Final Report

Interactive Multimedia Design Major Project (May 2015)

TTPL

Alastair Coleman

B00464712

PSG: 3 - Leo Galway

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to all those who provided me help and assistance in the completion of this project. Special thanks go to my final year project mentor, Leo Galway, whose contribution in providing suggestions and feedback, helped me to coordinate my project especially in the writing of this report.

Furthermore I would also like to acknowledge with much appreciation the additional staff of University of Ulster, who provided guidance during the process of building the TTPL web application – Peter Nicholl, Jonathan Wallace & George Moore.

Finally, a special thanks goes to the rest of the team at Amigo Studios, Aaron Colton and Andrew McKenna, whom contributed valuable help and support with comments, advice and feedback throughout the design, development and testing of the application.

CONTENTS

1. INTRODUCTION

1.1 Overview

The work undertaken in the development of the project covered the design and development of a web application for a team based football prediction game. This report covers the process throughout, from the refinement and definition of the concept; the design of the application both visually and functionally; and the implementation of these designs including testing with feedback from users.

1.2 Background

Football is a global game. Played actively by over 265million people. (FIFA.com, 2015) Supported by around half the worlds population – 3.5billion. (Topendsports.com, 2015) One way fans interact and engage in the sport is through the means of fantasy football, in one of it's many forms.

Fans participate in fantasy football not just an excuse to watch football all day, but for reasons of fanship, arousal, camaraderie, competition, control and ownership. (Billings and Ruihley, 2013) It's estimated that 10% of the UK population participate in some form of fantasy sport (primarily football) and these numbers only grow by the millions worldwide with the popularity of the English Premier League. (Werber, 2012)

Many fantasy football games are in existence, either real world player based (picking your own team of real players to score points) or prediction based (scoring points based on the results of matches). Some of the most popular of those are:

- Fantasy Premier League (http://fantasy.premierleague.com/)
- Telegraph Fantasy Football (https://fantasyfootball.telegraph.co.uk/)
- Sky Sport Super 6 (https://super6.skysports.com/)
- TalkSport Selco Predictor (http://predictor.talksport.com/)

Each of these have their own sets of rules, seeing players competing individually against one another, with the player with the most points winning at the end of the season.
What none of these do however, is allow users to play with friends (or other players) as a team to compete against other teams of players - just like the real world game. Quite simply team-based fantasy football games don't exist. The creation of such a game has the potential to add an innovative dimension to the existing fantasy football market.

1.3 Aims

The aim of the project was to produce a web platform for which users can come together as teams (friends or otherwise) to predict real world football matches and compete against other teams in a fun and socially engaging manner.

1.4 Objectives

The project was split into some key objectives to help achieve the projects aim:

- Creation of a strong brand
- The outline a site structure and user journeys
- Site design though the creation of wireframes and mockups
- Database design
- Creation of a backend system in a MVC model using Laravel
- Test each model as it is created and added to the system
- Implementation of design within views
- Ensure platform is fully functional online
- Obtain user evaluation and feedback

1.5 Target Market

The background research indicated half the world's population are now football fans. Various factors however limit those who can be targeted by the application - not all of them will have internet access, suitable devices or have an interest in fantasy sports games. In the best case scenario the target market for the application is everyone who enjoys playing fantasy sports games and are interested in football. Based on numbers from America & Canada (who's numbers playing fantasy football, or soccer as they know it, are increasing dramatically), the average fantasy sports player is people aged between 14-50 and predominantly male. (Edition.cnn.com, 2010) (Fsta.org, n.d.)

Due to the strong European influence of football on the world stage and more specifically the English Premier League, a UK audience of males aged between 14-50 has been chosen as the initial target market for the application which will be the base for future growth.

2. CONCEPT AND PLANNING

2.1 The Idea

Sports fans participate in fantasy sports for many reasons but the concept of this project pulls many of the reasons into one. The idea introduces the idea of camaraderie in playing together as a team (possibly with friends), yet still being competitive in wanting to both beat other teams together, and also outscore team mates.

On top of that a social aspect in being able to discuss both real live matches and the fantasy matches a players team is involved could be added, getting to know other players, making new friends...or rivals.

2.2 Details

The concept of the social team based football prediction games sees players competing in teams of five, made up either of friends, other random players or a mixture of the two. Each matchday, five games are available to predict, the captain (usually the creator of the team) must assign one game to each player. In doing so this then allows that player to predict that game. Points (or goals) are scored based on the prediction against the real world result.

In predicting games, the individual members of the teams are free to predict how they wish, they can submit their own predictions without influence or alternatively discuss the game with their team members to come up with a team prediction which can be submitted by that player (but only the player assigned the game can submit the prediction for that game).

Each matchday then sees these teams go head-to-head against each other in a league or cup format. The cumulative goals scored across the 5 members of each team will make up the goals scored for that team. And just like real world football this will then determine the winner and award the team points (3pts for a win, 1pt for a draw, 0 for a loss) or decide who

goes through in the event of a cup. Teams can then be ranked within their league based on the results against competing teams. Each league will consist of a set number of teams (20 for example) from which fixtures can then be generated.

A back-end system allows administrators to manage league and fixture generation as well as set up matchday fixtures to be predicted and enter results. The front end of the site which players use is a responsive website allowing the site to be used across all devices. (Marcotte, 2010) (Knight, 2011) The application can also be saved to a users phone in the form of mobile web app.

2.3 Scope

It is important to remember the scope that the project was built under. The biggest constraint was the time to develop it, as it had to be complete by Friday 1st May 2015. Cost was minimal, with only server costs to factor in, as it was being built from scratch on an open source framework. In terms of resources, they are plentiful both in the way of PHP code snippets and help that may be required. The only other scope we will be working within is the world of football as the idea is defined around that.

2.4 Management

Management of the project was an all important part of getting the project delivered on time. To manage the project, the project management tool *'TeamworkPM' (Teamwork.com, 2015)* was used. Given the objectives set out, these were further refined into tasks and sub-tasks e.g. branding needs research, mood boards, brainstorming leading to a visual identity (which again can be split into subtasks). Each of these tasks and sub-tasks, as well as key milestones, were then given a date and time for completion. Tasks could then be marked off once completed to get a visual representation of the progress of the project. The easiest way to display this data was to use a gantt chart (See Appendix A).

2.5 Methodology

To assist with the management of a project it is best advised to select a suitable methodology. There are a wide range available to choose from, some of which are:

Waterfall - An almost linear approach where you design, code, integrate, test, deploy in that straightforward order. It's rigid and inflexible however which isn't much use when there maybe be points where you want to go back and make changes or refinements. (Istqbexamcertification.com, n.d.) (Segue Technologies, 2013)

Agile - A much more iterative approach to development where individual components are built and deployed over a short periods of time. It gets a working product released quickly but doesn't follow a structured plan and relies on interaction within team to be successful. (Istqbexamcertification.com, n.d.) (Agilemethodology.org, 2015)

Prototyping - Quite a complex model but one which is quite flexible. Prototypes are designed, developed and reviewed until they are right before being coded into the main application. Works well with high user engagement but can be time consuming, and time was one of the major risks and constraints to the project. (Istqbexamcertification.com, n.d.) (Rouse, 2005)

Those are just three of the many possibilities but it was another methodology which seemed like it would work best for the project – **iterative and incremental development**. (Techopedia.com, n.d.) This approach seen the project planned out and built in stages, with the product being in a working condition after each iteration. The difference to the agile development methodology is that with the iterative and incremental approach the minimal features are built first to get the application in a working state, then built upon and added to rather than building the fully featured component from the beginning.

2.6 Requirements Specification

The *Volare Requirements Specification Template* is a world popular template, and guide, for writing requirements specifications. It allows the requirements to be stated with it's type, priority, rationale, fit criterion and dependencies. The fit criterion is a way of measuring the requirement to ensure the solution meets the requirement. Each requirement takes into account the projects purpose, it's users and the constraints. (Volere.co.uk, 2015)

Volare '*Snow Cards*' are a useful aid in working out the top level requirements of a project by outlining the attributes required for these. Examples of some snow cards relating to the project can be seen in figures 1 & 2.

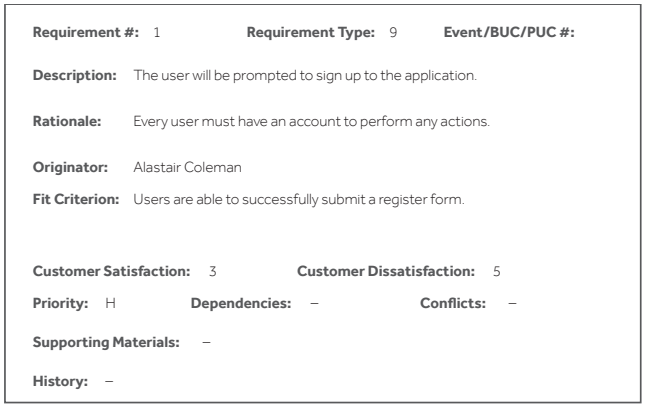


Figure 1 - Snow Card for requirement #1

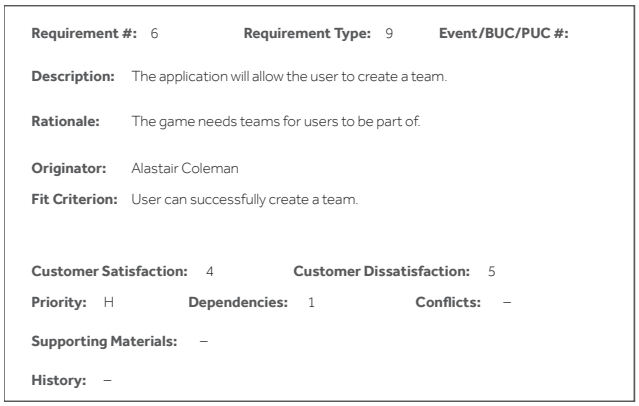


Figure 2 - Snow Card for requirement #3

Due to the volume of possible snow cards and the difficulty in updating and keeping track of cards, the requirements can be best outlined in a table, as seen in Appendix B.

3. DESIGN

3.1 Paper Prototyping

The paper prototyping process followed several stages leading to the creation of detailed refined sketches and style guides for the application.

3.1.1 Mind Map

Before beginning any sketches or mockups it's always a good idea to try and a vast array of thoughts and ideas down on paper. This way all the ideas can be seen in one place, reflected upon and the best ideas picked out that could come together and possibly harness new ideas for the project. One of the great ways of doing this is with a mind map to visually represent the data. This can be seen in Appendix C.

3.1.2 User Journey

After looking at the mind map and taking any ideas from it that could be used to further enhance the project, it was time to outline how the user would use the application. This was done by looking at potential user journeys within the application (Appendix C). This user journey helped outline the various pages and actions required within the application. These could then be sketched out to provide an idea what these may look like.

3.1.3 Initial Sketches

The initial sketches of the application were very low fidelity so lots of ideas thrashed out on paper quickly and easily. Sketches were created for various different aspects of the application – login and register, team creation or joining, profiles, match assignment, predictions, leagues, fixtures and the administrative side to the application. The sketches for various potential login/register screens can be seen in figure 3, with sketches for the various other screens within Appendix C.

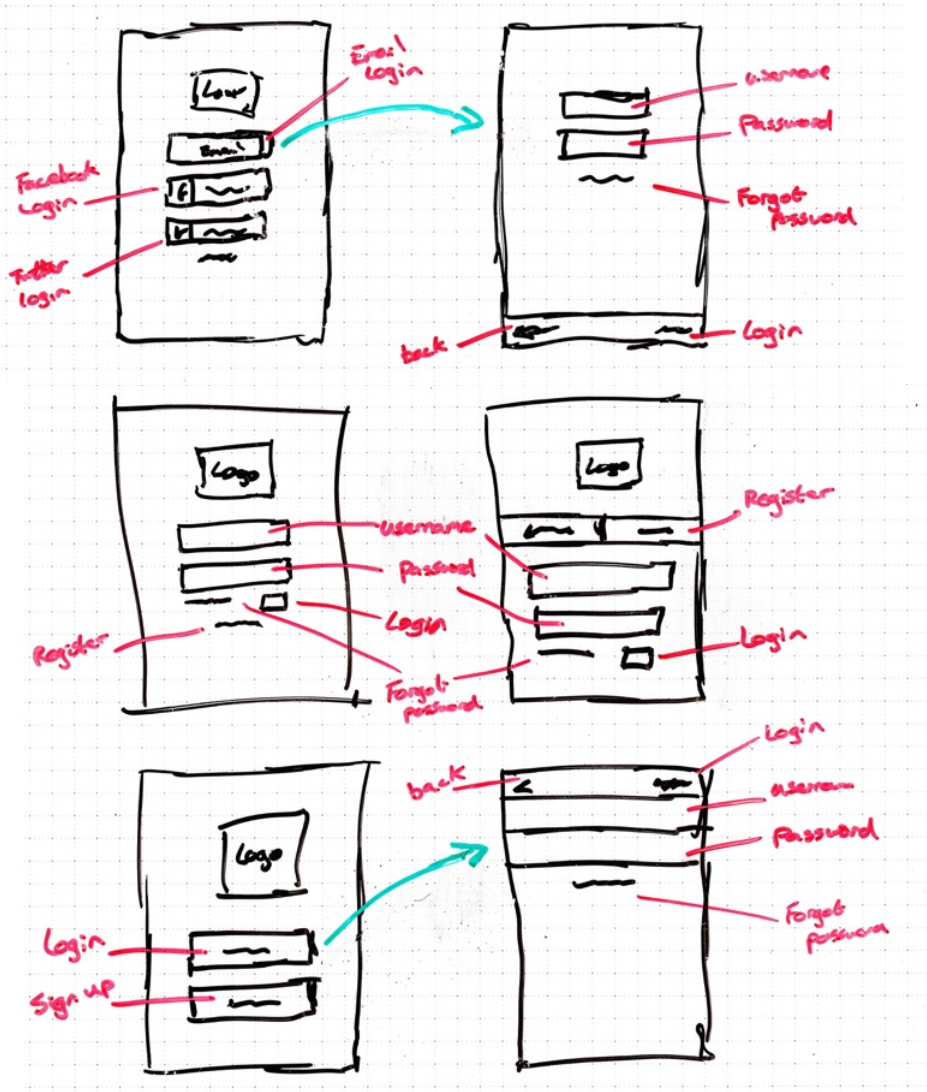


Figure 3 - Login Screens Paper Prototypes

3.1.4 Detailed Refined Sketches

Following the initial sketches more time was spent creating higher fidelity refined sketches of potential screens to outline the finer details within them. This included the introduction of some colour and imagery to begin to give the application a more visual representation of what the final application could look like. Two of the more detailed refined sketches can be seen in figure 4 with the remainder of the sketches available in Appendix C.

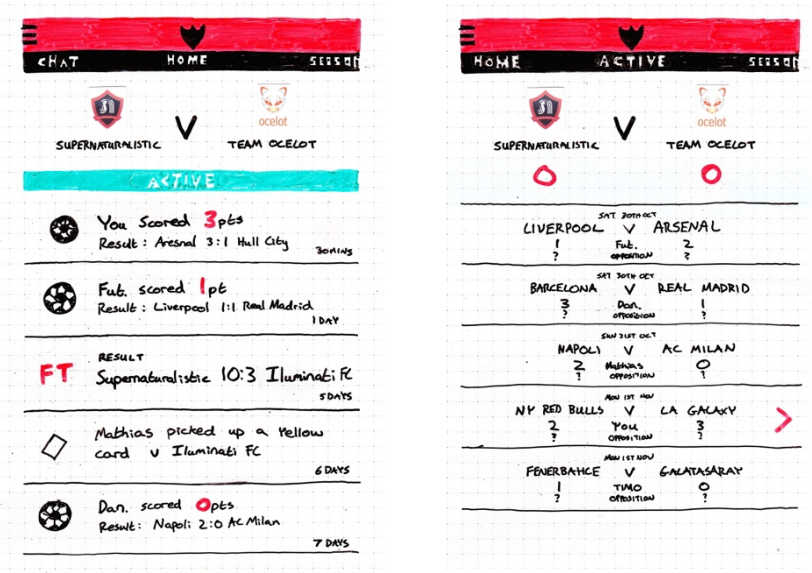


Figure 4 - Refined Detailed Screens

3.1.4 Style Tiles

Another thing that was created to help in the creation of digital visual mockups was a style tile. These helped to give a visual feel for how the application and site would be combining fonts, colours and imagery. Two possible style tiles were created, the chosen style tile can be seen in figure 5 and the alternative seen in Appendix C.

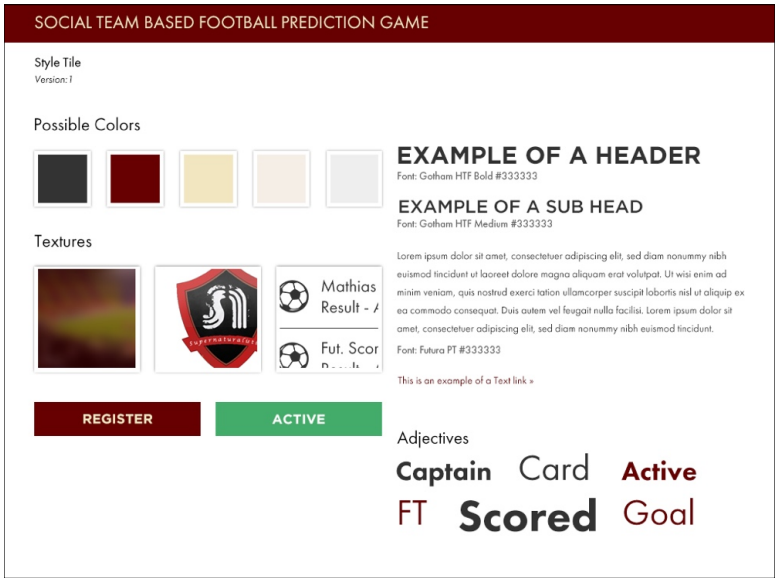


Figure 5 - Style Tile

3.2 Visual Design

The paper prototyping process helped to really refine the concept and give a clear vision for the project. The creation of higher fidelity digital visual designs were then created to show what the application would look like on screen and begin to take into account the users interactions within the system.

These visual designs for the application were created in Photoshop, and built upon the chosen style tile from the paper prototyping process. A few of screens created can be seen in figure 6.

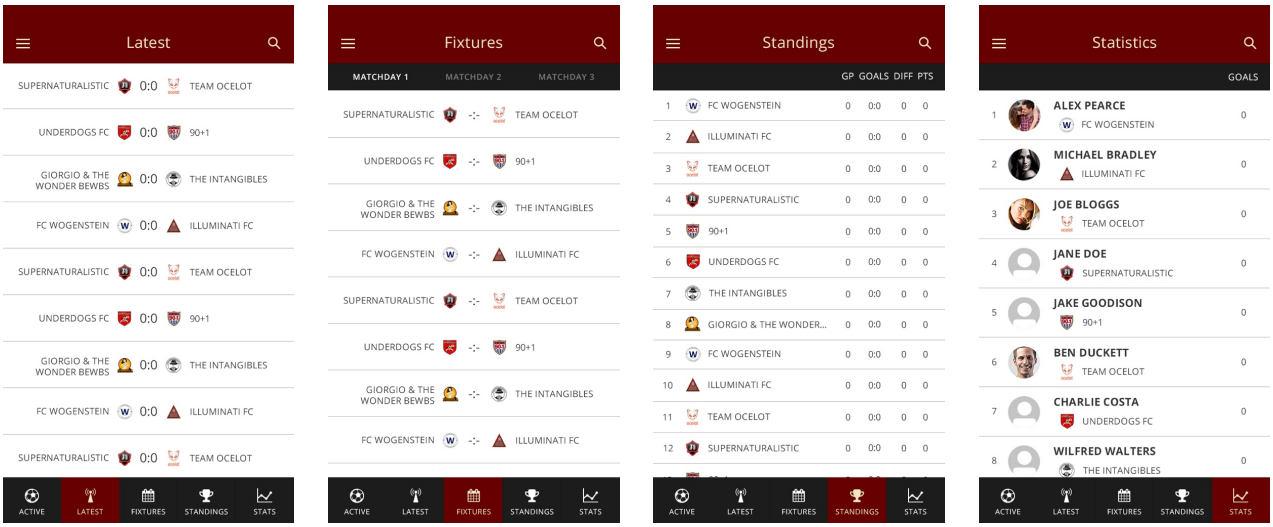


Figure 6 - Sample Visual Designs

In all, dozens of visual mockups were created, the remainder of which can be found in Appendix C, which display various aspects of the system and how users interact with them. The designs are clean, modern and use common interactions which users would find natural when using the application.

The screens created were used to develop an interactive design prototype which can be seen at the following url - http://invis.io/RM1UYIDDC.

3.3 System Design

The design of the system takes into account various components and their relationships. These can range from the system architecture to database design or high-level design which delves deeper into the workings of specific components such as data flow diagrams.

3.3.1 Client-Server Model

The client-server model outlines the network architecture of the system. The model outlines the various elements which lie on the client side and the server side of the system as seen in figure 7. The server uses a LAMP stack (**L**inux, **A**pache, **M**ySQL, **P**HP) installed on a VPS Cloud Server hosted on Digital Ocean. (Digitalocean.com, 2015) (Stackoverflow.com, 2015)

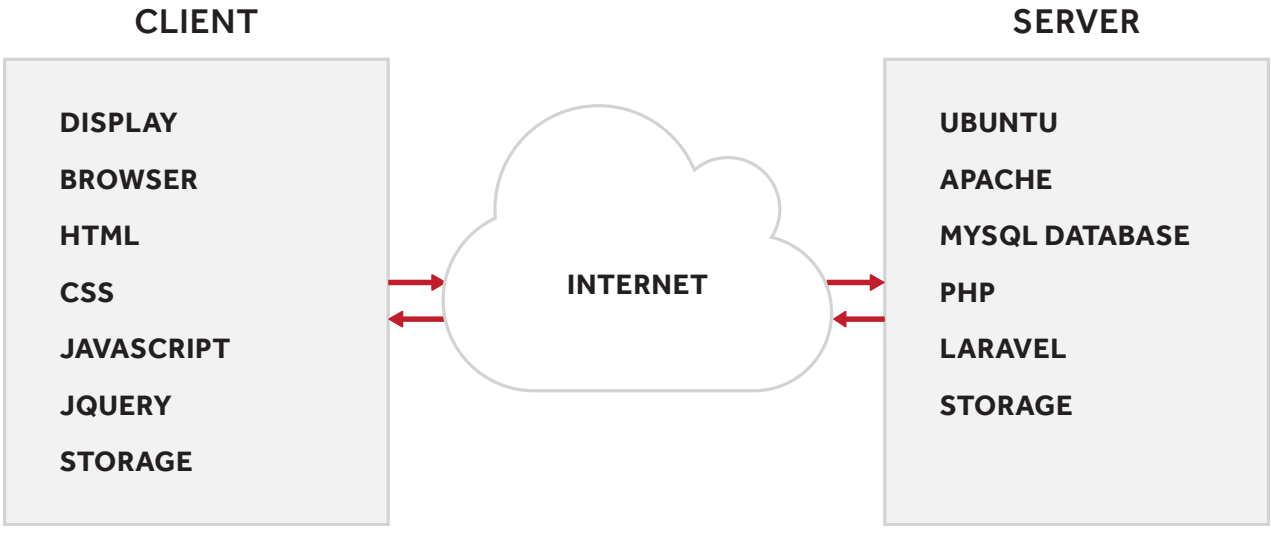


Figure 7 - Client Server Model

3.3.2 Data Flow Diagrams

There are two main components of data flow within the system. The first is the user login and registration, the second is the assignment and submission of predictions by the players

within the team. Figure 8 shows a simple diagram of how these two parts of the system work, first the login.

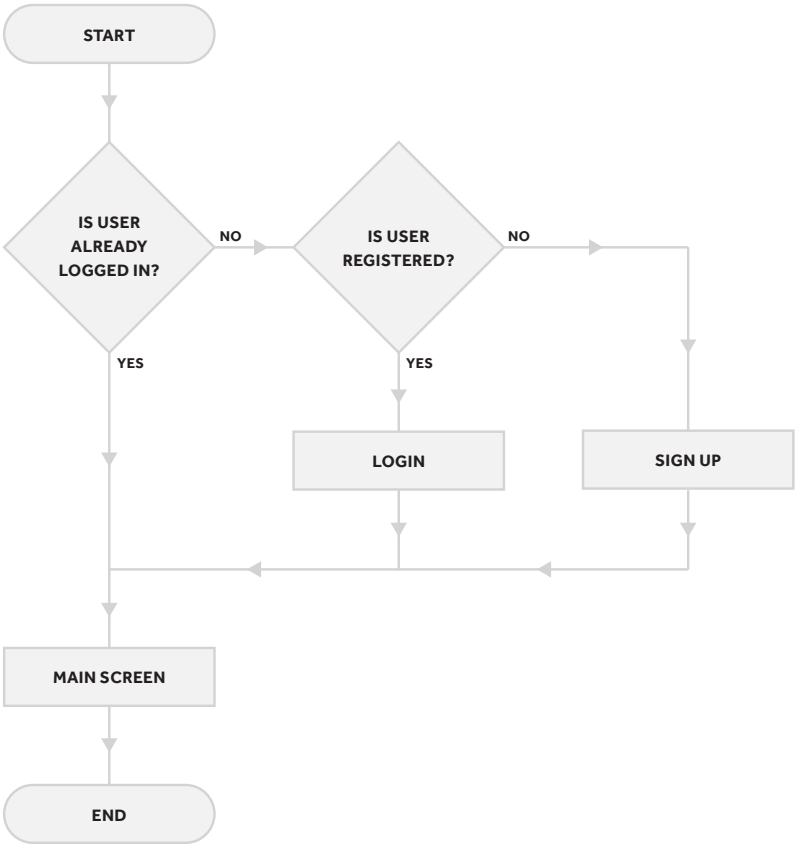


Figure 8 - Login Flow

The login flow diagram in figure 8 is a fairly simple one. If the user is already logged in, take them to the main screen, otherwise if they're registered they'll want to log in and if not they'll want to sign up. Upon either logging in or signing up they will then be taken to the main screen.

This is highly simplified, but it can be narrowed down and refined to show how each section will work and how the data flows within the system. The more complexed high-level data flow diagram in figure 9 shows how much more additional functionality is needed to actually manage the data throughout the process. Retrieving data from cookies, handing data through forms, validating it, storing and reading it from databases and just to manage user registration and login.

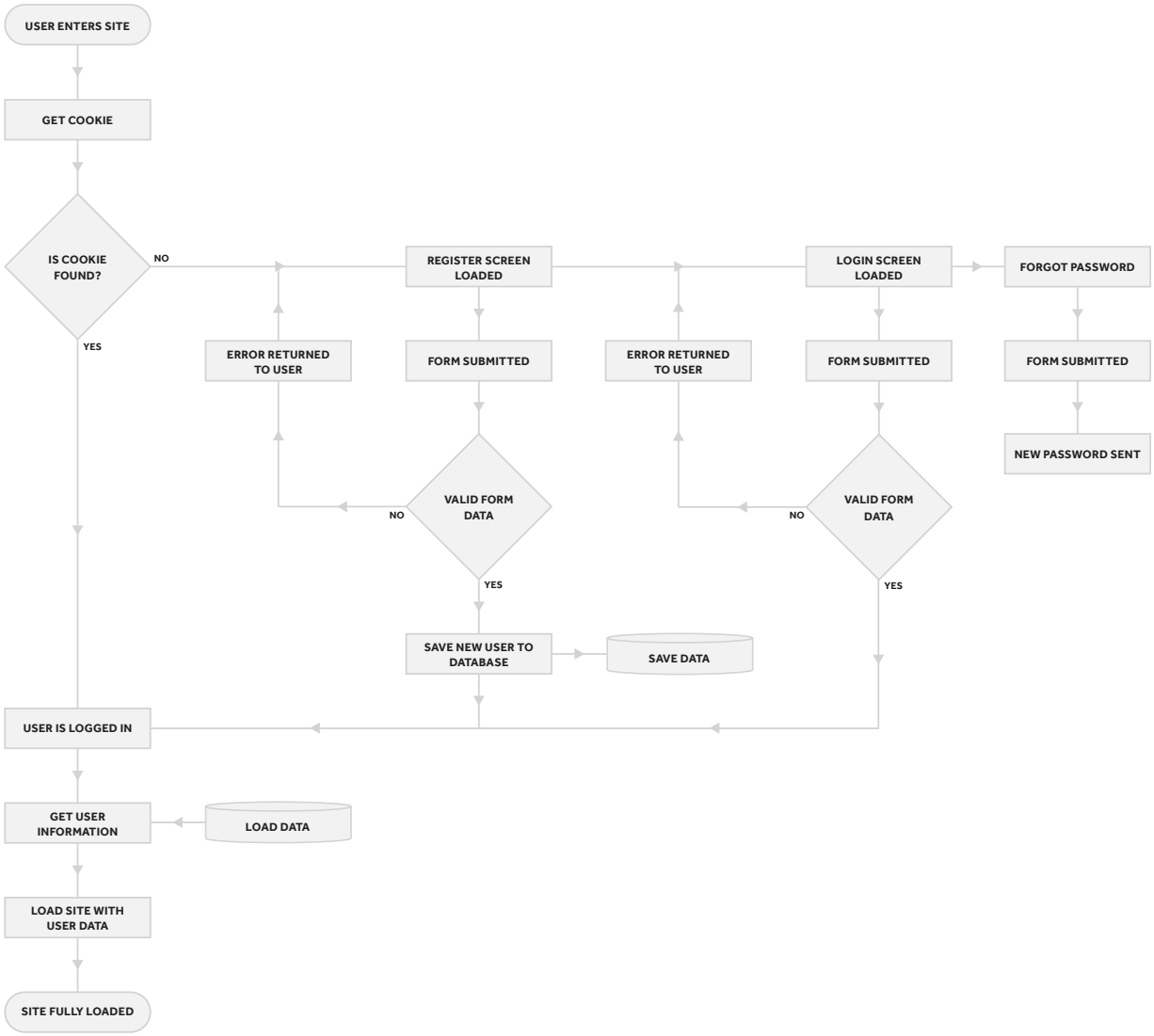


Figure 9 - Login Data Flow Diagram

The other major part of the system, the match assignment and predictions follows in a similar manner, the overview looks simple with the data flow diagram being much more complex.

As can be seen in figure 10, it's a fairly simple process – check if the user is the team captain if they are, let them assign games for the team. If not, check if games have been assigned and if they have been they can be predicted and allow the user to submit their prediction, otherwise the user can't predict.

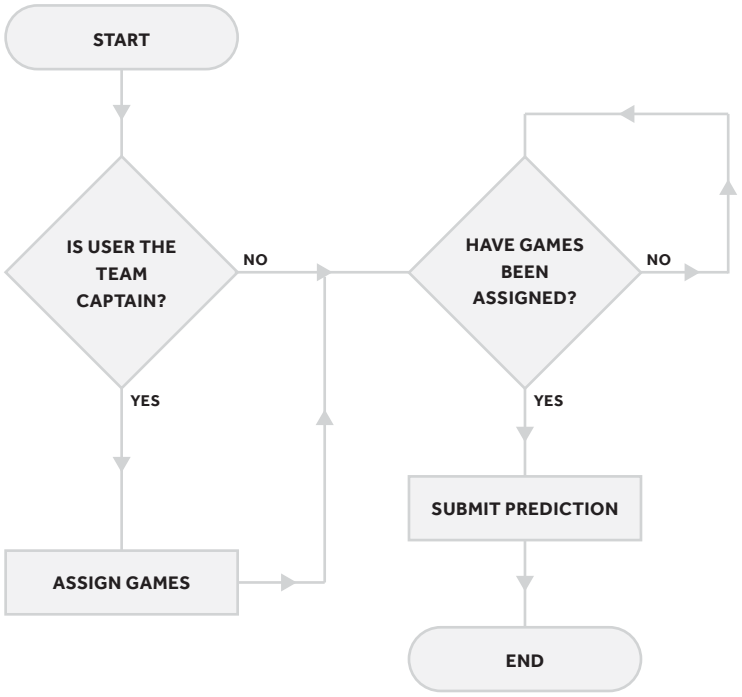


Figure 10 - Match Assignment & Predictions Diagram

Again, when each component in taken turn and the workings behind each component is looked at it helps understand how each section will work and how the data will flow within it. This can be seen in Appendix D.

Additional data flow diagrams include those of the players joining or creating a team and the administrators role within the game in choosing matches and updating results. Again these can be seen in Appendix D.

3.3.3 Model-View-Controller

Laravel uses an MVC model which was used for the project. The model-view-controller (MVC) model splits an application into three main components the model, the view, and the controller. Laravel also comes with a powerful routing engine which maps urls to controllers.

How each of these parts of the application are interlinked can be seen in figure 11 below. (Ehsan, n.d.)

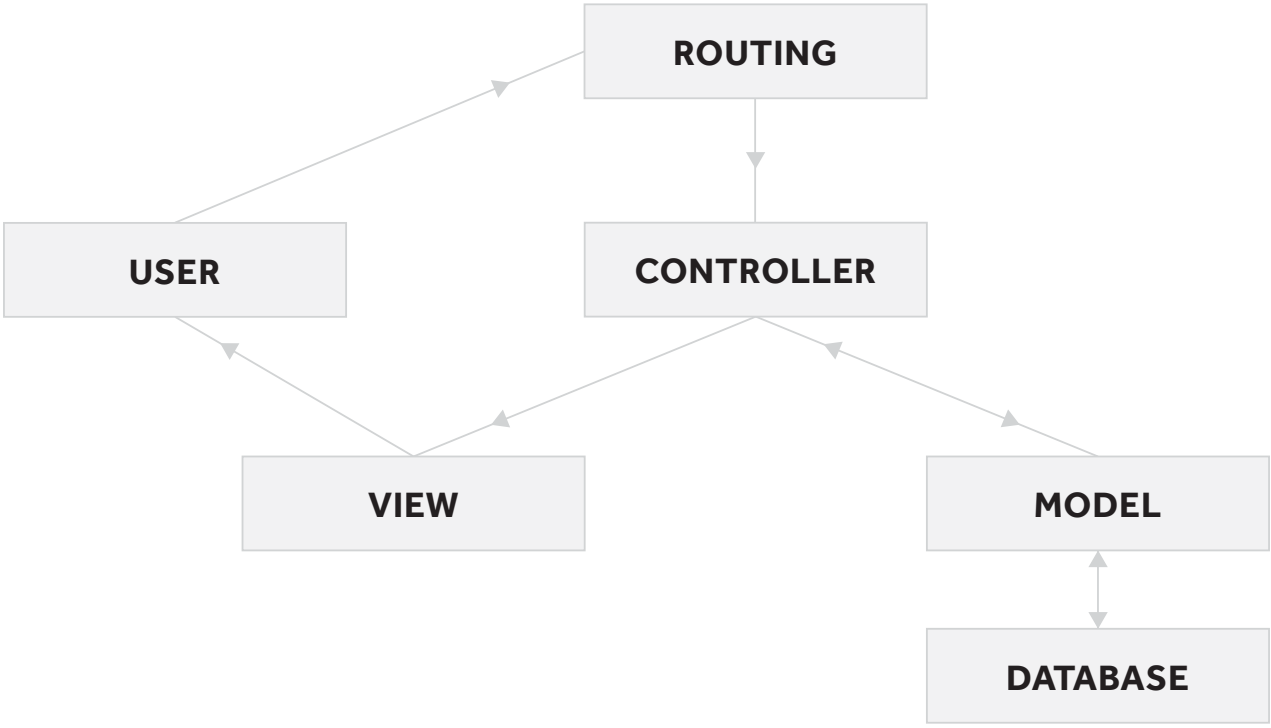


Figure 11 - Laravel MVC Architecture Diagram

The routing engine within Laravel not only matches the url to a controller but it also maps to methods within the controller and allowing variables to be passed between the view and controller easily. The controller then basically manages the operations on the server, controlling things. It processes inputs, decides on models to call, chooses views to be sent to the user and handles redirects.

The model is tasked with handling the data. Not only is it used to store and retrieve data from the database, it is used to validate data and perform any necessary operations on the data to be passed to the controller and views. These views are then responsible for the visual output sent to the users browser.

3.3.4 Database Design

An integral part of the system design is the design of the database. The data being stored for the application must be stored in a way in which the various aspects can be linked together in

an easy to work with manner within our application. The relational database design diagram for the application can be seen in Appendix D.

3.3.5 Game Mechanics

The final important factor within the system design is the intricacies of the game mechanics. This includes the scoring system and fixture generation.

Scoring

There are two calculations to be worked out on completion of matches and fixtures, these are calculating the users score based on prediction; and calculating the match day results based on these scores.

The calculations for the users predictions scores are based on the following criteria:

- 5 points if a user correctly guesses the exact score
- 4 points if a user correctly guesses the result and score for one team
- 3 points if a user correctly guesses the result
- 1 point if a user incorrectly guesses the result but correctly guesses score for one team
- 0 points if the user incorrectly guesses result and either teams score

Examples of the above calculations can be seen in table 1 below. Each of the different user predictions result in each of the possible scoring outcomes with the actually result remaining the same in each case.

Table 1 - Scoring Calculations

User Prediction	Actual Result	Points
2-1	2-1	5
3-1	2-1	4
1-0	2-1	3
1-1	2-1	1
0-0	2-1	0

The calculations for the match day results would be the same as those in league based football in real life whereby the winning team scores 3 points, losing team scores 0 points and both teams scoring 1 point if it's a draw. (Robinson, 2015) The result is calculated based on the total points accumulated by the 5 members of the team.

An example of how this is calculated can be see in table 2 below. Each of the players in the teams' scores have been calculated and from this the result of the match and points to be award can be calculated. As can be seen, Team 1 wins and is awarded 3 points for the win with team 2 receiving 0 points.

Table 2 - Team Scoring Calculations

Team 1		Team 2
5	Player 1	3
1	Player 2	1
3	Player 3	4
0	Player 4	1
1	Player 5	0
10	**Total**	**9**
3	**Points Awarded**	**0**

Fixture Generation

The generation of fixtures for the teams playing each other week by week could be generated in a number of ways. One of the major factors in the process was the number of teams playing on each match day, as an even number of teams are required to ensure each team has a match. Another factor was the entry of new teams into the game and how they can be integrated into the game.

One of the ideas in generating the fixtures would be have it on a divisional basis, whereby teams are limited to say 8 or 10 to a division so therefore there is always an even number of teams. The division will only get underway when the required number of teams has been reached, so for example there were 33 teams, the first 30 entered would be placed in 3 divisions of 10 with the remaining 3 places on a waiting list awaiting another 7 teams to enter. Once another 7 teams entered this divisions fixtures could then be created and the league get underway.

Another way of going about it would be to have fixtures randomly generated on a weekly basis. If there happens to be an odd number of teams, one team will have a bye for the week, however the fixture generation would need to take into account the number of byes a team has had so as a team wouldn't end up having a lot more byes than other teams. And in a similar manner it would need to take into account the teams played recently so teams didn't end up playing the same team multiple times in quick succession. Another disadvantage of this would be the teams entering later would be highly unlikely to catch the teams at the top of the league as they will have played far less games. To counteract this there would need to be a set start date for the game which after no more teams could be entered.

The method used for the current build of the application is a mix of the two. A set number of even teams will be allowed to enter with a start date set. This will run for half a season in which time new teams will be allowed to register for the second half of the season. Fixtures can then be generated for a set number of weeks, to decide the final standings at the end of the half season. Then depending on finishing positions teams can qualify for an end of season play-offs.

4. IMPLEMENTATION

4.1 Technology

In the creation of the project a wide range of technologies including server-side languages, client-side languages, a database, frameworks, libraries and other technologies were used.

4.1.1 Server-side

The choice of server-side language can be a difficult one, everyone has their favourites, but that doesn't mean it's necessarily the best for the job at hand. Server-side languages are those that run on the server to help create dynamic web-pages – handling user inputs, manipulating databases, displaying content from databases etc. PHP is the dominant server-side language on the market and this was a key factor in its choice for the project.

PHP (PHP: Hypertext Preprocessor)

PHP is a widely used (approximately 82% of websites' server-side language), open source and free to download scripting language. (W3techs.com, n.d.) It is compatible with almost all servers, web platforms and supports most databases making it an easy server-side language to get up and running. (W3schools.com, n.d.)

The language is a fairly simple one to learn with tons of built in functionality as well as many advanced features to create powerful web applications. As PHP is an open source language used by millions, there is plenty of great documentation and support available should any help be required. (Php.net, n.d.) (En.wikibooks.org, n.d.)

The primary reasons for the decision in using PHP was that, from experience, it is much simpler than other languages like Ruby to get up and running on a server, and as PHP is such a widely used language, much more help and resources were available for any issues that would arise.

4.1.2 Client-side

The languages used on the client-side are those used within the browser that the user sees or interacts with. A browser is used to render the code and display it on the screen for the user. The following are some of the client-side languages used.

HTML

HyperText Markup Language (HTML) is a markup language used to create and visually display a webpage within a browser. It uses tags to define different content types within a page which are then styled by the browser or CSS. HTML5 is the most current version, still being standardised, but most elements are supported by latest browsers. (Mozilla Developer Network, n.d.) (W3schools.com, n.d.)

CSS

Cascading Style Sheets (CSS) is a stylesheet language used to style web documents, allowing the manipulation of how the elements once a page appear within a browser. CSS3 is the current version, and like HTML5, is still being standardised, but the majority of browsers support most elements, with browser prefixes also available to help support. (W3.org, n.d.) (Mozilla Developer Network, n.d.)

JS/jQuery

Javascript (JS) is essentially the scripting language for web pages, allowing various actions to be performed client-side rather than server-side. It is a lightweight, object-orientated language which provides a huge amount of functions within a webpage such as change, delete, add, copy elements, change attributes, add styling, validate data, fetch data etc. This adds a huge amount of flexibility to what can be done with webpages. There are also various libraries available (jQuery being the most popular) which can be used to simplify the writing of the code to perform tasks more easily. (W3schools.com, n.d.) (Mozilla Developer Network, n.d.)

The project required all of the above client-side languages to make the application function to meet the desired requirements. HTML was required to create pages, CSS used to style them and then Javascript used to manipulate the data to create dynamic pages to better the user experience.

4.1.3 Database

A database is typically used to store a collection of data. This data can then be interacted with by the server-side language to create/read/update/delete the data as necessary. Databases are made up of tables, records and fields. A table is a collection of records of the same type, records a collection of fields and a field is a single piece of information. (Webopedia.com, n.d.) The choice for type of database for the project was MySQL.

MySQL

The most popular and widely used database system for the web. It uses standard SQL syntax, is easy use, fast and reliable. MySQL runs on a server and is capable of handling virtually limitless volumes of data, making it suitable for applications of all sizes. It can suffer from performance issues when there are high volumes of concurrent operations and isn't quite as stable as some alternatives. MySQL is free to download and use, and is found on most hosting platforms. (W3schools.com, n.d.) (Mack, 2014)

The reasons for choosing MySQL is that it is quick, easy to use, simple to set up, can handle plenty of data and is a familiar database system.

4.1.4 Framework

A framework for the web helps provide additional functionality which can be selectively changed and extended to meet the needs of the user. There are various frameworks available for lots of different languages, a few of which were used for the project. (Stackoverflow.com, 2010)

Laravel

Laravel is a free open source Model-View-Controller (more on that later) framework for PHP and has become the number one framework within the PHP community. The framework is lightweight and robust with plenty built in functionality and add ons to extend that even further. The composer install makes it simple to get started and it's Eloquent ORM is famed for making code within PHP much more readable and easy to write. (Way, 2012) (Otwell, 2015)

Bootstrap

Bootstrap is the most popular front-end framework for the web, it contains all the elements to build a responsive site quickly and easily to work across multiple devices. Bootstrap contains additional functionality to that found in other frameworks such as Foundation but also lacks others, so it just depends on the needs of the project as to which is best suited. (Mark Otto, 2015)

Laravel is a very good PHP framework, with some of it's built in functionality such as authentication coming in very useful to help streamline the development process for our application as well as its easy to understand and simple to write nature. The latest version of Laravel – Laravel 5, also comes with bootstrap installed out of the box, therefore it made developing a responsive application within the framework a simple process.

4.1.5 Other

There are a few other technologies that were used throughout the project that are outlined below.

SASS

Syntactically Awesome Stylesheets (SASS) is a stylesheet language which is a preprocessor for CSS. It allows for the use of nested rules, variables, mixins, selector inheritance, and more all within one file which is then used to generate our CSS. This allows the CSS to be written

quicker, easier and more elegantly than before. (Sass-lang.com, 2015) (Mindscapehq.com, n.d.)

Gulp

Gulp is a simple easy to use task runner used to automate tasks. It comes with a whole host of plugins to allow for the likes of minification of JS files, optimisation of images or compiling of SASS. (Gulpjs.com, 2015)

4.2 Feasibility Testing

To help test the feasibility of the project a functional prototype was developed to explore the key risk of the project – the use of Laravel as the framework. The functional prototype tested certain aspects of the initial system design and allowed for a deeper understanding of how going about building the system was going to work.

The choice of Laravel to be used as a framework for the project brought with it the Model-View-Controller (MVC) architecture which helps separate the code for these parts of the system to make it more easily readable and manageable. The Laravel framework proved to be helpful during the building of the prototype with its built in components for configuration, validation, event listening, mailing etc. By successfully managing to create a user registration and authentication system using Laravel it was deemed that the project was feasible and development of the application could continue.

4.3 Challenges

The main challenge in building the project was the use of Laravel as a framework to develop the application. Having never used Laravel before, learning the framework and being able to implement all the necessary features and functions was one of the most challenging factors. To help develop the understanding required, the documentation and Laracasts proved vital. (Laracasts.com, 2015)

In terms of challenges relating to coding, there were several which posed some challenges to deal with.

- The use of jQuery and AJAX to create more dynamic pages than those of the functional prototype which used static views and pages for each of the screens.
- The creation of private teams which could only be joined by using a password set by the team captain.
- Restricting access to the ability to assign matches to only the team captain.
- Restricting access to predictions so only the user assigned the match could predict that match.
- Restricting access to allow only members of a team to view the submitted predictions for that team until the match kicked off.
- The generation of fixtures at the beginning of season once team quota has been reached.
- The processing of matchday results to calculate the scores for each prediction and fixture, then update the in-game scoring and statistics accordingly.

Other less pressing challenges which posed risks to the project included the likes of data loss, time, extenuating circumstances etc for which every precaution was taken against. Some extenuating circumstances cannot be guarded against other than allowing plenty of spare time within the project. Other holidays and events had been accounted for.

The table in Appendix E gives a quick overview of the aforementioned challenges and risks and a few of the lower probability ones.

4.4 Achievements

Each of the challenges were overcome during the implementation of the project. Between the creation of the functional prototype and the building of the application, a new version of Laravel was released – Laravel 5. This brought with it some major changes to the framework, all of which had an impact on achieving the challenges posed by the project.

4.4.1 Laravel

The Laravel framework makes use of the PHP dependency manager '*Composer'* and its own CLI (command-line interface) called '*Artisan*' to help perform several of the common actions used within the framework. (Getcomposer.org, 2015) (Otwell, 2015) These commands can help migrate databases, create database schema, generate models, controllers and middleware and various other helpful functions.

The commands are run using terminal by navigating to the Laravel installation directory and using the php artisan command. Examples of it's usage to generate database schema and then run the migration to create the table in the database can be seen in figures 13 & 14. The schema generator makes use of an extension to the Artisan CLI called '*Laravel 5 Extended Generators*' which was installed via composer as seen in figure 12. (GitHub, 2015)

```
$ composer require laracasts/generators –dev
$ composer update
```

Figure 12 - Installing Laravel 5 Extended Generators via Composer

```
$ php artisan make:migration:schema create_teams_table –
schema="team_name:string:unique, captain_id:integer:foreign,
logo_url:string, private:boolean, team_code:string, played:integer,
won:integer, drew:integer, lost:integer, goals_for:integer,
goals_against:integer, points:integer, ranking:integer"
```

Figure 13 - Creating the Teams Migration

```
$ php artisan migrate
```

Figure 14 - Running the Migration

In being able to successfully create database schema, tables and their associated models which were also generated with the migration, it made the process of storing and managing our data much easier than it would have been.

The Laravel framework uses the Eloquent ORM to provide a simple ActiveRecord implementation for working with the databases. This uses the created models to retrieve and create records from the tables using an object orientated programming approach without having to write raw SQL unless you chose to do so. (Otwell, 2015)

Laravel's powerful routing engine was another important thing to learn about the framework. The ability to specify the types of HTTP requests allowed to specific urls and in turn map them to controllers and methods proved to be a tricky prospect at first however with continued use the concept was eventually grasped.

4.4.2 AJAX Integration with Laravel

In the development of the functional prototype all the views were made statically and each time a method was called it returned a view to the user requiring the page to refresh. For the final application jQuery and AJAX was used to post the data required and in turn a JSON response returned without the need to refresh the page.

As this was the first time using Laravel, the thought process behind doing this seemed to be daunting at first. In the end however it proved quite simple, using routes to handle the AJAX requests and then return a JSON response using Laravel's JSON method within its response helper. (Otwell, 2015)

This was used throughout the application with one of the best examples being that of the predictions submission – when a user enters their prediction and it is submitted via AJAX and a response to indicate that is was successful returned to the user. The prediction submission view (figure 15) and corresponding AJAX (figure 16) and method (figure 17) can be seen below.

The jQuery AJAX function (figure 16) simply attaches a click event handler to the submit button which is located at the top right of the screen as seen in figure 15. Once clicked a

spinner indicates that the button has been clicked, and the data within the form is serialised and stored in the data variable.

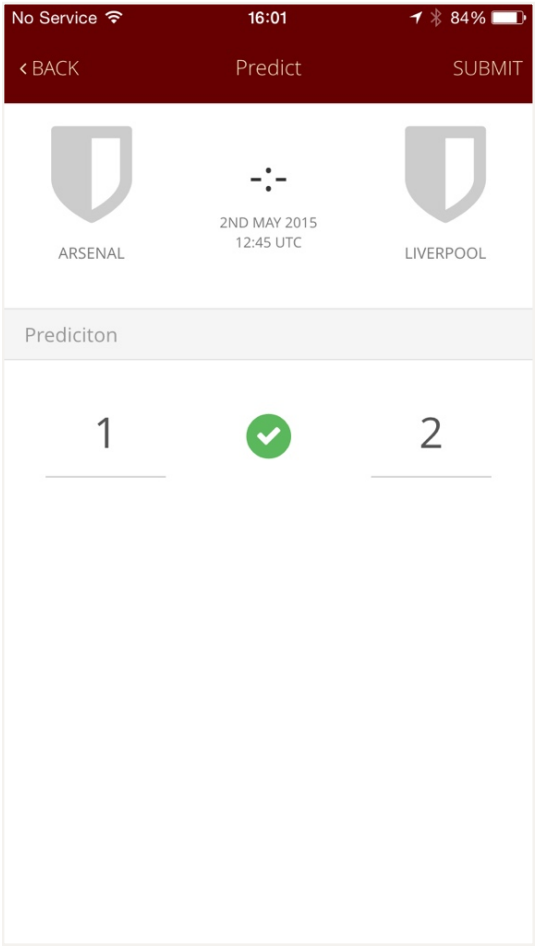


Figure 15 - Prediction Submission View

```
$('#submit-prediction').click(function(e){
  $('#status').html('<i class="fa fa-spinner fa-2x fa-pulse"></i>');
  var data = $('form').serialize();
  $.ajax({
    url: '{{ route('match.predict.save', $prediction->id) }}',
    data: data,
    type: 'POST'
  }).success(function(data){
    $('#status').html('<span class="fa-stack fa-lg success"><i class="fa
    fa-circle fa-stack-2x"></i><i class="fa fa-check fa-stack-1x
    fa-inverse"></i></span>');
  }).error(function(){
    $('#status').html('<span class="fa-stack fa-lg danger"><i class="fa
    fa-circle fa-stack-2x"></i><i class="fa fa-close fa-stack-1x
    fa-inverse"></i></span>');
  });
  e.preventDefault();
});
```

Figure 16 - Submit Prediction AJAX Call

```
public function predictSave($id)
{
  $prediction = Prediction::find($id);

  if (Auth::user()->id != $prediction->user_id)
  {
    return response()->json('You do not have permission to access this page.', 402);
  }

  $data = Request::all();
  $validator = Validator::make($data, $this->predictionRules, $this->messages);

  if ($validator->fails())
  {
      return response()->json($validator->messages(), 422);
  }

  $prediction->home_team_score = $data['home_team_score'];
  $prediction->away_team_score = $data['away_team_score'];

  if($prediction->save())
  {
    return response()->json('Happy days!', 200);
  }

  return response()->json('Oops, something went wrong with your prediction.', 422);
}
```

Figure 17 - Method to Save Prediction

The AJAX function then posts the data to the route '*match.prediction.save*' to save the prediction to the database, passing with it the unique id of the prediction. This route maps to the predictSave() function within the prediction controller (figure 17). This function gets the prediction from the database based on the unique id passed to the method through the url and checks to make sure the current user has been assigned that prediction, if they haven't an error is returned.

Otherwise, the submitted data is fetched and stored in a data variable which is then validated, against a set of rules defined in the controller, using Laravel's '*Validator*' service.

(Otwell, 2015) If the validation fails the errors are returned as JSON otherwise the '*home_team_score*' and '*away_team_score*' properties of the prediction object are updated to the submitted values. The prediction is then saved to the database and if successful a 200 HTTP response code is returned otherwise a 422 HTTP response code is returned complete with messages. The AJAX function receives these responses and indicates the success or failure to the user with respective icons.

4.4.3 Restricting Access

The ability to restrict access to certain parts of the application is required to ensure no unauthorised access to the application is obtained and stop any potential cheating within the game. Laravel includes an HTTP middleware which provides a convenient mechanism to filter HTTP requests within the application. (Otwell, 2015) This can be used to check the request before a method is ever called for the route and perform various actions such as redirecting the user if they aren't permitted to access a page.

```
$ php artisan make:middleware IsCaptain
```

Figure 18 - PHP Artisan Call to Create Middleware

```
class IsCaptain {

  public function handle($request, Closure $next)
  {
    if ($request->user() && !$request->user()->hasRole('captain'))
    {
      return redirect()->route('team.profile', $request->user()->team_id);
    }
    return $next($request);
  }

}
```

Figure 19 - Generated Middleware Class

To create a middleware the Artisan CLI is used (figure 18), creating a class that is then used to define the middleware. The example shown in figure 19, shows the middleware to check that the currently logged in used is the captain of a team. The function checks if a user is signed in and doesn't have the role of captain, redirecting them if this is the case. Otherwise it allows the user to continue with the request. This can then be defined within the *'Kernel'* class (figure 20) and called on any of the routes within the application, as shown in figure 21.

```
protected $routeMiddleware = [
    'auth' => 'TTPL\Http\Middleware\Authenticate',
    'auth.basic' => 'Illuminate\Auth\Middleware\AuthenticateWithBasicAuth',
    'guest' => 'TTPL\Http\Middleware\RedirectIfAuthenticated',
    'admin' => 'TTPL\Http\Middleware\IsAdmin',
    'teamless' => 'TTPL\Http\Middleware\HasNoTeam',
    'team.member' => 'TTPL\Http\Middleware\HasTeam',
    'team.captain' => 'TTPL\Http\Middleware\IsCaptain',
];
```

Figure 20 - Middleware Defined in Kernel

```
Route::post('/matchday/assign', ['as' => 'matchday.assign.create', 'uses' =>
'PredictionController@assignMatches', 'middleware' => 'team.captain']);
Route::post('/matchday/assign/validate', ['as' => 'matchday.assign.validate', 'uses' =>
'PredictionController@validateAssignedMatches', 'middleware' => 'team.captain']);
Route::get('/matchday/assign', ['as' => 'matchday.assign', 'uses' =>
'PredictionController@assignMatchesForm', 'middleware' => 'team.captain']);
```

Figure 21 - Middleware Called within Routes

Similar middleware was created to check if the user was an administrator, had no team or was a member of a team, as seen in figure 20. Once the challenge of creating the first middleware was successful it was simple enough to create the others in a similar manner.

4.4.4 Matchday Calculations

The final challenge posed was the processing of matchday results to calculate the scores for each prediction and update the in-game scoring and statistics accordingly. The processing of the matchday results needed to be ran once a result had been entered for a real world match within the administrative side of the application. On submitting a result the saveResult() method was called within the match controller which validated the data and

updated the database accordingly. Upon the match being successfully updated the processMatchPredictions() method is called (figure 22).

```
public function processMatchPredictions($match)
{
  foreach($match->predictions as $prediction)
  {
    $goals = 0;

    if ($prediction->home_team_score === $prediction->match->home_team_score)
    {
      $goals += 1;
    }

    if ($prediction->away_team_score === $prediction->match->away_team_score)
    {
      $goals += 1;
    }

    if (($prediction->home_team_score > $prediction->away_team_score) && ($prediction->match->home_team_score > $prediction->match->away_team_score))
    {
      $goals += 3;
    }

    if (($prediction->home_team_score < $prediction->away_team_score) && ($prediction->match->home_team_score < $prediction->match->away_team_score))
    {
      $goals += 3;
    }

    if (($prediction->match->home_team_score === $prediction->match->away_team_score) && ($prediction->home_team_score === $prediction->away_team_score))
    {
      $goals += 3;
    }

    $prediction->goals = $goals;
    $prediction->save();

    $user = $prediction->user;
    $user->goals = $prediction->user->predictions()->sum('goals');
    $user->played = $prediction->user->predictions()->count();
    $user->save();

    $team = $prediction->user->team;
    $team->goals_for = $team->members->sum('goals');
    $team->save();

    $this->rankUsers();

    $this->rankTeams();

  }
}
```

Figure 22 - Method to Process Match Predictions

The function takes the current match as a parameter and then uses a foreach to loop through each of the predictions for that match. This then calculates the goals awarded for that prediction by comparing the predicted scores to the actual match result scores and awarding goals as defined in table 1 within the system design. The total goals is then saved to the prediction.

Following the prediction being saved the user and team stats need updated to reflect the new results. The user is retrieved from the prediction object and the goals and played fields updated by using some special methods such as sum() and count() with are included within

Laravel's Query Builder. (Otwell, 2015) The users team is then fetched from the user object and the teams goals_for field updated again using the Eloquent ORM.

Two additional functions are then called to update the user and team rankings, rankUsers() and rankTeams() respectively. Both these functions work in exactly the same manner as outlined in the rankUsers() function in figure 24 to display the users rankings seen within the view in figure 23.

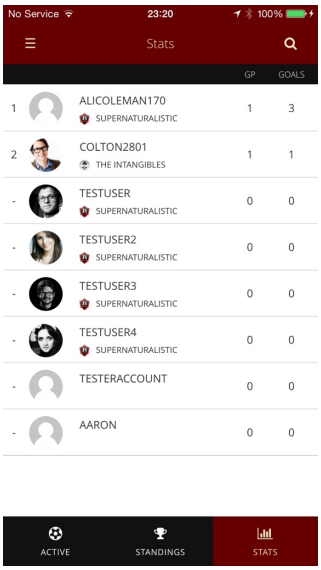


Figure 23 - User Standings View

```php
public function rankUsers()
{

  $users = User::whereNotNull('goals')->orderBy('goals', 'desc')->get();

  $goals = 0;
  $ranking = 1;

  foreach ($users as $index => $user)
  {
    if ($goals !== $user->goals)
    {
      $ranking = $index+1;
      $goals = $user->goals;
    }

    $user->ranking = $ranking;
    $user->save();
  }

}
```

Figure 24 - Method to Rank Users

Firstly all the users are fetched who's goals field is not null, meaning they have taken part in the game and are not just non-participating registered users. These results are returned to the users object variable ordered by the total number of goals they've scored in descending order. A goals variable is then set to 0 and an initial ranking variable to 1 before looping through every user which was returned.

The foreach loop checks if the goals variable is equal to the users goals total and if not gives them the ranking of their index number plus one (because index starts at 0). If the users goals is equal to the current goals variable then it set the users ranking to equal the current

ranking. This allows users to be ranked in joint 1st place or joint 2nd place etc if they have the same number of goals. The users ranking is then saved to the ranking field in the database.

5. TESTING

5.1 Approach

The box testing approach is a popular method of software testing of which there are two main methods – white box and black box.

5.1.1 White Box Testing

The white box testing approach focuses on the internal workings of an application, looking at the structure of the code. (Softwaretestinghelp.com, 2015) It primarily focuses on the security and input and output flows within the application, improving on the usability and design. The name white box testing comes from the ability to see and understand the inner workings of the software. (Williams, 2006) (Guru99, 2015)

5.1.2 Black Box Testing

Black box testing on the other hand is the approach in which the inner workings of the application aren't known, so only the end user experience can be tested. (Guru99, 2015) One advantage of black box testing is that no programming knowledge is required to test the application. The focus lies solely on the input and output of the program and is based entirely on the requirements specification. (Williams, 2006)

5.1.3 Grey Box Testing

There is a third method of testing known as grey box testing which uses a combination of white box testing and black box testing. With this approach the tester has a limited knowledge of the inner working of the application but doesn't require access to the code to test the application. It is well suited to web applications as it allows testing by both the users and developers. (Softwaretestingclass.com, 2012) (Janssen, n.d.)

5.2 Methods

5.2.1 Static Testing

Static testing methods are those which do not see the code within the software execute, but instead involves the examination of the code and documentation. This can include the reviewing of some of the non-functional requirements within the specification. (Wikipedia, 2015)

5.2.2 Dynamic Testing

The opposite of static testing whereby the application is run to test the workings of the code to ensure inputs and responses are as expected. This can be done by means of a manual process or an automated process such as unit testing. Dynamic testing is the method by which the functional requirements of the specification can be tested. (Rouse, 2012) (Tutorialspoint.com, 2015)

Unit Testing

The method of unit testing allows for test cases to be written which allow for the testing of functions or pieces of code. These rest cases can ensure that the code is returning the expected values of responses and ensure errors are handled correctly. It aims to help identify failures with the code or logic early in the development and allows future functionality to be continually tested against to ensure all code is working together as expected. (McFarlin, 2012)

5.2 Process

The process used for the testing of the application seen a grey box testing approach where a limited knowledge was known about the inner workings of the application. All of the methods mentioned were used to test the application against the requirements specification.

5.2.1 Static Testing

Static testing methods can be used to test some of the non-functional requirements from the specification. A number of the non-functional requirements, #24, #25, #28 and #29, depend on user feedback for the application which will be done via a user survey during user testing. This feedback can be reviewed to test against our non-functional requirements in specification. Other non-functional requirements can only be tested by reviewing statistics over time such as the scalability of the application (#33), the minimal downtime (#31), high percentage of returning users (#26) and time spent fixing issues (#37).

5.2.2 Dynamic Testing

All of the functional requirements of the application can be testing using dynamic testing methods. Laravel supports unit testing out of the box through the use of the PHP testing framework PHPUnit allowing test cases to be written for use within our application. (Phpunit.de, 2015) The unit testing within Laravel was used for the user registration and authentication functions of the application (requirements #1 to #5). The remainder of the application was tested manually by using the application and ensuring it responded to all possibilities in the intended manner according to our specification (requirements #6 to #23).

A couple of the non-functional requirements can also be tested at the current time through dynamic testing – page load times (#30) and cross-browser compatibility (#34).

5.3 Results

The unit testing for the creation of user accounts and authentication for the application seen 27 test cases which tested the various functions. These tests ranged from checking the correct views were returned, data validated correctly, correct JSON responses were returned and users were redirected to the correct pages. An example of one of these tests can be seen in figure 25.

```
public function test_ajax_login_requires_email()
{
  $user = $this->makeUser(['email' => 'test@test.com','password' => Hash::make('test'),
  'password_confirmation' => Hash::make('test')]);

  $this->addUser($user);

  $response = $this->call('POST', 'auth/login', ['email' => '','password' => 'test'], [], [],
  ['HTTP_X-Requested-With' => 'XMLHttpRequest']);

  $json = json_decode($response->getContent());

  $this->assertResponseStatus(422);
  $this->assertObjectHasAttribute('email', $json);
}
```

Figure 25 - Test to Check AJAX Login Requires Email

The test first creates a user and stores them to the test database in memory. It then calls a 'POST' request to the 'auth/login' url via AJAX and stores the response. The assert methods are then used to check that the response is as intended in that a 422 status is returned and the JSON returned has an attribute of email (this indicates an error with the email).

The initial test returned 4 failed assertions as seen in figure 26, these all related to the page redirecting to the wrong url upon logging in. This was corrected by changing the redirect path within the AuthController and, as can be seen in figure 27, all the tests then successfully passed.

```
FAILURES!
Tests: 27, Assertions: 61, Failures: 4
```

Figure 26 - Initial Tests Failed

```
OK (27 tests, 62 assertions)
```

Figure 27 - Tests Passed

The tests allowed requirements #1 to #5 to be marked at met. The remainder of the requirements needed manual testing to ensure they had been met. Table 3 shows the list of requirements that were tested and the outcome of these tests.

The results of the testing met the majority of the high priority functional requirements. The only issue encountered was problem with users being able to successfully join a team in

which an error was returned. This was corrected by successfully finding the code causing the error and updating it to ensure users could now successfully join a team.

Table 3 - Functional Requirement Testing

ID	Priority	Fit Criterion	Outcome	Response if Failed
1	H	Users are able to successfully submit a register form.	Passed	
2	H	Users are able to successfully submit a log in form.	Passed	
3	H	User data validates and registration successful or error	Passed	
4	H	User data validates and login successful or error returned.	Passed	
5	H	User can successfully reset their password.	Passed	
6	H	User can successfully create a team.	Passed	
7	H	Team validates or error is returned.	Passed	
8	H	User can successfully perform a team search.	Passed	
9	H	User can successfully join a team.	Failed	Error fixed to requirement passes.
10	H	The administrator can successfully submit 5 matches.	Passed	
11	H	The application can successfully set up a league.	Failed	None - future development
12	H	Every member of a team is correctly give a match to predict.	Passed	
13	H	Users are able to successfully submit a prediction.	Passed	
14	H	Users prediction validates or error returned.	Passed	
15	H	Prediction scores are calculated correctly.	Passed	
16	H	User scores are calculated correctly.	Passed	
17	H	Team scores are calculated correctly.	Passed	
18	H	Leaderboards rank teams correctly based on their score.	Passed	
19	L	Users can successfully submit a team comment.	Failed	None - future development
20	L	Users can successfully submit a match comment.	Failed	None - future development
21	L	Comment validates or error returned.	Failed	None - future development
22	L	Users can successfully send a message to another user.	Failed	None - future development
23	L	No other users can access other users messages.	Failed	None - future development

The additional non-functional tests, #30 and #34 were also tested to ensure they met the requirements of the application. The results of which can be seen in tables 4 & 5.

Table 4 - Browser Testing Results

Browser	Outcome	Response if Failed
Safari	Passed	
Chrome	Passed	
Firefox	Passed	
Internet Explorer	Passed	Slight SVG issue was resolved
Safari Mobile (iOS)	Passed	
Chrome Mobile (iOS)	Passed	
Chrome Mobile (Andriod)	Passed	

The browser testing involved loading the application and using it to perform all the common actions within the application to ensure they worked correctly on the latest versions of the listed browsers. As can be seen from table 4 all worked correctly barring a slight issue with SVG's not scaling properly within Internet Explorer, however this didn't affect the functionality of the application and was rectified by swapping the SVG out for a PNG.

Table 5- Page Load Time Testing Results

URL	Uncached	Cached
/	1363ms	786ms
/profile	1483ms	867ms
/match/{id}/predict	1039ms	843ms
/standings	1000ms	576ms
/stats	1434ms	693ms
/team/{id}	1048ms	841ms

The page load time tests were carried out on the production server within the Chrome browser using the '*Average Load Time Tester'* extension. (Chrome.google.com, 2015) The results of the testing across various pages within the application can be seen in table 5 for

both the initial load and then the reloading of the page with resources cached. Each page was loaded 10 times with the average taken.

As we can see all the pages currently load in under 2 seconds even when uncached, and all load in under 1 second when cached. As the application expands and grows these load times may increase a little but there are further measures that can be taken to increase load times due to time constraints haven't been taken to date in the project. At this stage however it can be said that requirement #34 has been met.

5.4 User Testing

A number of the requirements depended on user feedback for the application which could be carried out in the form of a survey. This survey was created on *'SurveyMonkey'* , an online survey and questionnaire tool which allows the creation of surveys to be completed online and results automatically compiled into graphs and figures. (Surveymonkey.com, 2015) The survey was distributed to 10 people who were asked to use the application and complete the survey with their initial feedback.

The questions on the survey aimed to test requirements #24 and #28. Each of the 5 questions provided multiple choice answers and the results would help ascertain if the requirements had been met. The questions asked on the survey were:

- How visually appealing is the TTPL Web Application?
- How convenient is the TTPL web application to use?
- How easy is it to understand the information on TTPL web application?
- Overall, are you satisfied with your experience using the TTPL web application, dissatisfied with it, or neither satisfied or dissatisfied with it?
- If the TTPL prediction game were available today, how likely would you be to recommend it to others?

The results of the survey can be seen within the evaluation section of this report.

6. EVALUATION

6.1 Survey Results

The results of the survey provided useful feedback for the application and allowed for the testing of some of the non-functional requirements. All 10 users asked to complete the survey done so and their results can be seen below.

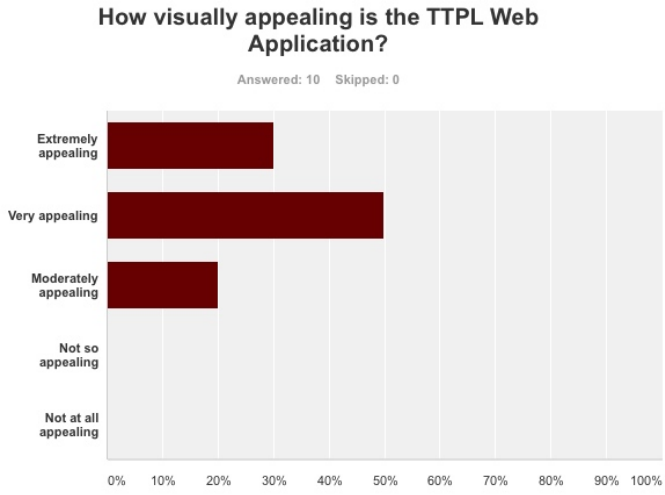


Figure 28 - Survey Question 1 Results

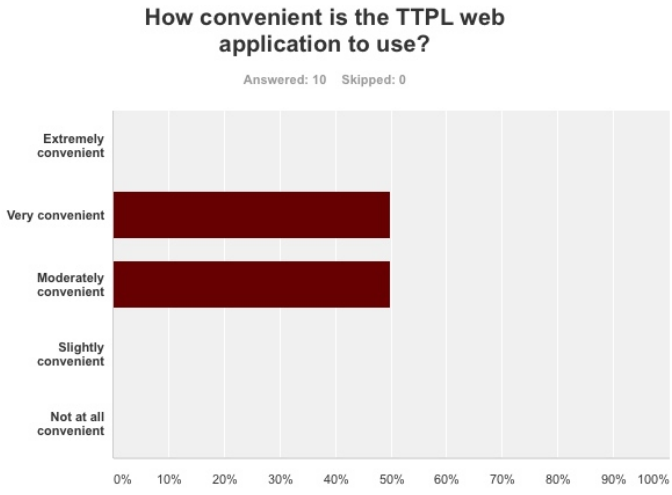


Figure 29 - Survey Question 2 Results

Figure 28 shows that the users in general found the application to be visual appealing with half the users saying it was very appealing and 30% saying it was extremely appealing. The results from his user feedback can be used to determine that requirement #24 was met successfully.

The ease of use of the application was testing using the user feedback from figure 29. This indicated that users found the application convenient to use, although no-one found it extremely convenient. Perhaps indicating that a few of the features could be changed slightly if further feedback was obtained as to why they found it not extremely convenient. Overall though we can say that requirement #28 was met.

The additional questions of the survey seen users happy with their experience in using the application and said they would recommend it to others if the application was available to use today. The only result that raised a few issues was over how easy the information was to understand in the application. A help page would possibly go a long way to helping some of the uses out who found it not so easy as the concept of the game may possibly be hard to grasp at first. The remaining results of the survey can be seen in Appendix F.

6.2 Project Outcomes

The development of the project seen the creation of a web application in which users can come together as teams (friends or otherwise) to predict real world football matches and compete against other teams. The web application built specifically for mobile can be accessed through a mobile web browser and saved to a users home screen where it can then mimic an application (figure 30).

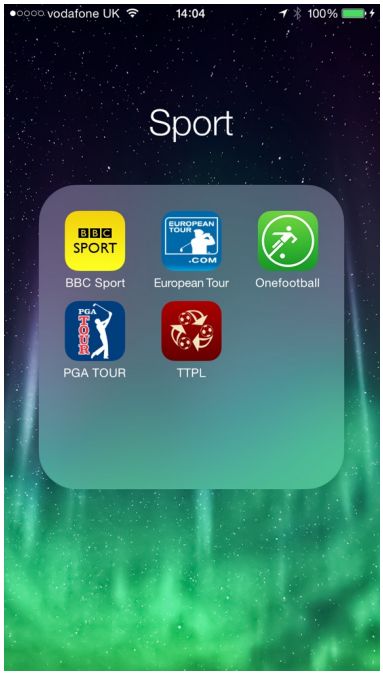


Figure 30 - Application Icon

Users have the ability to sign up and log in to the application, join or create teams and take part in the game. Private teams allow friends to play together safe in the knowledge that no random users can join their team while public teams ensure anyone can play the game as part of a team. The team captain (creator of the team) has the ability to assign matches every matchday for their team to predict. Users can then predict solely on the match they have been assigned.

Administrators of the application have the ability to access the back end of the application where they can manage upcoming matchdays, creating the fixtures to be predicted and the entering the results upon completion. They are also able to view various statistics about the current match day in question.

The algorithms within the game automatically calculate the scores and rank users based on the scoring system within the game. Users can then view the rankings of both teams and users from within the application.

The outcome of the project managed to meet all but one of the high priory requirements of the project due to the time constraints and unforeseen circumstances. The iterative and incremental development methodology in building the application however meant that despite not meeting the requirement the application is still in a working state and the project still meets its aim.

7. CONCLUSION

7.1 Summary

The creation of the TTPL web application brought with it many fun and exciting challenges that required new learning and progression of existing knowledge to design and develop the application. The process of going through the UX and system design as outlined in this report made the development of the application much easier when it came to the implementation as there was a clear plan outlined to its development. The use of Laravel in particular brought with it many challenges however now, having been used throughout the project, an understanding has been developed and skills within PHP enhanced in the respect of the object orientated approach to programming.

7.2 Reflection

Overall the development of the project went smoothly. The plan that was outlined for the project through the gantt chart, and the subsequent management of the tasks involved through the use of 'TeamWork PM', proved helpful throughout the development of the project to try and keep on top of the development.

However, as outlined in the risks, unforeseen and exceptional circumstances posed major risks to the schedule of the project and despite additional time being allowed for these, they still caused the scope of the project to be narrowed. The main cause for unforeseen circumstances was the work involved in running a business, Amigo Studios, outside of university and clients requiring urgent work done. (Amigo Studios, 2015) All work was stopped at the beginning of April with the exception of urgent bug fixes and server issues however more time was required and stopping all work at the beginning of March would maybe have been a better idea to get the full intended scope of the project completed.

The use of the iterative and incremental development methodology proved to be a good methodology to use. Being able to develop small pieces of the project then build and enhance those while having everything working at each stage was a great way to go about the development. This had its advantages over attempting to build a full component or the whole application then attempt to test and debug it.

Having the full control over the development of the project meant being in charge of complete design of the application. Despite being more of a developer, the design of the application is clean and modern, managing to display a lot of information in confined space simply and clearly. The code within the application can be refactored in places to make better use of some programming principals, such as DRY (Don't Repeat Yourself) and the single responsibility principle where each function should perform only a single task. (Programmer. 97things.oreilly.com, 2015) (Diggins, 2015) The code however does what it is set out to do and the use of Laravel made a lot of the tedious tasks such as SQL queries, form requests and authentication a breeze.

7.3 Future

In the future it is hoped that the application will be developed further to integrate the requirements that haven't been met and incorporate pages that would further enhance the application such as user profiles and further administrative functions. Once the application is developed to a working standard the idea is that is could be released to a small community for beta testing with a view to releasing it to the public. Further down the line, an API could be created on the back end which could then be used to build a more seamless application and be the base for building native apps and a desktop version of the application.

8. REFERENCES

1. Agilemethodology.org, (2015). *What is the Agile Software Development Methodology?*. [online] Available at: http://agilemethodology.org [Accessed 1 May 2015].
2. Amigo Studios, (2015). Amigo Studios - Web Design & Development, Graphic Design, Digital Marketing, Photography, Video Production in Belfast, Northern Ireland. [online] Available at: http://amigostudios.co [Accessed 1 May 2015].
3. Billings, A. and Ruihley, B. (2013). *The Fantasy Sport Industry*. Hoboken: Taylor and Francis.
4. Chrome.google.com, (2015). *Average Load Time Tester*. [online] Available at: https://chrome.google.com/webstore/detail/average-load-time-tester/pflmhongacoepfhkgidighbmppahnljh [Accessed 1 May 2015].
5. Diggins, C. (2015). The Principles of Good Programming. [online] Artima.com. Available at: http://www.artima.com/weblogs/viewpost.jsp?thread=331531 [Accessed 1 May 2015].
6. Digitalocean.com, (2015). *SSD Cloud Server, VPS Server, Simple Cloud Hosting | DigitalOcean*. [online] Available at: https://www.digitalocean.com [Accessed 30 Apr. 2015].
7. Edition.cnn.com, (2010). *The rise and rise of fantasy sports - CNN.com*. [online] Available at: http://edition.cnn.com/2010/SPORT/football/01/06/fantasy.football.moneyball.sabermetrics/ [Accessed 30 Apr. 2015].
8. Ehsan, M. (n.d.). *Architecture of Laravel Applications - Laravel Book*. [online] Laravelbook.com. Available at: http://laravelbook.com/laravel-architecture/ [Accessed 30 Apr. 2015].
9. En.wikibooks.org, (n.d.). *PHP Programming - Wikibooks, open books for an open world*. [online] Available at: http://en.wikibooks.org/wiki/PHP_Programming [Accessed 30 Apr. 2015].
10. FIFA.com, (2015). *Big Count*. [online] Available at: http://www.fifa.com/worldfootball/bigcount/ [Accessed 30 Apr. 2015].

11. Fsta.org, (n.d.). *Industry Demographics - Fantasy Sports Trade Association*. [online] Available at: http://www.fsta.org/?page=Demographics [Accessed 30 Apr. 2015].
12. Getcomposer.org, (2015). *Composer*. [online] Available at: https://getcomposer.org [Accessed 30 Apr. 2015].
13. GitHub, (2015). *laracasts/Laravel-5-Generators-Extended*. [online] Available at: https://github.com/laracasts/Laravel-5-Generators-Extended [Accessed 30 Apr. 2015].
14. Gulpjs.com, (2015). *gulp.js - the streaming build system*. [online] Available at: http://gulpjs.com [Accessed 30 Apr. 2015].
15. Guru99, (2015). *What is Black Box Testing?*. [online] Available at: http://www.guru99.com/black-box-testing.html [Accessed 1 May 2015].
16. Guru99, (2015). *White Box Testing - Ultimate Guide*. [online] Available at: http://www.guru99.com/white-box-testing.html [Accessed 1 May 2015].
17. Istqbexamcertification.com, (n.d.). *What is Agile model – advantages, disadvantages and when to use it?*. [online] Available at: http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 30 Apr. 2015].
18. Istqbexamcertification.com, (n.d.). *What is Prototype model- advantages, disadvantages and when to use it?*. [online] Available at: http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 30 Apr. 2015].
19. Istqbexamcertification.com, (n.d.). *What is Waterfall model- advantages, disadvantages and when to use it?*. [online] Available at: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 30 Apr. 2015].
20. Janssen, C. (n.d.). *What is Gray Box Testing? - Definition from Techopedia*. [online] Techopedia.com. Available at: http://www.techopedia.com/definition/16628/gray-box-testing [Accessed 1 May 2015].
21. Knight, K. (2011). *Responsive Web Design: What It Is and How To Use It - Smashing Magazine*. [online] Smashing Magazine. Available at: http://www.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/ [Accessed 30 Apr. 2015].

22. Laracasts.com, (2015). *The Best Laravel and PHP Screencasts*. [online] Available at: https://laracasts.com [Accessed 30 Apr. 2015].
23. Mack, J. (2014). *Five Advantages & Disadvantages Of MySQL*. [online] Datarealm. Available at: https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/ [Accessed 30 Apr. 2015].
24. Marcotte, E. (2010). *Responsive Web Design*. [online] Alistapart.com. Available at: http://alistapart.com/article/responsive-web-design [Accessed 30 Apr. 2015].
25. Mark Otto, a. (2015). *Bootstrap · The world's most popular mobile-first and responsive front-end framework.*. [online] Getbootstrap.com. Available at: http://getbootstrap.com [Accessed 30 Apr. 2015].
26. McFarlin, T. (2012). *The Beginner's Guide to Unit Testing: What Is Unit Testing? - Tuts+ Code Article*. [online] Code Tuts+. Available at: http://code.tutsplus.com/articles/the-beginners-guide-to-unit-testing-what-is-unit-testing--wp-25728 [Accessed 1 May 2015].
27. Mindscapehq.com, (n.d.). *What is Sass? - Mindscape*. [online] Available at: http://www.mindscapehq.com/products/web-workbench/what-is-sass [Accessed 30 Apr. 2015].
28. Mozilla Developer Network, (n.d.). *CSS*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/CSS [Accessed 30 Apr. 2015].
29. Mozilla Developer Network, (n.d.). *HTML (HyperText Markup Language)*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/HTML [Accessed 30 Apr. 2015].
30. Mozilla Developer Network, (n.d.). *JavaScript*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript [Accessed 30 Apr. 2015].
31. Otwell, T. (2015). *Artisan CLI - Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: http://laravel.com/docs/5.0/artisan [Accessed 30 Apr. 2015].
32. Otwell, T. (2015). *Eloquent ORM - Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: http://laravel.com/docs/5.0/eloquent [Accessed 30 Apr. 2015].
33. Otwell, T. (2015). *HTTP Middleware - Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: http://laravel.com/docs/5.0/middleware [Accessed 30 Apr. 2015].

34. Otwell, T. (2015). *HTTP Responses - Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: http://laravel.com/docs/5.0/responses [Accessed 30 Apr. 2015].
35. Otwell, T. (2015). *Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: http://laravel.com [Accessed 30 Apr. 2015].
36. Otwell, T. (2015). *Query Builder - Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: http://laravel.com/docs/5.0/queries [Accessed 30 Apr. 2015].
37. Otwell, T. (2015). *Validation - Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: http://laravel.com/docs/5.0/validation [Accessed 30 Apr. 2015].
38. Php.net, (n.d.). *PHP: What is PHP? - Manual*. [online] Available at: http://php.net/manual/en/intro-whatis.php [Accessed 30 Apr. 2015].
39. Phpunit.de, (2015). *PHPUnit – The PHP Testing Framework*. [online] Available at: https://phpunit.de [Accessed 1 May 2015].
40. Programmer.97things.oreilly.com, (2015). *Don't Repeat Yourself - Programmer 97-things*. [online] Available at: http://programmer.97things.oreilly.com/wiki/index.php/Don't_Repeat_Yourself [Accessed 1 May 2015].
41. Robinson, J. (2015). *Understanding the Premier League*. [online] About.com Sports. Available at: http://worldsoccer.about.com/od/soccer101/a/101_Prem.htm [Accessed 1 May 2015].
42. Rouse, M. (2005). *What is Prototyping Model? - Definition from WhatIs.com*. [online] SearchCIO. Available at: http://searchcio.techtarget.com/definition/Prototyping-Model [Accessed 1 May 2015].
43. Rouse, M. (2012). *What is dynamic testing? - Definition from WhatIs.com*. [online] WhatIs.com. Available at: http://whatis.techtarget.com/definition/dynamic-testing [Accessed 1 May 2015].
44. Sass-lang.com, (2015). *Sass: Syntactically Awesome Style Sheets*. [online] Available at: http://sass-lang.com [Accessed 30 Apr. 2015].
45. Segue Technologies, (2013). *Waterfall vs. Agile: Which is the Right Development Methodology for Your Project?*. [online] Available at: http://www.seguetech.com/blog/

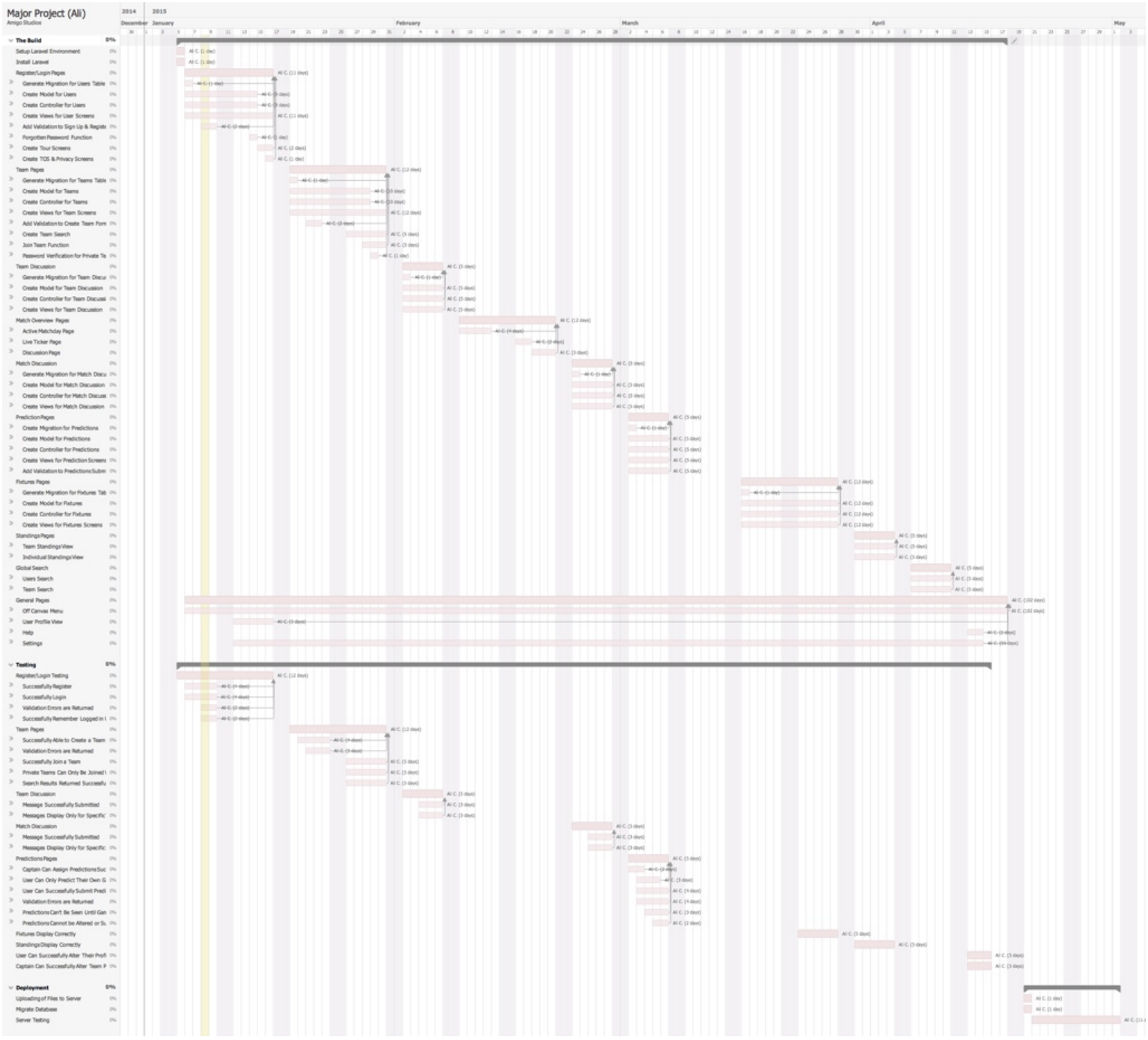
2013/07/05/waterfall-vs-agile-right-development-methodology [Accessed 1 May 2015].

46. Softwaretestingclass.com, (2012). *What is Gray Box Testing? | Software Testing Class | Software Testing Class*. [online] Available at: http://www.softwaretestingclass.com/gray-box-testing/ [Accessed 1 May 2015].
47. Softwaretestinghelp.com, (2015). *White box testing: Need, Skill required and Limitations — Software Testing Help*. [online] Available at: http://www.softwaretestinghelp.com/white-box-testing/ [Accessed 1 May 2015].
48. Stackoverflow.com, (2010). *What is a framework? What does it do? Why do we need a framework*. [online] Available at: http://stackoverflow.com/questions/2964140/what-is-a-framework-what-does-it-do-why-do-we-need-a-framework [Accessed 30 Apr. 2015].
49. Stackoverflow.com, (2015). *What is a LAMP stack?*. [online] Available at: http://stackoverflow.com/questions/10060285/what-is-a-lamp-stack [Accessed 1 May 2015].
50. Surveymonkey.com, (2015). *SurveyMonkey: Free online survey software & questionnaire tool*. [online] Available at: https://www.surveymonkey.com [Accessed 1 May 2015].
51. Teamwork.com, (2015). *Teamwork.com - The Best way to to Manage Your Projects & Team*. [online] Available at: https://www.teamwork.com [Accessed 30 Apr. 2015].
52. Techopedia.com, (n.d.). *What is Iterative and Incremental Development? - Definition from Techopedia*. [online] Available at: http://www.techopedia.com/definition/25895/iterative-and-incremental-development [Accessed 30 Apr. 2015].
53. Topendsports.com, (2015). *World's Most Popular Sports by Fans*. [online] Available at: http://www.topendsports.com/world/lists/popular-sport/fans.htm [Accessed 30 Apr. 2015].
54. Tutorialspoint.com, (2015). *Dynamic Testing*. [online] Available at: http://www.tutorialspoint.com/software_testing_dictionary/dynamic_testing.htm [Accessed 1 May 2015].
55. Volere.co.uk, (2015). *Requirements Specification Template*. [online] Available at: http://www.volere.co.uk/template.htm [Accessed 30 Apr. 2015].

56. W3.org, (n.d.). *Cascading Style Sheets*. [online] Available at: http://www.w3.org/Style/CSS/Overview.en.html [Accessed 30 Apr. 2015].
57. W3schools.com, (n.d.). *Introduction to HTML*. [online] Available at: http://www.w3schools.com/html/html_intro.asp [Accessed 30 Apr. 2015].
58. W3schools.com, (n.d.). *JavaScript Introduction*. [online] Available at: http://www.w3schools.com/js/js_intro.asp [Accessed 30 Apr. 2015].
59. W3schools.com, (n.d.). *PHP 5 Introduction*. [online] Available at: http://www.w3schools.com/php/php_intro.asp [Accessed 30 Apr. 2015].
60. W3schools.com, (n.d.). *PHP: MySQL Database*. [online] Available at: http://www.w3schools.com/php/php_mysql_intro.asp [Accessed 30 Apr. 2015].
61. W3techs.com, (n.d.). *Usage Statistics and Market Share of Server-side Programming Languages for Websites, April 2015*. [online] Available at: http://w3techs.com/technologies/overview/programming_language/all [Accessed 30 Apr. 2015].
62. Way, J. (2012). *Why Laravel is Taking the PHP Community by Storm - Tuts+ Code Tutorial*. [online] Code Tuts+. Available at: http://code.tutsplus.com/tutorials/why-laravel-is-taking-the-php-community-by-storm--pre-52639 [Accessed 30 Apr. 2015].
63. Webopedia.com, (n.d.). *What is Database (DB)? Webopedia*. [online] Available at: http://www.webopedia.com/TERM/D/database.html [Accessed 30 Apr. 2015].
64. Werber, C. (2012). *Fantasy sports finds 'real' foreign markets*. [online] MarketWatch. Available at: http://www.marketwatch.com/story/fantasy-sports-finds-real-foreign-markets-2012-10-16 [Accessed 30 Apr. 2015].
65. Wikipedia, (2015). *Static testing*. [online] Available at: http://en.wikipedia.org/wiki/Static_testing [Accessed 1 May 2015].
66. Williams, L. (2006). *Testing Overview and Black-Box Testing Techniques*. 1st ed. [ebook] Available at: http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf [Accessed 1 May 2015].
67. Williams, L. (2006). *White Box*. 1st ed. [ebook] Available at: http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf [Accessed 1 May 2015].

APPENDICES

Appendix A



Appendix B

Requirements Table

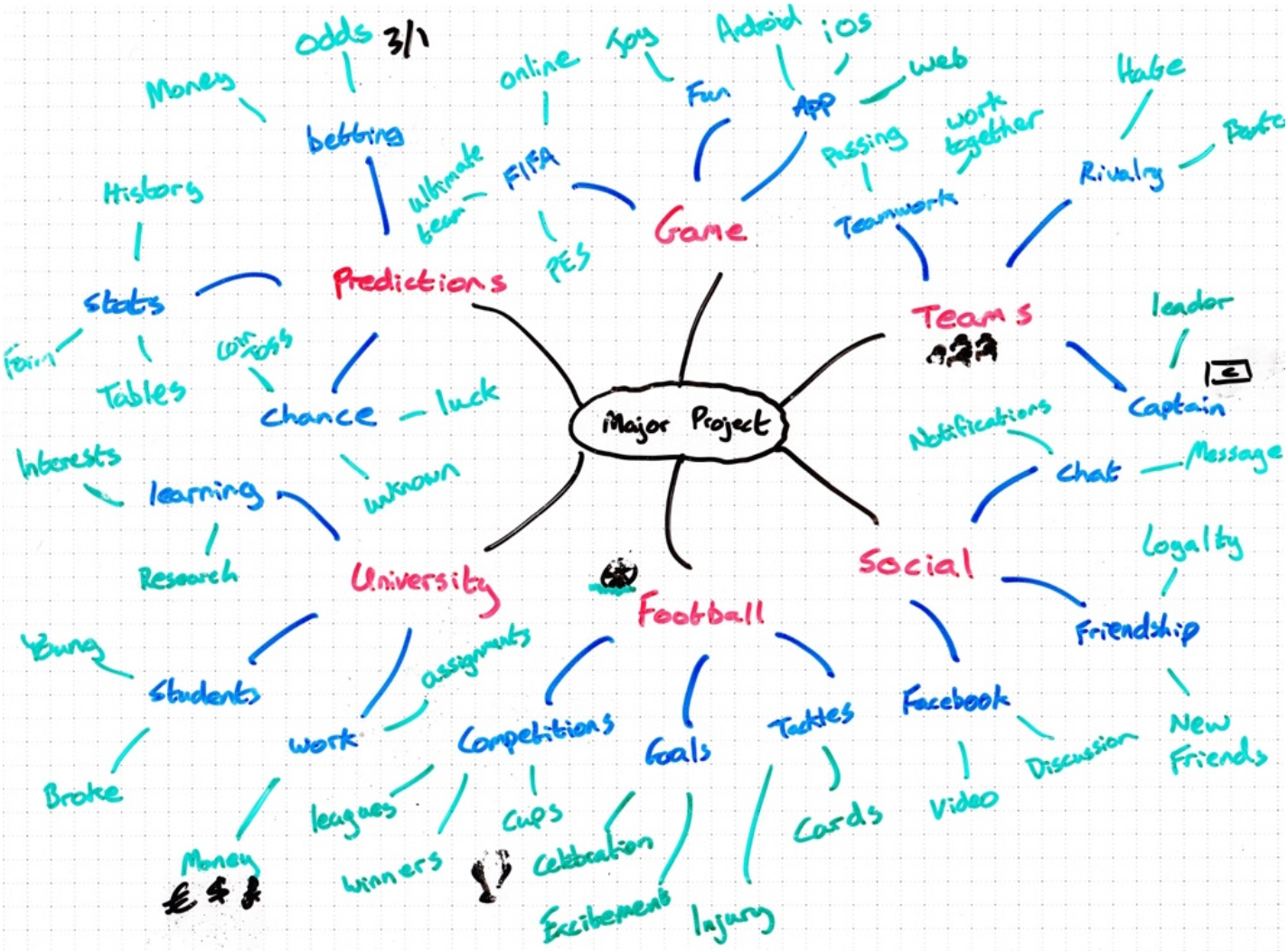
ID	Type	Priority	Description	Rationale	Fit Criterion	Dependencies
1	Functional	H	The user will be prompted to sign up to the application.	Every user must have an account to perform any actions.	Users are able to successfully submit a register form.	
2	Functional	H	The user will be prompted to sign in to the application.	The user must log into the system.	Users are able to successfully submit a log in form.	1
3	Functional	H	The application will validate registration form.	Limit fake or duplicate accounts, ensure valid emails and infomation.	Users data passes all checks and registration successful or error returned.	1
4	Functional	H	The application will validate sign in form.	The application must sign in the correct user based on data input.	Users data passes all checks and login successful or error returned.	2
5	Functional	H	The application will allow a user to retrieve forgotten passwords.	To avoid the loss of users.	User can successfully reset their password.	1
6	Functional	H	The application will allow the user to create a team.	The game needs teams for users to be part of.	User can successfully create a team.	
7	Functional	H	The application must validate new team creations.	The teams must not match other teams and meet certain restrictions.	Team passes all checks or error is returned.	6
8	Functional	H	The application will allow users to search for a team to join.	Users must be part of a team to compete in the team competition.	User can successfully perform a team search.	
9	Functional	H	The application will allow the user to join a team.	If a user finds a team they must be able to become a team member.	User can successfully joins team.	
10	Functional	H	The application must prompt admin to enter matches for each matchday.	Every game needs 5 matches to predict every week.	The administrator can successfully submit 5 matches.	
11	Functional	H	The application must assign fixtures to teams.	Teams need to be part of a league to get games to play.	The application can successfully set up a league.	
12	Functional	H	The application must allow captains to assign matches team mates.	Users must be given a game to assign.	Every member of a team is correctly give a match to predict.	10
13	Functional	H	The application must prompt the user to submit a match prediction.	Every member must be able to predict the game they've been assigned.	Users are able to successfully submit a prediction.	12

14	Functional	H	The application must validate the users prediction.	Predictions must be numbers and not be outrageously big.	Users prediction validates and it added to database or error is returned to user.	13
15	Functional	H	The application must calculate scores once match is over.	The results of the matches need to be known.	Scores are correctly calculated.	
16	Functional	H	The application must calculate in game match score after each match.	The teams need to know how things stand during each match.	Scores are calculated correctly.	
17	Functional	H	The application must calculate the final result of in game matches.	Stats need updating and leaderboards something to rank.	Scores are calculated correctly.	
18	Functional	H	The application must rank teams based on their scores.	Leaderboards needed to rank the teams.	Leaderboards rank teams correctly based on their score.	
19	Functional	L	The application must allow users to chat amongst their team.	Users need to be able to chat to their team.	Users can successfully submit a comment.	
20	Functional	L	The application must allow users to chat to opposition teams.	Users need to be able to enjoy the social side to the game.	Users can successfully submit a comment.	
21	Functional	L	The application must validate comments.	The application must protect against spamming and wumming.	Comment passes all checks or error returned.	19,20
22	Functional	L	The application must allow users to send private messages.	Allows users to chat to each other without other seeing it.	Users can successfully send a message to another user.	
23	Functional	L	The application must keep private messages private and secure.	Private messages are private for a reason.	No other users can access other users messages.	22
24	Non-Functional	H	The application must be attractive to look at.	To help provide a good user experience.	Users feedback that the application looks good.	
25	Non-Functional	H	The application must be easy to use.	Uses must be able to do what they need to provide a good experience.	Users can use the application without help under sampling.	
26	Non-Functional	H	The application must make users want to use it.	The game needs users to return and keep using it.	High percentage of returning users.	
27	Non-Functional	H	The application must be easy to administer.	To make the game as easy to manage and run as possible.	Administrators spend very little time using the backend system.	
28	Non-Functional	H	The application must be able to be used by users with no training.	Users want to be able to just pick up and play.	Feedback from first time users is that it was easy to use.	
29	Non-Functional	H	The application shall use icons that are easily recognisable to the users.	The application can't be complicated for the user to use.	Users never have to ask what icons mean.	

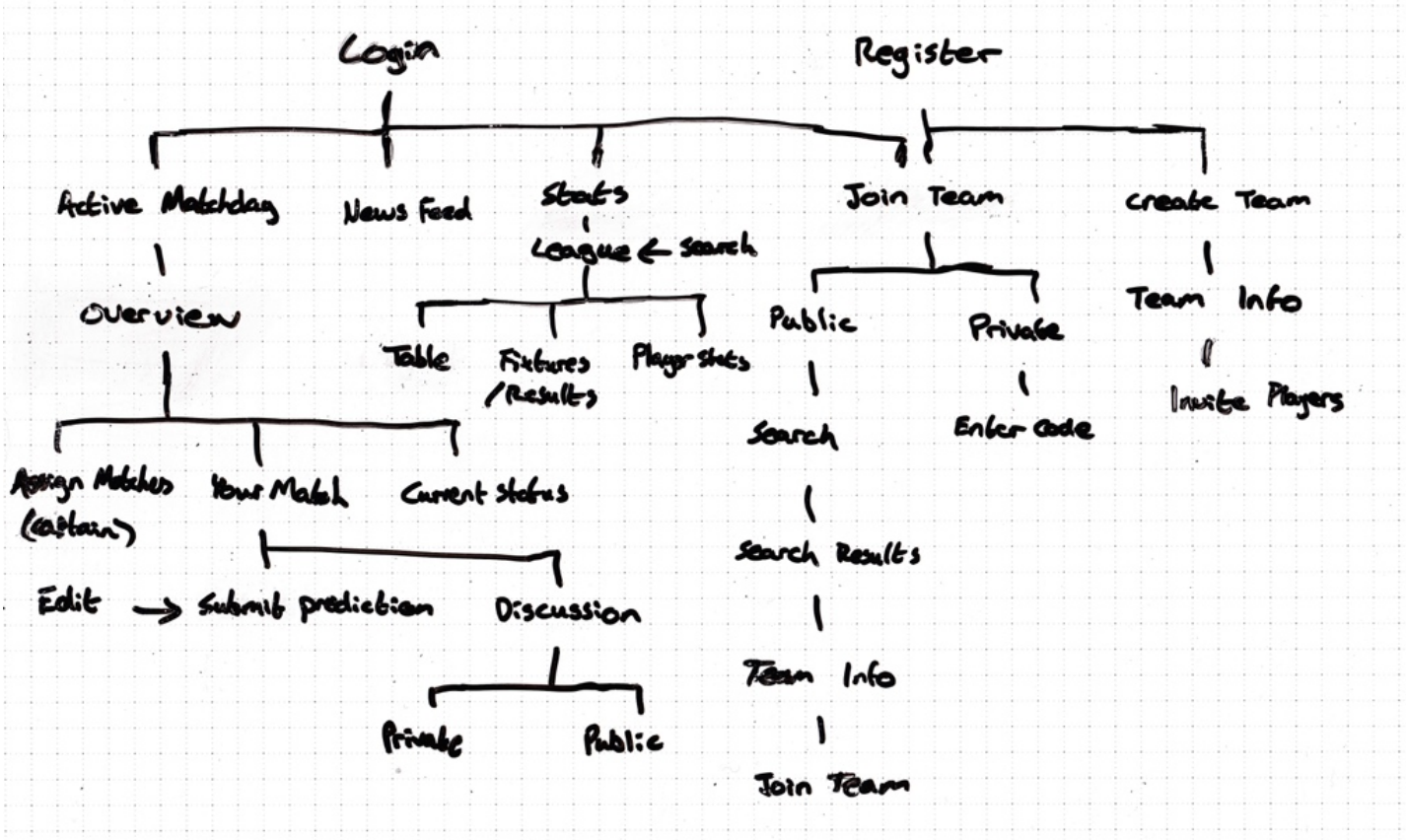
30	Non-Functional	M	The application must be fast enough to avoid the user having to wait.	Slow load times impact negatively on user satisfaction.	Load times of all pages must be less than 2 seconds.	
31	Non-Functional	L	The application must be always available to use.	Unreliable applications lose the trust of users.	Minimum of 99% uptime.	
32	Non-Functional	L	The application must be able to handle large number of simultaneous users.	Multiple users need to be using the application at once.	Multiple users are able to operate the application at one time.	
33	Non-Functional	L	The application must be able to scale to 100,000 users.	There has to be the capacity for future growth.	Increasing numbers of users see no issues with the system.	
34	Non-Functional	H	The application will be fully functional on the latest versions of major browsers.	Not everyone uses the same browser.	Application works on all the latest major browsers.	
35	Non-Functional	H	The product all protect all user data.	May be private information shared.	Thorough testing sees no data leaks.	
36	Non-Functional	H	The application must be easily installed on the server and be transferable.	Has to be easy to run, maintainable and upgradable.	Setup is smooth.	
37	Non-Functional	L	The application must be easily maintained.	To reduce the amount of downtime if issues occur.	Time spent fixing problems is minimal.	
38	Non-Functional	M	The application must backup data periodically.	Incase something goes wrong not everything is lost.	Backups are automatically saved every 24hrs.	

Appendix C

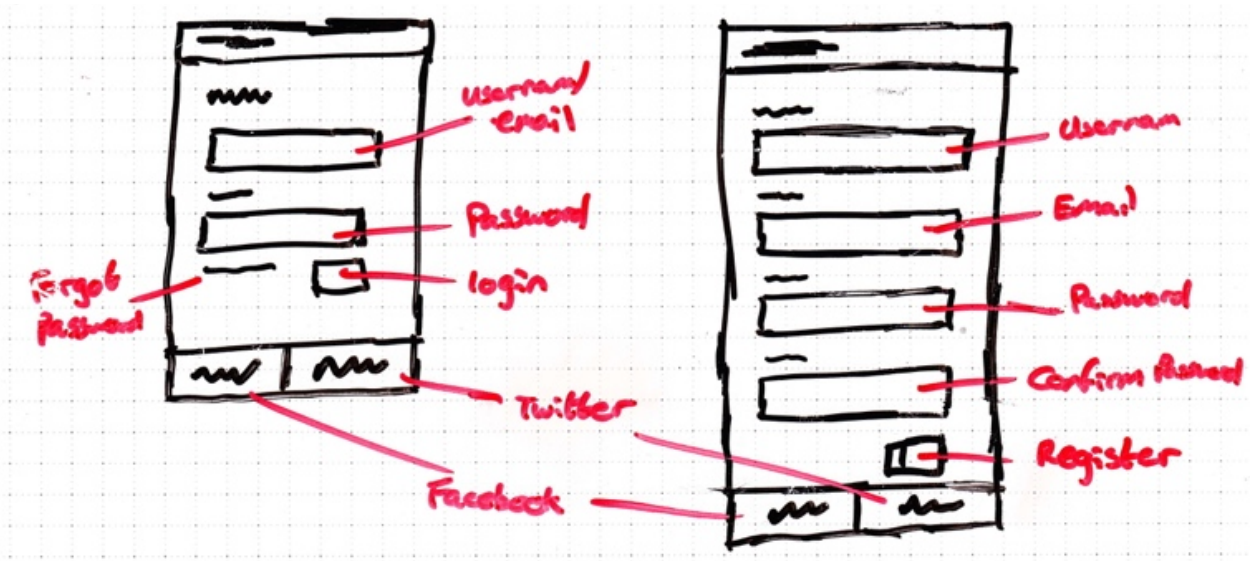
Mind Map



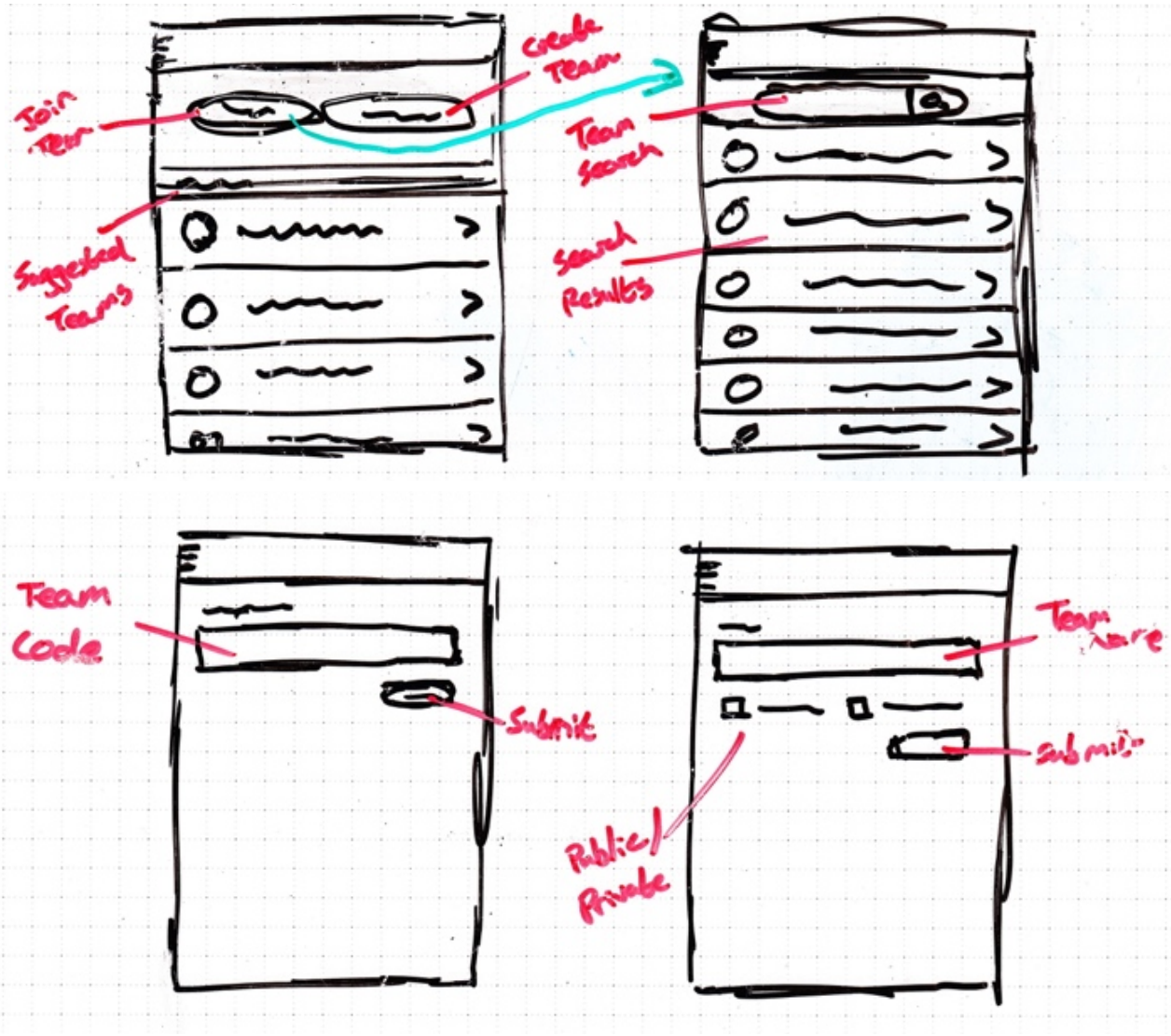
User Journey



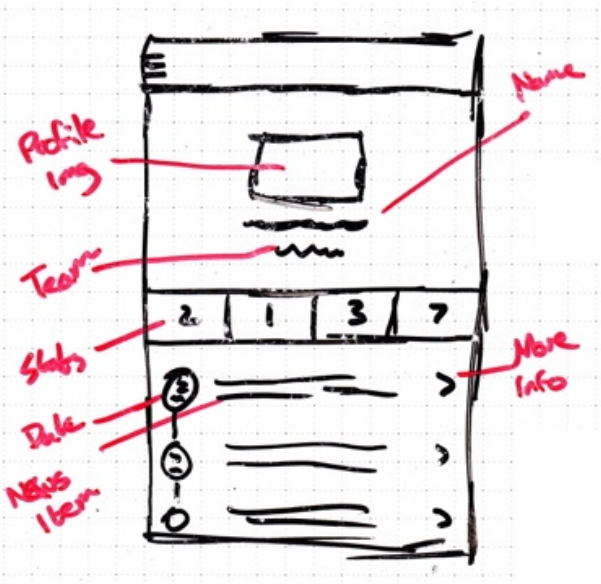
Login With Social Login Options



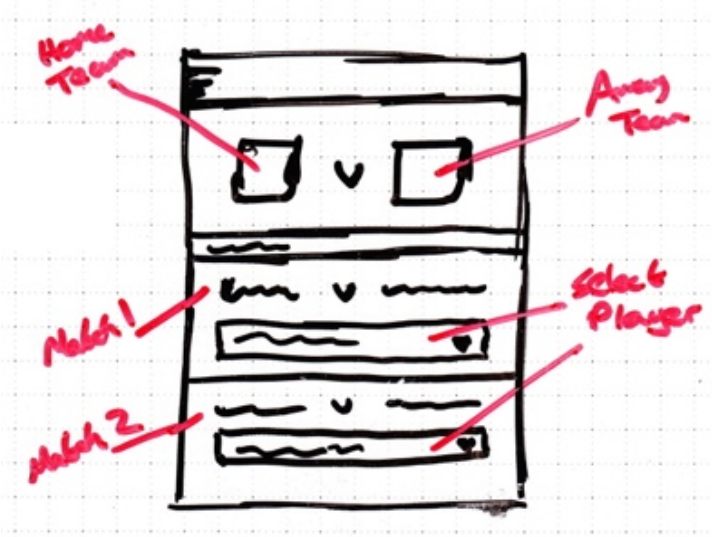
Create/Join Team Screens



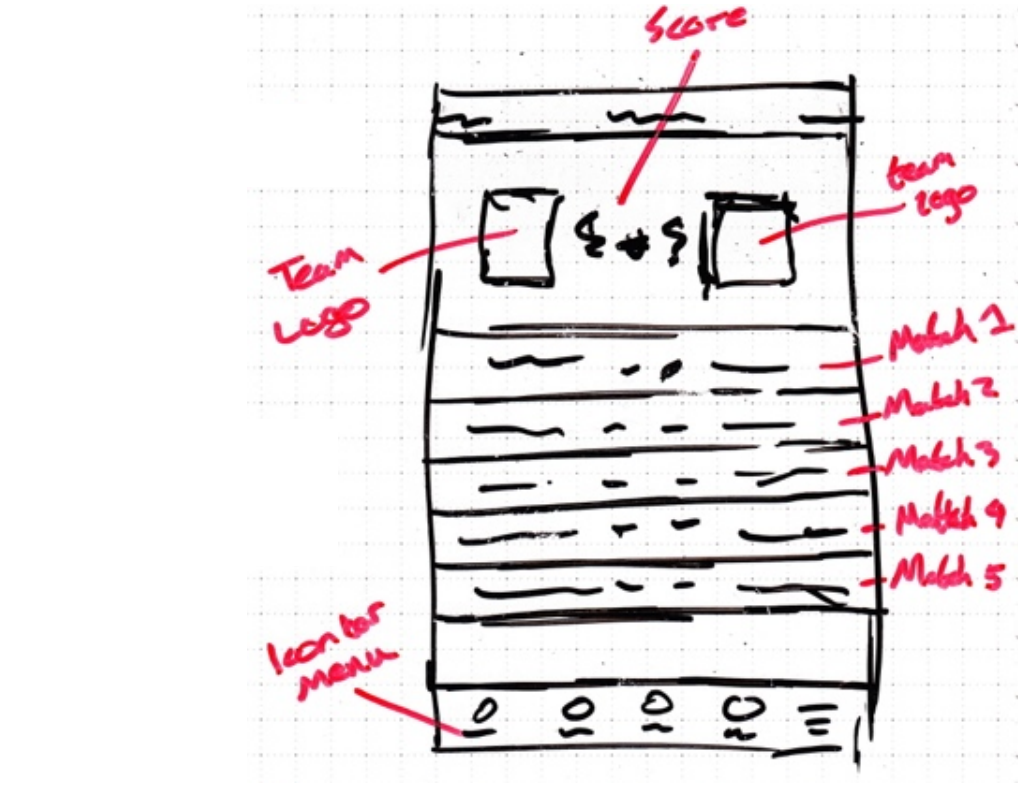
Profile/Team Pages

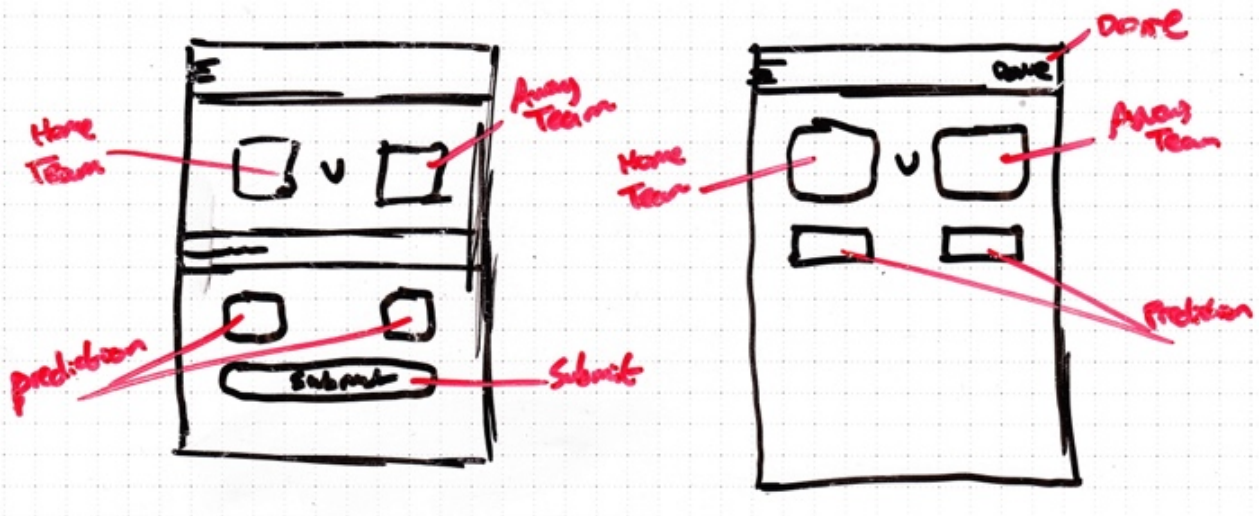


Captain Match Assignment

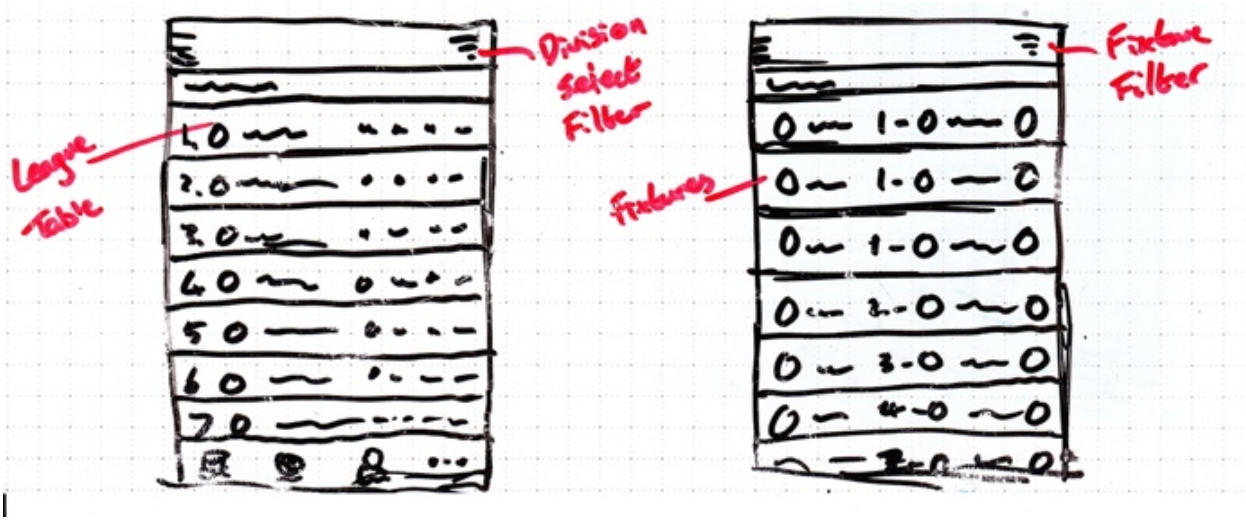


User Predictions

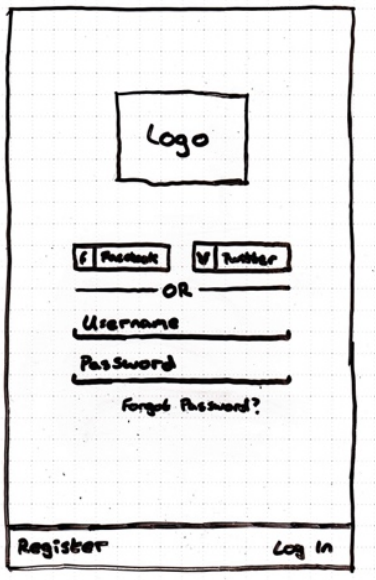




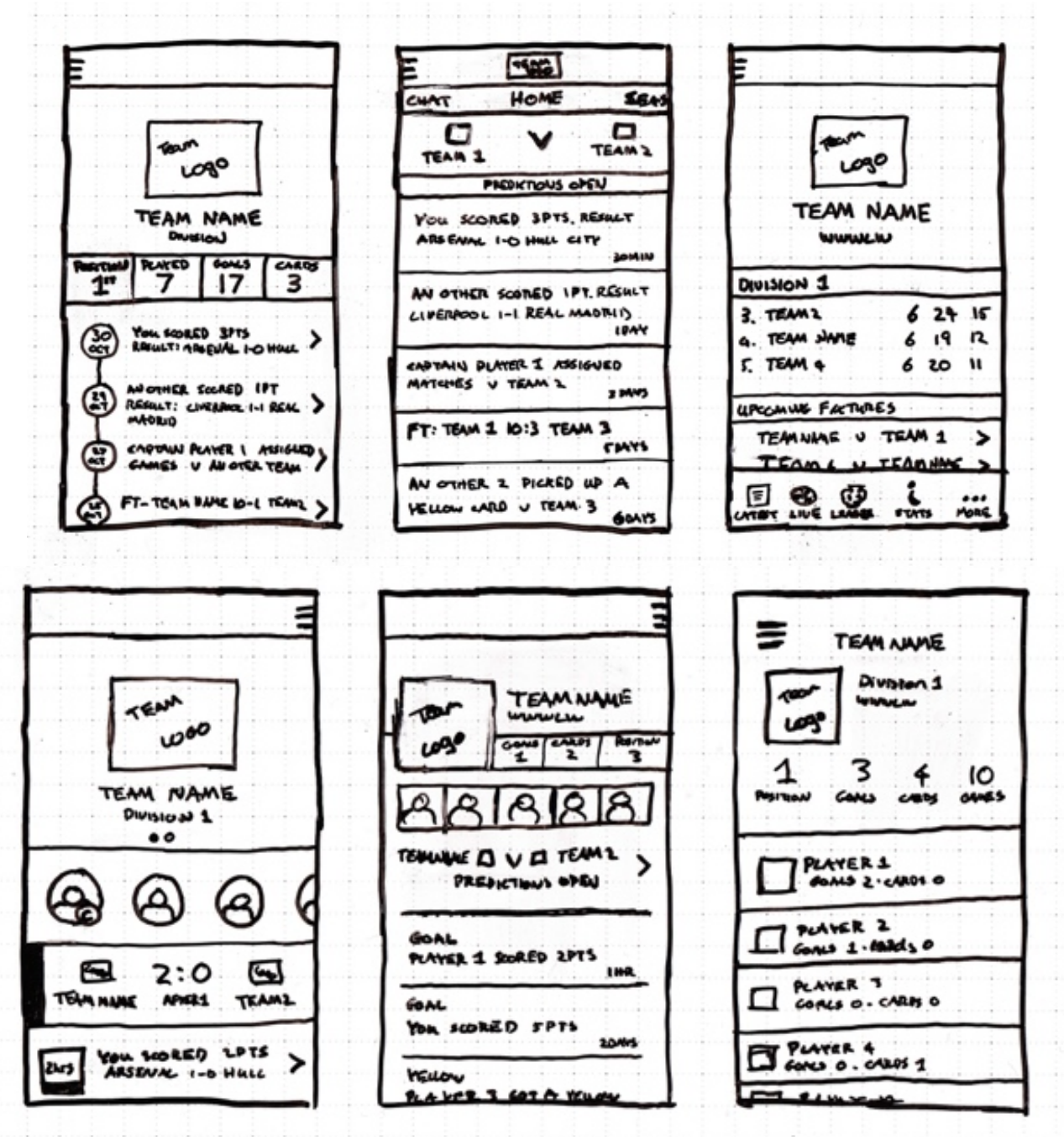
League Tables/Fixtures



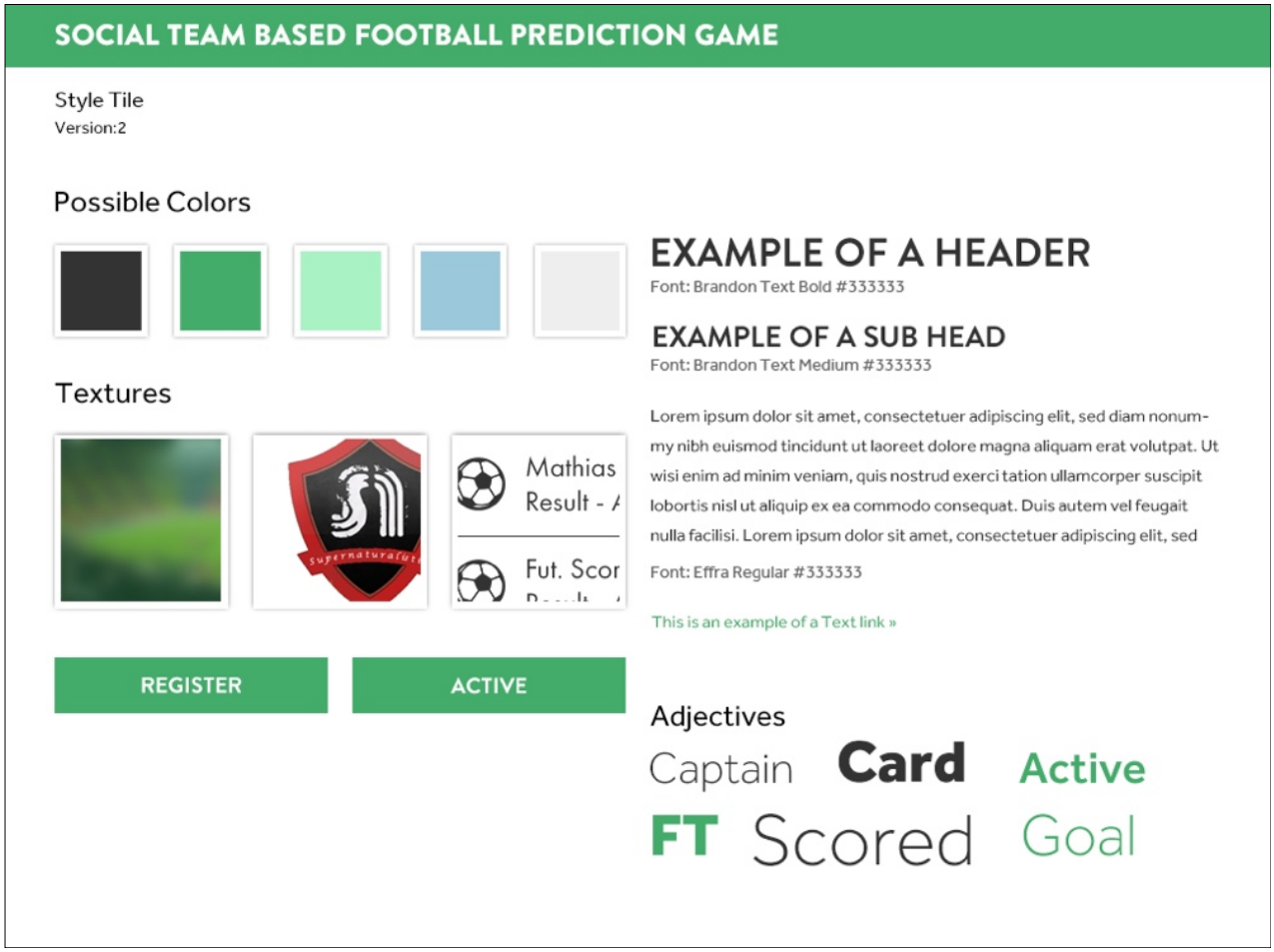
Refined Login



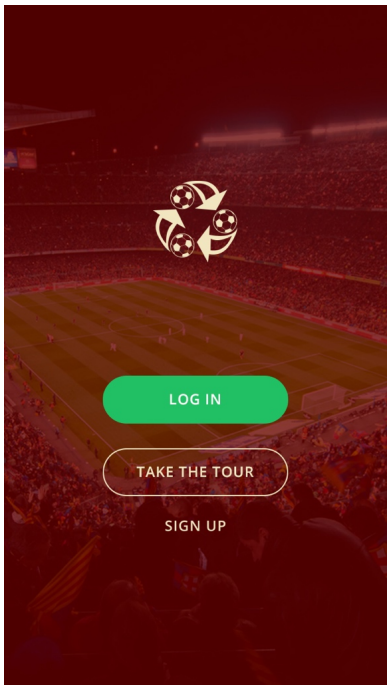
6-Up


TEAM NAME
DIVISION
PLAYED
GOALS
CARDS
7
17
3
CHAT
HOME
TEAM 1
TEAM 2
PREDICTIONS OPEN
DIVISION 1
UPCOMING FIXTURES
TEAM NAME
TEAM
LOGO
2:0
PLAYER 1
PLAYER 2
PLAYER 3
PLAYER 4

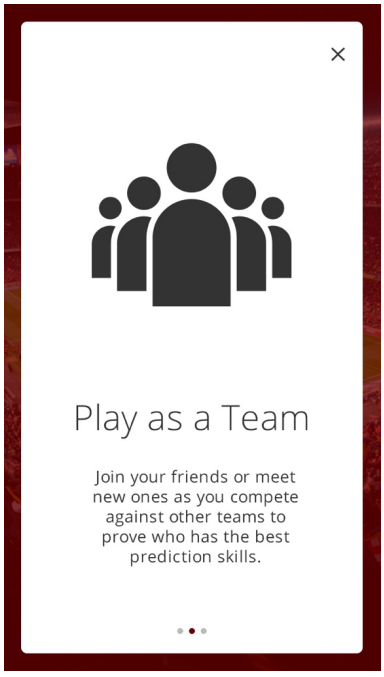
Alternative Style Tile

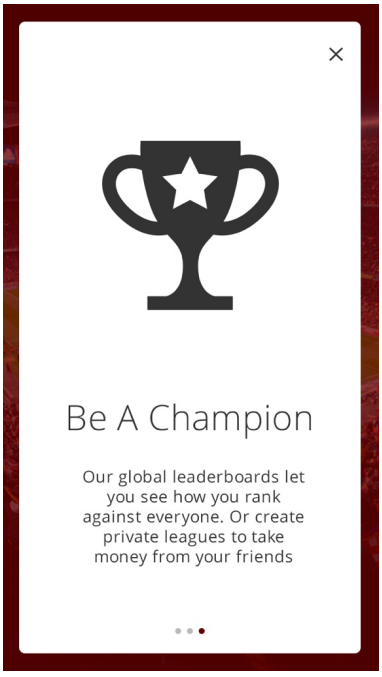


Home Screen Visuals

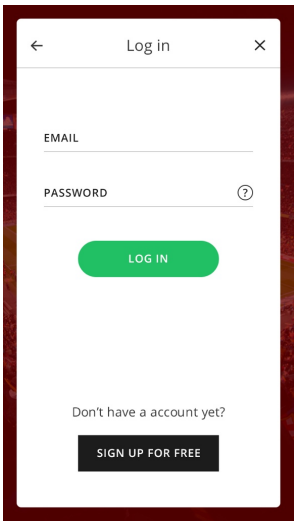


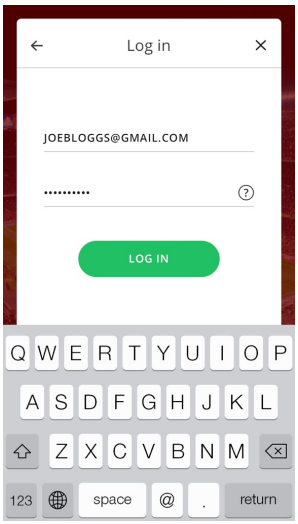
Tour Visuals

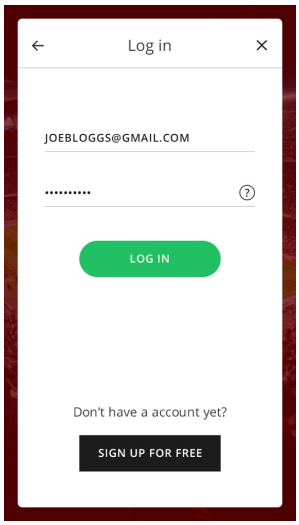


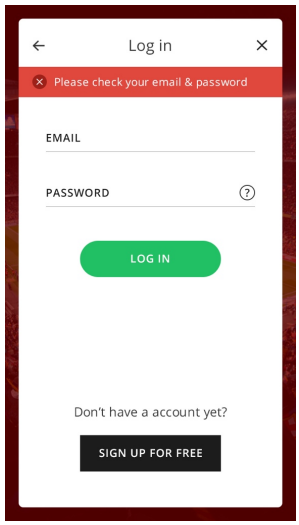


Login Visuals

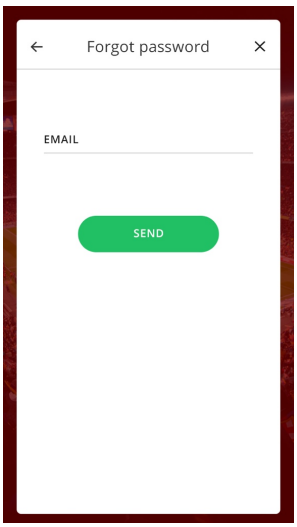


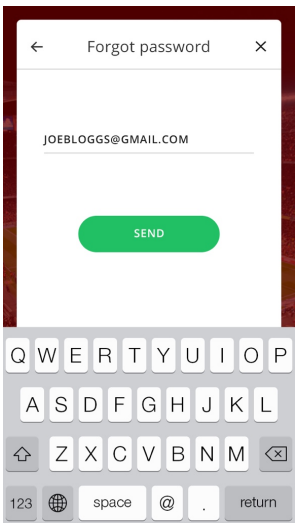


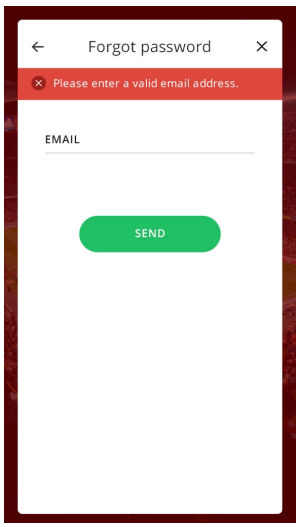


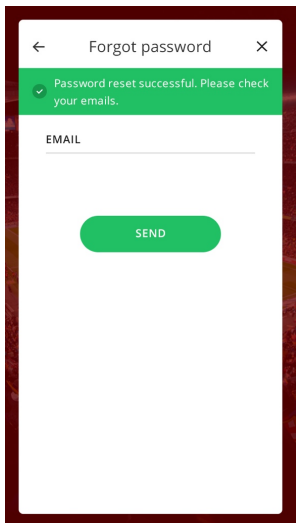


Forgot Password

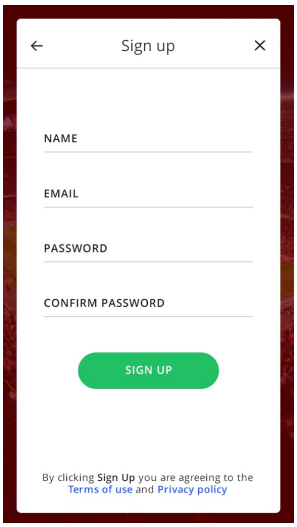


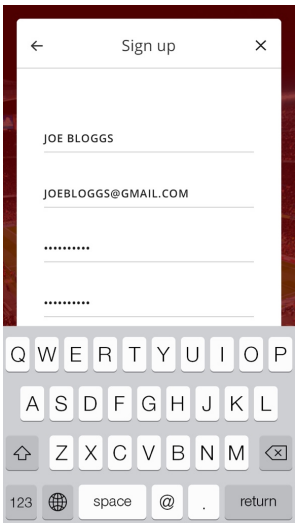


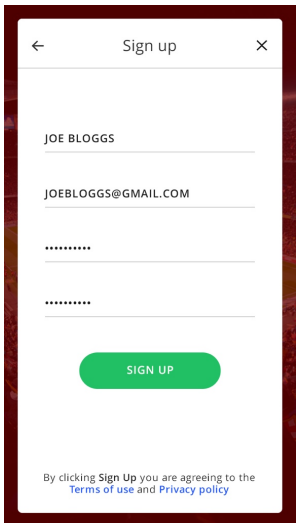


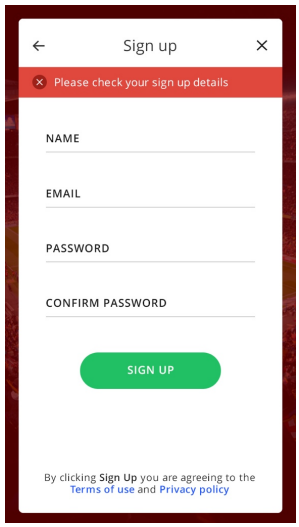


Sign Up Visuals

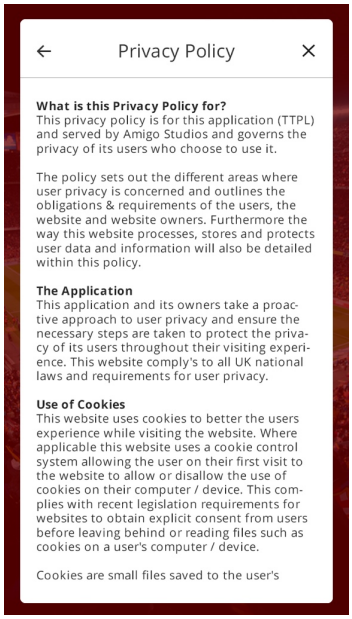


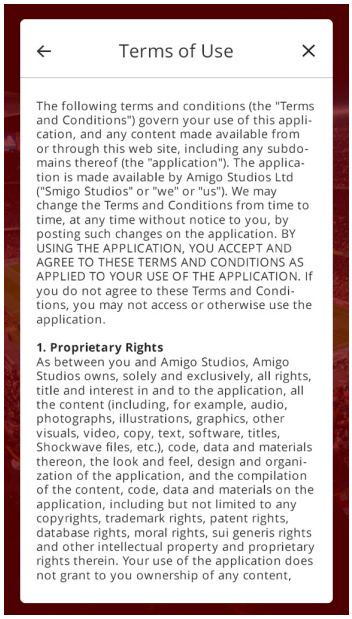




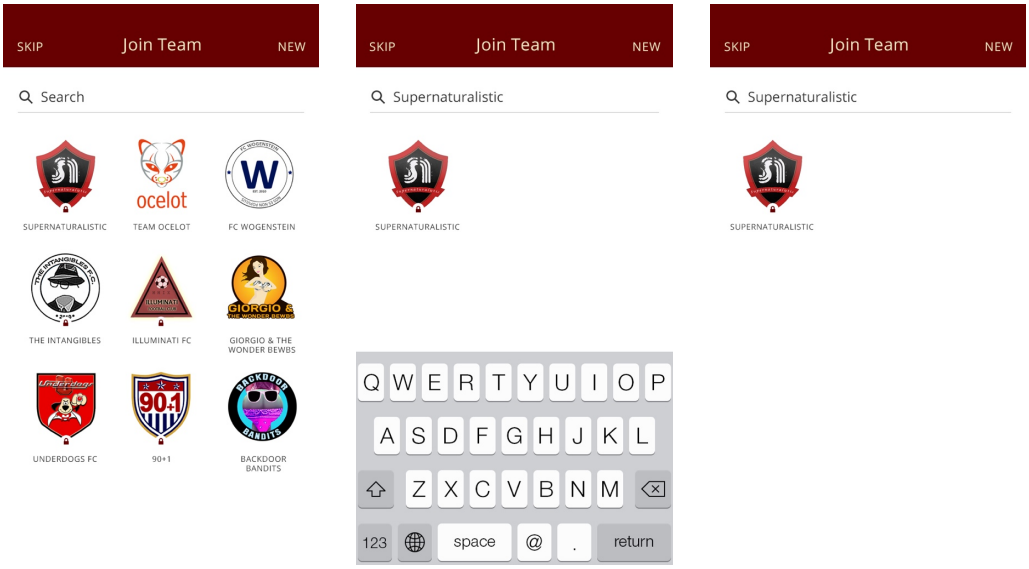


Terms of Use & Privacy Policy Visuals

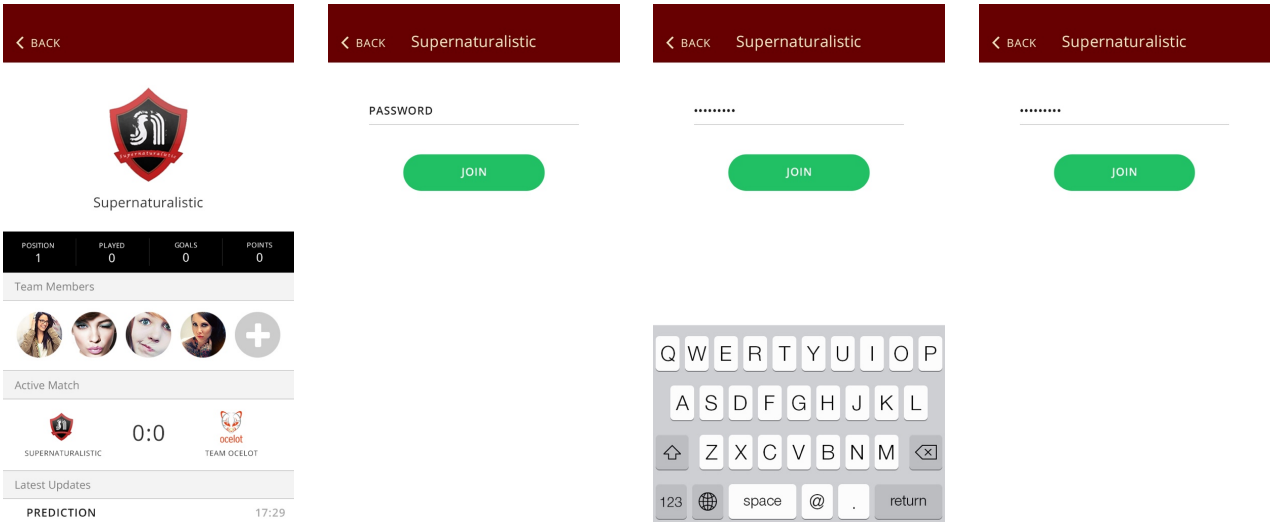




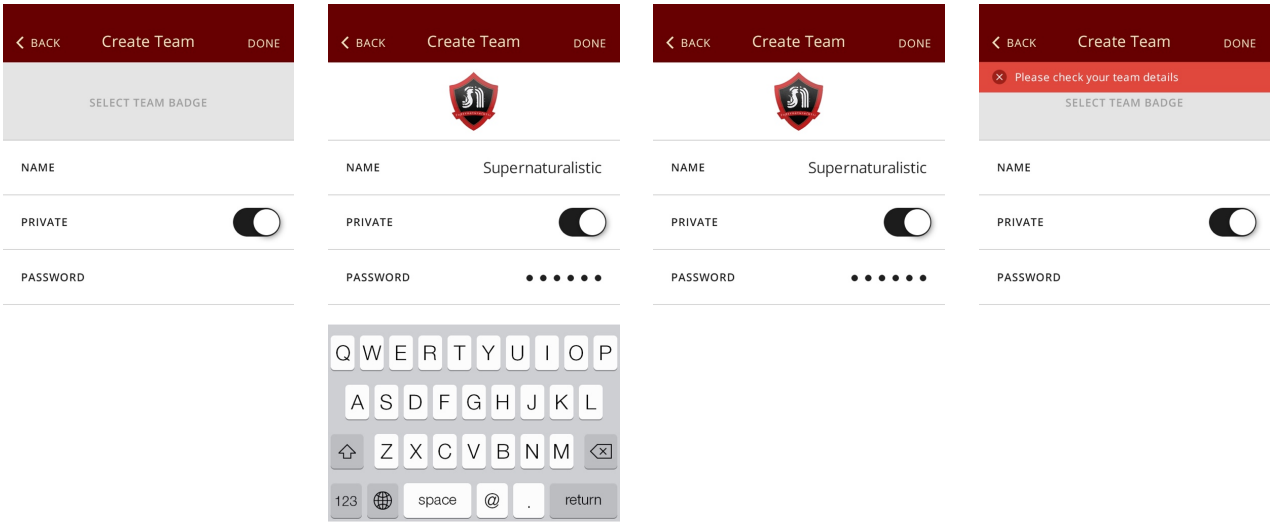
Team Search



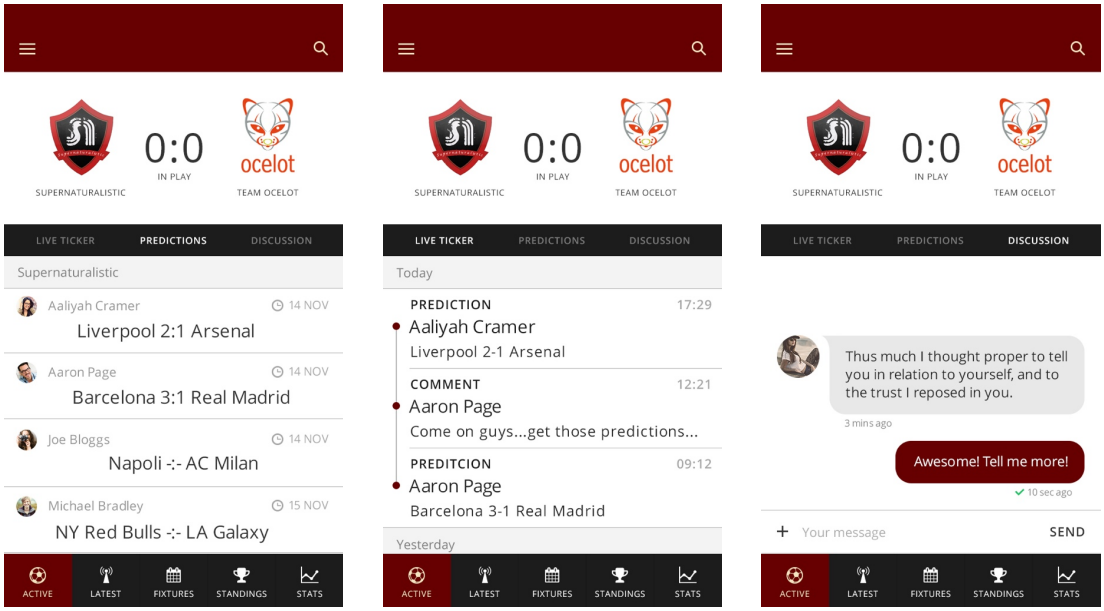
Join Team



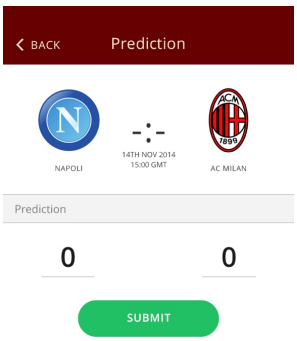
Create Team

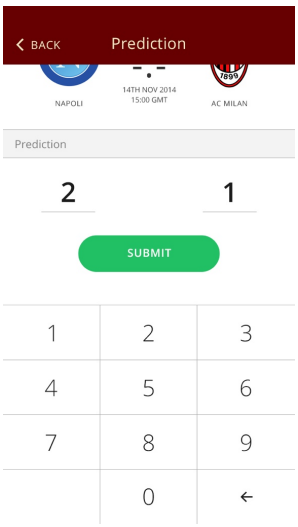


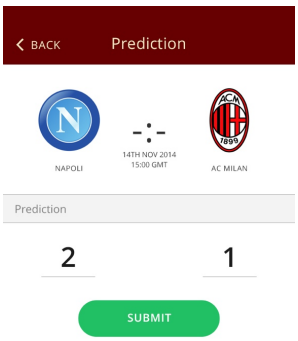
Active Match Screen

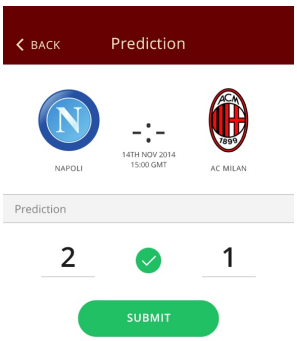


Submit Prediction Screen

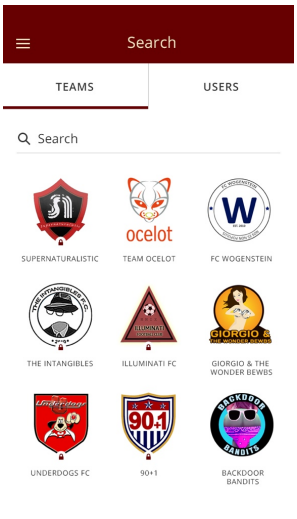


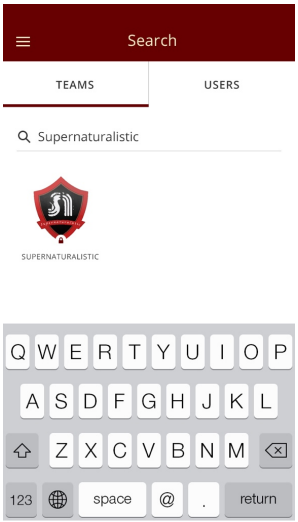


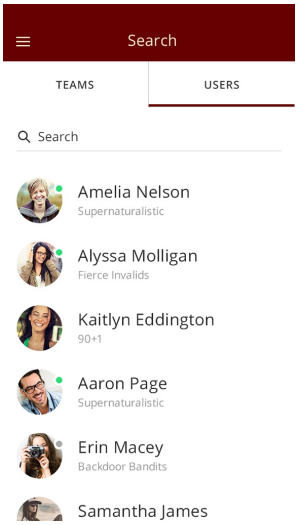


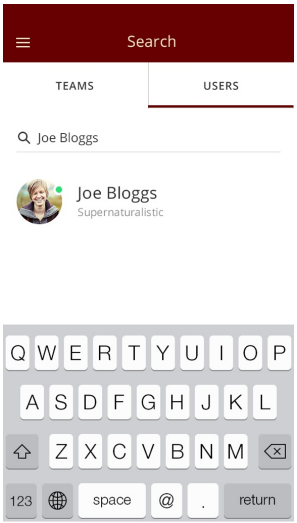


Search

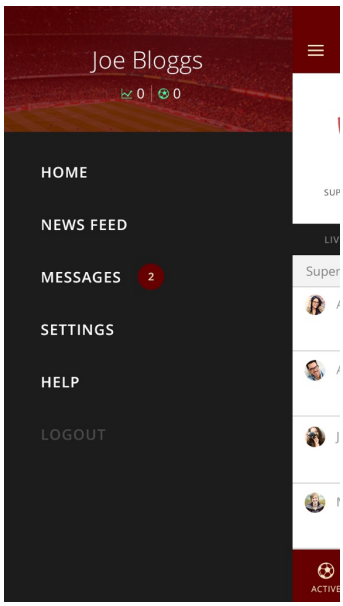


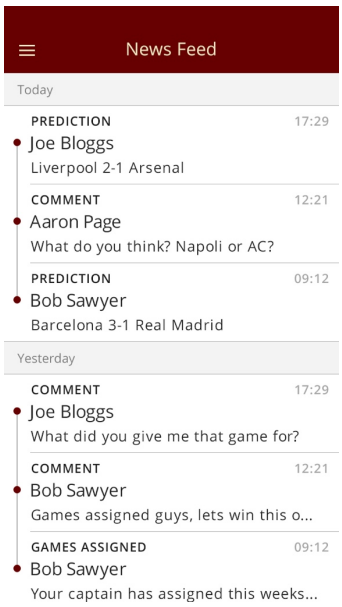


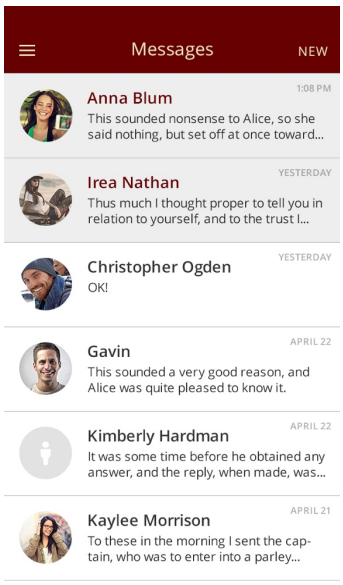


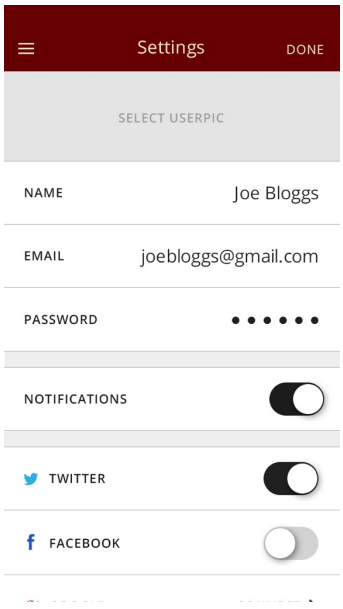


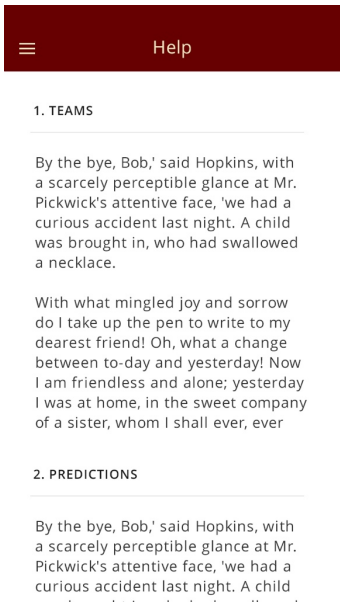
Additional Screens



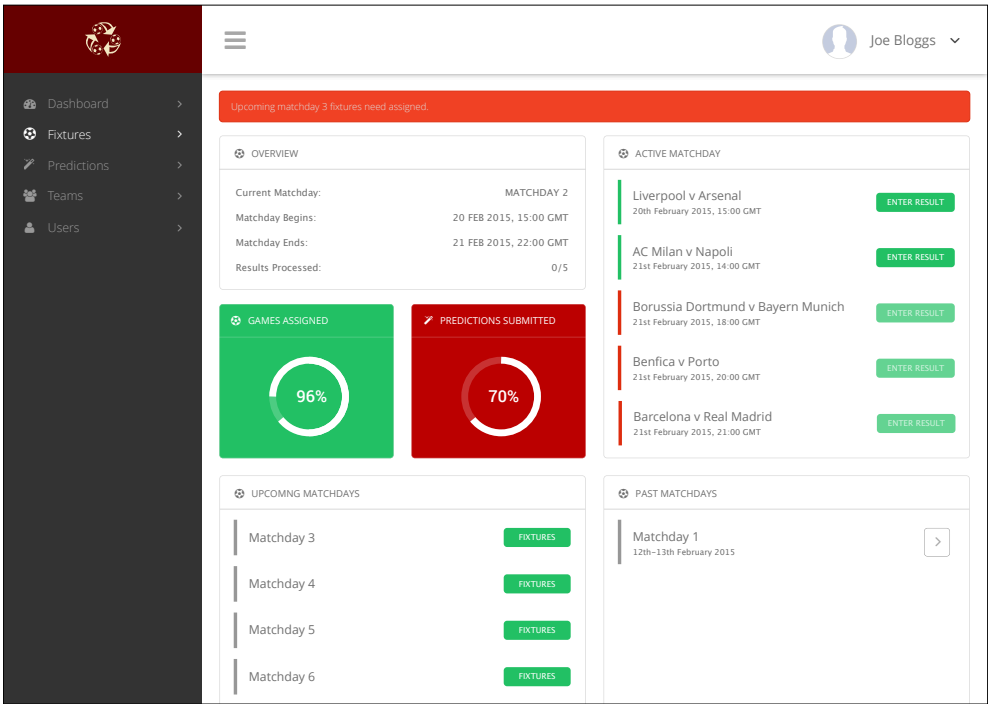




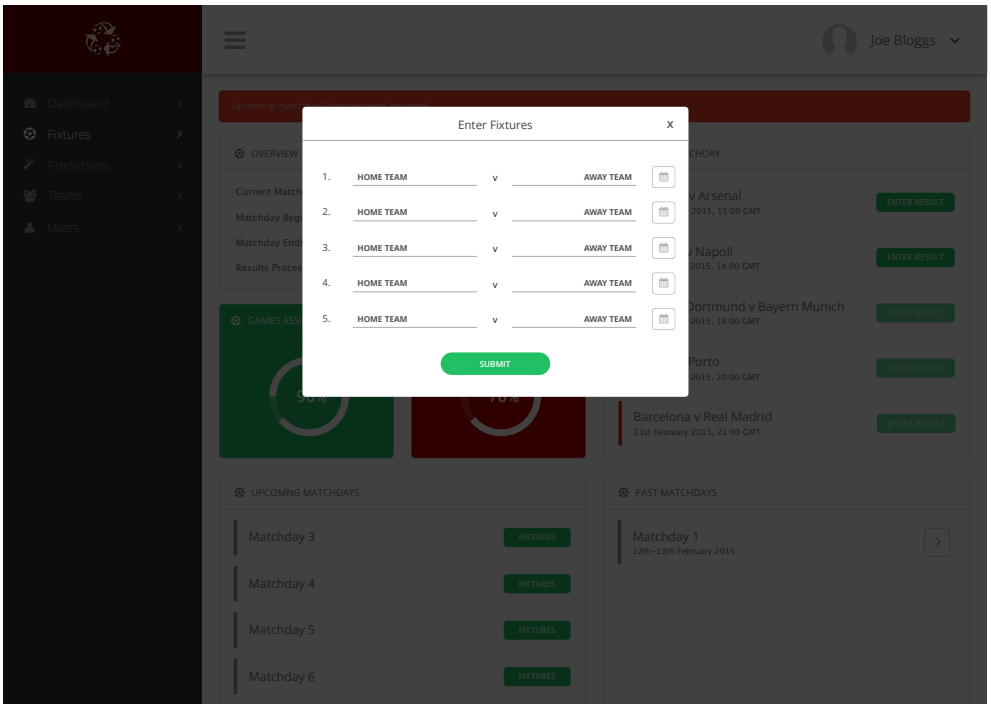




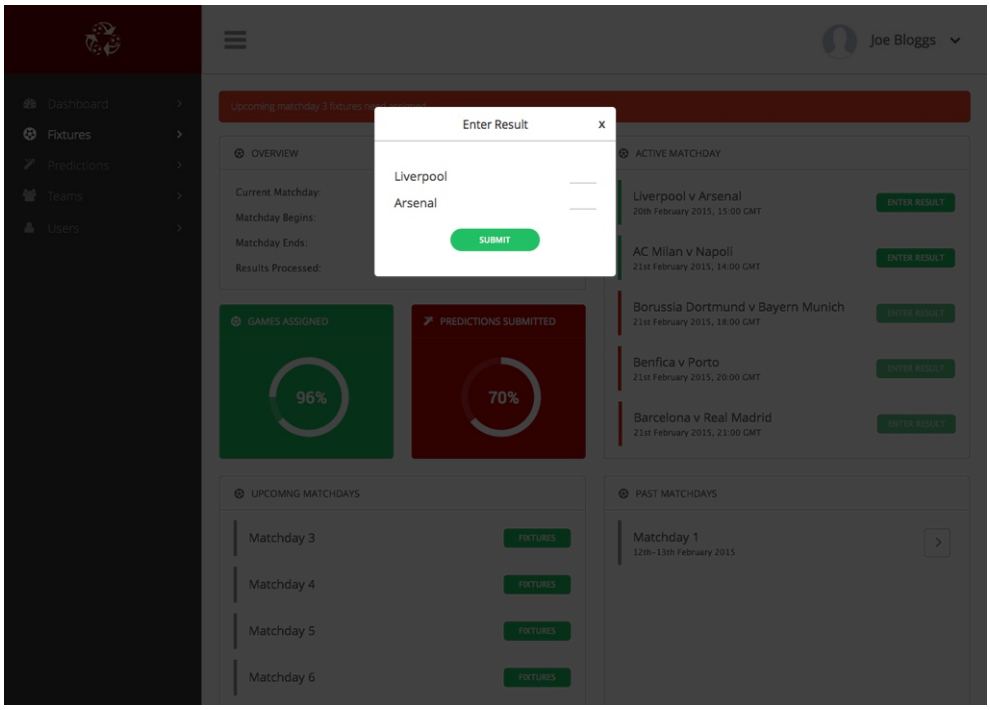
Admin Dashboard



Fixture Entry Screen

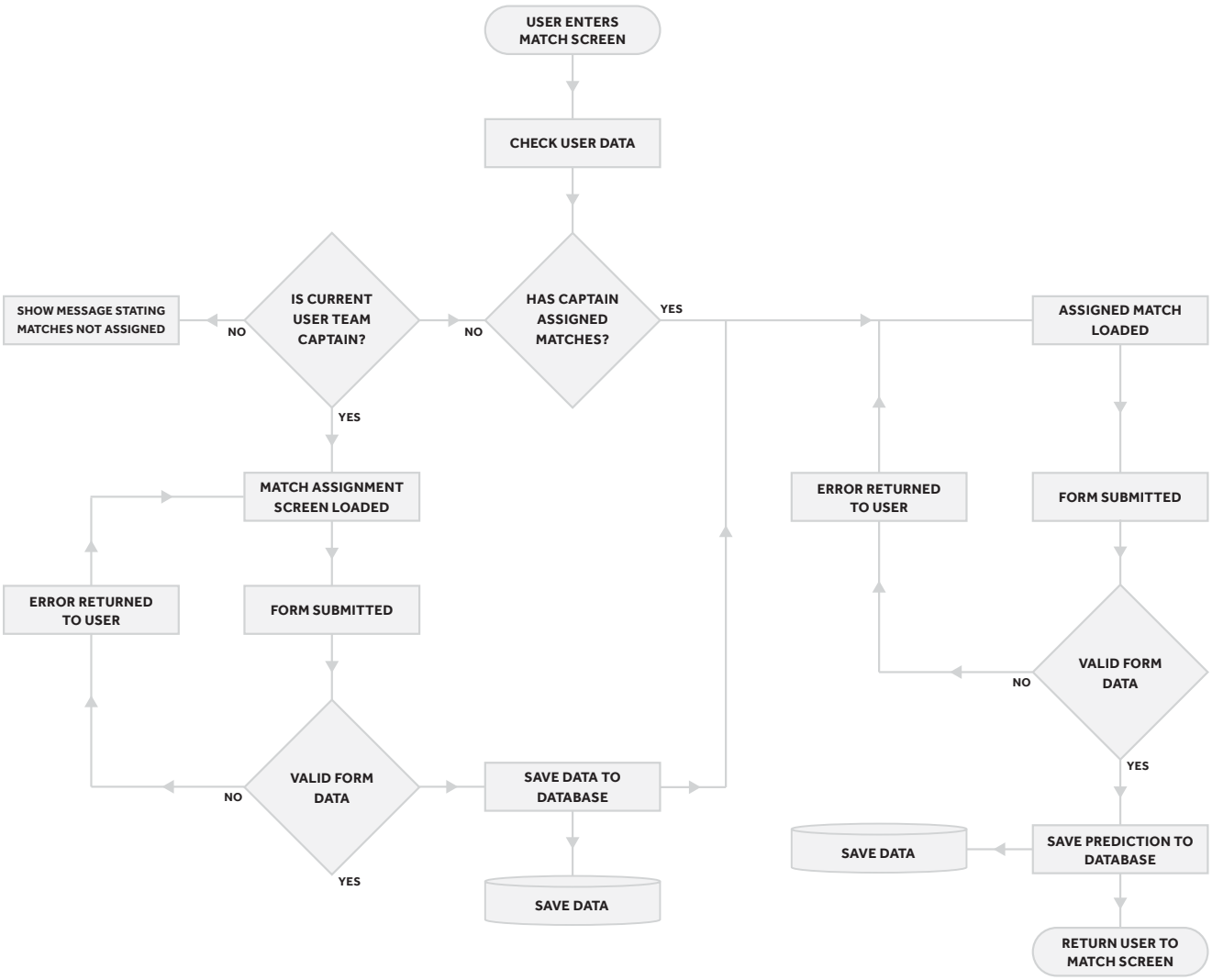


Results Entry Screen

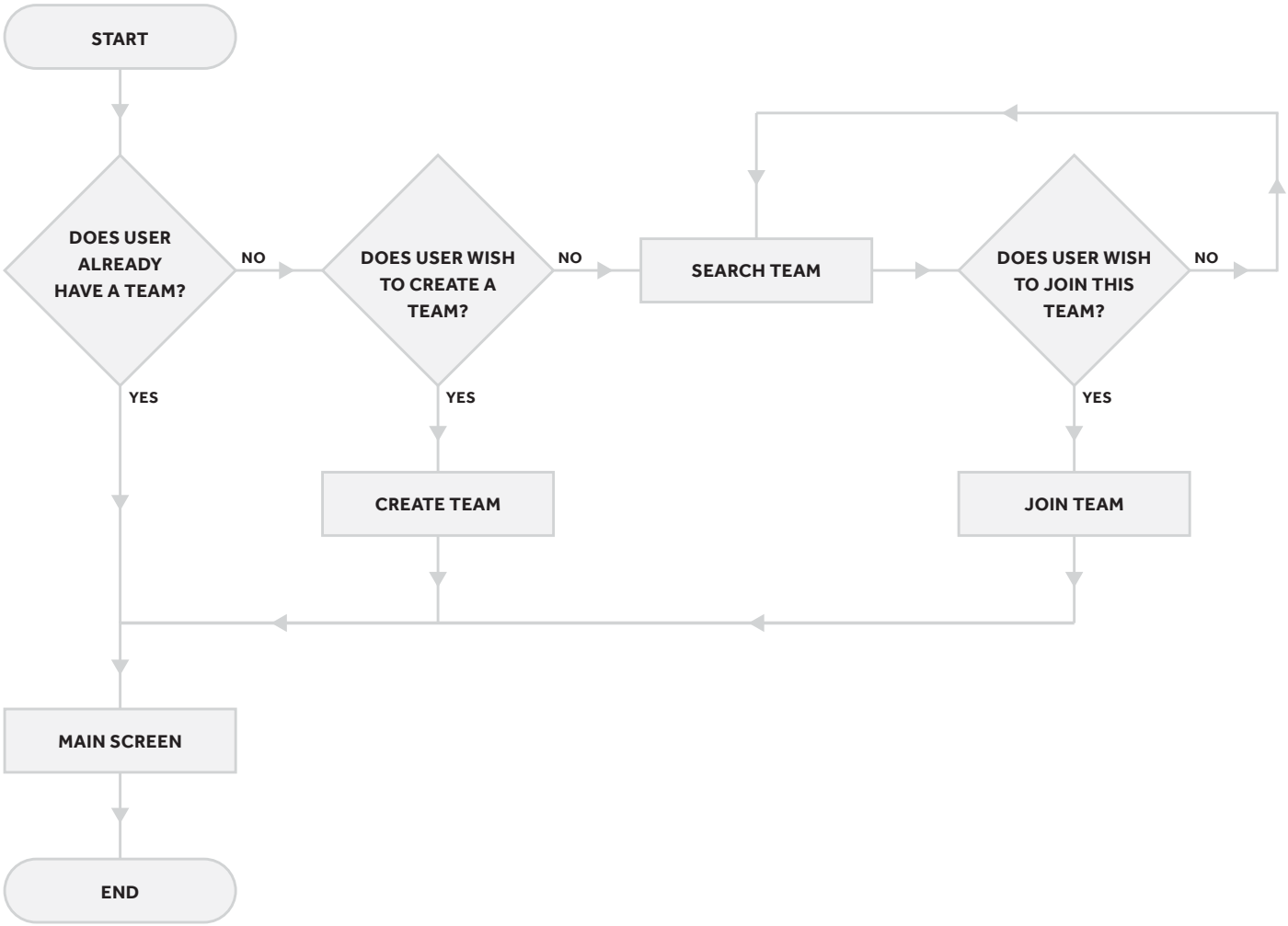


Appendix D

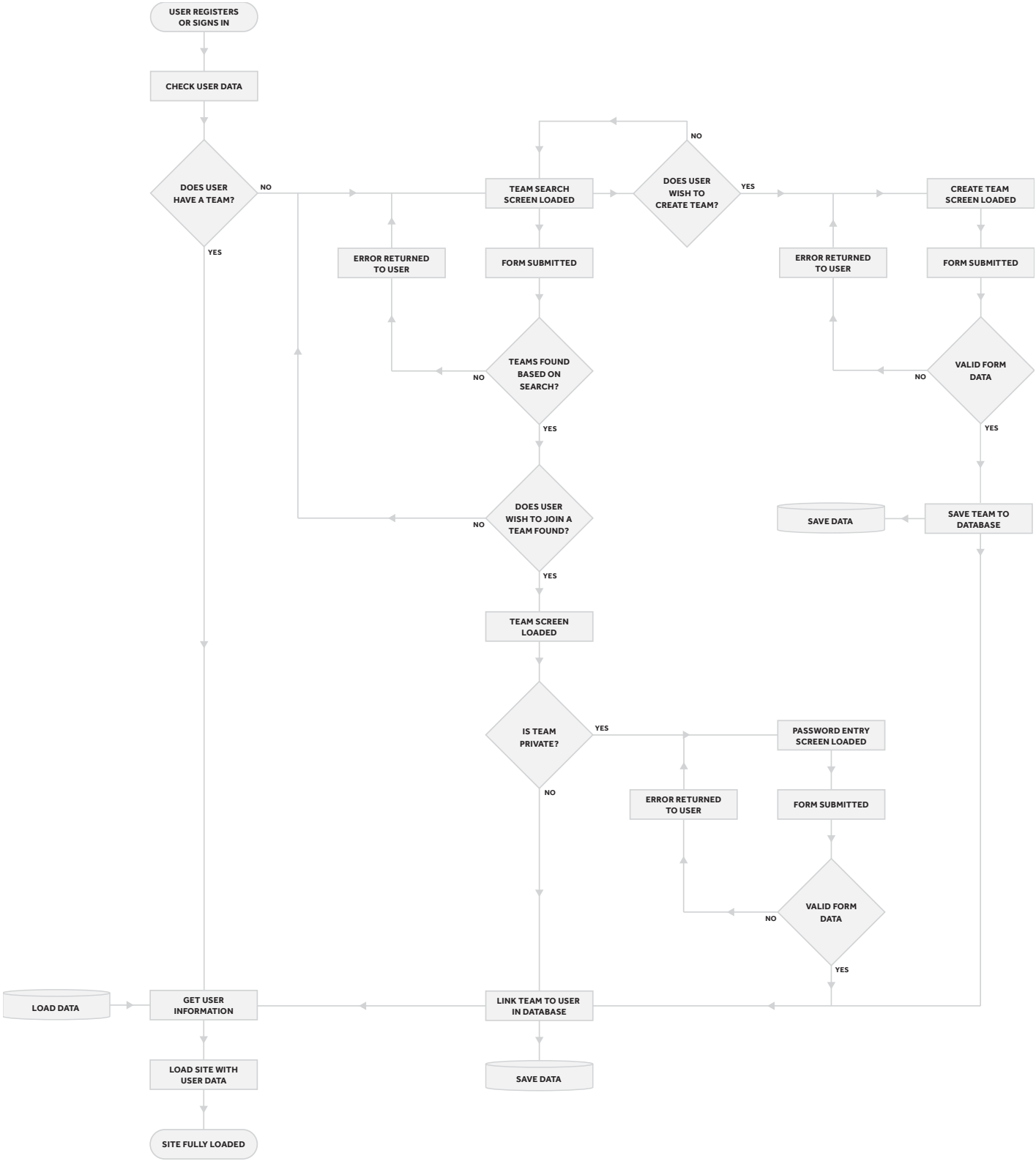
Match Assignment & Predictions Diagram



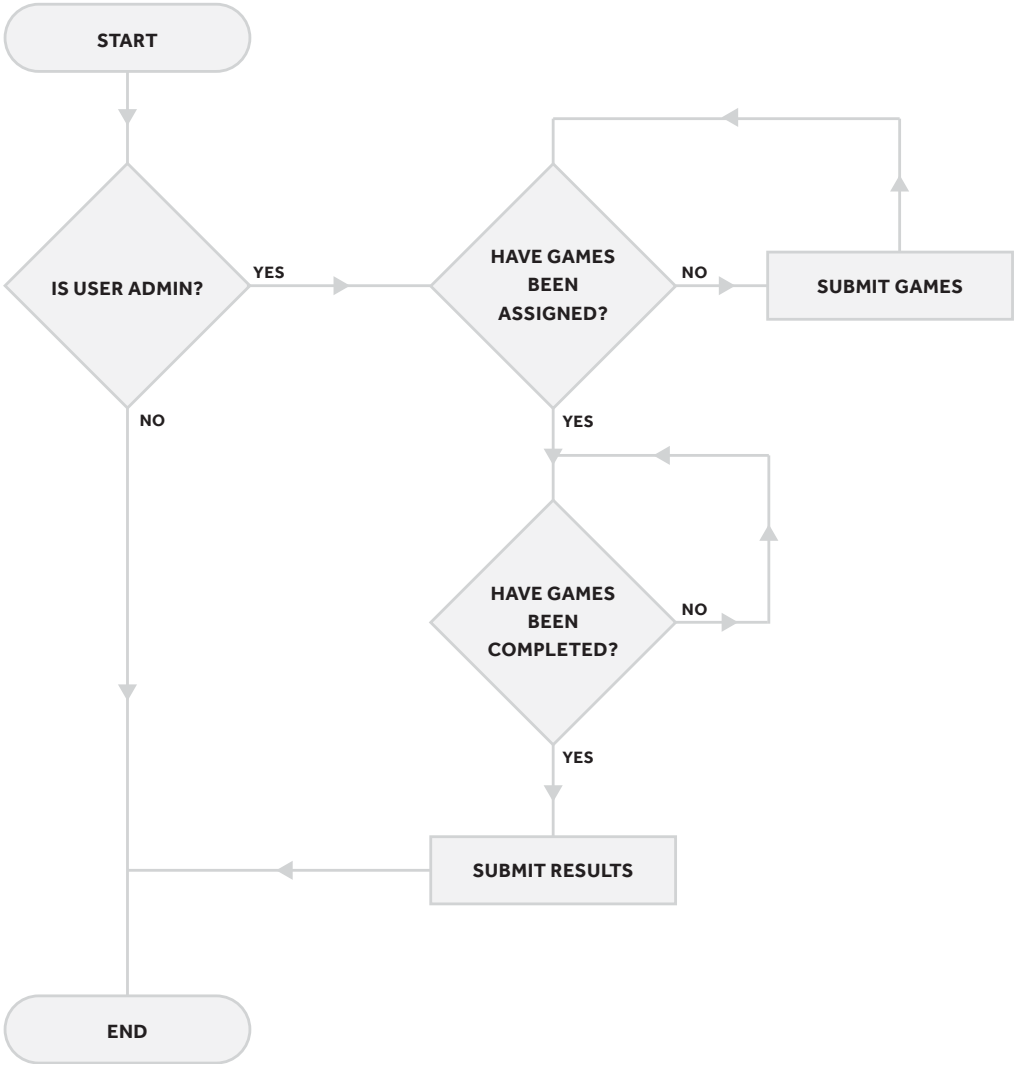
User Join/Create Team Diagram



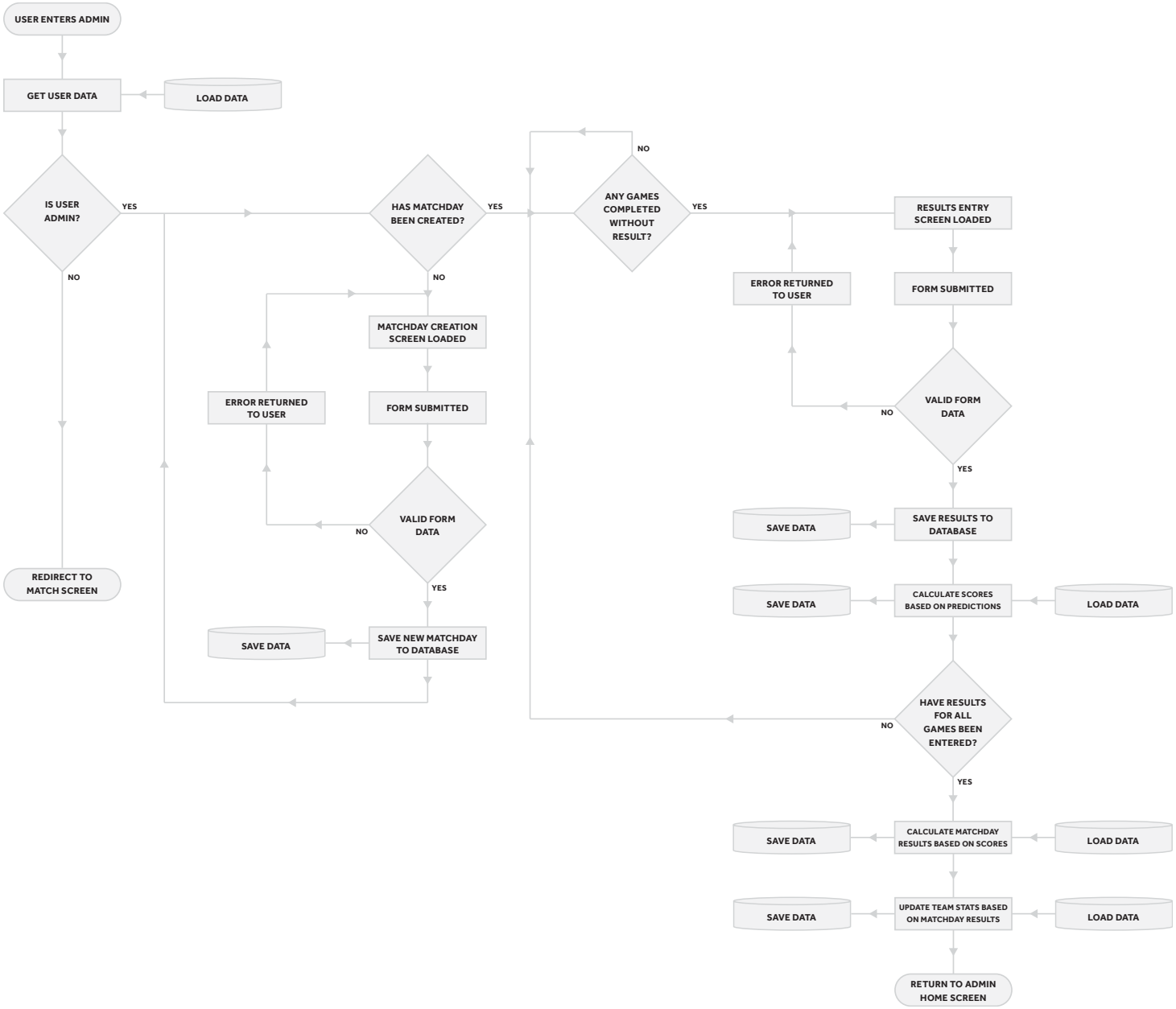
User Join/Create Team Data Flow Diagram



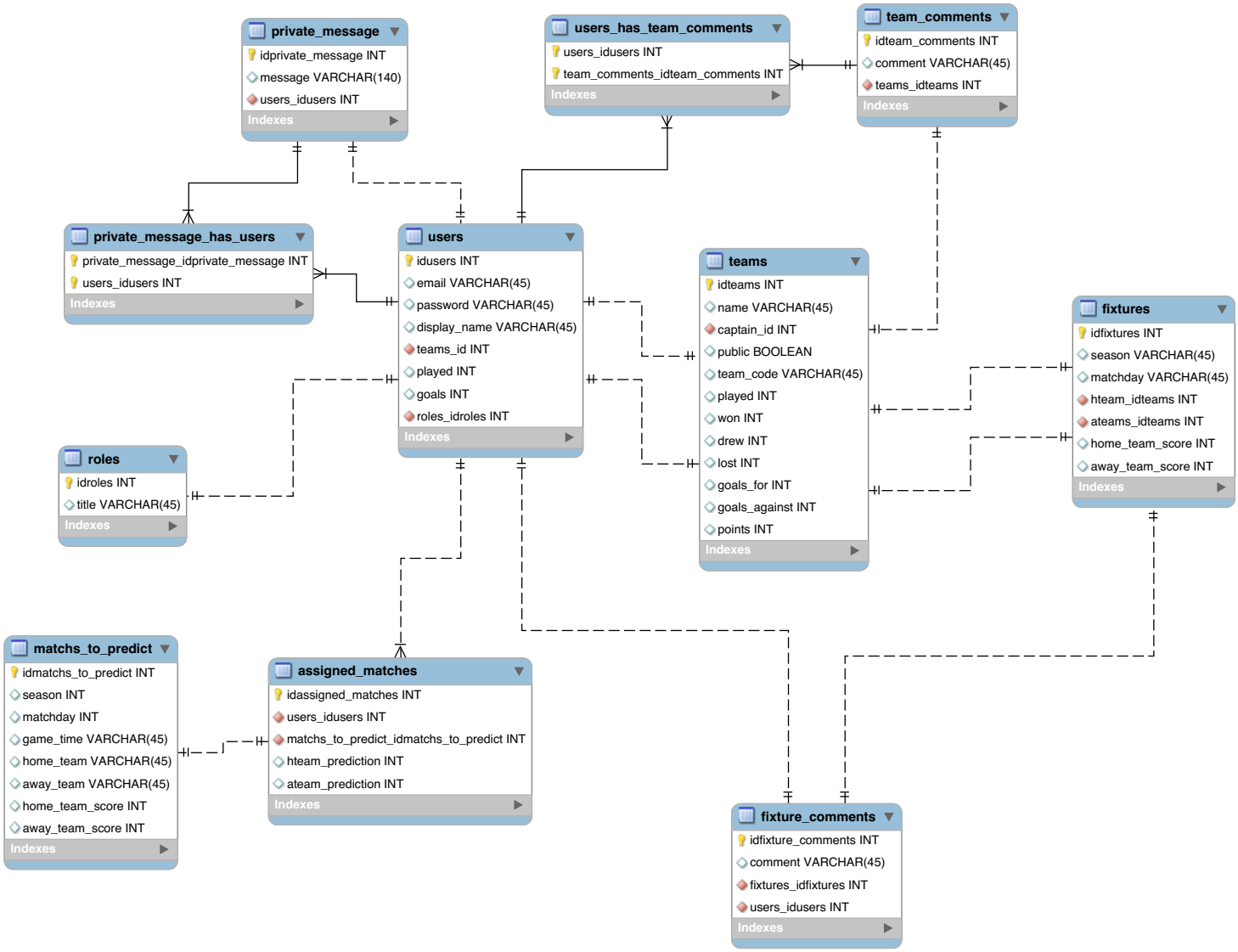
Admin Match & Results Entry Diagram



Admin Match & Results Data Flow Diagram



Database Design



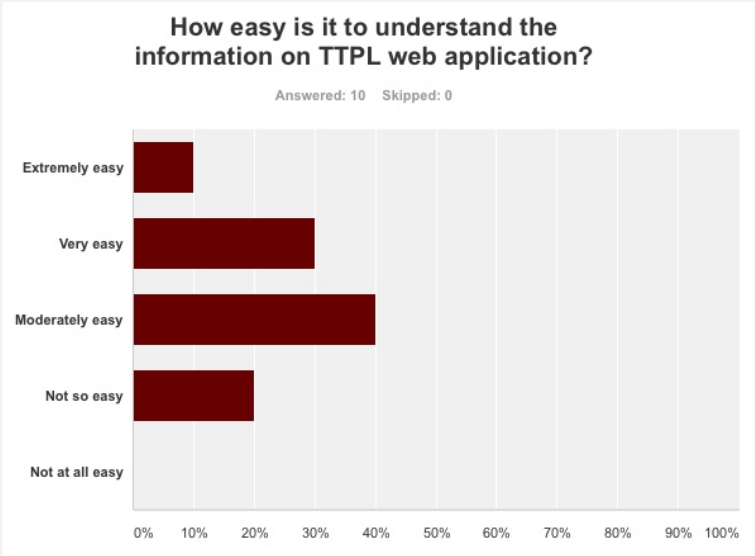
Appendix E

Risks Register

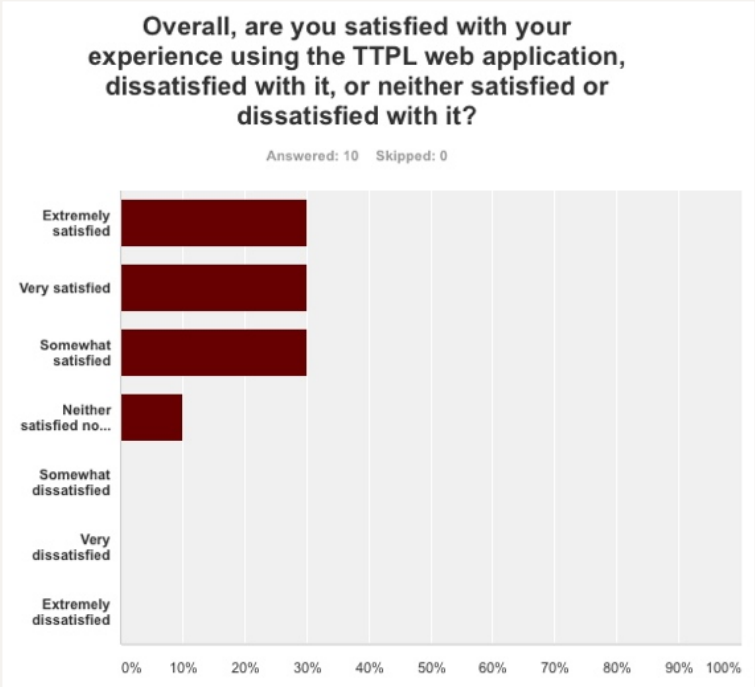
	Risk source		PROBABILITY			IMPACT			Result	IMPACT AREAS			Mitigation / Response plan
			Low	Medium	High	Low	Medium	High		Cost	Schedule	Performance	
1	New to Laravel Framework	Ali C.			9			7	63		x	x	Gain a thorough understanding of Laravel through the reading of documentation and reading material as well as using the framework completing tutorial and some system functionality through a prototype.
2	Unforeseen Circumstances	Ali C.		6				9	54		x		Unforeseen circumstances cannot be predicted which may cause disruption to the project. More time has been allocated to tasks that it should conceivably take.
3	Exceptional Circumstances	Ali C.			9		4		36		x		Additional time has been allowed for exceptional circumstances which may cause disruption to the project. More time has been allocated to tasks that it should conceivably take.
4	Fixture Generation Issues	Ali C.		5				7	35		x	x	The fixture generation could have been a complicated process however it has been simplified for the purpose of the initial build of the project. Fixtures will be generated on a weekly basis to allow for the inclusion of new teams.
5	Poor Project Management	Ali C.	3					7	21		x		The use of project management tool (teamwork) will be used to track all tasks and deadlines to ensure they are met. Daily notifications are issues to ensure tasks are getting completed on time.
6	Keeping User Data Secure	Ali C.	3				5		15			x	Use of Laravel as a framework which provides built in security measures such as guarding against form injection, hashing passwords. A secure server can also be used later on for full production use.
7	Design Changes	Ali C.		5		3			15		x	x	The addition or modification of features may require design changes but in using Laravel's tempting system this means these can be implemented quicker and easier.
8	Addition of Features	Ali C.		4		3			12		x		The use of Laravel allows for a pretty modular system meaning additional features can be added more easily. The use of git also allows braces to be created off the master project to test new features before adding them to the master project.
9	Data Loss	Ali C.	1					9	9		x	x	All project files will be worked on through a Dropbox business account to ensure all files are backed up on the cloud with unlimited version history. Git will also be used for version control to allow us to roll back changes or create branches of the project to work on.
10	Slow Site Performance	Ali C.	1				6		6			x	Making good use of caching and minification will minimise site load times as well having powerful servers to handle multiple requests.
11	Power Loss	Ali C.	1			3			3		x		Alternative places of work and power sources are in place with all required data being accessible from anywhere.

Appendix F

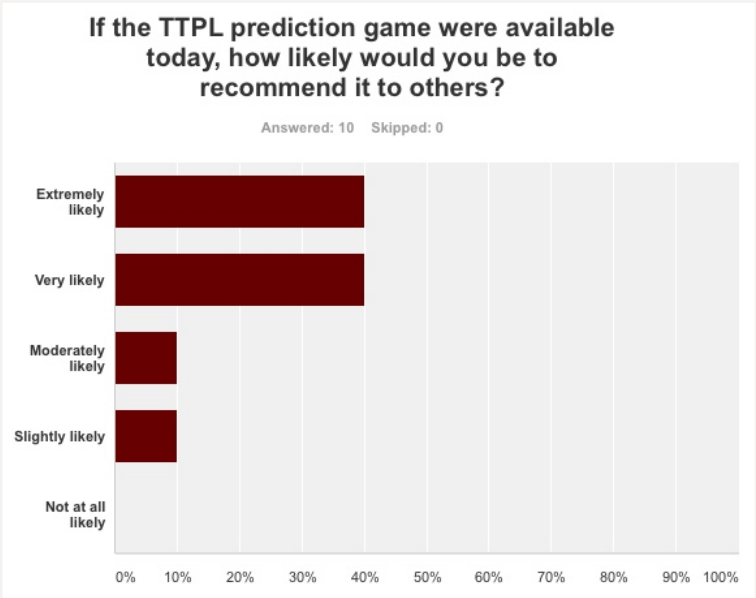
Question 3



Question 4



Question 5


If the TTPL prediction game were available today, how likely would you be to recommend it to others?
Answered: 10 Skipped: 0
Extremely likely
Very likely
Moderately likely
Slightly likely
Not at all likely
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%