Football Match Prediction Web Application - SureThing

Marina Shchukina

A report submitted as part of the requirements
for the degree of Computer Science
at the School of Computing Science and Digital Media
Robert Gordon University, Aberdeen, Scotland

March 2015

Supervisor Dr. Robert McDermott, Dr. Richard Glassey

Abstract

The aim of the project is to build a web application simulating the football betting experience and addressing two main issues. Firstly, filling an existing void for a system that makes football match prediction customizable and transparent to the user. For each upcoming match, the application will provide the user with an overall probability of either of the two teams winning the match based on the current football statistics and expressed in percentage. The user would then be able to influence this probability value by adjusting the weights for each of the several factors that contributed to that result. Depending on the values of the weights, prediction may be different for different users. In the long run, users of the web application would be able to create their own "betting system" by constantly re-adjusting the default prediction weights and hopefully coming up with a set of weights that works the best. Secondly, allowing the users to analyse their past performance and compare their results and prediction weights with the other users of the application.

The stated above would be achieved by taking several steps. First, the background research will be carried out. On completion of the research, current similar websites will be researched and a set of requirements will be created to assess users' needs. After that a layout and overall design of the application will be produced, as well as the desired behavior of its features. Once the prototyping is completed, the main project deliverable, i.e. working web application will be developed and throughly tested.

Acknowledgements

I would like to acknowledge and extent my gratitude to the following people who have made the completion of this project possible:

Dr. Roger McDermott for his support and guidance in this project Dr. Richard Glassey for his initial help and valueable advice My husband Murray and baby daughter Scarlett for their support and patience.

Declaration

I confirm that the work contained in this BSc (Hons) project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed ..... Date .....

Marina Shchukina

Contents

List of Tables

List of Figures

Contents

Chapter 1

Introduction

A short paragraph introducing the topic the chapter examines.

1.1 Background

A number of pages about the background of the project.

1.2 About this Thesis

This is the thesis of *Insert Full Name Here*, submitted as part of the requirements for the degree of MSc Computing: Software Technology at the School of Computing, Robert Gordon University, Scotland.

A number of paragraphs detailing the main expectations of this body of work.

1.3 Problem Areas

The main challenges associated with the project are:

1.4 Report Structure

A short conclusion summarising the chapter.

Chapter 2

Background studies and objectives

In this chapter the background of the project will be discussed. The chapter will take the reader through the history of sports betting with a particular focus on online gambling. The primary and the extended objectives of the project will be outlined and the professional considerations will be addressed.

!!!For this chapter put lots of references (especially in the BG subchapter)

2.1 History And General Information

Gambling is nothing new. Since time began, people have been betting on the outcome of an event, be that the victor in a gladiator competition, the winning team in a football match, or the first horse past the finishing post.

2.2 Objectives

People have always been interested in games with the element of luck and therefore gambling is one of the oldest forms of entertainment of mankind. The rise of the Internet and mobile devices has made remote gambling more available for a wide variety of users. The reason for that could be that Internet applications and websites can be are easily accessed 24/7. Amongst the most popular types of online gambling can be found card games, dice games, electronic games (such as poker), betting on sporting events, etc. Sports betting is no longer associated solely with horse racing. Among all types of sports gambling, football gambling is a leading industry with a share about 70When it comes to any sports betting (football betting including), the user is trying to predict the result of the event and placing the money on the outcome. This prediction can be made based on a “hunch” or by using logic

and domain knowledge, in a lot of cases by both combined. Naturally, this gave rise to a vari- ety of betting software systems that are attempting to predict the next match re- sult. Most of the time those betting systems work as a "black box" not allowing the user to influence the prediction output and preventing the user from understand- ing the exact logic used inside the system. Football bettor can have various strategies when making a betting decision. As mentioned above, the user can buy a prediction software and simply follow the tips suggested by that system. Another option is to make a decision influenced by the opinion of the other tipsters, experts' opinions or rumours. However, if the aim is to achieve sustainable profit (or minimise the loss from betting) , most experi- enced bettors would ignore betting tips and predictions of others and go for the pure facts trying to make their own prediction. To make this happen, the bettor has to aggregate several pieces of information from various sources. This action has to be repeated for every single match. From my experience, the necessity to repeat an action many times could lead to a creation of interesting software solutions. That is how I got inspired to create an application that would aggregate this information for the user and therefore act as an interactive decision supporting system.

2.2.1 Primary Objectives

The purpose of my final year project is to create a web application that can help users to predict football match results and make profitable bets. There are many similar websites and applications. However, I think my application is different from the other ones. The key feature of the app is the fact that the prediction output is transparent to the user and can be easily adjusted and customised.

2.2.2 Extended Objectives

Hello!!

The purpose of my final year project is to create a web application that can help users to predict football match results and make profitable bets. There are many similar websites and applications. However, I think my application is different from the other ones. The key feature of the app is the fact that the prediction output is transparent to the user and can be easily adjusted and customised.

2.3 Problem Specification

2.3.1 Limitations and Evaluations

Hello!!

2.4 Social, legal and ethical issues

Legal Finding free API for a newly created betting application may become a non-trivial task. Therefore, I am considering using "web scraping" as one of the options to load most recent football data into the system. It can be said that there is a fine line between collecting information using the "web scraping" technique and stealing it. Most of the websites have a copyright disclosure defining the rules for the use of the information they provide. Thus, I will carefully read the disclosure statements and follow them along legally and ethically.

Ethical Due to the nature of the application it is inevitable that it will store some basic user data in its database. The application must take all the necessary precautions to protect the stored data and sensitive information. The application will not disclose personal data of its users to any third parties.

Social There are several advantages of using the application for a rational bettor. First, the use of it will hopefully lead to more profitable football betting and also reduce the amount of thoughtless bets. Secondly, the use of application will save time spent on gathering information before making a betting decision.

Professional Although the main aim of the application is to provide transparent prediction to the user, there is still certain amount of calculation happening in the background. I assume that user will trust the betting system when making a betting decision. Therefore ensuring the accuracy of the calculations and providing good test coverage is a very important part of the application development.

Hello!! a wee summary: what we discussed in this chapter In this chapter we discussed the history. I also introduced the objectives of this paper and lined up several use cases making clear for the user how the application is going to work.

In general, the application is not aimed to promote gambling. Moreover, it supports a more sensible and measured approach to football betting. In this chapter we discussed... (the main conclusion for the chapter)

Chapter 3

Requirement Analysis

Before designing a piece of software, it is important to have a good understanding of the project requirements. In this chapter a number of functional and non-functional requirements will be listed covering all the aspects of the future web application.

3.1 Target Audience

The intended users of the application will be people with interest in football. More specifically, people interested in predictions of football results and football betting. With this in mind, the age range of potential users will be 18 and up.

3.2 Target Audience Questionnaire

The target audience research aims to gather information on how... Due to the spread of target users, an electronic questionnaire was used to collect the results. A full break down of the questions asked and the answers received can be found in the appendices, reference X.

3.3 Researching the Potential competitors

Before gathering the project requirements, it is good practice to conduct research on what current websites are already available to football fans with interest in betting. The research can be a source of inspiration and could also help to avoid potential design mistakes. During the analysis, it is important to attempt to understand the main purpose of the analysed websites, as well as the way they present

information to the user and communicate with them.

This section is concerned with websites that can be useful for predicting football results. In our context, these are the various online sources of information a football punter would turn to before making a betting decision unless the decision is based solely on intuition. In general, several different types of such websites can be found online, namely:

1. Sports news websites
2. Football statistics websites
3. Bookmakers websites
4. Communities
5. "Black-box" prediction applications

This section of the report looks at one or two examples of each of the categories presented above, analysing the weak and strong points of the chosen website and discussing usefulness of the whole category from the point of view of a football punter.

3.3.1 Sports news websites

This category represents football news websites. Into this category fall both general sports websites with a football section and football news websites, such as:

- BBC Sport - http://www.bbc.co.uk/sport
- The Guardian Sport News - http://www.theguardian.com/uk/sport
- Sky Sports - http://www.skysports.com/
- The Times Sport section - http://www.thetimes.co.uk
- Football365.com - http://www.football365.com
- Goal.com - http://www.goal.com

The sports news websites aim to present the news alongside the essential football statistics. The information is usually not as detailed as on the football stats websites, however the very latest football news compensates for this drawback. Each of the *news* websites named above (BBC News, The Guardian, Sky Sports) have a football section that presents the reader with the combination of football news and stats. One of the most popular football news websites is BBC Sport Football.

3.3.1.1 BBC Sport Football

BBC Sport Football is a brilliant news website. It offers a very neat and simple interface and does not overwhelm the reader with irrelevant graphics. Although, it almost looks too minimalistic, the user can still get all the most important information about football teams and players. The website provides authomatically updated live scores accross all featured football leagues.

BBC Sport has a very impressive news coverage of both major and minor British football leagues, as well as the main european leagues. It can also take pride in high quality writers (journalists) contributing to the website. An interesting feature is videos embedded in the webpage.

As already mentioned above, the main drawback is that the stats are kept to a bare minimum. This can be a problem for a serious football fan that wants to analyse the details of the game from all possible angles. However, for the purpose of a punter this level of statistics should be sufficient.

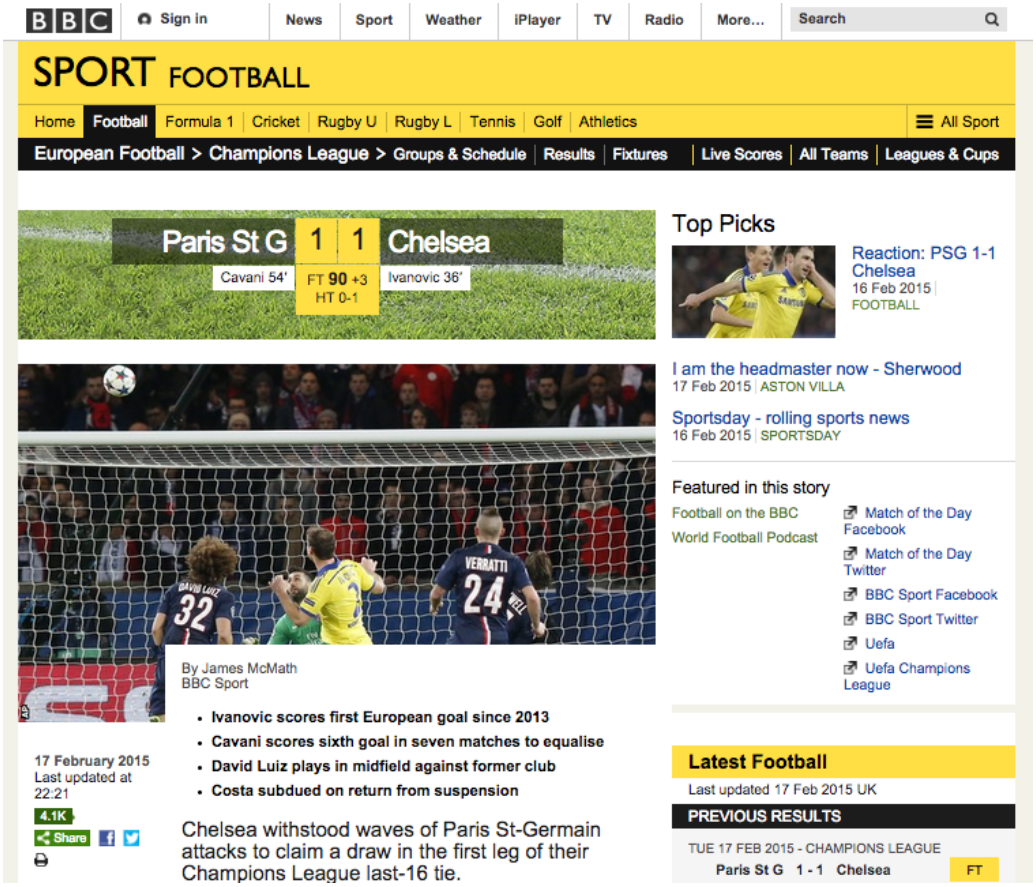

Figure 3.1: BBC Sport - Football section

3.3.2 Football statistics websites

There can be found a large selection of football statistics websites focusing on detailed stats and analysis on football matches, teams and players. These are some examples of websites in this category:

- WhoScored - http://www.whoscored.com
- Squawka - http://www.squawka.com
- WhoIsInjured - http://whoisinjured.com

3.3.2.1 WhoScored

Among all the football stats websites I have analysed, WhoScored is one of the most impressive ones. It has a lot of statistics, but most of it seems to be quite relevant. The website is extremely well designed, and its navigation is intuitive. WhoScored offers statistics and deep analysis on the major European divisions, as well as providing data on over 500 leagues and 15,000 teams. As to the data source, the website is supported by Opta, the largest and a very reliable live sports data company that stands behind BBC Sport, Sky Sports and other significant UK sports news providers.

The way "WhoScored" presents information on particular matches, both upcoming and played, is detailed while being uncluttered,which makes it both very useful and easy to use. This is definitely an aim of this particular project and I will be using a similar format when designing the website.

3.3.2.2 Squawka

Squawka is another website worth looking at.

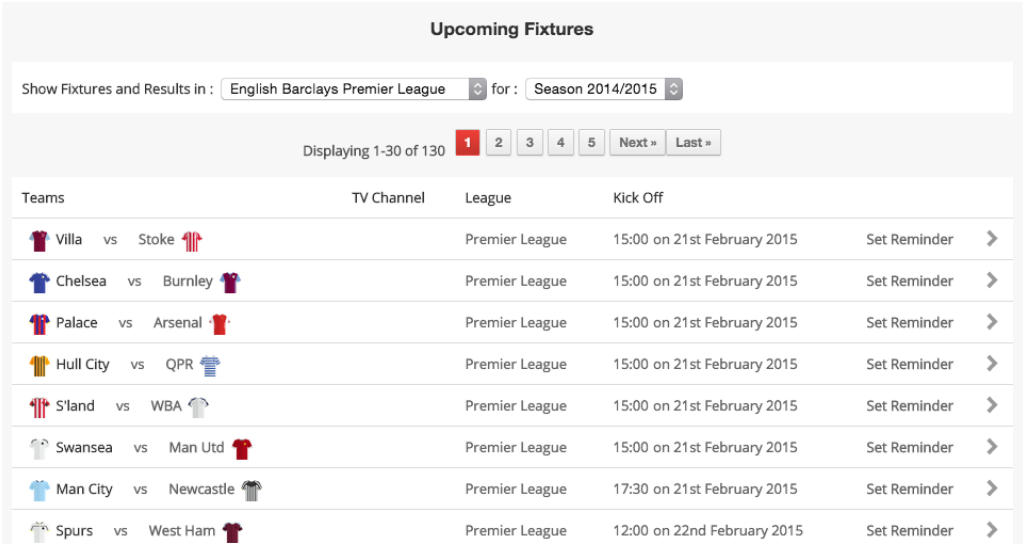

Figure 3.2: Squawka

It is an application for football fans that uses real-time data visualisations to explain the game. The main idea behind it is to show users the live stats as the game is being played.

From a visual point of view, Squawka has a nicely designed, pleasant interface. However, it is a little bit heavy on the client-side (Javascript), which contributes to sometimes slow performance. Another downside of the website is an extensive amount of adverts that distracts the user form the main content.

3.3.3 Bookmakers Websites

With the arrival of the Internet many existing bookmakers opened up web based operations to complement their existing business. Within short period of time online sports became very popular with punters all around the world. Most online bookmakers contain high quality sports statistics that aims to support users' betting decisions.

Names like Ladbrokes, Bet365, Williamhill are probably one of the most popular bookmakers online.

- Paddy Power http://www.paddypower.com/bet
- Ladbrokes http://betting.ladbrokes.com/en/football-betting
- Bet365 http://www.bet365.com
- WilliamHill http://sports.williamhill.com/bet/en-gb/betting/y/5/Football.html

3.3.3.1 Bet365

According to the [23], "Bet 365 Group Limited, is a United Kingdom based gambling company. Bet365 is one of the world's leading online gambling groups with over 14 million customers in two hundred countries". In my opinion, Bet365 is one of the nicest websites among bookmakers. The website is very well structured, it has intuitive, user-friendly navigation and is easy to use. Bet365 offers a free live streaming service and an impressive coverage of live sports statistics.

3.3.4 Communities

Communities is a category of websites with an interesting idea behind it. These websites are specialised social networks for sports fans and punters. So far, websites of this type are not extremely popular, maybe because gambling is more of an individual pursuit.

Many of the features of websites in this category, such as experts tips, users' comments, forums, can be very useful from a punter's perspective, especially when dealing with a league you are not overly familiar with. I have analysed three websites in this category.

- OLGB Betting Community - http://www.olgb.com
- Vital Football News and Fans community - http://www.vitalfootball.co.uk
- Punters Lounge - http://www.punterslounge.com/

OLGB is a friendly community for punters with many interesting features and tools. It will be analysed in more detail in a separate subsection below. Vital Football is a "network" website. It runs a separate website for every football club from the Premier League and the Football League in England and the Scottish Premier League with each "club site" having their home page, own editors and a forum. Punters Lounge is a "betting and poker community". Its most interesting feature is a forum for sports punters. The website also offers free betting advice tips, live sports streaming and odds comparison.

3.3.5 OLGB Betting Community

OLGB community market themselves as a website for punters who share their expertise and work together to maximise their betting profit, so it is very relevant to this project. OLGB has a wide variety of features and tools. However, real punters' opinion and tips is the main focus of this website. For example, the user can navigate to an upcoming event page on OLGB website (in the section "free tips") and check how many other users predicted either team to win. Users' choice is usually justified with an comment. For each option OLGB suggests the best odds from one of the most popular bookies. This saves user the trouble of going to a website like OddsChecker[**?**] to compare the odds. The comments feature is very interesting and it can be applied as an "optional requirement" for this project.

OLBG also runs a virtual betting game (tipster competition) where users can "bet" virtual money on real betting events. The tipsters in the top 100 of the tables for each sport each month receive a real money prize. Users of the website can also see a leaderboard of the most successfull tipsters. The leaderboard contains information on the amount of tips made within certain period of time, LSP (Level Strike Profit), ROI (Return on Investment) for each tipster. The virtual betting feature is very relevant to this project. A simplified version of the leaderboard could also be implemented.

OLGB has more to offer. It is a punter's paradise in a way. For example, there is a betting forum, betting blogs section, betting school and much more. The community website has a tight connection to several popular bookmaker websites. For example, it offers a tool to help users to compare bookmakers websites. It also promotes free bets and various "bookies" promotions.

The website has also several drawbacks. Firstly, the interface looks a little outdated and it takes a while to find your way through the complicated and rather confusing navigation. Secondly, the "Help"

section does only answers some questions and can be slightly disorientating for a new user.

3.3.6 Black-box prediction applications

These are betting applications that are being marketed as systems that can predict the outcome of a football match using their own unique statistical models and calculations. Known as black-box systems, these applications avoid sharing the exact logic used inside the system with the user. The punter subscribes to a betting system online and simply follows its suggestions, for example, estimated outcome of a football match, overestimated events that the user should bet on, etc. These are the example apps in this category:

- Math Betting - https://apps.betfair.com/trading/mathbetting/
- Footbee - http://footbe.net/en/
- Vitibet - http://www.vitibet.com/
- Forebet.com - Mathematical Prediction http://www.forebet.com/en/

When paying for the subscription for a black-box betting application, the users assume that they are gaining access to a smart system created by football experts. Additionally, there is an expectation that the application has complicated statistical models and calculation analysing a large database of football statistics behind each betting tip. The users also hope that they are paying for a *secret* betting system that is known to and is used by only limited number of other punters. Summing up, a black-box system application sounds like an easy way to sustainable profit. Unfortunately, this does not have to be the case.

First of all, the complexity of a betting system does not necessarily correlate with its profitability. The logic behind a successful betting system can be relatively simple and would still work. Secondly, in order to predict a football match result, it might be enough to analyse only relatively recent events (for example last 6 matches, recent players' performance, etc.), without having to perform computations on large set of historic football data. This is due to the fact that over time many various factors can cause change in team performance. Therefore, analysing data from several months ago would not help the punter to make more precise prediction. (put reference to the article!!!)

3.3.7 Conclusion

After having analysed the above websites, I came to the following conclusion. A well-chosen combination of several websites would definitely be able to provide enough information to make a thoughtful

betting decision.

Many punters have their own football betting system (betting strategy). Although even the best system cannot guarantee success, it can greatly increase the prospects of making a profitable bet. Therefore, before making a prediction, such a punter will conduct a little research each time. The aim of such research would be to make note of the relevant statistics for the teams involved in the game (depending on the "input variables" of the betting system), for example previous results for both teams, their performance at home and away, players statistics (injuries, suspensions, transfers), recent change of team managers, etc. The problem is that many football stats websites offer way too detailed information for this purpose, therefore the relevant stats needs to be "hand-picked" from several sources for each match.

The application will attempt to put all the relevant statistics in one place and break the information down into logical "prediction modules". In addition, the application will enable users to pick the input variables and assign them a weight of user's choice, representing the importance of the input variable for the match results prediction.

3.4 Requirements

Computer Systems life cycle consists of five phases: Requirements and Analysis, Design, Implementation, Testing and Evolution. Then, after Evolution, the cycle can start again with an additional set of requirements. [**?**] defines requirement as follows: "A *requirement* conveys an essential property that the system must or should satisfy." Requirements analysis involves gathering information in order to meet customer needs and defining what the future application is expected to do. This phase of software engineering is especially important in the industrial environment, when developing an application for a customer. In this case, clarifying the requirements in the early stages of the project would help to ensure that both sides understand and agree on the feature set of the future application.

Although it is not very likely that requirements for this project will change during the development process, defining requirements can be very beneficial. The detailed requirements analysis will aid understanding how different parts of the project are expected to interact with each other, as well as how the application will communicate with its users.

For better transparency project requirements have been split into functional and non-functional requirements. Functional requirements will be further subdivided into mandatory and optional, depending on the degree of constraints.

Before outlining project requirements, I would like to start with some definitions relevant to the whole

project.

3.4.1 Definitions

Application Football League - in order to reduce unnecessary complexity at this stage of the project, the application will be only supporting one league (Premier Barclays League).

Matches Overview – a list of upcoming and played matches presented on the main page of the application

Dashboard - an interface available to authorised users, containing a list of played ("archived") and upcoming matches saved by the user

Prediction Module – my own term, standing for blocks of football statistics for each of the playing teams (for example, league position), that are evaluated against each other and based on the result of such evaluation, a match prediction (in favour of one of the teams) is made. Application calculates "prediction value" for each of the modules.

Match Result Prediction - to calculate a match result prediction, each prediction value is first multiplied by its weight. The weights would be different for each user unless default system settings are being used). Then, all of the weighted values are added together and based on this value a prediction is made.

Default System Settings - application has a set of default weights for each prediction module. These weights are used in the prediction calculation by default.

Default User Settings - each user of the application can save a set of prediction weights that are different from the default system settings. From the moment these weight are saved, they will be used by default for each match committed by the user.

Match Specific Prediction Settings - each user of the application can also save a set of prediction settings specific for only one match.

Match result - "Home Win" in case of the win of the hometeam, "Away Win" in case of the win of the awayteam, "Draw" for the draw

3.4.2 Functional Requirements

Functional requirements describe the behaviour of the application in terms of its functionality. These are the "must have" functions of the application addressing the business targets that application must

satisfy. Good functional requirements must be concise, complete and unambiguous.

In order to add structure to the design and development process, the project has been broken down into high level features of the future application. The functional requirements are grouped by the functionality related to these features.

3.4.2.1 Authentication and User Profile

These are the requirements related to the basic functionality of the web application, such as account registration, login and account management.

- The application should allow users to register and create a new account with the application
- User will be able to register using a standard web form
- For the registration purposes user will provide a valid email address and a password
- User will confirm a password in a separate input field
- After a successful registration application will send an email containing a confirmation link to the user
- On a successful confirmation of an email address, user will be successfully registered
- In case of any technical problems with the initial confirmation email, the application will be able to issue a new email and send it to users on their request
- The application should allow users to sign into their accounts using a standard web form
- When signing in, user should provide valid credentials, otherwise an application will throw a validation error
- The application will allow users to manage their account by changing personal information (for example, location, favourite football team)
- The application will allow users to change the email address associated with their account
- The application will provide an option to change user password for security reasons

3.4.2.2 Matches Overview

Below are listed requirements relating to the matches overview.

- On the main page of the application user will be presented with a list of upcoming matches for the current season in the league
- User will be also able to view a list of played matches and switch between upcoming and played matches using navigation tabs
- For each of the unplayed matches user should be able to navigate to the match page and see more details about the match
- From the main page user will be able to save any unplayed match to the dashboard
- For each of the played matches user will be able to navigate to the match page and see more details about the played match

3.4.2.3 Prediction

The predicted outcome of a football match will be calculated after evaluating several factors that can influence the match result. An example of such a factor could be a previous match result, league position of each team, team performance at home or away, the recent change in team management or players' injuries and suspensions. Each of those factors will be represented by an equivalent "prediction module" in the application and a match view. The main outcome of each prediction module will be a "prediction value" expressed as a percentage. This value will tell the user how likely is either team to win the match.

Finally, the betting system will assign a weight (depending on user settings) to each prediction value. The weights (or prediction settings) determine the relative importance of each factor. By applying the weights user indicates that some factors are more important to the outcome of this particular game than others. The final result will be calculated as a weighted average.

These are the requirements related to the prediction of a football match result.

- To calculate prediction values for each module, application uses default system settings in absense of default user prediction settings or match specific settings
- The application uses default user prediction settings (explicitly set by the user using specil form) in absence of match specific settings
- Application uses match specific settings in case user has set them for this match
- Match result outcome will be calculated as a weighted average of prediction values for each of the prediction modules

3.4.2.4 Upcoming Match View

Users should be able to view detailed statistics of the upcoming match. These are the requirements related to this functionality.

- User should be able to see a set of prediction modules
- Each module should contain relevant piece of football statistics
- For each module it should be clear, which team is more likely to win and what is a probability of this outcome, given that match prediction is based solely on this module
- User should be able to see what weights are being used for each prediction module
- User should be able to set new match specific prediction weights

3.4.2.5 Played Match View

After the game has been played, the user will be able to navigate to the played match page and view information summarising the performance of all users who placed the bet, as well as personal feedback for the current user. This view should be more about the retrospective analysis of the users' betting strategies, ie. the weights used for modules, rather than the statistics from the actual match, ie. shots on target, possession, etc. Hopefully, the information presented in this view will allow users to compare their results with the fellow punters and encourage them to analyse their own betting strategy and optimise performance.

These are the requirements related to the played match view.

- User should be able to see general match information
- User should be able to see brief summary of football stats related to different prediction modules and recorded at the time the match was played
- User should be able to see an overview of the website population performance, more specifically performance of users, who committed bet for this match
- User should be able to see visualisation of the website population's choice of prediction weights used for this match
- It should be clear from the view what was the result of a bet for the user. The view should also indicate the user's choice and prediction percentage

3.4.2.6 Dashboard

Dashboard is a key view of the application.

The idea behind the dashboard in this application is similar to online shoppping experience: user saves an item to the shopping basket and can later submit a purchase or cancel it. In our case, user browses through the list of matches on the main page of the application and can save matches to the dashboard for a latter review. The requirements outlined below describe the dashboard functionality in more detail.

- from the overview of the matches on the main page, user can add a match and save it to the dashboard
- user can delete matches from the dashboard
- user can override the system's recommended prediction modules weights (called "betting system settings") using the" betting system" tab in the dashboard. Any next match saved to the user dashboard will be shown with these user default betting settings.
- user can override the user default betting settings and have different betting settings for each saved match. User can save a match with the new weighting percentage.
- After completing the betting settings, user can "commit to bet" a saved match. After that match betting settings cannot be changed.
- If the match is saved in the user dashboard and it changes its status from "unplayed" to "played", application will evaluate user's commit and estimate whether user won a virtual bet or not based on the actual match result.

User can "commit to bet" a match Committed games are marked in the dashboard (red dot) Played games are marked in the dashboard (grey font) Once user committed games has started (observer, onchange on a filed), it should indicate the winnings and write to db Fix the kits

3.4.2.7 Optional requirements

- A simple yet effective notifications system should be implemented. The application will notify users about their betting performace, eg. whether the user won or lost the bet, how did the user do compared to other bettors.
- The system will provide the users with the latest sports news.

3.4.3 Non-functional Requirements

Non-functional requirements specify how the system is going to perform.

- **Usability** - The application interface should be easy to understand and learn for a new user. The navigation of the website should be highly intuitive.
- **Responsiveness** - The application should be fully responsive. The websites will be tested for a variety of screen resolutions and the minimum screen resolution will be 640x960 (DVGA - iPhone).
- **Performance** - Optimising performance will be crucial for the application, as there is a direct correlation between the application response time and the user experience. Performace problems will be detected and eliminated as soon as they appear.
- **Cross-browser support**. The application will be supported on a minimum set of web browsers, such as Chrome, Internet Explorer 9+, Safari, Firefox.
- **Maintainability** - The focus should be on delivering clear and maintainable code. The code needs to be easily understandable by other developers. For this purpose the best practices of software development and the used languages will be utilised throughtout the implementation phase of the project.
- **Extensibility** – The system will be developed with a large-scale application in mind. It should be easy to apply ongoing changes to the project.

3.5 Overall Architecture

The high level components of this system are quite simple.

Design of an application a a whole, overall design (just boxes and lines) Architectural diagram (overview) (aosabook.org/en/moodle.html -example), quite high level

3.6 Choice of Third Party API

After analysing the functional requirements, it became apparent that this type of application will need the latest football data in order to operate correctly. The easiest way to load such data into the application would be to integrate the application with a third party API. The process of finding an

appropriate API for the project will be described in this section.

After brief research, one thing became apparent. Live football data is a very desirable product and therefore it is not easy to find free of charge live football data API. The key problem is that the data the application needs has to be very recent. Real-time data in particular is very expensive, because of its use by the gambling industry for betting on various markets as the games are going on. It is much easier to find free historical football data.

This is a (not complete) list of API providers that I researched about.

- Optasports.com (http://www.optasports.com) Opta is an industry leader. It provides a wide range of XML feeds covering many sports. The feeds include fixtures and results, live scores, live player stats and many more. Data provided by Opta is very reliable and is used by top-notch clients, such as BBC Sports, BT Sport, Sky Sports, as well as many betting providers and newspapers.

- Openfooty (http://www.footytube.com/openfooty/service.php) Openfooty is an interesting project with very detailed API documentation. However, a quick look at the developer forums shows a stale community and many questions about why no one seems to actually be able to get a developer key. Unfortunately, I also did not manage to obtain a key for this API.

- football-api (http://www.football-api.com) is a paid API service but does offer the English Premier League endpoints for free (demo use). The API will restrict by IP addresses and limit calls based on your package. Includes endponts for Competitions, teams, standings, live scores, fixtures and commentaries. See the pricing page.

However, there can be found many high quality APIs that are not for free. how I came to choose the API I am using in the application

3.7 Project Plan

The project progress timetable is presented in the Gantt diagram below. I found it appropriate to set In general, my two main milestones will be completing the first prototype of the application before the Christmas break and completing the second prototype by the end of April, 2015 (this includes all the testing and bug fixes).

The first prototype will have implemented most of the basic features of the application (my development part is broken down into features - viz. the Gantt diagram). The second prototype is the final version of the application; it will include all the planned features and the graphical design. I will try

to make an even progress on the report throughout the whole time available, as it can be seen from the diagram.

3.8 Conclusion

A short conclusion summarising the chapter.

Chapter 4

Application Prototype

Before making a start on of the implementation phase, a lot of effort was put into the creation of the application prototype. Prototyping is a process of developing the initial model of the future application in order to determine the correct application structure, its functionality and the general concept. A prototype is just a model and may differ from the final product.

The project requirements outlined in the previous chapter of this report were used in order to create a mind map representing the navigation between the pages. This helped understanding what exactly is expected to be seen on each page of the website and what is the user journey in terms of the navigation. Wireframes were created for all the pages of the website. For this purpose was used just paper and pencils to aim flexibility.

In terms of the methodology, a hybrid of Agile and the traditional Waterfall approach was used for this project. Speaking of the traditional Waterfall approach, some of planning was made beforehands, for example requirements specifications, use-case diagrams, wireframes, etc. On the other hand, for the whole implementation phase was used test-driven approach that utilises the best of the agile techniques.

In general, Agile methodology focuses on team communication and project transparency. Nevertheless, one of its advantages is an extreme flexibility, therefore most of the basic components of Agile can still be effectively used by a single person. The key feature of the version of Agile adopted for the project was breaking down the project workload into clearly defined units of work, each associated with an iteration, and setting a milestone for each of them. Excel sheets were used for defining the set of tasks for each iteration. GitHub issue tracker was also used as a supporting productivity tool for this project. Code related tasks were recorded as issues for the project GitHub repository. The issue tracker appeared to be a very efficient tool for keeping the focus. TDD or test driven develoment was another key Agile technique adopted for the project. The unit tests covering business logic were

always written before the implementation and the next sprint has not been started unless all tests from the previous sprint passed.

In summary, in this section requirements from the analysis phase will be converted into the system design. First, several design cornerstones (website structure using Mind maps, wireframes, ...) were set in order to complete the initial prototype of the application. Other elements of the application design (use cases, database diagram) were updated in iterations inline with the Agile methodology.

4.1 Mind Map

After defining the set of use cases, I have created a block diagram of the future application. To accomplish this task, was used a program called MindJet enabling creation of a big scale mind maps. , project structure using mindset

The first step was made a block diagram of the project. I used a mind map as brainstorming tool taking the useless into consideration. It helped me to break my application down into separate webpages and blocks and to define the navigation between them.

4.2 Competitors

After all the important requirements and ideas were put on paper, it is worth taking a look at the potential competitors' websites and application. This step is an important stage of an application prototyping process: it allows to learn from the best practices and possibly avoid potential errors.

Basically considered, of course, direct competitors. We studied not only the top social networking sites, but less popular. Delved into a huge reservoir of applications: all they had to learn to understand their behavior, how they communicate with the user, as they present the information.

We also analyzed the experience of indirect competitors. For example, for our section of music is iTunes and other specialized applications. When it comes to photos, then we photographed absolutely everything - in Instagram, Mobli etc. All this has given us an understanding of how to lead the audience in a variety of applications.

4.3 Wireframes

When speaking about prototyping, in the early stages the first choice of many designers is often a piece of paper and a pencil. Sketching has a number of advantages when compared to the use of the editors, such as Fireworks or Photoshop for prototyping. When using editors, it is easy to get distracted by brushing up unnecessary details too early. On the opposite side, sketches offer a lot of flexibility. It is easy to add notes, make small changes or replace an outdated sketch with a fresh one.

In case of this project, each of the sketches represented a separate "view" of the website. The scale of a "view" might differ. For example, some sketches represented a whole page (home page, dashboard page, etc.), others just outlined certain blocks of the website, such as header, footer, user profile container in more detail. I always added a lot of comments to explain the navigation and sometimes expected output. Sketches were one of the most powerful tools I used during the prototype process.

Scans of the drawings First website prototype in a photoshop/Fireworks Early prototype using html and css, using bootstrap should be quick and easy

4.4 Visual Design: Branding, icons, Font

Branding - number of names for the application Bootstrap was used as a framework on the front-end. Flat design is a big trend of the last years. It was decided to use the flat design in order to not distract user from the content. The design is a mixture of free templates and UI freebies created specifically for Bootstrap.

4.5 Use Cases

As mentioned above, use cases and the database diagram presented below, were prototyped in iterations. In this report will be presented a completed, merged version of the set of use cases and the database design. For use cases UML will be used to design in a clear and readable manner.

4.6 Database Design

what database was used Use this link to describe the ORM and its advantages: http://www.aosabook.org/en/sqlalchemy.html

4.6.1 Database Schema

Describe how the database was designed (what we need to capture and how I gradually added table by table). Start with user, as it is the cornerstone of the application (see forthergill) The database class diagram presented is a result of numerous iterations. Present also database before and after.

4.6.2 Database Class Diagram

Chapter 5

Implementation

From a practical point of view, the aim of this project was to create a working web application that also makes use of the best available development practices, well-designed architecture and is easy to maintain and extend in the future. The application is a relatively small-scale one, but it was developed with a future large-scale application in mind, that would support a large number of concurrent requests and stay highly responsive. Therefore, great emphasis was put on the scalability and performance.

This chapter examines the process of SureThing development.

5.1 Choice of Technologies

The application will be built using a set of front end and back end technologies. In this section will be justified the choice made in favour of each particular technology used in the project.

5.1.1 Front End

The markup of the future application will be coded using HTML 5 [21]. The application markup will be built using BEM front end development approach [25]. BEM (short for "Block Element Modifier") is a popular semantic model for markup and a way to organise sections of a website into purposeful blocks and to optimise CSS. The idea behind is to logically break the HTML down into *independent* blocks, which will allow arbitrary placement of the block anywhere on the page, including nesting the block inside another block. The approach can be very beneficial for large websites, allowing the code to be reused across pages or even projects. However, a small project like the SureThing can also benefit from BEM by making use of independent, context-free CSS that can be easily amended in the future [18].

CSS3 is used to define the visual presentation of the application. In general, CSS has certain limitations of its syntax capabilities. For example, it does not allow the use of variables, macros, mixins (reusable blocks of styles) functions and other features associated with object-oriented development, which inevitably leads to the creation of immensely repetitive stylesheets. In order to overcome those limitations, SASS preprocessor [9] will be used in this project. SASS (short for Syntactically Awesome Stylesheets) is a powerful language that extends CSS with a choice of useful functionality, all in CSS-compatible syntax. Use of SASS would allow to make CSS code more efficient and easily maintainable.

In addition to that, SureThing will make use of a popular CSS framework Bootstrap 3 [20]. Bootstrap provides a number of ready solutions for designing the layout of the future application. Therefore, the overall architecture of the markup will be defined by identifying BEM blocks and elements. This would bring structure into the code across all front end technologies used in the development. BEM blocks and elements will be complemented with appropriate Bootstrap classes in order to speed up the development process and make the application fully responsive.

JavaScript, specifically JQuery library [15], will be used to add animations and improve overall user experience from using the application.

In order to handle time-consuming and repetitive tasks on the front end side, the application will utilize the task-based command-line tool Grunt. This software comes with a variety of plugins serving different purposes. For this project will be used *grunt-sass* to compile SASS stylesheets into CSS complemented with *grunt-watch* to allow continuous development, *grunt-css* plugin to combine the all external CSS files into one and *grunt-uglify* plugin in order to reduce the size of JavaScript files and speed up loading of the web page in a browser. This is a screenshot of grunt output for this project running in terminal window.

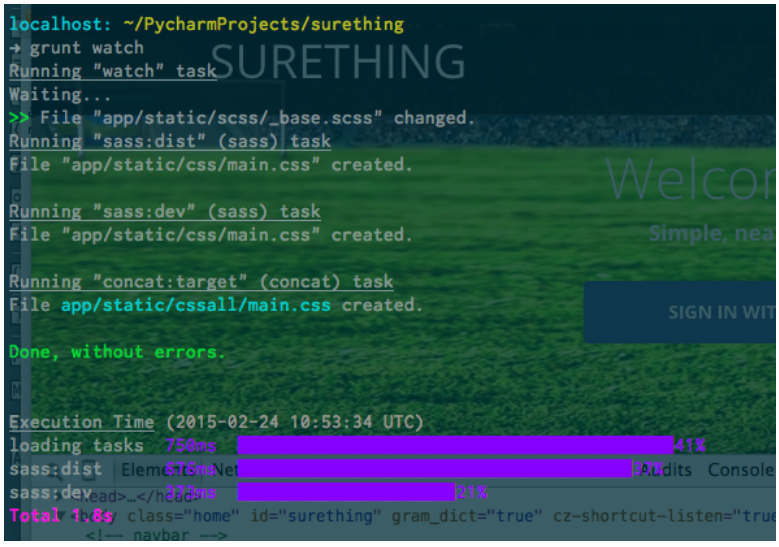

Figure 5.1: A screenshot from the terminal output after running Grunt.

In addition, RequireJS [19], a powerful asynchronous script loader will be used for effective management of JavaScript dependencies. It can load modules in asynchronous manner if desired and thus improve overall website performance.

5.1.2 Back End

For making reasonably accurate football results predictions the application requires latest footaball data. Live data would have to be frequently loaded into the system and processed in an appropriate way. Therefore, there would be a need for at least one separate module dealing with a third party football data API and containing business logic to manipulate the received data. The API wrapper is expected to be integrated into the web application, but separated from the presentation, it also has to be relatively easy to execute as a standalone module, encouraging a nicely decoupled design. Based on the above assumptions, Python was chosen as a primary back end language for this project being known as a language well suited to data manipulation.

The back end of the web application will be built using Python web framework Flask [16]. It is a lightweight framework (the official name is "Python microframework") with a great choice of third-party libraries (e.g. Flask-SQLAlchemy or Flask-Login) that can extend the feature set of the framework core in various ways. Flask application is minimalist to begin with, but it can grow with the project needs. For the purpose of this project this is an advantage compared to the full-featured frameworks like Django that have a lot of functionality already built-in in the basic installation. In addition, availability of developer-friendly documentation and low learning curve makes Flask a short way to get a simple, Python-powered web site up and running. Therefore, Flask appears to be a great choice for a small project like SureThing.

SQLAlchemy was chosen as database solution for this project [2]. This is a powerful database framework that supports several databases back ends and offers the high-level Object Relational Mapper (for short, ORM). Using ORM provides a great level of abstraction when working with databases. For example, SQLAlchemy uses classes that map to each table in a database. This means that the records interaction can be kept the same regardless of the underlying database system. This offers a lot of flexibility and, for example, allows to use different database systems for development and production environment. According to [8], "Flask-Migrate extension, based on a migration framework Alembic and written by a lead developer of SQLAlchemy, provides a powerful solution to handle database alterations and make database schema updates easily manageable"

5.2 Application Architecture

Application architecture is a base of a good quality software. The architecture of SureThing was from a big part dictated by the used framework Flask, that uses a variation of MVC for Python called "MTV" (Model-Template-Controller). Nando Florestan [7] in his blog post describes this pattern in the following way:

> "The template contains HTML content and presentation logic. It is written in a templating language ... It gets data from the view and outputs a web page. The view (also sometimes called "controller"), written in Python, is just glue code. It uses the web framework to put everything together. The model layer is essentially a persistence layer: its most important dependency is SQLAlchemy. The model knows how to save the data, constituting the most reusable code in the entire project."

However, on the top of the standard MVC architecture SureThing requires few extra components to manage the loading of the data from an external source. Hence, this is the final layout of the application architecture:

- **Model Layer.** Contains Python classes that represent database models and related logic. According to Florestan, [7], model layer "represents the essence of your system without the details of a user interface."

- **View Layer.** Holds association between URLs requested in the browser and Python functions or *routes*. Routes are declared using an appropriate function decorator. Below can be found an excerpt from the application code. This is a header of a route function that represents the index page.

```
@main.route('/', methods=['POST', 'GET'])
@main.route('/index', methods=['POST', 'GET'])
@templated()
def index():
```

Figure 5.2: Index page route example.

- **Template Layer.** This is the mediator layer between an HTTP request and the application logic. It consists of a number of Jinja2 templates holding only presentation logic.

- **External Services Layer** [7]. Contains API wrapper class that accesses and manipulates live football data. In general, the functionality of the application can be further extended in many ways. In the future, Football-API might not be the only external source of information. The project might make use of another third-party API or even web scrapping. Eventually, all additional modules related to the interaction with external sources of data will become part of

this layer.

- **Threading.** The application requests live football data frequently. Loading the data is a costly I/O operation that may become a bottleneck unless performed asynchronously. Threading component of the apllication defines a class *DataUpdateThread* that takes care of writing the data to the server every 100 seconds. This task is performed in a separate thread.

During the development process, a lot of effort was put into keeping the Template Layer as thin as possible in order to reduce the loading time in the browser and improve the overall performance of the application.

5.3 Patterns And Conventions

Flask offers an excellent extendable core of functionality; its API is also very minimalistic and easy to understand. One of the main advantages of this framework is that it gives a developer a lot of freedom to decide how to structure the application. As Matt Wright [24] puts it: " without patterns or conventions your applications will loose architectural integrity and be difficult to understand by others". In this section a number of various patterns, conventions and tools used during application development will be described and explained.

5.3.1 Application Factory

The use of factory pattern is crucial to a Flask application. For example, SureThing app defines various configurations to be used in different environments (development, testing, production). However, because the application instance is created in the global scope, there is no way to apply those configurations. The problem is that "by the time the script is running, the application instance has already been created, so it is already too late to make configuration changes." [8] To get around this problem a creation of the application was moved into a separate function, *create_app()*. The name of configuration name is passed into the function as a parameter. This solution also allows us to create multiple instances of the application and make testing of various confurations easier.[17]

```
def create_app(config_name):
    app = Flask(__name__)

    #loading configurations into the app
    app.config.from_object(config[config_name])
```

Figure 5.3: Function responsible for creating the application instance.

5.3.2 Blueprints

Blueprints are related to the View Layer introduced in the section Application Architecture [**?**]. A large application is divided into smaller parts and each part is implemented with help of a blueprint. This concept helps to develop a *modular* web application. SureThing was divided into two parts: *main* and *auth*. *auth* holds the endpoints associated with the authentication and user profile related tasks, for example *login()*, *edit_profile()*. On the other hand *main* is in charge of the rest of the application. Notice how those different blueprints are registered on the application instance inside the *create_app()* function:

```
#attach routes and custom error pages here
from main import main as main_blueprint
app.register_blueprint(main_blueprint)

from .auth import auth as auth_blueprint
app.register_blueprint(auth_blueprint, url_prefix='/auth')
```

Figure 5.4: Blueprints registration.

5.3.3 Database Migrations

The database scheme for this project was designed in an iterative way, models and relationships between them were added gradually as the application was growing. Therefore, it was crucial to find a tool that allows effortless updates of the database. To manage frequent database updates was used Alembic database migration tool that was developed specifically by Mike Bayer, the author of SQLAlchemy. The tool can be added to Flask as an external plugin, Flask-migrate. After installation and initial configuration of the plugin, it allows to migrate the database with two simple commands to be subsequentially run in the terminal: *db upgrade* and *db migrate*. Alembic makes migration easier and prevents the developer from the necessity to delete and recreate the database each time there is a need for migration.

5.3.4 Exceptions

In general, it is considered a good practice to take advantage of standard libararies of a programming language, in our case Python. First of all, it allows developer to save time implementing a piece of functionality from scratch. Secondly, it makes it easier for other developers to read and maintain the code. Exceptions are built into Python at the language level. Using them will lead to cleaner code with no major performance impact. "In a way, try blocks are like transactions. [The] catch has to leave ... program in a consistent state, no matter what happens in the try. For this reason it is good practice to start with a try- catch- finally statement when you are writing code that could throw

exceptions. "[**?**]

SureThing will make use of Exceptions in order to identify and manage failures when making an API call over HTTP. The FootballAPIWrapper class has a private method that calls the API and collects the data in a JSON format from the remote server: *_call_api(action=None, **kwargs)*. The method takes into account the possibility of errors occuring during code execution. When the program calls the API, either JSON data is returned or an Exception is thrown.

As it can be seen from the method definition above, one of the required parameters is *action* that is set to None, unless the value is passed in during the method call. Action is a string that needs to be added to the base url in order to indicate the set of data that is being accessed. Specifying the action is required by the API and the possible values of the parameters (actions) are: competition, standings, today, fixtures, commentaries. For example action *today* will returns the matches scheduled today. The *_all_api* method will raise an Exception, if the action is not being supplied. Various exceptions are thrown when the program is attempting to connect to the remote server [4]. Python library *requests* is used to take care of this type of errors. [14] If the domain name does not resolve, the HTTP request will fail before we establish connection. In that case, the program will throw a *requests.exceptions.ConnectionError*. If the remote server is not functioning or the request is structured incorrectly, the server will respond with an bad response status code and the *_call_api* method will raise a *requests.exceptions.HTTPError*.

5.4 Other Implementation Processes

In this section a number of other implementation processes and used tools will be discussed.

5.4.1 PEP8

5.4.2 PyLint

5.4.3 Version Control

Git version control system was used throughout the development process. Git is known for being a very useful tool for collaboration across teams of developers. However, it has also many benefits for a solo developer. For example, it helps to track changes and restore previous versions of a project, as well as view the code at any point in the past. The project codebase was uploaded to GitHub that is [22] "a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features... (It also)

provides web-based graphical interface". For this project GitHub issue tracker was used as a "to-do list" to keep the record of tasks ("issues" in GitHub terminology) that needed to be completed in each agile iteration. Custom labels were used to distinguish different types of issues in the GitHub issue tracker, for example, performance, design, bug, optional tasks, etc.

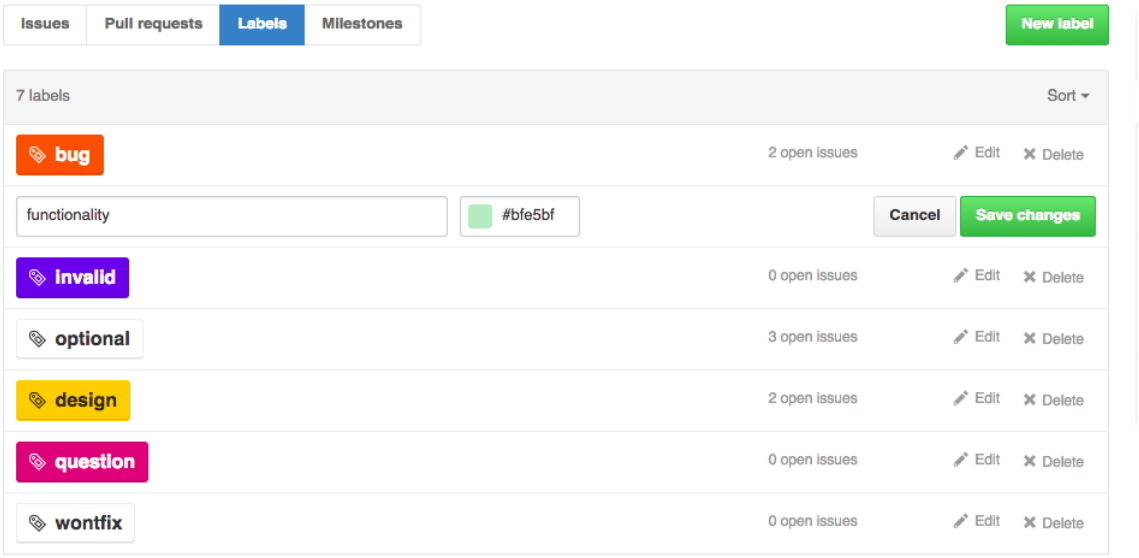

Figure 5.5: GitHub. Custom labels.

GitHub also allows the users to filter out the issues of a similar type, based on the assigned label.

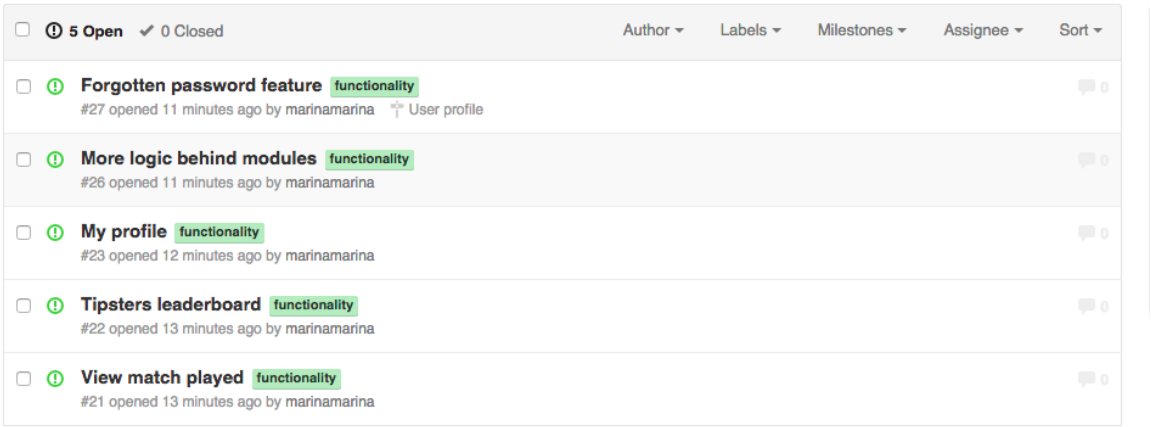

Figure 5.6: GitHub. Issues related to the functionality of the application.

Critically speaking, I missed the option to assign issues various level of importance and order the issues based on their priority.

5.4.4 Own Validation in Forms

Flask-WTF is a Flask extension that offers integration with WTForms and it was used to handle forms in this project. In order to make sure the application is secure, the validation has to be implemented preferably on the server side or both on the client- and server-side of the application. WTForms has many built-in validators that can simplify developer's life. For example **DataRequired** makes the input field mandatory, **Email** checks that the provided input is a valid email address, **EqualTo** helps to ensure that the passwords in the fields "Password" and "Confirm Password" supplied during the user registration are identical. However, sometimes the built-in functionality does not cover all the application needs. In that case, there is an option to create a custom validator that is a basically a Python function returning another function (a validator) that throws an exception every time the user violates the prescribed validation rule. Custom validators can be imported into the module describing forms and used in the same way as a built-in validator would be used. I have separated the validators out into a separate module. The set of custom validators can be further extended, however, there is just one at the moment: *validator_user_already_registered()*.

```
# Custom validators
def validator_user_already_registered():
        """custom validator used in the registration form"""

        def _user_already_registered(form, field):
            kwargs = {field.id: field.data}

            if User.query.filter_by(**kwargs).first():
                raise ValidationError(message='User with this ' + str(field.id) + ' is already registered!')

        return _user_already_registered
```

Figure 5.7: Validator function that checks if user with this username has already been registered with the SureThing application

In the example above the validator function checks if a user with provided username is already in the database. If the user is found, the ValidationError exception is thrown and the new user is prevented from submitting the form.

5.4.5 Custom Macros

Jinja2 is a default template engine that comes in one package with Flask. It is also one of the most widely used template engines for Python. In order to add some extra presentation logic to our application, custom macros can be used. Jinja2 macro is simply a template function that can be used within HTML in order to avoid developers writing repetitive code. For this project I found macros extremely useful. Custom macros were separated out into a separate template file _macros.html.

One of the example usages was rendering form fields. Each form field related macro contained a piece

of HTML code specifically designed for forms in this application, as well as logic for dispaying error messages. Among this type of macros can be named *render_field(field)*, *render_checkbox(field)*, *render_submit_field(field)*. Some of those macros needed to be adjusted to enable their usage in a specific view. These are the examples of such "adjusted" macros: *render_submit_field_match_preview(field)*, *render_embedded_field(field)*. For example, *render_field* takes care of rendering any standard input field accross the application, whereas *render_embedded_field(field)* manages rendering an input field embedded within a prediction module on the Upcoming Match View.

Another macro, *teamkitimage(match, home=1)* renders an image representing a football club. Based on the provided arguments, the function displays an image of a home or away team kit for a specific club.

5.4.6 Integration with third-party API

Many production Python web applications rely on external application programming interfaces (APIs). API can be also reffered to as "third party services" or "external platform".[**?**] SureThing requires constant access to current football data. After choosing an appropriate API, it has to be integrated into the application.

There is a variety of tools available for developers for accessing web APIs. Those three options were considered when choosing an appropriate tool:

- Helper library (such as Runscope or Apiary) Using a helper library has an overhead of learning how to use another piece of software.
- urllib2, standard Python module *urllib2* module offers very simple implementation and provides most of the required HTTP capabilities, but the API is thoroughly broken and features critical for performance are missing, for example connection re-using/pooling.
- urllib3
- requests, another Python library for handling HTTP requests. It offers a lot of control over the HTTP calls through the use of its powerful features.

After some experiments with other urllib2, urllib3 and requests, *requests* was chosen as the libarary for this project.

All interaction with the Football-API, including processing the received data, was separated out into a module *football_api_wrapperpy* or just "wrapper" for short. This module contains only one class, FootballAPIWrapper. Fields of the class accomodate the key elements of the interaction with the

API that will be re-used in different methods of the wrapper, for example base url, path to the data directory.

```
def __init__(self):

    self._premier_league_id = '1204'
    self._base_url = 'http://football-api.com/api/?Action='
    self._basedir = os.path.dirname(__file__)
    self._datadir = os.path.abspath( os.path.join(self._basedir, '..', 'data') )
    self._proxy_on = False
```

Figure 5.8: Football API wrapper, fields

To get the response from Football-API takes about 11s, therefore it is nessessary to move the API calls into a task queue so they do not block the HTTP request-response cycle for the rest of the web application.

5.4.7 Visual Effects

A number of visual effects was implemented with the help of jQuery, AJAX and the Websockets (Flask extention Flask-SocketsIO) to improve the user experience.

A good example is an alert dismissal. SureThing generates many alerts that give the user feedback with regards to the action they have just taken. User has to dismiss the alert manually each time. To avoid this extra thing for a user to do, the alerts are removed from the page using jQuery fadeOut() and then slideUp() animation methods that fades the pop-up and then removes it with a sliding motion after 200ms of its appearance on the page. For compactness the screenshot shows the page opened in a browser simulation of an iPhone screen size. Responsive web design with regards to this project will be examined in more detail in the following subsection.

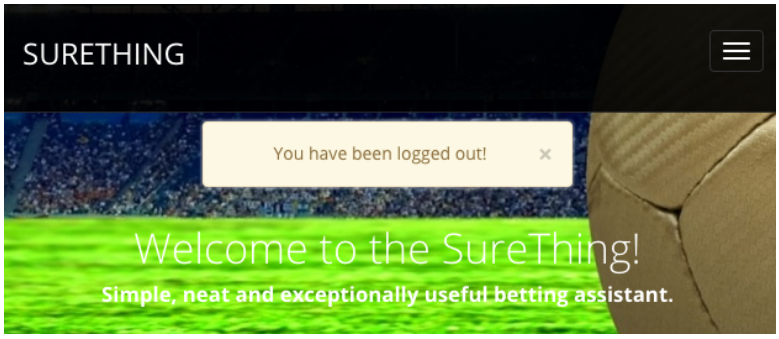

Figure 5.9: An example of an alert on the page appearing after user has logged out

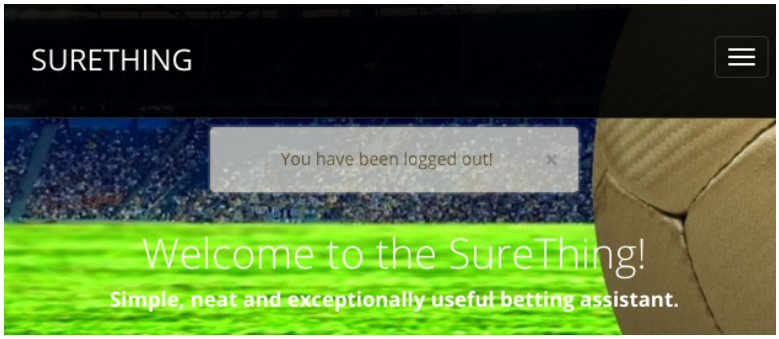

Figure 5.10: The same alert fading out.

Another example is a new message notification. Once user received a new in-app message notifying them about the results of their bets, the envelope-shaped icon on the top navigation menu that represents the in-app Inbox turns orange.

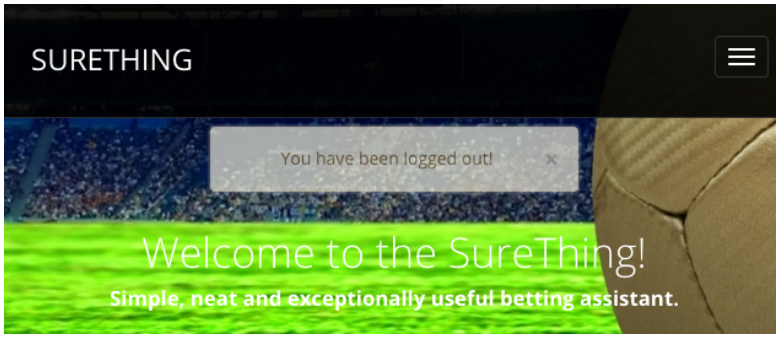

Figure 5.11: The same alert fading out.

When user navigates to the Inbox and reads or deletes all the new messages, the icon immediately turns grey notifying that there are no more unread messages in the box.

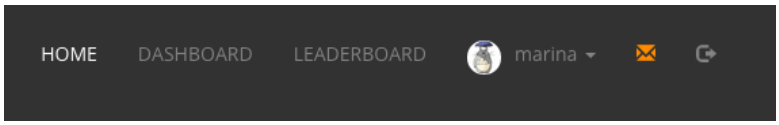

Figure 5.12: You have new messages. Desktop View.

This is what the same top navigation menu looks like for mobile users.

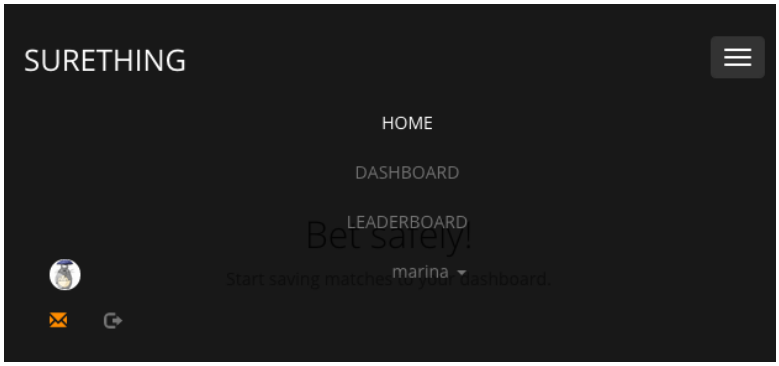

Figure 5.13: You have new messages. Mobile View.

User has opened the last unread message and the message icon has turned grey. Notice, how the top navigation menu partly covers the message view. The menu rolls back into a compact mode once the user clicks on the menu icon in the top right corner.

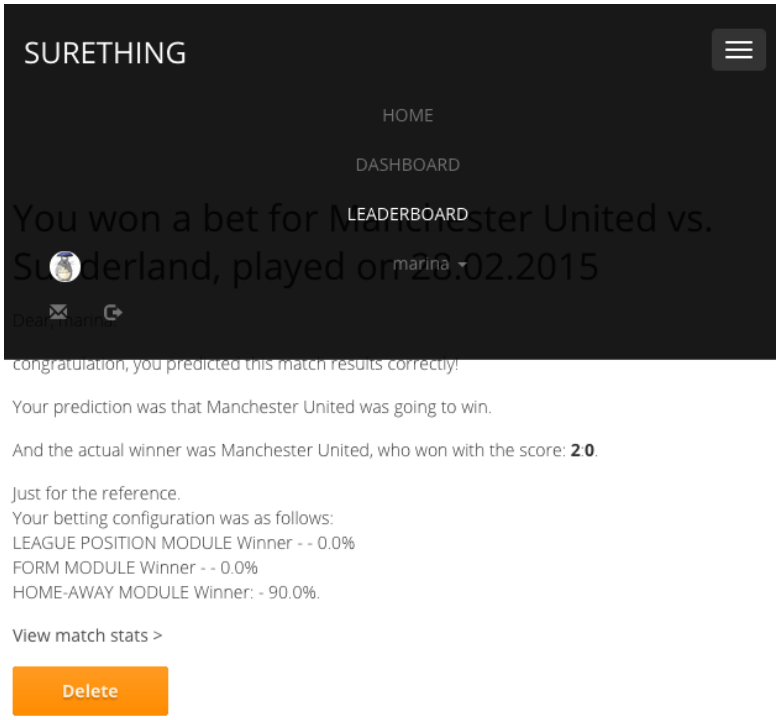

Figure 5.14: User just read the last new message, the icon turns grey.

5.4.8 Responsive Design

SureThing is a fully responsive web application. The application utilizes Bootstrap3. see Appendices

5.5 Features Implementation

In this section will be described the technical details of the project implementation. Each susbsection is bound to the high level feature of the application, as indroduced in the chapter "Requirements Analysis" 3.4.2. "User journey", or possible user interactions with the system, will be demonstrated for each view.

5.5.1 Authentication and User Profile

The application requires authentication functionality. In order to simplify the development process, one of the useful Flask's extentions, Flask-Login, was utilised to handle the common tasks of logging in and out, as well as new users registration. For the new users the application offers a registration

form. On form submission, the application sends user an email with verification token, expecting them to confirm the email address and complete the registration. It should be noted that both username and the email address provided during the registration should be unique, otherwise the application throws a validation error.

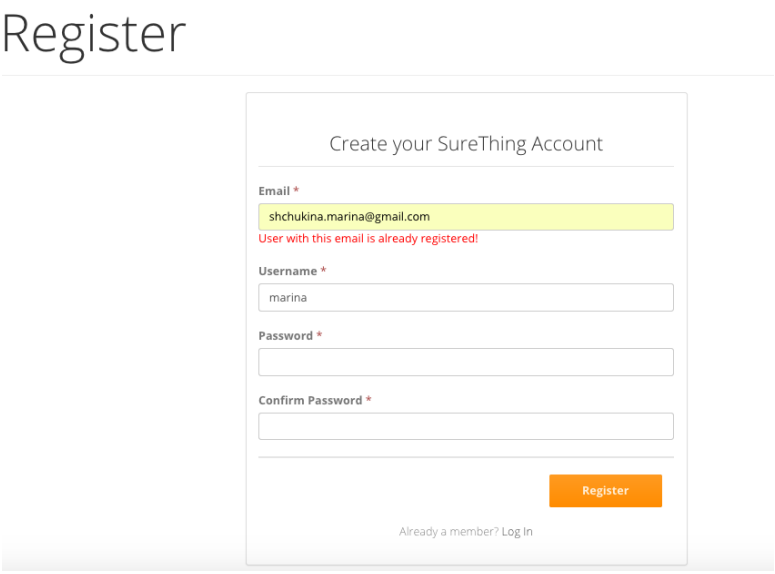

Figure 5.15: SureThing offers registration form for the new users. However, the email address is expected to be unique.

5.5.2 Matches Overview

Matches Overview is a view displayed on the main page of the application. It contains lists of upcoming and played matches and the user can toggle between those two lists using navigation buttons. In both lists matches are grouped by dates as follows:

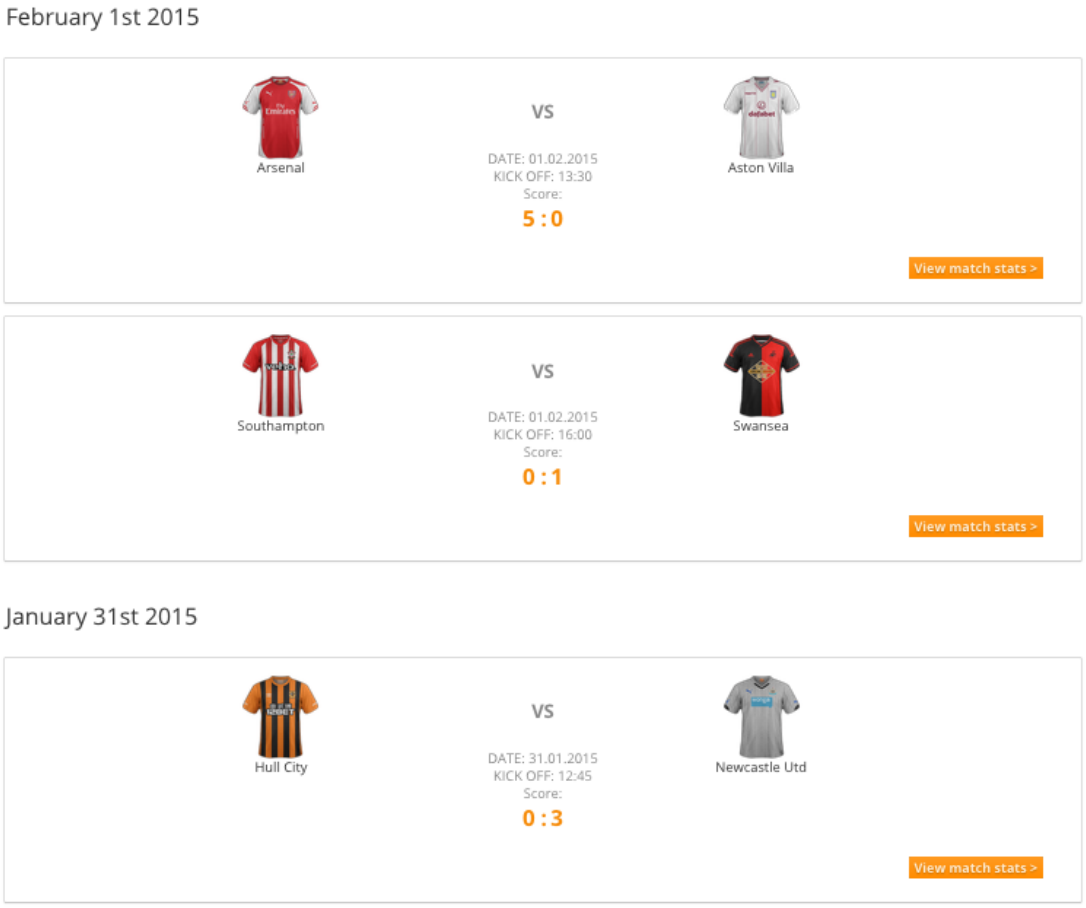

Figure 5.16: Matches in the overview are grouped by dates sorted in ascending order.

Each panel representing a match contains the most basic information about the event, such as names of participating teams, kick-off date and time. The screenshot below is an example of a panel displaying an unplayed match. The navigation buttons can be seen just above the first match in the list. Match already saved to the Dashboard by the authenticated user is indicated by a floppy disk icon on the right hand side of the panel.

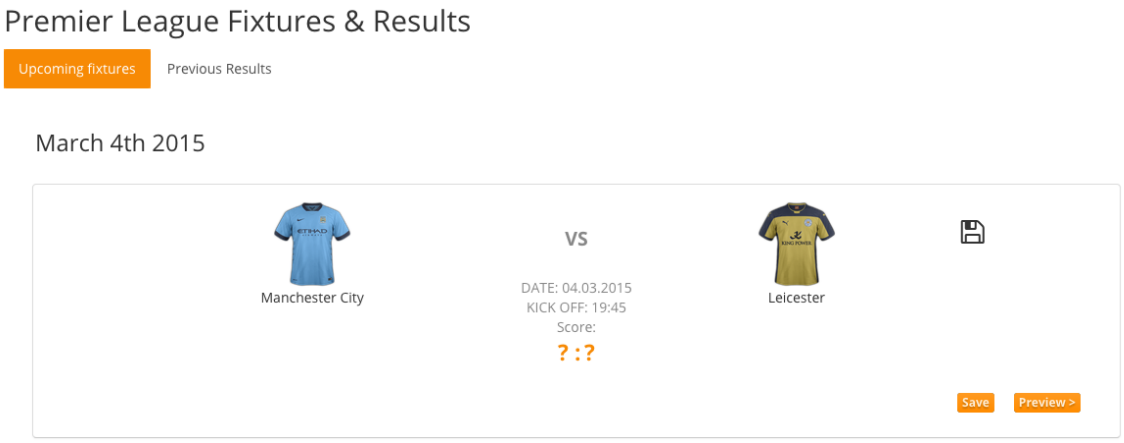

Figure 5.17: Example of an unplayed match displayed in the overview.

Below can be found a screenshot of a played match panel. Notice that the final score of the match is

displayed and instead of two buttons "Save" and "Preview" there is only one button - "View match stats".

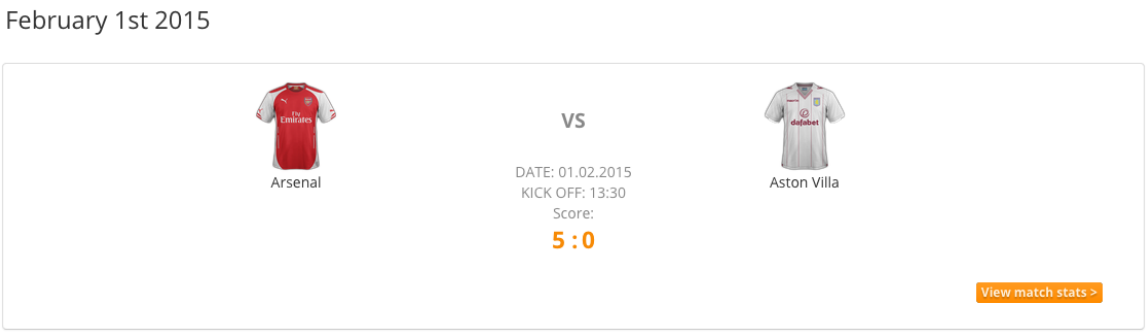

Figure 5.18: Example of a played match displayed in the overview.

In case the match is being played at the very moment, it still belongs to the list of "unplayed" matches and is displayed with a small badge "LIVE", indicating live event. A match is considered as "played" as soon as the full time score is available. Hence, it is possible to make "bets" until the moment the match is considered "played".

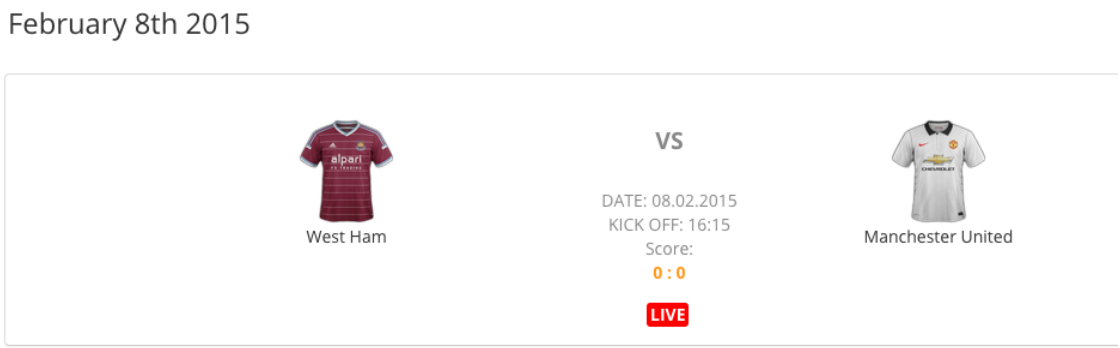

Figure 5.19: Example of a live match displayed in the overview.

User Journey

Just above the list can be found simple navigation allowing to switch between the lists of unplayed and played matches. On the right hand side of each list item there is a "Preview" button for an upcoming match and a "View match stats" button for a played match. By clicking those buttons user can navigate to views with more detailed information about the match (*Upcoming Match View* or *Played Match View*). User can save the match to the dashboard by clicking "Save" button. This action can be carried out only for an unplayed match.

5.5.3 Upcoming Match View

Implementation of this view was one the most complex development tasks of the whole project. This is the essense of SureThing - view allowing the user to predict match results.

Authenticated SureThing user can navigate to this view either by clicking a "Preview" button on an upcoming match panel in the *Matches Overview* or by clicking the same button on a saved match panel in the user *Dashboard* (if the match has already been saved by the user). Depending on the user route to this view, the Upcoming Match Preview will be displayed differently.

Read-only mode

If the user is coming to the Upcoming Match Preview from the *main page*, the view will display the match header (containing general information about the teams, last played game, match kick-off time and date, etc.) and a list of prediction modules with **prediction values** calculated based on the relevant piece of statistics for each of the teams. These are the prediction modules available in this view:

1. Module League Position
2. Module Form
3. Module Home/Away

"Save" button can be found at the very bottom of the view.

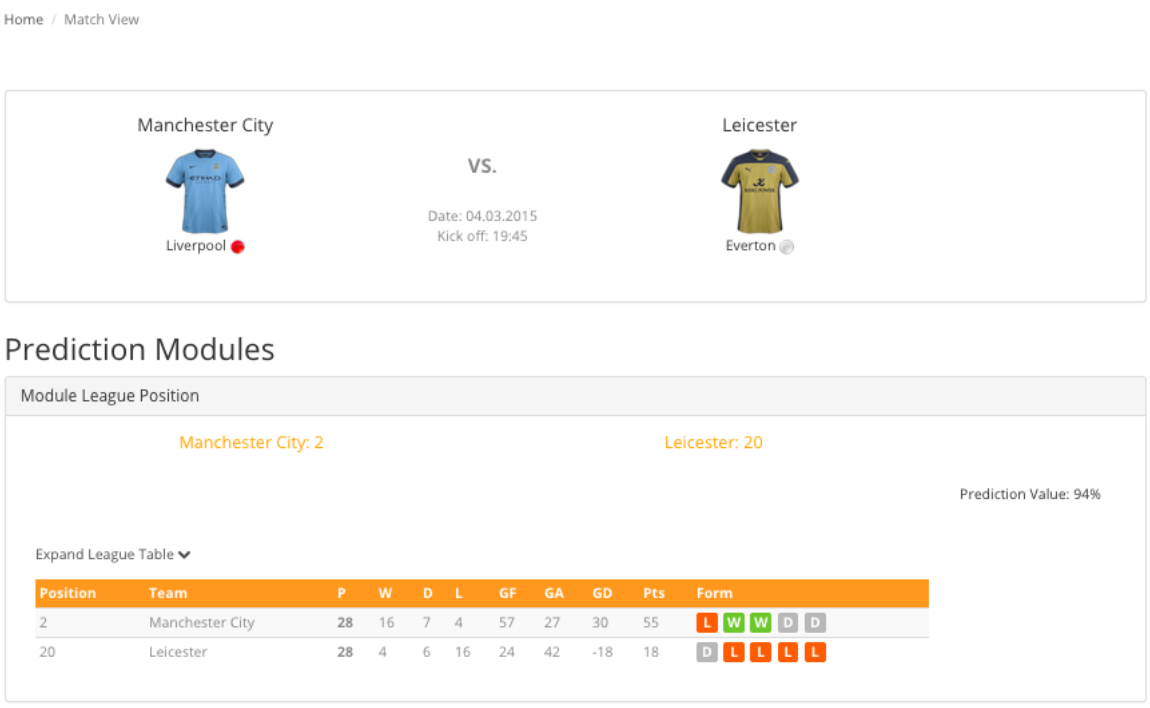

Figure 5.20: Upcoming Match View in the "read-only" mode: match header and the first prediction module, Module League Position.

This information should be sufficient for the user to decide, whether it is worth saving the match to the dashboard for a later revision. An unauthorised user would be able to see the same content, but the "Save" button will be disabled. We can say that if user navigates to the Upcoming Match View from the main page, they can see the view in the **read-only mode**. It is also important to note that this is one of few views that are available for an unauthorised user.

Prediction mode

In case user has already saved the match to the dashboard and navigates to the view by clicking on a saved match panel, the view will enable the prediction feature. This time the view is displayed in the **prediction mode**. In each of the prediction modules user will be able to see an embedded input field with weight percentages inside the field. Below can be found a screenshot of a same part of the view as the one above, displayed in prediction mode.

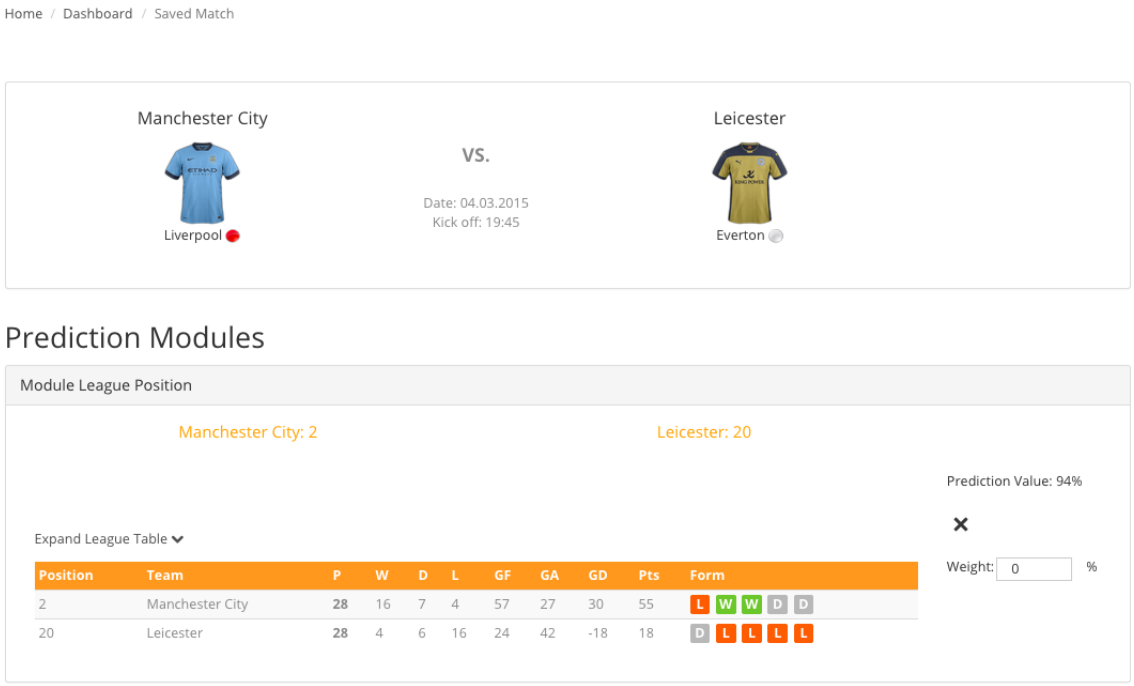

Figure 5.21: Upcoming Match View in the "prediction" mode: match header and the first prediction module, Module League Position.

The values displayed inside the fields will be used for calculating the overall match result prediction. If this is the first time user previews the match, the prediction weights will be either the **system default** (in case user has not set the prediction settings in the dashboard yet) or the **user default** prediction settings. If user has already visited this page before and set the match specific settings, the values displayed inside the input fields will be the **match specific** ones. Any module can be eliminated from the prediction by setting its weight to 0%. The total sum of prediction weights must equal to 100%.

These are the prediction modules available to the user in the prediction mode:

1. Module League Position
2. Module Form
3. Module Home/Away
4. User Hunch

Notice, the extra module is available in this view - User Hunch. This module is very important to the prediction process. User can personalise the prediction by choosing Home, Away or Draw value in the User Hunch module panel. The module will be explained in more detail in the subsection, "Prediction" [**?**]

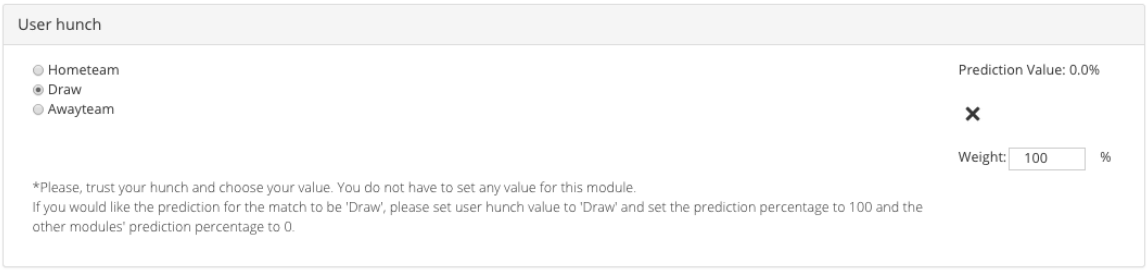

Figure 5.22: Module User Hunch.

The overall prediction is displayed underneath all modules, in a separate panel.

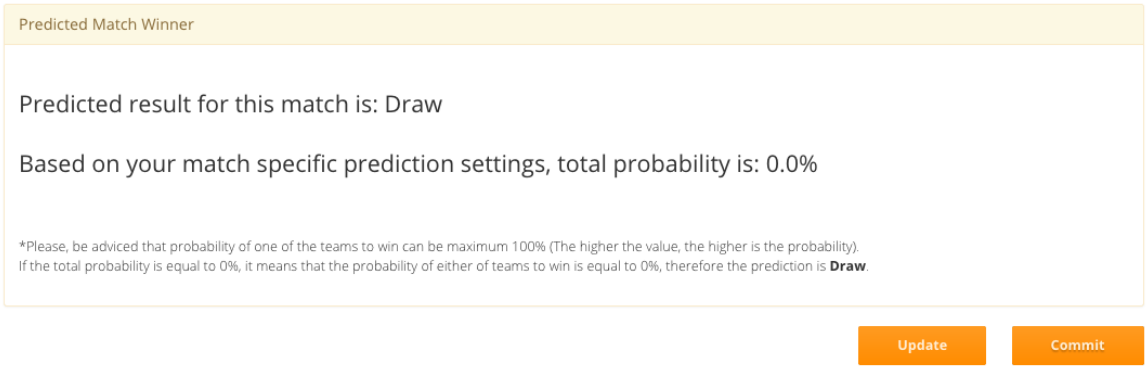

Figure 5.23: Match result prediction.

User journey

In the **read-only** mode the user can save the match to the dashboard by clicking "Save" button.

In the **prediction** mode the user can update the prediction settings by overriding the current values in each of the input fields and pressing the button "Update" that is located at the bottom of the view. The overall prediction output will change every time the user updates the prediction settings or changes the hunch value. Once satisfied with the prediction outcome, the user can commit the match by pressing "Commit" button that is located next to "Update". Before the end of the match, the user can still navigate to the committed match view and see the all the details, including the prediction values, used weights and the final prediction. However, this time "Update" and "Commit" buttons will be disabled.

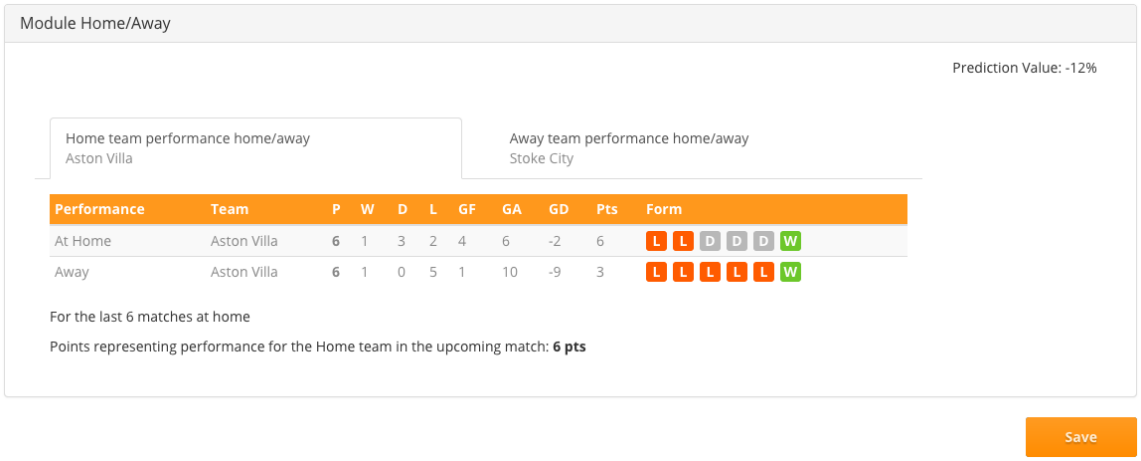

Figure 5.24: An example of a prediction module in the Match Preview, user navigated from the main page.

If the user comes to the upcoming match preview from the *dashboard*, they will be able to see more information related to the actual result prediction and betting. First of all, in each prediction module they will see an input fields for setting match specific prediction weights. Secondly, they will see a user hunch module. Finally, at the very bottom of the overview they will see calculated prediction result and two buttons - one to save the match specific settings and another to commit the bet.

most Explain how was implemented user hunch: combination of Flask Ajax and Sockets IO!!!

5.5.4 Prediction

After getting familiar with the *Upcoming Match View*, the next logical step is to introduce the reader of this report with the Prediction feature of the application. This will aid understanding how to use the *Upcoming Match View* and what kind of information is available to us in this view.

The implementation of the Prediction feature was already briefly outlined in the chapter "Requirements Analysis", subsection "Definitions" [**?**]. In this subsection I would like to decribe this key feature of the application in more detail. The application has three levels of prediction settings (weights). Firstly, it is the "default prediction settings" - a set of weights recommended to new users by the system. Once the user is registered with the application, they can set their own set of weights that will override the default settings. From the moment those weights are saved in the database, they will apply to every newly saved match. The application also allows users to set match specific settings that will only apply to one match.

how the overall prediction is calculated, weighted prediction

5.5.5 Played Match View

User can navigate to this view by clicking the button "View match stats" on a panel representing a match that has already been played. Unlike the *Upcoming Match View*, the *Played Match View* looks the same to the users coming to this page both from their dashboards and from the matches overview page.

The view contains already familiar match header, prediction statistics and a personalised feedback for the authenticated user.

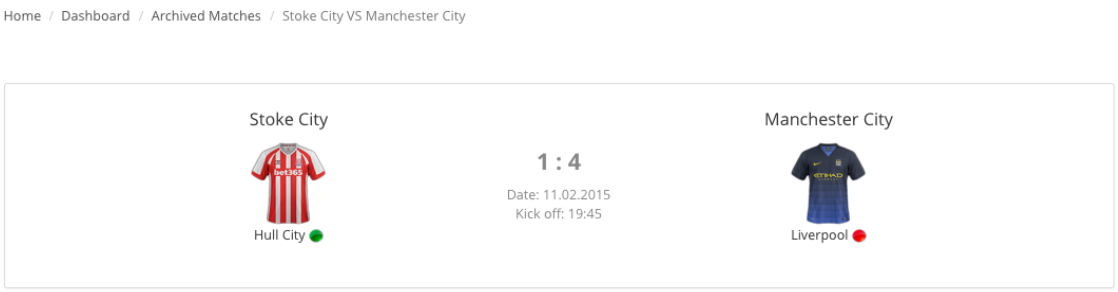

Figure 5.25: Played Match View, match header.

Prediction statistics block contains information on the betting performance accross the SureThing users with regards to this match. Stats contain the information on the number of users who saved the match to their dashboards, made a bet on the match, won or lost the bet.

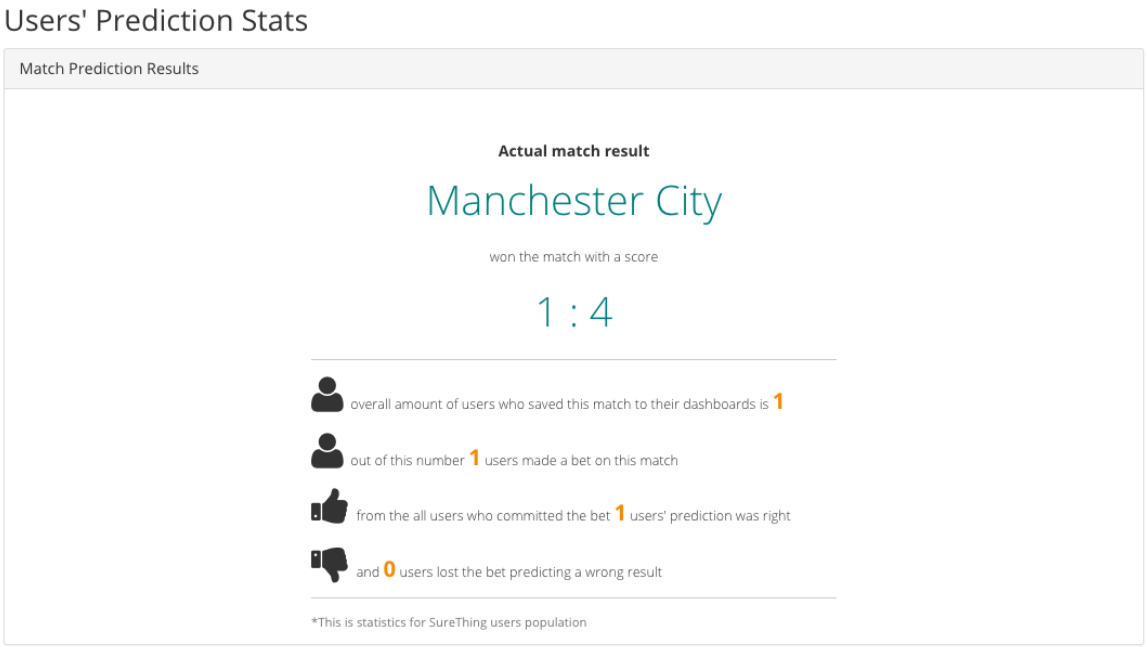

Figure 5.26: Played Match View, users' prediction statistics.

The view also displays a bar chart illustrating a breakdown of the result prediction, namely how many

users bet on "home", "draw" and "away" result.

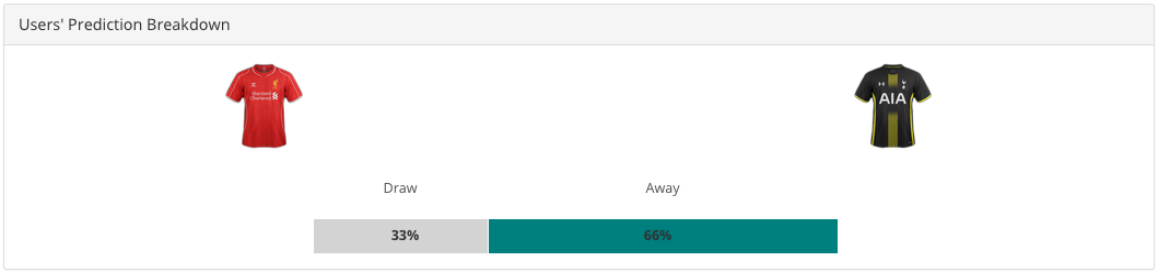

Figure 5.27: Prediction breakdown.

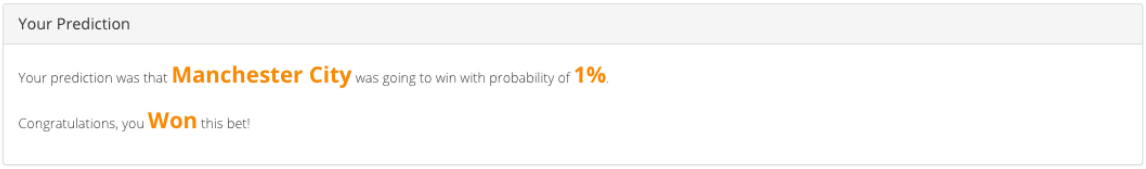

Figure 5.28: Played Match View, feedback provided for authenticated users.

5.5.6 Dashboard

SureThing dashboard is a centralised user space that provides convinient shortcuts to all the important prediction-related pages and tools. Dashboard can be used to store and view matches, change default prediction weights and make bets. It can be navigated to by clicking on item "Dashboard" in the navigation menu of the website, located at the top of the page. On navigating to the dashboard user can see a list of saved matches ordered by date and a *dashboard menu* icon in the top left corner of the view. This is a screenshot of a typical dashboard view (only one match is saved):

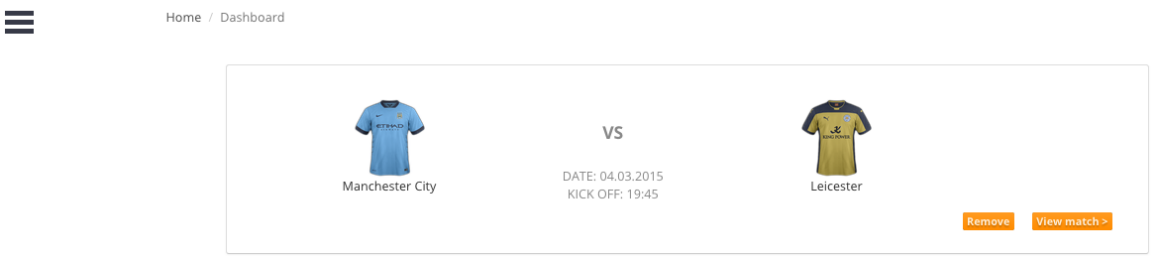

Figure 5.29: Dashboard view.

Matches that already have been committed by the user have grey background and the predicted winner is highlighted. If the match has already been played, the result of the bet, either "Win" or "Loss" also appears on the match panel.

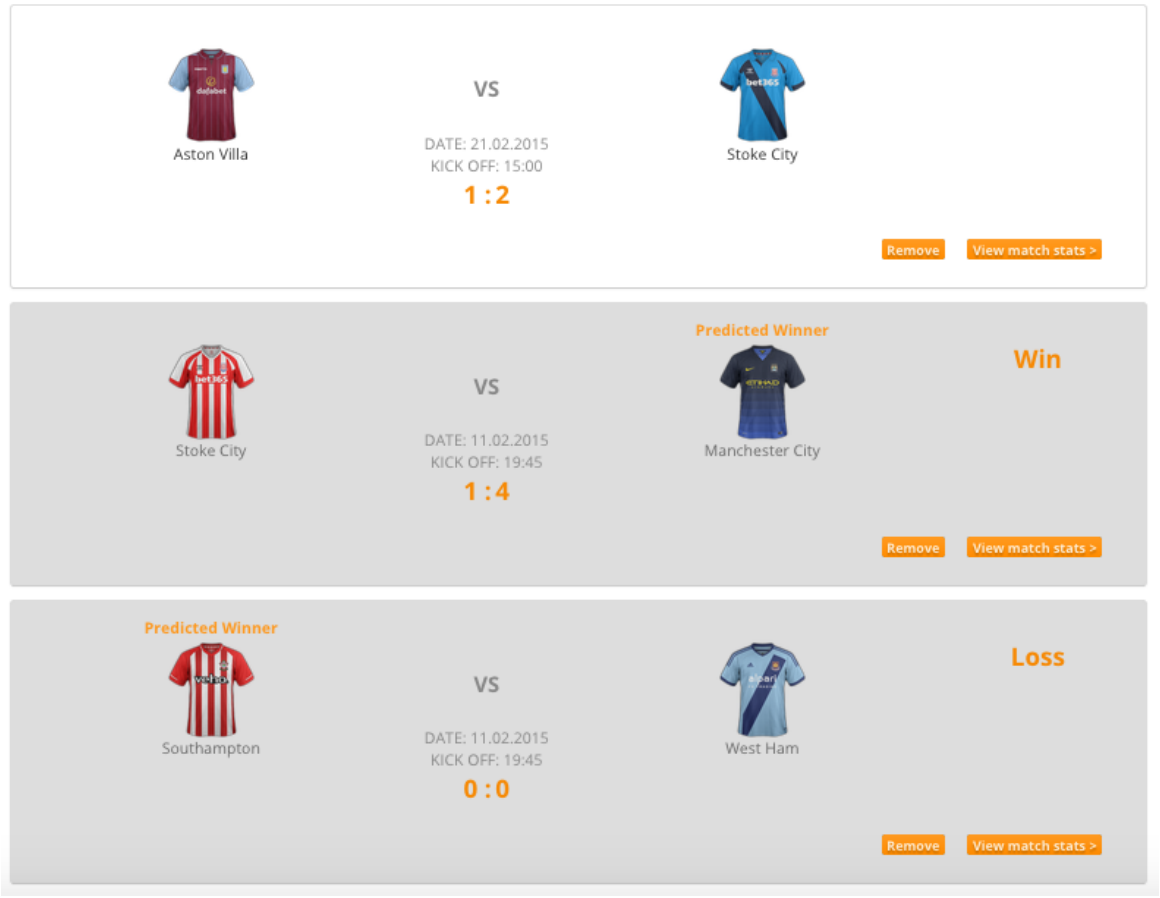

Figure 5.30: An example of a Dashboard View with matches that have been committed and played.

In case user does not have any matches saved, the dashboard looks as follows:

Figure 5.31: Dashboard with no saved matches.

In the top left corner user can see a small menu icon representing the dashboard menu. Clicking the button opens up the menu containing three items:

- Upcoming Matches. Displays all saved matches that have not been played yet.
- Archived Matches. Displays all saved matches that have already been played.
- Prediction Settings. A form that allows the user to set default prediction weights.

A panel containing the dashboard menu slides in and covers part of the page. It can be easily dissmissed by choosing an item from the menu or clicking the ”close” icon.

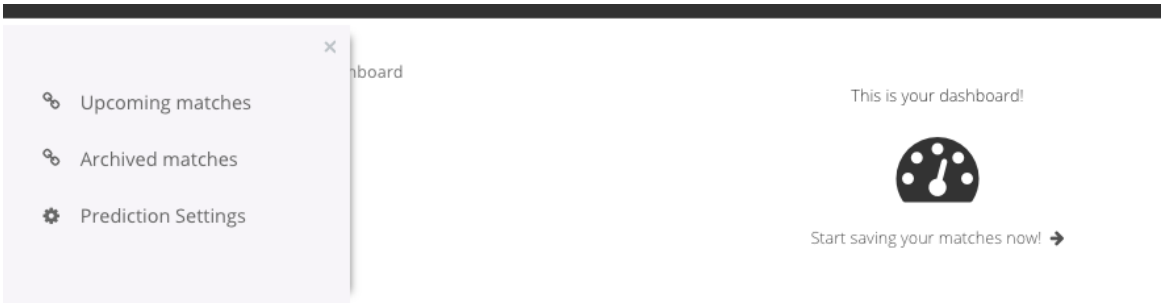

Figure 5.32: Dashboard menu.

User journey

5.5.7 Notifications

This feature represents the one-way communication flow from SureThing to the application user. Every time a match previously commited by the user is finished (from the technical point of view, the match is changing its status to "played"), the application sends users messages notifying them whether their prediction guess was successfull. Such messages contain detailed information about the user prediction, as well as a link to the relevant *Played Match View*.

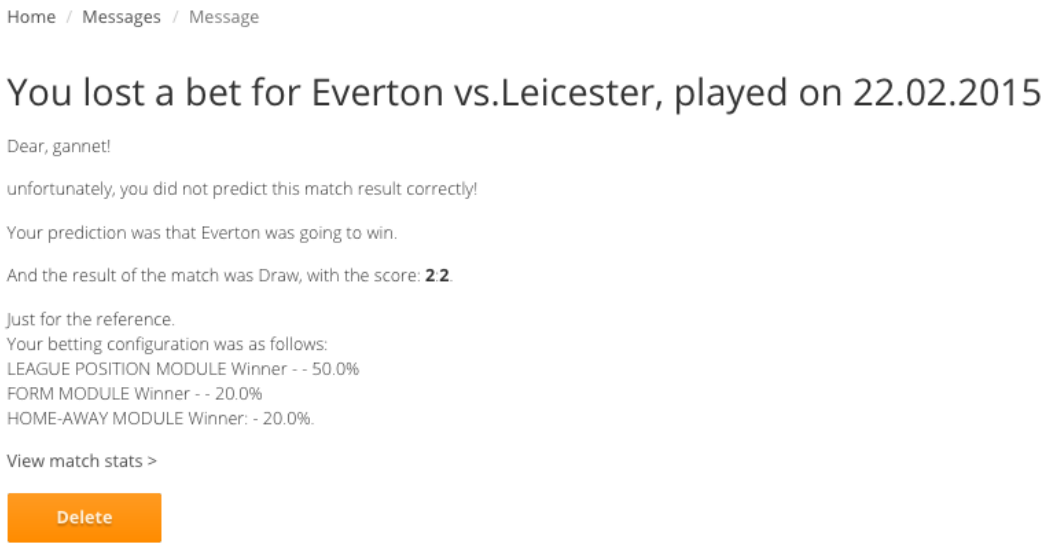

Figure 5.33: An example of a message sent to the user.

User journey

Authenticated user can navigate to the notifications inbox by clicking at the envelope-shaped icon located on the right-hand side of the navigation menu. The orange colour of the icon in the screenshot below indicates that the inbox contains unread messages, otherwise the icon color is grey.

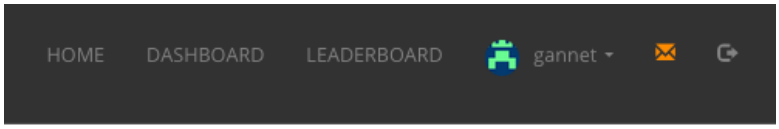

Figure 5.34: Navigation menu panel with an inbox icon.

On clicking the icon user is taken to the notifications inbox. Unread messages are displayed in bold font.

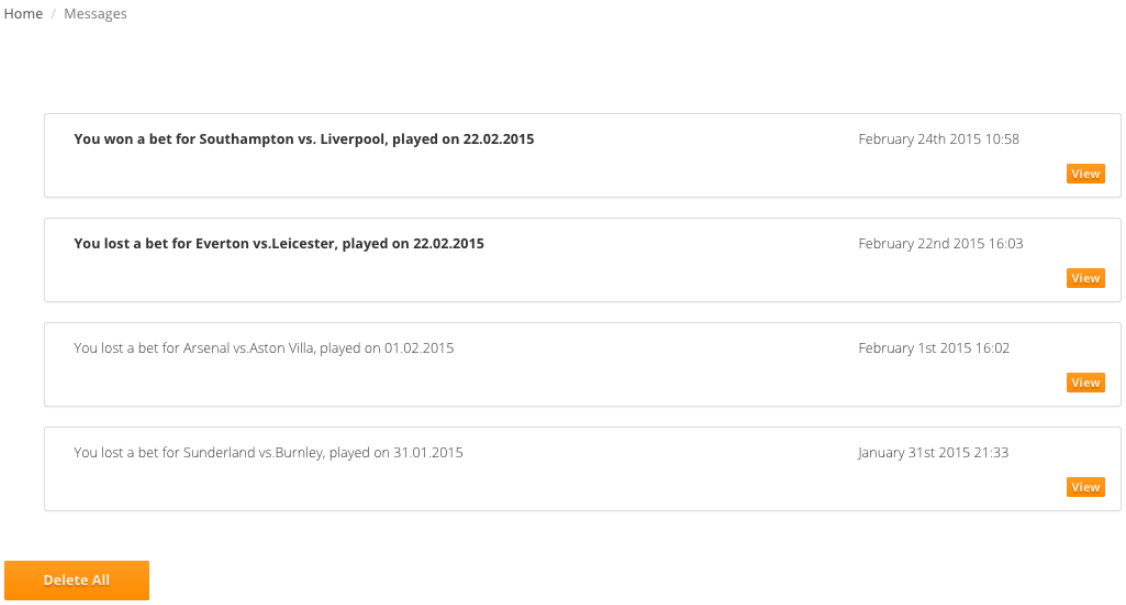

Figure 5.35: User inbox.

5.5.8 Leaderboard

The Leaderboard is a view that offers a table capturing betting performance accross the population of the website. This page can be navigated to by clicking a "Leaderboard" entry in the navigation menu of the application. The Leaderboard displays the most basic information about application users: usernames, location and a favourite team. It also contains the bets related statistics: games committed, won and lost. The table is ordered by the amount of win points of each user.

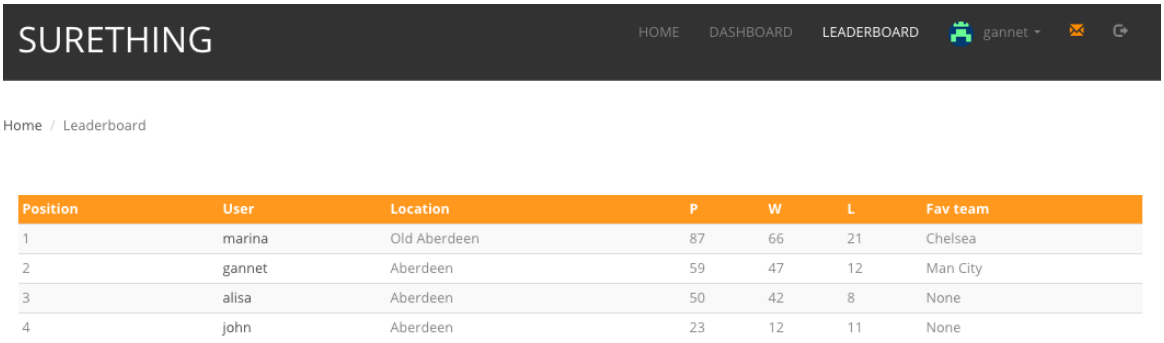

Figure 5.36: Leaderboard.

5.6 Application Performance

Performance is a very important aspect of a web application. DEF

response time

sockets, threads multithreading how I fixed performance on Match.update_all_matches

5.7 Deploying the Application

Cloud Deployment is the most recent trend in application hosting. The formal name of this technologu is Platform as a Service (PaaS). In the PaaS model, a service provider offers a fully managed platform in which applications can run.

5.8 Possible Future Enchancement

The application can be further developed in many ways.

When putting together the project requirements, a number of optional requirements were outlined.

Although, one of the key features of the application is to try not to overhelm the user with statistics, as opposed to many football stats websites, this view would need a little bit more additional information to complete the big picture.

5.9 Conclusions

The main conclusions for this chapter.

Chapter 6

Testing & Evaluation

This chapter evaluates the overall project and provides results of tests carried out.

6.1 Testing

6.1.1 Unit Testing

The implementaion phase of the project was carried out in accordance with Agile Development. One of the cornerstones of Agile philosophy is Test Driven Development or TDD. The essense of TDD is to write the tests before even writing the production code .

According to Miguel Grindberg[8], "There are two very good reasons for writing unit tests. When implementing new functionality, unit tests are used to confirm that the new code is working in the expected way. ... A second, more important reason is that each time the application is modified, all the unit tests built around it can be executed to ensure that there are no regressions in the existing code; in other words, that the new changes did not affect the way the older code works."

For this project it was especially important to provide good tests coverage for the business logic behind the model layer and the external service layer of the application [**?**].

Tests in this project are performed using Python *unittest* library.

6.1.2 Continuous Integration with Travis CI

As the application grows, it may become to take too long to run the unit tests. Therefore, it is worth automating this process by setting up a "Continuous Integration" or CI server. As a CI server

was chosen Travis CI being easy to set up and available for free as a part of the GitHub Student Developer Pack. The service takes care of the unit testing allowing the developer to focus purely on the development process. Travis builds are triggered automatically when developer checks in the project code into the GitHub repository. Intergating Travis CI was just a matter of creation a configuration file, travis.yml.

```
1  language: python
2  python:
3    - "2.7"
4
5  # command to install dependencies
6  install: "pip install -r requirements.txt"
7
8
9  # command to run tests
10 script: python manage.py test
```

Figure 6.1: Travis CI configuration file.

A Travis status icon indicating whether the tests passed or failed was embedded into the README file. This is a convinient feature that helped to keep an eye on the build status from the GitHub repository.

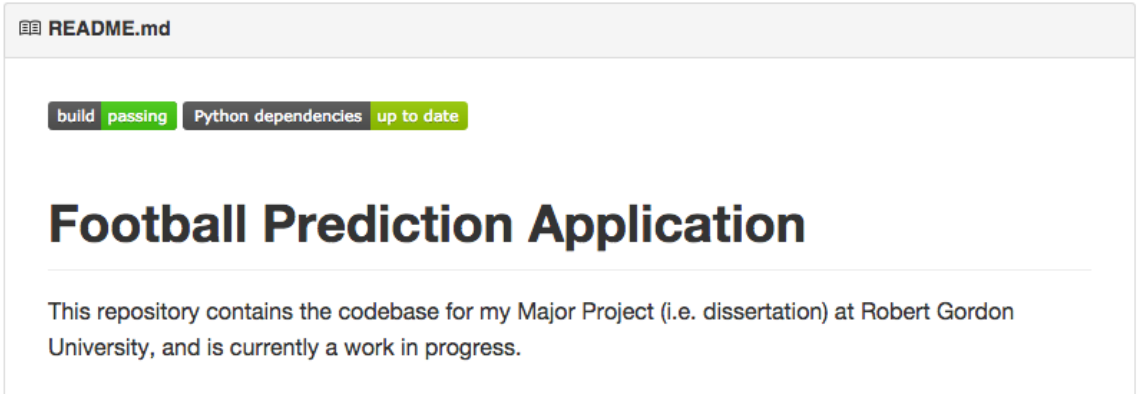

Figure 6.2: An extract from README on GitHub. Travis status icon indicates that the last build passed.

6.1.3 System Testing

System Testing – This will test the system as a whole. This will be run by the developer taking into account the users requirements. Test cases will be created from the requirements with the inputs and expected outputs noted before the testing starts. Some of the test cases may satisfy more than one of the requirements. The tests will be carried out via the black box testing technique.

6.1.4 User Acceptance Testing

6.2 Evaluation

6.2.1

6.2.2 Future Development

6.3 Conclusions

The main conclusions for this chapter.

Chapter 7

Conclusion

This chapter summarises the main outcomes and conclusions resulting from this body of work.

7.1 Assessment of Success

There are meny aspects of this project that should be inspected in order to decide if the project was successful or not.

Using Agile development approach allowed me to concentrate on the result without wasting my time creating final diagrams and overviews before starting with the implementation. It helped me to cut down the preparation phase to the bare essentials, reduced project overhead and made the whole development process more efficient.

Overall, I feel the project has been very successful with a well-designed web application as a result.

7.2 Improvements and Future Work

Further development that could be carried out in the future. In the future I expect to put more leagues, so more English and European leagues are supported More modules (Injuries and suspensions, manager), allowing user to choose from a large variety of inputs and even remove a module from the view

more detailed information in the played match, a view containing feedback

Appendix A

Wireframes

When the app is launched it silently registers with the server allowing the user to use the app immediately. The user is then shown the middle-top screen.

1. From the middle top screen, the user can follow arrow 1 by clicking on the middle button "Pick Mission" to pick a Mission (Run around Arran or Egg for example) and then pick a start and end location. After confirming these choices, the user is taken back to the middle top screen, or at any time can click the "Home" button to return.

2. The user can also view their current acheivements by clicking the "Achievements" button on the middle top screen, following arrow 2. These achievements will be grouped by tabs by category - Distance, Time, Stage and Mission based achievements.

3. The user can follow arrow 3 from the middle top screen to notify the app that they are starting an exercise period, telling the app to track their distance. If a Mission and start and end location are not picked (as in point 1) then they will instead be redirected to this screen and are unable to start exercising until this choice has been made. Once they have successfully advanced to this screen, it will display their current progress as they move showing the user how close to completion of their current stage and overall route they are.

4. When the user has finished exercising, they will click the "End Session" button and be taken to the first summary screen - following arrow 4. Here statistics from their exercise will be shown and the option to share this on several social media outlets.

5. The user can then move to the second and final summary screen, following arrow 5, where they will be shown any achievements they were awarded during that session. The user will also have the option to share these on social media outlets. From here, the user can click the "Home" button and be taken back to the middle top screen.

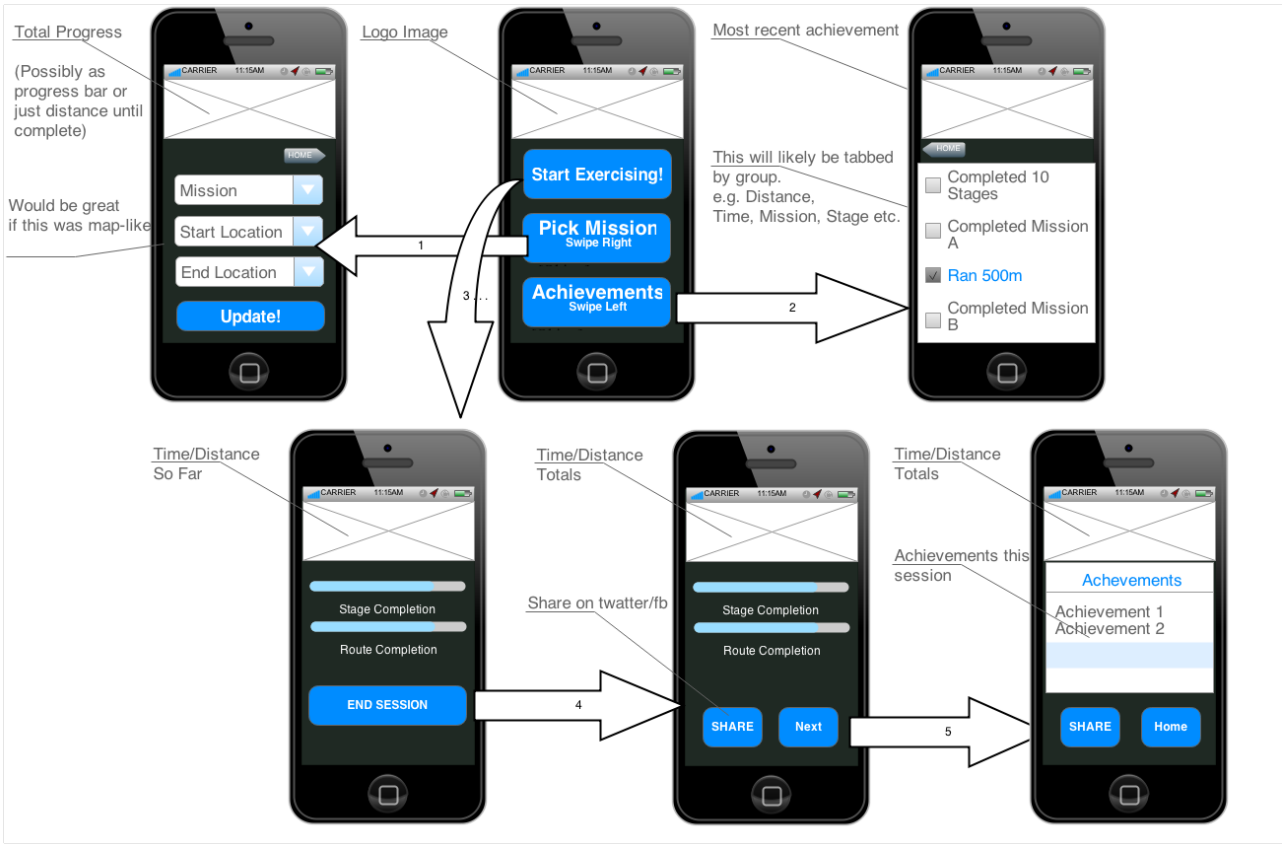

Figure A.1: Wireframes, initial design

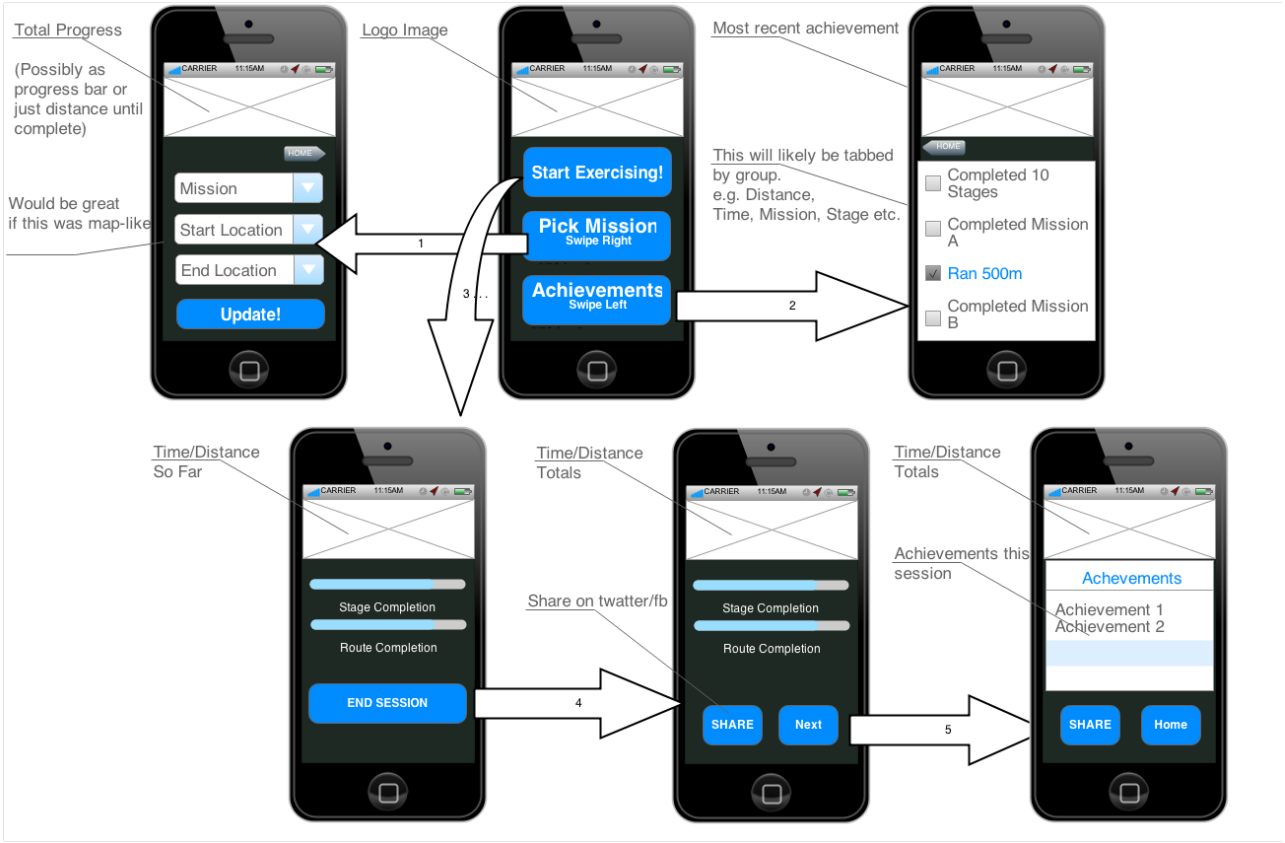

Figure A.2: Inital sketch of wireframe ideas

Appendix B

Installation Instructions

The code can be checked out using git by executing the following command in the terminal:

See the following command :

```
$ git clone git@github.com:marinamarina/sure-thing.git
```

Installation instructions are found at the following url:

https://www.github.com:marinamarina/sure-thing/blob/master/README.md.

If any issues arise regarding installation of any part of the system, do not hesitate to contact me at

1014481@rgu.ac.uk

Appendix C

Project Specification

Summary of the project outline.

C.1 Functional Requirements

some text here

C.2 Non-Functional Requirements

some text here

Appendix D

Project Management

Discussion on how the project was managed. What things impacted the success of the project. How does the continually revised versions of the project plan compare to the initial draft developed at the start of the project. Did everything run according the schedule. Did elements such as exams & coursework have any impact.

Appendix E

Another Appendix

This appendix makes use of the *rotating* package to rotate both figures and tables ninety degrees allowing for large datasets and illustrations to be represented.

Heading 1	**Heading 2**	**Heading 3**	**Heading 4**	**Heading 5**	**Heading 6**	**Heading 7**	**Heading 8**	**Heading 9**	**Heading 10**
aaa	bbbb	cccc	dddd	eeee	ffff	gggg	hhhh	iiii	jjjj
aaa	bbbb	cccc	dddd	eeee	ffff	gggg	hhhh	iiii	jjjj
aaa	bbbb	cccc	dddd	eeee	ffff	gggg	hhhh	iiii	jjjj
aaa	bbbb	cccc	dddd	eeee	ffff	gggg	hhhh	iiii	jjjj
aaa	bbbb	cccc	dddd	eeee	ffff	gggg	hhhh	iiii	jjjj
aaa	bbbb	cccc	dddd	eeee	ffff	gggg	hhhh	iiii	jjjj

Table E.1: A much longer caption that will not be listed in the list of tables page.

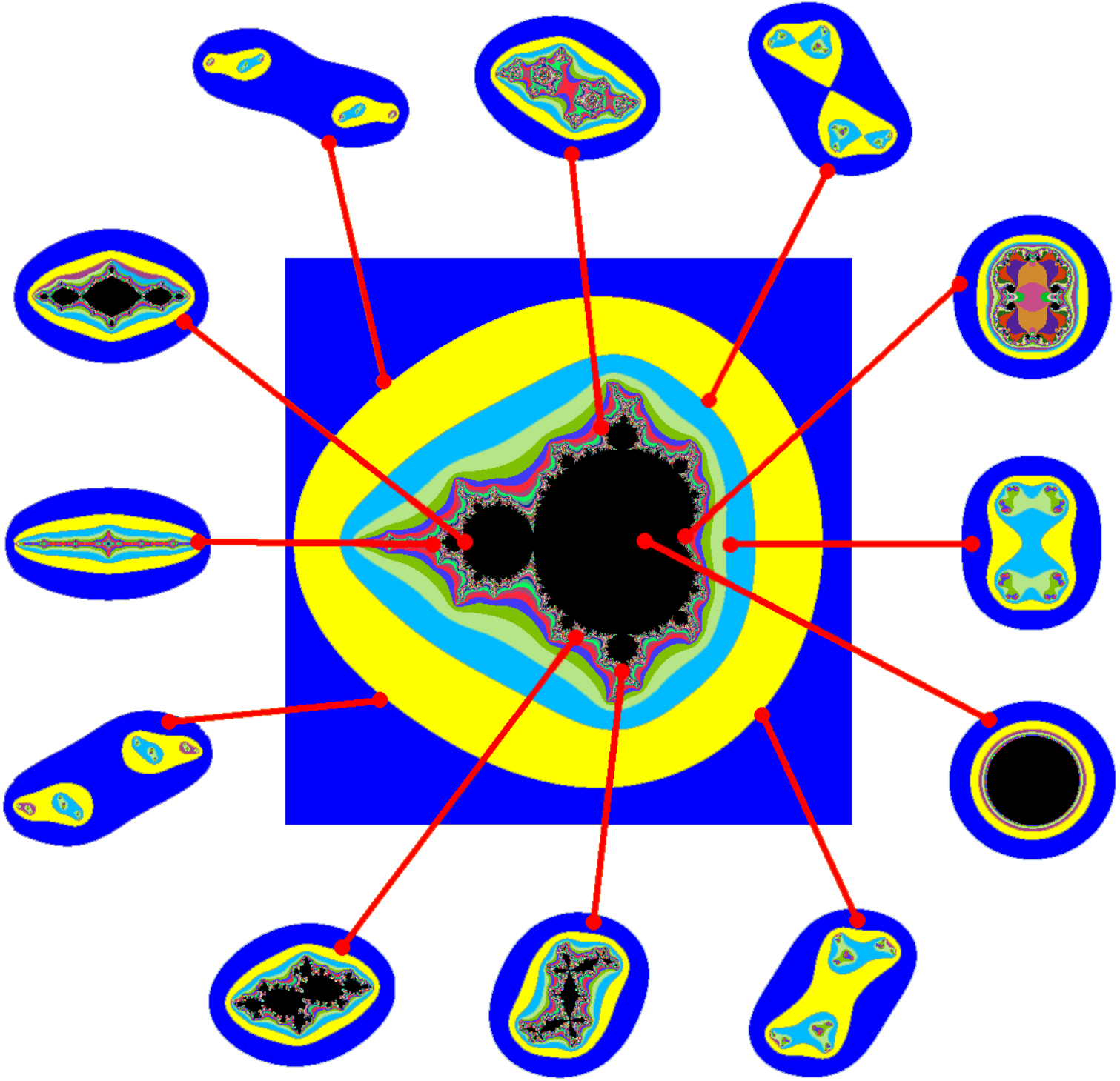

Figure F.1: A much longer caption that will not be listed in the list of figures page

Appendix F

Implementation

Third-Party Code and Libraries

The slides from the formal presentation should be provided here in not more than two pages.

Appendix G

Project Log

The following is a weekly summary of the work carried during the development of this body of work. It covers tasks that were completed, tutorials that were worked through, articles that were read and reviews of discussions / meetings held with the project supervisor and other third parties.

Week Beginning: Monday 27/09/2010

First week working on the project. Had a meeting with supervisor and discussed some of the issues related to the project. The first deliverable is due for the end of next week (project outline & ethics form).

- Downloaded and Installed LaTeX (MikTeX full install), Ghostscript, Ghostview & Winshell.
- Started to get to grips with the LaTeX system by making simple modifications to the template and editing the project log.
- Developed a Mind Map to clarify understanding of project elements.
- Prepared an initial draft of project plan in the form of a Gantt chart.
- Prepared and revised 1 page draft of project summary & filled in ethics form.

Bibliography

[1] TO BE ADDED. Jinja2, full featured template engine for python. [Online]. Available from: http://jinja.pocoo.org/, 2015. [Accessed 14-Oct-2014].

[2] Michael Bayer and various contributors. Sqlalchemy orm. [Online]. Available from: http://docs.sqlalchemy.org/en/rel_0_9/orm/, 2014.

[3] Open Web Community Development Bocoup. The javascript task runner. [Online]. Available from: http://gruntjs.com/, 2015.

[4] John Boxall. A python guide to handling http request failures. [Online]. Available from: http://www.mobify.com/blog/http-requests-are-hard/, 2014.

[5] Odds Checker. Your bets bet. [Online]. Available from: http://www.oddschecker.com/, 2015.

[6] Douglas Crockford. Json (javascript object notation). [Online]. Available from: http://martinfowler.com/articles/continuousIntegration.html#PracticesOfContinuousIntegration, 2015.

[7] Nando Florestan. Large web apps in python: A good architecture. [Online]. Available from: http://nando.oui.com.br/2014/04/01/large_apps_with_sqlalchemy__architecture.html, 2015. [Accessed 5-Mar-2015].

[8] Miguel Grindberg. *Flask Web Development. Developing Web Applications with Python.* O'Reilly Media, 2014.

[9] Chris Eppstein Hampton Catlin, Natalie Weizenbaum and various contributors. Sass preprocessor. [Online]. Available from: http://sass-lang.com/, 2015.

[10] ECMA International. Ecmascript language specification, 2011.

[11] Matt Makai. Api integration. [Online]. Available from: http://www.fullstackpython.com/api-integration.html, 2015.

[12] Harry JW Percival. *Test-Driven Development with Python.* " O'Reilly Media, Inc.", 2014.

[13] Ronald A. Radice and Richard W. Phillips. *Software Engineering. An Industrial Approach.* Prentice-Hall, Inc., 1998.

[14] Kenneth Reitz. Requests. [Online]. Available from: http://docs.python-requests.org/en/latest/api/, 2015.

[15] John Resig. jquery. [Online]. Available from: http://jquery.com/, 2015. [Accessed 4-Jan-2015].

[16] Armin Ronacher and various contributors. Flask micro framework. [Online]. Available from: http://flask.pocoo.org/, 2015.

[17] Armin Ronacher and various contributors. Flask micro framework - application factories. [Online]. Available from: http://flask.pocoo.org/docs/0.10/patterns/appfactories/, 2015.

[18] Maxim Shirshin. Scaling down the bem methodology for small projects. http://www.smashingmagazine.com/2014/07/17/bem-methodology-for-small-projects/, 2014.

[19] tobeadded. Requirejs, javascript file and module loader. [Online]. Available from: http://requirejs.org/, 2015. [Accessed 4-Jan-2015].

[20] Twitter. Twitter bootstrap css3 framework. [Online]. Available from: http://getbootstrap.com, 2015.

[21] W3C & WHATWG. Html (hypertext markup language). http://www.w3.org/html/, 2014.

[22] Wikipedia. Github — Wikipedia, the free encyclopedia. [Online]. Available from: http://en.wikipedia.org/wiki/GitHub, 2015. [Accessed 9-Mar-2015].

[23] Wikipedia. Bet365 — Wikipedia, the free encyclopedia. [Online]. Available from: http://en.wikipedia.org/wiki/Bet365, 2015. [Accessed 22-Feb-2015].

[24] Matt Wright. How i structure my flask applications. [Online]. Available from: http://mattupstate.com/python/2013/06/26/how-i-structure-my-flask-applications.html#s2l, 2013. [Accessed 12-Mar-2015].

[25] Yandex. Bem methodology. http://https://bem.info/, 2005.