SOUTHAMPTON SOLENT UNIVERSITY

MARITIME AND TECHNOLOGY FACULTY

BSc (Hons) Software Engineering

Academic Year 2014-2015

M. TOMLIN

Website Application using a game server API

Tutor: M. Udall May 2015

**This report is submitted in partial fulfilment of the requirements of Southampton Solent University for the degree of BSc(Hons) Software Engineering**

Acknowledgements

Mark Udall – Gave general support and advice throughout the project.

Sheila Baron – Gave helpful advice for the documentation and report approach.

Abstract

This project carried out with the purpose of creating website based application for users of the highly popular and competitive game, League of Legends. The application functions as a player tool to help users pick game characters against their opponent, to give a competitive edge over their adversary.

The application uses the game developers’ API to request the users account related data for calculating better and more accurate character suggestions.

The system was primary programmed in PHP and HTML to produce the website. It also makes use of a CSS-driven framework to produce the user interface for the website.

CONTENTS

**Introduction7**

**Aim and Objectives8**

Aim8

Objectives8

**Background and Context9**

**Specification11**

**Methodology15**

Process Model - XP15

Process Iterations16

Planning16

Web Technologies17

Legal 19

Testing Strategy20

Usability/Acceptance Testing 20

**Design21**

Database Entity Relationship Diagrams21

Wireframe Designs21

**Issues arising from Implementation & Testing22**

Issue 1 - Retrieving Ranked Game Data 22

Issue 2 - Reformatting Summoner Name for JSON extraction24

Issue 3 - Getting Champion Names from their returned IDs27

Issue 4 - Displaying Champion art for the results tables29

Issue 5 - Vote scaling with rating algorithm 31

Issue 6 - UI implementation with CSS 32

Issue 7 - Extracting Statistics for MyStats page34

Issue 8 – Materialize Framework Tables37

**Results39**

**Evaluation and Conclusions40**

Product40

Process42

**Recommendations for Further Work43**

**References44**

**Appendices46**

Appendix A – Riot Games API Webpage46

Appendix B – Final Phase Plan47

Appendix C – Recorded Task Hours49

Appendix D – Prototype51

Appendix E – Cookie Banner52

Appendix F – Usability and Acceptance Survey53

Appendix G – Database ERD60

Appendix H – Design Wireframes61

Appendix I – Voting Algorithm Pseudo code66

Appendix J – System Screenshots67

FIGURES

**Figure 1 – Example of similar application – Source:** (SoloMid Network, 2015)**10**

Figure 2- API document's Summoner request **22**

Figure 3- Summoner request JSON structure **24**

**Figure 4- PHP Document: sstrtolower function**2**4**

**Figure 5- Stack Overflow: str\_replace solution**2**5**

**Figure 6- Reformatted summoner name code26**

**Figure 7- Database ID correction27**

**Figure 8- GitHub: static champion data JSON**2**8**

**Figure 9- Adding image field to table SQL29**

**Figure 10- Artwork images display in tables30**

**Figure 11- iteration 1 MyStats table32**

**Figure 12- CSS class declarations snippet33**

**Figure 13- Array output error34**

**Figure 14- json\_decode output parameter error34**

**Figure 15- stdClass Object extraction35**

**Figure 16- Stat calculations using stdClass object extraction35**

**Figure 17- 1st iteration table spanning37**

**Figure 18- 2nd Iteration result tables38**

Acronyms

|  |  |
| --- | --- |
| Acronym | Definition |
| AJAX | Asynchronous JavaScript and XML |
| API | Application Programming Interface |
| APP | Application |
| CSS | Cascading Style Sheet |
| cURL | Client URL Request Library |
| ERD | Entity Relationship Diagram |
| HTML | Hypertext Mark-up Language |
| IEEE | Institute of Electrical and Electronics Engineers |
| JSON | JavaScript Object Notation |
| PHP | PHP: Hypertext Preprocessor |
| SAX | Simple API for XML |
| SQL | Structured Query Language |
| UI | User Interface |
| URL | Uniform Resource Locator |
| XML | Extensible Mark-up Language |
| XP | Extreme Programming |

Introduction

This project will be to design and create a web application (App) for a computer game (third-party game app) for the game ‘League of Legends’. The game’s developers (Riot Games Inc.) have released the game servers API for third party developers (see appendix A) and an API key was acquired. The website will make use of the game developer’s API to retrieve the user’s in-game data. The user’s data will be used for the application to give the user statistics and recommendations for their current game based on the user’s available account data.

Aim and Objectives

Aim

To create a fully functional and responsive web application for players of the game ‘League of Legends’ to give a competitive edge, making use of the game developer’s available API.

Objectives

* Learn and understand how to use the game API
* Create user and game information database tables
* Create prototype website for testing & basic functionality
* Design and develop the web application
* Design the develop the website’s aesthetics and navigation
* Develop dynamic and responsive website interaction
* Develop website from test user feedback

Background and Context

The reason for choosing this project is related to a computer game the author has been playing for the last two years called ‘League of Legends’. The game has received a huge growth in the number of players within the last year “67 million playing every month, 27 million playing every day” (Riot Games Inc., 2014). Due to this and the very high competitiveness of the game, many third party website applications have been created to help players get an edge over their opponent. The author wishes to create his own third party web application for the game.

To understand the function and purpose of the application, the game firstly has to be described to some detail to give context. The game currently features 121 playable characters (‘champions’) that a player can chose to play for each game. There are currently other web-apps for the same game that allow users to enter the character of their opponent. The web-app will recommend a small list of characters to pick for that match, to counter their opponent and gain an edge before the match starts.

The decision to design a web-app came after the author regularly used a few of these already available apps (see figure 1 below for example). While the existing web-apps are very easy to use and functional, there could be a bigger scope for improvement of these applications concept. The plan for creating the web-application involves requesting the user’s opponent’s character and the user’s in-game account name. With the account name, the app can request previous match data about the player/user from the game server using the game developer’s API. This data can then be used (an example can be characters frequently played by the user) to determine a more personalised character recommendation list for the user. This approach improves the functionality of similar existing applications by making the app function personally for the user.

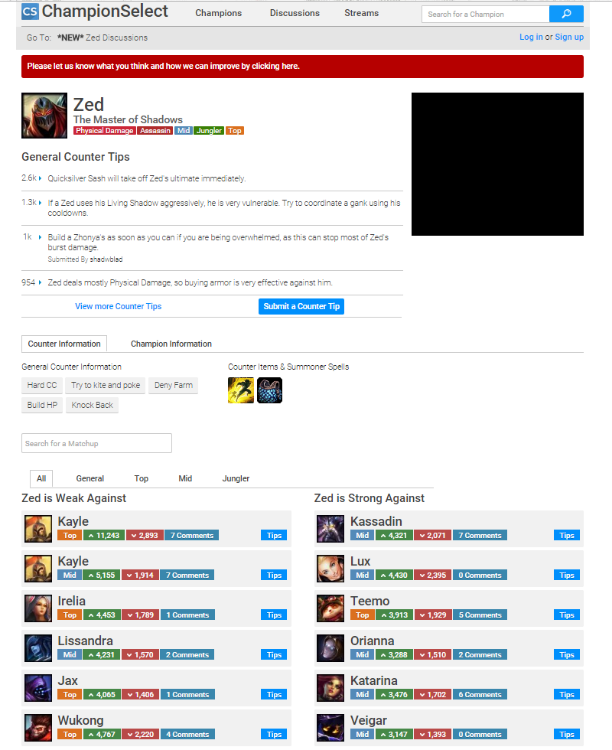


Figure - Example of similar application – Source: (SoloMid Network, 2015)

The decision to create this application as a website is because firstly the user will want to use the application when the user is picking a character for a game. This part is done in the game client application, which is a small desktop application. Because users will already be running a game application on their desktop, the author feels that from user experience that it’s most convenient to run a third party game app in a web browser as opposed to its own desktop application. Secondly it’s already tried and popular approach for majority of all the third party apps for the game, users are already accustomed to using web application for this particular game. Thirdly the author is confident he will be able to write the application as a website using the scripting language JavaScript to be functional “"building full-blown applications in JavaScript is not only feasible, but increasing popular" (MacCaw, 2011). The application will also be expecting heavy use of PHP for contacting the API via cURL requests, which the author has been familiar with using in previous university related assignments.

Specification

1. Form for a personal champion counter listing result

1.1. Data required for form:

1.1.1. Summoner name

1.1.2. Champion

1.1.3. Server

2. Send & Retrieve data to and from Riot Games API

2.1. Data required for request:

2.1.1. Summoner name

2.1.2. Server

2.2. Data retrieved:

2.2.1. Champion win rates

2.2.2. Play frequency of champions

3. A champion results page

3.1. Champion information displayed:

3.1.1. Name

3.1.2. Splash art

3.1.3. Common game role/roles

3.1.4. Counter Items

3.1.5. Counter summoner spells

3.2. List of champions strongest against chosen champion

3.2.1. List in order by rating system

3.2.1.1. Each listing can be voted on

3.2.1.2. votes effect rating of champion listing in database

3.2.2. User able to add non-existing champion to listing

3.3. List of champions weakest against chosen champion

3.3.1. List in order by rating system

3.3.1.1. Each listing can be voted on

3.3.1.2. Votes effects rating of champion listing in database

3.3.2. User able to add non-existing champion to listing

4. Rating system for champion matching in a listing

4.1. Effected by average of user rating of listing

4.2. Effected by user win rate with champion, retrieved from API

4.2.1. Weight of effect adjustable by user via account settings

5. Account creation and log in system

5.1. Details required:

5.1.1. Summoner name (game account name)

5.1.2. Email address

5.1.3. Password (not game account password)

5.1.4 Username

5.2. Forms required:

5.2.1. Log in

5.2.2. Register

5.3. Can logout

6. User statistics page

6.1 Uses summoner name to generate stats from the API

6.1.1. Most played champions

6.1.2. Highest win rate champions

6.1.3. Recent match history stats

6.1.4. KDA ratio ((kills + assist) / deaths)

6.1.5. Average Gold per game

Non-functional requirements

N1. The website needs to render successfully on every web browser

N2. The website needs to render successfully regardless of screen resolution

N3. The results webpage will need to display the correct champion art chosen

N4. Data encryption is required for storing user details

Constraints

C1. User requires a Summoner name (game account name) to use application.

C2. User requires ranked game data linked to their game account for a personalised listing.

Evaluating project process

How did this iteration of the project improve from the previous.

To what extent is the project progress aligning with the plan.

How much use is being of the API.

Evaluating the products

How suitable are the software tools used, for the software requirements.

How developed is the aesthetics and navigation of the website.

To what extent have the software requirements been met.

To what extent does the website fill the user’s needs.

Methodology

Process Model - XP

Taking into account that this will be the author’s first solo software project, he’s concluded that choosing a suitable software development process will bring a huge impact into the productivity and development of the project. Upon researching different software development methodologies it has become clear that choosing a software evolutionary/iterative process will suit the project’s circumstances. The reasoning for this is due to his inexperience with creating and developing a whole software application as a website. The requirements specification and design won’t be clear once the development process starts. Therefore having a flexible and open to change development methodology is most suited for the context of the project. C. Dorman states that software evolution is necessary because the domain of software itself evolves, this is also as the volatility of requirements (Dorman, 2011).

The author has experience working with an Agile development methodology in a team, so choosing a similar iterative process will be beneficial. Industry standards depicts Agile development as only applicable in teams, however many lone software development professionals argue it can be used effectively for solo projects. According to an article (Doll, 2002) agile methodology can easily lend itself to rapid application development, even for solo programmers. For this project the author plans to use Extreme programming (XP) as his development methodology, which is a type of agile development. The decision was based on the XP core focuses, these being:

* Continuous Integration
* Test Driven Development
* Simplicity
* Iterative releases

The process reflects quality and responsiveness for any changing requirements, this will be ideal in context with the project. A real world example of the use of XP can be the company Advoco that states “XP is the practice and pursuit of effective simplicity, as applied to software development” (Advoco Services Inc., 2014). However there is conflict with part of XP’s core practices which is paired programming “software in XP is built by two programmers, sitting side by side, at the same machine.” (Jeffries, 1999-2014). An adapted version of XP will need to be used to keep the methodology’s core focuses, but suitable for a solo software project.

Process Iterations

As stated in the Final Phase Plan (see appendix B) that was created during the risk resolution phase, the amount of software process iterations was to be three. However due to unforeseen implementation issues and time management, only two iterations were done (not including prototype).

The first iteration involved adapting and developing the prototype (see appendix D) created during the risk resolution phase. The focus of this iteration was on implementation of all the functionality for the website correctly. The visual design and aesthetics of the UI were developed however not to professional or planned standard by design.

The second iteration is the last performed iteration of the project. This iteration focused on the development of the website’s UI, layout, aesthetics and interaction. Functionality was developed to include more stats for the actual character the player is matching up against. The statistics page was slightly improved by ordering the table by times played for the characters.

Planning

Planning for the project’s implementation and iteration cycles were done during the risk resolution phase (see appendix B). This artefact is a tabled plan stating the required tasks, their start & due date, estimated hours and priority. This is split up into separate iterations.

The actual dates and hours recorded for this project during each task have been listed in the Final Phase Task Record document (see appendix C).

Web Technologies

HTML:

Because this software will be produced as a website, HTML of course will be used. HTML is language used to visually render webpages to web browsers, so this will definite be used for the entirety of the project implementation.

PHP:

PHP is used to perform the large majority of functionality from the website. Its main use is for sending API requests via cURL and for querying and altering the database. The PHP in this project also was heavily needed for decoding and extracting the JSON returned from the API. The data extracted from returned JSON is reformatted and manipulated within PHP to give the desired values or information. Outputting any responses or values

In the second iteration of the implementation, separate PHP scripts were created and used solely to contact API for requests and processing the returned data. This cuts out the bulk of the PHP code that was being reused in different webpages to lower cohesion.

CSS:

In order apply visual styles and formatting for HTML webpages, CSS is used as a highly recommended method. A basic external CCS file is used as the method for all webpages of the first iteration of the project.

The second iteration made use of a CSS Framework to allow the project website to be cross-browser compatible, visually improved and more responsive interaction. Also to significantly improve the UI from the first iteration’s, as the UI development by the author through regular CSS wasn’t to standard previously designed. This was due to lack of experience with implementing CSS to a professional standard by the author.

Database:

The database for this project is hosted on the Edward2 server, owned by the university. The reasoning for hosting the project database there is because firstly because its use is free as a resource. Secondly is the familiarity with using MySQL databases with phpMyAdmin as a client, which is all pre-installed.

For populating the database tables with champion data, because there are 124 champions within the game, only a sample of 20 were stored for appropriate testing needs.

Web Technologies – Adopted during 2nd iteration

Materialize Framework:

As stated on their webpage Materialize is “a modern responsive front-end framework based on Material Design” (Materialize, 2015). In order to implement the desired UI for this second iteration, a CSS heavy framework was required to reduce implementation time and increase the visual aesthetic and interactivity.

The Framework makes use of a column and row design for declaring any <div> tag containers, implemented through declaration of classes. This results for easy and visually appealing content on the webpages. Use of the framework was taught from reading Materialize’s webpage implementation guides.

JavaScript:

The use of JavaScript was unexpectedly only needed during the second software iteration. The JavaScript that was used can be split into two functions. The first is for use with specific Materialize framework components such as select forms and tabs. The second is for sending AJAX requests and receiving the returned data, which can only be done through JavaScript.

AJAX:

This project didn’t necessarily need the use of AJAX for functionality. However to reduce PHP processing, and reuse of code, web services were used. The only way (at least known to the author) to contact a web service from a webpage is through AJAX, because it’s asynchronous. Meaning it’s the request is sent and returned immediately without reloading the webpage.

Legal (Use of Artwork and Cookies)

Art Resource Copyright:

As detailed in the risk resolution report, Upon reading the Riot Games guidelines for community use of their intellectual property it was clear that using game resources such as art work was fine for free applications and websites, however using the league of legends or riot games logo was not. “Game art’s a yes… But logos are a no” (Riot Games Inc., 2015).

Cookies:

Cookies are major part of many websites and have many legal restriction in place. The use of cookies that is unknown to the user can be in breach of EU privacy laws and the UK’s data protection act. As stated by the ICO (Information Commissioner's Office, 2015) “websites in the UK need to obtain consent from visitors in order to store on and retrieve usage information”.

For the purpose of implementing the user account registration and log in system for the project website, PHP session are used. This stores information about the current user’s session with a web application as a variable. Its use is similar to how a Cookie functions so research was needed to determine is legality. According to W3Schools (W3Shools, 2015) it states that “A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the users computer”. Therefore it doesn’t pose any know legal issues.

For the sake of practise and safety with these laws, a JavaScript plugin called Cookie Consent (Silktide Ltd, 2015) has been used from a third party to implement to the project website. This plugin is open source and displays a banner at the top of the website for visitors about the use of cookies, with the option to deactivate the session cookie from the website. When running the plugin it didn’t detect any cookies from the project website so no banner is displayed. The privacy setting can be shown with the plugin stating no cookies are present as well. See appendix E for a view of the plugin implemented to the project website.

Testing Strategy

For testing the project websites effectively and functionally for review and changes, the testing strategy can be broken up into four sets of tests. These are:

1. Functionality testing
2. Usability testing
3. Interface testing
4. Compatibility testing

Usability/Acceptance Testing

In order to effectively test the usability of website, the target market which are League of Legends players should be testers for the context of this project. Because of this user can also perform acceptance testing as there are expectations of how the website should perform. A survey was created to test for both, and was completed by five different game users (see appendix F).

Design

Database Entity Relationship Diagrams

Database ERD’s were created at the start of each process iteration to help design and implement the database to store all the tables and values required for meeting website functionality.

The database ERD created from the first iteration (see Appendix G.1.) shows the initial database structure with data fields that were presumed required considering the use and requirements form the prototype database.

The second iteration’s ERD (see Appendix G.2.) updates from the previous iteration by including extra data fields to the champion and users tables to increase information returned to the results page and better establish user login functionality. Another reason is to be more complicit with the champion API data such as matching ids, this reduces the need for API requests in order retrieve static champion data.

Wireframe Designs

Visual designs for the interface of the webpages were made as design Wireframes using Microsoft Visio. The wireframe produced during the first iteration (see Appendix H) shows the basic design and layout planned for implementation. The interface shown in the wireframe was designed to be achievable through CSS.

Upon the second design iteration it was clear that the UI designed from the first wasn’t implemented to an acceptable standard and doesn’t all follow the same layout. This is due to lack of experience with using base level CSS for professional UI’s.

The decision to reuse the same UI design, but have it implemented through a framework instead was made. The reasoning is because firstly learning to implement the regular CSS to the design would be too time consuming. Secondly if the UI is jeopardised by simplifying the design to be easier to implement, it goes against two of the original project objectives which is to achieve good website interaction and aesthetic.

Issues arising from Implementation & Testing

This section of the report will identify some of main issues that arose through both implementation phases of the project. Explanations of how the solutions were solved and their impact have been added.

Note: a Champion refers to a game character and a Summoner refers to the player’s account from the game

Issue 1 – Retrieving Ranked Game Data

Although retrieving account information from the API was practised with the prototype during the risk resolution phase. It was overlooked from the API document that in order to the retrieve account data from the player’s ranked games (necessary for affecting the rating), the summoner ID (account ID) would be needed as a parameter. Because the Riot API consists of many different request URLs for widely differing data returns, different parameters are necessary each.

The summoner ID, like most account IDs isn’t known to the account holder, therefore asking user to input their summoner ID is out of the question.

Solution:

All the account realated API requests were checked from Riot’s API document for one that returned the summoner ID with parameters aqquired from the search form (summoner name and server). A suitable request was found that returned relevent accounnt information, including the summoner ID.

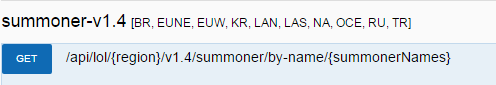


Figure - API document's Summoner request

Extracting the ID from the JSON returned however had unforeseen problems that extended this issue, see issue 2 for details of this.

Impact on Project:

This was one of first unforeseen problems related to the API. The only solution for it was found (without an external web service) within a reasonable time. However this means an a additional cURL request to the API will needed each time ranked game statistic are needed, which is necessary for majority of the functionality.

Additional time: 3hrs (additional cURL requests implemented on separate webpages)

Issue 2– Reformatting Summoner Name for JSON extraction

Leading on from issue 1, when implementing and testing the cURL request to receive a summoner ID, the format of the JSON returned for this particular API request caused unforeseen problems. This lead to complications with extracting the actual summoner ID.

The issue is to do with the unexpected JSON structure returned by the request. What is meant by this is when testing the request in Riot’s API document after to check the for the JSON structure it returns. It was revealed that the summoner ID and other returned data for the account wasn’t returned directly, but contained within an object that shares the same name as the account but in lowercase (see fig. 3).

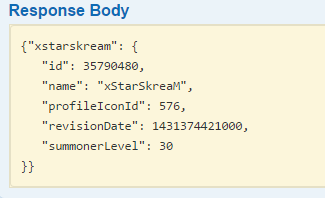


Figure - Summoner request JSON structure

The reason for this is because of requests functionality. It was realised that from the URL of the request (see fig. 2) can take as a parameter, multiple summoner names. This explains the JSON structure, as it would need to return an array of account objects for each one entered. Upon further reading of the document it notes that accounts are all referenced and returned on their database as the same as the summoner name, but all lowercase characters without any spaces.

Solution:

From analysing the API document it’s clear that in order to access the JSON object with a given summoner name, reformatting that name to be lowercase and have no spaces will be required.

I order to find a solution for reformatting the string value, some research had to be done for how to accomplish this within PHP. For finding out how to change a string into lowercase a built-in function was discovered with PHP’s documentation manual (The PHP Group, 2015).

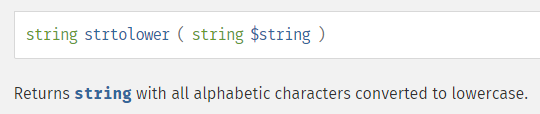


Figure - PHP Document: sstrtolower function

For finding a way to remove any spaces in a string, stack overflow listed the solution as an answer from a similar problem posed (Byers, 2010). The solution is another built-in function from PHP that is also listed in the PHP Manuel (The PHP Group, 2015) called ‘str\_replace’. From the manual it states that it can replace given characters from a string with a given replacement. This means it can check the string for any spaces and replace them, in this case with no space (using ‘’).

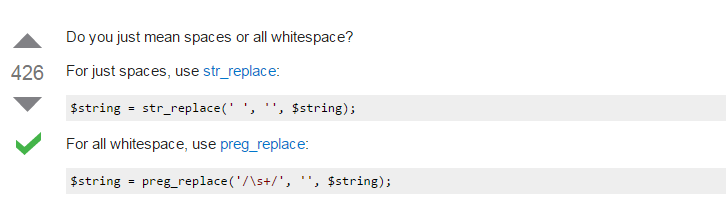


Figure - Stack Overflow: str\_replace solution

Implementing the solution wasn’t a problem as it can be done in single line of PHP by combineing the two built-in functions. From there extracting the summoner ID from the JSON was relatively simple (see fig. 6).

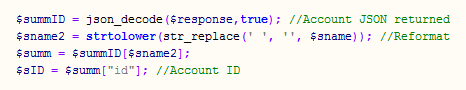


Figure - Reformatted summoner name code

Impact on Project:

This took some extra unexpected implementation time that was extended from issue 1. This issue required time to analyse the problem from errors with extract the JSON and required reading from outside sources for implementing the solution.

Additional time: 3hrs

Issue 3 – Getting Champion Names from their returned IDs

When extracting all the ranked statistics for an account, from the ranked stats API request. The JSON returned gave an array character objects with contained statistics which was expected. However what wasn’t expected is that each champion object didn’t contain the name of the champion, only the ID. The champion name is needed to property output the table of results to the user, as well as to interact with the suggested counter champion choice (weak against table) to effect the rating which is contained in the project’s database.

Solution:

There were two possible solutions found for this problem. The first, which wasn’t used is to send a third cURL request to the API for extracting the champion name with the ID as a parameter. This could then be linked with the database for the counter picks table, by referencing between the two with the character name. However this approach requires more software processing in PHP because of the need for an additional API request, as well as extracting the returned JSON, just to get the champion name.

The second and used solution was to alter the entries in project database’s Champions table to match have matching IDs with the same champions stored on API server. This means the right champion names can be displayed because they accessed through database, using the match IDs as a reference.

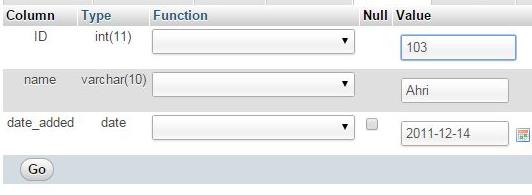


Figure - Database ID correction

Because I only had the champion IDs from the API requests, finding out which correct ID number matched to which champion was necessary to correct the existing database champion IDs. Getting all the matching IDs to names could be checked through the API request described in the other solution. However a shared GitHub file (GitHub-Nagash, 2012) from a user was found that contained the copied JSON of the static champion data from the API, for majority of the champions. A lot of champions were missing from this file so getting the IDs for the remaining existing database sample data was done through the API.

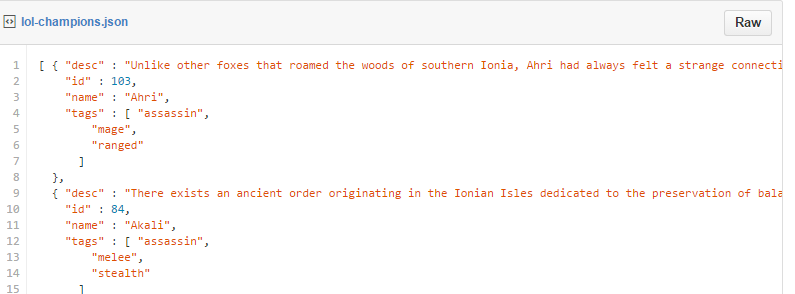


Figure - GitHub: static champion data JSON

Impact on Project:

This could have been avoid if planning with matching champion data in the database with the API was considered during the database design phase. Finding the right IDs for the champions stored in the database and changing their values was very time consuming task but was necessary for allowing the main functionality with the results table.

Additional time: 4hrs

Issue 4 – Displaying Champion art for the results tables

As designed in the UI wireframes, the results tables will need to include champion art to illustrate each result in the table. Displaying the correct artwork to a champion as a result would require storing the images in the database, however to implement this was unknown at first.

Solution:

A Stack Overflow user (M. Harrison, 2010) suggested the easiest way is to store the file path to the image as a ‘varchar’ (character variable) data field. To implement this the artwork files will need to be copied from game to a folder stored server side with the website (allowed use of game art, see legal section for more). The Champions database table will need to be altered to include an extra data field to store as text the file path to the image server-side.

G:\Y3\Final Year Project\screenshots\database_2.jpg

Figure - Adding image field to table SQL

Once all the champion entries in database were labelled to art file path. Adding and displaying the art in the result table of the website worked perfectly (see fig. 10).

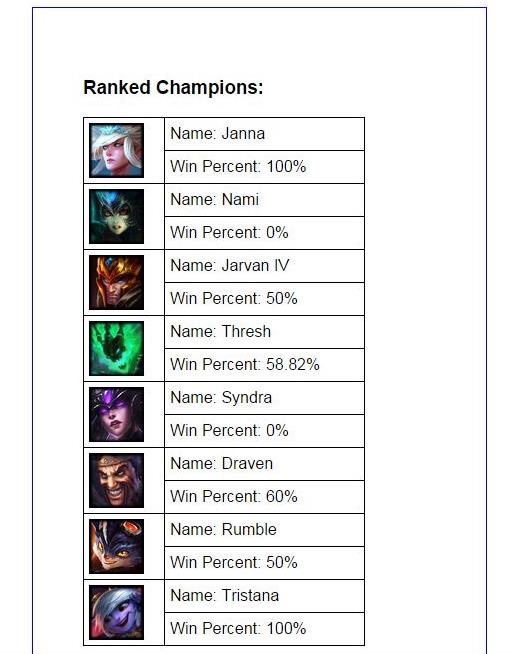


Figure - Artwork images display in tables

Impact on Project:

This was an expected task to perform, however finding the solution that did the same function as storing images to the database and referencing each file path to a champion database entry was very time consuming to perform.

Additional time: 3hrs

Issue 5– Vote scaling with rating algorithm

During the design phase some pseudo code was produced (see appendix I) that outlined how voting on a champion match up would affect the rating value and scale with amount of votes already. However during implementation there were errors in how the rating was effected when a result rating was already altered by the player data.

Solution:

In order to save a lot of implementation time for scaling the votes to change the rating. Instead to avoid errors in rating calculation, it was decided to change the amount a vote effected the rating by having it change the value in the database statically. This means having the rating value changed by a static amount of 100 for every vote.

Impact on Project:

This goes against 4.1 in the specification, however can be justified as only a necessary feature once large volumes of users are voting on specific champion matchups. Its purposed was to make the ratings not go too out scale in relation to other characters. Further development of the project can appropriately solve is function.

Issue 6 – UI implementation with CSS

As previously stated in the methodology and design, there were many issues relating to implementing the CSS correctly for the UI. This includes creating the right sizing, layout and positioning of all <div> tag containers for the basic UI. As well as producing the correct sizing for cells (artwork fits appropriately) in HTML tables and having some cells span vertically (see fig. 11 below).

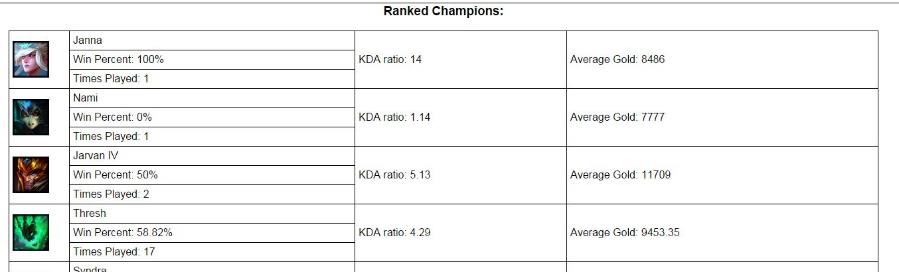


Figure - iteration 1 MyStats table

Solution:

W3schools (W3Schools, 2015) was heavily used as a learning resource for fixing many small but numerous bugs for implementing CSS. Their pop-up code editor was very helpful for plotting correct table structures and setting the right sizes for certain columns with the use of CSS classes.

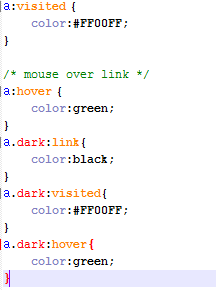
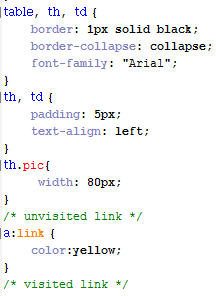


Figure - CSS class declarations snippet

Figure 12 above shows some of the new CCS practices with classes that was adopted during implementation of UI for the first iteration.

Impact on Project:

The issues with implementing a professional UI through base CSS was the implementation time and need for good prior experience. Its impact resulted in the website UI being to an unsatisfactory standard to the proposed design. This meant the focus of the second iteration relied heavy on re-implementing the UI rather than developing the functionality.

Additional time: 7hrs (time spent implementing CSS and solving bugs)

Issue 7 – Extracting Statistics for MyStats page

This issue spanned the results page as well, however was more prevalent on the player’s account statistics (MyStats) page. The issue was with the extracting and outputting of the decoded JSON for the ranked game statistics. When attempting to implement and display the win percent for champions, during testing it was outputting ‘Array’ instead. See figure 13 below for a screenshot, this is what identified the issue.

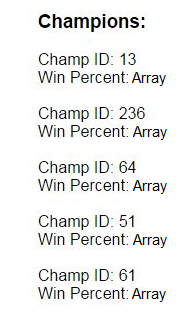


Figure - Array output error

Solution:

After searching through posts on Stack Overflow for why it was literally outputting ‘Array’ it was suggested that it was trying to outputting a PHP associative array into HTML and that the result.

Upon researching for how I was getting an associative array and how to print its data to HTML, I found out cause for returning this unexpected format. From reading off the PHP learning resource DevelopPHP (DevelopPHP, 2015) it detailed that function that was being used to decode the JSON for PHP was set to return an associative by setting the second parameter to true.



Figure - json\_decode output parameter error

The reason for why it was set to true was because of the function being used to return extract the summoner ID was done with method. The extraction method worked then because that data structure only had a depth of two levels (see fig. 3). This same extraction method was used on ranked data returned that gave the original error, showing it could only access second level or array (the champion) but not the statistic it contains so outputted ‘Array’ instead.

For solving this problem, reading from the DevelopPHP page about the json\_decode function. When the function is not set to true for the second parameter it instead outputs a PHP stdClass object (standard class object) from the JSON. A method for referencing properties in PHP objects was already known to the author. So the json\_decode function was changed to output a PHP object instead, the ranked stats were then extracted with a better known method properly. See figures 15 and 16 for the resulting code.



Figure - stdClass Object extraction

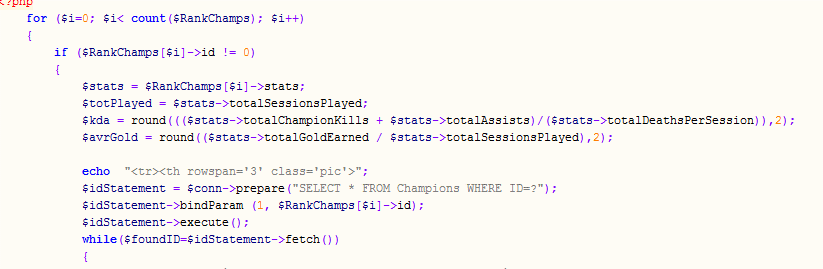


Figure - Stat calculations using stdClass object extraction

Impact on Project:

Extended the implementation time for manipulating the ranked games stats for use on the results and MyStats pages.

Additional time: 2hrs

Issue 8 – Materialize Framework Tables

There were many HTML and CSS features in the Materialize Framework that were implemented to UI well. However the framework’s tables were restricting/lacking two features expected, that posed some design problem for the UI.

The first is the option to span a table cell over more than one cell. This was used in all the tables from the first iteration to improve layout of the tabled results to be more readable (see fig. 17).



Figure - 1st iteration table spanning

The second is the ability to highlight a chosen row from the table. This was needed as a design trait to visually show any characters in the ‘weaker than’ (counter) table that had their rating effected, by the user’s ranked stats. This feature wasn’t used in the previous iteration however was planned to adequately solved by this chosen framework.

Solution:

The first feature couldn’t be solved without rewriting the framework CSS which had no formatting inside the file, so making any corrections could unknowingly break the framework. In order to reduce data cluttering from the results tables, the voting buttons were rearranged to appear on top of one another in their single cell. This was done to reduce width needed for displaying the remaining table’s values clearer. Also the buttons were given stylised arrow symbols to reduce increase cognitive usability. The arrow symbols are available as HTML code so can be immediately implemented without resources. These arrows were found from reading a character code webpage (Hoffman, 2015) and are available in Unicode hexadecimal also.

The second design feature was solved by adding a column labelled ‘played’ to indicate whether the rating was adjusted. This wasn’t an ideal solution as it’s visually less appealing and more intrusive. However it was the only solution achievable through the table framework that informs the user correctly (see fig. 18 below).

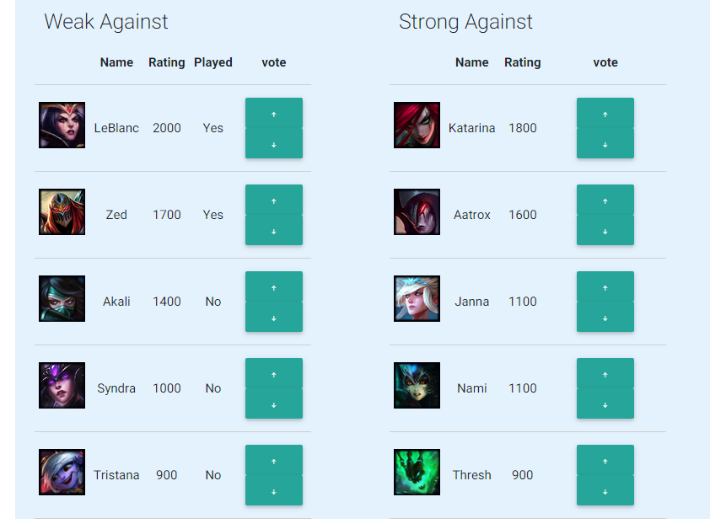


Figure - 2nd Iteration result tables

Impact on Project:

This issue negatively impacted the readability and format of the tabled results to the original design plan. However the framework’s tables are visually more appealing than regular html tables. Also the voting buttons were better laid out in this framework as they were automatic formatted by the framework appropriately within their single cell.

Additional time: 3hrs

Results

The website produced for this project after the second process iteration resulted in a fully functional website application. By combining the functionality of the application developed in the first iteration, with the UI developed during the second, the project website application has been realised.

For a look of the finished (although up for further development) website see appendix J for a view of all the web pages.

Evaluation and Conclusions

Product

The final website product was shared and tested by five different users who play the game. A survey was created using Survey Monkey and fill in by the test users (see appendix F). The first page of the survey questioned the usability of the website, the second page queried acceptance. The survey results were gather and made in several graphs for better analysis.

From the user feedback it was clear that the usability of the website was widely agreed as very good. Meaning the finished website is very intuitive to use.

The fifth question was very useful as it asked how the website can be improved. From the responses nearly all the test user were suggesting to include more information on both the results page and account stats page. One user mention “website feels empty”. These comments suggest that the webpages are lacking a lot of information normally present similar websites.

To what extent does the website fill the user’s needs:

For question 2 of the survey, it was informative to know that four out of five test users had used a similar website app for the game before. This shows there are prior expectations from majority of the users when reviewing the website.

The results from question 3 showed an average rating of 3.4 out of 5 for usefulness to the player. This means the necessity of the website to a player is adequate, however this can definitely be improved with development of extra functionality.

How suitable are the software tools used, for the software requirements:

Question 4 asked about the usefulness of the counter lists being produce based on their ranked match data. Results showed an average of 3.8 out of 5, meaning API functionality with the counter list was reasonably received, however the listings can definitely be improved upon, such as cell highlighting for clarity of effected characters.

To what extent have the software requirements been met:

With reference to the specification majority of functionality was implanted with minimal bugs. However there were a few pieces of functionality that were never implemented, these are:

3.1.4 Counter Items & 3.1.5 Counter summoner spells

Would have helped to have implemented these, as it would increase the information on the results for the matched character. However this information isn’t stored on the API so the data would have to be self-collected or through a separate web service.

3.3.2 User able to add non-existing champion to listing

This could have been implemented relatively easily as the author is relatively experienced in adding database entries through a website, this function however was overlooked when trying to meet project deadlines.

4.1 Effected by average of user rating of listing

The pseudo code (see appendix I) was written already however due to the complexity, the implementation wasn’t realised (see issue 5 for more).

4.2.1. Weight of effect adjustable by user via account settings

This was another overlooked function that’s reasonable to implement by applying an additional weight to how it’s calculated with the win rate.

Process

Planning:

The planning stage could have been done a lot better inn terms of scheduling and work documents. Although there was room with the iteration to redevelop the designs and work plan beforehand. This was overlooking in favour of focusing on correct implementation.

The risk resolution phase was the main highlight for the planning of the project. Many potential risks were either mitigated or better analysed through making a prototype website (see appendix D).

Project Management:

The project management for this final phase was similar to the planning. It was lacklustre during the final phase, as shown by the lack of project documents produced.

Also a lot of time wasn’t well spent during the implantation and between iteration cycles. This was due to falling behind on the due date for task in the final phase project schedule (see appendix C) regularly.

Methodology:

The methodology for the most part was well thought out with justified uses, for different web technologies to implement the website. There was also close attention to the legality of certain website laws in action that led to the use of a cookie banner (see appendix E).

The process was carefully chosen with justification however, when implementing the iteration style of agile and XP, there was lack of focus for re-engineering the design and functionality, instead the iteration was used to mainly implement the short comings of the previous iteration cycle.

Recommendations for Further Work

More information relating to the matched champion

This was specifically mention by two test that there wasn’t enough information relating to the opponents character. As stated in the specification but not implement. Adding data about counter items and spells would improve the product.

Allow users to add listings for the results

As stated in evaluation this should have been implemented given ease of it. This feature would improve the user interaction with the system greatly and reduce developer work for adding the data entries to the Vs tables that produce the listings.

Enable comments to be written for listing specific matches

This was allow suggested by a test user and is rather reasonable to implement. Like the previous recommendation, user interaction would increase and addition data about specific character matchups could be stored.

Improved clarity of played champion listings

As mentioned in issue 8, the listing for champions played by the user should have better indication than table column stating it. A solution most likely can be found with third party web plugin.

More statistics should be added to the account stats page

The account page should defiantly have more account statistics displayed as there are many other API request for getting account data. Such as recent games’ data and current game data from a live match.

References

Advoco Services Inc.. (2014). Solutions: Methodologies[online][viewed 13 May 2015]. Available from: http://www.advoco-services.com/methodologies/

Byers, M. (2010, Jan). Stack Overflow: How to strip all spaces out of a string in php?. [online][viewed 14 May 2015]. Available from: http://stackoverflow.com/questions/2109325/how-to-strip-all-spaces-out-of-a-string-in-php

DevelopPHP. (2015). Decode JSON Formatted Data[online][viewed 14 May 2015]. Available from: https://www.developphp.com/page.php?id=860Cachedjson\_decode

Doll, S. (2002). Agile programming works for the solo developer[online][viewed 14 May 2015]. Available from: http://www.techrepublic.com/article/agile-programming-works-for-the-solo-developer/

Dorman, C. (2011, January). An experience report of the solo iterative process. Detroit: Wayne State University Theses.

GitHub-Nagash. (2012, JAN). League of legends champion data in JSON format[online][viewed 14 May 2015]. Available from: https://gist.github.com/nagash/1688617

Hoffman. (2015). Arrows HTML Code and Unicode Hexadecimal[online][viewed 14 May 2015]. Available from: http://character-code.com/arrows-html-codes.php

Information Commissioner's Office. (2015). Guide to data protection[online][viewed 14 May 2015]. Available from: https://ico.org.uk/for-organisations/guide-to-data-protection/

Jeffries, R. E. (1999-2014). What is extreme programming[online][viewed 14 May 2015]. Available from: XProgramming: http://xprogramming.com/what-is-extreme-programming/#pair

M. Harrison. (2010, Nov). Storing Images in DB - Yea or Nay? [online][viewed 14 May 2015]. Available from: http://stackoverflow.com/questions/3748/storing-images-in-db-yea-or-nay

MacCaw, A. (2011). JavaScript Web Applications. In A. MacCaw, JavaScript Web Applications (p. 1). San Francisco: O'Reilly Media.

Materialize. (2015). Materialize[online][viewed 14 May 2015]. Available from: http://materializecss.com/

Riot Games Inc. (2014). Riot Games: Developers[online][viewed 14 May 2015]. Available from: https://developer.riotgames.com/

Riot Games Inc. (2014, 11 11). Riot Games: Our Games[online][viewed 14 May 2015]. Available from: http://www.riotgames.com/our-games

Riot Games Inc. (2015, JAN 15). Legal jibber jabber[online][viewed 14 May 2015]. Available from: http://www.riotgames.com/legal-jibber-jabber

Silktide Ltd. (2015). Cookie Consent[online][viewed 14 May 2015]. Available from: http://sitebeam.net/cookieconsent/

SoloMid Network. (2015). Champ Select [online][viewed 14 May 2015]. Available from: ChampionSelect: http://www.championselect.net/champions/zed

The PHP Group. (2015). PHP String Functions[online][viewed 14 May 2015]. Available from: http://php.net/manual/en/ref.strings.php

W3Schools. (2015). CSS Tutorial[online][viewed 14 May 2015]. Available from: http://www.w3schools.com/css/default.asp

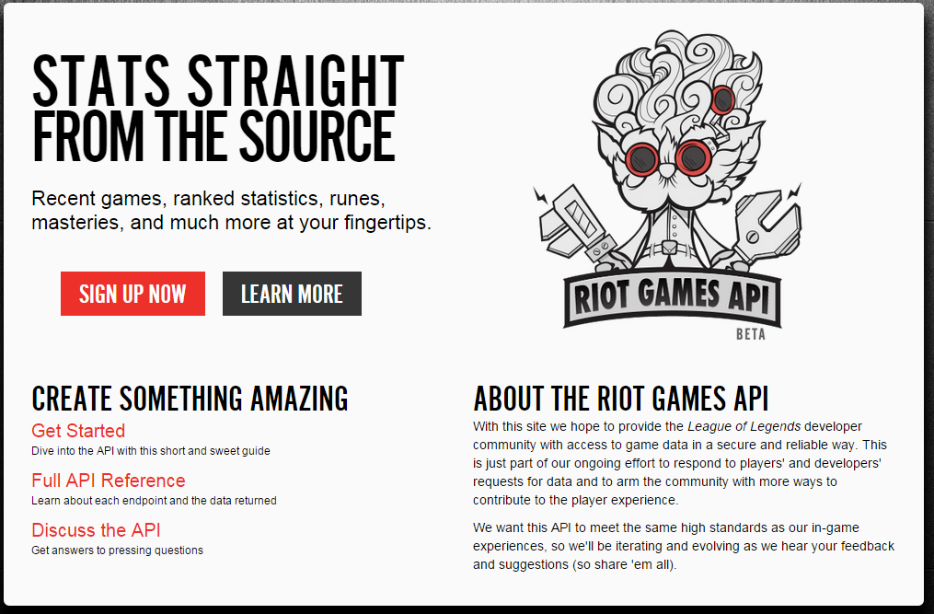
W3Shools. (2015). PHP 5 Sessions[online][viewed 14 May 2015]. Available from:http://www.w3schools.com/php/php\_sessions.asp

.

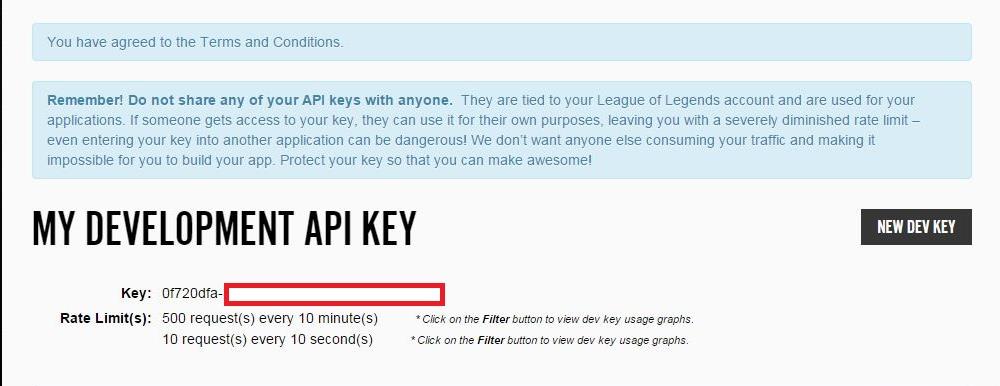
Appendices

Appendix A – Riot Games API Webpage

A.1. – API sign up page



A.2. – Acquired API key



Appendix B – Final Phase Plan

**First Iteration**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **Description** | **Start date** | **Date due** | **Priority** | **Estimated Hrs.** | **Comments** |
| 1. | Draw design wireframes | 19/01 | 21/01 | Medium | 4 | Using Microsoft Visio |
| 2. | Create external CSS file for website | 20/01 | 24/01 | Medium | 3 |  |
| 3. | Design Database Structure | 20/01 | 24/01 | High | 3 | Make an ERD |
| 4. | Deploy Database onto server | 21/01 | 24/01 | High | 4 | Onto Edward server |
| 5. | Populate Database with test data | 21/01 | 24/01 | Medium | 2 |  |
| 6. | Create homepage with Form | 24/01 | 26/01 | High | 4 | Won’t require PHP |
| 7. | Create layout for results page | 24/01 | 26/01 | Medium | 6 | Without functionality |
| 8. | Re-design & implement rating system | 26/01 | 27/01 | Medium | 2 | Javascript |
| 9. | Generate champion listings | 27/01 | 30/01 | High | 9 | Most intensive programming |
| 10. | Add voting for listings | 30/01 | 01/02 | Low | 4 |  |
| 11. | Create log in and register account feature | 01/02 | 02/02 | Medium | 3 |  |
| 12. | Create account page | 02/02 | 04/02 | Medium | 7 | Layout and functionality |

**Second Iteration**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **Description** | **Start date** | **Date due** | **Priority** | **Estimated Hrs.** | **Comments** |
| 1. | Re-draw design wireframes | 04/02 | 06/02 | High | 5 | Using Microsoft Visio |
| 2. | Re-create external CSS file for website | 05/02 | 09/02 | High | 3 |  |
| 3. | Re-Design Database Structure | 05/02 | 07/02 | Medium | 3 | edit ERD |
| 4. | Deploy and populate new database | 07/02 | 09/02 | Medium | 2 |  |
| 5. | Re-create homepage with Form | 09/02 | 11/02 | High | 4 | Won’t require PHP |
| 6. | Re-create layout for results page | 09/02 | 11/02 | Medium | 6 | Without functionality |
| 7. | Re-design & implement rating system | 11/02 | 12/02 | Medium | 3 | Javascript |
| 8. | Generate champion listings | 12/02 | 15/01 | High | 11 | Most intensive programming |
| 9. | Add voting for listings | 15/02 | 17/02 | Low | 4 |  |
| 10. | Create log in and register account feature | 17/02 | 18/02 | Medium | 3 |  |
| 11. | Re-create account page | 18/02 | 20/02 | Medium | 7 | Re do Layout and functionality |

**Final Iteration**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **Description** | **Start date** | **Date due** | **Priority** | **Estimated Hrs.** | **Comments** |
| 1. | Re-draw design wireframes | 22/02 | 24/02 | High | 7 | Using Microsoft Visio |
| 2. | Re-create external CSS file for website | 24/02 | 02/03 | High | 5 |  |
| 3. | Re-Design Database Structure | 24/02 | 26/02 | Medium | 4 | edit ERD |
| 5. | Deploy and populate new database | 26/02 | 02/03 | Medium | 3 |  |
| 6. | Re-create homepage with Form | 02/03 | 06/03 | High | 6 | Won’t require PHP |
| 7. | Re-create layout for results page | 02/03 | 06/03 | High | 8 | Without functionality |
| 8. | Re-design & implement rating system | 06/03 | 08/03 | Medium | 5 | Javascript |
| 9. | Generate champion listings | 08/03 | 14/03 | High | 14 | Most intensive programming |
| 10. | Add voting for listings | 15/03 | 17/03 | High | 5 |  |
| 11. | Create log in and register account feature | 17/03 | 20/03 | High | 4 |  |
| 12. | Re-create account page | 20/03 | 24/03 | High | 9 | Re do Layout and functionality |

**Additional**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **Description** | **Start date** | **Date due** | **Priority** | **Estimated Hrs.** | **Comments** |
| 1. | Project report | 01/05 | 15/05 | High | 16 |  |
| 2. | Design poster | 17/05 | 25/05 | High | 8 |  |
| 3. | Prepare presentation | 17/05 | 25/05 | High | 6 |  |

Appendix C – Recorded Task Hours

**First Iteration**

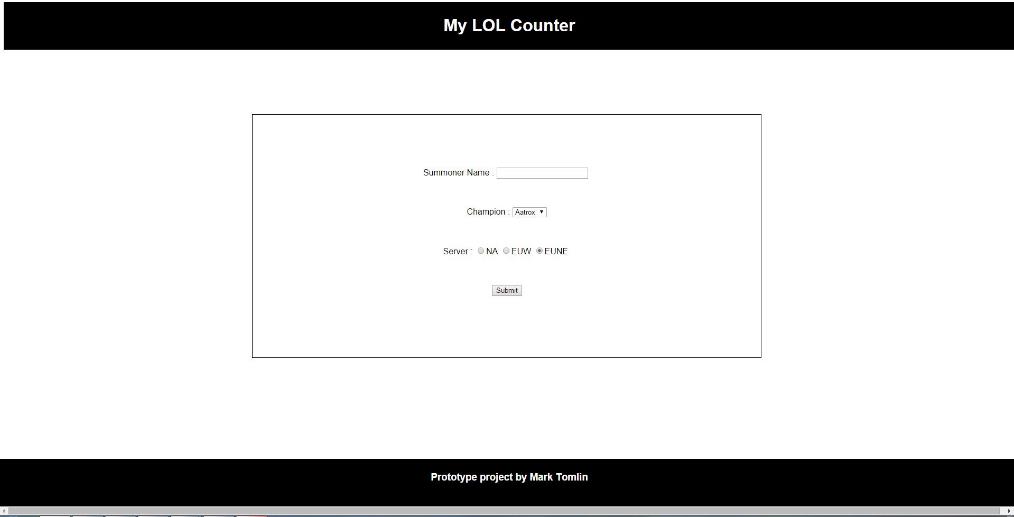
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **Description** | **Date due** | **Date completed** | **Estimated**  **Hrs.** | **Hrs. recorded** | **Comments** |
| 1. | Draw design wireframes | 21/01 | 21/01 | 4 | 3 | Contain 5 necessary pages |
| 2. | Create external CSS file for website | 24/01 | 02/02 | 3 | 2 | Developed from prototype |
| 3. | Design Database Structure | 24/01 | 02/02 | 3 | 2 | Made an ERD |
| 4. | Deploy Database onto server | 24/01 | 04/02 | 4 | 1 | Not many changes from prototype needed |
| 5. | Populate Database with test data | 24/01 | 04/02 | 2 | 5 | Sample of 20 champions |
| 6. | Create homepage with Form | 26/01 | 25/02 | 4 | 4 | Without functionality |
| 7. | Create layout for results page | 26/01 | 30/02 | 6 | 5 | Basic layout and div containers |
| 8. | Re-design & implement rating system | 27/01 | 07/03 | 2 | 4 | Displays in tables. Couldn’t clarify entry effected |
| 9. | Generate champion listings | 30/01 | 16/03 | 9 | 16 | Took much longer anticipated |
| 10. | Add voting for listings | 01/02 | 18/03 | 4 | 5 | Vote scaling wasn’t added |
| 11. | Create log in and register account feature | 02/02 | 19/03 | 3 | 3 | Used PHP sessions |
| 12. | Create account page | 04/02 | 20/03 | 7 | 8 | Some code reused from results page |

**Second Iteration**

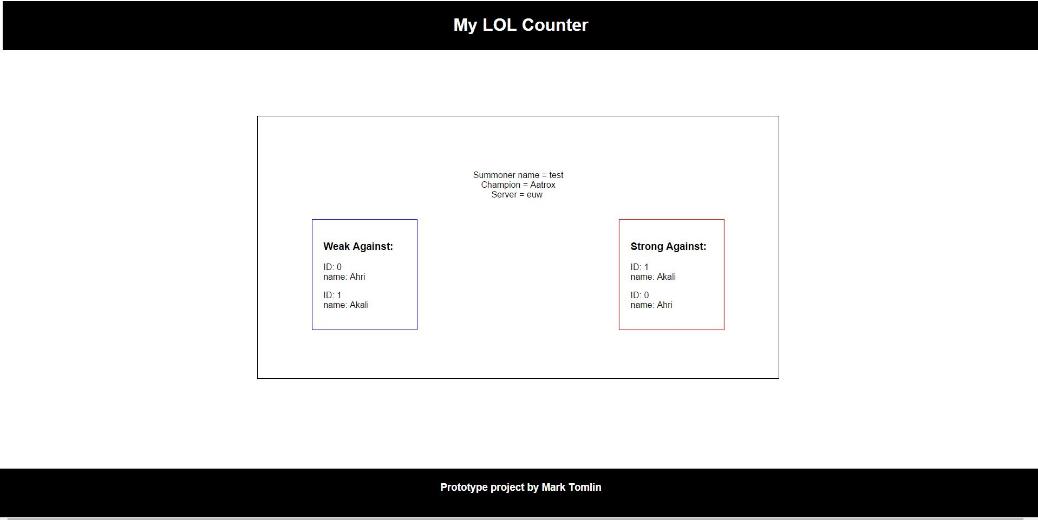
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task** | **Description** | **Date due** | **Date completed** | **Estimated**  **Hrs.** | **Hrs. recorded** | **Comments** |
| 1. | Re-draw design wireframes | 06/02 | N/A | 5 | 0 | Reused for framework |
| 2. | Re-create external CSS file for website | 09/02 | 04/04 | 3 | 1 | Installed framework CSS and resources |
| 3. | Re-Design Database Structure | 07/02 | 03/04 | 3 | 2 | Made new ERD to fit changes realised during 1st iteration |
| 4. | Deploy and populate new database | 09/02 | 16/03 | 2 | 3 | Changes were done during 1st iteration task 9 |
| 5. | Re-create homepage with Form | 11/02 | 13/04 | 4 | 7 | Had learn to use framework components |
| 6. | Re-create layout for results page | 11/02 | 15/04 | 6 | 8 | Had learn to use framework components |
| 7. | Re-design & implement rating system | 12/01 | 15/04 | 3 | 1 | Slightly altered from 1st iteration |
| 8. | Generate champion listings | 15/02 | 18/04 | 11 | 5 | Table formats had to be altered |
| 9. | Add voting for listings | 17/02 | 18/04 | 4 | 2 | Slightly altered from 1st iteration. Buttons were changed for framework use |
| 10. | Create log in and register account feature | 18/02 | 25/04 | 3 | 3 | Took longer due to implementing framework for forms |
| 11. | Re-create account page | 20/02 | 27/04 | 7 | 10 | Extra functionality than 1st iteration |

Appendix D – Prototype

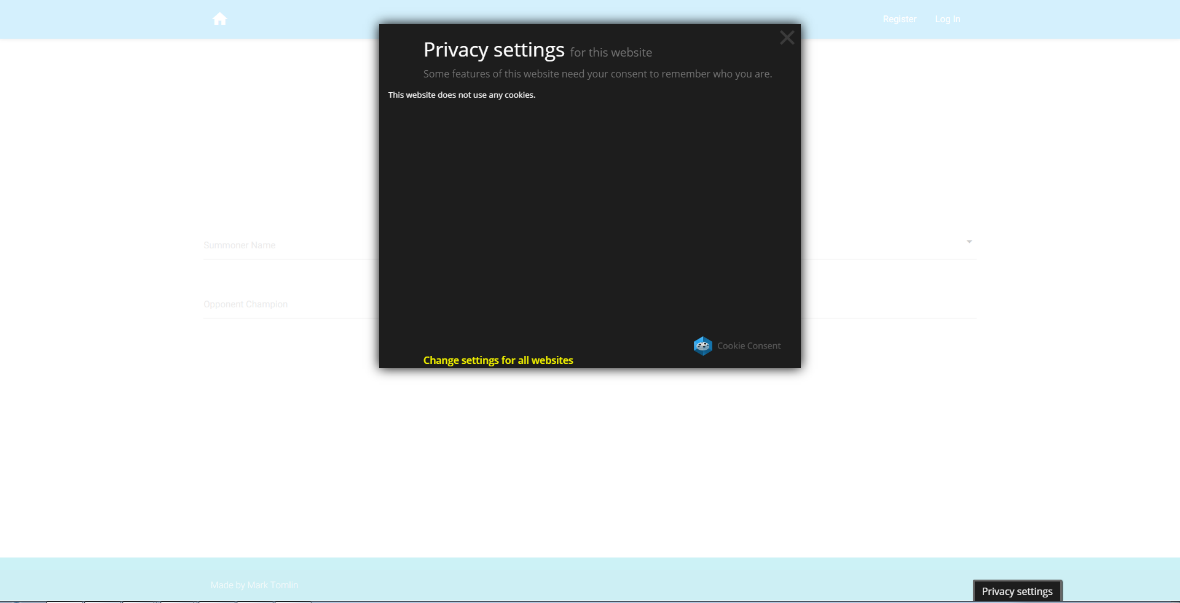
D.1. – Home Page



D.2. – Results Page

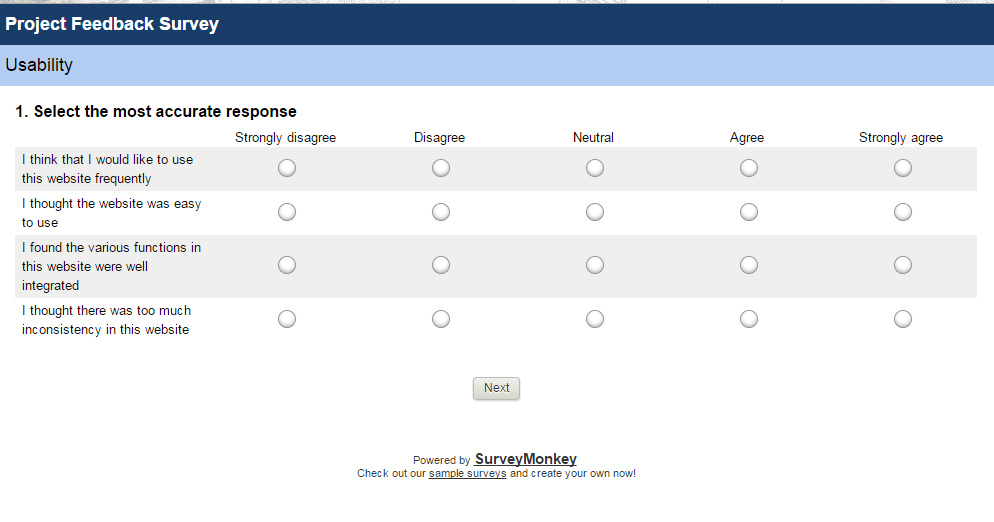


Appendix E – Cookie Banner

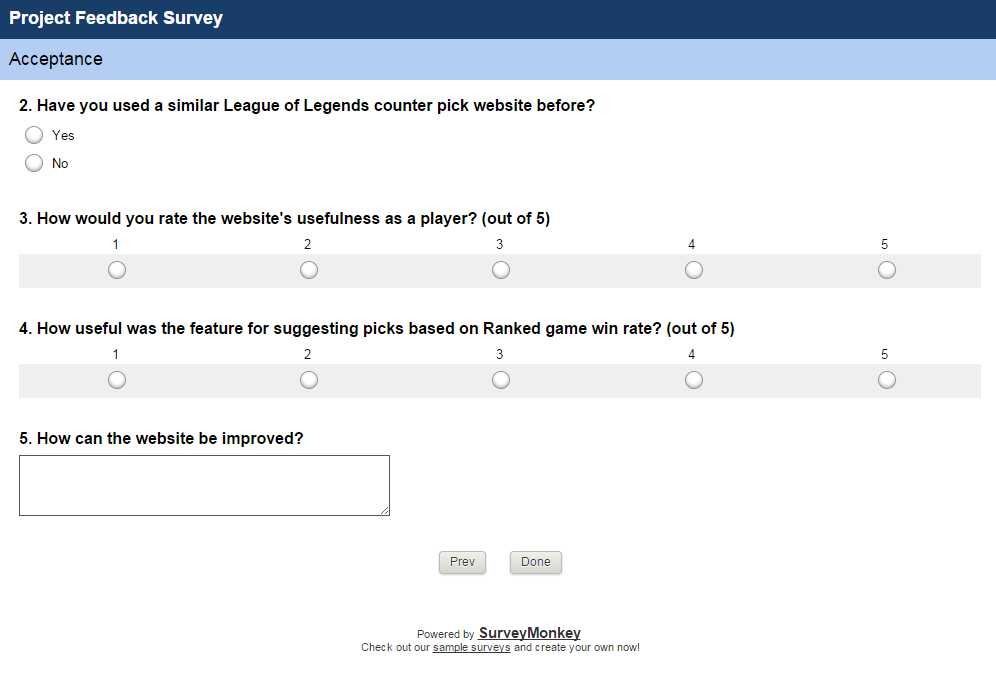


Appendix F – Usability and Acceptance Survey

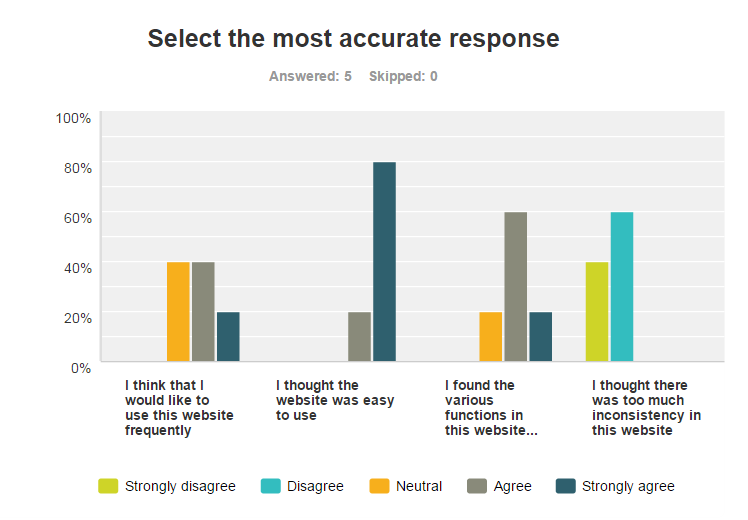
F.1.1. – Survey page 1



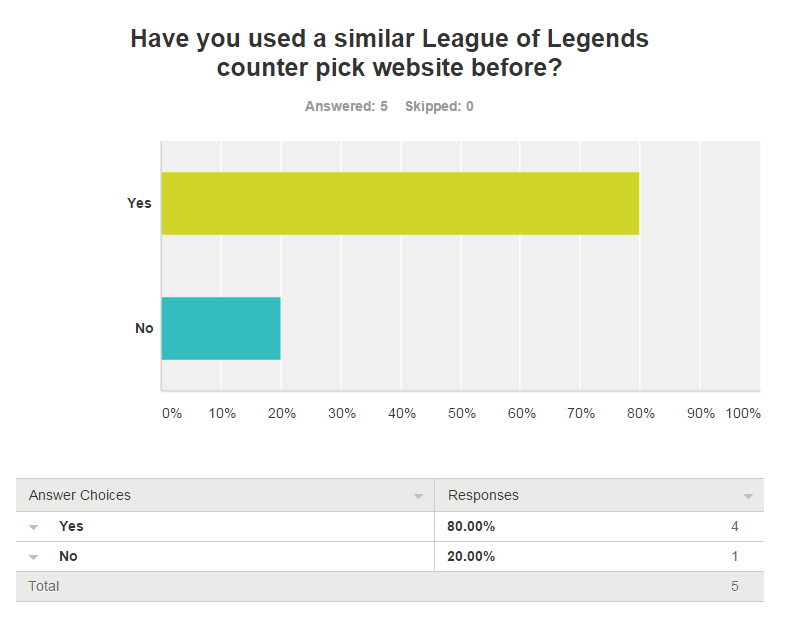
F.1.2. – Survey page 2



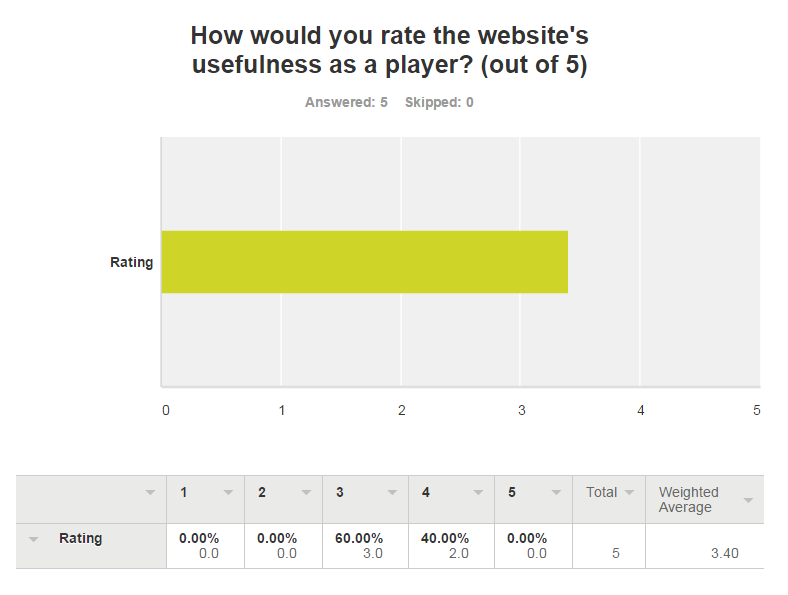
F.2.1. – Survey Results page 1 – Question 1



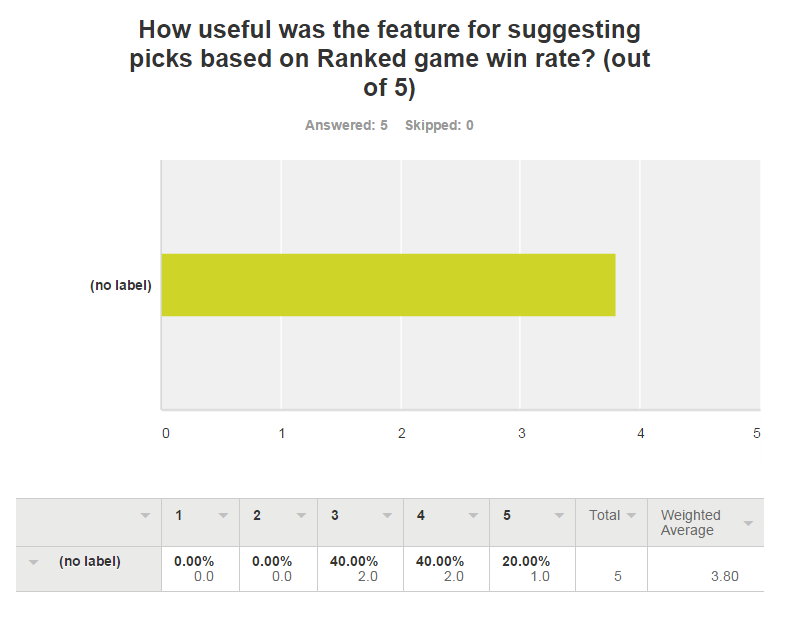
F.2.2. – Survey Results page 2 – Question 2



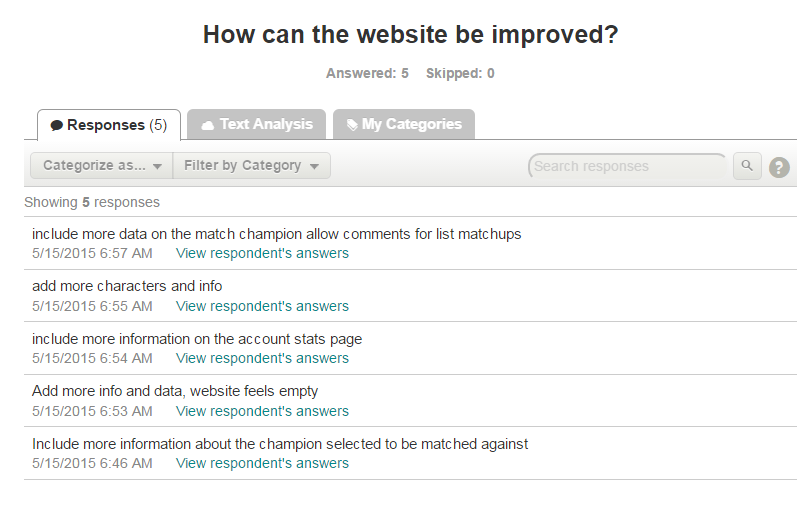
F.2.2. – Survey Results page 2 – Question 3



F.2.3. – Survey Results page 2 – Question 4

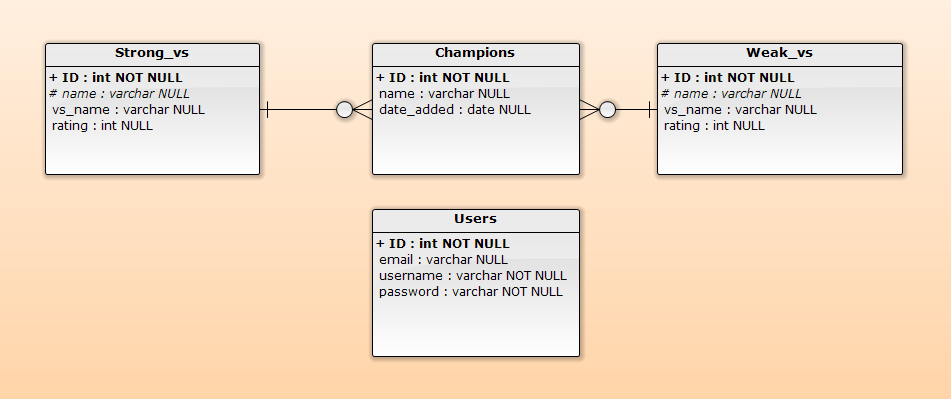


F.2.3. – Survey Results page 2 – Question 5

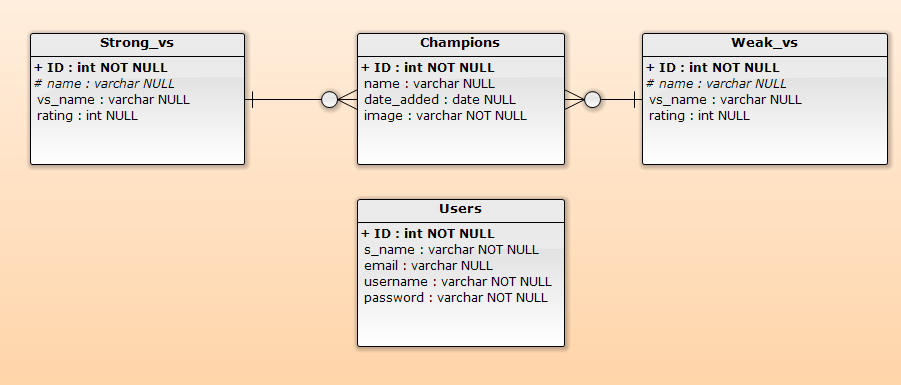


Appendix G – Database ERD

G.1. – First Iteration

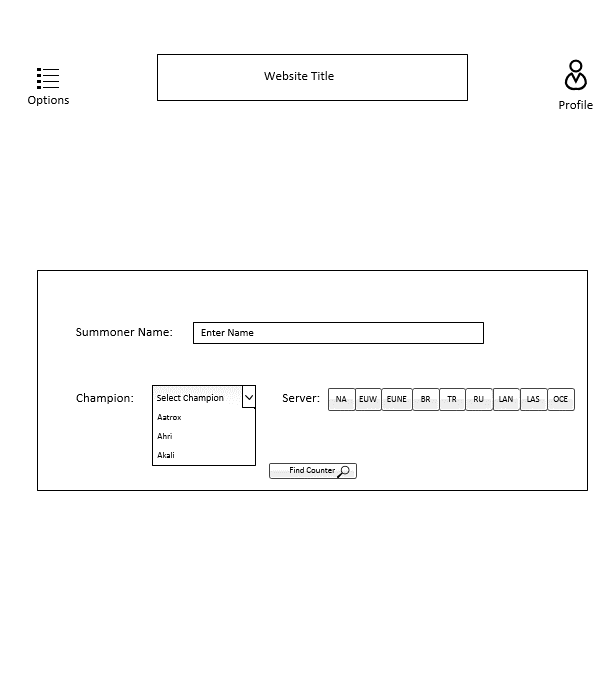


G.2. – Second Iteration

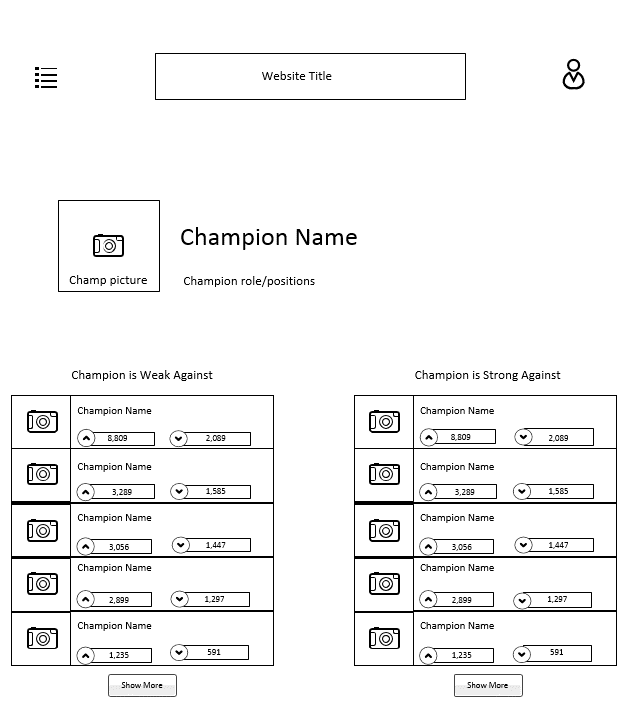


Appendix H – Design Wireframes

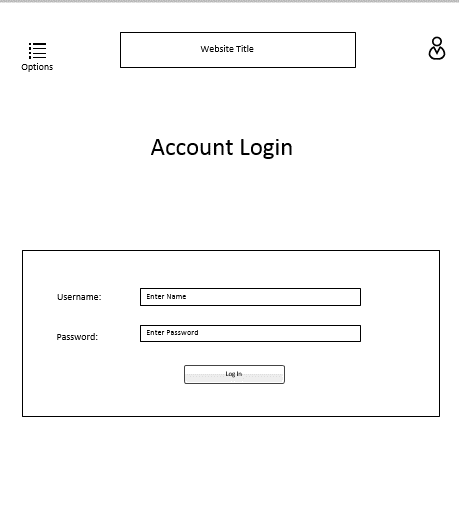
H.1. – Homepage/Search



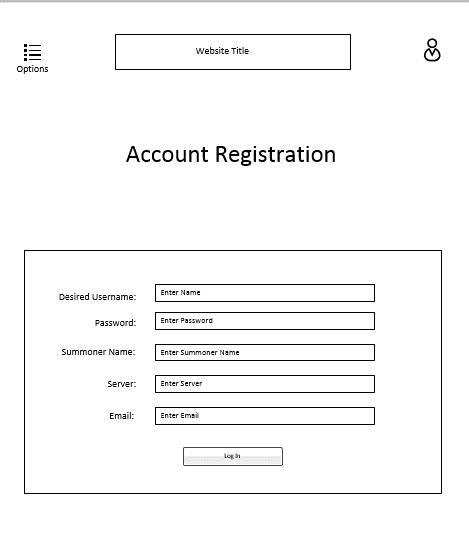
H.2. – Results Page



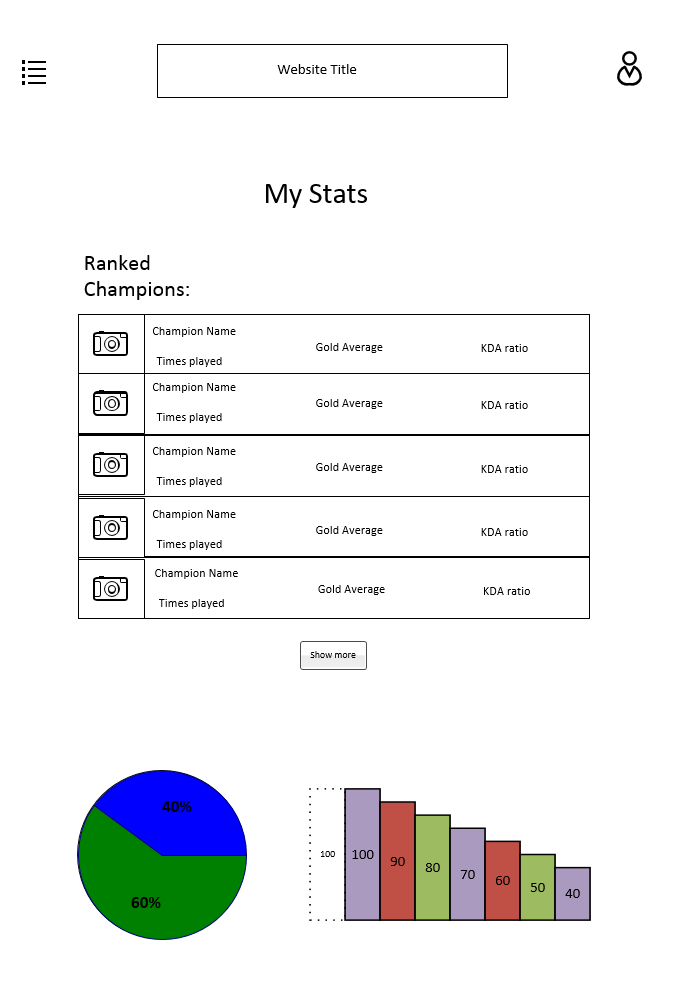
H.3. – Account login form



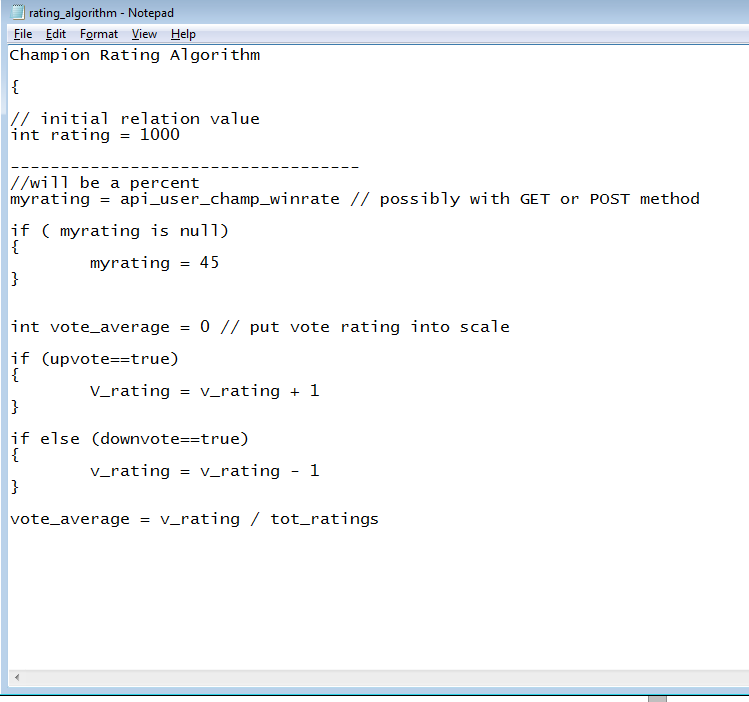
H.4. – Account registration form



H.5. – Account statistics page

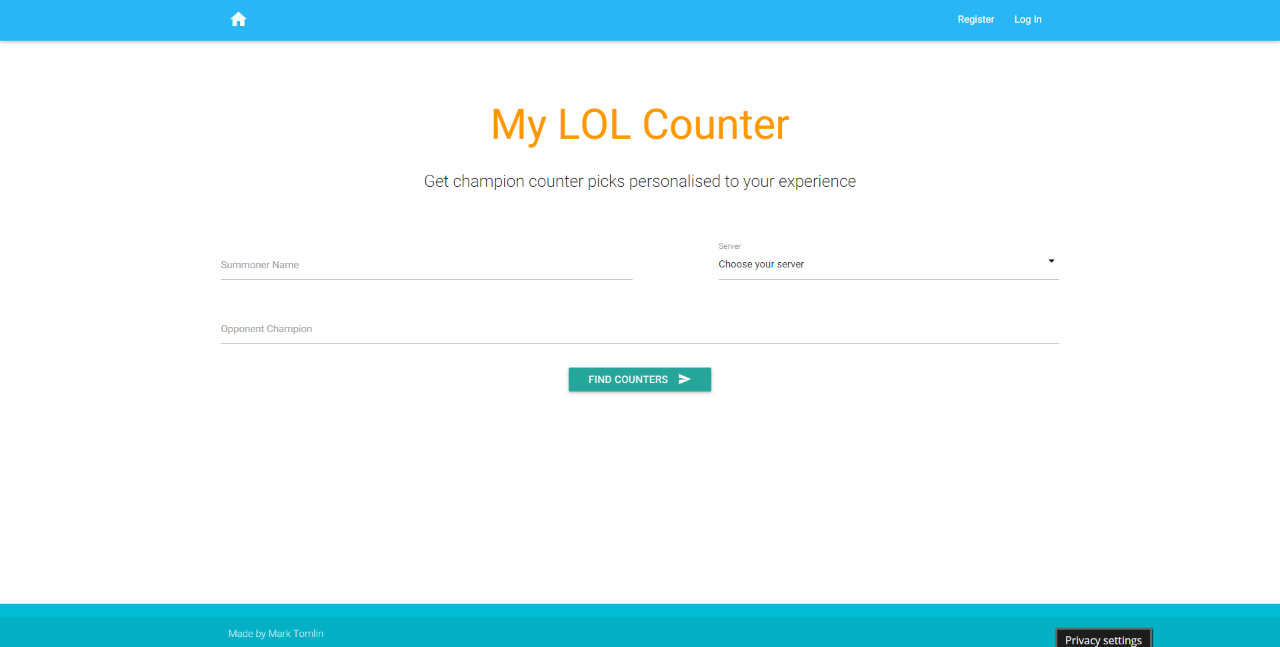


Appendix I – Voting Algorithm Pseudo code

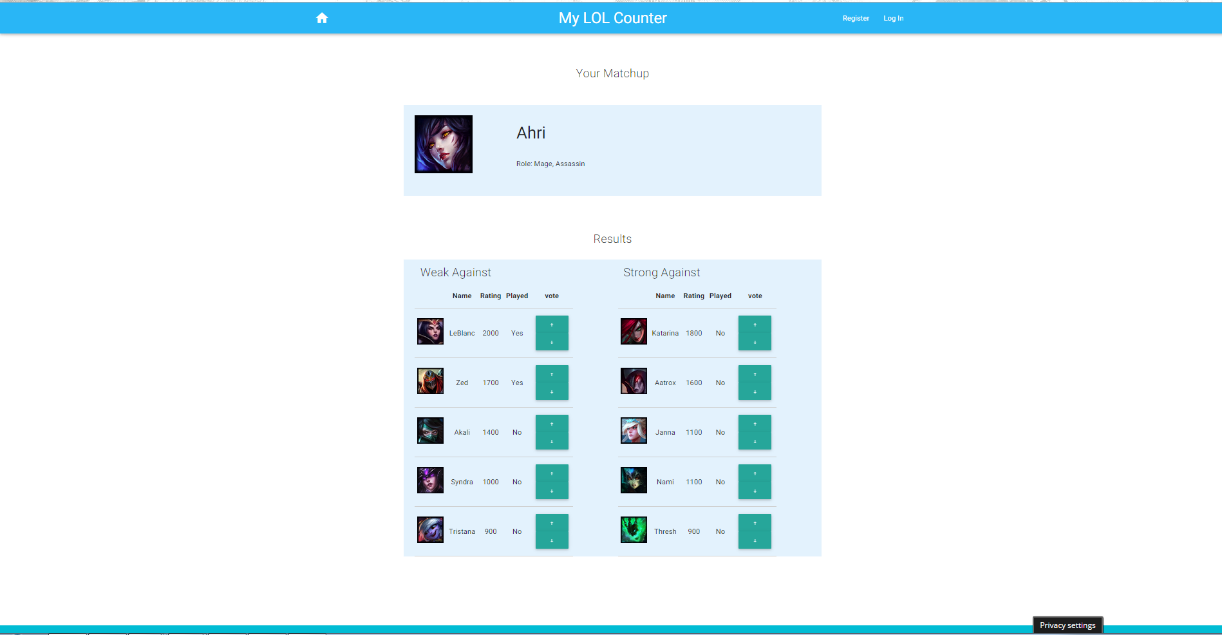


Appendix J – System Screenshots

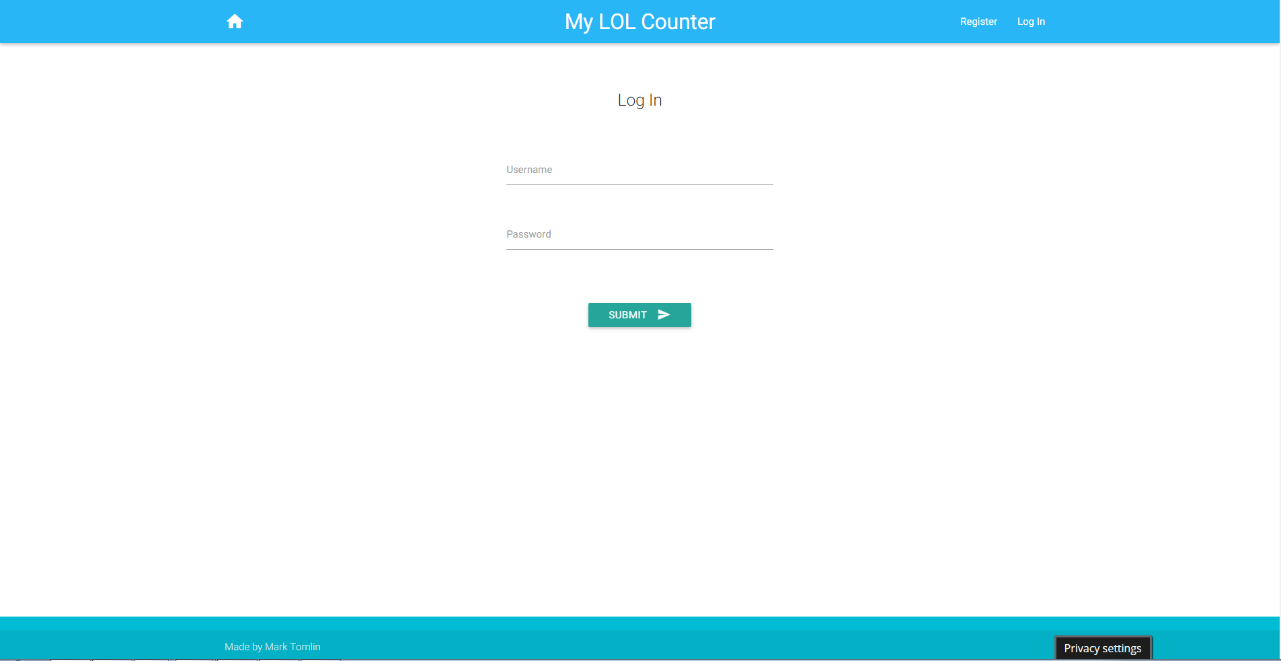
J.1. – Home page



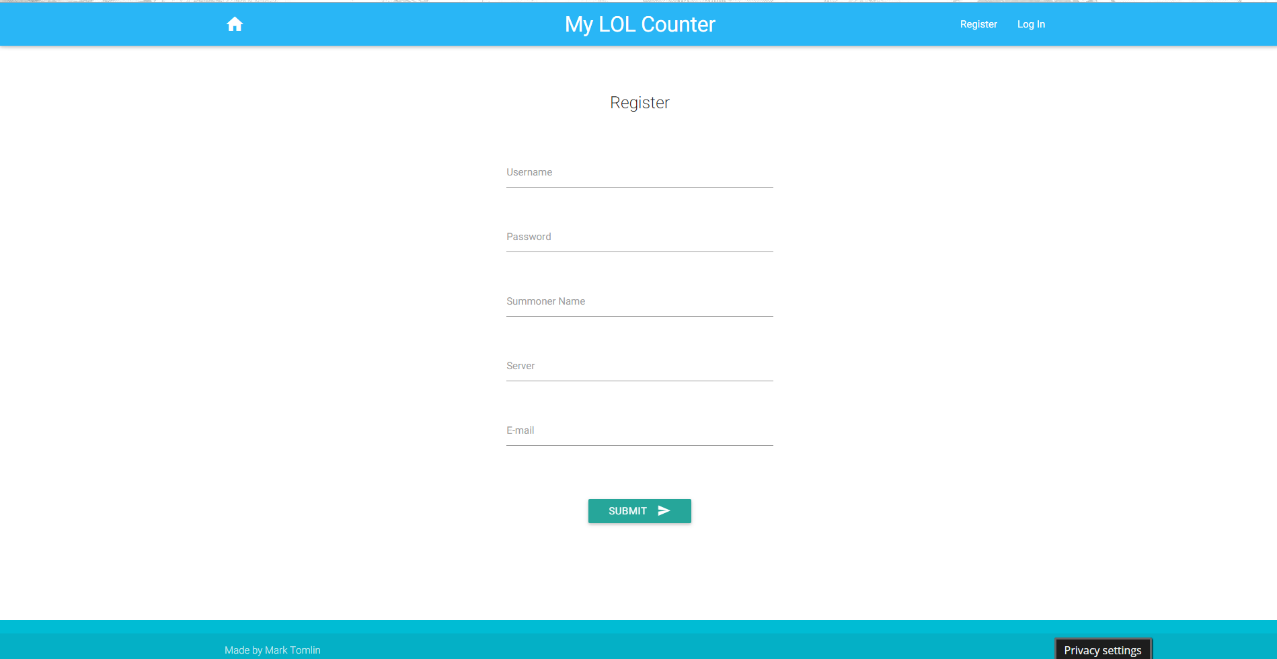
J.2. – Results page



J.3. – Login page



J.4. – Registration page



J.5. – My Stats page

