League of React

COMP 4983 X1 Project Report

Jordan Pippy

100136734

Jordan Pippy

100136734

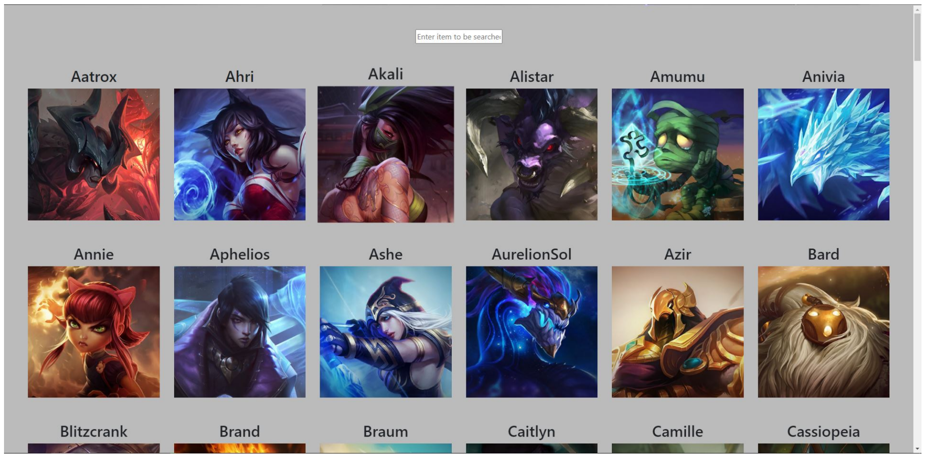
Contents

1 Overview

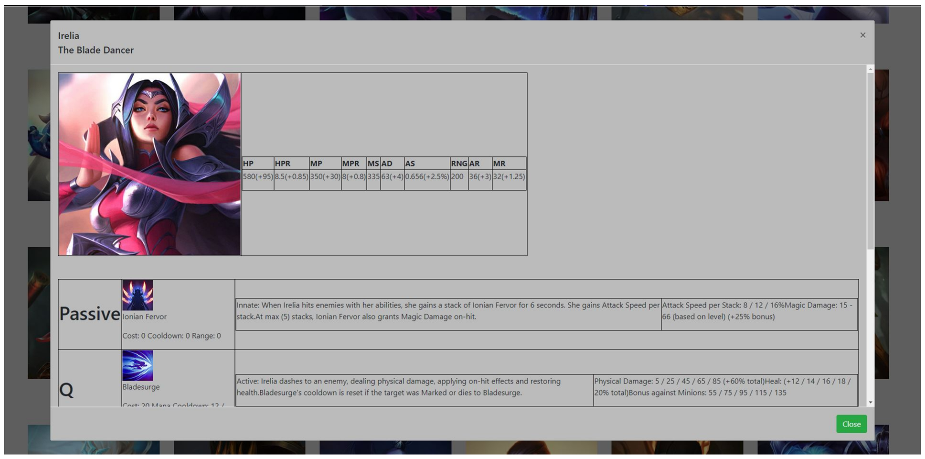
In my project proposal, I said I wanted to set out to make a web application for the game: League of Legends. The project was supposed to be an online character database that players could reference while playing League of Legends. I wanted it to primarily help new players learn the game in a way that is less overwhelming than it currently is. Seeing the huge character pool of 151+ characters can scare away many players. I also wanted it be usable for experienced players who just need to know more detailed information about the characters rather than just the basics. I tried to lay out all of the data in a way that would not be intimidating and has an intuitive design. The searching feature really helps with that as you only need to sort-of be able to spell the character's name rather than recognize them from a picture

I wanted League of React to be efficient, easy to use, useful, and visually appealing. I think I succeeded in all of those exept for the last one. While my project is not ugly or an eye-sore, it is not pretty. But all things considered, I'm okay with that. I was learning a variety of new technologies and encountered a slew of difficulties even in the technologies I already understood. All in all, I learned a lot over the course of this project. I had never delved too deep into web development before, and while it still may not be my forté, I thoroughly enjoyed it and am excited to continue with this project and start many others in the web dev world.

Below is a picture of the homepage, I wanted to throw in what the actual application looks like. This will come up on startup as soon as all the databases queries finish excecuting.



Below is a picture of the modal. Whenever a character is clicked on, a modal with all of that character's information is generated and shown to the user.


Irelia
The Blade Dancer

HP	HPR	MP	MPR	MS	AD	AS	RNG	AR	MR
580(+95)	8.5(+0.85)	350(+30)	8(+0.8)	335	63(+4)	0.656(+2.5%)	200	36(+3)	32(+1.25)

Passive	Ionian Fervor Cost: 0 Cooldown: 0 Range: 0	Innate: When Irelia hits enemies with her abilities, she gains a stack of Ionian Fervor for 6 seconds. She gains Attack Speed per stack.At max (5) stacks, Ionian Fervor also grants Magic Damage on-hit.	Attack Speed per Stack: 8 / 12 / 16%Magic Damage: 15 - 66 (based on level) (+25% bonus)
Q	Bladesurge Cost: 20 Mana Cooldown: 12 /	Active: Irelia dashes to an enemy, dealing physical damage, applying on-hit effects and restoring health.Bladesurge's cooldown is reset if the target was Marked or dies to Bladesurge.	Physical Damage: 5 / 25 / 45 / 65 / 85 (+60% total)Heal: (+12 / 14 / 16 / 18 / 20% total)Bonus against Minions: 55 / 75 / 95 / 115 / 135

2 Github / Installation

2.1 Databases

On github there is a readme that goes through most of the installation, but I will also talk about that here.
The github repo has two branches, 'main' and 'deploy'. The deploy branch is for the sole purpose of deploying the app on Heroku, it is set up how they want it and it works for them. The main branch is the actual project, that is what I would like to be focused on for marking. Both branches have roughly the same code. There are the differences of file structure and main using MySQL while deploy uses PSQL. There are no instructions for setting up the PSQl database, as it was more of a hack getting it to work with Heroku.

2.2 Installation

The main thing with the database is that it looks for a user named 'root' with a password 'mypw'. Both of these can be changed in the 'testproject-server/index.js' file. Other than that specific instance, I don't think it really matters what the credentials are. I included a file 'dump.sql', it is an export of the database from MySQL Workbench. I tried to import it using the same tool with success. I had no problems at all with it, so hopefully that keeps up. Getting that database to work will hopefully be the hard part.

2.3 Front-End

Next we need the front-end and back-end node_modules.
front-end:
'cd ./testprojectweb'
'npm install'
'npm run build' (this will take a little while)

Thats it for the front-end.
'cd ..'

2.4 Back-End

back-end:
'cd ./testprojectserver'
'npm install'
'node index.js'
Browse to localhost:3000.

2.5 Scripts

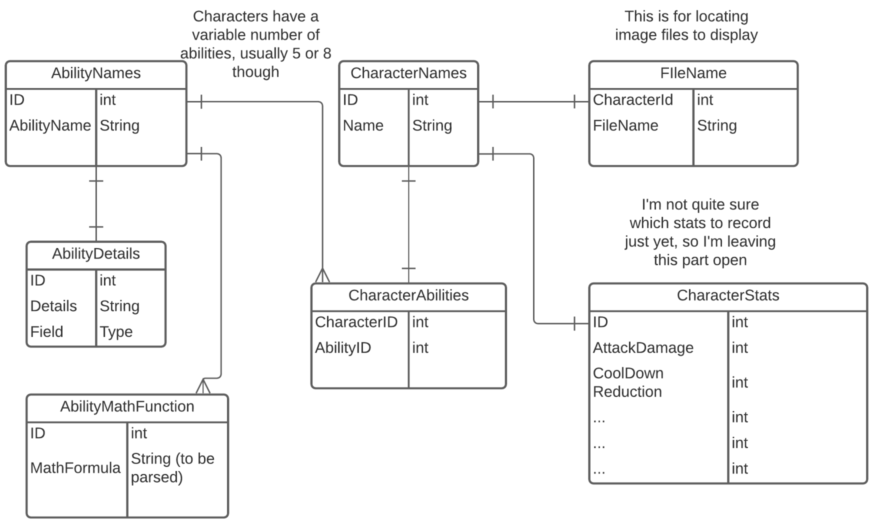
There is also a folder called 'scripts'. This is not important for running the app. They are scripts I used to turn the raw HTML into useable data. I wanted to include them to show that parsing data took an unforeseen portion of my time. There was nearly 1000 lines of python code to write in order to get all the data I needed. I included as much of the code as I could find.

3 Databases

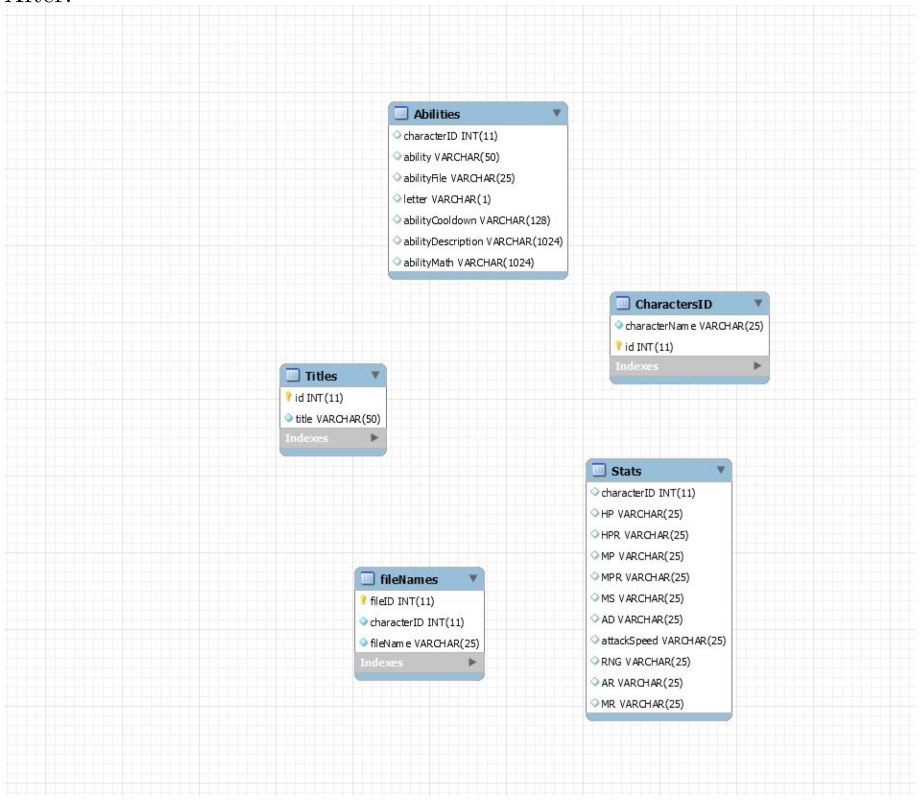
3.1 Comparing Schemas

In my project proposal, I listed a database schema (subject to change). Now, I want to compare what I thought the database would look like, and what it actually looks like in the end.

Before:



After:



The 'After' image was generated by MySQL Workbench, so it is pretty accurate to what the database actually looks like. Interestingly enough, none of the tables are connected to each other. This was a bit of an oversight seeing it now. The back-end server has the SQL statements that link up the data from multiple tables, instead of the database doing that itself. It is far too late to change this, as right now, my project is working. Getting a couple of lines on a diagram does not seem worth the risk of any more late nights with the project. I think the schema did stay fairly true to the proposed schema.

The main differences are:

- The new one is not quite as normalized as the original one.
- Each ability does not have its own id, this was because it would be redundant to do. Abilities are not used in any other table (just like everything else isn't... oops..).
- All ability information was put into a single table, this is for much the same reason as the former. Other tables didn't need to reference the ability table, so why should it?

3.2 Deployment

For this project, I used MySQL as the database. I said that in the proposal and it was true, right up until I deployed the website. I tried to use MySQL as the database for deployment, and it worked for 4-5 days. Then it just died. So for the deployment branch of this project, I used PSQL. PSQL is not much different, other than the tools for it don't seem to be quite as good as MySQL Workbench. I used HeidiSQL for the PSQL database. The nice thing about that is it supports lots of database languages, whereas MySQL Workbench does not.

4 Data Gathering

4.1 Overview

More of this project than I anticipated was dedicated to data gathering. When I took on this project I had an API that claimed to have all of the information I would need. The API came from Riot Games, the creators of League of Legends. When I looked closer at the data, it was outdated, incorrect, or just incoherent. I looked for another API and ran into the same issue, then another. I had three API's fail on me before I found the one I had success with.

4.2 The Winning API

The API I chose in the end was just raw HTML for all of characters, which at the time was 151 of them. Each HTML file was between 1200-2000 lines, there was a lot of parsing to do. This is where the 'scripts' directory on github comes into play. I had to write most of those before I could even start the front-end or the back-end of the project. This ate up large amounts of my time, far more than I budgeted for. I eventually got all of the data parsed out into plain text format, then I needed more scripts to get the data into the database. Once all of this was done, it was time to start the actual project.

5 The Project

5.1 Front-End

For the front-end of the project I decided to use React.js, I ended up liking react quite a bit. I have never used a framework before, and thought that would be a nice place to start. I liked the feel of coding JSX elements rather than raw HTML. I don't think I did a great job of seperating my components, but hey, it was my first try. The hardest part for me was understanding the data flow. I didn't realize until late into the project how the data could move up and down the inheritance tree. Once I got that enlightenment however, the code became far more efficient. I separated my React.js classes into 3 components.

- 'Home' Which handles the inital loading of the characters and renders them relative to the search bar.
- 'HomepageHandler' Which handles sorting the characters, onClick functionality, and data routing.
- 'IconModal' Which when clicked, loads a modal displaying information about the character. This makes several queries to the database.

5.2 Back-End

For the back-end of the project I decided to use Node.js. I have only ever used PHP for a server-side scripting language and wanted to try something new there as well. I had some issues understanding how requests and responses work, as well as how to handle a GET request. Now it serves files like never before. I really like Node.js as a server-side language. Node.js feels far more fluid than PHP and has lots of support and modules that are incredibly easy to use and install using npm/npx. I will definetly continue to use Node.js as a back-end language from here on out.

5.3 Front-End, Meet Back-End

I had significant difficulty with getting the back-end to serve the front-end files. To this day im not totally sure what actually fixed the issue for me. All I did was remake the project and re-design the file structure to separate the server directory from the front-end directory. It could have also been expressing the front-end files statically within the Node.js server.

6 Conclusion

In conclusion, I'm very happy with how the project turned out. I feel like I met most of the goals I set for it and only missed some of the smaller ones. The big goal was to make an easily accessible character database. I feel that League of React can definitely help new players learn League of Legends. I also think that I have created a resource that veteren players could use. Some of the smaller goals I missed are:

- Multiple tools to help get the desired information quickly
- More appealing front-end design

I definetly want to keep working on this project in the future, because I see real potential in it. I think it has a chance to compete with the other products out there that serve similar purposes.