# SOFT355 Coursework: Dota 2 Match Analyser

Edward Gavin – 10555083

## Functionality

The application I have created is a Dota 2 (Defence of the Ancients) match and statistics viewer. The main purpose of the application is to allow the user to view statistics for a specific match, as well as looking up data on the heroes that are available to play in the game, such as their win rates and recent professional matches that they have been played in.

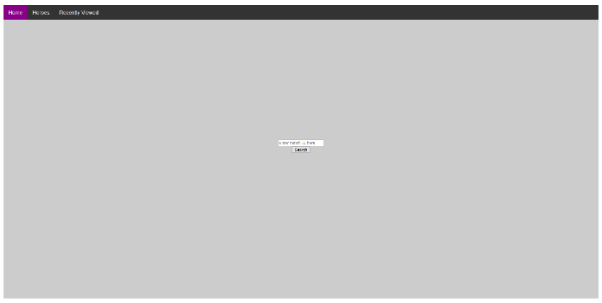
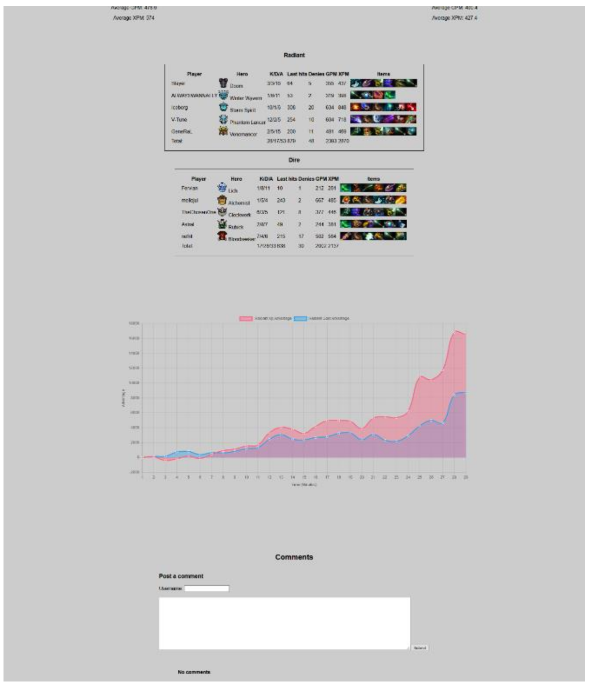


Figure 1: The home page of the application



Average GPM: 478.6
Average XPM: 574

Average GPM: 400.4
Average XPM: 427.4

Radiant

| Player | Hero | K/D/A | Last hits | Denies | GPM | XPM | Items |
|---|---|---|---|---|---|---|---|
| Slayer | Doom | 3/3/15 | 64 | 5 | 355 | 437 | |
| ALWAYSWANNAFLY | Winter Wyvern | 1/8/11 | 53 | 2 | 319 | 388 | |
| Iceberg | Storm Spirit | 10/1/5 | 306 | 20 | 634 | 848 | |
| V-Tune | Phantom Lancer | 12/2/5 | 254 | 10 | 604 | 718 | |
| GeneRaL | Venomancer | 2/5/15 | 200 | 11 | 481 | 469 | |
| Total | | 28/17/53 | 879 | 48 | 2393 | 2870 | |

Dire

| Player | Hero | K/D/A | Last hits | Denies | GPM | XPM | Items |
|---|---|---|---|---|---|---|---|
| Fervian | Lich | 1/8/11 | 10 | 1 | 212 | 281 | |
| meilojul | Alchemist | 1/5/4 | 243 | 2 | 667 | 485 | |
| TheChosenOne | Clockwerk | 6/3/5 | 121 | 8 | 377 | 448 | |
| Astral | Rubick | 2/8/7 | 49 | 2 | 244 | 381 | |
| neth | Bloodseeker | 7/4/8 | 215 | 17 | 502 | 564 | |
| Total | | 17/28/33 | 638 | 30 | 2002 | 2137 | |

Comments

Post a comment

Username:

Submit

No comments

Figure 2: The match details page of the application

The application has been made using Angular. From the home page, the user can interact with the application by either using the search bar to search for a specific Match ID or by using the navigation bar at the top of the page.

The user can interact with the Match Statistics page in a few ways. Firstly, the user is able to hover over an item image to view a tooltip containing the name of that item. Next, the graphs on the page are interactive and the user is able to toggle which data sets are visible by clicking on the them at the top. Finally, there is a comment section available on the page as well, this section allows the user to post a comment about the match that can then be seen by anyone else who is viewing the page. The server that this page uses to retrieve its data is built using Node.js and Express, along with a MongoDB database to store comment information for the page. Socket.io is also in use on this page to enable server-client communication as well, providing the user with a live updating list of comments. All the Dota 2 specific data being used on this page, and in the application in general, is provided by the Opendota API and the Opendota constants library. All of the graphs in the application are being made with ng2-charts (an Angular 2+ wrapper for Chart.js).

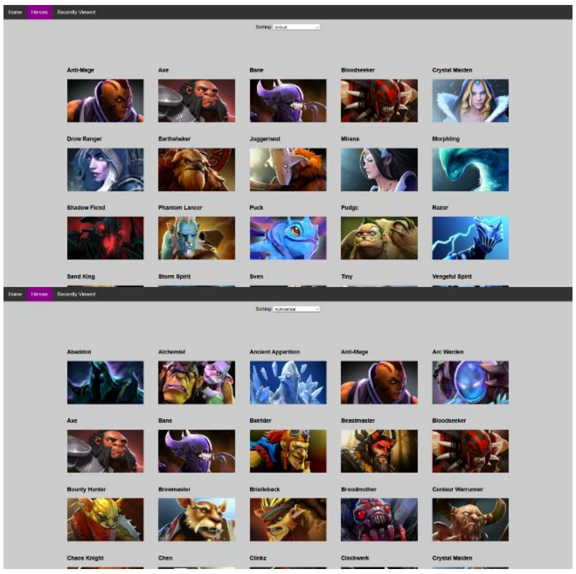
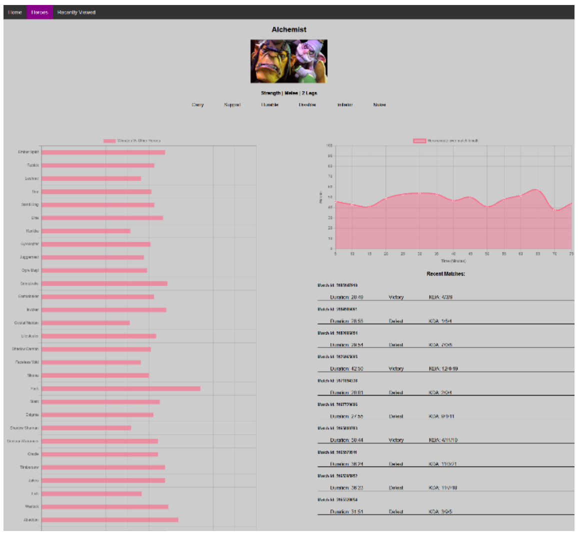


*Figure 3: The heroes page, in default sorting (Top) and Alphabetical sorting (Bottom)*

The Heroes page shows the user a list of all of the heroes that are available to play within the game. The user can interact with the page by using the dropdown list at the top to change the sorting order and they are also able to click on any of the heroes to view more details about them. Hovering over a hero on this page will also change the background colour of the icon depending on the primary attribute of the hero.

A specific hero's detail page will show graphs depicting their win rate vs other heroes, as well as their win rate depending on the length of the match. These graphs can be interacted with in the same way as the ones on the match details page. It will also show a list of recent tournament matches the hero has been played in, which can be clicked on to open their respective match details.

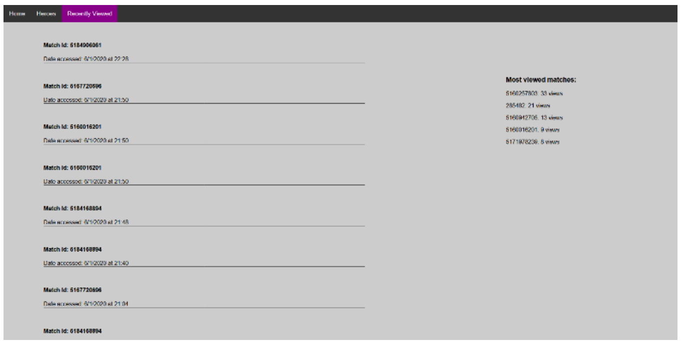
Finally, the recently viewed matches page shows a list of the ten most recent matches that have been viewed by other people, as well as the top 5 most viewed matches of all time. The user can click on any of the matches in the list to view the details of them and the list will update whenever a new match is viewed by another user.



Recent Matches:

```
Match Id: [illegible]
    Duration: 20:49      Victory      KDA: 4/3/8
Match Id: [illegible]
    Duration: 28:55      Defeat       KDA: 1/5/4
Match Id: [illegible]
    Duration: 28:54      Defeat       KDA: 7/3/6
Match Id: [illegible]
    Duration: 42:50      Victory      KDA: 12/4/19
Match Id: [illegible]
    Duration: 20:01      Defeat       KDA: 2/3/4
Match Id: [illegible]
    Duration: 27:55      Defeat       KDA: 8/1/11
Match Id: [illegible]
    Duration: 50:44      Victory      KDA: 4/11/19
Match Id: [illegible]
    Duration: 38:24      Defeat       KDA: 11/3/21
Match Id: [illegible]
    Duration: 36:23      Defeat       KDA: 11/7/10
Match Id: [illegible]
    Duration: 31:51      Defeat       KDA: 3/9/5
```

*Figure 4: The hero details page*



**Match Id: 5184906061**
Date accessed: 6/1/2020 at 22:28

**Match Id: 5167720696**
Date accessed: 6/1/2020 at 21:50

**Match Id: 5160016201**
Date accessed: 6/1/2020 at 21:50

**Match Id: 5160016201**
Date accessed: 6/1/2020 at 21:50

**Match Id: 5184168894**
Date accessed: 6/1/2020 at 21:48

**Match Id: 5184168894**
Date accessed: 6/1/2020 at 21:40

**Match Id: 5167720696**
Date accessed: 6/1/2020 at 21:04

**Match Id: 5184168894**

**Most viewed matches:**
5160257803: 33 views
285482: 21 views
5160912706: 13 views
5160016201: 9 views
5171978239: 8 views

*Figure 5: The recently viewed page*

## Requirements and Design

The target audience for this application is Dota 2 players. The ability to search for match data was included as many players like to go back and look at previous matches that they have been in to see how they have done. Within the match page, the commenting feature was included as it would allow users to discuss the match and what happened which would perhaps allow some users to share unique insight on the match.

The ability to view the statistics of individual heroes was included as it can help players see how a hero is currently performing against other heroes or what their late-game potential is like, which could be useful to them during the drafting phase of a match. Also, the ability to view recently viewed matches was included to show players what everyone else is currently viewing, as well as showing which matches are popular now.

The system uses client-server and server-client communication. The client interacts with the server by sending HTTP requests to the routes which have been defined using Express and the server can interact with the client using web sockets with Socket.io.

On the server, the code is structured into a few files called match-server.js, logic.js, and db.js. The server file is where all the express routes are defined and is the main access point for the server. The db file stores the data and code for handling the database connection and the object structure. Finally, the logic file then stores all the code for getting data from the Opendota API and formatting that into a JSON object that can then be sent to the client. I believe this structure is appropriate as it splits up the separate parts of the application so that each file serves one purpose, and it makes it easier to test the server logic for the unit tests. However, if any more features were to be added to the server, it may be necessary to start splitting up the logic file and putting other parts of the code, such as code to get data from the constants library or code to format parts of the data, into another file to keep it cleaner.

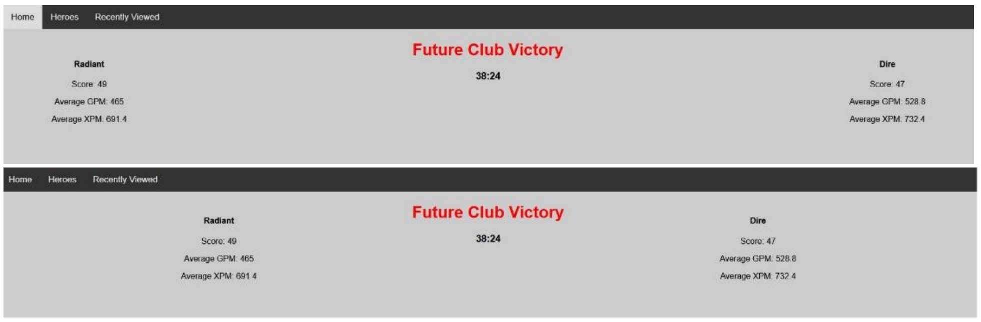
As the client application is an Angular App, the code is structured so that each page has its own module, which is then made up of multiple components. Using separate modules for each page means that the main app module does not get bloated with every single component on the site and means that other pages don't have imports for features that they do not need. Using multiple components means that I can then break up each page into smaller parts that serve a single purpose, which would in turn then keep their controllers cleaner and more readable.

Diagrams of use cases and application structure can be seen in Appendix A.

## Testing and DevOps

During the development of this application, user testing was performed. For this testing, I would give the users a set of instructions to follow that would take them through the various parts of the application, with tasks that would also test to see how intuitive some more hidden functionality was (See Appendix B for the list of tasks). During testing, I found that though some tasks did give a few users some trouble, all were able to complete the tests without any extra prompting from me. However, users were still able to offer feedback on some of the parts of the page design that they thought could be improved.

For example, it was pointed out by a user that it was possible to submit blank comments and repeatedly submit the same comment. As a result of the testing, this was updated to reject empty comments as well as clearing the comment box once the user had posted their comment. It was also mentioned by a few users that as the rest of the match data was centre aligned, the team data in the scoreboard looked out of place when it was at the edges of the page, so this was brought closer to the middle as a result.



Future Club Victory

38:24

Radiant

Score: 49

Average GPM: 465

Average XPM: 691.4

Dire

Score: 47

Average GPM: 528.8

Average XPM: 732.4

Figure 6: Match scoreboard, before (top) and after (bottom)

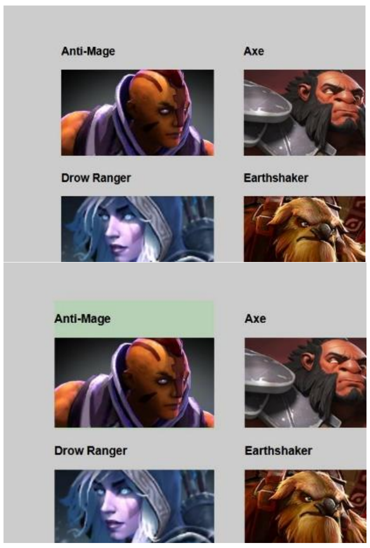


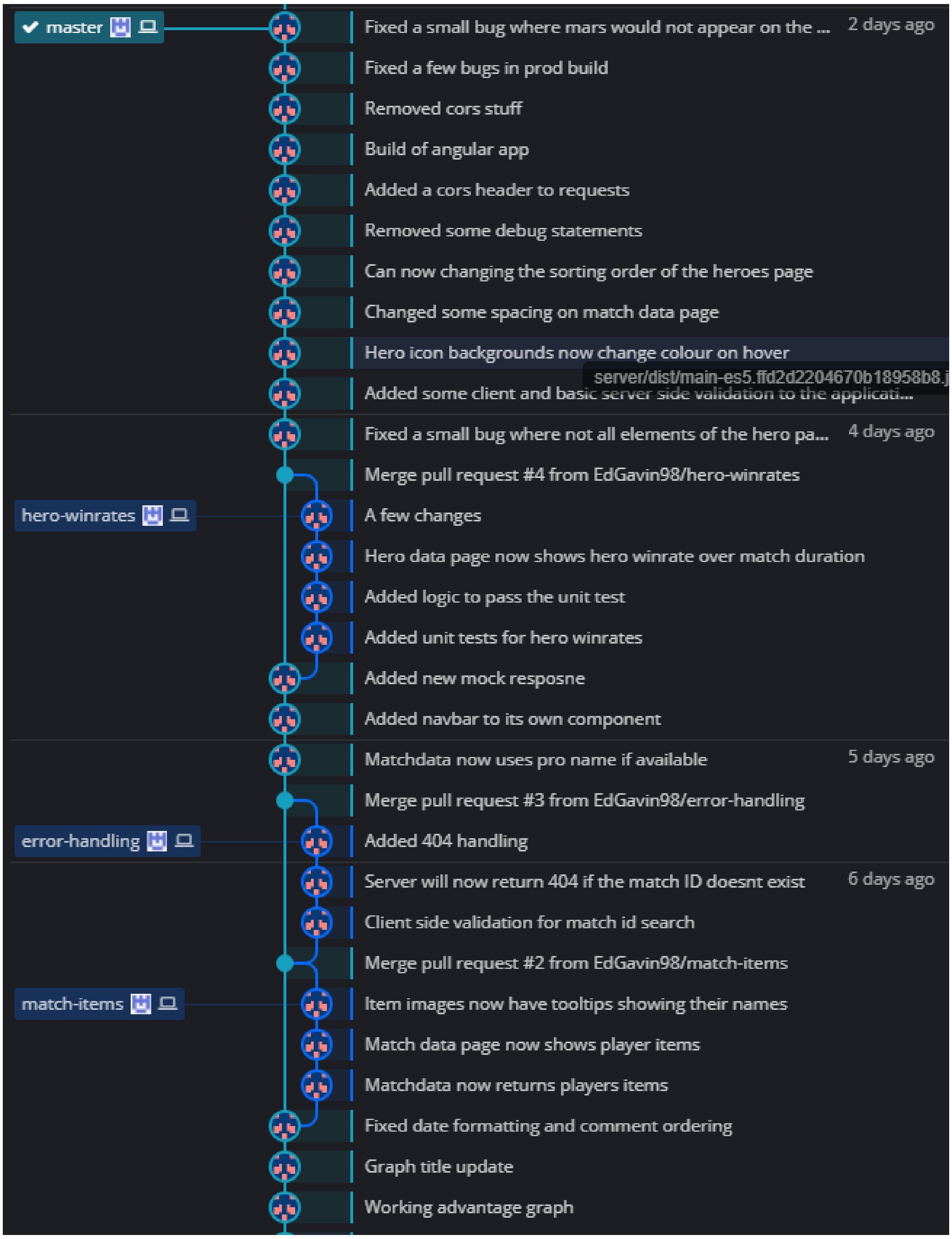
Figure 7: Hero page backgrounds, before (top) and after (bottom)

As a result of the usability testing users also pointed out some errors in the heroes page, such as the fact that unlike other links, the hero icons didn't change their background colour when you hovered over them. Another user also pointed out that it could be difficult to search for a specific hero as they were ordered by ID rather than alphabetically. As a result of this feedback, the hero page was altered so that the background colour would change on hover (Uses the hero's primary attribute to decide the colour) and the ability to sort the heroes in various ways was added (can be seen in Figure 3).

During development, unit testing was used on the server to make sure that it was processing the data it received from the Opendota API properly before sending that data to the client. This helped validate that any new feature that was added to the server was working properly and that the client was receiving all the data that it should be. To perform the unit testing, the NPM package Nock was used to allow me to intercept and mock the API calls that the server was making and replace them with JSON data that was stored with the tests. Mocha and Chai were used to for the testing suite and assertion library.

I believe this testing strategy was appropriate as the server-side tests allowed me to ensure the integrity of the data that I was sending to the client and the usability testing I performed allowed me to make sure that the target audience would be able to use the application intuitively.

Within my development environment, I used git for the version control system. I made use of branching whenever I was working on a new feature. This allowed me to keep the master branch free of broken code and meant new features wouldn't be added to the master branch until they were finished and worked.



master

hero-winrates

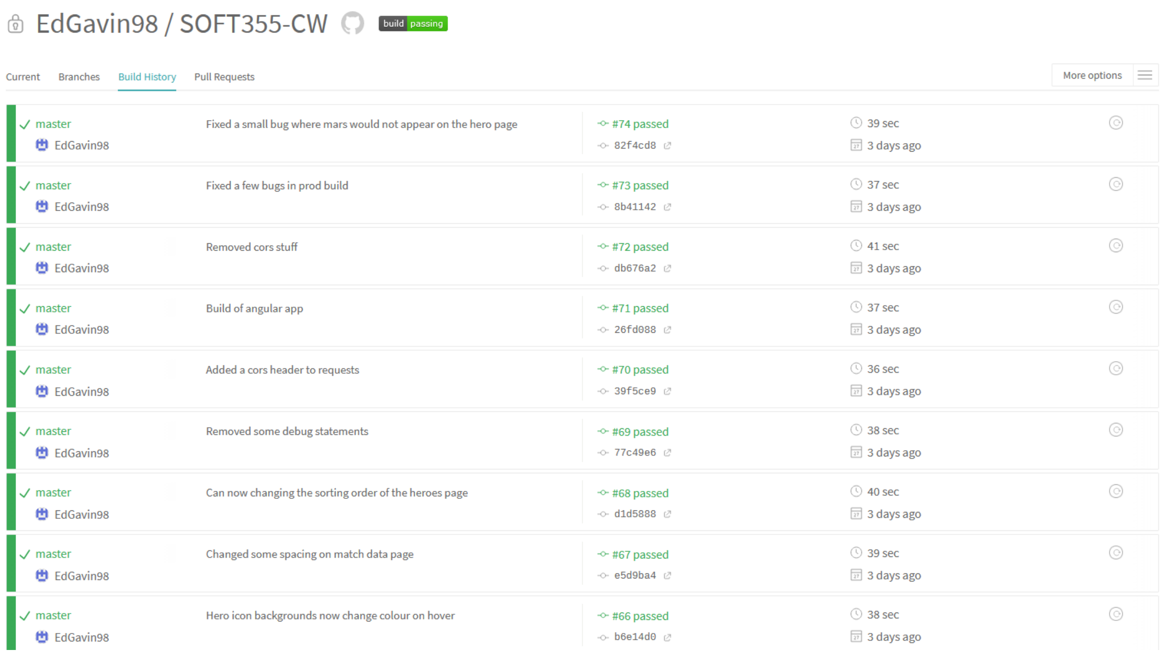
error-handling

match-items

```
Fixed a small bug where mars would not appear on the ...   2 days ago
Fixed a few bugs in prod build
Removed cors stuff
Build of angular app
Added a cors header to requests
Removed some debug statements
Can now changing the sorting order of the heroes page
Changed some spacing on match data page
Hero icon backgrounds now change colour on hover
server/dist/main-es5.ffd2d2204670b18958b8.j
Added some client and basic server side validation to the applicati...
Fixed a small bug where not all elements of the hero pa...   4 days ago
Merge pull request #4 from EdGavin98/hero-winrates
A few changes
Hero data page now shows hero winrate over match duration
Added logic to pass the unit test
Added unit tests for hero winrates
Added new mock resposne
Added navbar to its own component
Matchdata now uses pro name if available   5 days ago
Merge pull request #3 from EdGavin98/error-handling
Added 404 handling
Server will now return 404 if the match ID doesnt exist   6 days ago
Client side validation for match id search
Merge pull request #2 from EdGavin98/match-items
Item images now have tooltips showing their names
Match data page now shows player items
Matchdata now returns players items
Fixed date formatting and comment ordering
Graph title update
Working advantage graph
```

*Figure 4: Overview of some of the commit and branch history (Taken from GitKraken)*

For CI, Travis was used as my platform. This would monitor my git repository and run my suite of tests every time I pushed my changes to the remote repository. Once it had finished running the tests it would send a notification about the status of the build and whether it was broken, working, or had been fixed.



EdGavin98 / SOFT355-CW build passing

Current Branches Build History Pull Requests More options

| | | | |
|---|---|---|---|
| ✓ master<br>EdGavin98 | Fixed a small bug where mars would not appear on the hero page | #74 passed<br>82f4cd8 | 39 sec<br>3 days ago |
| ✓ master<br>EdGavin98 | Fixed a few bugs in prod build | #73 passed<br>8b41142 | 37 sec<br>3 days ago |
| ✓ master<br>EdGavin98 | Removed cors stuff | #72 passed<br>db676a2 | 41 sec<br>3 days ago |
| ✓ master<br>EdGavin98 | Build of angular app | #71 passed<br>26fd088 | 37 sec<br>3 days ago |
| ✓ master<br>EdGavin98 | Added a cors header to requests | #70 passed<br>39f5ce9 | 36 sec<br>3 days ago |
| ✓ master<br>EdGavin98 | Removed some debug statements | #69 passed<br>77c49e6 | 38 sec<br>3 days ago |
| ✓ master<br>EdGavin98 | Can now changing the sorting order of the heroes page | #68 passed<br>d1d5888 | 40 sec<br>3 days ago |
| ✓ master<br>EdGavin98 | Changed some spacing on match data page | #67 passed<br>e5d9ba4 | 39 sec<br>3 days ago |
| ✓ master<br>EdGavin98 | Hero icon backgrounds now change colour on hover | #66 passed<br>b6e14d0 | 38 sec<br>3 days ago |

*Figure 8: Overview of the Travis build history of this project*

I used this CI system to stay notified about how the builds were doing and get immediate notifications if something had broken once it was pushed to the git repository. It was also used at the end of developing a feature as Travis would begin running a new test when I opened a pull request to merge the feature into the master branch and the results of this test would be displayed within the pull request so that I knew everything would be working once I merged into the master branch.

## Reflection

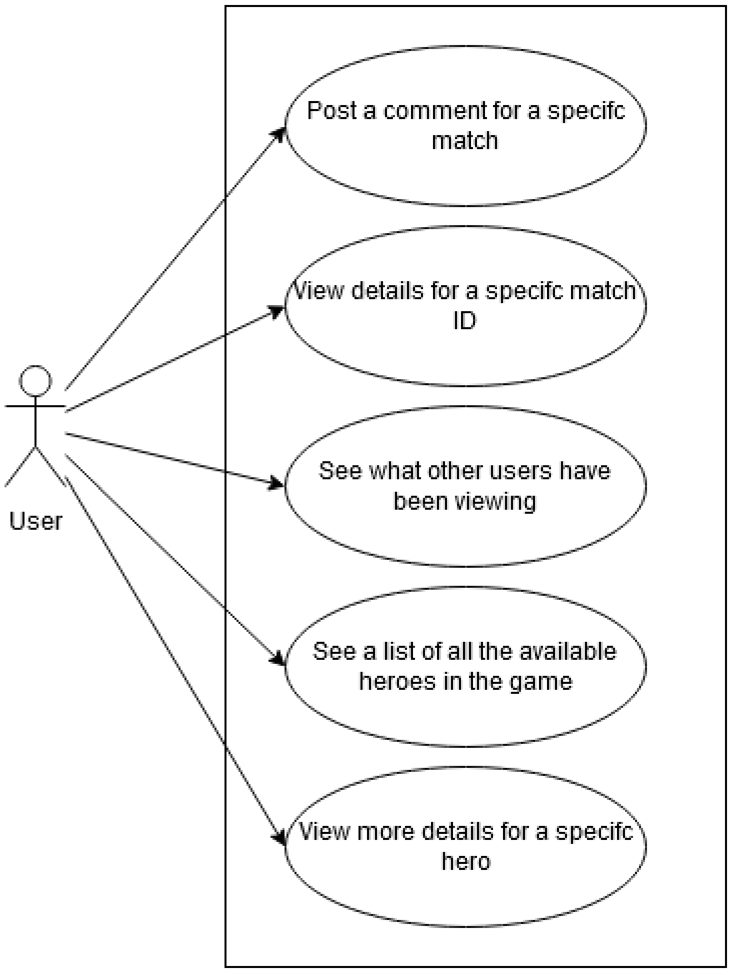
Overall, I believe the project went very well. The final application runs very smoothly and there are no major bugs within it. I believe choosing to use the MEAN stack was a good choice for an application of this type as Node, Mongo and Express gave me the ability to easily develop the server-side parts of the application and on the client-side, Angular allowed me to be able to easily add and keep track of new components and features as the development went on. Usability testing also returned good results and the application is well-liked by members of its target audience.

However, I have learned a lot over the course of this project and if I were to go and create a new project, or modify this one in the future, there are things I would change. For example, I would add automated testing to the client-side part of the application as well as the server-side, this would have allowed me to make sure the application was doing what I wanted it and may have even caused me notice or at least consider bugs such as being able to post blank comments earlier. Also, as Angular is a Typescript based system, another thing I would like to improve on is using more Typescript based features in the development of the application, as well as taking more advantage of the other tools that Angular offers.

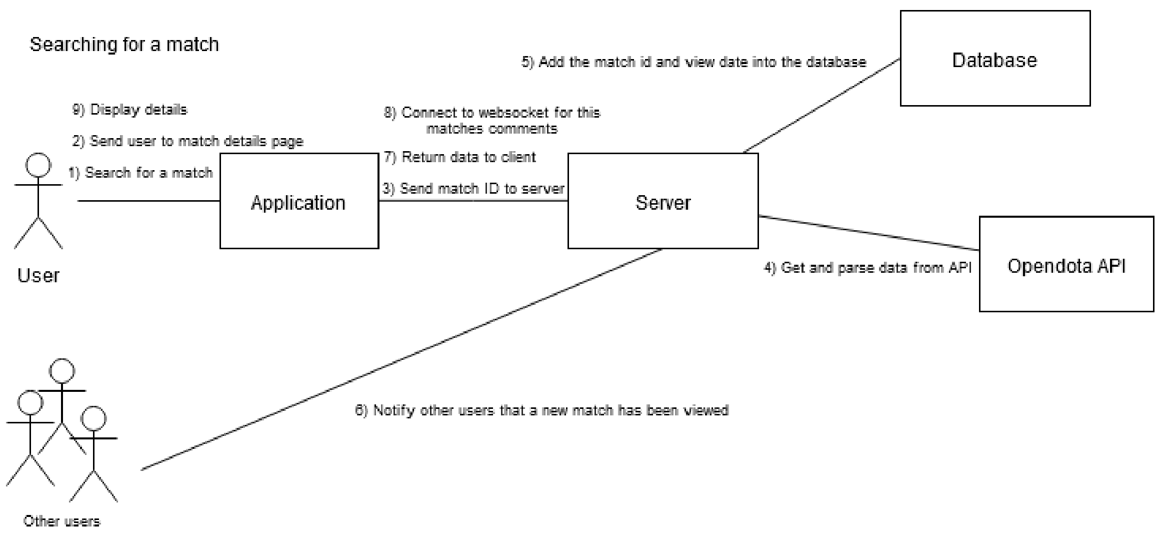
## Appendices.

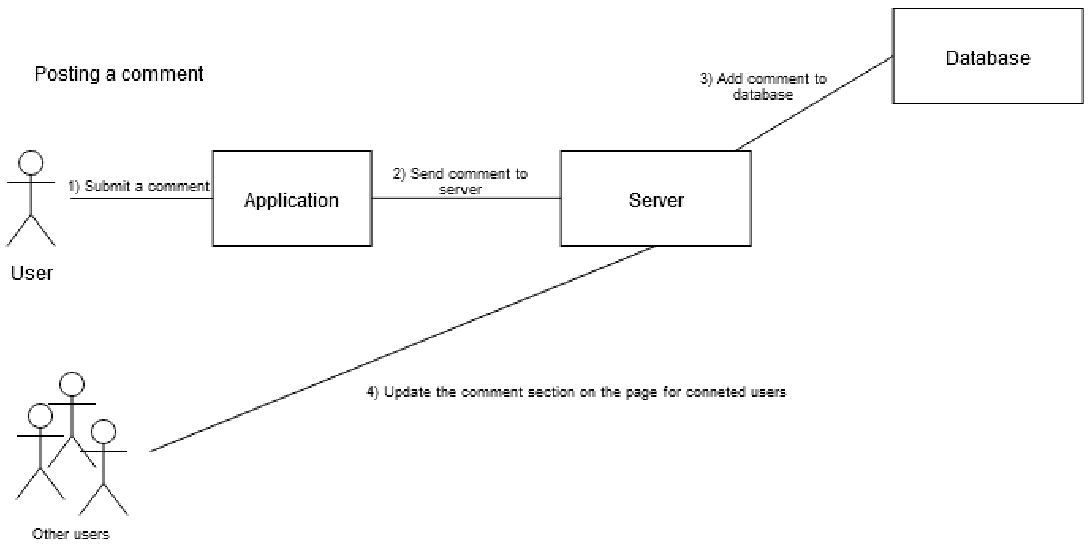
### Appendix A – Diagrams

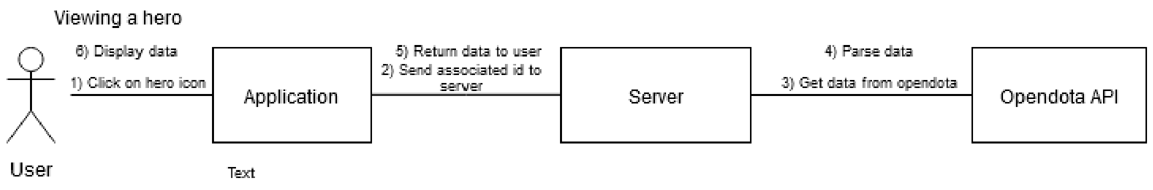
Use case:

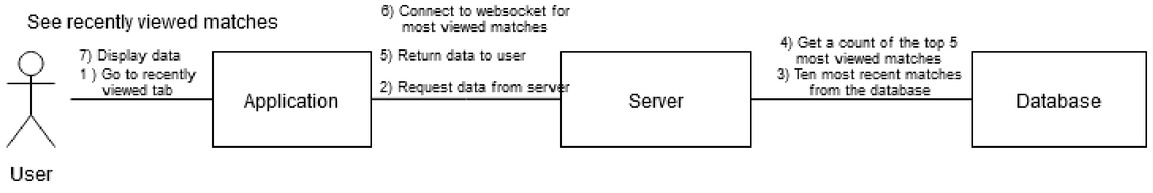

Post a comment for a specifc match
View details for a specifc match ID
See what other users have been viewing
See a list of all the available heroes in the game
View more details for a specifc hero
User

Requirements modelling:

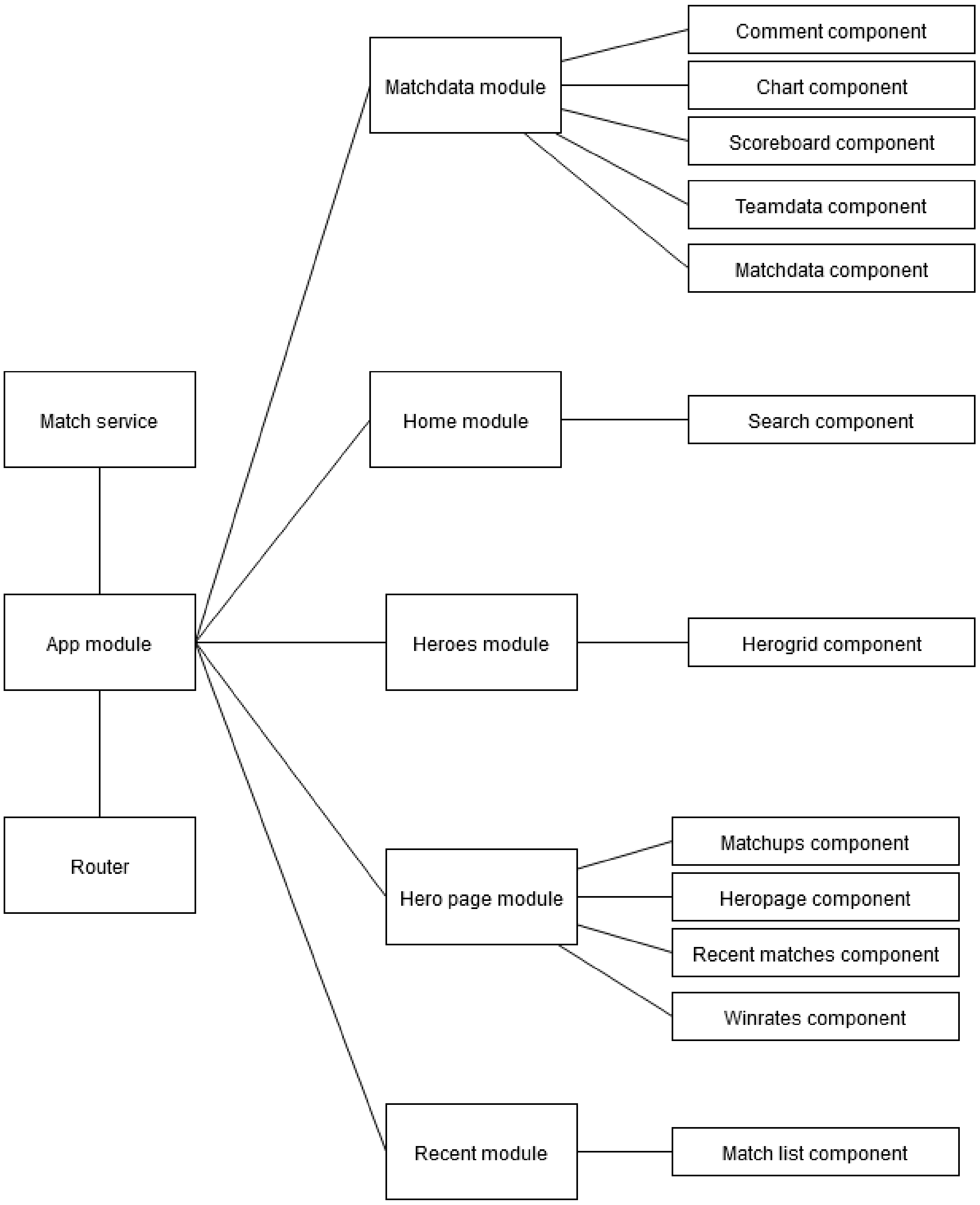

Searching for a match
5) Add the match id and view date into the database
Database
9) Display details
8) Connect to websocket for this matches comments
2) Send user to match details page
7) Return data to client
1) Search for a match
Application
3) Send match ID to server
Server
User
4) Get and parse data from API
Opendota API
6) Notify other users that a new match has been viewed
Other users


Posting a comment
Database
3) Add comment to database
1) Submit a comment
Application
2) Send comment to server
Server
User
4) Update the comment section on the page for conneted users
Other users




See recently viewed matches
6) Connect to websocket for most viewed matches
7) Display data
1 ) Go to recently viewed tab
Application
5) Return data to user
2) Request data from server
Server
4) Get a count of the top 5 most viewed matches
3) Ten most recent matches from the database
Database
User

Final diagram of app:



## Appendix B – Test task list

- Search for a match
- What is the average GPM for radiant?
- What is the name of an item? (not from memory)
- Set the graph so that it only shows the gold advantage
- Post a comment
- View a list of all the heroes
- View more details for Skywrath Mage
- What is the win rate for Skywrath Mage against Huskar
    - Due to the height of the graph, this task was testing to see if the user would realise that they can hover over a bar to see the value
- View details for one of Skywrath Mage's matches
- View the matches other players have been looking at
- View details for one of those matches