



National College of Ireland

BSHCSD4

Software Development

2022/2023

Elijah McNamara

x19403554

x19403554@student.ncirl.ie

Football IQ Technical Report

Contents

Executive Summary	3
1.0 Introduction	4
1.1. Background	4
1.2. Aims.....	4
1.3. Technology.....	5
1.4. Structure	5
2.0 System.....	5
2.1. Requirements.....	5
2.1.1. Functional Requirements.....	5
2.1.1.1. Use Case Diagram	7
2.1.1.2. Requirement 1: Comparison feature	7
2.1.1.3. Description & Priority.....	7
2.1.1.4. Requirement 2: Smart filter search bar	9
2.1.1.5. Description & Priority.....	9
2.1.1.6. Use Case	9
2.1.1.7. Requirement 3: Hamburger bar.....	10
2.1.1.8. Description & Priority.....	10
2.1.1.9. Use Case	11
2.1.1.10. Requirement 4: Login screen	12
2.1.1.11. Description & Priority.....	12
2.1.1.12. Use Case	12
2.1.1.13. Requirement 5: Parallax and carousel	14
2.1.1.14. Description & Priority.....	14
2.1.1.15. Use Case	14
2.1.1.16. Requirement 6: Dark mode/light mode.....	16
2.1.1.17. Description & Priority.....	16
2.1.1.18. Use Case	16
2.1.1.19. Requirement 7: Call-to-action.....	17
2.1.1.20. Description & Priority.....	17
2.1.1.21. Use Case	17
2.1.2. Data Requirements	19
2.1.3. User Requirements	19
2.1.4. Environmental Requirements	19

2.1.5.	Usability Requirements.....	19
2.2.	Design &	20
2.3.	Implementation	20
2.4.	Graphical User Interface (GUI).....	22
2.5.	Testing.....	23
2.6	Evaluation	25
3.0	Conclusions	26
4.0	Further Development or Research	27
5.0	References	28
6.0	Appendices.....	28
6.1.	Project Proposal	28
	Objectives	28
	Background	28
	State-of-the-Art.....	29
	Technical Approach.....	29
	Technical Details	29
	Special Resources Required	30
	Project Plan	30
	Testing.....	31
6.2.	Ethics Approval Application (only if required)	32
6.3.	Reflective Journals	32
6.4.	Other materials used	42

Executive Summary

This executive summary provides an overview of the development of a football statistics website using Typescript and Next.js. The goal of this project was to create a modern and user-friendly website for football and statistics fans. The website includes features such as a powerful search bar, a comparison feature to compare players, and a top players feature. The website also offers good SEO and accessibility, while matching the competition in terms of performance.

One of the main strengths of the project is its scalability due to its development in Typescript. This allows for the addition of more features without affecting the site's performance or scalability. Additionally, the use of API's ensures that the information is up-to-date and dependable.

Despite its strengths, the project faced limitations such as time constraints and API restraints. Overall, this project has resulted in the development of a feature-rich and user-friendly football statistics website that is easily scalable and reliable.

1.0 Introduction

1.1. Background

As football and statistics fascinate me, I realised that it would be an excellent idea to combine the two. Furthermore, I learned the basics of JavaScript and React during my internship, so I decided that implementing this technology would be a smart and challenging way to tackle such a project, while also massively improving my skills, as this is the technology that I want to use going forward in my career. Moreover, when researching other football statistic websites, I noticed no implementation of React, along with subpar design and information presentation. This gave me the motivation to improve upon this, while also learning a lot about modern programming technologies, while giving me the chance to develop my career.

I chose this project out of any other potential candidate, because I knew it would be something I would enjoy to develop and research. As this is a big undertaking, I wanted something to motivate me, and I wanted to know I was improving upon what was already available, instead of doing something because it seems the easiest and involved the least amount of dedication. Furthermore, I wanted to implement features that interested me, and work on ideas and software that always puzzled me because of how they worked. Design, React, API's, and problem solving were all main reasons for me to start development on this, and during the development stage, learning about how to improve on loading speeds, compression, and general good habits excited me. Coupled with my passion for football and interest in statistics, this project idea stuck out further than the rest, and is why I decided to choose this project. Finally, I also wanted to host the completed project and purchase a domain name to make it live so people can use what I developed.

1.2. Aims

My project aims are:

- **Compete:** I want this project to compete with other similar websites on the market in a data and information viewpoint, but overtake them in a design, technology, and user experience aspect.
- **Design and absence of modern technologies:** My goal is to combat this the best way that I can while learning how to develop useful and wanted technologies in the industry.
- **Think like a programmer:** This aim is personal to me, but especially important, as I want to go through the stages of not knowing how, to struggle, failure, learning, overcoming, and succeeding.

1.3. Technology

The technologies that I have chosen to use to complete this project are as follows: React, JavaScript, Next.js (includes Node.js), selected API's, HTML, styled components (CSS), GitHub, Figma, external resources for assistance, a hosting platform, and VScode. This is the main tech stack that I plan on using, although more could be added to it in the future.

JavaScript, React, and Next.js are the main languages, libraries, and frameworks that I plan on using. This selection will build the majority of the site and supply most of the functions that are available throughout. JavaScript will be the base language I'll be working with, while React will be the framework that I'll use to write the code (compiles down to JavaScript), while Next will supply the server-side rendering, and making the development generally quicker, more user-friendly, and offering better performance.

The API's that I plan on using will provide the football statistical content that dynamically updates, with the styled-components being an npm package that uses the best bits of ES6 and CSS to style apps. Moreover, I will use Figma wireframing software to mock up the rough wireframes of each component that I plan to develop.

I am planning on hosting the website on a platform such as 'Render' or 'Netlify' to manage the final project when it's completed. I will develop the project on VSCode because it's a lightweight editor with a good UI, and GitHub and its active workflow to make branches, commit and push code, merge branches to the main, review pull requests, and to improve on CI/CD.

1.4. Structure

Section 2.1 contains information regarding the system itself, such as the functional and non-functional requirements. Section 2.2 illustrates the design architecture, with implementation detail in Section 2.3. Wireframes of the application can be seen in Section 2.4, testing described in 2.5., and Section 3.0 outlines the conclusions and final thoughts surrounding the project.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

Below is a ranked list (in order of highest importance) of the functional requirements in my project. These requirements are the backbone of my project, as they are what the user is going to interact with the most, as they serve the most amount of use, output, and feedback. These requirements include:

- 1.** Comparison feature (compare any league or player, in a specific time range (season stats or all-time stats)) – this feature provides data on players, clubs, and leagues. My system accomplishes this by using multiple API's, allowing the user to choose between records from the same category (e.g., player and player, not player and club) and making it visible throughout the application. I aim to incorporate this data behind every functional button click on the website, as to avoid the retain the users attention for the longest possible time.
- 2.** Smart filter search bar – this feature provides the user with the accessibility and ease-of-use in navigating all the information from the API. This feature will smartly filter the sites' information with every character that's typed in the search bar, to gather the most relevant information the user is looking for.
- 3.** Hamburger bar – this feature stores the system and account settings, as to avoid clutter on the main pages of the website. UI is important and is a lacking addition to many of the competitors in this space, so freeing up space and having it all in one area, makes more room for the features and statistics on the page.
- 4.** Login screen – this is the first page that the user will see upon visiting the site. It will prompt the user to login or signup using their credentials, along with the standard necessary dependencies (Forgot your password?, etc). Upon entering these details, the user will be brought to the main page of the site where most of the functional requirements will be accessible.
- 5.** Parallax and carousel – these are visual methods of interacting with the webpage the user on, by prompting the user with a slide, quickly escorting them top another part of the site, without the need of additional navigation.
- 6.** Dark mode/light mode – gives the user the ability to customise how their webpage looks and feels when navigating through the app. The functionality is housed behind a single button that will be on every page to make it easy to go back and forward.
- 7.** CTA's (call-to-action) – this is a feature that's used on my application to share any statistic that the user wants to social media, by implementing an action button labelled 'share'. Other implementations of this feature include the 'Sign up' button in my login page and is a feature that I plan on using more throughout the application.

Other requirements that are non-functional include:

- User-friendly UX and UI.
- Infographics to go with stats
- Fast loading speeds.
- Domain name.
- Responsive design.
- Transfer news section.
- Facts section (historical data from an API).
- Testing code with Jest.
- Mobile first design.
- Deployment to server.

2.1.1.1. Use Case Diagram

2.1.1.2. Requirement 1: Comparison feature

2.1.1.3. Description & Priority

This is the main functional requirement of the project and is number 1 priority, as the idea behind this application function-wise, is to be able to easily view stats and compare them with that of the corresponding category.

Use Case

Requirement 1:

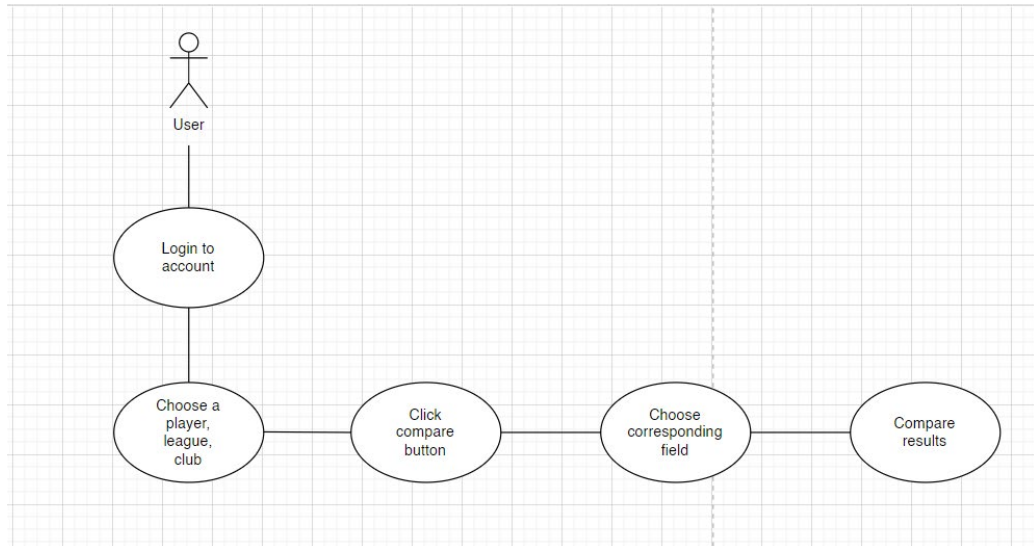
Scope

The scope of this use case is to allow the user to click a button and allow them to compare the stats of players, leagues, and clubs.

Description

This use case diagram below outlines the process of the user clicking a button and being able to compare statistics.

Use Case Diagram



Flow Description

The user chose to compare two corresponding fields by clicking the compare and was shown the results.

Precondition

The user must be logged in to access this feature.

Activation

The user must choose at least one field before they have the ability to compare.

Main flow

1. The user logs or signs into their account.
2. The user clicks on a field or uses the smart filter search bar.
3. The user clicks on the compare button.
4. The user chooses a corresponding field to compare.
5. The user is shown the results of their actions.

Alternate flow

A1: user doesn't click corresponding field.

1. The application doesn't proceed with action.
2. The application prompts the user to choose corresponding field.
3. The user chooses an appropriate field to continue with their action.

Exceptional flow

E1: The user wants to compare more than two corresponding fields.

4. The application let's the user know only 1 comparison can happen at a time.
5. The continues with their comparison.

Termination

The user leaves the comparison screen by clicking the 'x' button.

Post condition

The application returns the user back to where they were before the comparison screen.

2.1.1.4. Requirement 2: Smart filter search bar

2.1.1.5. Description & Priority

A description of the requirement and its priority. Describes how essential this requirement is to the overall system. This is a feature that smartly filters search results based on the characters that the user enters. It is a number 2 priority for the project, as it allows the user to quickly search anything to show them statistics.

2.1.1.6. Use Case

Requirement 2:

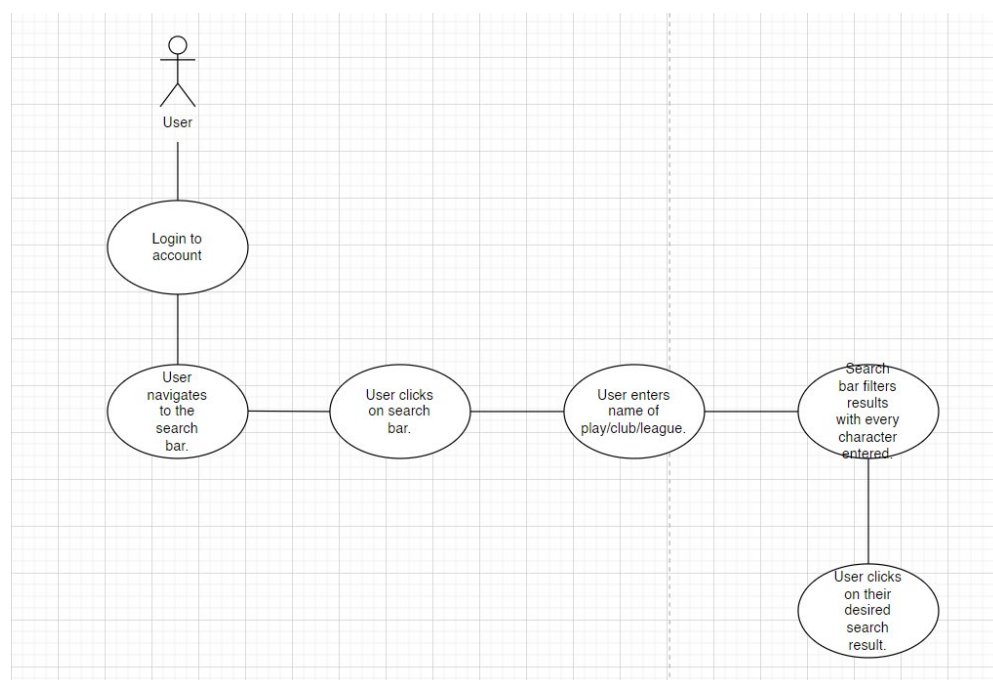
Scope

The scope of this use case is to provide a search bar function at the top of every page that transports the user to their entered search query.

Description

This use case diagram below outlines the user clicking the search bar and utilising its functionality.

Use Case Diagram



Flow Description

User does the necessary actions to get their desired information.

Precondition

The user must be signed into an account before they can use the search bar.

Activation

The user must click the search bar and enter a character to use this feature.

Main flow

6. The user logs into their account.
7. The user clicks the search bar.
8. The user enters a character to begin using the feature and filter the results.
9. The user clicks on their desired search.

Alternate flow

A1: Entered search doesn't match any result on the API/web page.

6. The application tells the user that no results can be found.
7. The user alters their search term.
8. The user finds their desired search result and proceeds with their use of the site.

Exceptional flow

E1: The user is already utilising the comparison functional requirement.

9. The application hides the navigation bar to focus on the content.
10. The user must stop using this feature in order to use the search bar.
11. The user backs out of this feature to access the search bar again.

Termination

The user clicks on a surrounding area of the screen, or they choose a search result.

Post condition

The application brings the user to their desired search result.

[2.1.1.7. Requirement 3: Hamburger bar](#)

[2.1.1.8. Description & Priority](#)

This part of the project is a number 3 priority requirement, and its purpose is to declutter the main pages of the application by putting all of the system and account settings under one button click to make navigation and the UI appeal better to the user.

2.1.1.9. Use Case

Requirement 3:

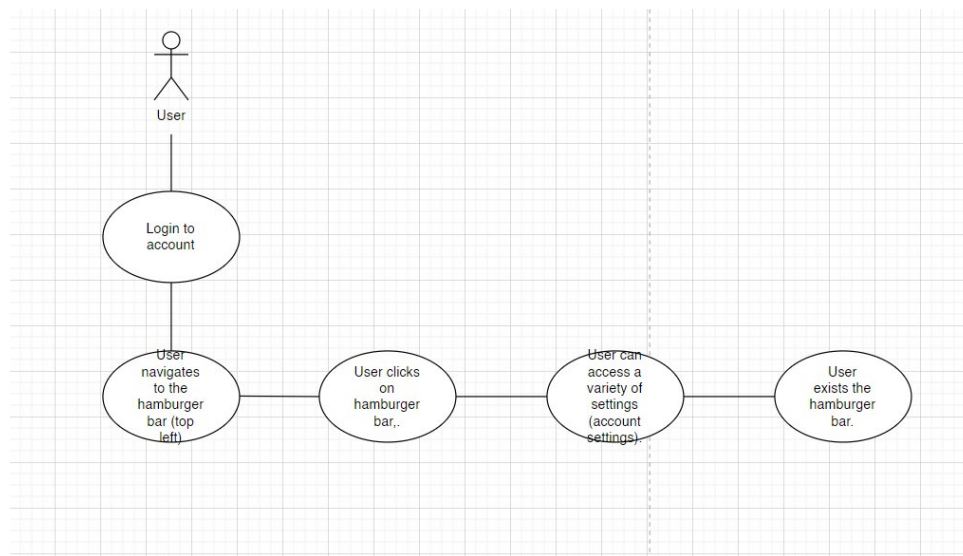
Scope

The scope of this use case is to put all of alterable site setting in one place.

Description

This use case diagram below outlines how the hamburger bar is accessible and what it entails.

Use Case Diagram



Flow Description

Application provides a good UI that indicates where the site/account settings lie for the user to access.

Precondition

The user must not be using any other function requirement.

Activation

This user must have an account to see option, which is then clickable to navigate to the settings.

Main flow

10. The user has an account.
11. The user isn't using any other functional requirements.
12. The user clicks on the hamburger bar in the navigation bar.
13. The user changes their desired settings.

Alternate flow

A1: Hamburger bar doesn't get used.

12. The user is using the other functional requirements and is using the site for its statistics uses.
13. The user has no use for the hamburger bar.
14. The user will never use the hamburger bar.

Exceptional flow

- E1: Hamburger bar doesn't provide setting that user wants to change.
15. The user doesn't have access to a particular setting that they're looking for.
 16. The user leaves the hamburger bar and returns to using the app.
 17. The use hamburger bar gets used less frequent.

Termination

The user clicks a 'back' button to leave the hamburger bar.

Post condition

The user is brought back to the page where they were on before they accessed the hamburger bar.

[2.1.1.10. Requirement 4: Login screen](#)

[2.1.1.11. Description & Priority](#)

The login/sign in screen is the first screen that the user will see once they visit the website and is a level 4 priority requirement, as although required to see the statistics and use the other functional requirement, it's not required thereafter, only to sign in.

[2.1.1.12. Use Case](#)

Requirement 4:

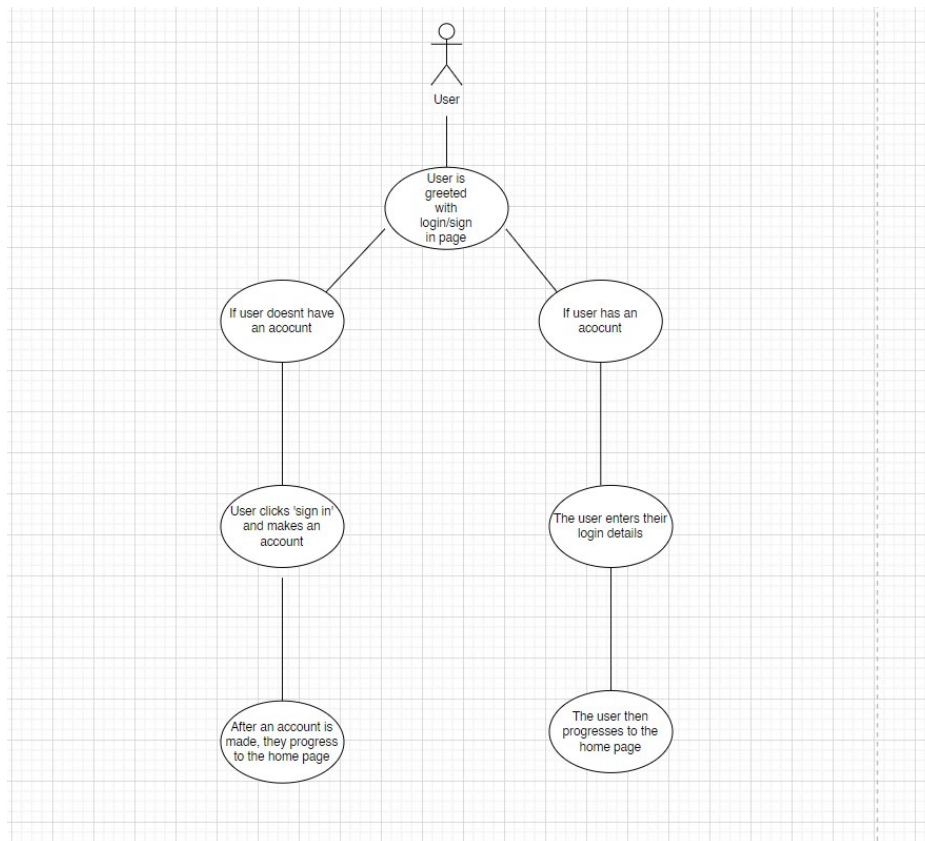
Scope

The scope of this use case is to allow a user to make an account to Football IQ, and access all of the live and dynamic statistics and information, and to use the functional requirements on the site.

Description

This use case diagram below outlines how a user can use the login/sign in screen, depending on their account status.

Use Case Diagram



Flow Description

Upon visit of the site, the user is met the option to sign in or log in.

Precondition

The user must have an internet connection.

Activation

The user must enter the correct URL into the web browser.

Main flow

14. The user visits the application.
15. The user enters their login details.
16. The application verifies the user.
17. The user is brought to the main page of the site.

Alternate flow

- A1: The user doesn't have an account.
18. The application provides the user an option to make an account,.
 19. The user enters an email address and a password.
 20. The user makes an account
 21. The user is brought to the main page of the site.

Exceptional flow

E1: The user forgets their password.

22. The application will provide the means for a user to change their password.

23. The user successfully changes their password.

24. The user re-enters their login details.

25. The user is brought to the main page of the site.

Termination

The login page will terminate once the user successfully enters their login details, or the user leaves the application completely.

Post condition

The user is brought to the main page.

[2.1.1.13. Requirement 5: Parallax and carousel](#)

[2.1.1.14. Description & Priority](#)

This part of the project gives the user another way of interacting with the information on the screen and provides them a different way to navigate to various parts of the site. It is a number 5 requirement as it doesn't provide any core functionality, as it is mainly a feature for visual flair and ease of use.

[2.1.1.15. Use Case](#)

Requirement 5:

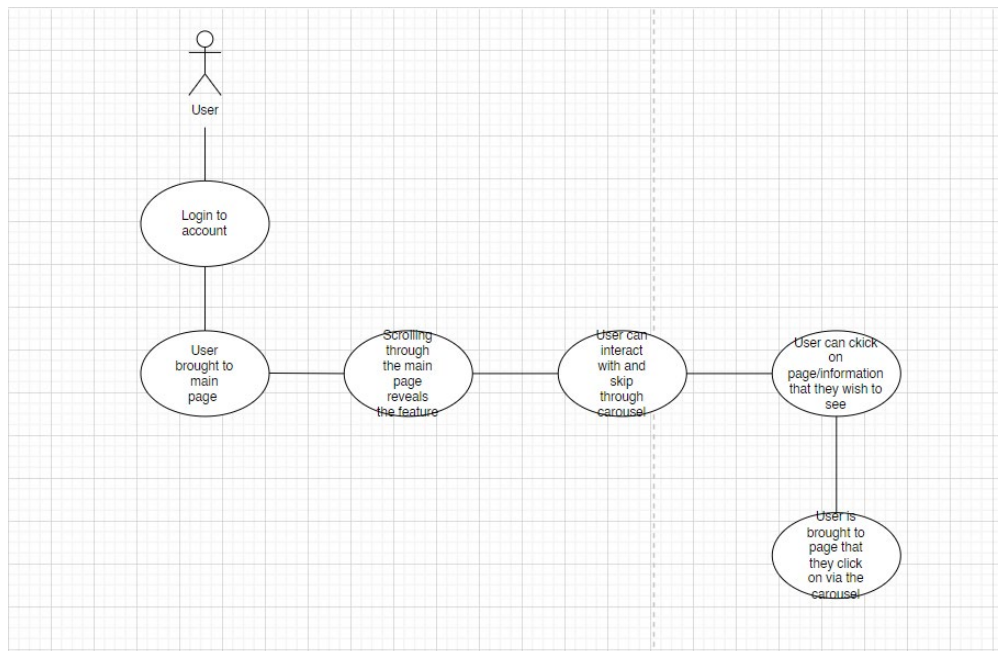
Scope

The scope of this use case is to allow for the user to navigate through the site easier and to notify them of different pages that they might not have known otherwise.

Description

This use case diagram below outlines how the user can interact with the parallax and carousel in the application.

Use Case Diagram



Flow Description

User can navigate through the application quickly and efficiently via the carousel.

Precondition

The user must be signed in and navigate through main page.

Activation

Clicking on the carousel slide will activate the feature and.

Main flow

18. The user logs into the application.
19. The user navigates to the carousel via the main page.
20. The user clicks on their desired slide.
21. The user is brought to the page which they clicked.

Alternate flow

- A1: The user uses other methods of navigation.
26. The user uses the search bar and other tabs on the site to navigate.
 27. The user clicks on their desired statistic.

Exceptional flow

- E1: The carousel provides little use.
28. The user navigates the site without help from the carousel.
 29. The user uses options which are immediately obvious.
 30. The carousel gets little use due to the user not scrolling far enough to access it.

Termination

The user clicks on a slide and is brought to another statistics page, or the user scrolls on the site, deactivating the carousel.

Post condition

The carousel become unactive and isn't used unless the user clicks the arrow button.

2.1.1.16. Requirement 6: Dark mode/light mode

2.1.1.17. Description & Priority

This functional requirement allows the user to alter the visual state of the application, allowing the sense of ownership and customisability when on the site. It is a number 6 priority, as the website can function as expected without it.

2.1.1.18. Use Case

Requirement 6:

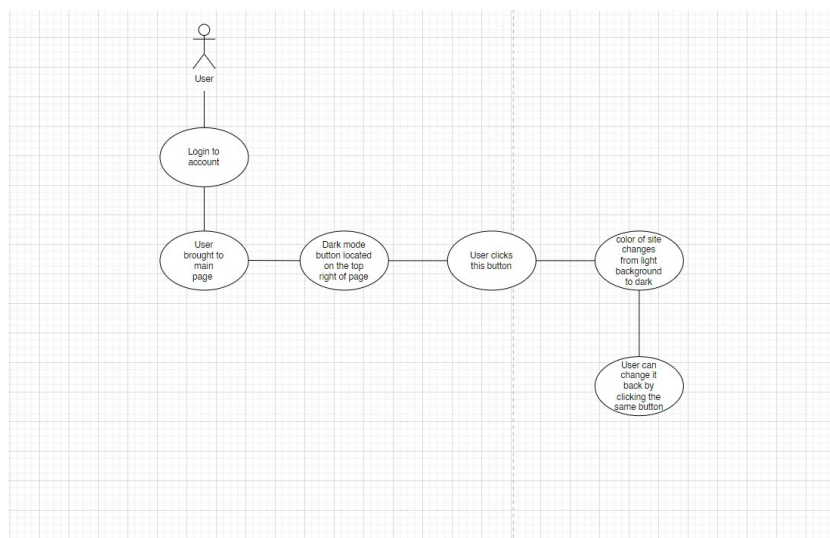
Scope

The scope of this use case is to allow the user to change the look and feel of the application when navigating through its pages and content.

Description

This use case diagram below outlines how a user can interact with the dark/light mode button.

Use Case Diagram



Flow Description

User can change the look and feel of the site by clicking the dark mode button.

Precondition

The user must be signed in.

Activation

This use case starts when the user clicks the button.

Main flow

22. The user logs into the site.
23. The user clicks the button located in the top right corner.
24. The application changes appearance to the opposite of whatever the application was already on.
25. The user can change the state back by clicking the same button.

Alternate flow

- A1: The user accesses the button via the hamburger bar.
31. While the user is in the settings (hamburger bar), they change the appearance here, rather than in the navigation bar.
 32. The application's appearance changes.
 33. The user can change the appearance back by clicking the dark mode button in the navigation bar or in the hamburger bar.

Exceptional flow

- E1: Contrast between information on screen and background don't match.
34. The application's appearance is worse on dark mode due to poor contrast.
 35. The user doesn't use dark mode for this reason.
 36. This functionality and feature is taking up resources and must be fixed.

Termination

Clicking the dark mode will return the application back to the original colour.

Post condition

Light mode is default, and the user has the option to return it to dark mode.

[2.1.1.19. Requirement 7: Call-to-action](#)

[2.1.1.20. Description & Priority](#)

This feature allows the user to share information and statistics they find, online and on social media. It is a number 7 priority as its function is to gain more users and attraction to the site, and is disguised as a feature.

[2.1.1.21. Use Case](#)

Requirement 7:

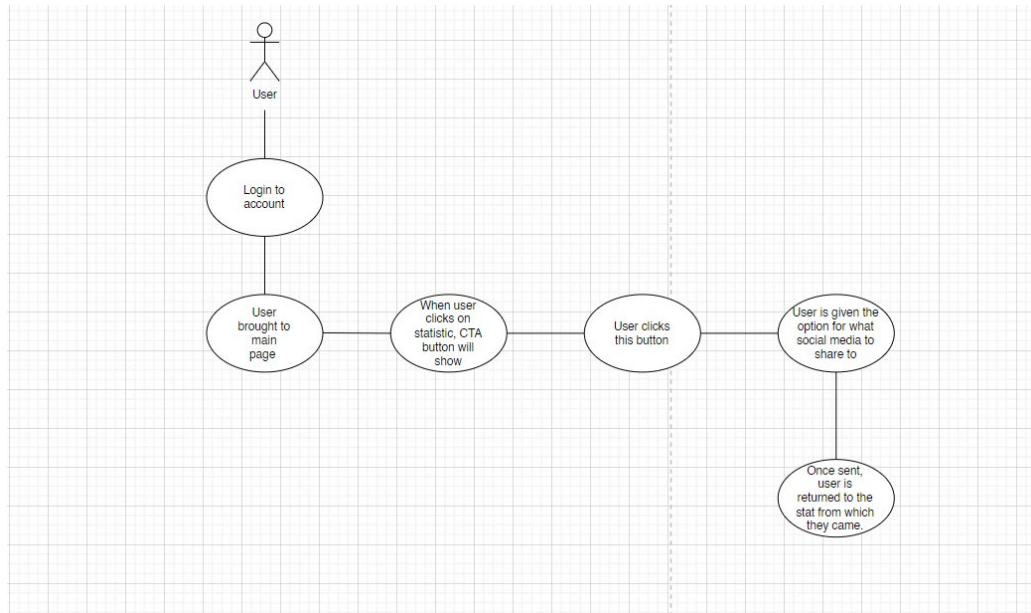
Scope

The scope of this use case is to share information on social media.

Description

This use case diagram below outlines how a user may interact with this call-to-action feature.

Use Case Diagram



Flow Description

User can share statistics straight from the stat page to any social media platform.

Precondition

The user must be logged in and be using the comparison feature for this feature to work.

Activation

This use case starts when a user is comparing statistics and clicks the 'Share' CTA button on the stat page.

Main flow

26. The user logs into application.
27. The user uses the comparison feature.
28. The user sees the CTA button beside the stat.
29. The user clicks it and shares it on social media.

Alternate flow

- A1: User doesn't use the feature.
37. The user logs in to their account.
 38. The user uses the comparison feature.
 39. The user sees the CTA button but chooses not to use it.

40. The user leaves the comparison screen.

Exceptional flow

E1: Feature takes too long to load

41. The user logs in to their account.

42. The user uses the comparison feature.

43. The user clicks it and shares it on social media.

44. The user cannot continue due to long loading speeds.

45. The user returns to the stat page.

Termination

The user leaves CTA by clicking the 'x' button.

Post condition

The user is returned to the page from which they came.

[2.1.2. Data Requirements](#)

This application will use multiple API's to get the information needed for the statistics. One of these API's include 'API-Football' from a reputable site called Rapid-API. Here, is where I get all of the information the user could need, such as, line-ups, coaches, players, top scorers, standings, statistics, transfers, and predictions. In addition, there will be more API's used in the future, especially for the transfer news section and the facts section (non-functional requirements), as this will need live and fresh data to work, and the API mentioned doesn't provide this data.

[2.1.3. User Requirements](#)

The user should expect this application to provide all necessary information on football around the world, from the Premier League standings to the top scorer in a specific team. The data provided should also be accurate, and update as changes occur. Th user may also have access to many features to use as a means of visualising this data, while navigating a modern, user-friendly site, that's easy on the eye and intuitive. Nothing is required from the user other than the creation of an account, and they can have full access to what the site provides.

[2.1.4. Environmental Requirements](#)

There are no environmental requirements for the development of this project, as there are no risks relating to health, safety, or the environment. The website will not be demanding on servers and will require average CPU usage. Furthermore, no laws will be broken, as access to my API's are free and I have authorisation for its use.

[2.1.5. Usability Requirements](#)

I aim to make this application accessible to all demographics in its design, while the content is designed to target football and statistic-minded people. It will be straight-forward to learn the features, as everything will be labelled correctly and designed intuitively. The performance will only be bottlenecked by the users internet speed, as the application will be developed using a modern tech-stack and up-to-date

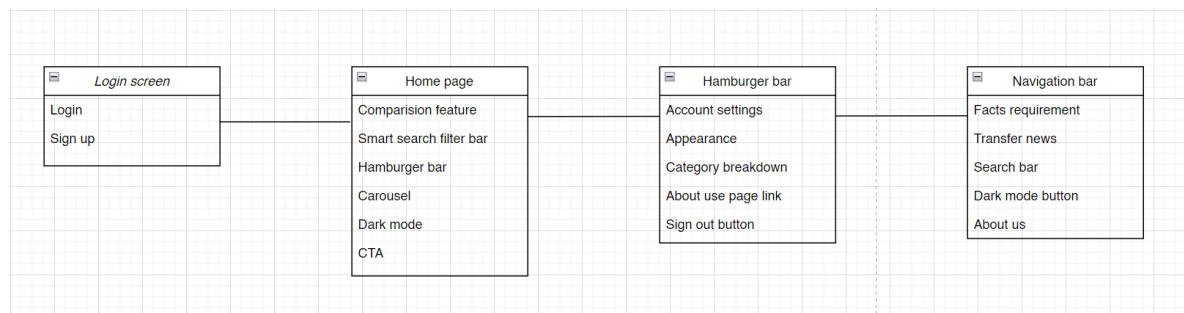
development practises. In addition, the user-friendliness is a major part of the development, as everything will serve a purpose, with the content taking centre stage, and the functional/non-functional requirements acting as supports to deliver this content. Error handling will be a pivotal part of my development, so with frequent unit and increment testing, along with an active workflow set up in my GitHub repository to automatically test the code as I push it, errors will be nullified as much as possible.

2.2. Design &

The design takes on the appearance of any modern web interface built around my system requirements and the content that it provides. This design is built using the Figma software (see 2.4. Graphical User Interface for more) and its components are developed with the React framework. The behaviour surrounding the application is based off common React syntax, called hooks. 'useState', 'useEffect', and the component tag are important to develop the behaviour, along with standard JavaScript syntax, such as 'onClick' and 'onmouseover' will play a part in the behavioural component of the application.

Some of the algorithms used in the application include the 'Big O' notation, when using arrays, iterations, recursions (to write clean, more efficient code), lists, and more. These are the bases of my code and is how I plan to meet one of the non-functional requirements, being loading speed, as using proper development practises will result in better written code and less errors.

An architecture diagram can be found below containing the main functionality and where the user can find them:



2.3. Implementation

My system uses multiple API's, allowing the user to choose between records from the same category (e.g., player and player, not player and club) and making it visible throughout the application. I aim to incorporate this data behind every functional button click on the website, as to avoid the retain the users attention for the longest possible time.

Furthermore, as mentioned in section 2.2, the main algorithms used in the application are the 'Big O' notation, arrays, iterations, recursions, and lists, and will all play their parts in the development of the application. In addition to these algorithms, I will be using class components in React, which is standard across most modern web apps built in ReactJS. These components are simple classes made up of multiple functions that add functionality to the application. Typescript works as a superset of JavaScript which lets you add type information to your code. This then lets you find bugs in your code quicker. An example of some Typescript code can be found below:

```
export default function Home({ players, team }: IHome) {
  return (
    <Container>
      {players.map((player: player, i: number) => {

        return (
          <div key={i} value={player.name}>
            {player.name}
            {player.number}
            {team.name}
            <img src={team.logo} alt="ff" />
          </div>
        );
      })}
    </Container>
  )
}
```

This code imports players and teams from the API that I'm using and from the 'IHome' type. Then, it maps over the players' array and returns a div with the players name, number, team name, and badge.

The following code demonstrates how the API is implemented and where the data comes from:

```
import axios from "axios";
import type { AxiosResponse } from "axios";

async function getPlayersBySquadId(id: string):
Promise<AxiosResponse<any>> {
  const response: AxiosResponse<any> = await axios.get(
```

```

    `https://api-football-v1.p.rapidapi.com/v3/players/squads`,
    {
      params: { team: id },
      headers: {
        'X-RapidAPI-Key':
'3e93f54308mshcc56d624809a4a9p144a30jsn829d33d2f0e4',
        'X-RapidAPI-Host': 'api-football-v1.p.rapidapi.com'
      }
    }
  );

  return response;
}

export const playersApi = {getPlayersBySquadId};

```

This code imports Axios from the Axios library, then defines a function that takes in a squad ID and returns a Promise of an AxiosResponse. Axios is used to make a GET request to the API, where Squad ID is passed as a parameter to the API. Finally, the headers are passed into the API, and Axios returns a response.

2.4. Graphical User Interface (GUI)

Mock-Up 1: This first mock-up outlines the general look of the login screen, before the user can get access to the main parts of the application. The screen begins with a welcome message and lets the user know what to do next, which is to enter their details. After the details are entered, the sign in button has to be clicked to progress to the home page of the site. If a user doesn't have an account, the bottom of the mock-up gives the user the option to make an account by clicking the bold text which acts as a link to an account creation page.

Mock-Up 2: This mock-up is what the user is going to see once they sign into their account. This is the home page, where the user will find all of the requirements and how they will interact with the site. The top of the page will say the name of the site, and underneath will have a separating line with the hamburger bar, search bar and dark mode function just below that. Beneath that will be dropdown menus for the categories on data that the application provides, such as teams, players, leagues, competitions, and countries (this could be expanded). The main component of the page includes what the comparison feature looks like. It will be a component where the user selects corresponding fields (player and player, not player and club) to compare. The triangles represent the field (player) and underneath shows the stats.

Figure 1 Mock-up 1

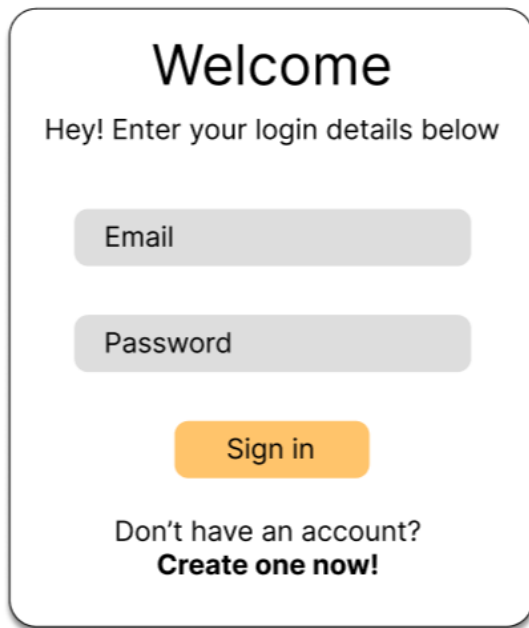
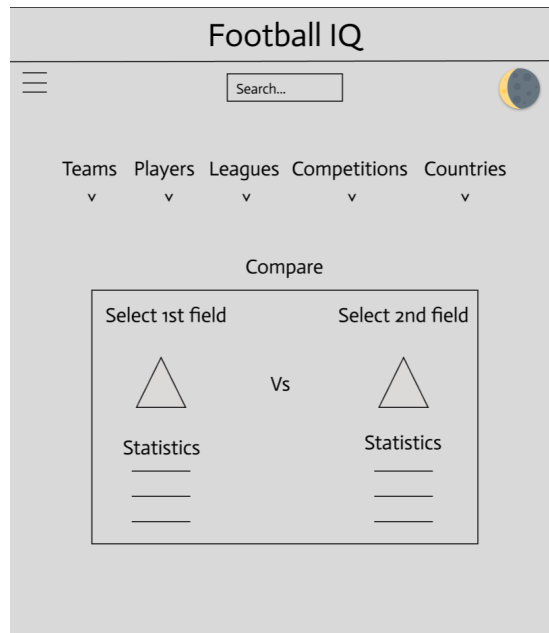


Figure 2 Mock-up 2



2.5. Testing

The testing framework that I used to carry out the testing for this project is Jest, made for React/JavaScript. The unit tests have been organized into different test suites based on the functionalities of the application, and the integration tests are used on the API's that I used that communicate with the rest of the application. Furthermore, I used system testing at the end to ensure that the functionality of the end result of the code was uncompromised. The results of the tests can be found below.

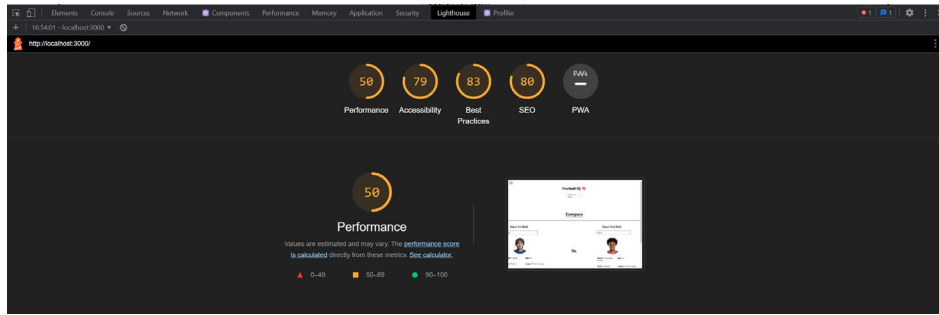
System Testing:

System testing is an essential part of the software development process. It ensures that the system meets the specified requirements and functions correctly. During the system testing of Football IQ, I executed various tests to identify defects and ensure that all functionalities work as intended.

Types of System Tests Performed:

Functional Testing: Functional testing verifies that the website functions as expected by executing test cases that simulate user behaviour. During functional testing, I tested the website's search functionality, navigation, and other features to ensure that they work correctly. This was successful as I encountered zero bugs or errors.

Performance Testing: Performance testing was done to ensure that Football IQ'S loading speed, among other things, are optimized and that it can handle multiple user requests simultaneously without crashing. I conducted a lighthouse test which is a software made by Google that can test the performance, among other statistics from the developer tolls in the browser. The results can be found below:



Security Testing: Security testing was done to ensure that the website is secure, and that user data is protected. The only security risk in my website is in regard to the login feature, and that is handle by Next auth and Google.

Compatibility Testing: Compatibility testing was performed to ensure that the website works correctly across different browsers, operating systems, and devices. During this testing, I verified that the website is useable and adapts so that it doesn't break or become dysfunctional.

Test Results:

All system tests were conducted successfully, and the website passed all the tests. Functional testing confirmed that all features work as intended, performance testing revealed that the website is fast and responsive, security testing confirmed that the website is secure and no vulnerabilities were found, and compatibility testing confirmed that the website works well across all platforms.

Integration Tests:

Part of my testing strategy is to implement integration testing to test the interaction and communication between different software modules or components that have already been tested independently. I wrote a set of integration tests to ensure that my website's APIs are working as expected, however, I ran the tests, and only one test passed. I will outline the tests I wrote, the test that passed, and the actions I will take to fix the tests that failed.

The test to check the players statistics API worked, but the tests to check the other statistic and data API's, such as teams and leagues, failed.

Actions to Fix the Failed Tests

To fix the failed tests, I will first look at the test that passed and analyse it to see why it succeeded. I will ensure that the code structure of the passed test is consistent with the failed tests and try to isolate the differences in the code.

After analysing the passed test, I will look at the error messages from the failed tests and try to understand the nature of the error. For the API's, the error messages include messages like "teamStatisticsApi - getTeamStatisticsById - should return team statistics by team and league"

and “Cannot find module 'api/leagues' from '__integration-tests___/leagues.integration.test.ts’”. I will use appropriate resources to counter these errors and use the passing test as shown below to fix them.

Once I have identified the issues with the failed tests, I will make the necessary changes to the code and run the tests again. I will continue this process until all tests have passed.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS
at Object.require (__tests_/comparison-default-player.test.ts:2:1)
FAIL tests_/handle-select-change.test.ts
  ● Test suite failed to run

    Cannot find module './useTeamSelector' from '__tests_/handle-select-change.test.ts'

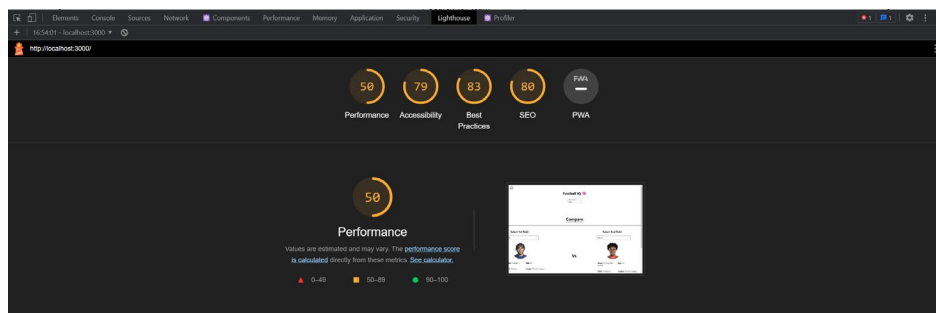
      1 | import { renderHook } from '@testing-library/react-hooks';
      2 | import { act } from '@testing-library/react';
    > 3 | import useTeamSelector from './useTeamSelector';
        | ^
      4 | import { teamsApi } from '../api/teams';
      5 |
      6 | jest.mock('../api/teams');

    at Resolver.throwModNotFoundError (node_modules/jest-resolve/build/resolver.js:427:11)
    at Object.require (__tests_/handle-select-change.test.ts:3:1)

Test Suites: 8 failed, 1 passed, 9 total
Tests: 3 failed, 1 passed, 4 total
Snapshots: 0 total
Time: 3.923 s, estimated 9 s
Ran all test suites.
PS C:\Users\Elijah MacNamara\Dropbox\Year 4\Computing Project\Football_iq>
```

2.6 Evaluation

The aforementioned software ‘Lighthouse’ was used to analyse the system testing, and to see where could be improved. The following is an in-depth look at these results:



These results are from the Lighthouse report which provides an overview of how well my website performs in terms of different metrics. The results are on a scale of 0 to 100, with higher scores indicating better performance.

The performance score of 50 indicates that the website is performing at a moderate level. This score is impacted by numerous factors such as page load times, time to first byte, and other performance-related metrics. A low score in this area can indicate issues such as slow server response times, large or unoptimized images, inefficient code, or API issues.

The accessibility score of 79 indicates that the website is moderately accessible to users with disabilities. This score is impacted by factors such as the use of appropriate HTML tags, adequate colour contrast, and proper keyboard navigation. A low score in this area can indicate issues such as insufficient contrast between foreground and background colours, poor use of headings, or missing alternative text for images.

The best practices score of 83 indicates that the website follows a good set of web development best practices. This score is impacted by factors such as proper use of HTML and CSS, adherence to security standards, and use of modern web technologies. A low score in this area can indicate issues such as outdated code, insecure connections, or poor use of web standards.

The SEO score of 80 indicates that the website is moderately optimized for search engines. This score is impacted by factors such as the use of appropriate meta tags, proper use of headings, and quality of the website's content. A low score in this area can indicate issues such as duplicate content, missing meta tags, or poor-quality content.

Overall, while some of these scores are moderate, there are still areas where the website could be improved. These issues can be improved upon throughout its life, which can lead to a better user experience and increased traffic.

3.0 Conclusions

There are several advantages and disadvantages, strengths, and limitations to this project. The website is built to provide a user-friendly interface that is easy to navigate while providing in-depth football statistics and player comparison features. Some of the key features of the website include a powerful search bar, a dropdown menu, dark mode, infographics, Google authentication, a comparison feature, a share button, top players feature, and an about us page.

One of the significant advantages of the project is its user interface and user experience. The website's design is modern and sleek, with a clean interface that is easy to navigate, with well presented features. This makes it easy for users to find the information they need and perform searches without difficulty. Additionally, the search bar and dropdown menu make it simple for users to find the information they are looking for. The infographics are a welcome addition touch that provides visual context to the statistics, rather just having numbers on the screen.

Another strength of the project is its use of API's to gather information. The API's allow the website to stay up-to-date with current statistics and provide accurate information. The use of Google authentication is also beneficial, as it allows users to create accounts and sign in easily using 2FA.

Despite the advantages of the project, there are also some disadvantages and limitations. One significant disadvantage is the reliance on API's. If an API is down or unavailable, the website's functionality will be impacted. This can also lead to issues with the accuracy of the information presented on the website. Additionally, the website's focus on football statistics may limit its appeal to those who are not interested in football or sports in general.

The project's strengths are also limited by the resources and time constraints I had. Creating a high-quality website with advanced features like a comparison feature, top players feature, and infographics requires a lot of time and resources, especially since I didn't know Typescript/React or use a Next.js application before, it was a struggle to add small features,

which meant adding larger ones took up a lot of time, and this was on top of other modules and assignments that I had to complete on top of this project. Given these limitations, I feel the project wasn't able to reach its full potential, as I under-estimated the time it would've taken to implement what I wanted, such as a transfer news, and more ways to visualise information and data, like a parallax. I also would have liked to add stats on the leagues and countries and add more intractability.

As I mentioned previously, along with not knowing the tech-stack in which the project was developed in and learning it from scratch, there were undoubtedly other challenges that I faced in the creation of this project. One of the most significant difficulties was working on this project by my own. As this was an individual project, I had to take on the role of a designer, frontend and backend engineer, QA lead, tester, coming up with useful features, what to exclude, where to find reliable API's, along with being my own teacher when things weren't going so smooth. Everything required a lot of energy and focus, so much so that it simulated having a full-time job. However, it was worth it, as I learned so much over this period about a wide range of topics. It also gave me a career head-start, as I want to be a frontend developer post college, so it's beneficial that I have exposure to a lot of what is worked on in the industry, as per my internship back in 3rd year.

Overall, my website based on football statistics is a good starting point in learning and building on the tech-stack that I've learned. Although it has its strengths and limitations, the website's strengths lie in its user interface, its use of API's to gather information, and its advanced features like the comparison feature, infographics, and the search bar. However, the project's limitations stem from its reliance on API's and its narrow focus on football statistics. If given more resources and time, I could've added more features to the website and addressed some of the limitations.

4.0 Further Development or Research

Assuming I had more time and resources to develop this project further, I have a plethora of extra features that I'd like to have added. These ideas developed over the course of the creation of Football IQ, but considering the time constraints and resources, I had to prioritise. These ideas and extra features include:

- The aforementioned addition of adding statistics to the leagues and countries.
- Expanding to support more leagues.
- Adding a transfer news section (Twitter API).
- Setting up database so that users can star certain players to keep an eye on them.
- Add a parallax or carousel to help visualise information.
- Machine learning and predictive analytics: With more time and resources, the project could incorporate machine learning algorithms and predictive analytics to provide users with more advanced insights. For example, the website could use machine learning algorithms to predict the outcome of future matches based on historical data.

5.0 References

N/a

6.0 Appendices

6.1. Project Proposal

Objectives

My project is going to be a website called 'Football IQ', where it takes live information from a variety of application programming interfaces (API's) and stores that data on a site built in the form of React, JavaScript, Node.JS, Next.JS, and other languages. To my knowledge, there are no football statistic websites that are built in this way, so I hope to add to the industry by making my website available to the market, showing similar data to what the users are used to, but displayed and stored in a unique way. Some additional features that I plan on adding to the site to make it more user friendly include fast loading times, user-friendliness being site-wide, easy on the eye and intuitive user interface and design, being able to share any stat on social media (Call to action), a smart filter search bar, bespoke logo and motto, hamburger bar, website footer, about page, login page for different accounts, infographics to accompany stats, description of what the site does, responsive web design, and the potential use of a parallax and carousel as an alternate/additional method to display data.

My objectives for this project are for that data I'm using, to be easily accessible, along with a burdenless design. I plan on using a hamburger bar containing tabs, settings, a means to switch accounts, and other relevant settings. In addition to this, I want to implement a smart filtering search bar to assist in the ease of use. I want all of this will be accompanied with fast loading speeds, and due to the amount of information that will be available, this is going to be crucial if I want the site wants to see any traffic or be used for a prolonged period of time. Quick, dependable, and burdenless is the aim, and using the tech stack that I mentioned earlier to the best of my ability will be key if I want to complete these objectives.

Background

As football and statistics fascinate me, I realised that it would be an excellent idea to combine the two and considering that I learned the basics of JavaScript and React during my internship, I also came to the conclusion that implementing this technology would be a smart, but also challenging way to tackle such a project, while also massively improving my skills, as this is the technology that I want to use going forward in my career. Moreover, when researching other football statistic websites, I noticed no implementation of React, along with subpar design and information presentation. I plan to become accustomed to the syntax and how they're used by frequently working on the project and using reliable resources, such as YouTube and W3Schools, while also doing further research into API's and how to implement them into a project, and how best display information on a website to maximise the user experience, by taking inspiration from a variety of like-minded websites and opposite ones, to get the best layout ideas. In practise, if any features become begin to take up too much time, I aim to adjust and adapt to the situation by removing these features, and hopefully improving on already existing features.

I also plan to purchase a unique domain name and make the site live after the project is finished. This is the overall end goal for me, and it would be a really good to have on my GitHub repo list, especially as everything that I hope to touch on were used heavily during my internship, and therefore would be a great learning experience and project to have for the future.

State-of-the-Art

There are a variety of applications that already exist surrounding what I'm planning to develop, such as fbref.com and SoccerStats.com which all do something similar. However, I have noticed that the designs and UI's of such applications are subpar, and could be improved upon, while also making the user experience much more intuitive. Furthermore, I will be implementing languages such as React, Node, and Next.JS, technologies, to my knowledge, aren't implemented in these applications, so I hope to make my application easier to update, and reuse certain components, which will make development more streamline and while also helping with fetching information and improving loading speeds.

Technical Approach

There are multiple directions which I could go with the development of my application, such as test-driven development (TDD), and behaviour-driven development (BDD), the former which I plan on using, as to avoid any major bugs down the line, squash them during development, and to simplify my code, which also improves the quality. I believe that the ideal method for me for progressing successfully during this project is by using note-taking software, breaking down tasks into manageable chunks, giving myself deadlines and a quota of items to finish per week, and by having larger goals by using a Kanban board to visualise my workflow, work-in-progress, and to maximise efficiency. These are methods of work that I implemented during my work placement in third year with Travelport, and it work out really well for my style of working and learning, while also being a useful skill (Agile workflow) to have been developed further, as it's commonplace in the technology industry.

I also plan to make a GitHub repository, make pull requests, make branches off the master branch, and push code to have a log of everything I have completed and when I completed it. This is for a few reasons, with the main points being to gain further practice in using an industry-standard software, to get me into the development mindset and habits for future developments, and to assist me in my monthly reports to get a more detailed breakdown in what I have done over the months, as with a lot of modules and other college work to do, any milestones and activities that got completed can get blurred or forgotten about.

Technical Details

The main languages I'm choosing to development this application include HTML, CSS, JavaScript, React Library, Node.JS, Next.JS, and PostgreSQL, which are the frontend languages, backend languages based on JavaScript, and the database. My reasonings for using React and Node, even though I have to become more familiar with them, are because with React, it becomes easier to use once you get better with JavaScript, so adoption shouldn't be too much of a struggle, and it can reuse components, making development more streamline making maintenance a breeze, with Node.js offering better scalability when compared to any other JavaScript server language, while also being seen as a 'full-stack' JS framework, so it can be used for client-side and server-side applications.

Furthermore, front-end developers aren't required to know any algorithms other than basic loops and logic, but as I feel I will be working with big data structures during this project, I feel that implementing necessary algorithms, such as the 'Big O Notation' to help with loading times, will help with my critical thinking, problem-solving, improve the user experience within the app, and help with general career progression. Moreover, the approach that I am going to take will be developing front end components first using JavaScript and React and then implementing the backend work on top of that. Furthermore, since my knowledge is limited on React, Next, and Node, I will have to do tutorials to get up to speed and use important resources to match the goals and features I want to implement.

Special Resources Required

Having access to the correct information for my project will be key, so I must choose the appropriate API's and location from where they come from, with RapidAPI, a free and reliable source for API's being a top contender. Additionally, using the facilities from NCI, such as the library, and asking lecturers who have thought me technologies and modules in past years for exams, and other continuous assessments will also prove to be useful, as there is a plethora of information and reading material to be learned and implemented which will be relevant for my project.

Project Plan

The first steps I must take before starting this project, is to make sure I have an overall idea of everything I want to implement throughout the entire project, and then break it down into manageable pieces, week by week. This includes the features I want to include (depending on complexity of feature, I'll split it up into smaller pieces), tutorials I need to take (depending on the feature I'm doing at that time, the tutorial work will be based off that), and any other documentation that must be done, such as the monthly reports. Below is a rough draft of deadlines I gave myself and imposed by the college in the form of due dates:

9/10/22 – submit project pitch video.

30/10/22 – submit project proposal.

28/11/22 – implement one API into project.

12/13/22 – finish midpoint project documentation.

15/12/22 – improve prototype by adding more (skeletonised) features.

20/12/22 – submit midpoint documentation, presentation, and prototype.

28/12/22 – make more wireframes to use to develop more pages and features.

3/1/23 – continue developing my application.

4/1/23 – submit my final 1st semester monthly reflective journal.

15/1/23 – continue working on my project documentation.

1/2/23 – more work on my project.

1/3/23 – think about purchasing a domain name.

Initially, once my pitch and proposal and pitch has been accepted, I am going to set up a GitHub repo containing the necessary files for my project. This because it's good to have a location with all of the work I have completed that can help me get a job, and it's also good as a recovery location in case my Dropbox or laptop cease to function. After this, I'll set up my project by making a simple website and building upon that.

I plan to make wireframes that will contain the overall look, layout, and feel of that the final design is going to look like, although I'd assume as the more I develop, the more these wireframes are going to change. I plan to use Figma for the wireframing software, which is where mock-ups will be made of the user-interface, such as button placements, menu locations, and much more. I plan to do one page at a time, as to not get lost in the preparation side of development, while making pull requests and working off the master branch on my GitHub repository for every new feature I plan to implement.

In addition to this, instead of using test-driven development, I will test as I go, using the Jest framework for JavaScript. Since I have only touched on this during my internship, I will need to go back over it using tutorials from the previously mentioned sources. I will make tests for all major features that I add, such as login form, and the smart filter search bar. This approach is better for my needs, as since my knowledge of Jest is limited, doing TDD goes against the purpose of doing it, and at least this way, I learn all the languages and I get to develop at the same time, so taking into account the time constraint that I'm under, it makes sense to me to do it this way.

After I have a design I'm comfortable with for the first page, the development will take place, while learning what to do as I do it, using tutorials on JavaScript, React, and Node.js. I feel this will get the best out of me time and energy wise, as the information that I learn will be fresh in my mind ready to be used in my website. After the initial page is developed, I aim to link an API and a database together that can use the information, store it on my database, and present it on the page that I developed. Once the API and database are in working order, I plan to add all remaining functionality to the page and finish off the frontend work, such as matching the data from the API to what is in the Figma design. Once this is complete, I aim to follow the same pattern of development for the other pages until the project is complete.

Once the implementation of all the features is complete, I will go through a variety of QA tests to make sure that there are as few bugs as possible, while also making sure that the testing that I have done is correct and every feature is in working order. This is paramount, as a good user-experience is the backbone of this project. Furthermore, if I notice that a feature doesn't work integrate correctly with the website, or just isn't necessary, I will have no hesitation in removing it, as having unused or flawed code can be a security risk.

Finally, after the website and all documentation are complete, I plan to deploy the website on a server and purchase a domain name to have it publicly accessible. This has always been the end goal, as I want my work to be seen by as many eyes as possible and have a use in people's lives, instead of something that I forget about after I'm finished with it. I want to keep updating it in the future and make it the possible service it can be.

Testing

For the testing phase, I plan to use the aforementioned Jest framework based on JavaScript, to confirm that all components and features can work together without problems and that they fulfil the requirements set by me. In addition to this software testing, I will be sharing the site with my family and friends to get their feedback on the interface, how intuitive the

site feels, and how useful the features are. The purpose for these tests is to help identify system and interface errors, so when the site becomes live, it is in its most polished state. In addition to these testing approaches, I will be doing my own tests, as I will be the most familiar with how it's going to work. These tests will include vigorous testing in both the alpha and beta stage, throughout each software version (every addition of new components to make sure new work doesn't break any old work.), and to test every use-case possible in the site, and to make sure the site works and adapts to the behaviours of a user, such as using the website in multiscreen. Once the performance and feedback match my personal requirements, that's when the testing is complete, and the website is completed.

6.2. Ethics Approval Application (only if required)

N/a

6.3. Reflective Journals

October - This month involved me coming up with an idea for my project pitch. I plan to make a website that contains data on Europe's top 5 football leagues and players, while making it well presented, having a user-friendly interface and smart, useful features. Furthermore, I also completed my project proposal, which went through my project and every step that I plan to take throughout its development. This includes my objectives, my approach, and the technology stack I plan to use. Finally, I had to fill out the 'Ethics Declaration form', which outlined how I have permission to use the data source, and if I was using human subjects and if I required their data.

In addition, I have received feedback regarding my project pitch. It has been 'Accepted with Amendments', with the following feedback: "Using the data for other purposes is interesting, that's what you have to focus on and make it a visual piece. This is an interesting idea but one which is quite similar to one of last year's projects. It will be important to explore what will be new about this project and to ensure that there is sufficient technical difficulty to achieve the necessary grade."

November - This month, I managed to get a start on my final year project, titled 'Football IQ'. Initially, I started off with doing tutorials for my chosen technologies, but this proved to be ineffective and used up a lot of time, that could have been spent doing the actual project, so instead I've decided to only use tutorials when I really need it, as to avoid learning anything unnecessary or unrelated to my project. This has proven to be effective, as I feel like I got more work done this way.

The first bit of actual work I got done was making a Figma wireframe for my Login screen that I plan on having, which can be found below. I chose to use Figma because that's what I interacted most during my internship at Travelport. Making wireframes before starting the design is a great way to avoid wasting time and to know exactly what you plan on developing before you develop it, and it makes development much more enjoyable, because there are no blockers or design inconsistencies as you develop.

Welcome
Hey! Enter your login details below

Email

Password

Sign in

Don't have an account?
Create one now!

After I developed this part of the app, I wanted to incorporate Node.js, the server I planned on using. I managed to get it set up and working, which was great, but after I tried to get my project set up on Next.js, a framework which handles the tooling and configuration needed for React, and provides additional structure, features, and optimizations for your application. I decided that I was wanted the date to be pre-rendered on the server-side, as opposed to it being rendered on the client-side, which meant that Next.js already handled the server, so I didn't need to use Node in addition to Next. This was unfortunate, as I spent a lot of time configuring Node, and it turned out that I didn't even need it.

After this was sorted out, I decided to work on one of the API's I was going to use. This took more time than I expected, and I thought it'd be relatively straight forward, but then I learned that I needed a secret key to have access it, and then there were other teething issues regarding my knowledge of React in order to hit the API, and have it displayed on the page. This took a lot or research and trial and error, but I feel I can do it with less trouble in the future.

December – This month I was working towards finishing my mid-point presentation, implementing code, and features, finishing up the documentation, and preparing slides and making a video demonstrating what I have done so far. Below are screen shots of what I have done so far, which shows that my API's work and they are fetching the correct data and displaying it appropriately. This includes using the API to get a club name, crest, and squad information, along with the names of the Europe's top five leagues and the biggest European competition. This is also styles using styled-components which are native to React.

Football IQ



- Name: Kepa
Number: 1
Age: 28
Position: Goalkeeper
Club: Chelsea
- Name: G. Stonina
Number: null
Age: 18
Position: Goalkeeper
Club: Chelsea
- Name: R. James
Number: 24
Age: 23
Position: Defender
Club: Chelsea
- Name: W. Fofana
Number: 33
Age: 22
Position: Defender
Club: Chelsea
- Name: M. Bettinelli
Number: 13
Age: 30
Position: Goalkeeper
Club: Chelsea
- Name: Thiago Silva
Number: 6
Age: 38
Position: Defender
Club: Chelsea
- Name: K. Koulibaly
Number: 26
Age: 31
Position: Defender
Club: Chelsea
- Name: B. Humphreys
Number: 42
Age: 19
Position: Defender
Club: Chelsea
- Name: É. Mendy
Number: 16
Age: 30
Position: Goalkeeper
Club: Chelsea
- Name: T. Chalobah
Number: 14
Age: 23
Position: Defender
Club: Chelsea
- Name: César Azpilicueta
Number: 28
Age: 33
Position: Defender
Club: Chelsea
- Name: J. Brooking
Number: null
Age: 20
Position: Defender
Club: Chelsea
- Name: E. Beach
Number: null
Age: 19
Position: Goalkeeper
Club: Chelsea
- Name: B. Chilwell
Number: 21
Age: 26
Position: Defender
Club: Chelsea
- Name: Marc Cucurella
Number: 32
Age: 24
Position: Defender
Club: Chelsea
- Name: A. Gilchrist
Number: null
Age: 19
Position: Defender
Club: Chelsea

Leagues



Serie A



Premier League



La Liga



Bundesliga



UEFA Champions League



Below shows code snippets and how I implemented the API's and how I styled the data shown on screen:

```
79
80 <LeagueDiv>
81
82 <LeagueTextDiv>
83 <h1 style={{ borderBottom: '4px solid black' }}>
84   Leagues
85 </h1>
86 </LeagueTextDiv>
87
88 <LeaguesList>
89
90 {leagues.map((league: league) => {
91   const leaguesToReturn = [39, 78, 61, 135, 140, 2];
92
93   if (!leaguesToReturn.includes(league.league.id)) return;
94
95   return (
96     <League key={league.league.id}>
97       <LeagueStatsList>
98         <LeagueNames>
99           {`${league.league.name}`}
100         <img src={league.league.logo} /> Do not use <img> element. Use <Image /> from 'next/image' instead. See: https://nextjs.org/docs/messages/no-img-element
101         </LeagueNames>
102         {`/* Logo: ${league.league.logo} */</li> */`}
103       </LeagueStatsList>
104     </League>
105   );
106 });
107 </LeaguesList>
108
109 </LeagueDiv>
```

```

22  async function getLeagues(): Promise<AxiosResponse<any>> {
23      const response: AxiosResponse<any> = await axios.get(
24          `https://api-football-v1.p.rapidapi.com/v3/leagues`,
25          {
26              headers: {
27                  'X-RapidAPI-Key': '3e93f54308mshcc56d624809a4a9p144a30jsn829d33d2f0e4',    "mshcc": Unknown word.
28                  'X-RapidAPI-Host': 'api-football-v1.p.rapidapi.com'    "rapidapi": Unknown word.
29              }
30          }
31      );
32
33      return response;
34  }
35
36  async function getLeaguesByLeagueId(id: string): Promise<AxiosResponse<any>> {
37      const response: AxiosResponse<any> = await axios.get(
38          `https://api-football-v1.p.rapidapi.com/v3/leagues`,
39          {
40              params: { league: id },
41              headers: {
42                  'X-RapidAPI-Key': '3e93f54308mshcc56d624809a4a9p144a30jsn829d33d2f0e4',    "mshcc": Unknown word.
43                  'X-RapidAPI-Host': 'api-football-v1.p.rapidapi.com'    "rapidapi": Unknown word.
44              }
45          }
46      );
47
48      return response;
49  }
50  export const leaguesApi = {getLeagues, getLeaguesByLeagueId};

```

```

138  export async function getServerSideProps(context: any) {
139      const { data } = await playersApi.getPlayersBySquadId("49");
140      const leagues = await leaguesApi.getLeagues();
141      console.log(data.response[0])
142      return {
143          props: {
144              players: data.response[0].players,
145              team: data.response[0].team,
146              leagues: leagues.data.response
147          },
148      }
149  }

```

```

65 export const LeaguesList = styled.ul`
66   color: white;
67   width: 100%;
68   display: grid;
69   justify-content: center;
70   flex-direction: row;
71   grid-template-columns: 1fr 1fr 1fr;
72   `;
73
74 export const League = styled.li`
75   margin-bottom: 20px;
76   `;
77
78 export const LeagueStatsList = styled.ul`
79   display: grid;
80   grid-template-columns: 1fr 1fr 1fr;
81   color: black;
82   `;
83
84 export const LeagueDiv = styled.div`
85   margin-top: 5%;
86   `;
87
88 export const LeagueTextDiv = styled.div`
89   display: flex;
90   justify-content: center;
91   align-items: center;
92   margin: 0 auto;
93   flex-direction: column;
94   `;
95
96
97 export const LeagueNames = styled.div`
98   display: flex;
99   justify-content: center;
100  align-items: center;
101  margin: 0 auto;
102  flex-direction: column;
103  `;
104
105

```

January – This month, I added more features to the site, such as incorporating an API in which the user can click to see what league they want to access, which then allows them to choose the clubs stats, etc. in addition to this, I have also been doing more research and learning about my chosen technologies, such as React and Typescript, along with analysing how I can improve the User Interface and improve this so that my features and information is easily accessible and not hindering the user’s experience in any way. To do this, I’ve been viewing and using many other well-designed websites and mimicking their layouts to use in my site.

February – This month I implemented a dropdown menu that gives the user the ability to select their desired league, team, and player to gather stats on. I have also made a wireframe for the login

screen that shows a message if the details the user enters is incorrect. The wireframe can be found below:

The wireframe shows a login screen with the following elements:

- Welcome** (Large heading)
- Hey! Enter your login details below (Text)
- Email (Input field)
- Password (Input field)
- Sign in (Orange button)
- Oops, that's incorrect. Try again! (Red error message)
- Don't have an account? Create one now! (Text with bolded link)


I am currently developing a menu/hamburger bar that will house all of the settings and options, such as a dark mode toggle and account settings. This will be in the top left corner of the screen. In addition to this, I want to start the marquee player comparison feature this month.

March – This month, I developed a menu/hamburger bar that contains all of the settings and options, such as a dark mode toggle and account settings. This is at the top left corner of the screen. This is making the site come together, as the user can actually interact with the application, see statistics of their own choosing, and change their preference.

Last month I said I wanted to get a start on the marquee feature, which is a player comparison feature, but I was not able to get a start on this, as the hamburger bar took more time than anticipated, and I also had to accomplish other modules, projects, and CA's in the meantime. This is currently the goal for the next month along with starting the transfer news section, in which I will be using an API for the information.

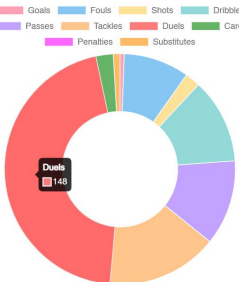
April – I've made a lot of progress this month. For starters, after I completed the hamburger bar, I added infographics to the player stats, in which everything was packaged in the form of a modal, so when a player is clicked on, a modal appears showing an interactive chart, that can be narrowed down to show individual stats by hovering over a specific part of the chart, accompanied with additional stats below. This was also done with each individual club too, minus the piechart. Screenshots of this can be found below:

R. James




Age: 24
Nationality: England
Team: Chelsea
Height: 182 cm
Weight: 82 kg
Position: Defender

Key Statistics (All Comps)



Statistics:

Goals Fouls Shots Dribbles
Passes Tackles Duets Cards
Penalties Substitutes



Key Statistics:

Team: Chelsea

Country: England

League: Premier League

Current Form: WDLWLWWWDDLLLDLWDDLLWDLDLLL

Games Played: 32

Total Wins: 10

Total Losses: 13

I also made an about page containing information on what the website offers. A screenshot of this can be found below:



Football IQ

About Us

Welcome to Football IQ, a website dedicated to providing the most comprehensive and up-to-date information on all aspects of the beautiful game. Our site is designed to offer football fans and enthusiasts the opportunity to dive deep into the world of football statistics and to explore the many fascinating stories that are hidden within the numbers.

Our site is built on a foundation of simplicity and user-friendliness, with a focus on providing a seamless user experience that is both informative and enjoyable. We understand that there are many different types of football fans out there, and we aim to cater to all of them with our wide range of features and tools.

At the heart of our site is our search bar, which allows you to easily find the information you are looking for, such as team or player statistics. Our dropdown menu makes it easy to navigate our site and find the information you need quickly and easily.

We also offer a range of customization options, including a sleek and stylish dark

In addition to the dropdown menu, I have incorporated a search bar that searches for any team in the leagues provided. An example of this can be shown below:



Football IQ

- Premier League
- La Liga
- Bundesliga
- Serie A
- Ligue 1
- UCL

In the dropdown menu above, I incorporated a search bar, and when a user enters a character, the search bar filters the teams containing those characters, shown below:



Football IQ

- Manchester City
- Manchester United
- Chelsea
- Elche
- Borussia Monchengladbach

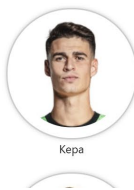
When I type in the characters 'che', the options that contain those letters appear for the user to click on. Once clicked, the user will be brought to the squad to interact with the team further, as seen below:



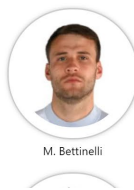
Football IQ



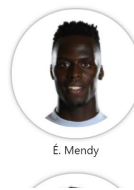
Chelsea



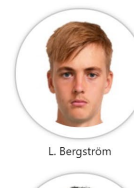
Kepa



M. Bettinelli



É. Mendy



L. Bergström

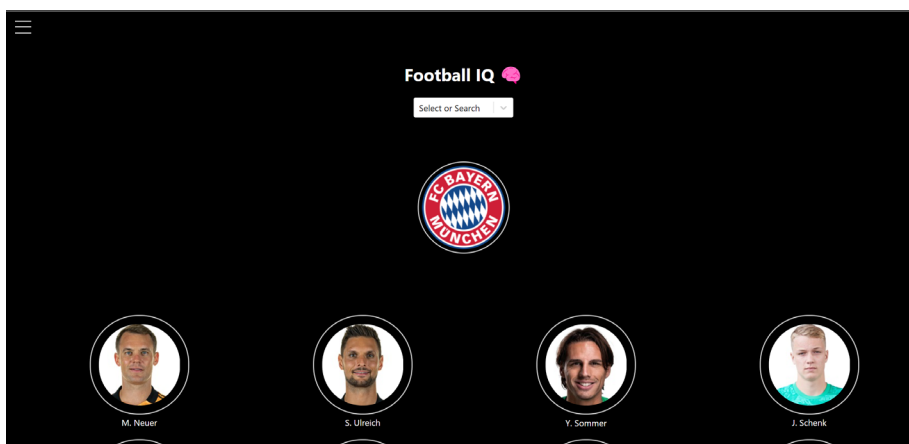
Furthermore, the implementation of the login screen and database has been finalised, where a user can make an account, and be brought to the main page. My plans for this feature include being able to save teams and specific players to keep an eye on their stats for future reference. The account details such as the username and password are hashed and stored in a Postgres database. They are shown below:

Login

Don't have an account? [Sign up here!](#)

	id [PK] integer	email character varying (50)	password character varying (255)	name text	role text
59	73	TEST1	\$2a\$10\$IYosB5sbSaKEBsdFPFajR0ws9P4tfzX.j6DVNhMm0OKK4RqzxSqdi	[null]	[null]
60	74	TEST2	\$2a\$10\$GiGGWIZLTXDe5mkMpHkbuugx.ZCamyg6xBRZlj4IP6e3C24wvAeu	[null]	[null]
61	75	TEST3	\$2a\$10\$LszbrrCgV6RoweVDkK8I9e8gOq7XrAZAWtVe2J/fgzv5yV3MtmJGe	[null]	[null]
62	76	TEST4	\$2a\$10\$f1Gt7m4JmfeghxJqgbNbK0c4XLAgUrBxmh5uiqOiYHzNy8qpHy2...	[null]	[null]
63	77	TEST5	\$2a\$10\$EkZjMYU7B9iczXgwD5Igw.1BZRxpVq2LPID32sfKlxfrEOxCVNXv6	[null]	[null]
64	78	TEST6	\$2a\$10\$ZnQ7YMBKMI3aB/5bxVrLse5mfZBY6Iss.TV3mSZNSdqHJdyeuQ...	[null]	[null]
65	79	TEST7	\$2a\$10\$rV7cCnmpdji/s60sxN51ukVku6xvJI4AS7DRsisRDvwwA.Kjhkva	[null]	[null]
66	80	TEST8	\$2a\$10\$8hNQKLPgwtFzlx5OxbDG4.gGKBupBB/bMrBU3uezJ08WZekilxWBi	[null]	[null]
67	81	TEST9	\$2a\$10\$LF3McSJeofEhpM0uh.X7Yed8Agr1x6OSTG.0Q21J/Yq0JLo5SrUfs	[null]	[null]
68	82	TEST10	\$2a\$10\$mXf40wPgUYIHeCM6fDVvn.25S2lvKULF05Vi2NBVahSLpSVFnIZ...	[null]	[null]
69	83	TEST11	\$2a\$10\$rxPGMdAWYj9P227oEyRbheoS.en18J9Xxp94s5oOCGTelr3C2fSI6	[null]	[null]
70	84	TEST12	\$2a\$10\$S2ccHdv7LJ0.4QbDAes4XetxkzjfUSTC4VvkCEIkWjYepTUsA0ayq	[null]	[null]
71	85	TEST13	\$2a\$10\$2WYuHYuS41YXzYp5WO130044dIA6j8UpV.LgmRFGM.cmh91kxjX.	[null]	[null]
72	86	TEST14	\$2a\$10\$GdJRGaxq4cbD8OUkUZWTlun2ZKIPYPXoEzUWw18LFi2uSwbwX...	[null]	[null]

Other changes include general styling fixes and adjustments such as to the dark mode implementation, this is what it looks like site-wide:



From this point on, I am to do a few more things. Implementing the main feature being the comparison feature will take up most of my time, introduce CTA'S, the aforementioned saving players and clubs to array, and a transfer news section which gives news on transfers, along with other visual and non-visual tasks, finalising the UI and fixing bugs and testing.

6.4. Other materials used

N/a