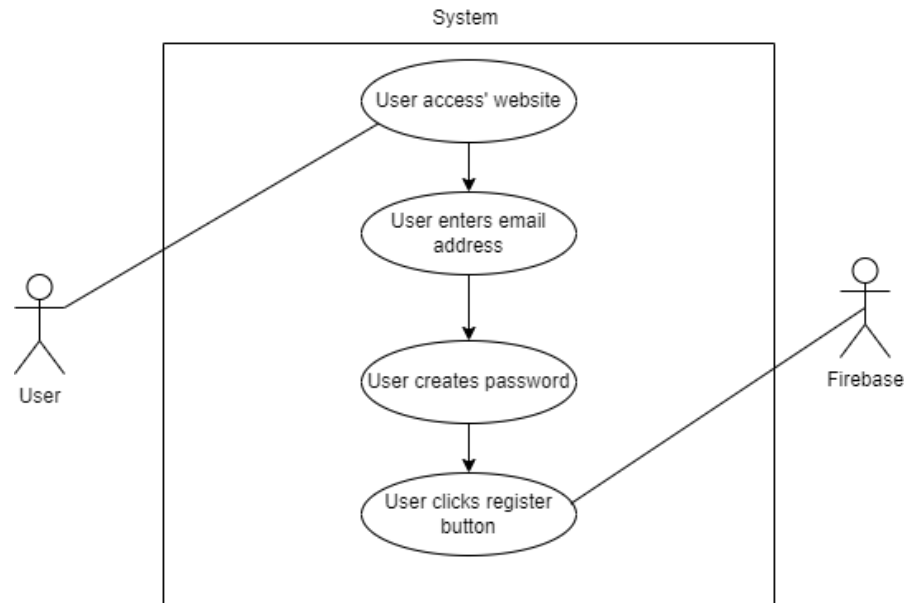
##### **Scope**

The scope of this use case is to get the user registered in the system, so they can then access the app by logging in.

##### **Description**

This use case describes the registration section of the application.

##### **Use Case Diagram**



## Flow Description

### Precondition

The system is in initialisation mode when the app is opened

### Activation

This use case starts when the user starts to type their email address in

### Main flow

1. The system identifies the user needs to register
2. The <User> enters there email address
3. The <User> creates a password
4. The <User> clicks the register button
5. The credentials are sent to be stored in the database and the user can then login using the details provided recently

### Alternate flow

A1 : Wrong details

1. The user enters an email address with no @ symbol
2. The <User> tries to enter the wrong credentials
3. The system will alert the <User> that they must enter an email address that contains the @ symbol

### Exceptional flow

E1 : Already registered

- 1The <User> enters their details
2. The <User> presses register
3. The system will alert the user they already have an account

### Termination

The system terminates when the user clicks the register button or exits the application

**Post condition**

The system goes into a wait state until the user logs in with the details they registered with.

2.1.1.4. Requirement 2 User login

2.1.1.5. Description & Priority

This requirement is essential for using the application. Without logging in the user cannot access any features of the application

2.1.1.6. Use Case

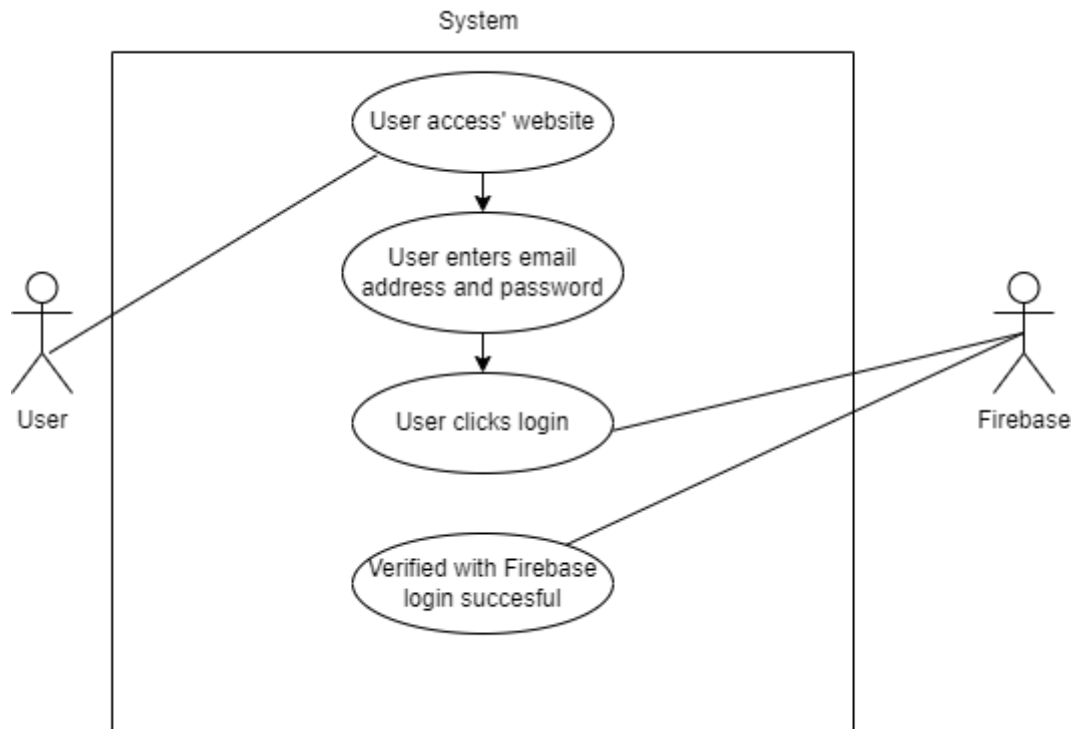
**Scope**

The scope of this use case is to get the user logged onto the application so they can use the features of the app.

**Description**

This use case describes how the user can log in, and how the users details are verified by Firebase (database).

**Use Case Diagram**



### Flow Description

### Precondition

The system is in initialisation mode when the app is opened

### Activation

This use case starts when an <User> enters their email address in the field

### Main flow

1. The system identifies the user is registered
2. The <User> enters the correct details the registered with
3. The <User> clicks the login button
4. The system logs the user in successfully

### Alternate flow

A1 : <Wrong login details>

1. The <User> enters a different password/email
2. The <User> clicks the login button

The system should alert the user that the email/password entered is wrong

### Exceptional flow

E1 : Not registered attempt at logging in

1. The <User> enters their email address and password
2. The <User> clicks the login button

### **Termination**

The system presents the next stage when the user is logged in and lands on the home page

### **Post condition**

The user is logged in successfully or isn't given access to the app

#### [2.1.1.7. Requirement 3 User should be able to pick a team](#)

#### [2.1.1.8. Description & Priority](#)

The main functionality of the application is to choose a team of 5 players from the Premier league who you think will accumulate the most points for that particular game week.

#### [2.1.1.9. Use Case](#)

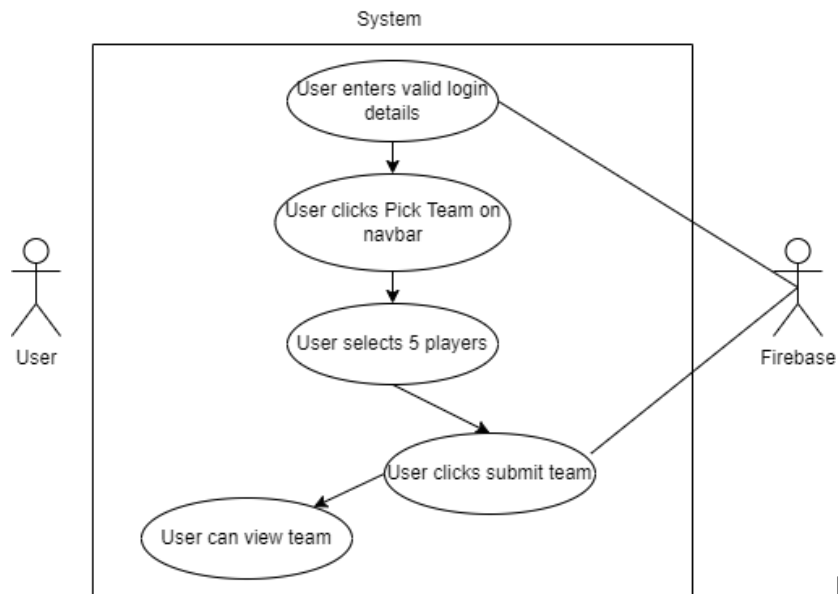
##### **Scope**

The scope of this use case is the main functionality of the app, to choose 5 players for your team to try to receive the highest points tally as possible to beat other competitors

##### **Description**

This use case describes the navigation to the "Pick Team" screen, where the user will select their 5 players of choice before submitting their team for the week

##### **Use Case Diagram**



### Flow Description

#### Precondition

The system is in initialisation mode when the app is opened

#### Activation

This use case starts when an <User> logs in successfully

#### Main flow

1. The <User> logs into the app successfully
2. The <User> clicks the "Pick Team" option on navbar
3. The <User> selects 5 player of their choice form the list of all Premier league players
4. The <User> is hits the "Submit" button to lock in their team for the game week

#### Alternate flow

A1 : <Doesn't press submit>

1. The user navigates through step 1,2,3 but doesn't press the submit button, the team wont be saved and the user could miss out on picking a team for the game week

#### Termination

The system presents the next stage when the user receives a notification that their team is locked in and they can view their submitted team.

#### Post condition

The system goes into a wait state for the Premier League game week to start. commence.

2.1.1.10. Requirement 4 The user should be able to create a private league

2.1.1.11. Description & Priority

High priority function, users will want to create private leagues with their friends for competition with each other. Work colleagues, college friends, any type of friend all love a fantasy league over competing against thousands of other people who use an application.

2.1.1.12. Use Case

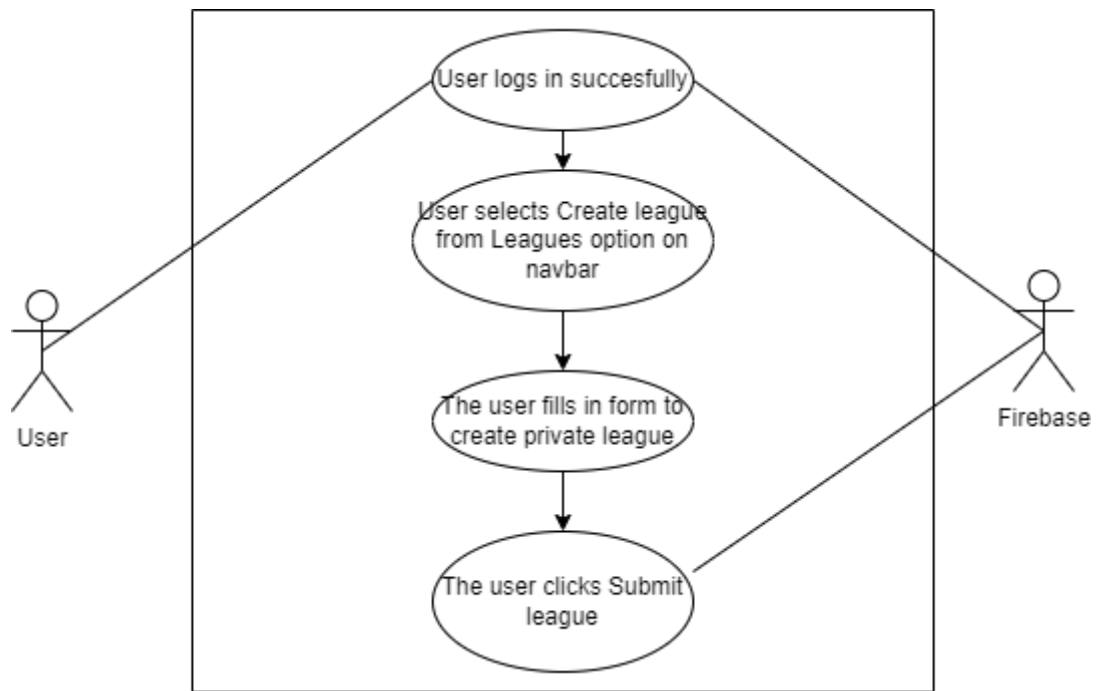
#### **Scope**

The scope of this use case is to allow users to create private leagues between them and some friends which can be closely monitored rather than checking on the public leagues to compare how they are doing against each other.

#### **Description**

This use case describes the process of setting up a private league.

#### **Use Case Diagram**



### Flow Description

#### Precondition

The system is in initialisation mode when the app is opened.

#### Activation

This use case starts when an <User> logs in successfully.

#### Main flow

1. The system identifies the <User> is logged in
2. The <User> then hovers over the "League" option and selects "Create League" from dropdown
3. The <User> fills in the "Create League" form and clicks "Submit"
4. The <User> can view the league and participants

#### Alternate flow

A1 : Fill form in incorrectly

1. The <User> is missing a required field or inputs wrong data into the field
2. The use case continues at position 3 of the main flow

#### Termination

The system presents the next stage when they receive a notification that their league is created successfully, and they can view other users in the league.

#### Post condition

The system goes into a wait state for the Premier League game week to start.



#### 2.1.1.13. Requirement 5 User should be able to view leagues

#### 2.1.1.14. Description & Priority

Medium priority, this function gives users a visual to see how they are doing competing against other people whether a private or public league. A list of leagues will be seen on the “View Leagues” page.

#### 2.1.1.15. Use Case

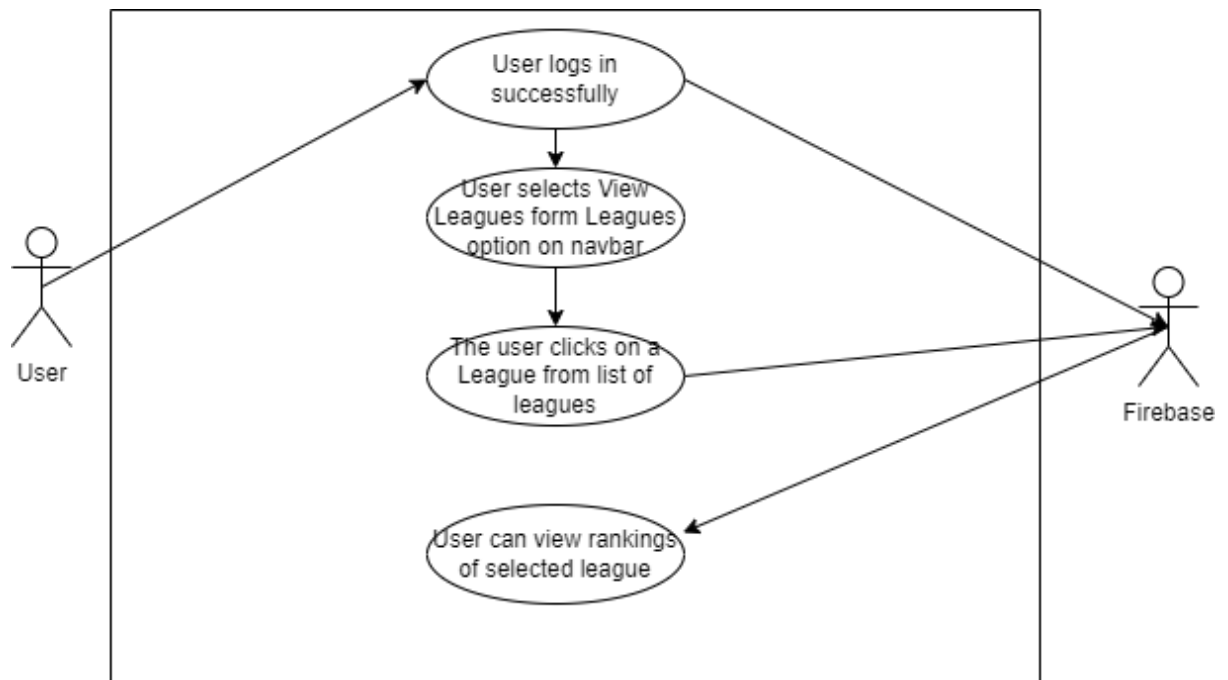
##### **Scope**

The scope of this use case is to allow the user to see visually how the leagues they are participating in are standing.

##### **Description**

This use case describes the how to navigate to view the leagues the user has joined or any public league.

##### **Use Case Diagram**



### Flow Description

### Precondition

The system is in initialisation mode the user opens the app

### Activation

This use case starts when an <User> logs into the app.

### Main flow

1. The system identifies the <User> is logged in successfully
2. The <User> clicks “View leagues” option from the navbar
3. The <User> clicks on one of the leagues they are in form the list
4. The <User> can see the league table

### Termination

The system presents the next action when the user selects a different option from the navbar or logs out.

### Post condition

The system goes into a wait state

## 2.1.2. Data Requirements

The only private data Fantasy Futsal will take from the user is there email address and a unique password they chose themselves. This will be the first thing the user needs to do upon opening the website. When the user registers for the site, their details will be saved and secure in a Firebase database. Without a valid email address containing @ symbol the user will not be able to register successfully resulting in not being able to use all the functionality the website offers.

Firebase will handle authentication so that an imposter can't login using your email address with an incorrect password.

### 2.1.3. User Requirements

To join in on the fun of the Fantasy futsal application, the user is required to register by filling in their email address and creating a password or using the login with Google option provided before logging in to be able to play on the site. When the user registers their details are stored in the Firebase database so they can login using their credentials at any time on any device.

The user must have access to internet to access the website.

I would recommend the user should have at least some knowledge of football, that is if they want to compete in the leagues they have entered.

### 2.1.4. Environmental Requirements

This section explains the environmental requirements needed to use the application. Fantasy Futsal can be accessed from any device once the user has internet.

### 2.1.5 Non-Functional requirements

#### 2.1.5.1 Performance

The performance time of the website must be sharp to keep users interested. As REACT uses components to show/hide pages instead of actual HTML pages, this is an advantage in the response time area as the website doesn't have to render an entire new page when the user selects a different link. Cookies and caches are also implemented to save data, so the user isn't requesting new information all the time.

#### 2.1.5.2 Reliability

All the data used to predict transfer and salary values for players is taken from real data. The data is taken from the top 900 football teams in the world. The API used from Rapid API is updated weekly making it an ideal source for data and reliable.

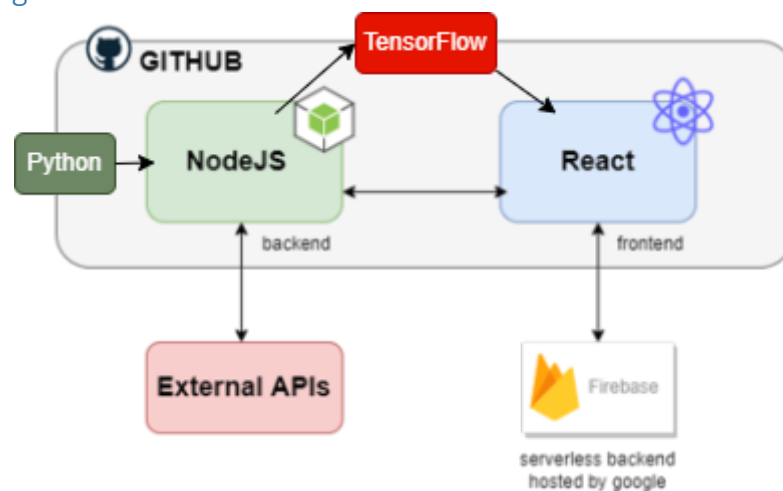
In terms of connection, reliability will vary depending on how strong the user's connection to Wi-Fi is.

#### 2.1.5.3 Back-ups

The code will be stored on GitHub as well as locally. Firebase offers a cloud platform if security ever feels a threat.

## 2.2. Design & Architecture

### 2.2.1 System Diagram



*System architecture Diagram*

The figure above shows the main components that make up the system of the Fantasy Futsal website. The diagram gives a clear visual of how all the technologies communicate with each other. The external API's and Firebase data can't be shared on GitHub for breach of data reasons. The rest of technology code can be pushed to GitHub for reliability and re-usability. The backend is wrapped in a Docker container as a way to install tensor flow as there was issues with the installation, this was added late, hence why it isn't in the system diagram.

### 2.2.2 Database design

Firebase and mongoose are the databases for the project and has made life extremely easy when connecting to the database and the handling of data. The databases will consist of four tables. The user, team and league tables in the database will consist of the user's input taken from the end user while they are using the website. The player's table is populated with football players data and statistics pushed from a CSV file for the purpose of the machine learning model and for data availability. Mongoose was added very late in the project, as Firebase kept maxing limits of data. This was due to the high volume of data needed for the machine learning model.

Communication between the two databases had to be configured. This was down to the fact that the Firebase database handled authentication and I needed to pull user IDs to match who was pushing teams and creating leagues to the Mongoose database.

Each schema is designed in a separate file in the model's folder. The snapshots below give a better visual on the fields in each of the schemas.

### Users table

The users table contains five fields. Id, name, photo, email and created. The type of data being stored, and requirement status can also be seen underneath the field headers.

```
2 //User data saved in DB
3 const userSchema = new mongoose.Schema({
4   _id: {
5     type: String,
6     required: true,
7     unique: true
8   },
9   name: {
10    type: String,
11    required: true
12  },
13  photo: {
14    type: String,
15    required: true
16  },
17  email: {
18    type: String,
19    required: true
20  },
21  created: {
22    type: Date,
23    default: Date.now
24  }
25 });
26
```

*Users' schema*

### Team table

The team schema contains ten fields. It holds the user ID of the user saving the team, five spots for the team itself, a completed status, transfer value, how many points the team accumulated and when the team was created.

```
2 // team data saved to DB
3 const teamSchema = new mongoose.Schema({
4   user_id: {
5     type: String,
6     required: true,
7     unique: true
8   },
9   playerOne: {
10    type: Object,
11    required: true
12  },
13  playerTwo: {
14    type: Object,
15    required: true
16  },
17  playerThree: {
18    type: Object,
19    required: true
20  },
21  playerFour: {
22    type: Object,
23    required: true
24  },
25  playerFive: {
26    type: Object,
27    required: true
28  },
29  completed: {
30    type: Boolean,
31    required: true
32  },
33  transferValue: {
34    type: Number,
35    default: 0
36  },
37  points: {
38    type: Number,
39    default: 0
40  },
41  created: {
42    type: Date,
43    default: Date.now
44  }
45 });
```

*Team schema*

### League table

The league schema contains seven fields. League name, members, number of members, public, owner ID (which is equal to the user ID of the creator), code and created.

```

2 //League data saved to DB
3 const leagueSchema = new mongoose.Schema({
4   league_name: {
5     type: String,
6     required: true,
7     unique: true
8   },
9   members: {
10    type: Array
11  },
12  number_of_members: {
13    type: Number
14  },
15  public: {
16    type: Boolean,
17    required: true
18  },
19  owner_id: {
20    type: String,
21    required: true
22  },
23  code: {
24    type: String,
25    unique: true
26  },
27  created: {
28    type: Date,
29    default: Date.now
30  }
31 });

```

*League schema*

### Players table

The players schema contains ten fields. This contains the largest amount of data about the football players and their statistics.

```

2 //Player data saved to DB
3 const playerSchema = new mongoose.Schema({
4   p_id: {
5     type: Number,
6     required: true,
7     unique: true
8   },
9   name: {
10    type: String,
11    required: true
12  },
13   number: {
14     type: Number,
15   },
16   photo: {
17     type: String,
18   },
19   position: {
20     type: String,
21   },
22   teamName: {
23     type: String,
24     required: true
25   },
26   teamId: {
27     type: Number,
28     required: true
29   },
30   transfer: {
31     type: Object
32   },
33   statistics: {
34     type: Object
35   },
36   extraData: {
37     type: Object
38   }
39 });

```

*Player's schema*

## 2.3. Implementation

This phase of the project took the longest and was completed over seven months starting in November and finishing in May. The implementation was broken up into three main sections. The backend, frontend and data collecting. Data collecting is described as its own section in terms of implementation but will be under the backend section of the implementation section. The reasoning behind naming it a section is because how much time and work went into it, which will be clearly visible in the following sections.

The project plan laid out a foundation for the implementation phase. Due to difficulties in the project some of the due dates needed to be moved around to progress somehow week by week. The plan was to set up the project in two parts for separation of concerns, frontend, and backend. The backend was to be started in the first semester and finished first early second semester. The frontend would then take over before joining the two.

Due to many difficulties gathering data, the frontend was started early to make up the time lost with the setbacks. The backend was started in November and finally completed in May while the frontend was started late January and finished in May.

With many problems arising when trying to implement the tensor flow models, a quick solution was needed to get around some issues that kept occurring. The project was wrapped in a Docker container to bypass these issues. Docker is software platform that provides everything your project needs to run it. (What is Docker? | AWS, 2022) Being unsure why my project was running into errors as documentation was followed to a tee, Docker was the ideal solution and worked first time.

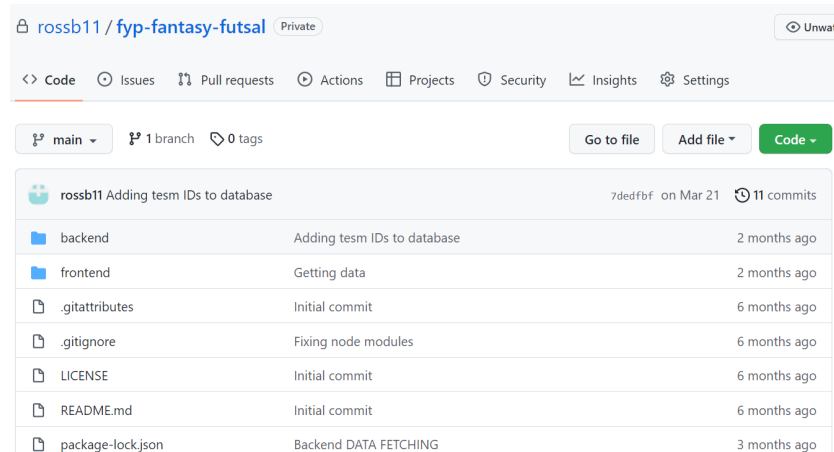
### 2.3.1 Project set up

Setting up the project consisted of creating a new project folder in Visual studio code, the text-editor used to develop the project. Two separate folders were created in the project so the frontend and backend could be worked on separately. This is a quality feature that VS code offers as developers can work on two different folders within the project without any confliction occurring. (Code?, 2022)

The next stage of setting up involved creating a GitHub repository and connecting to the project so that code can be pushed up for back up. GitHub is one of the most popular collaboration tools used in tech companies all over the world. GitHub commits were made throughout the project, saving the



work done after each small phase. This made it easy to track the last piece of work done and continue where I left off.

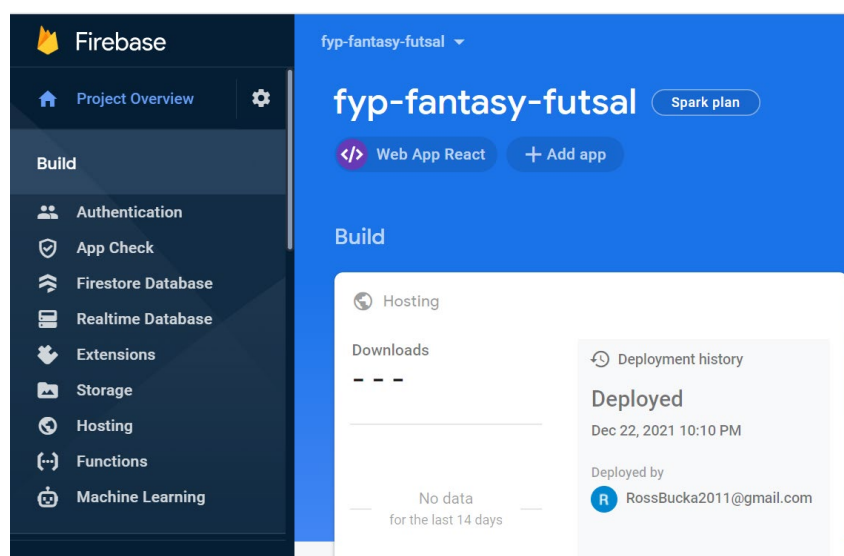


The screenshot shows the GitHub repository page for 'fyp-fantasy-futsal' by user 'rossb11'. The repository is private and has 11 commits. The commit history table is as follows:

Commit	Message	Time
rossb11	Adding testm IDs to database	2 months ago
	Adding testm IDs to database	2 months ago
	Getting data	2 months ago
	Initial commit	6 months ago
	Fixing node modules	6 months ago
	Initial commit	6 months ago
	Initial commit	6 months ago
	Initial commit	6 months ago
	Backend DATA FETCHING	3 months ago

*Final year project Fantasy Futsal GitHub repository*

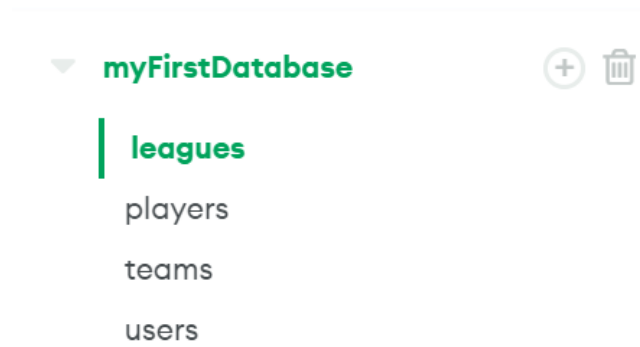
The last phase of setting the project up was creating the database. Firebase was chosen after some research on databases was carried out. Firebase is a Google owned cloud-based system. Firebase handles things like authentication and offers real-time database feature which allow developers to whip up databases quick and easy. (Advantages and Disadvantages of Firebase, 2022) Once the database was set up everything was ready to start implementing the project.



*Fantasy Futsal Firebase database*

Due to data storage limits been hit, I needed to migrate to another database to store large chunks of data for the machine learning model. MongoDB was the chosen database. MongoDB is a cloud-based application that is known to work best with the JavaScript language. One reason Mongo was

picked for the project was it is widely documented. Having never used it before this was a key reason for the choice along with some of the other quality features it offers. (Advantages Of MongoDB, 2022) Once the database was set up everything was ready to start implementing the project.



*Mongoose database with all existing tables*

### 2.3.2 Backend

The backend of the project was started first, as it was expected to take the longest amount of time to develop. As mentioned earlier in the report, JavaScript and Node.js is the language and library selected to implement the server-side of the project. Node.js is a backend JavaScript library. Node allows us to handle requests from the frontend and sends responses. (What is Node.js And How It Works, 2022) Having used Node.js before in other projects, I thought it would be a good idea to implement the entire project using JavaScript.

This part of the implementation included setting up node modules, data collecting, creating a machine learning model, connecting, and making requests to API's and posting data to the databases.

#### 2.3.2.1 Node modules and node.js set up

The first step taking to setup the backend was running the “sudo apt install npm” in the backend directory. Having Node already on my device, next was running the “npm init” command. This command will set up the app.js and package.json files which generate code for the local server. The express package is used to handle the https requests to navigate to different URL's.

In the end the node modules was wrapped in the Docker container, so the logs are in Docker when the backend is started.

#### 2.3.2.2 Rapid API's and requests

Rapid APIs provided me with a large amount of the data needed in the project. Numerous different APIs are used for different functions on the website. A list of the APIs used can be seen below in figure.

```
fyp-fantasy-futsal > backend > routes > JS index.js > ...
13
14 // Team API routes
15 router.get('/team', auth, team.getTeam);
16 router.post('/team/create', auth, team.postTeam);
17 router.delete('/team/delete', auth, team.deleteTeam);
18
19 // Players API routes
20 router.get('/players/:id', auth, players.getPlayers);
21 router.get('/player/:id', auth, players.getPlayer);
22 router.get('/player-squads/:id', auth, players.getPlayerSquads);
23 router.get('/player/search/:value', auth, players.getSearchPlayer);
24 router.get('/training-data', players.getTrainingData);
25
26 router.get('/standings', auth, league.getStandings);
27
28 // League API routes
29 router.get('/league', auth, league.getLeague);
30 router.get('/league/:id', auth, league.getLeagueById);
31 router.get('/all/league', league.getAllLeagues);
32 router.post('/league/create', auth, league.postLeague);
33 router.post('/league/join/:id', auth, league.joinLeague);
34 router.delete('/league/delete', auth, league.deleteLeague);
35
36 // User API routes
37 router.get('/users', auth, user.getUsers);
```

*Rapid APIs used*

The first routes seen in the figure is the CRUD functionality for the team on the website. Next is the API's used within the project for things like getting the current premier league teams for this years 2021/2022 season, getting players, getting squads and getting statistics. The League routes shown is the CRUD for leagues created on the application. Lastly the users get route for pulling in users when creating a league, this allows the owner of the league to add users to the league when there filling in the create league form. The CRUD for team and league will be described below.

### League POST, GET, DELETE

The league POST request sends the inputted information to the MongoDB database to be stored. The league object seen contains all the data that will be sent before saved in the database. If the league is not public it will need a code for the private league, this is the final field seen in the object.

```
module.exports.postLeague = async (req, res) => {  
  const league = new League({  
    league_name: req.body.league_name,  
    members: req.body.members,  
    number_of_members: req.body.members.length,  
    public: req.body.public,  
    owner_id: req.user.uid,  
    code: !(req.body.public) ? uuidv4() : null  
  });  
  
  await league.save();  
  
  res.status(201).json({  
    msg: 'League Successfully Created',  
    type: 'success'  
  });  
};
```

#### *League POST request*

Another function developed in the POST league file is to allow the user to join a league. If the user clicks the join league button their data will be posted to the league. Validations for private league codes can be seen, the user must enter the correct code to successfully join the league.

```

23 module.exports.joinLeague = async (req, res) => {
24
25   League.findOne({_id: req.params.id}, async function (err, result) {
26     if (err) { console.error(err) }
27     if (!result) {
28       res.status(200).json({});
29     }
30     if (result) {
31       console.log(`League's being fetched for ${req.user.email}`);
32
33       console.log(req.body.password);
34       console.log(result.code);
35       if (req.body.password !== result.code) {
36         return res.status(201).json({
37           msg: 'Incorrect password for ${result.league_name} reach out to the admin for the
38             type: 'incorrect'
39         });
40       }
41       let data = result.members;
42       data.push(req.user.uid);
43       result.members=data;
44       result.number_of_members=data.length;
45       await result.save();
46       return res.status(201).json({
47         msg: 'Team Successfully Created',
48         type: 'success'
49       });
50     }
51   });

```

### *POST join league*

Next is the GET leagues file. The functions used in here are to display a league on its own, all leagues and leagues the user is a part of. The screenshots below give a better visual of how these were implemented.

```

16 //Gets league
17 module.exports.getLeague = async (req, res) => {
18   League.find({}, async function (err, result) {
19     if (err) { console.error(err) }
20     if (!result) {
21       res.status(200).json({});
22     }
23     if (result) {
24       console.log(`League's being fetched for ${req.user.email}`);
25       let data = [];
26       result.forEach(league => {
27         if (league.members.includes(req.user.uid) || league.owner_id === req.user.uid) {
28           data.push(league);
29         }
30       });
31       res.status(200).json(data);
32     }
33   });
34 };

```

### *Get leagues*

```

36 module.exports.getLeagueById = async (req, res) => {
37   League.findOne({_id: req.params.id}, async function (err, result) {
38     if (err) { console.error(err) }
39     if (result.length === 0) {
40       res.status(200).json({});
41     }
42     if (result) {
43       if (result.members.includes(req.user.uid) || result.owner_id === req.user.uid) {
44         users = result.members;
45         users.push(result.owner_id);
46         User.find({_id : { "$in" : users}}, async function (err, userResult) {
47           if (err) { console.error(err) }
48
49           Team.find({user_id : { "$in" : users}}, async function (err, teamResults) {
50             let teamData = [];
51             if (err) { console.error(err) }
52             teamData.push(teamResults);
53             let league = {
54               ...result._doc,
55               users: userResult,
56               teams: teamData
57             }
58             return res.status(200).json(league);
59           });
60         });
61       } else {
62         return res.status(400).json({});
63       }
64     }
65   });
66 }

```

*Get league by ID*

```

68 module.exports.getAllLeagues = async (req, res) => {
69   League.find({}, async function (err, result) {
70     if (err) { console.error(err) }
71     if (!result) {
72       res.status(200).json({});
73     }
74     if (result) {
75       let data = [];
76       result.forEach(league => {
77         data.push(league);
78       });
79       res.status(200).json(data);
80     }
81   });
82 };

```

*GET all leagues*

The final league route is the delete. This function will remove any league the user doesn't want anymore. `Await result.remove()` is the function used to delete the league.

```

fyp-fantasy-futsal > backend > controller > league > JS delete.js > ...
1  const League = require('../../models/League');
2
3  module.exports.deleteLeague = async (req, res) => {
4    League.findOne({_id: req.body.id, owner_id: req.user.uid}, async function (err, result) {
5      if (err) { console.error(err) }
6      console.log(result);
7      console.log(`League being deleted for ${req.user.email}`);
8      await result.remove();
9      res.status(200).json({
10        msg: 'League Successfully Deleted',
11        type: 'success'
12      });
13    });
14  };
15

```

*Delete league*

### Team POST, GET, DELETE

The team POST request sends the users picks to the database to be saved for the coming weeks games. In this file, we see the five players being selected before their scores for the week being tallied up. Each player's score is added to the next before totalled. This is done by calling the calculate player function which is shown below.

```

34 const calculatePlayer = async (player) => {
35   console.log(player);
36   let points = 0;
37   if (player.value !== null) {
38     options.url = 'https://api-football-v1.p.rapidapi.com/v3/players';
39     options.params = {id: player.p_id, season: '2021'};
40
41     try {
42       const {data} = await axios.request(options);
43       let players = data.response[0];
44       console.log(players);
45
46       points = points + Math.floor(players.statistics[0].games.minutes !== null ? players.statist
47       points = points + Math.floor(players.statistics[0].goals.assists !== null ? players.statist
48       if (player.position === "Goalkeeper" || player.position === "Defender") {
49         points = points + Math.floor(players.statistics[0].goals.total !== null ? players.statist
50       } else if (player.position === "Midfielder") {
51         points = points + Math.floor(players.statistics[0].goals.total !== null ? players.statist
52       } else if (player.position === "Attacker") {
53         points = points + Math.floor(players.statistics[0].goals.total !== null ? players.statist
54       }
55       points = points + Math.floor(players.statistics[0].penalty.saved !== null ? players.statist
56       points = points - Math.floor(players.statistics[0].cards.yellow !== null ? players.statist
57       points = points - Math.floor(players.statistics[0].cards.red !== null ? players.statist
58     } catch (error) {
59       console.error(error);
60     }
61   }
62   return points;
63 }

```

*Calculate player in POST.js team file*

```

5 module.exports.postTeam = async (req, res) => {
6   Team.findOne({user_id: req.user.uid}, async function (err, result) {
7     if (err) { console.error(err) }
8     if (result) {
9       result.playerOne = req.body.playerOne;
10      result.playerTwo = req.body.playerTwo;
11      result.playerThree = req.body.playerThree;
12      result.playerFour = req.body.playerFour;
13      result.playerFive = req.body.playerFive;
14
15      let points = 0;
16      points = points + await calculatePlayer(req.body.playerOne.value);
17      points = points + await calculatePlayer(req.body.playerTwo.value);
18      points = points + await calculatePlayer(req.body.playerThree.value);
19      points = points + await calculatePlayer(req.body.playerFour.value);
20      points = points + await calculatePlayer(req.body.playerFive.value);
21      console.log(points);
22      result.points = points;
23
24      console.log(`Team being created for ${req.user.email}`);
25      await result.save();
26      res.status(201).json({
27        msg: 'Team Successfully Created',
28        type: 'success'
29      });
30    }
31  });
32 }

```

*Team POST*

The GET team saves the team selected by the user, before pulling the team to display on the UI.



```

3  module.exports.getTeam = async (req, res) => {
4      Team.findOne({user_id: req.user.uid}, async function (err, result) {
5          if (err) { console.error(err) }
6          if (!result) {
7              const team = new Team({
8                  user_id: req.user.uid,
9                  playerOne: {
10                     value: null
11                 },
12                 playerTwo: {
13                     value: null
14                 },
15                 playerThree: {
16                     value: null
17                 },
18                 playerFour: {
19                     value: null
20                 },
21                 playerFive: {
22                     value: null
23                 },
24                 completed: false
25             });
26
27             console.log(`Team being created for ${req.user.email}`);
28             await team.save();
29             res.status(200).json(team);
30         }
31         if (result) {
32             console.log(`Team being fetched for ${req.user.email}`);

```

#### *GET team*

If the user wishes to delete their entire team, they can do so easily with the delete button. Below is the code to remove the team.

```

fyp-fantasy-futsal > backend > controller > team > JS delete.js > ...
1  const Team = require('../models/Team');
2
3  module.exports.deleteTeam = async (req, res) => {
4      Team.findOne({user_id: req.user.uid}, async function (err, result) {
5          if (err) { console.error(err) }
6          console.log(result);
7          console.log(`Team being deleted for ${req.user.email}`);
8          await result.remove();
9          res.status(200).json({
10             msg: 'Team Successfully Deleted',
11             type: 'success'
12         });
13     });
14 };
15

```

#### *Delete team*

### **Player GET**

The player get.js file was one of the first files set up to call the APIs. Inside this file contains get player, get players squad, get players, get search, and get training data. All these requests are used in different locations around the website. The search player is used on the pick team page to allow users to search for the player they would like to pick for their team. The get training data loops through the players, if a transfer value and statistics are available for the player it runs the function. If either of these are missing it will not be used for the machine learning model.

```
58 module.exports.getSearchPlayer = async (req, res) => {
59   const val = req.params.value;
60   options.url = `https://api-football-v1.p.rapidapi.com/v3/players`;
61   options.params = {search: val, league: '39', season: '2021'};
62
63   try {
64     const {data} = await axios.request(options);
65     let players = data.response;
66
67     players = await attachTransferValue(players);
68
69     res.status(200).json(players);
70   } catch (error) {
71     console.error(error);
72   }
73 };
74
```

*GET search*

```
75 module.exports.getTrainingData = async (req, res) => {
76   Player.find({transfer: {$exists: true}, statistics: {$exists: true}}, function (err, data) {
77     if (err) {
78       res.status(400).send(err.message);
79     }
80     res.status(200).json(data);
81   });
82 };
```

*GET training data*

**Users GET**

The users GET is used to pull users onto the UI so that they can be added to leagues. This can be seen when creating a league, a list of the app's users will be underneath the league name field.

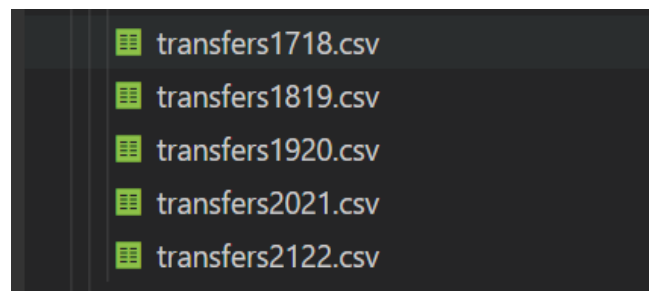
```
3 module.exports.getUsers = async (req, res) => {
4   // Setting up the call to external API with options / URL
5   User.find({}, async function (err, result) {
6     if (err) { console.error(err) }
7
8     let data = [];
9     result.forEach(element => {
10      if (req.user.uid !== element._id) data.push(element);
11    });
12    console.log(`Users being fetched for ${req.user.email}`);
13
14    res.status(200).json(data);
15  });
16 };
17
```

*GET users*

#### 2.3.2.4 Data collecting

Data collecting was the trickiest part of the project to implement. A lot of breaks were needed after each step to rethink about how to go about the next step. In the end two different types of data needed to be collected and joined to make up the training data. The Rapid API used for players gave me all the statistics needed for all players but didn't contain the transfers over the last year. As the application is based on the Premier League which contains the highest transfers recorded, I needed data from top leagues with high transfers occurring so the model would predict accurate values.

After a while I found a perfect CSV file that can be found on GitHub of transfers of every year dating back to 1992. I decided the last five years of transfer would be best suitable with the rise in transfers that have been seen. The CSV's used are saved and can be seen in the csv-data folder.



*CSV data*

Once we had the data needed, the next step was to join the statistics and transfer fees so the training data was ready to go.

In the data-collection folder is where the data is manipulated before joined using python. The fetch transfers file reads in the CSV before removing some data so that it isn't inconsistent. With free transfers and loan moves occurring the first step was to remove these as they would throw off the model when predicting.

```

13 fastCsv
14   .parseStream(readableStream)
15   .on("error", (error) => {
16     console.log(error);
17   })
18   .on("data", async (row) => {
19     if (row[5].includes('£') && !row[5].includes('Loan')) {
20       data.push(row);
21     }
22   })
23   .on("end", () => {
24     console.log(data.length);
25     parseData(data);
26   });
27

```

*Fetch transfer, making sure a fee is paid (£) before removing loan moves*

The next obstacle was cleaning up the fee so all fees were the same format. The snapshot below shows how this was done.

```

function cleanFee(transferValue) {
  return transferValue.includes('m') ?
    parseInt(parseFloat(transferValue.replace('m', '').replace('£', '')) * 1000000) :
    parseInt(parseFloat(transferValue.replace('Th.', '').replace('£', '')) * 10000)
}

```

*Clean fee function for consistency*

The fetch statistics file is self-explanatory, this pulls in all the statistics of players with a transfer fee, before saving the data collected.

```

9 Player.find({transfer: {$exists: true}}, async function (err, docs) {
10   if (err){
11     console.log(err);
12   }
13   else{
14     for(const player of docs) {
15       options.url = `https://api-football-v1.p.rapidapi.com/v3/players`;
16       options.params = {id: player.p_id, season: player.transfer.year.split("/")[0]};
17       try {
18         const {data} = await axios.request(options);
19         const extras = data.response[0].player;
20         delete extras.id;
21         delete extras.name;
22         delete extras.photo;
23         player.extraData= extras;
24         player.statistics = data.response[0].statistics[0];
25         await player.save();
26         console.log(`Stats fetched for ${player.p_id}`);
27       } catch (error) {
28         console.error(error);
29       }
30     }
31   }
32 });

```

*Fetch statistics*

After the statistics and transfers were collected and joined the data was saved as a JSON file before being pushed to the database and stored in a players table.

```
teamId: 38  
> transfer: Object  
> statistics: Object  
> extraData: Object  
__v: 0
```

*MongoDB, transfer, and statistic data showing*

### 2.3.2.5 Machine learning model

TensorFlow is the machine learning tool that predicts transfer values for players in the Fantasy Futsal website. Tensor flow is regarded the best machine learning tool on the market now, due to Google's high search engine. (Top 10 Best Machine Learning Tools for Model Training - neptune.ai, 2022) The Google owned tool is backed up with large amounts of documentation making it quick and easy to learn.

Once the data was pre-processed, as explained in the data collecting section, it was time to build and train the model to predict transfer values. The first step was installing TensorFlow. Unfortunately, it wasn't a great start to creating my models as I ran into numerous issues trying to install TensorFlow. After some time trying to de-bug the error, nothing was working. This is when I decided to wrap the project in a Docker container. Thankfully once the project was imaged on Docker, TensorFlow installed successfully on the first try.

At the start the model was throwing some crazy predictions. This was down to the statistics and positions of players. A decision was made to create four models to tune them more accurate. A model for every position was created, goalkeeper, defenders, midfielders, and attackers.

The machine learning can be seen in the TensorFlow folder in the backend. This contains a tensorflow.js and train.js file. The tensorflow.js creates the models, starts the machine learning, defines the 2d arrays, contains configurations, shows the batch size used and how many times its ran before attaching the predicted value to the player. The batch size takes 30 random players from the training data and runs the model 200 times to predict a value.

```
7  const getModel = (type) => {
8    if (type === 'Attacker') {
9      return attackingModel;
10   } else if (type === 'Midfielder') {
11     return midfieldModel;
12   } else if (type === 'Defender') {
13     return defendingModel;
14   } else if (type === 'Goalkeeper') {
15     return goalKeepingModel;
16   }
17 };
18
19 const machineLearning = async () => {
20   console.log('Machine Learning Started');
21   let players = await Player.find({
22     transfer: {
23       $exists: true
24     },
25     statistics: {
26       $exists: true
27     }
28   });
29
30   // training data
31   const [attackingPlayersArray, attackingPlayers] = await trainAttackers(players);
32   const [defendingPlayersArray, defendingPlayers] = await trainDefenders(players);
33   const [midfieldPlayersArray, midfieldPlayers] = await trainMidfielders(players);
34   const [goalkeeperPlayersArray, goalkeeperPlayers] = await trainGoalkeepers(players);
35 }
```

*Creating model objects, starting machine learning, and creating arrays*

```

91  const attachTransferValue = async (players) => {
92    players = await players.map(async (currPlayer) => {
93      if (currPlayer.statistics) {
94        const model = await getModel(currPlayer.statistics[0].games.position);
95        currPlayer = new Player({
96          p_id: currPlayer.player.id,
97          name: currPlayer.player.name,
98          number: null,
99          photo: currPlayer.player.photo,
100         position: currPlayer.statistics[0].games.position,
101         teamName: currPlayer.statistics[0].team.name,
102         teamId: currPlayer.statistics[0].team.id,
103         transfer: null,
104         statistics: currPlayer.statistics[0],
105         extraData: {
106           age: currPlayer.player.age
107         }
108       });
109     });
110     if (currPlayer.position === 'Attacker') {
111       const tensor = await trainAttackers([currPlayer]);
112       if (tensor[0].length === 0) return currPlayer;
113       const result = model.predict(tf.tensor2d(tensor[0]));
114       currPlayer.transfer = {
115         feeValue: result.dataSync()[0],
116         feeString: '£' + (result.dataSync()[0] / 1000000).toFixed(2) + ' Million'
117       };
118       return currPlayer;
119     } else if (currPlayer.position === 'Midfielder') {

```

*Attaching predicting value to the player and if statements done for each position*

The train.js defines the top five leagues at the top of the file, being the English Premier league, German Bundesliga, French Ligue 1, Portuguese Primeira and Italian Serie A. After the leagues are defined the training functions are created. This will only take players that have statistics and predicts on the stats outlined below. Each model has specific statistics as Goalkeepers could not be predicted on scoring goals etc.

```

1  position = ['Attacker', 'Midfielder', 'Defender', 'Goalkeeper'];
2  league = [
3    'Premier League',
4    'Bundesliga 1',
5    'Ligue 1',
6    'Primeira Liga',
7    'Serie A',
8  ];
9
10 const trainAttackers = async (players) => {
11   players = players.filter(player => player.statistics.games.rating !== null && player.statistics.
12   let playersArray = players.map(player => {
13     const top5League = league.includes(player.statistics.league.name) ? 1 : 0;
14     const rating = parseFloat(parseFloat(player.statistics.games.rating).toFixed(2));
15     const playerPosition = position.indexOf(player.statistics.games.position) + 1;
16     const under25 = parseInt(player.extraData.age) < 25 ? 1 : 0;
17     const hoursPlayed = parseInt((player.statistics.games.minutes / 60).toFixed());
18     const goals = player.statistics.goals.total;
19     const assists = player.statistics.goals.assists;
20
21     return [rating, playerPosition, top5League, under25, hoursPlayed, goals, assists];
22   });
23   // Remove all null values
24   playersArray = playersArray.map((player) => {
25     return player.map(index => index === null ? 0 : index);
26   });
27   return [playersArray, players];
28 }

```

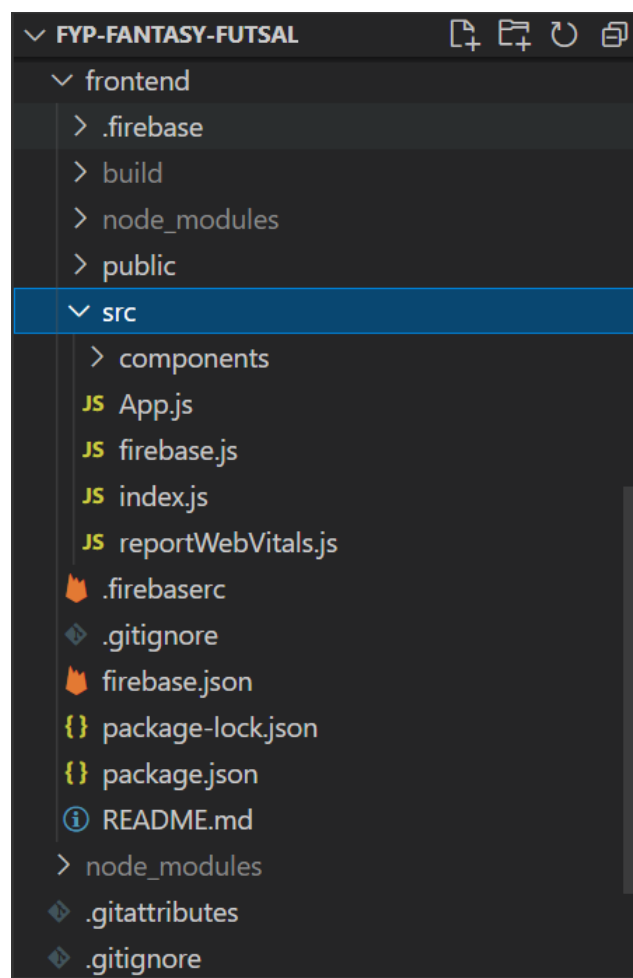
*Train.js attacking model*

### 2.3.3 Frontend

The client-side of the project is completed using the REACT JavaScript library. REACT was a library I was unfamiliar with, but with JavaScript being a preferred language of mine I decided to take up learning the popular growing library. REACT has many advantages when building websites, one that stuck out for me upon becoming comfortable with the library was the use of REACT components. Using REACT meant that the website will be a single page application with components making up the views by hiding and showing each component depending on what the user is looking for. (Creation, 2022) Not wanting to develop lots of HTML pages and style them all, REACT was a breath of fresh air to learn and implement the frontend of the website. REACT allows you to create default components and nested components. Default components are consistent across every page/component of the website, while nested are pulled in when the user clicks on the link the intend to open.

The client-side implementation will talk about connecting to the Firebase database for registering and logging in to the website and the components that make up the UI.

The first step was to create a new REACT app by changing directory to inside the frontend folder. From here, the “npx create-react-app Fantasy-Futsal-App”. This REACT command generates a structured folder with files inside so you can begin coding straight away. The figure below shows the folders and files of the frontend. Added component folder contains all the nested component files.



*REACT generated folder structure*



### 2.3.3.1 Firebase and authorisation

With the database being set up on firebase already, the next step was to connect the project to firebase. The first action taken was creating the Firebase.js file to make it easier to locate the database auth. Firebase make it extremely quick easy to scale up database's so the connection was quite simple. A couple of commands ran in the projects directory, that are available from the Firebase website, and small piece of configuration in the Firebase.js file and the database was connected. (Add Firebase to your JavaScript project | Firebase Documentation, 2022) Firebase authentication and tables will be discussed under the components section.

```
frontend > src > JS firebase.js > ...
1  import firebase from "firebase";
2  const firebaseConfig = {
3    apiKey: "AIzaSyB...",
4    authDomain: "...",
5    projectId: "f...",
6    storageBucket: "fir...",
7    messagingSenderId: "...",
8    appId: "1:90476...",
9    measurementId: "G-...",
10 };
11
12 const app = firebase.initializeApp(firebaseConfig);
```

*Database config for connection, Firebase.js file*

### 2.3.3.2 REACT components

The components implementation makes up the user interface of the website. The app.js file that was created in the generated files when the REACT app was created will be the default component. This is the main file that all other components get imported to, to be used on the website. When the user clicks on a component link, REACT quickly changes the view to the chosen components. The figure below shows the layout of the main component.

```

JS App.js M X
frontend > src > JS App.js > ...
1  import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
2  import Home from './components/Home';
3  import Pickteam from './components/Pickteam';
4  import ViewLeague from './components/ViewLeague';
5  import Navbar from './components/Navbar';
6  import CreateLeague from './components/CreateLeague';
7  import About from './components/About';
8
9  function App() {
10   return (
11     <div className="App">
12       { /* Navbar */ }
13       <Navbar />
14       { /* Router */ }
15       <Router>
16         <Routes>
17           <Route path="/" caseSensitive={false} element={<Home />} />
18           <Route path="/pickteam" caseSensitive={false} element={<Pickteam />} />
19           <Route path="/viewleague" caseSensitive={false} element={<ViewLeague />} />
20           <Route path="/createleague" caseSensitive={false} element={<CreateLeague />} />
21           <Route path="/about" caseSensitive={false} element={<About />} />
22         </Routes>
23       </Router>
24     </div>

```

*App.js, this is the single page of the application with nested components imported*

As seen above, there are six nested components that make up the navbar and pages of the website. The nested components are navbar, home, pick team, view league, create league and about. To show the nested components, they must be imported into the main component. This can be seen at the top of the figure. The six components implementation will be discussed in detail below, with the aid of code snippets for reference.

**Navbar Component** - The Navbar component is the only nested component that is consistent across all pages. This was the first component developed which gave me a structure to follow when moving on to the rest of the components. Inside this component contains a layout of a navbar designed using bootstrap. Each component contains a link that the user can click to show the desired page.

```

JS Navbar.js X
fyp-fantasy-futsal > frontend > src > components > JS Navbar.js > [0] Navbar
22   aria-expanded= false
23   aria-label="Toggle navigation"
24   >
25     <span className="navbar-toggler-icon"></span>
26   </button>
27   <div className="collapse navbar-collapse" id="navbarSupportedContent">
28     <ul className="navbar-nav me-auto mb-2 mb-lg-0">
29       <li className="nav-item">
30         <a className="nav-link active" aria-current="page" href="/">Home</a>
31       </li>
32       <li className="nav-item">
33         <a className="nav-link" aria-current="page" href="/pickteam">Pick Team</a>
34       </li>
35       <li className="nav-item">
36         <a className="nav-link" aria-current="page" href="/viewleague">View Leagues</a>
37       </li>
38       <li className="nav-item">
39         <a className="nav-link" aria-current="page" href="/createleague">Create League</a>
40       </li>
41       <li className="nav-item">
42         <a className="nav-link" aria-current="page" href="/about">About</a>
43       </li>
44     </ul>
45     {user ?
46       <ul className="navbar-nav me-auto mb-2 mb-lg-0">
47         <li className="nav-item" style={{marginTop: '0.5rem'}}>

```

*Bootstrapped navbar*

Another feature added to the Navbar is the logout button. This appears when a user is logged in successfully. The logout feature is implemented in the Firebase.js file before being imported to the Navbar component.

```
2 import { logout } from "../../firebase";
```

*Logout import from Firebase to Navbar.js*

The code shown below is the function that is exported from the Firebase.js file. Then, using an onclick, this function kills the user's session when they presses the logout button.

```
54 const logout = () => {
55   auth.signOut();
56   localStorage.removeItem("user");
57   window.location.href = window.location.href;
58 };
```

*Logout function, firebase.js*

Once the Navbar was complete, it was time to create the landing component of the site.

**Home Component** - The home component is the first page seen when a user opens the website. The authentication of the project is also handled here. With the database connected, all that was left to do was take users input to populate the database. The user must register first before they can access the website. After the user has registered, they can proceed to login and explore the app. The home page changes depending on whether a user is logged in or not. When the user is not logged in they will see fields available to login/register, whereas when they are logged it will show the users team (If they have one picked and the logout button.

The below snapshot shows the home components layout.

This piece of code determines whether a user is logged in and changes the view to the user's team if they are logged in successfully.

```
return (
  <div>
    <div className="container">
      <div className="row mt-5">
        <div className="col-12 col-lg-8">
          {user !== null && team !== null ?
            <>
              <h3 style={{textTransform: 'uppercase'}}>{user?.displayName} Team <span className="badge bg-
              <Team usersTeam={team}></Team>
            </>
            :
            <>
              </>
            </>
          }
        </div>
      </div>
    </div>
  )
```

*Home component logged in*

If the user is not logged in the following piece of code will run. This will show fields for the user to register/login.

```

<div className="col-12 col-lg-4">
  {user !== null ?
  <div>
    <h3 style={{textTransform: 'uppercase'}}>{user?.displayName} Team</h3> <br/>
    <button className="btn btn-danger" type="button" onClick={logout}>Logout</button>
  </div>

  : <form style={{margin: '10px'}}>
    <div className="form-group" style={{padding: '10px'}}>
      <label htmlFor="exampleInputEmail1">Email address</label>
      <input type="email" className="form-control" aria-describedby="emailHelp" placeholder="Enter email" value="" />
    </div>
    <div className="form-group" style={{padding: '10px'}}>
      <label htmlFor="exampleInputPassword1">Password</label>
      <input type="password" className="form-control" id="exampleInputPassword1" placeholder="Password" value="" />
    </div>
    <div className="d-grid gap-2" style={{padding: '10px'}}>
      <button className="btn btn-success" type="button" onClick={() => registerWithEmailAndPassword(email, password)}>Register</button>
      <button className="btn btn-primary" type="button" onClick={() => signInWithEmailAndPassword(email, password)}>Login</button>
      <button className="btn btn-danger" type="button" onClick={signInWithGoogle}>Login/Register with Google</button>
    </div>
  </form>
}
</div>
</div>

```

*Home component before logging in*

**Pick team Component** – The pick team component allows a user to pick five players and submit their team. As the user is choosing the players, the players selected will move to the football pitch on the left side of the UI to give a better visual of who they picked. If the user changes or deletes their team, an alert can be seen to give confirmation of their update.

```

<div className="container" style={{marginTop: '2em'}}>
  {success || deleted ?
  <div className="row">
    <div className="alert alert-success" role="alert">
      <strong>Success!</strong> Your team has been updated
    </div>
  </div>
  :
  <></>
}

```

*Team updated alert*

The user is giving a £50 million budget which they must keep under when selecting their team. If the user goes over their budget they will be alerted and the save button will be disabled.

```

{teamValue > 50000000 ?
<div className="row">
  <div className="alert alert-warning" role="alert">
    <strong>Warning!</strong> You are over budget reduce your budget by £{(Math.abs(budget) / 1000000).toFixed(2)} million
  </div>
</div>
:
<></>
}

```

*Over budget warning*

If the team chosen accumulates to under £50 million the user will have an option to save the team.  
The delete option is always available to the user.

```
170 <div>
171   { teamValue < 50000000 ?
172     <button className="btn btn-success" type="button" onClick={() => saveTeam()}>Save <i className="fa fa-save" aria-
173       :
174     <button type="button" className="btn btn-danger disabled">Over Budget</button>
175   }
176   <button className="btn btn-danger" type="button" onClick={() => deleteTeam()}>Delete Team <i className="fa fa-ti
177 </div>
178 </div>
```

*Save allowed if budget and delete team*

When the user is selecting their team, they have two options on searching for players. The first option is search by name and second is a dropdown of all the teams in the Premier league, where the user can pick a team and browse through that team's squad.

```
<div className="col-12 col-lg-6">
  <h3>SEARCH</h3>
  <div className="input-group">
    <input type="text" className="form-control" placeholder="Search Player" aria-label="" value={search} onChange={searchPlayers}>
    <div className="input-group-append">
      <button className="btn btn-success" type="button" onClick={() => searchPlayers()}>Search <i className="fa fa-search"></i></button>
    </div>
  </div>
  <h3 class="mt-3">SEARCH BY TEAM</h3>
  <select class="form-select mt-3 mb-3" aria-label="" onChange={(e) => setCurrTeam(e.target.value)}>
    {leagueTeams.map((curr, idx) => {
      return (<option value={curr.team.id} key={idx}>{curr.team.name}</option>)
    })}
  </select>
```

*Searching functionality*

When a player is selected the set selected player function will run moving the players photo onto the user football pitch for confirmation of the selection.

```
<button type="button" className="btn btn-outline-primary" onClick={() => selectPlayer(player)}>Select Player</button>
<img src={player.statistics.team.logo} style={{height: '30px', paddingLeft: '.5em', paddingRight: '.5em'}}/>
<button type="button" class="btn btn-secondary" data-bs-toggle="modal" data-bs-target="#pickTeamModal" onClick={() => setSelectedPlayer(player)}>
  <i className="fa fa-info" aria-hidden="true"></i>
</button>
.v>
```

*Select player functionality*

**View league Component** – The view leagues component is split into two sections. One side displays all the leagues available, with a button to join and the other side shows all the leagues created by the user, the owner of the leagues.

The all-leagues side will show the number of members in the league. If one of the members include the user ID, the user will see “You joined already” on the button. If the user is not already a part of the league and the league is public, they can join the league by pressing the join league button. If the league is private and the user attempts to join, they will be prompted with a pop up box to enter a unique code to join the league.

```

<div className="col-12 col-lg-6">
  <h3>All Leagues</h3>
  <ul class="list-group">
    {leagues.map((league, idx) => {
      return (
        <li class="list-group-item" style={{display: "flex", flexFlow: 'wrap', justifyContent: "space-between", alignItems: "center"}}>
          <div>
            <p style={{margin: '0'}}>
              {league.league_name} <span className="badge bg-info">Members: {league.number_of_members}</span> <span>
            </p>
          </div>
          <div>
            {league.members.includes(user.uid) || league.owner_id === user.uid ?
              <a href={"/league/" + league._id} class="btn btn-outline-secondary">You already Joined!</a>
              :
              <button type="button" class="btn btn-outline-success" onClick={() => joinLeague(league)}>Join League</button>
            }
          </div>
        </li>
      )
    })}
    {enterCode !== league._id ?
      <div className="form-group mt-2" style={{flexGrow: '1', width: '100%'}}>
        <label for="formGroupExampleInput">League Password</label>
        <input type="text" class="form-control" id="formGroupExampleInput" placeholder="Enter your password">
        <button type="button" class="btn btn-outline-success" onClick={() => joinLeaguePrivate(league)}>Join League</button>
      </div>
      :
      <></>
    }
  </ul>
</div>

```

### *All leagues side of view leagues*

The opposite side to all leagues will show all the leagues that the user has created as well as the number of members in the league.

```

<div className="col-12 col-lg-6">
  <h3>Your Leagues</h3>
  <ul class="list-group">
    {myLeagues.map((league, idx) => {
      return (
        <li class="list-group-item" style={{display: "flex", justifyContent: "space-between", alignItems: "center"}}>
          <div>
            <p style={{margin: '0'}}>
              {league.league_name} <span className="badge bg-info">Members: {league.number_of_members}</span> <span>
            </p>
          </div>
          <div>
            <a href={"/league/" + league._id} class="btn btn-outline-secondary">View League</a>
          </div>
        </li>
      )
    })}
  </ul>
</div>
</div>

```

### *Your leagues div*

**Create league Component** – The create league component starts with an alert if a user creates a league successfully.

```

<div className="container mt-5">
  {success ?
    <div className="row">
      <div className="alert alert-success" role="alert">
        <strong>Success!</strong> Your league has been created
      </div>
    </div>
    :
    <></>
  }

```

### *League created successfully alert*

The your leagues side is a list of the leagues created by the user. The user can click on the view league button to be brought to the leagues personal page where they can see the leagues standings.

*Your leagues div*

46

```

<div className="container mt-5">
  <div className="row">
    <div className="col-12" style={{textAlign: 'center'}}>
      <h3>POINT SYSTEM</h3>
      <table class="table table-striped mt-2">
        <thead>
          <tr>
            <th>Yellow card</th>
            <th>Red card</th>
            <th>Penalty save</th>
            <th>Goal scored by a goalkeeper or defender</th>
            <th>Goal scored by a midfielder</th>
            <th>Goal scored by a forward</th>
            <th>Assist</th>
            <th>For playing up to 60 minutes</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>-1</td>
            <td>-2</td>
            <td>3</td>
            <td>5</td>
            <td>4</td>
            <td>3</td>
            <td>2</td>
            <td>1</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>

```

*Point system table*

Under the point system, I thought it would be a good idea to show all the transferred players that were used for the machine learning model.

```

<div className="row" style={{textAlign: 'center', marginTop: '3em'}}>
  <h3 style={{textTransform: 'uppercase'}}>Players Used for our machine learning model</h3>
  {trainingData.map((player, idx) => (
    <li key={{idx}} lass="list-group-item" style={{display: 'flex', justifyContent: 'space-between', align-items: 'center'}}>
      <img src={player.photo} style={{height: '50px', borderRadius: '50%', border: '3px solid blue'}} alt="Player photo" data-bbox="240 610 320 630" />
      <p>
        {player.name}<br/>
        {player.transfer.feeString}
        { /* <button type="button" className="btn btn-primary">Pick me</button> */ }
      </p>
      <div>
        <img src={player.statistics.team.logo} style={{height: '30px', paddingLeft: '.5em', padding: '2px'}} alt="Team logo" data-bbox="330 680 380 700" />
        <button type="button" class="btn btn-secondary" data-bs-toggle="modal" data-bs-target="#aboutTeamModal">
          <i className="fa fa-info" aria-hidden="true"></i>
        </button>
      </div>
    </li>
  ))}
</div>
<PlayerInfo player={selectedPlayer} id="aboutTeamModal"></PlayerInfo>
</div>

```

*Pulling in training data*

**League component** – The league component allows a user to click the view league button and shows the standings of the selected league. It starts off with a ternary operator which checks if the league is public or private. If the league is public, it will display the leagues access code at the top of the page.



```

{ league !== null ?
  <div className='row'>
    { league.public ?
      <></>
    :
      <div class="alert alert-info" role="alert">
        <strong>Heads up!</strong> This League is private the password is {league.code} send to your friends for them to join.
      </div>
    }
  }
}

```

### *Displaying league code*

Next the league name and status are displayed. Underneath the league name is a list of all the users of league showing the users name and the amount of points their team accumulated. A ternary operator determines if the current ID is the users and will show the “Your team” button if the IDs match. The “View team” button will be seen for all other users that don’t match the current logged in ID.

```

65 <div className='col-12 col-lg-6'>
66 <h3>{league.league_name} <span className="badge bg-info">{league.public ? 'PUBLIC' : 'PRIVATE'}</span></h3>
67 <ul class="list-group">
68   {league.users.map((curr, idx) => {
69     return(
70       <li style={{display: 'flex', justifyContent: 'space-between', alignItems: 'center', textTransform: 'capitalize'}} key={idx} classNa
71       <p style={{margin: '0'}}>
72         {curr_id === user.uid ? 'Your Account:' : ''} {curr.name !== '' ? curr.name : curr.email} Points: {fetchPoints(curr_id)}
73       </p>
74       {curr_id === user.uid ? <button type="button" class="btn btn-dark" onClick={() => updateTeam(curr)}>Your Team</button> : <button
75       </li>
76     )
77   })}
78 </ul>
79 </div>

```

### *Your team/view team button*

The next div will display the team of the selected user. This will display once the user has selected and saved a team and the team is not equal to null.

```

    <div className='col-12 col-lg-6'>
      <h3 style={{textTransform: 'capitalize'}}>{teamName !== '' ? teamName : 'Your Team'}</h3>
      {team !== null ?
        <Team usersTeam={team}></Team>
      :
        <></>
      }
    </div>
  </div>
  :
  <div class="spinner-border" role="status">
    <span class="sr-only">Loading...</span>
  </div>
}
</div>

```

### *Displaying users team*


**Player info component** – The player info is a cool component that allows users to click the user info button on a player on the about page. This will trigger a pop-up page with all the players statistics and information.

```




###### Predicted Transfer Value: {player.transfer.feeString} Avg Rating: {player.statistics.avgRating}



{player.teamName}



- Cards {player.statistics.cards.yellow}
- Dribbles Attempts {player.statistics.dribbles}
- Duels Total {player.statistics.duels}
- Fouls Committed {player.statistics.fouls}
- Games Appearances {player.statistics.games}
- Goalkeeper? Goals {player.statistics.goals}
- Passes Key {player.statistics.passes}
- Penos {player.position !== "Goalkeeper" ? Key {player.statistics.passes} : Shots Total {player.statistics.shots}
- Tackles Total {player.statistics.tackles}


```

*Data shown for player selected*

**Team component** – The team component gives the structure of the team for when the user selects their players. The FootballBackground.png is used to style the background. Four rows are created to split up the team into positions. This component is then called into the pick team and home component to be displayed. Developing this in its own component instead of separately in the pick team and home component is a good example of how REACT components are re-usable and easy to pull into different component when needed.

```

typ-fantasy-futsal > frontend > src > components > JS Team.js > ...
4  const Team = (props) => {
5    console.log(props.usersTeam);
6    return (
7      <div className="container" style={{backgroundImage: "url(/FootballBackground.png)", backgroundSize: 'cover', height: '100vh'}}>
8        <div className="row">
9          <h3>Attack</h3>
10         <div className="col" style={{display: 'flex', justifyContent: 'center', textAlign: 'center'}}>
11           <TeamPlayer player={props.usersTeam.playerFive}></TeamPlayer>
12         </div>
13       </div>
14       <div className="row">
15         <h3>Midfield</h3>
16         <div className="col" style={{display: 'flex', justifyContent: 'center', textAlign: 'center'}}>
17           <TeamPlayer player={props.usersTeam.playerFour}></TeamPlayer>
18         </div>
19       </div>
20       <div className="col" style={{display: 'flex', justifyContent: 'center', textAlign: 'center'}}>
21         <TeamPlayer player={props.usersTeam.playerThree}></TeamPlayer>
22       </div>
23       <div className="row">
24         <h3>Defence</h3>
25         <div className="col" style={{display: 'flex', justifyContent: 'center', textAlign: 'center'}}>
26           <TeamPlayer player={props.usersTeam.playerTwo}></TeamPlayer>
27         </div>
28       </div>
29       <div className="row">
30         <h3>Keeper</h3>
31         <div className="col" style={{display: 'flex', justifyContent: 'center', textAlign: 'center'}}>
32           <TeamPlayer player={props.usersTeam.playerOne}></TeamPlayer>
33         </div>
34       </div>
35     </div>
36   );
37 }

```

*Positional structure*

**Team player component** – The team player shows on the background of a pitch if a player is selected in each position or if it has no selection. If the value is null, the component will show “No selection”. If a player is selected the component will switch to display the players photo, name and predicted transfer value.

```
JS TeamPlayer.js X
fyp-fantasy-futsal > frontend > src > components > JS TeamPlayer.js > [e] TeamPlayer
1  import React, {useState} from 'react';
2  import PlayerInfo from './PlayerInfo';
3
4  const TeamPlayer = (props) => {
5    const value = props.player.value;
6
7    return (
8      <div>
9        {
10          value === null?
11            <div style={{ background: 'rgba(0, 0, 0, 0.5)', padding: '0.5em', borderRadius: '20px', color: 'white' }}>
12              <small>No Selection</small>
13            </div>
14          :
15            <div style={{ background: 'rgba(0, 0, 0, 0.5)', padding: '0.5em', borderRadius: '20px', color: 'white' }}>
16              <img className="team--player--img" src={props.player.value.photo} style={{height: '5em', borderRa
17              <p>{props.player.value.name} - {props.player.value.transfer.feeString}</p>
18            </div>
19          }
20        </div>
21      )
22    }
23
24  export default TeamPlayer
```

*No selection/selected player component*

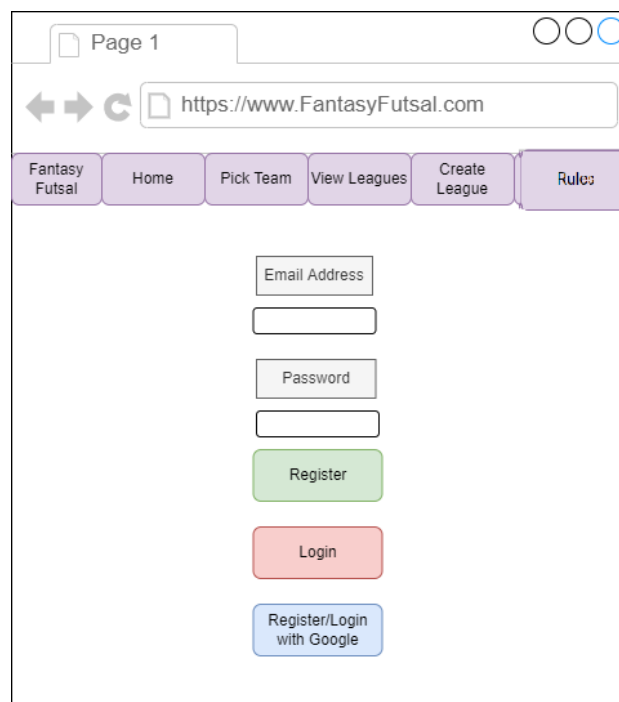
## 2.4. Graphical User Interface (GUI)

### 2.4.1 Component wireframes

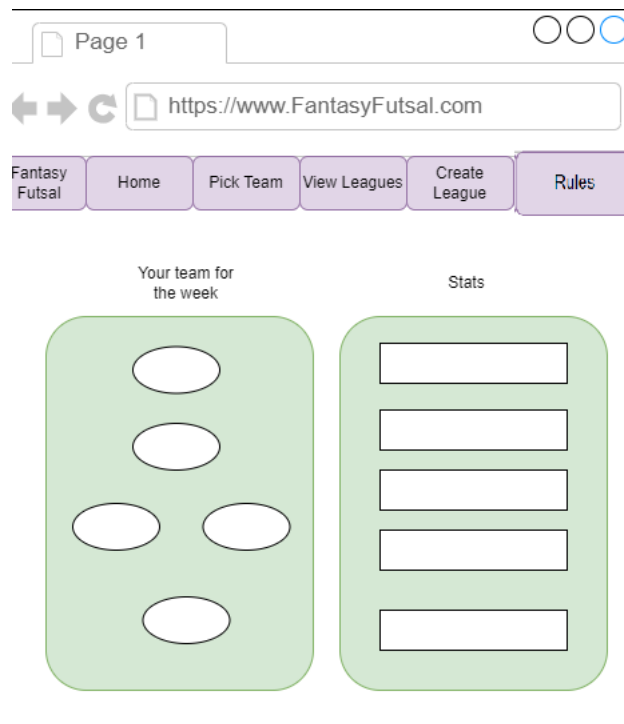
The frontend design of the website started with mock up wireframes. This gave a good indication of what I wanted the REACT components to look like. A wireframe was designed for every page so I could visualise how to go about developing the components best.

The navbar design feature was key for ease of use in the website, this is the only feature consistent over all pages.

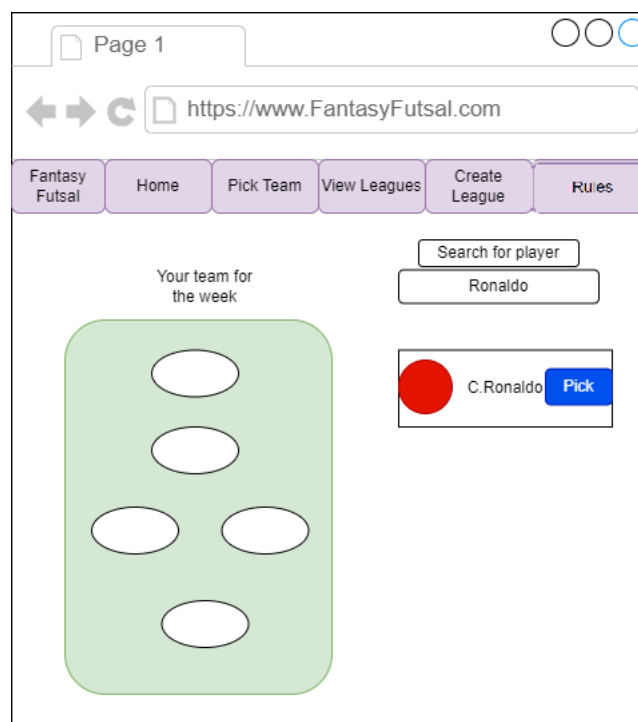
The figures shown below gave a clear structure of how the pages should look come the end of the project.



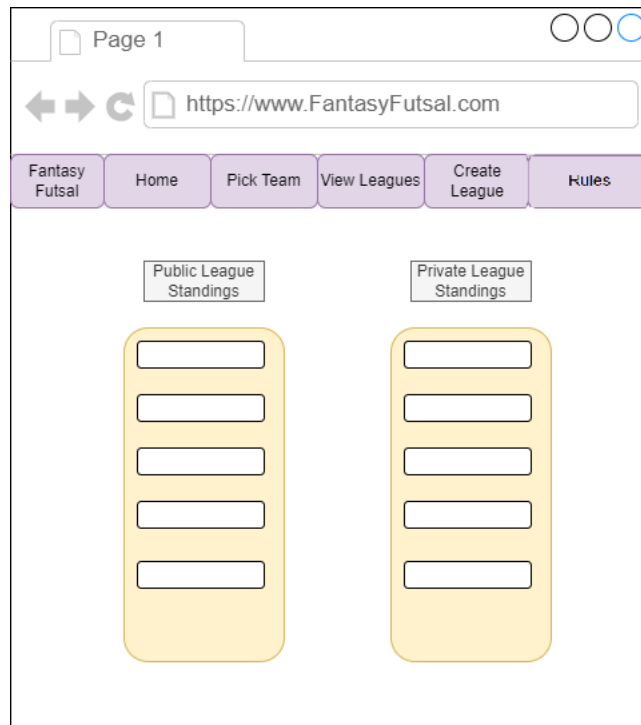
*Register/login wireframe*



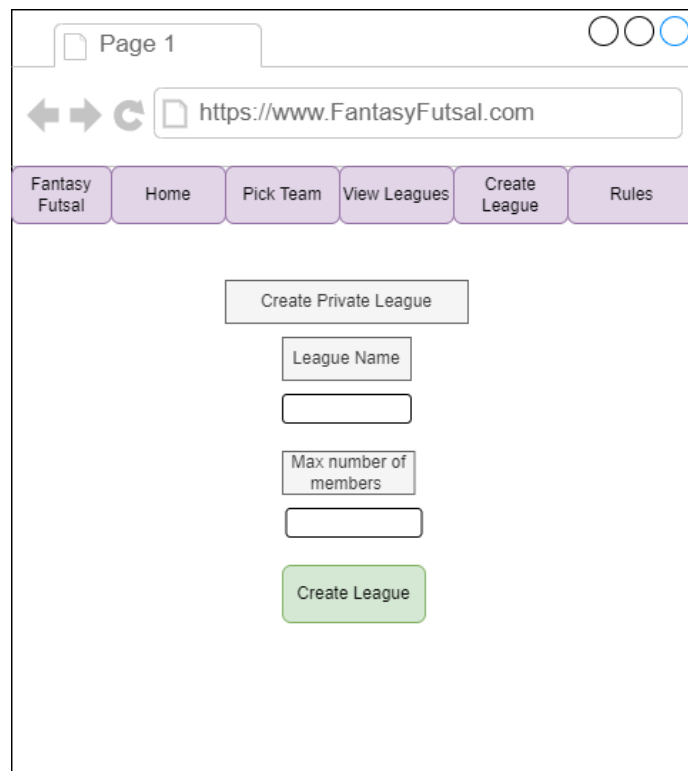
Home page wireframe



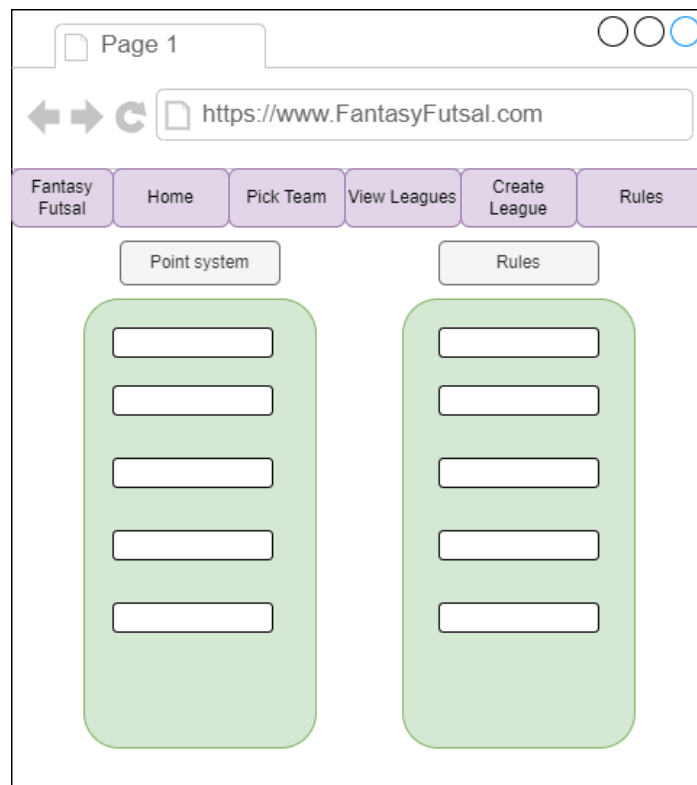
Pick team wireframe



*View league wireframe*



*Create league wireframe*



*Rules wireframe*

The wireframe designs seen above were constantly referred to when developing the pages. Each page is a REACT component which makes them re-usable if needed, for example the rules component could be easily slotted into the home page for more users to view, if users were finding it hard to understand the rules and score system of the site.

#### 2.4.2 GUI pages

All the user interface that will be available to see in Fantasy Futsal include:

##### Home component before logging in

Fantasy Futsal
Home
About

Email address

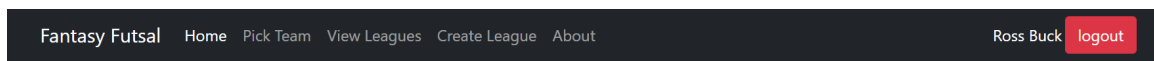
Password

Register

Login

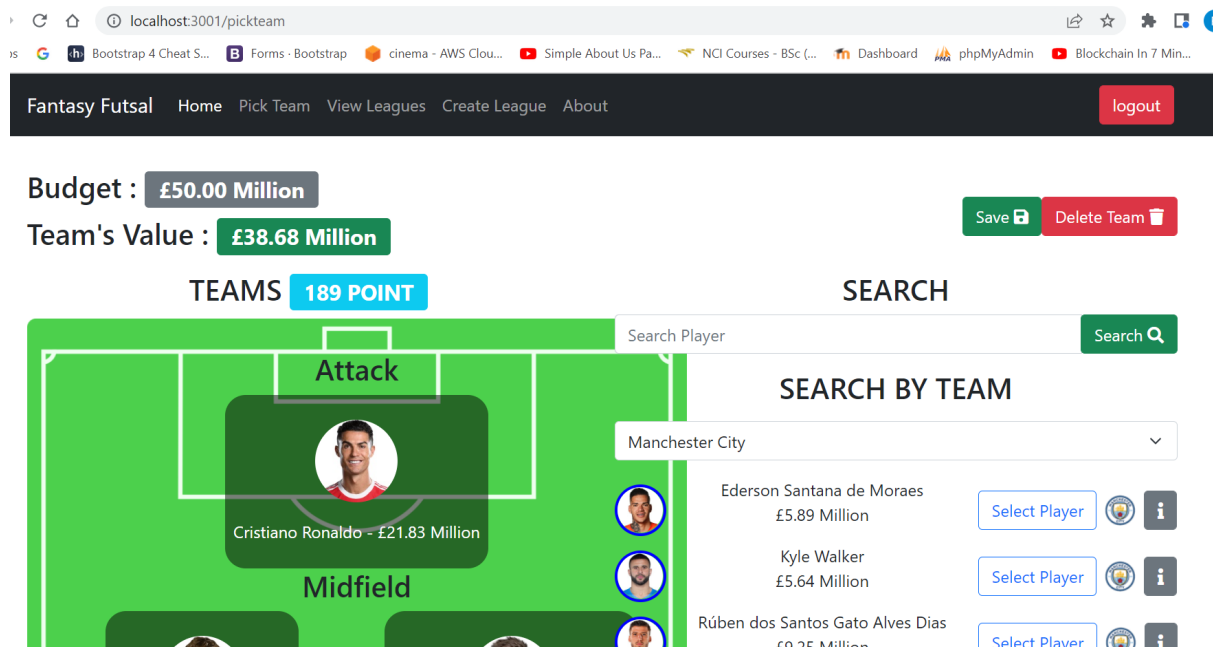
Login/Register with Google

## Home screen



Home component when logged in

## Pick your team screen



Pick team component



## Create a league screen

localhost:3001/createleague

Fantasy Futsal Home Pick Team View Leagues Create League About [logout](#)

### Your Leagues

Stittsers league **Members: 2** **Type: Private**

Password: 38540611-679c-411d-abe9-9b70d49878a3

[View League](#)

### Create Leagues

League name

Add Users

Max members  
0

☐ Private League

[Submit](#)

Create league component

## View league screen

localhost:3001/viewleague

Fantasy Futsal Home Pick Team View Leagues Create League About [logout](#)

### All Leagues

This is a test league <b>Members: 3</b> <b>Type: Public</b>	<a href="#">Join League</a>
Testing league <b>Members: 1</b> <b>Type: Public</b>	<a href="#">Join League</a>
This is a private league <b>Members: 1</b> <b>Type: Private</b>	<a href="#">Join League</a>
RossB League <b>Members: 2</b> <b>Type: Private</b>	<a href="#">Join League</a>
Big league <b>Members: 1</b> <b>Type: Public</b>	<a href="#">Join League</a>
Small league <b>Members: 1</b> <b>Type: Private</b>	<a href="#">Join League</a>
Testing league 13th Friday <b>Members: 2</b> <b>Type: Private</b>	<a href="#">Join League</a>

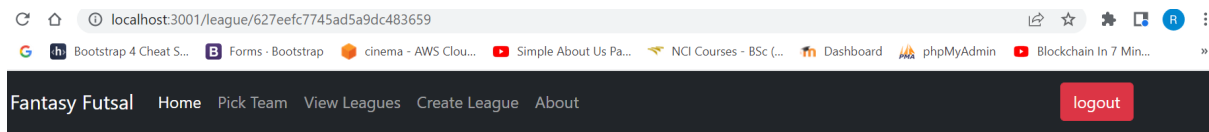
### Your Leagues

Stittsers league **Members: 2** **Type: Private**

[View League](#)

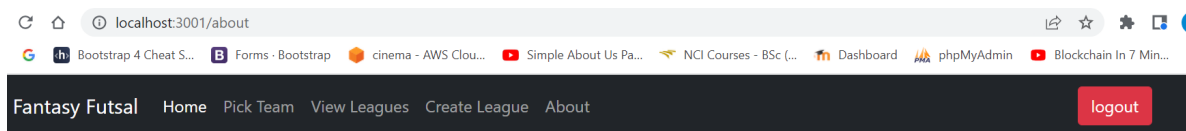
View league component

## Leagues screen



League component

## About Page



### POINT SYSTEM

Yellow card	Red card	Penalty save	Goal scored by a goalkeeper or defender	Goal scored by a midfielder	Goal scored by a forward	Assist	For playing up to 60 minutes
-1	-2	3	5	4	3	2	1

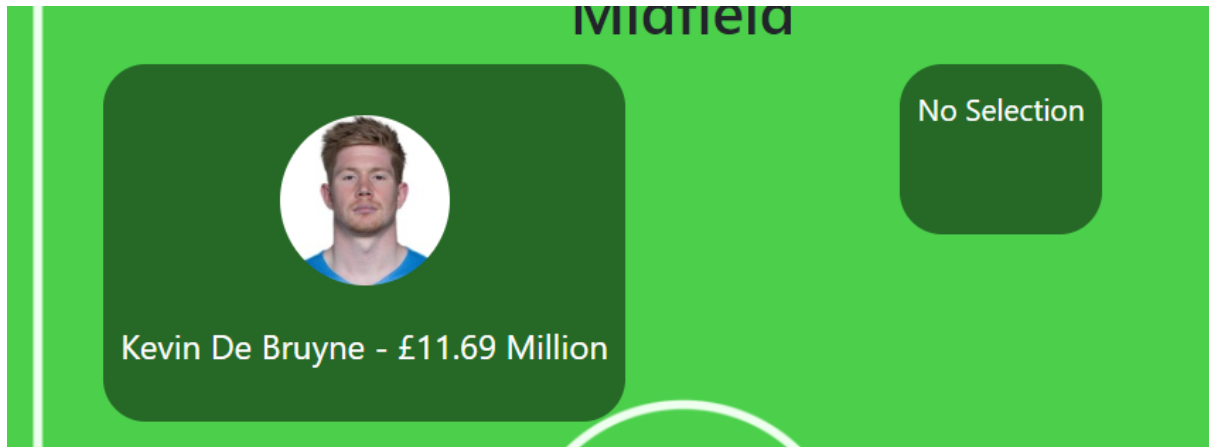
### PLAYERS USED FOR OUR MACHINE LEARNING MODEL

	João Pedro £3.60m	 
	E. Mwepu £11.70m	 
	T. Minamino £7.65m	 

About component, score system and all players used for machine learning model

### Team player component

The two visuals of the team player component, when a player is selected and when there is no selection.



*Team player component with and without selection*

### Team component

The team component that re-used on the home and pick team components.




*Team component*

## Player info screen

João Pedro


Attacker

✕



Predicted Transfer Value: £3.60m

Avg Rating 6.566666%

  
Watford

Cards	0	0	
Dribbles	Attempts 4	Past 0	Success 3
Duels	Total 12	Won 5	
Fouls	Committed 1	Drawn 0	
Games	Appearances 3	Lineups 0	

## 2.5. Testing

Due to lack of time and the deadline approaching, I fell short in the testing section of the project. There were plans in place for a lot more testing to be sure the website was stable and secure. Having taken a testing role in my internship I felt this would be a section I could strive in.

Test cases will be implemented to be sure each function is working as it should and that backend is storing correctly in the right tables etc.

## 2.6. Evaluation

Overall, I'm happy with how the project turned out. All the aims of the project were complete so I would class the project as a success. Although I had planned for the project to be finished a lot earlier and tried my best to allow for issues in the plan, some issues are just inevitable and need more time to conquer. There was a lot of new technologies that I've never used before implemented in the project, which I enjoyed and would like to use again in future projects. If I was to restart my project today with all new technologies, I would allocate more time to research and documentation. I would spend more time creating a project plan as I have more knowledge of sub-tasks and where issues can occur.

I feel the issues I ran into helped me broaden my knowledge on how to break down issues and find a solution. The new technologies used that were not in the plan from the start, gave me a different perspective on how to approach issues and learning new technologies that will stand to me for the future of my career.

Using Firebase for authentication saved a serious amount of time. Firebase make it so easy to set up the database and login system but fell short on storage. MongoDB turned out to be suitable back up database for storage and like Firebase is very easy to get started even with no previous experience using the tools.

I really enjoyed my first experience using REACT. Being able to develop re-usable components on the frontend was cool and stops any repetitive code being written.

Node.js was something I experienced using before and always works well in projects. I think if I was to create another app, I would go for REACT for frontend and Node for backend again. Using JavaScript for the entire project was another advantage as I didn't have to keep changing back to different syntax which can be confusing when developing a project.

TensorFlow turned out to be a hassle installing onto my device, but once we used Docker to get around this it turned out to be a nice technology to use.

Every language, framework or platform used all integrated well together and there was no serious issues that couldn't be worked out with some rethinking.

Having the frontend and backend in two separate branches meant I needed to host them both separately when the project was complete. The frontend is hosted using Firebase and the backend is hosted with Heroku.

### 3.0 Conclusions

In the end, I believe Fantasy Futsal is a very good application with serious potential. All objectives and aims were completed and more. I was very surprised at how good the app turned out and am extremely happy with the result. With extra time and resources Fantasy Futsal could be developed to a high scale application with some more added quality features.

Better time management would have made the project a bit easier, but I got it completed in the end. With more experience now I believe I could develop a more realistic project plan and stick to it. I've gained knowledge on how to go about solving problems and using new technologies that will stand to me forever.

A huge success of the project was the machine learning models. They predicted values the right way that I planned. A lot of the values were in and around the same price which is correct as all of these players play in the same league and are at a similar standard so it should be close. The values predicted are a massive drop to what is being spent in today's transfer market, but this is what I expected as I don't believe the players are worth what today's market would say. I believe my machine learning models to be more accurate than today's market values.

From start to finish the project was challenging and a success. I enjoyed the whole process and developed some good habits along with improving my skills. I feel it has got me ready to take the next step of my career.

### 4.0 Further Development or Research

With the right software team to add features and improve the website in minor ways, I could see a high-quality website going to market and becoming successful. There is only one real competitor to this idea on the market, and Fantasy Futsal capitalises on their weakness'. For this reason, I think Fantasy Futsal could take the entire market in a couple of years.

I believe the machine learning model of this idea will be used at a larger scale in the future and in real world sports. Statistics is already being used to negotiate players' salaries by players themselves its only matter of time before computers are used to predict them as they could predict very accurate figure if trained properly. There is a huge opportunity for someone or some place to create a universal salary and value predictor for all sports. This could evolve into a real data Fantasy football application that is connected to the Premier League which could predict what footballers are worth on a weekly basis after their performance using statistics. Real figures could be used in the Fantasy futsal application, and I would think freely of adding new quality features if I felt they were right.

Kevin De Bruyne turned out to be the first high profile Premier League footballer to drop his agent and use data analysis to predict his salary for the year with Manchester City. (Lee, 2022) I believe this will have a domino effect on players all over the world. I would recommend a system being developed to predict everyone's transfer and salary across all the professional leagues of the world. Salaries could differ from week to week depending on players stats which would only be fair in my mind, you get out the work you put in.

## 5.0 References

Code?, W., 2022. *What is Visual Studio Code?/ Features And Advantages / Scope & Career*. [online] EDUCBA. Available at: <<https://www.educba.com/what-is-visual-studio-code/>> [Accessed 7 May 2022].

Apiumhub. 2022. *Top 7 benefits you get by using Github / Apiumhub*. [online] Available at: <<https://apiumhub.com/tech-blog-barcelona/using-github/>> [Accessed 7 May 2022].

Linkedin.com. 2022. *Advantages and Disadvantages of Firebase*. [online] Available at: <<https://www.linkedin.com/pulse/advantages-disadvantages-firebase-nav-adalyn>> [Accessed 7 May 2022].

Lee, S., 2022. *De Bruyne's new contract agreed after data analysts convinced him Man City were able to match his ambitions - and vice versa*. [online] The Athletic. Available at: <<https://theathletic.com/2503139/2021/04/07/de-bruyne-new-contract-agreed-after-data-analysts-convinced-him-man-city-were-able-to-match-his-ambitions-and-vice-versa/>> [Accessed 7 May 2022].

MongoDB. 2022. *Advantages Of MongoDB* [online] Available at: <<https://www.mongodb.com/advantages-of-mongodb>> [Accessed 9 May 2022].

Creation, M., 2022. *React JS: Advantages and Disadvantages*. [online] Massive Pixel Creation. Available at: <<https://massivepixel.io/blog/react-advantages-disadvantages/>> [Accessed 9 May 2022].

Designveloper. 2022. *What is Node.js And How It Works*. [online] Available at: <<https://www.designveloper.com/blog/what-is-nodejs/>> [Accessed 9 May 2022].

Firebase. 2022. *Add Firebase to your JavaScript project / Firebase Documentation*. [online] Available at: <<https://firebase.google.com/docs/web/setup>> [Accessed 9 May 2022].

Amazon Web Services, Inc. 2022. *What is Docker?/ AWS*. [online] Available at: <<https://aws.amazon.com/docker/>> [Accessed 12 May 2022].

neptune.ai. 2022. *Top 10 Best Machine Learning Tools for Model Training - neptune.ai*. [online] Available at: <<https://neptune.ai/blog/top-10-best-machine-learning-tools-for-model-training>> [Accessed 12 May 2022].

## 6.0 Appendices



# National College of Ireland

## Project Proposal

### How statistics are going to change the world of professional football

03/11/2021

Salary predictor

Software Development

2020/2021

Ross Buckley

18501723

X18501723@student.ncirl.ie

#### Contents

1.0	Objectives.....	64
2.0	Background .....	65
3.0	State of the Art.....	65
4.0	Technical Approach.....	66
5.0	Technical Details .....	66



6.0	Special Resources Required .....	67
7.0	Project Plan .....	67
8.0	Testing .....	69

## 6.1 Objectives

The projects aim is to be able to successfully predict a weekly salary and transfer market value for a professional footballer. I will connect the application to an API which will pull numerous types of data about professional football players and their statistics for the season played. Then ill use footballers who have been transferred before as training data for footballers who haven't been transferred before. TensorFlow will be the machine learning tool I will use to try predicting an accurate transfer market price and wage for the footballers. Once I tune TensorFlow into an accurate prediction I can move onto the frontend. The frontend will be a fantasy futsal league. The user will be giving a transfer budget and salary budget to buy the players they want. The transfer and salary predictions from the machine learning model will be the figures used for the players in the fantasy league. The user cannot go over either of the transfer or salary budgets allocated.

For the frontend, the aim is to create a fun fantasy futsal application where people can create private leagues to compete against their friends and see how they compete in public leagues against all users that have registered for the app. The user should be able to interact with the UI by creating a team of 5 players in which he thinks will do well in the game week of the real premier league. There will be a point system implemented for the players, this will make it competitive so people can play against each other for fun, the person with the most points at the end of the season will win. The point system will be something along the lines of: a goal scored by the player = 5, clean sheet for goalkeeper/defenders = 3, yellow card for player = -1, red card for player = -5. The person with the highest accumulated points at the end of the football year will be the winner.

## 6.2 Background

I chose this project because I wanted to do something involving football as it's a big passion of mine, both as a fan and player. As I grew older, the business side of football becomes something I loved hearing about as much as the games being played. Transfers and agents are something we see regularly talked about now a days. Contracts and negotiations have never been more complicated. There is always conflict with players, clubs and agents when trying to sign a new contract for the players parent club or when moving to a new club. Where seeing a lot more negotiations breaking down in this era which was unheard of if you go back 20 years ago. I think this type of project will be implemented on a higher scale in the future and there will be no agents causing problems in the football world anymore. Players will be paid the wage they deserve which their statistics reflect. As well as some teams paying over the price for certain footballers. Transfer fees will also be predicted as well so teams don't get ripped off for players who are overpriced.

Following my project plan as best I can, is my initial plan. Allowing some days for studying new technologies/tools I will be using and extra days for any coding difficulties I run into. Chipping away week by week will slowly but surely bring me to the completion of my project. Making sure I'm consistent with work weekly and checking in with my supervisor for advice and feedback on my progress. If I need to re-evaluate anything I will speak to my supervisor and get the best solution on my problem before implementing a new plan about tackling it.

## State of the Art

In my research I've found thousands of salary predictors but I'm yet to find one in the football industry. Predictors have been developed for many other occupations but none in the field I'm going to implement which gives it a uniqueness. This is a real-world problem that I think will be squashed with this exact idea in the future. Statistics have already been used once by a footballer playing in the Premier League who sacked his agent because he felt he could negotiate his contract himself. Kevin De Bruyne will be the instigator for salaries based on statistics. In the future, there will be much simpler negotiations with the help of these machine learning predictors.

Looking at the frontend side of my project, there is a similar application out there. The Premier League offer a fantasy football application. The user can register, login, create

private leagues between friends and view public leagues including the league of all users in Ireland and all users of the app in total.

On the Premier leagues fantasy website, the user must select 11 players to start the match week and has a bench of four players in this application and the point system is very basic compared to the one I plan to develop. My application will only require the user to pick 5 players. This means the user must pick their players carefully. Their points system is based on goals, cards, and clean sheets. Although my point system will include these its main points will be giving to the players for their match rating in the game, this will be a new feature for this type of application.

The user can transfer just one player each week for free and if they wish to transfer any more players, they will be deducted points from their total point count.

So although there is some similar functions and views, the Fantasy Futsal application will have many unique features and differ completely overall from the Premier leagues already popular website.

### 6.3 Technical Approach

Having never used any machine learning tools or REACT JavaScript framework before, I plan to watch some tutorials and study documentation on both these. This will be key for me to master both in my project. When I feel confident in both these areas, I will begin coding my project following a detailed project plan. Being new to the tools I'm using, I am expecting to miss some tasks in my plan. This will be updated anytime necessary during my project timeline in any place where needed.

Use Cases diagrams will be used to identify and explain requirements. I will aim for five or six use cases that will be the main functionality of my application. Each use case will be coded one after another into my project, my project plan will outline the tasks needed to do this.

Different algorithms will be trialled to see which works best, followed by a proof of concept to show how and why it's the best one for my application.

Again, the breakdown of my project will be through a project plan using an excel spreadsheet. This will break everything up nicely and give me a good indication visually where I am at with the aid of a Gantt chart.

My milestones will replicate the submission dates on Moodle. I will set targets for each submission date for the project, giving extra days for any unknown inconvenience that can occur.

### 6.4 Technical Details

Bootstrap will be used for styling the project giving it a nice touch and easy on the eye. JavaScript will be my main developing language used. This will call the API to retrieve the data I want for machine learning. Node.js will handle all backend JavaScript code. REACT will

be the JavaScript library I will use for the frontend of the application. Firebase will save a lot of time handling authentication and validations for registering and logging into the app. As well as handle the database for the project.

My machine learning will be done with TensorFlow. Tuning in TensorFlow to give an accurate output will be my most difficult task. I will chop and change the model created until I see accurate predictions, comparing to today's market. Firebase also offers a machine learning tool where you can upload your TensorFlow model to Firebase and it offers some cool features to play with, so this is an option for the machine learning part of the project.

Searching and sorting algorithms will be implemented to sort some data from highest transfer market player to lowest so I can create a view for the application. A search algorithm will also be developed so it will be possible to search for a particular player you are trying to find.

#### 6.5 Special Resources Required

At the moment there are no special resources are needed as of now in my project.

#### 6.6 Project Plan

Project plan is attached in upload with this document. Subject to change.

	A	B	C	D
1	Task	Start date	Duration	
2	1. Project pitch			
3	1.1 Research idea	16/10/2021	1	
4	1.2 Practice Video	16/10/2021	1	
5	1.3 Record video and upload	17/10/2021	1	
6	2. Project proposal			
7	2.1 Research technical tools/languages	18/10/2021	7	
8	2.2 Fill in proposal up to project plan	25/10/2021	3	
9	2.3 Create excel project plan	28/10/2021	4	
10	2.4 Finish off project proposal first draft and upload	01/11/2021	3	
11	2.5 Set up meeting with supervisor and get feedback	07/11/2021		
12	3. Setting up project workspace			
13	3.1 Create workspace on Visual studio	15/11/2021	1	
14	3.1 Create repo for frontend	15/11/2021	1	
15	3.2 Create repo for backend	15/11/2021	1	
16	3.3 Call in Node.js backend	15/11/2021	1	
17	3.4 Call in REACT for frontend	15/11/2021	1	
18	3.5 Research tools for frontend	16/11/2021	4	
19	3.6 Call in Firebase server for frontend	21/11/2021	1	
20	4. API (backend)			
21	4.1 Find suitable API	22/11/2021	1	
22	4.2 Get API key	22/11/2021	1	
23	4.3 Connect API in backend	22/11/2021	1	
24	4.4 Parse data to ensure working correctly	22/11/2021	1	
25	5. Machine learning			
26	5.1 Research TensorFlow/Firebase machine learning	23/11/2021	4	
27	5.2 Decide which tool suits project best	27/11/2021	1	
28	5.3 Import modules	27/11/2021	1	
29	5.4 Set seed value	27/11/2021	1	
30	5.5 Set up path directory	27/11/2021	1	
31	5.6 Read data in/process data	27/11/2021	1	
32	5.7 Create train/test data split	27/11/2021	1	
33	5.8 Create data shards	27/11/2021	1	
34	5.9 Processing batch and test data	27/11/2021	1	
35	5.9.1 Create model	27/11/2021	1	
36	5.9.2 Model aggregation	30/11/2021	5	
37	5.9.3 Model training	05/12/2021	1	
38	5.9.4 Repeat process until accurate prediction	05/12/2021	40	

	A	B	C
39	6. Midpoint presentation/documentation/video		
40	6.1 Create slide presentation	16/12/2021	
41	6.2 Fill in documents needed	17/12/2021	1
42	6.3 Practice video presentation	18/12/2021	1
43	6.4 Upload video presentation	18/12/2021	1
44	6.5 Upload codebase so far	21/12/2021	1
45	7. Login/ Register page		
46	7.1 Create register view	06/01/2022	2
47	7.2 Create validations for register page	06/01/2022	2
48	7.3 Create login page	08/01/2022	2
49	7.4 Create validations for login	08/01/2022	2
50	8. Statistics page		
51	8.1 Create statistics view	10/01/2022	2
52	8.2 Parse data from API to view for different stats about	10/01/2022	2
53	8.3 Create search bar for footballers	12/01/2022	3
54	9. Pick a team		
55	9.1 Create Transfer budget for team	13/01/2022	21
56	9.2 Create wage budget for team	13/01/2022	21
57	9.3 Create validation for 5 players	13/01/2022	21
58	9.4 Create other validations	13/01/2022	21
59	9.5 Parse predicted transfer value/wage for players over	13/01/2022	21
60	10. Leagues		
61	10. Create league view	03/02/2022	28
62	10.1 Create private league view	03/02/2022	28
63	10.2 Create button to generate private league with unique code for other members to join	03/02/2022	28
64	10.3 Create overall league view for all members	03/02/2022	28
65	11. Final upload		
66	11.1 Testing of application	01/03/2022	7
67	11.2 Thesis	08/03/2022	60
68	11.3 Video presentation practice	06/05/2022	
69	11.4 Video presentation upload	08/05/2022	
70	11.5 Viva examination prep	TBC	
71	11.6 Viva examination	TBC	
72	11.7 Project showcase prep	TBC	
73	11.8 Project showcase	TBC	

## 6.7 Testing

Test cases will be writing for each of the main functionalities of my project after implementation. I hope to put a regression test pack together containing all test cases and testing done for when my project is complete. Sanity testing will be done through each phase also to make sure validations and error handlers are in place and working correctly.

An end to end test will be the final test I complete to show the project is working as it should be.

For the integration testing I've done some research on the services I'll be using and found that a lot of them have been used together by other people which is a good indication. As there will be API's used in the project its essential to test if they work correctly with other technologies in the project. JQuery is an option for testing the JavaScript code. When coming to the end of the project, a decision will be made whether to use an external testing tool online. This decision will be made on the state of the project coming to the end of the semester.

#### 6.8 Ethics Approval Application (only if required)

Not needed so far in project

## 6.9 Reflective Journals

### 6.9.1 October

#### Supervision & Reflection Template

<b>Student Name</b>	Ross Buckley
<b>Student Number</b>	X18501723
<b>Course</b>	Bsc(Hons) in Computing

#### Month: October

##### What?

Reflect on what has happened in your project this month?

Last month I recorded a video pitching my project idea, explaining my plans on what tools, frameworks and languages I plan to use. The video was around three minutes long and explained as much as I possible could about what I plan to create.

Next, I wrote up my project proposal as detailed as possible at this moment in time.

Searched for an API suitable for my project and found the perfect one.

Starting looking at ethics forms and where I will need permission

##### So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

This meant a chunk of the planning for my project has been completed, leaving me with a good idea how im going to go about my work and what to mindful of. It gives me direction and I think when my project plan is finished I can start with small bits of coding next.

My successes include feeling prepared. I have a good understanding how to go about my work now, whilst before I was unsure where to start and how to progress.

Finding my API with the data I needed was a big success as a lot of data analysis projects fall short from not having enough or the correct data needed.

Challenges remaining include Ethics forms but I'm certain ill have this fixed in the next two days. Also creating a detailed project plan, I bound to miss some tasks as a lot of the function I plan to implement are completely new to me.



**Now What?**

What can you do to address outstanding challenges?

I plan to meet with my supervisor to clear up everything I need for the Ethics part of my project.

As for my project plan I need to research more about machine learning so I can write as many detailed tasks as possible that'll aid me progressing in my project. How to implement it first of all, followed by tuning it in for better accurate predictions.

**Student Signature**

Ross Buckley

## 6.9.2 November

### Supervision & Reflection Template

<b>Student Name</b>	Ross Buckley
<b>Student Number</b>	X18501723
<b>Course</b>	Bsc(Hons) in Computing (SD)

### Month: November

#### What?

Reflect on what has happened in your project this month?

The first task I did this month was receive feedback from my project proposal. From this feedback I updated my Project plan, filling in dates and adding more tasks that came to mind.

Next I created my projects workspace. Creating two separate repo's to work on backend and frontend independently, this is so it doesn't cause friction when moving to later stages of the project. For the frontend I called in the JavaScript REACT package, and also Firebase to handle stuff like database/machine learning.

For the backend, Node.js will be used to handle routes etc. Then I started calling in API's in the backend. So far there has been 5 called in and these will all be used for different parts of the project. A league, squad, transfer, player and search API are the five that will be put to use.

After this some thought has gone into the machine learning aspect of the project. Research was done and is being continued now.

Next, research for my technical approach was done. Getting familiar with the layout and what's going to be done, as well as updating my project proposal.

Met with Frances every Monday to discuss progress and problems.

#### So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

My success' so far have been deciding on my project idea 100%. Creating the workspace where it'll be coded.

Decided what libraries and frameworks will be used. Implemented the two repos for frontend and backend as well as called in packages to aid with these, REACT and Node.js.

Found suitable API's for project. Called in API's, routed them and parsed data to ensure its working correctly.

Researched machine learning.

**Now What?**

What can you do to address outstanding challenges?

More research on Machine learning needs to be done. Sticking to project plan is essential. Time management must be perfect as other module work is piling up to. Meeting with supervisor will be set up for feedback on Project proposal and tips on technical report.

**Student Signature**

Ross Buckley

### 6.9.3 December

#### Supervision & Reflection Template

<b>Student Name</b>	Ross Buckley
<b>Student Number</b>	X18501723
<b>Course</b>	Bsc(Hons) in Computing

#### Month: December

##### What?

Reflect on what has happened in your project this month?

This month consisted of connecting more API's, getting the structure of the project developed (navbar etc) and completing the login and register page

The midpoint presentation was complete.

Fixing up the project proposal and getting the requirements nailed down.

Video presentation of work to date and some presentation slides

Met with Frances every Monday to discuss progress and problems.

##### So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

This meant the project backbone is complete.

I have a good idea of how pages and the frontend will work.

I know what API's will be needed in different sections of the app.

I've worked with all the tools I plan to use so far so have a better understanding of all of them.

**Now What?**

What can you do to address outstanding challenges?

Next the machine learning model will be constructed to get predicted transfer market values and salary values

Before moving onto the frontend of the app

**Student Signature**

Ross Buckley

#### 6.9.4 January

##### Supervision & Reflection Template

<b>Student Name</b>	Ross Buckley
<b>Student Number</b>	X18501723
<b>Course</b>	Bsc(Hons) in Computing

##### Month: January

###### What?

Reflect on what has happened in your project this month?

The past month consisted of fixing the bug in the register/login as users' data wasn't being saved in the database but was allowing to register and login successfully.

I took a break from the backend as I was running into obstacles constantly trying to get the data needed and decided to focus on the frontend for now.

Navbar was created with all the pages and a logout button.

Going back to the backend some get requests created for pulling in data needed.

First was pulling in the current premier league players and getting their team ID's. This was done by creating a fetch players JavaScript function to get the team ID.

After this a fetch stats function is created to loop through the API and pull the stats of the players who have an ID of a premier league team.

These were then saved as CSV files, so I don't need to keep bouncing off the API with too many requests. The data pushed to the machine learning model needed to be CSV. Saved as English premier league data csv

Met with Frances every Monday to discuss progress and problems.

###### So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

Made a start on the frontend of the website.

Register and login system is working sound.

Step closer towards gathering all training and test data for machine learning model.

<p><b>Now What?</b></p> <p>What can you do to address outstanding challenges?</p> <p>Next is creating the components that are in the Navbar (website pages).</p> <p>Get all statistics for Premier league players for test data.</p> <p>Transfer just for the premier league isn't a large number so I must find more resources for more leagues for training data.</p>	
<b>Student Signature</b>	Ross Buckley

### 6.9.5 February

#### Supervision & Reflection Template

<b>Student Name</b>	Ross Buckley
<b>Student Number</b>	X18501723
<b>Course</b>	Bsc(Hons) in Computing

#### Month: February

##### What?

Reflect on what has happened in your project this month?

Fetch transfer function set up for transferred players.

Joined CSV's of players data (clubs, club ID etc) with the players statistics, easiest way to do this was using python and pandas library.

Route for pick team component fixed, pick team component designed, search bar for searching players implemented, button to pick player for team, team crest and player photo shown on search

Searching for more suitable transfer data because I still don't have enough training data

CORS bug fixed not sure where it came from, but website kept crashing due to it.

Met with Frances every Monday to discuss progress and problems.

##### So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

Step closer to getting all data needed.

Frontend pick team page designed the way I wanted.

Bugs fixed so website bug free again

##### Now What?

What can you do to address outstanding challenges?



Search web for data resources

Need to get more suitable training data and save the data to the Firebase data base.

Connect team and league component to database

Get machine learning model to predict accurate transfer and salary values

**Student Signature**

Ross Buckley

### 6.9.6 March

#### Supervision & Reflection Template

<b>Student Name</b>	Ross Buckley
<b>Student Number</b>	X18501723
<b>Course</b>	Bsc(Hons) in Computing

#### Month: March

##### What?

Reflect on what has happened in your project this month?

Stats.js function created for pulling transferred players statistics. This was another tricky problem as I only needed transferred players that cost a fee and there was a lot of loaned transfers and free transfers. Needed to fill in ID's manually to get the correct data.

Routes fixed for all other components/pages.

About (Rules and points system page) designed. Create league form created for users to create a private league.

Met with Frances every Monday to discuss progress and problems.

##### So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

Data gathering is improving and taking big steps to being complete.

Routes for each page complete and working correctly.

Create league page complete.

About page complete, just showing info about website and the points system for Fantasy futsal

##### Now What?

What can you do to address outstanding challenges?

Still need more training data for machine learning model, hard to find but I think I have a way to gather the info of transferred players I just need to create a function to get their ID's, then I can pull all their statistics from the API im using.

Need to connect the team and league component to the database to save users teams and private leagues created. Should be straightforward enough.

**Student Signature**

Ross Buckley

### 6.9.7 April

#### Supervision & Reflection Template

<b>Student Name</b>	Ross Buckley
<b>Student Number</b>	X18501723
<b>Course</b>	Bsc(Hons) in Computing

#### Month: April

##### What?

Reflect on what has happened in your project this month?

Fetch players version 2 function created and saved in database with team ID and player ID

A lot of projects and exams held me back on the final project as I needed to take time to focus on other modules with numerous deadlines coming around the same time

Met with Frances for until we finished up in college

##### So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

All ID's collected that were needed to extract the statistics needed for the machine learning model

All other modules finished late April so I can focus on completing the final year project and technical report

##### Now What?

What can you do to address outstanding challenges?

Every problem that was raised in the project is now over and I have good knowledge on how to complete the project, data collecting turned out to be extremely difficult

Machine learning model will be implemented once the training data function is set up and all frontend stuff will be complete after the machine learning model predicts accurate scores

More database tables will be created for users data, training data and test data

##### Student Signature

Ross Buckley

