Online Games Tournament Management

Ethan Horrigan

B.Sc.(Hons) in Software Development

April 21, 2020

Final Year Project

Advised by: Dr John French

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

1 Introduction **5**

2 Context **6**
2.1 League of Legends 6
2.2 Rating 7
2.3 Custom Games 7
2.3.1 Competitive Tournaments 8
2.3.2 More filler 8
2.4 Filler 9

3 Methodology **11**
3.1 Development Methodology 11
3.2 Testing 11
3.3 Source Control 13
3.4 Technologies Selection Criteria 13

4 Technology Review **15**
4.1 Angular 15
4.1.1 Why Angular? 16
4.2 SQLite 16
4.3 PostgreSQL 17
4.4 PgAdmin 18
4.5 Heroku 18
4.6 Firebase 18
4.7 Flask 18
4.8 RiotWatcher 18
4.9 Postman 18
4.10 JSON 18
4.11 Karma and Jasmine 18
4.12 Python Unittest 18
4.13 The Glicko System 19

4.14 Gale Shapley Algorithm 19
4.15 Bootstrap and Angular Material 19
4.16 Bcrypt 19

5 System Design **20**

6 System Evaluation **21**

7 Conclusion **22**

About this project

Abstract Competitive online gaming has seen a significant increase in popularity in recent times, whether watching or participating, competitive games can consume a large portion of our free time. Organising tournaments require organisation and rules. To ensure the rules are upheld require some form of administration from a system or individual. Issues can occur when an individual is responsible for managing these tournaments, for example, if a tournament has a fixed schedule but the person responsible for managing the tournament is unavailable, then the tournament game must be postponed. Administrators are also required to ensure matchmaking fairness between teams which can be very time consuming and inefficient.

Authors Ethan Horrrigan

Chapter 1

Introduction

Competitive online gaming has seen a significant increase in popularity in recent times. The estimated global esports audience was estimated at 335 million people in 2017 generating a revenue of more than $900M with an estimated growth of over $1600M in 2021. [1] Yuri Seo and Sang-Uk Jung [2] outlined why people play or spectate competitive games. The main factors include entertainment and gaining a better understanding of a game. Whether spectating or participating, competitive games can consume a large portion of our free time. Organising matches need some form of administration to ensure rules are upheld. A person or system is usually responsible for this. Issues occur when an individual is responsible for managing these games, for example, if a match has a fixed schedule but the person responsible for managing the game is unavailable, then the tournament game must be postponed. Administrators are also required to ensure matchmaking fairness between teams which can be very time consuming and unpredictable. These factors are the reason why I've developed a service for people to manage and administrate their events or matches.

Chapter 2

Context

- Provide a context for your project.
- Set out the objectives of the project
- Briefly list each chapter / section and provide a 1-2 line description of what each section contains.
- List the resource URL (GitHub address) for the project and provide a brief list of the main elements at the URL.

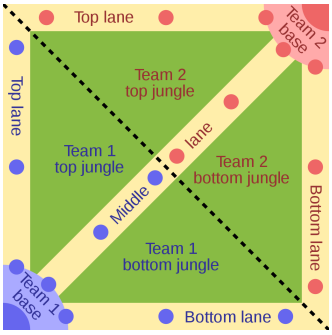
2.1 League of Legends

NOTE: Relevant information to my solution. NOTE: Remove usless information. NOTE: How tournaments work (Online) League of Legends is a free-to-play multiplayer online battle arena (MOBA). A team consists of five players on either side. The map consists of three lanes (Top lane, Middle lane and Bottom lane) and a Jungle on each side of the map. Each player is responsible for fulfilling these lanes and controlling their characters (champions). The main objective of the game is to defeat the enemy nexus (base). Players can accomplish this by fighting the opposition in solo fights or team fights. Players can purchase items throughout the game to strengthen their character, all players gain experience and gold by killing minions and other players. This is the main source of income for players to upgrade their items. Players can build damage or resistances depending on what characters the enemies are playing. League of Legends is heavily Team-Based. The skill level of a team is an important condition for team effectiveness. The team's skill level is based on the combination of skills and attributes of its members.

2.2 Rating

2.3 Custom Games

This creates a very competitive environment. Players also acquire ranks, which are broken up into several tiers: Iron, Bronze, Silver, Gold, Platinum, Diamond, Master, Grandmaster and Challenger. All tiers have four divisions within each tier up until master. A division contains 100 league points (LP) that players gain or lose depending on match outcome. 78.31% of the player base are either bronze, silver or gold. Whereas only 0.02% of players are challenger tier [3].



League of Legends is heavily Team-Based. The skill level of a team is an important condition for team effectiveness. The team's skill level is based on the combination of skills of its members. A team consists of five players on either side. Each player is responsible for controlling their characters (champions) to defeat the enemy in solo duels and/or contriubte in team fights with the end goal of destroying the nexus. Players can purchase items throughout the game to strengthen their character, all players gain experience and gold by killing minions and other players. This is the main source of income for players to upgrade their items. Players can build damage or resisitances depending on the enemies character.

2.3.1 Competitive Tournaments

Quisque vel erat a justo volutpat auctor a nec odio. Sed rhoncus augue sit amet nisl tincidunt, vitae cursus tellus efficitur. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque et auctor dui. Fusce ornare odio ipsum, et laoreet mi molestie sed. Cras at massa sit amet ipsum gravida aliquam. Nulla suscipit porta imperdiet. Fusce eros neque, bibendum sit amet consequat non, pulvinar quis ipsum.

2.3.2 More filler

Donec fermentum sapien ac rhoncus egestas. Nullam condimentum condimentum eros sit amet semper. Nam maximus condimentum ligula. Praesent faucibus in nisi vitae tempus. Sed pellentesque eleifend ante, ac malesuada nibh dapibus nec. Phasellus nisi erat, pulvinar vel sagittis sed, auctor et magna. Quisque finibus augue elit, consequat dignissim purus mollis nec. Duis ultricies euismod tortor, nec sodales libero pellentesque et. Interdum et malesuada fames ac ante ipsum primis in faucibus.

Donec id interdum felis, in semper lacus. Mauris volutpat justo at ex dignissim, sit amet viverra massa pellentesque. Suspendisse potenti. Praesent sit amet ipsum non nibh eleifend pretium. In pretium sapien quam, nec pretium leo consequat nec. Pellentesque non dui lacus. Aenean sed massa lacinia, vehicula ante et, sagittis leo. Sed nec nisl ac tellus scelerisque consequat. Ut arcu metus, eleifend rhoncus sapien sed, consequat tincidunt erat. Cras ut vulputate ipsum.

Curabitur et efficitur augue. Proin condimentum ultrices facilisis. Mauris nisi ante, ultrices sed libero eget, ultrices malesuada augue. Morbi libero magna, faucibus in nunc vitae, ultricies efficitur nisl. Donec eleifend elementum massa, sed eleifend velit aliquet gravida. In ac mattis est, quis sodales

neque. Etiam finibus quis tortor eu consequat. Nullam condimentum est eget pulvinar ultricies. Suspendisse ut maximus quam, sed rhoncus urna.

2.4 Filler

Phasellus eu tellus tristique nulla porttitor convallis. Vestibulum ac est eget diam mollis consectetur. Donec egestas facilisis consectetur. Donec magna orci, dignissim vel sem quis, efficitur condimentum felis. Donec mollis leo a nulla imperdiet, in bibendum augue varius. Quisque molestie massa enim, vitae ornare lacus imperdiet non. Donec et ipsum id ante imperdiet mollis. Nullam est est, euismod sit amet cursus a, feugiat a lectus. Integer sed mauris dolor.

Mauris blandit neque tortor, consequat aliquam nisi aliquam vitae. Integer urna dolor, fermentum ut iaculis ut, semper eu lacus. Curabitur mollis at lectus at venenatis. Donec fringilla diam ac risus imperdiet suscipit. Aliquam convallis quam vitae turpis interdum, quis pharetra lacus tincidunt. Nam dictum maximus lectus, vitae faucibus ante. Morbi accumsan velit nec massa tincidunt porttitor. Nullam gravida at justo id viverra. Mauris ante nulla, eleifend vitae sem vitae, porttitor lobortis eros.

Cras tincidunt elit id nisi aliquam, id convallis ex bibendum. Sed vel odio fringilla, congue leo quis, aliquam metus. Nunc tempor vehicula lorem eu ultrices. Curabitur at libero luctus, gravida lectus sed, viverra mi. Cras ultrices aliquet elementum. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed metus ante, suscipit sit amet finibus ut, gravida et orci. Nunc est odio, luctus quis diam in, porta molestie magna. Interdum et malesuada fames ac ante ipsum primis in faucibus. Mauris pulvinar lacus odio, luctus tincidunt magna auctor ut. Ut fermentum nisl rhoncus, tempus nulla eget, faucibus tortor. Suspendisse eu ex nec nunc mollis pulvinar. Nunc luctus tempus tellus eleifend porta. Nulla scelerisque porttitor turpis porttitor mollis.

Duis elementum efficitur auctor. Nam nisi nulla, fermentum sed arcu vel, posuere semper dui. Fusce ac imperdiet felis. Aenean quis vestibulum nisl. Integer sit amet tristique neque, at suscipit tortor. Morbi et placerat ante, vel molestie dui. Vivamus in nibh eget massa facilisis accumsan. Nunc et purus ac urna fermentum ultrices eget sit amet justo. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Cras elementum dui nunc, ac tempor odio semper et. Ut est ipsum, sollicitudin eleifend nisl eu, scelerisque cursus nunc. Nam at lectus vulputate, volutpat tellus vel, pharetra mauris. Integer at aliquam massa, at iaculis sem. Morbi nec imperdiet odio. In hac habitasse platea dictumst.

Mauris a neque lobortis, venenatis erat ut, eleifend quam. Nullam tincidunt tellus quis ligula bibendum, a malesuada erat gravida. Phasellus eget tellus non risus tincidunt sagittis condimentum quis enim. Donec feugiat sapien sit amet tincidunt fringilla. Vivamus in urna accumsan, vehicula sem in, sodales mauris. Aenean odio eros, tristique non varius id, tincidunt et neque. Maecenas tempor, ipsum et sollicitudin rhoncus, nibh eros tempus dolor, vitae dictum justo massa in eros. Proin nec lorem urna. In ullamcorper vitae felis sit amet tincidunt. Maecenas consectetur iaculis est, eu finibus mi scelerisque et. Nulla id ex varius, ultrices eros nec, luctus est. Aenean ac ex eget dui pretium mattis. Ut vitae nunc lectus. Proin suscipit risus eget ligula sollicitudin vulputate et id lectus.

Chapter 3

Methodology

3.1 Development Methodology

Project meetings were established at the beginning of development, Initial meetings consisted of brainstorming and considering project ideas. During this period, I conducted research on various technologies that could possibly be used throughout the project. I began development once the project was defined and understood what technologies were suitable for use throughout. Every week I would meet with my supervisor and discuss what has been implemented in the past week and what I will work on for the upcoming week. I took an iterative approach in the development of this project so I could see significant developments in the project.

3.2 Testing

I opted to use System Testing I opted to use System Testing as the main type of testing for the project as this suited my workflow. I wanted to implement the functionality of client-side elements before testing. Unit tests were carried out near completion of development on individual components for both server-side and client-side. Jasmine and Karma was the framework used to test the functionality of web components.

Python's Unittest was used to test server-sided functions ensuring that both HTTP Requests and the Matchmaking algorithm operated as expected.

End to end testing (e2e) was used to test the interactions and relationships between the backend and the presentation layer of the application. E2e testing was a great way to ensure that the components of the application worked together cohesively and also the application functioned correctly at a high-level overview. I concluded that unit tests were not sufficient enough, as unit tests only tested isolated elements of my project. I needed to test how the application's components operated as a combination. E2e testing was the best way to accomplish this. Test cases were generated by scenarios in the following ways: [4]

- (1) Identify the input data that meet the conditions associated with the component based on different testing techniques (e.g. unit tests).
- (2) Determine the expected results from input data.

The main way I generated test cases was based on application usage, e.g., one component can be affected by several conditions, and each condition can be satisfied by multiple data. For example, the registration element may have input data such as username, summoner name and password. Therefore, the conditions for this test case include

- 1) Valid username;
- 2) Valid summoner name;
- 3) Valid password;

The first test case satisfied these inputs and then the second test case took the exact input from the first scenario proving that duplicate usernames cannot be inserted into the database.

3.3 Source Control

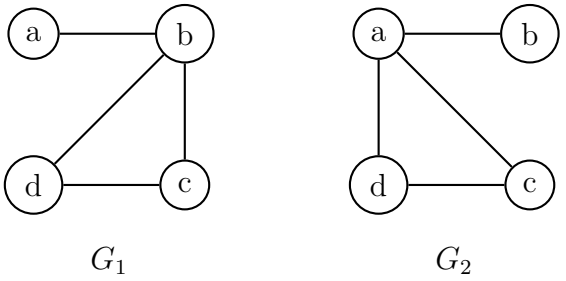
GitHub was used for source control and project management. Initially, I was using Trello for task management but this quickly became complicated to associate updates with unfinished tasks of the project. Therefore I changed the projects task management to GitHub's Issues section. I posted issues for any viable element that needed to be implemented into the project and when one of these elements were complete I would close the corresponding issue on GitHub. Each issue was categorized with tags depending on the type. These tags include:

- To-do: Tasks that have yet to be implemented.
- Tests: Types of tests that have been or need to be carried out.
- Bugs: Issues or bugs that occurred throughout the project and how they were solved.
- In progress: In progress are tasks that are currently being implemented.
- Completed: Finished tasks.
- Enhancement: When a completed part of the project has been upgraded, changed or removed.

This method of task management proved to be a lot more manageable compared to my previous method of using Trello. I could easily compare my current tasks to my commits on GitHub. Anytime I had implemented a significant change or addition to my project, I would perform commit it to through git and push the change.

3.4 Technologies Selection Criteria

The primary development environment used throughout the project was Visual Studio Code, The main reasons why I chose this environment is because it supports debugging, Git control, syntax highlighting, intelligent code completion and for its customisability. Both Frontend and backend of the project were developed in this environment. I used Angular which is an open-source web application framework led by the Angular Team at Google. The reasoning behind choosing Angular is because I wanted the project to be a web application as compared to a hybrid application, Angular seemed to be the most viable framework for this application. When researching options for the



G_1 G_2

Figure 3.1: Nice pictures

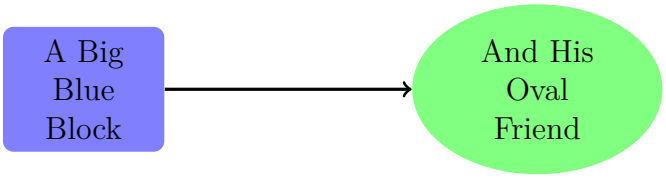


Figure 3.2: Nice pictures

database server, the two main options were python and flask or using MEAN stack (Mongo, Express.js, Angular and Node). I wanted the database to be a relational instead of using a NoSQL database, so a NoSQL database did not suit. I also wanted to use a technology that I'm not familiar with. These factors

- Agile / incremental and iterative approach to development. Planning, meetings.
- What about validation and testing? Junit or some other framework.
- If team based, did you use GitHub during the development process.
- Selection criteria for algorithms, languages, platforms and technologies.

Check out the nice graphs in Figure 3.2, and the nice diagram in Figure **??**.

Chapter 4

Technology Review

About seven to ten pages.

- Describe each of the technologies you used at a conceptual level. Standards, Database Model (e.g. MongoDB, CouchDB), XMl, WSDL, JSON, JAXP.
- Use references (IEEE format, e.g. [1]), Books, Papers, URLs (timestamp) – sources should be authoritative.

4.1 Angular

Angular is an open-source web application framework led by the Angular Team at Google. It is often used for building Single Page Applications (SPA). What is a Single Page Application? In a web application, when you navigate to a different page, the entire page is reloaded, in a SPA, only the view of the content requested is reloaded. SPA provides a fluid experience for the user. A good example of a Single Page Application is Twitter. Since this application is a SPA, navigating between pages was smooth. A constant array of Routes is declared for every component.

```
const routes: Routes = [
{ path: 'mypath', component: MyComponent}
{ path: 'mypath2', component: MyComponentTwo}
{ path: 'mypath3', component: MyComponentThree}
];
```

4.1.1 Why Angular?

- Components

Angular allows you to create components that provide functionality, styling and views.

- Dynamic Routes

TODO:

- Data Binding

Allows accessing of data from Typescript code to the html page view. This eliminates the process of implementing data binding myself. Example:

```
// TypeScript String Variable
myString: string = "Hello, World";

// Data Binding on the HTML Page
{{myString}}
```

- Testing

Angular includes testing frameworks (Jasmine, Karma and Protractor) for e2e testing and unit testing. When creating a new component, A template spec file is also created where test cases for each component can be easily written.

4.2 SQLite

SQLite is an open-source relational database. I used SQLite in the development of the project so I could audition how data was structured for the entire application. Each table went through iterations of changes until I was satisfied with the database schema. SQLite database is stored as a file locally [5] instead of running as a stand-alone process. This made it easier to develop a prototype database and understand how data will be interpreted when deploying. When I finally developed a functioning database I converted to a PostgreSQL production database. This was a smooth transition as both databases were relational. This meant queries didn't change and only how the database connected to the server and had to be changed.

- Connection to SQLite Database:

```
db_connect = create_engine('sqlite:///dev_database.db')
```

- Connection to PostgreSQL Database:

```
connection = psycopg2.connect(user=user, password=db_password, host=host, port=p

cursor = connection.cursor(cursor_factory=RealDictCursor)
```

4.3 PostgreSQL

PostgreSQL (Postgres) is a Relational Database Management System (RDBMS). [6] Postgres is known for its reliability, data integrity and extensibility. The main reason why I chose Postgres as my production database is because of its extensibility, ensuring my application is scalable for future growth. Postgres also provides concurrency meaning queries can be read in parallel allowing multiple users to use the database at the same time.

Tables: Matches and Participants both contained a match id primary key, I could access match data from both tables using a match id number. These tables were used in match creation and joining.

match id	match type	match name	date	outcome	admin
Row 1.2	Row 1.2	Row 1.3	Row 1.4	Row 1.5	Row 1.6

Table 4.1: Matches table.

match id	username	summonername
Row 1.2	Row 1.2	Row 1.3

Table 4.2: Participants table.

4.4 PgAdmin

PGADMIN TODO

4.5 Heroku

Here's some nicely formatted XML:

4.6 Firebase

Here's some nicely formatted XML:

4.7 Flask

Here's some nicely formatted XML:

4.8 RiotWatcher

Here's some nicely formatted XML:

4.9 Postman

Here's some nicely formatted XML:

4.10 JSON

Here's some nicely formatted XML:

4.11 Karma and Jasmine

Here's some nicely formatted XML:

4.12 Python Unittest

Here's some nicely formatted XML:

4.13 The Glicko System

Here's some nicely formatted XML:

4.14 Gale Shapley Algorithm

Here's some nicely formatted XML:

4.15 Bootstrap and Angular Material

Here's some nicely formatted XML:

4.16 Bcrypt

Here's some nicely formatted XML:

Chapter 5

System Design

As many pages as needed.

- Architecture, UML etc. An overview of the different components of the system. Diagrams etc... Screen shots etc.

Column 1	Column 2
Rows 2.1	Row 2.2

Table 5.1: A table.

Chapter 6

System Evaluation

As many pages as needed.

- Prove that your software is robust. How? Testing etc.
- Use performance benchmarks (space and time) if algorithmic.
- Measure the outcomes / outputs of your system / software against the objectives from the Introduction.
- Highlight any limitations or opportuni-ties in your approach or technologies used.

Chapter 7

Conclusion

About three pages.

- Briefly summarise your context and ob-jectives (a few lines).
- Highlight your findings from the evalua-tion section / chapter and any opportuni-ties identified.

Bibliography

[1] M. Sjöblom, J. Hamari, H. Jylhä, J. Macey, and M. Törhönen, "Esports: Final report," *Tampere University*, 2019.

[2] Y. Seo and S.-U. Jung, "Beyond solitary play in computer games: The social practices of esports," *Journal of Consumer Culture*, vol. 16, no. 3, pp. 635–655, 2016.

[3] Y. Kou, X. Gui, and Y. M. Kow, "Ranking practices and distinction in league of legends," in *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, pp. 4–9, 2016.

[4] X. Bai, W.-T. Tsai, R. Paul, T. Shen, and B. Li, "Distributed end-to-end testing management," in *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, pp. 140–151, IEEE, 2001.

[5] C. Newman, *SQLite (Developer's Library)*. Sams, 2004.

[6] B. PostgreSQL, "Postgresql," *Web resource: http://www. PostgreSQL. org/about*, 1996.