Milestone 2 - Project Outline

GitHub repository : https://github.com/qihang-dai/SteamDB

1. Motivation for the idea/description of the problem the application solves

Steam is the leading PC platform for games. Over its many years online, Steam has amassed a plethora of reviews for its games. These reviews represent a great opportunity to break down the satisfaction and dissatisfaction factors around games and genres, as well as sentiment over time. We want to provide a platform that helps people see the reviews of the games on steam and choose the games suiting them.

2. List of features you will definitely implement in the application

1. User login and register. As a game store, we need to provide a user login and register function. Users can register an account and login to the website. We will store the user information in the database. This user table would also hold the information about games the user bought, how long they have played the game, and the review they left for the game.
2. Main Page for Browsing games: We will provide a main page that displays today's recommended games information. Users can browse the games and click on the game to see more details. It should randomly display 10 games out of the top 100 games with the highest rating and another 10 games that are similar to users recently played games.
3. Search Bar: We will also provide a search function for users to search for games. The search function will be implemented by using the LIKE operator in SQL. We will also provide a filter function for users to filter the games by tags, release date, and price. The filter function will be implemented by using GroupBy and having operators in SQL.
4. Review: We will provide a review function for users to review the games. Users can leave a review for the game and rate the game. The review will be stored in the database. We will also provide a function for users to see the reviews of the game, users see other users' reviews and ratings.
5. Game Details: We will provide a game details page for users to see the details of the game. The game details page will display the game name, developer, publisher, release date, price, tags, and description. We will also provide a function for users to buy the game. If the user has bought the game, the user can see the game in the library. If the user has not bought the game, the user can see the buy button. If the user clicks the buy button, the user will be redirected to the payment page.

3. List of features you might implement in the application, given enough time

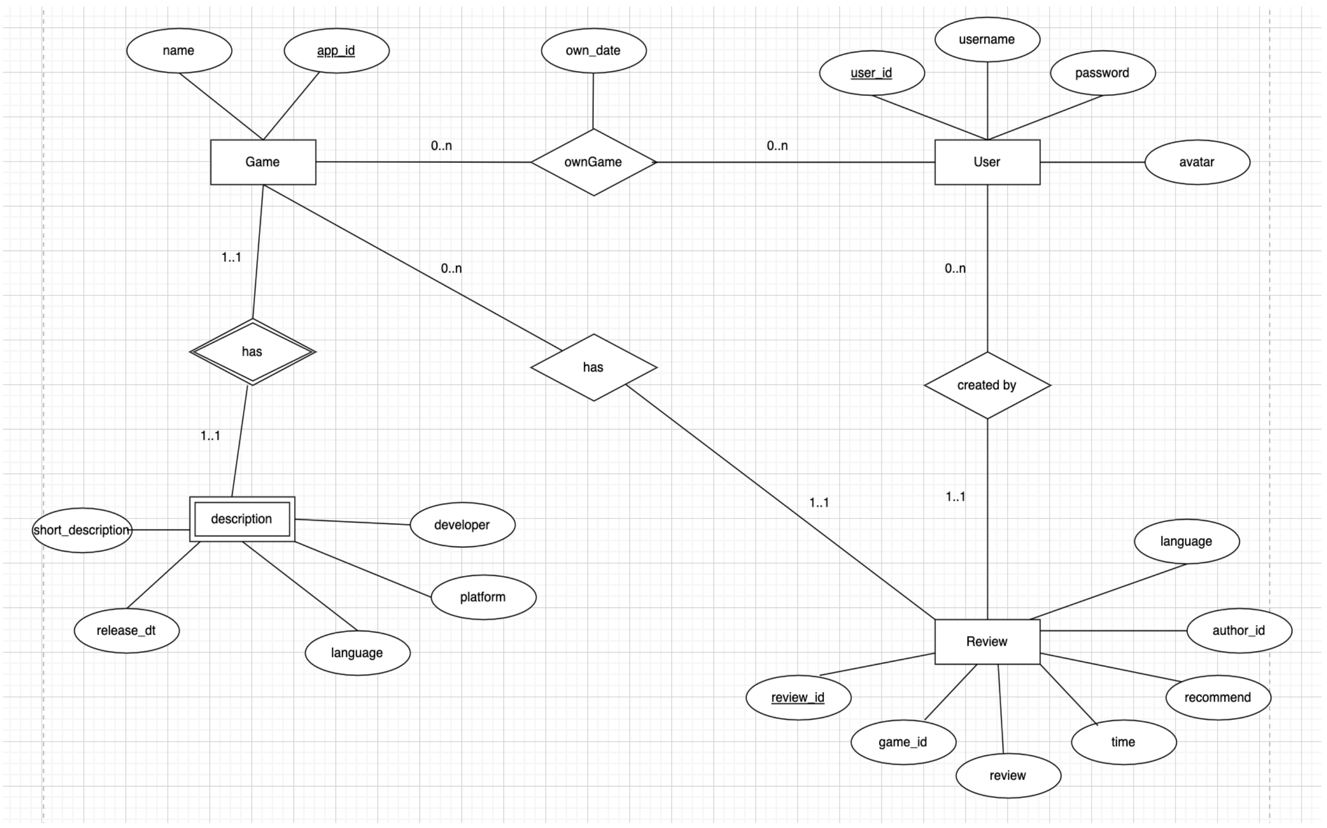
1. Like / Dislike: We will provide a like / dislike function for users to like or dislike the game. The like / dislike information will be stored in the database and may be used for

recommendation. If one user dislikes a game, the game will not be recommended to the user.
2. Friends: We will provide a friends function for users to add friends. Users can see their friends' library and review. Users can also see their friends' recently played games.

4. List of pages the application will have.

(1) Login page: User enters user_id and password to login. Then users will jump to the game page.
(2) Register page: User can create an account by entering user_id and password. Also, users can have a username which could be modified in the future.
(3) Personal page: This page includes information about user_id, username and games liked by the user. Users can choose whether this information is visible to other users.
(4) Game page: List top-rated games and recommend games to users based on the user’s interest. By clicking the name or icon of the game, users can jump to the game detail page.
(5) Game detail page: Display the game information, including developer, platform, release date of the game. Also,users can rate the game (and add reviews to the game).
(6) Friend page: Users can add friends on this page and view information of friends.

5. Relational schema as an ER diagram



6. SQL DDL for creating the database

```
CREATE TABLE Game(
        app_id int,
        name varchar(255) NOT NULL,
        PRIMARY KEY(app_id)
)

CREATE TABLE ownGame(
        app_id int,
        user_id int,
        owndate Datetime,
        PRIMARY KEY(app_id, user_id),
        Foreign Key(app_id) REFERENCES Game(app_id),
        Foreign Key(user_id) REFERENCES User(user_id)
)

CREATE TABLE User(
        user_id int,
        username varchar(40) NOT NULL,
        password varchar(100) NOT NULL,
        avatar varchar(255),
        PRIMARY KEY(user_id)
)

CREATE TABLE Review(
        review_id int,
        game_id int NOT NULL,
        author_id int NOT NULL,
        review varchar(255),
        time Datetime,
        recommend boolean,
        language varchar(40),
        PRIMARY KEY(review_id),
        FOREIGN KEY (game_id) REFERENCES Game(app_id),
        FOREIGN KEY(author_id) REFERENCES User(user_id)
)

CREATE TABLE Description(
        app_id int,
        short_description varchar(255),
        release_dt Datetime,
        language boolean,
        platform varchar(255),
        developer varchar(255),
        PRIMARY KEY(app_id),
        Foreign Key(app_id) REFERENCES Game(app_id)
```

7. Explanation of how you will clean and pre-process the data.

We will do data pre-processing base on below steps:
1. Use the data from the Kaggle data set, following:
https://www.kaggle.com/datasets/nikdavis/steam-store-games and
https://www.kaggle.com/datasets/najzeko/steam-reviews-2021
2. For necessary features like game id or user id, remove the rows with illegal values or missing values. For unnecessary features like game release date or game language, set the illegal values or missing values to a default value.
3. Utilize Python to split the Kaggle csv file into serval sub csv files with the structure exactly as the structure we designed in section 6.
4. Establish a relational database instance on AWS, and connect it with the backend server.
5. Create tables on our SQL DDL in section 6.
6. Upload the pre-processed csv file into target tables.
7. Make sure the primary keys in each table are unique, and foreign keys correctly corresponded.

8. List of technologies you will use.

database: mysql, mongoDB, AWS RDS (HOLD DATABASE), AWS EC2 (HOLD FRONT END)
web: react, node js, css expressjs python

9. Description of what each group member will be responsible for

1. front end : Jingxuan Bao & Qihang Dai
2. back end : Peihan Li & Jingru Wang
3. data preprocess : Jingxuan Bao
4. database design : Qihang Dai
5. search algorithm : Peihan Li
6. search query optimization : Jingru Wang
7. recommendation algorithm : Jingxuan Bao & Qihang Dai & Peihan Li & Jingru Wang