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 personal win rates and play rates for champions in League of Legends across different periods of time (patches). This application is designed to be a companion application to League of Legends players. Because League of Legends is a large and complex game, players would benefit from collecting and interpreting each champion's performance in the games that they have played. Each user of the application would be able to add their own matches, and the application would be able to calculate the personal win rate and play rate of every champion that the user has played. This would provide valuable insight to the user about the strongest champions to pick for future games, and which champions appear weak in the current patch. The application will also be able to provide basic statistics for each champion, like starting armour, starting health, and starting mana, to aid in theorycrafting and comparisons.

The data management application being used is MySQL, which is currently being hosted locally on the user's computer. The backend is implemented using the Node.js framework. Data being stored in the database includes a comprehensive list of all champions in the game, their base statistics, and match histories that include a list of champions that were played during their games. In addition, user login information is also stored, with the passwords being protected through the bcrypt algorithm.

Firstly, a comprehensive list of champions and their associated facts and statistics was imported from Kaggle, then refactored and cleaned manually in Excel, before being converted into SQL insert statements.

Since the matches data relies on match histories unique to the user, this dataset would need to be populated by the user itself using the adding matches feature. For demo purposes, a large dataset containing nearly 5000 matches was imported from Kaggle, refactored and cleaned in Excel, converted into SQL insert statements for one user called SYSTEM.

From there, the application will implement several features:

- Get base statistics for a single champion. This will return information ranging from the champion's defined playstyle, its health at level 1 and level 18, its blue essence price to buy from the in-game shop, and other useful information.
    - Example: find statistics related to Fiora

    -
      ```
      SELECT * FROM champions WHERE LOWER(champion_name) = 'Fiora';
      ```
- Calculate champion play rate and win rate. Play rate is calculated by number of matches played with the champion divided by the total number of matches recorded by the user in the database. Win rate is calculated similarly, where the number of times the user played that champion that was on the winning team is used instead of simply the number of times the champion was played.
    - Example: Get number of matches won by Morgana as well as number of games played to calculate Morgana's win rate for SYSTEM:

```
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’) AND author = ‘SYSTEM’);
```

```
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))) AND author = ‘SYSTEM’);
```

- Get list of matches where a champion was played. This query can be filtered to limit searches for champions played in a certain position, or if the match was won while playing them.
  - Example: find all matches where Garen was played by SYSTEM 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’) AND
author = ‘SYSTEM’);
```

- Add new matches into the application. All fields are mandatory. Only the author of the match entry may view, edit, and delete the match. The user must be authenticated to perform this action.
  - 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, authored by SYSTEM

```
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. If there is an in-game patch that changes a champion’s base statistics, the user may update it themselves.
  - Example: change Gnar’s base armour to 35

```
UPDATE champions SET armour = 35 WHERE champion_name = ‘Gnar’;
```

- Update match history. If the user made a mistake while creating a new match, they can rectify the error. The user must be authenticated to perform this action.
  - Example: change red jungler to Udyr for match with match ID 2

```
UPDATE matches SET red_jungle = ‘Udyr’ WHERE match_id = 2;
```

- Delete a match. This removes the match from the database. The user must be authenticated and must own the match to perform this action.

- Example: delete a match with ID 2

```
DELETE FROM matches 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.

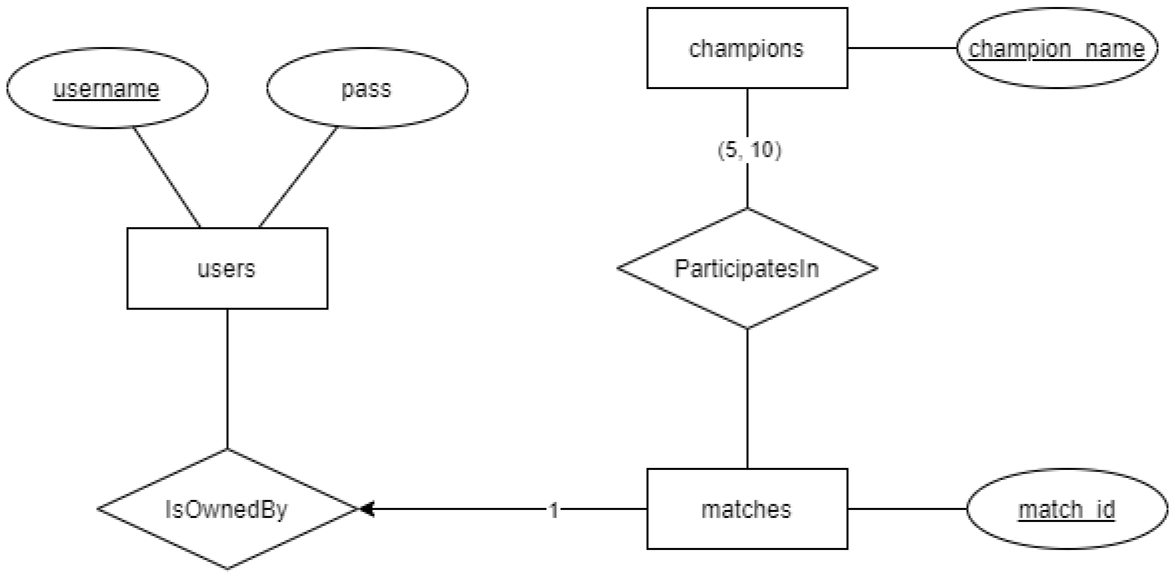
Users will be allowed to add new match data into the database by providing the champion roster for each team and designated which team won. Champion statistics and match data can be updated by the user to account for adjustments from game patches and possible mistakes in the match data respectively.

Task 2

In the database schema, champion_name must be unique within the champions table. 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. Hence, there can be between 5 and 10 distinct champions in one match.

In the users table, username must clearly be unique. Every match in matches is created by exactly one user.



The following describes all of the attributes for each entity/table:

```
champions(champion_name, champion_name, champion_title, class,
playstyle, date_release, blue_essence_price, rp_price,
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, patch, author)
● users(username, pass)
```

Task 3 (Plan for getting data)

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. We then cleaned them up in Excel by enforcing consistent spelling through the find and replace function before using a script to generate SQL insert statements to create sql scripts 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

No performance issues at this time. All queries return quickly (even with our “production” dataset of 5600+ matches. This is because the application only queries primary keys, and all primary keys are already indexed automatically in MySQL.

Task 9

Task 10

Task 11

- Comparing winrates to playrates in League of Legends across different periods. Each period is defined by a patch version, and is the responsibility of the user to add their data to the correct patch.
- Production data would be taken from Kaggle to be inserted into the database. This would be for demo purposes since the application relies on individual users adding their own data to produce insights relevant to them.
- 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

System support/Description of Platform

- Users will run the program locally on their computer
- Mac OS and Windows are the supported operating systems for now
- 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, editing, and deletion features
  - Improved match history endpoints by limiting actions to authenticated users
  - Refined matches table by adding a field for patches and match author
- Patel, Jenish
  - Responsible for developing the application's frontend
  - Initialized tables for viewing players data
  - Implemented API fetching data setup for match history
  - Added backend endpoints to match history table
  - Implemented Dashboard page which uses backend endpoints
- 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
  - Implemented Authentication services which used backend endpoints
  - Created Champions page which utilized champions/stats endpoint for individual champion stats
- 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 and editing features
- Implemented backend user authentication, register, change password, and session management features
- Modified champions statistics to only show statistics of the user currently logged in