2023

IFN666 Web APP – Capstone Project

IFN666

Web APP – Capstone Project

Link to demo video:

https://youtu.be/utwx-dWoBo0

Li-Chen (Jeannie) Huang

10883789

4/6/2023

Contents

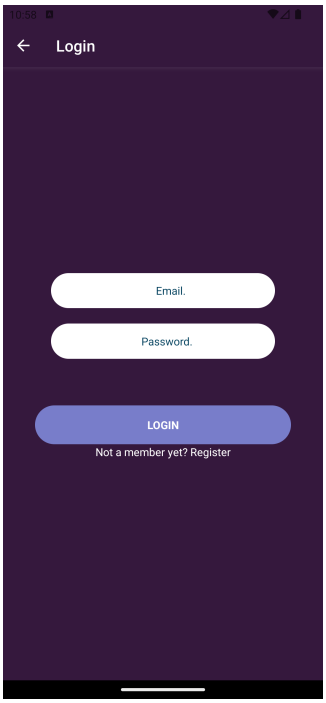
Introduction....2
Purpose & description....2
Application Completeness....3
Technical Description....3
Use of API End Points....4
1. Everything endpoint (News API) (Get)....4
2. Standings by league id endpoint (API-Football) (Get)....4
3. Teams Information (API-Football) (Get)....4
4. Register Api (Self-constructed) (Post)....4
5. Login Api (Self-constructed) (Get)....4
6. Update Api & Delete Api(Self-constructed (Post)....4
7. GetFavTeam Api(Self-constructed) (Get)....4
Dataflow and Screen flow....4
1. Register....5
2. Login....5
3. Favourite Team....5
Directory Architecture....5
Testing and limitations....6
Test cases....6
Limitations....8
Discussion....8
Appendix....9
Appendix A - User's Guide....9
Appendix B – Test cases figures....10
Appendix C – Self-checking against CRA....12

Introduction

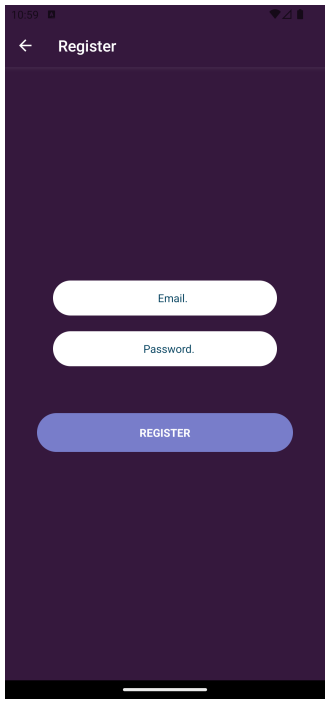
Purpose & description

The app constructed in this assignment provides user with the informations of the Premier League football. The purpose of this app is to present data related to the league itself and each teams to the user, along with corresponding news and their links. Notably, it also applies a backend server that handles data for user's registration, login, and favourite teams. It presents a landing screen, a news screen, a login screen, a registration screen, an overall team standings page, and a user-customised favourite team screen that leads to individual team's visualised statistics with of the 2022 season.

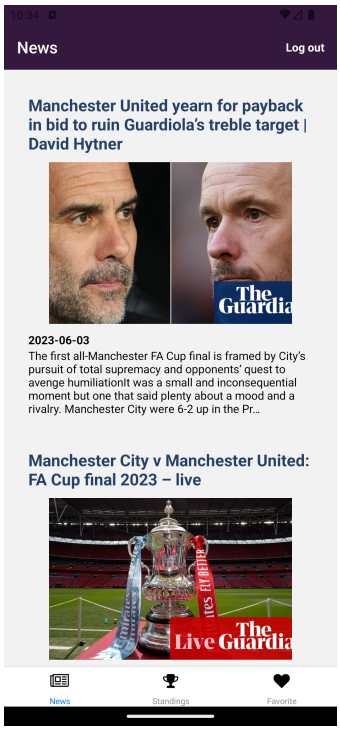
Landing screen



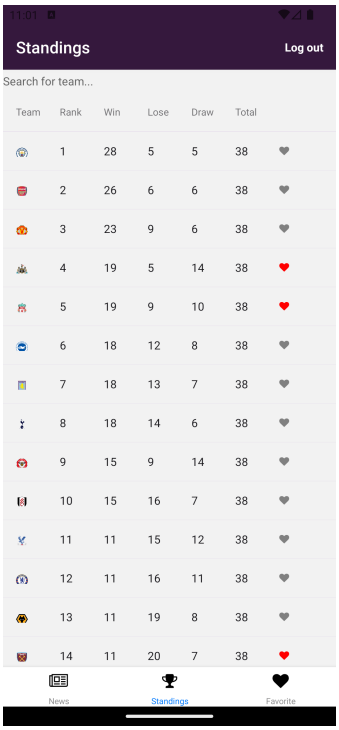
Login Screen



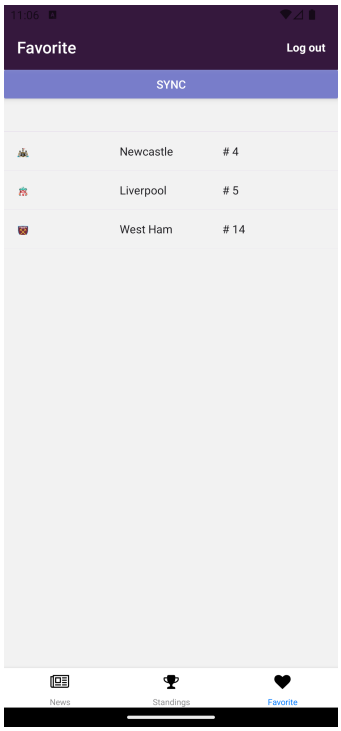
Register Screen



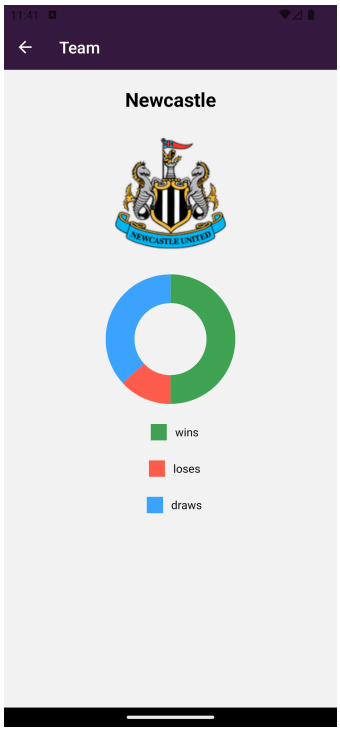
News screen



Standing Screen



Favourite Screen



Team Screen

Application Completeness

This application successfully displays:

1. An aesthetic landing screen
2. A register screen which allows user to register their email and password. Email format examined.
3. A login screen which only allows user to sign in with the correct email, email format and password
4. A news screen with images, original links, and description that showcases latest league news
5. A real-time team standing screen that displays current statistics and allows the user to search for team names and add teams to their favourite list
6. A favourite team screen which displays the user's favourite teams
7. Multiple team statistic screen, which utilises data visualisation to display match statistics
8. A backend database that stores require data

I believe it is reasonable that I claim a **[Grade of 6 and 7 level]** since it suffices all of the requirements in the specification, including both the client side and the server side.

Technical Description

The APIs used for this application are:

1. API-Football from Rapid Api (https://rapidapi.com/api-sports/api/api-football)
2. News Api (https://newsapi.org)
3. Self-constructed Apis: Register Api, Login Api, Update Api, Delete Api, GetFavTeam api

Use of API End Points

This section briefly lays out the API endpoint this application implements and demonstrates the page that this endpoint supports.

1. Everything endpoint (News API) (Get)

The everything endpoint of the news api returns the data of every news it holds information to, and it takes in parameters to filter these data. Here we apply the parameter of date starting from yesterday of the time querying, and the query key word be "Premier+League" to have it return the newest correspondence related to the Premier League. The data we utilised from this endpoint is the image url of the news, the headline, the link to the news website, the description, and the date.

2. Standings by league id endpoint (API-Football) (Get)

This endpoint provides with the real-time standings statistics given a league. We retrieve the standing of the Premier League along with each team's total, won, lost, and drawn game number to demonstrate it in a table.

3. Teams Information (API-Football) (Get)

This application also retrieves data from the team information endpoint to support the data from statistics page.

4. Register Api (Self-constructed) (Post)

Allows user to register for a new account with email and password. It verifies the email format, checks whether the email already exists in the database, and add the new data into the database if passed.

5. Login Api (Self-constructed) (Get)

Allows user to login to the application. It verifies the email format, checks whether the email and passwords matches.

6. Update Api & Delete Api(Self-constructed (Post)

Allows user to add and delete teams from their favourite team list in the backend database.

7. GetFavTeam Api(Self-constructed) (Get)

Get a user's favourite team list to support with functions related to favourite teams.

Dataflow and Screen flow

This section briefly describes the data flow of the backend and front end of this application. Below is a screenshot of the MySQL database structure for this application. Notably, the passwords are hashed for security reasons, while the team list contains array of team id that the user have chosen, stored as JSON format in MySQL.

id	email	hash	team_list
1	jeannie123@gmail.com	$2b$10$oZbJ8yotcD7UHDvdFJTwa.wTqKFpOI…	[12, 13, 4, 34, 51]
2	test@gmail.com	$2b$10$XVU52A2gqBkohFRRYFSbg.pdeXQjY…	[12, 13, 15, 4, 50, 66, 34, 42, 47]
3	tst@gmail.com	$2b$10$K.D7dLGm4ey8PDMyD9crYOJknGlp.0…	12
7	123@gmail.com	$2b$10$htgKKUl8T4cGPn/o/TLPbe0Be6TVVW…	NULL
NULL	NULL	NULL	NULL

Screenshot: database structure

1. Register

Once a registration successes, the user's email and hashed password would be stored into the database. With a initial team list value as null.

2. Login

In order to construct a successful login feature, a function that takes in the login api's response message and return a boolean value is constructed. This serves as a gate keeping method to determine whether a login page should be navigated to the home page or not. While successfully login, the email of the user will also be passed in globally to serve for further use.

3. Favourite Team

When a team object is initiated, a property called "favourite" would be set according to the user's team_list column from the backend. This list would then be modified when the update/delete api is called by the client. The favourite screen also utilises a get method that calls the team list from backend and display a user's chosen team.

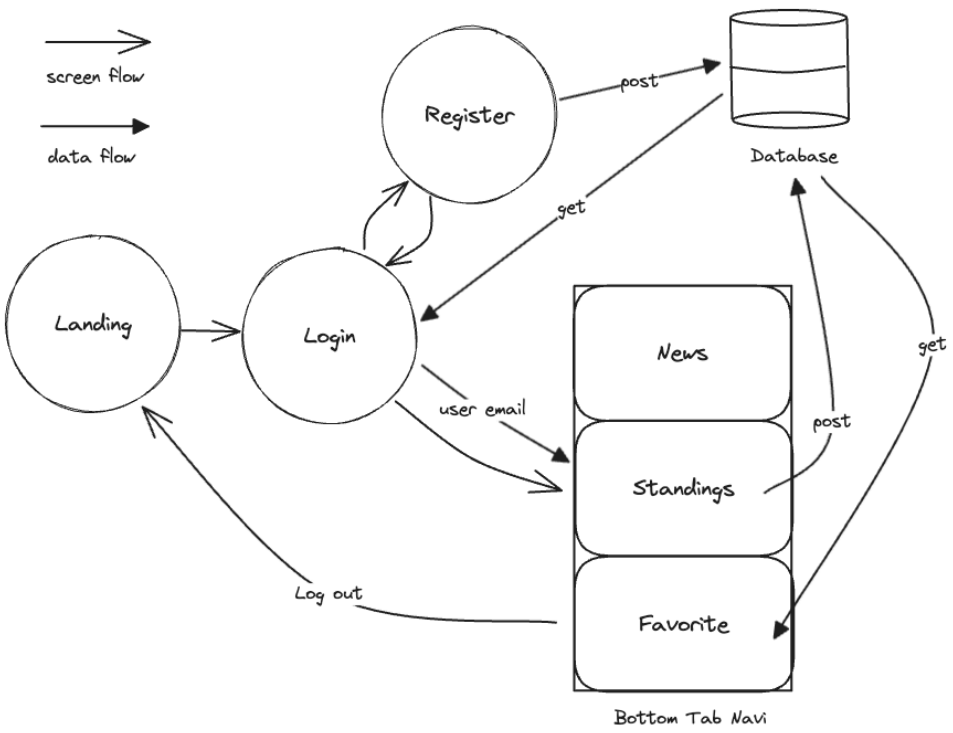
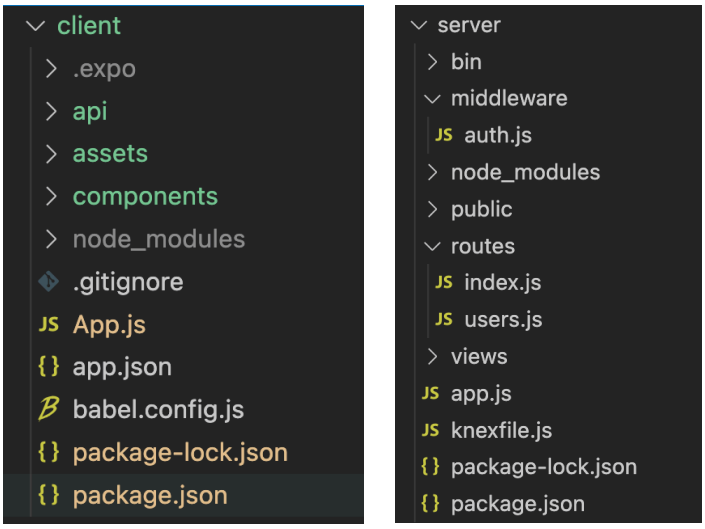


Diagram: Screen and data flow for application

Directory Architecture

This application is divided into two project folders: **client** and **server**. For the client side, there are mainly three folders under src: **api, assets components**. The api folder contains all function that holds the fetch method to retrieve data from different end points. The assets folder is conducted of images used for the banner and the landing screen. The rest of the components are aggregated in the components folder.

For the server side, the routing of self-constructed apis are defined in the router folder. As middleware stores functions for the authentication and CORS, the bin folder accounts for domain settings while the knex file holds for a settings to access the MySQL database.



Screenshot: directory architecture

Testing and limitations

Test cases

Below is the test plan for this application. The screenshots from the final column could be found in appendix 2.

Task	Expected Outcome	Result	Screenshot(s)
Landing screen			
Landing screen correctly shown	Page title be “Be in the game Premier league information click anywhere to start”	PASS	Figure 1
Click anywhere function	Click anywhere on screen should lead to login page	PASS	N/A
Registration screen			
No email or pass word input	Display message: Input incomplete	PASS	Figure 2
Email not in correct format	Display message: Please enter a valid email format	PASS	
User not exist	Display message: User does not exist	PASS	
All data correct	Store user data in backend and navigate to login screen	PASS	
Login screen			
No email or pass word input	Display message: Input incomplete	PASS	Figure 3
Email not in correct format	Display message: Please enter a valid email format	PASS	

User not exist	Display message: User does not exist	PASS	
Password doesn't match user	Display message: Password doesn't match	PASS	
All data correct	Redirect to news screen	PASS	N/A
News screen			
Headline and its link is correctly shown	Showing headlines with link to the original page	PASS	Figure 9
Image is shown	Showing correct image according to different news	PASS	
Date is correct	Date of the news should be no older than yesterday	PASS	
Description is correct	The brief description of the news are correctly displayed	PASS	
Standings screen			
Search bar	When being input, the table filters out the corresponding teams	PASS	Figure 4
Logos, team name, and statistics	Logos, team name, and statistics are correct	PASS	Figure 4
Favourite team displayed correctly	The heart icons are filled according to the user's favourite team list	PASS	Figure 5
Add to/delete from favourite	When clicked, heart icon is filled/ dismissed, and team id added to team list	PASS	Figure 5
Favourite screen			
No data display	If there is no favourite team, the screen shows "no data"	PASS	Figure 6
Team display	Team from the user's favourite team list is displayed with correct information	PASS	Figure 7
Team link	When clicked on, the team row links to its specific team info page	PASS	N/A
Sync button	When clicked on, updates on favourite teams are shown	PASS	N/A
Team Info screen			
Display	Team name, logo, and piechart displays correctly	PASS	Figure 8

Limitations

The limitations of this application is listed as follow:

1. The visualisation of team statistics could be in more detail
2. In the news section, although rare, some unrelated news would be pulled in
3. The descriptions of the news sometimes are unclean

Discussion

This application displays some critical informations for the premiere league football. Although the data is acquired from a authorised API source, there are still some concerns regarding the copyrights that was used in this application. For instance, the team logos and the league logos. Therefore, if in the future this application were to be deployed to a commercial environment, copyrights are undoubtably something to be further discussed.

Appendix

Appendix A - User's Guide

Dear user:

The first thing you see is the landing page, which will prompt you to click anywhere. When clicked on, the login page will show. If you already have an account, just login with your infos. If not, tap on "Not a member yet? Register" to register for a new account. Please make sure that you are using a valid email address here as the system would not let you register if you don't. After registering successfully, the application will redirect you to the login page. From here just use the account that you just created to login.

When you enter the main page, you will see three tabs below: news, standings, and favourites. In the news section, if you tap on the news title/ image, the application will open your browser and redirect you to the original source of the news.

The Standing screen allows you to search for your favourite team and add them to your favourite list by tapping on the heart icon. When a team is added into the list, you can view them from the Favourite screen. In addition, if you want to view a visualised graph of the team stats this season, just tap on the team's row and you'll be lead to the specific team info page, where you'll find a doughnut chart displaying the team's wins, loses, and draws for the season.

Finally, if you are wish to log out, just tap on the logout button at the top right corner in any of the three main screens. This will lead you back to the landing screen, where it all started.

Happy exploring!

Appendix B – Test cases figures

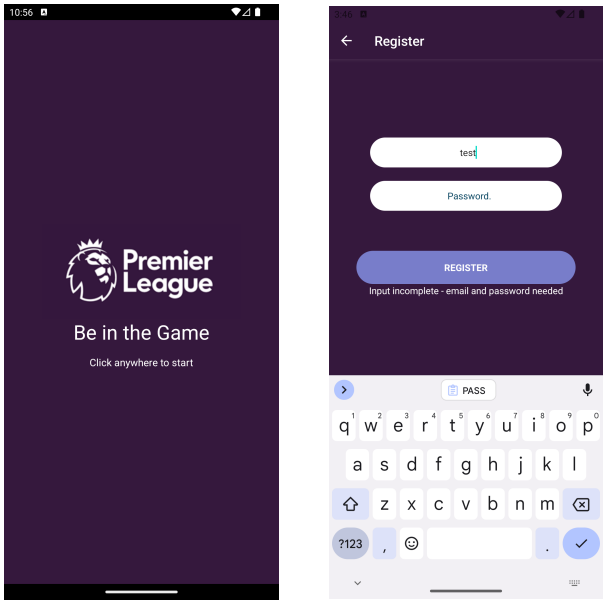


Figure 1

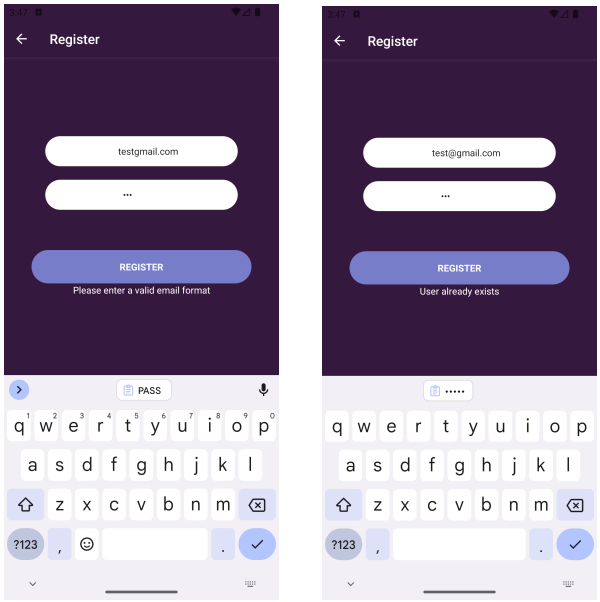


Figure 2

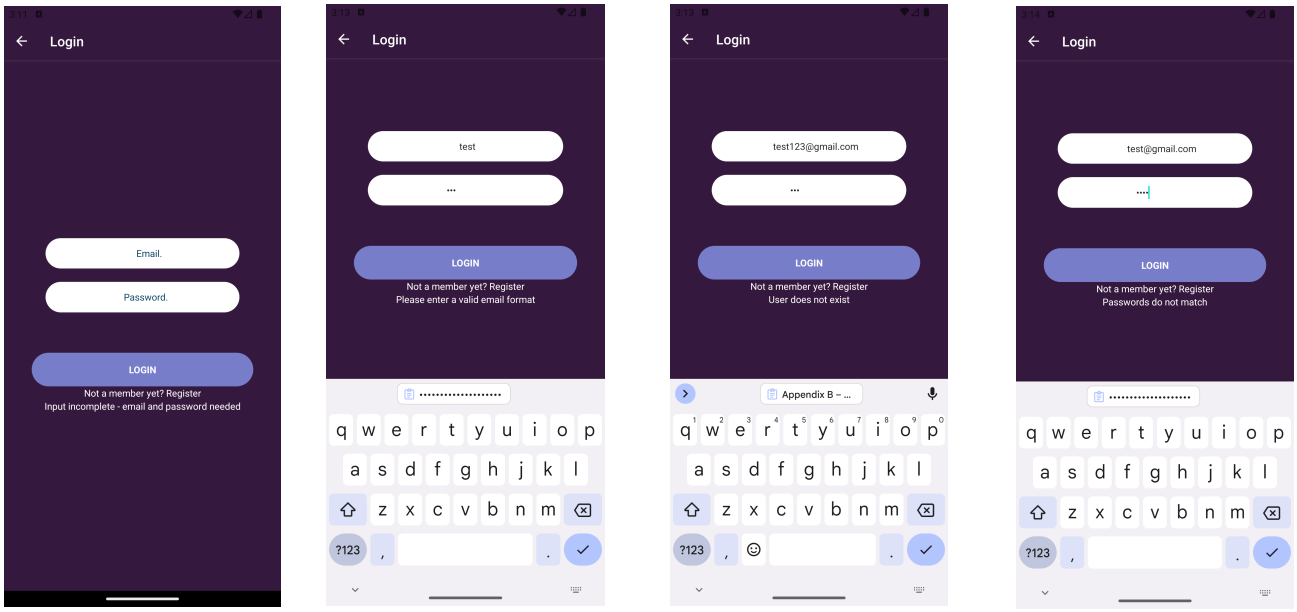


Figure 3

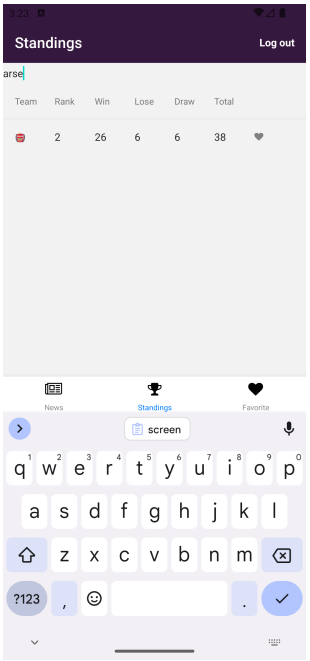


Figure 4

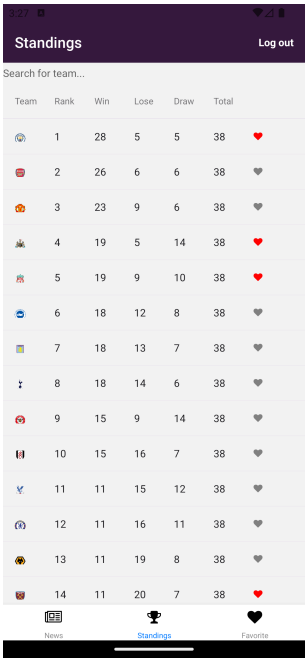


Figure 5

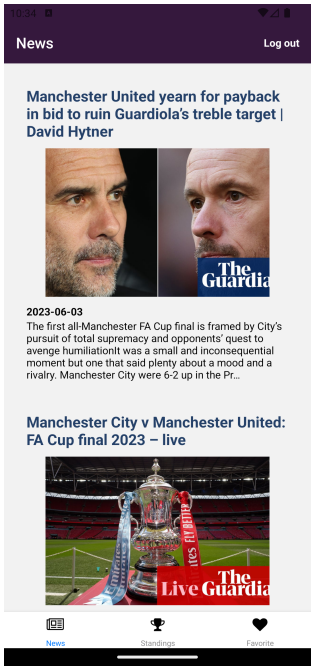


Figure 9

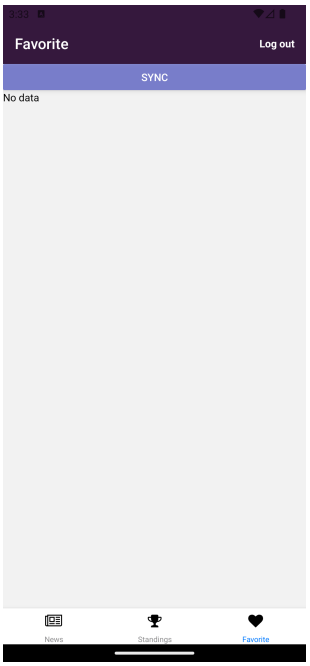


Figure 6

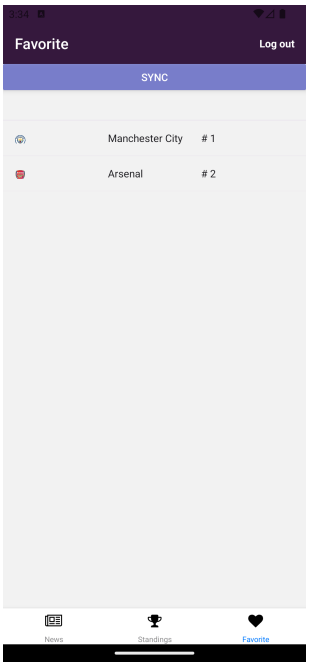


Figure 7

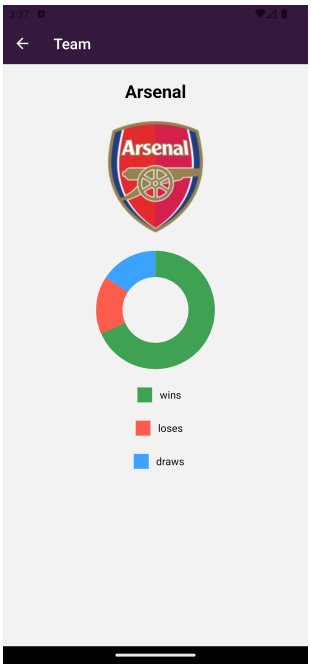


Figure 8

Appendix C – Self-checking against CRA

	7 (85-100) %	**6** (75-84) %	**5** (65-74%)	**4** (50-64)%	**3** (40-49)%	**2** (25-39)%/**1** (<25%)
Front-end Mobile Application Functionality (20 marks)	The client side application implements all of the functionality as listed, and is thoroughly professional in its implementation and performance.	The client side application implements all of the functionality listed as necessary, and is mostly professional in its implementation and performance. Some high level features may have minor issues or limitations.	The client side application implements all of the functionality listed as necessary to a good standard.	The client side application implements the basic functionalities to a good standard, but there may be some minor issues.	The client side application implements some of the functionality listed but the functionalities are poorly implemented.	The application does not work or does not meet the specification or is fundamentally flawed.
Front-end Application Robustness (10 marks)	The application is robust, without any noticeable errors and handles service failures and errors gracefully.	The application is robust and executes without noticeable errors and handles gracefully most service failures.	The application is robust and executes with only occasional errors and handles gracefully most service failures.	The application basically works, but the robustness is questionable and there are obvious errors and service failures are not handled well.	The application basically works, but there are numerous errors and failures.	The application is markedly incomplete and results in numerous errors.

Front-end Application UI Design (10 marks) Note: this is purely based on function and usability. There is no direct assessment of the quality of the graphic design.	The application user interface looks professional, and the principal use cases are a natural fit for the user interface, allowing execution without undue effort or confusion. The user has no trouble navigating the app.	The application looks professional mostly, and the principal use cases are readily executed, though there may be some clumsiness in the workflow. The user generally has little trouble navigating the app.	The application is clean and generally uncluttered, and the principal use cases are reasonably easily executed, though there are some questionable choices in the design. Navigation may not always be straightforward.	The application is somewhat clean and uncluttered, but overall the design is clumsy and doesn't operate very well. Navigation may present some problems. .	The application is poorly laid out, but the basic functionality remains usable. Navigation is difficult.	Layout is poor and usability very weak. The site is barely usable.
40 Marks in total for front-end	38					
Backend functionality, error responses and application reliability. **(20 marks)**	The server successfully implements all of the API endpoints according to specifications [or above and beyond expectations]. [See discussion of 6-7 levels in the spec.] The application is robust and executes without noticeable error and handles service failures and errors gracefully. Error conditions are returned and response codes are perfectly according to specification.	The server supports all of the API endpoints but there are some limitations in the implementation. [See discussion of 6-7 levels in the spec.] The application is robust and executes without noticeable error and handles service failures and errors gracefully. Error conditions are mostly returned in accordance with the API spec, with perhaps occasional incorrect response codes.	The server supports some API endpoints. The application is mostly robust and usually executes without error. Service failures and errors are handled adequately. Error conditions are basically in accordance with the API spec, but there are may be a number of incorrect response codes.	The server supports some API functionality as specified. The application basically works, but the robustness is questionable and there are obvious errors and service failures are not handled well. Error conditions are often not in accordance with the API spec	A basic express app with some functioning routes presented, but the API is markedly incomplete or poorly implemented. The application basically works, but there are numerous errors and failures. The API error responses are not to specification.	The server does not meet the specification or is fundamentally flawed or not implemented at all. The application is markedly incomplete and results in numerous errors.

Back-end Application Architecture and Middleware – DB Connectivity, Logging and Security. **(10 marks)**	The application architecture is professional and uncluttered. The routes are well-organised and handled by appropriate routers. There is good use of middleware for DB connectivity, logging and security. User credentials are handled professionally.	The application architecture is reasonably professional and uncluttered. The routes are well-organised and routers are usually organised appropriately. There is good use of middleware for DB connectivity, logging and security, but there may be some variations in the approach. User credentials are handled correctly.	The application architecture is good but some aspects are awkward, with some routes not handled especially well, and routers not split appropriately. There is some use of middleware, but this is not as clearly thought out as it needs to be. There may be some inappropriate handling of user credentials.	The application architecture is basically sound but the route organisation is very clumsy in parts, and there is too much functionality handled at the application level or in large routers. There is limited use of middleware. Hashing of passwords is poorly implemented.	The application architecture is not well chosen, the routes are not well organised and there is little or no use of middleware. Security is poor. Hashing of passwords is not attempted.	The application architecture is flawed and there is no use of middleware. Security is deeply flawed or not attempted.
30 Marks in total for backend	28					

Development & Process **Code Quality** **(10 marks)** Demonstrate a high level of familiarity with the JavaScript programming language and a near professional level of skill in its application to problems of significant size and complexity	The application shows clear evidence of a professional approach to development, with a coherent modular structure and code quality at a near professional level. For example, program is decomposed into simpler components. Code is clear and simple.	The application shows good evidence of a professional approach to development, with a coherent modular structure and code quality at a near professional level, apart from some minor variations.	The application shows some evidence of a professional approach to development, with a reasonably coherent modular structure and competent, but not always professional level code quality.	The application shows limited evidence of a professional approach to development, with ad hoc or unclear organisation of the application and variable levels of code quality.	Application development has been ad hoc and little more than hacking, with no obvious organisation. Code exhibits numerous defects when compared to the standard expected.	Application development is deeply flawed, with little structure and poor code quality. For example, horrendously complex code, poorly indented with poor choice of identifiers and no comments.
Report and Reflections (10 marks)	The report is thoroughly professional and addresses each of the listed requirements in detail and with only occasional errors of grammar or structure. There are good critical reflections on the development process, UI design and the technical components of the application.	The report is thoroughly professional and addresses each of the listed requirements in detail and with some more frequent errors in grammar or structure.	The report is professional but lacks some detail in a small number of the listed requirements. There may be occasional errors of grammar or structure.	The report is adequate, but the coverage is deficient in a number of the listed requirements. Grammar and structure may be somewhat variable, but are overall ok.	The report is somewhat adequate, but the coverage is deficient in many of the listed requirements. Grammar and structure are of variable quality.	The report is flawed and doesn't meet the requirements. There may be whole sections missing or poorly covered. There is no coherent professional report structure as required
Video demo **(10 marks)**	The video presentation is professional and is able to present all the use cases including some edge cases.	The video presentation is professional and is able to present all the use cases with some small errors.	The video presentation is relatively strong but lacks some rigor and does not present all the use cases.	The video presentation is adequate but does not present all the use cases.	The video presentation is somewhat adequate but does not present the majority of the use cases.	The video presentation is basic and does not present all of the use cases.

30 Marks in total for code quality and documentation	26					
TOTAL (100 Marks)	92					