Members: Chang-Syuan Wu, Andrew Chan, Jenish Patel, Parathan Vakeesan

Description of Application

Task 1

The project is an application that allows users to compare win rates and play rates for champions in League of Legends across different periods of time. The data management application being used is MySQL, which is currently being hosted locally on the user's computer. However, the database should be optimally hosted remotely, which will be considered sometime in the future. The backend will be implemented by the node.js framework. Data being stored in the database includes a comprehensive list of all champions in the game, their play statistics, and match histories that include a list of champions that each time picked during their games. To start things off, a large match history and champion list compiled and imported from Kaggle needs to be imported into the local database.

From there, the applications will implement several features:

- Get base statistics for a single champion
    - Example: find statistics related to Fiora
    - ```
      SELECT * FROM champions WHERE LOWER(champion_name) = "Fiora";
      ```
- Calculate champion play rate and win rate
    - Example: Get number of matches won by Morgana as well as number of games played to calculate Morgana's win rate
    - ```
      SELECT COUNT(*) AS count FROM matches WHERE ("Morgana" IN (LOWER(blue_top), LOWER(blue_jungle), LOWER(blue_mid), LOWER(blue_adc), LOWER(blue_support)) AND result = "Blue") OR ('Morgana' IN (LOWER(red_top), LOWER(red_jungle), LOWER(red_mid), LOWER(red_adc), LOWER(red_support)) AND result = "Red");
      ```
    - ```
      SELECT COUNT(*) AS count FROM matches WHERE ("Morgana" IN (LOWER(blue_top), LOWER(blue_jungle), LOWER(blue_mid), LOWER(blue_adc), LOWER(blue_support))) OR ('Morgana' IN (LOWER(red_top), LOWER(red_jungle), LOWER(red_mid), LOWER(red_adc), LOWER(red_support)));
      ```
- Get list of matches where a champion was played
    - Example: find all matches where Garen was played and won
    - ```
      SELECT * FROM matches WHERE ("Garen" IN (LOWER(red_top), LOWER(blue_jungle), LOWER(blue_mid), LOWER(blue_adc), LOWER(blue_support)) AND result = 'Blue') OR ("Garen" IN (LOWER(red_top), LOWER(red_jungle), LOWER(red_mid), LOWER(red_adc), LOWER(red_support)) AND result = 'Red');
      ```
- Add new matches into the application

    - Example: add a match with Red team: [Lee Sin, Rumble, Sett, Varus, Lulu] vs Blue team: [Viego, Xin Zhao, Lucian, Ezreal, Leona], Red side win
    - ```
      INSERT INTO matches (blue_top, blue_jungle, blue_mid, blue_adc, blue_support, red_top, red_jungle, red_mid, red_adc, red_support, result) VALUES ('Lee Sin', 'Rumble', 'Sett', 'Varus', 'Lulu', 'Viego', 'Xin Zhao', 'Lucian', 'Ezreal', 'Leona', 'Red');
      ```
- Update champion statistics
    - Example: change Gnar's base armour to 35
    - ```
      UPDATE champions SET armour = 35 WHERE champion_name = "Gnar";
      ```
- Update match history
    - Example: change red jungler to Udyr for match with match ID 2
    - ```
      UPDATE matches SET red_jungle = "Udyr" WHERE match_id = 2;
      ```

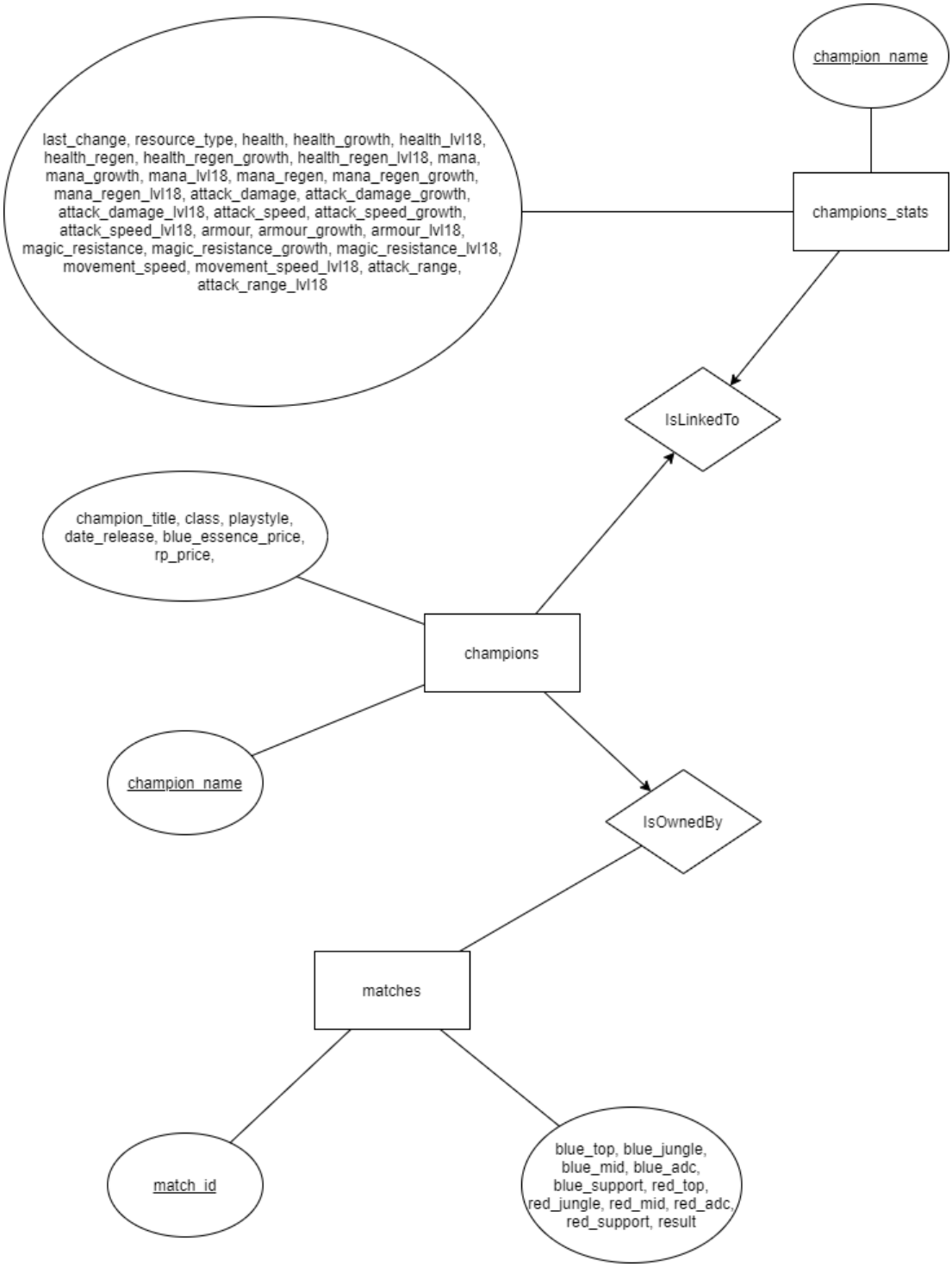
From there, the application will query the database for champion statistics for a specific champion, all of the matches where the champion was played, and filter displayed matches based on the position that the champion played and if the chosen team won or lost.

Users will be allowed to add new match data into the database by providing the champion roster for each team and designated which team won. Champion statistics and match data can be updated by the user to account for adjustments from game patches and possible mistakes in the match data respectively.

Task 2

In the database schema, champion_name must be unique within the champions table and champions_stats and match_id must be unique for the matches table (they are the primary keys of those tables).

Additionally, champions are unique for each team in each match; there cannot be duplicates of the same champion on the same team but the same champion can be on both the red and blue teams.
      ```



- `champions(champion_name, champion_name, champion_title, class, playstyle, date_release, blue_essence_price, rp_price)`
-

```
champion_stats(champion_name, last_change, resource_type,
health, health_growth, health_lvl18, health_regen,
health_regen_growth, health_regen_lvl18, mana, mana_growth,
mana_lvl18, mana_regen, mana_regen_growth, mana_regen_lvl18,
attack_damage, attack_damage_growth, attack_damage_lvl18,
attack_speed, attack_speed_growth, attack_speed_lvl18, armour,
armour_growth, armour_lvl18, magic_resistance,
magic_resistance_growth, magic_resistance_lvl18,
movement_speed, movement_speed_lvl18, attack_range,
attack_range_lvl18)

matches(match_id, blue_top, blue_jungle, blue_mid, blue_adc,
blue_support, red_top, red_jungle, red_mid, red_adc,
red_support, result)
```

Task 3

We have extracted a dataset of 5600+ matches and a dataset of champion stats (All champions currently in League of Legends) from Kaggle and converted them into sql scripts to insert into our database schema.
The original data came in .csv format and we cleaned them up in Excel before using a script to generate sql insert statements to populate our database.
A current user of the application only needs to run the sql scripts to populate their database.

Task 4

We have an api that can be queried using REST endpoints (can be tested in a web browser, will return a JSON). The final user interface will also be web-based but made with React.

Task 5

See test-sample.sql and test-sample.out

Task 6

We are using JavaScript for both the frontend and backend. The frontend uses React while the backend uses Express and Node.js.
The backend interfaces with a local MySQL database using the mysql2 package while the frontend queries the backend using REST endpoints.

Task 7

We have extracted a dataset of 5600+ matches and a dataset of champion stats (All champions currently in League of Legends) from Kaggle and converted them into sql scripts to insert into our database schema

Task 8

Task 9

Task 10

Task 11

- Comparing winrates to playrates in League of Legends across different periods (will need to determine specific ways of dividing periods and how Kaggle supports this data related to time)
- Production data would be taken from Kaggle to be inserted into database
- Users can add their own match histories to the data and would affect the overall stats shown
- The end user of such an application are League of Legend players who are interested in the analytics related to experience and its correlation to winning

Description of Platform

- Users will have to run the program locally for now but we may try to get it hosted on something?
- This will be a web application built using React in the frontend, Node in the backend and using mysql as the database (preliminary)

Member Contributions

- Chan, Andrew
    - Responsible for developing the application's backend
    - Helped clean imported datasets
    - Implemented match history viewing, filtering, and editing features
- Patel, Jenish
    - Responsible for developing the application's frontend
    - Initialized tables for viewing players data
    - Implemented API fetching data setup for match history
- Vakeesan, Parathan
    - Responsible for developing the application's frontend
    - Initialized dashboard and match history pages layout
    - Added routing between pages
    - Created navigation bar used as default for all pages
- Wu, Chang-Syuan
    - Responsible for developing the application's backend
    - Found, cleaned, and imported datasets
    - Implemented base code for frontend client and backend API
    - Implemented champion base statistics and play statistics viewing feature