Database Project

Michael Kim (mkim225) [315] (mkim225@jhu.edu)

Ehong Sun (esun7) [415] ([esun7@jhu.edu](mailto:esun7@jhu.edu))

Backend

(Refer to the code comments above the api call. Type in the comment url after the website url)

<https://tft-database-backend.vercel.app/>

(The base url will say CANNOT GET/. Only use the api links to find the queries)

Frontend

<https://tftfrontenddatabase-7begs1zjj-kimpeppys-projects.vercel.app/>

Our database is storage of certain aspects of the game TeamFight Tactics(TFT). It stores champions, augments, and matches for set 10 of TFT.

Description of TFT:

TFT is a video game that involves a player creating a team of champions to fight other players’ team. Each player must create the best team and outlast other teams. They are placed based on how long they last in the game.

https://en.wikipedia.org/wiki/Teamfight\_Tactics

Since TFT is constantly being updated and is a complex game with a lot of information, we wanted to include all the information of TFT into one dataset. However, the complexity of the game is too much to gather as the game is constantly getting changed. This leads us to only work on the most recent version of the game database. Thus in order to make our database sufficiently complex, we added in the ability for players to find and filter their matches. They can filter matches either by the place they got for the match (1st, 2nd, 3rd, etc) or the champion they used in said match. We inputted our champions manually, but for our matches, we generated them from Riot’s (the company that developed TFT) APIs and then parsed them. Due to access to said APIs being two days, we only got 8 matches. We also used open source json files (dragon.json) for items (equipables within the game). We found parsing the match files particularly hard due to how the json file was formatted and the natural volatility of the data. For example, the two main sections of the json file were “metadata” and “info”. Within info, you then had “participants” which was further split into a multitude of fields including but not limited to “units”, “augments”, “traits”, etc. The fields I listed, among others, were all 2D arrays containing extra data for their elements. This was further complicated with the fields “unit” since each player would have a varying size of the “unit” array. Thus, we would create an array of fixed size, populate it with a player’s units, and fill the rest with blank units when inserting it into our database. We would have to do something similar with our API points since user input of our code would be varying as well.

Our code works through two main tabs, champions and matches. Each of these tabs will display all champion and matches (naturally only matches that include the player). You can then filter both of these tabs based on different categories. For champions, you can filter by cost, class, and origin. As mentioned before, matches are filter either through place and/or champion.

Our major area of specialization was an advanced GUI. We wanted to make it look like a tft database you might find online and wanted to specifically replicate the function and experience of these two websites (<https://tftactics.gg/champions> and <https://tactics.tools/>) or minor. Our minor area of specialization was our data. In addition to using Riot’s API, we also generated data for each participant by making their names.

Our website uses node.js to run sql queries in the mySQL database.

Challenges:

* Getting the info on champions and items
  + Since tft is relatively new, there were not enough sources to get the game information.
  + We end up having to use an open-source file called dragon.json, which include all information relative to the game itself (each item is either an object or a champion).A screenshot of a computer

    Description automatically generated
  + Since the dragon.json file was extremely long and complicated, we must create scripts to extract information from the database.A computer screen shot of a program code

    Description automatically generated
* Getting matches
  + For some reason, Riot is extremely strict on the use of their API, only allowing people to use it for two days before they have to renew their api key.
  + This prevent us for making this database dynamic as we need to approve our project to use the api key forever.
    - This is a month process, which at the time when we started this project, we have three weeks.A screenshot of a phone

      Description automatically generated
  + We ended up using one of the team member’s account to extract about 8 matches from their history using the Riot API.
  + The database only consist of matches that Michael has played, here is a match.json that was take from one of Michael’s game.A screen shot of a computer code

    Description automatically generated
  + For each match, there are about 8 players, so we have to generate about 56 new participants in the participant database(1 is Michael and the other are random people).
  + We gave every participant random names since we couldn’t find everyone’s real game id.

Info on the backend:

* Server.js
  + Where our api points call the SQL procedures
  + Run node server.js to run the server assuming the database is connected
    - There is a good chance the sql credentials in the submission is not correct as it is currently connected locally. However, there should be an URL that has the working project
* sqlQueries
  + Where our queries that run the database is stored
* Any Extract javascript files
  + Mostly helper files used to get the files that are stored in the json folder
* Json folder
  + Data taken from dragon.json that includes items or champions

Instructions on the website

* Champions Database
  + Use the checkmarks to filter out the champions
  + Type participant name to show what matches they appear in
* Matches Database
  + Use the player text to find what players played in what match
  + Use the place drop box to find what place they are in

Instructions on the backend

* In the url, type the api/[command] to test out api calls you are curious in
  + For example
    - https://tft-database-backend.vercel.app/api/select/champions

Project Strengths

* A natural, intuitive UI
* Easy, desirable filter options for quick champion and match look up
* In depth information about matches and champions that allow users to efficiently analyze their matches

Project limitations

* Since we only have one set, user cannot look at stats of previous sets
* We wanted to make a team builder (code that would allow users to build potential teams before entering a match)
* Trending champions or composition of champions or win rates for certain compositions that the user has used.
* Riot API Key are valid for only two days so this project cannot get real data from any players. In this case, we are only using data that are from one player who played 8 different matches.

Here is a following list of sql schema

* Abilities.sql
* Augments.sql
* Champions.sql
* Game.sql
* Items.sql
* Participant.sql
* Teamcomps.sql
* Unit.sql

Rest of the queries are sql procedures.