Semester-Long Project

# *Gaming Hub Website*

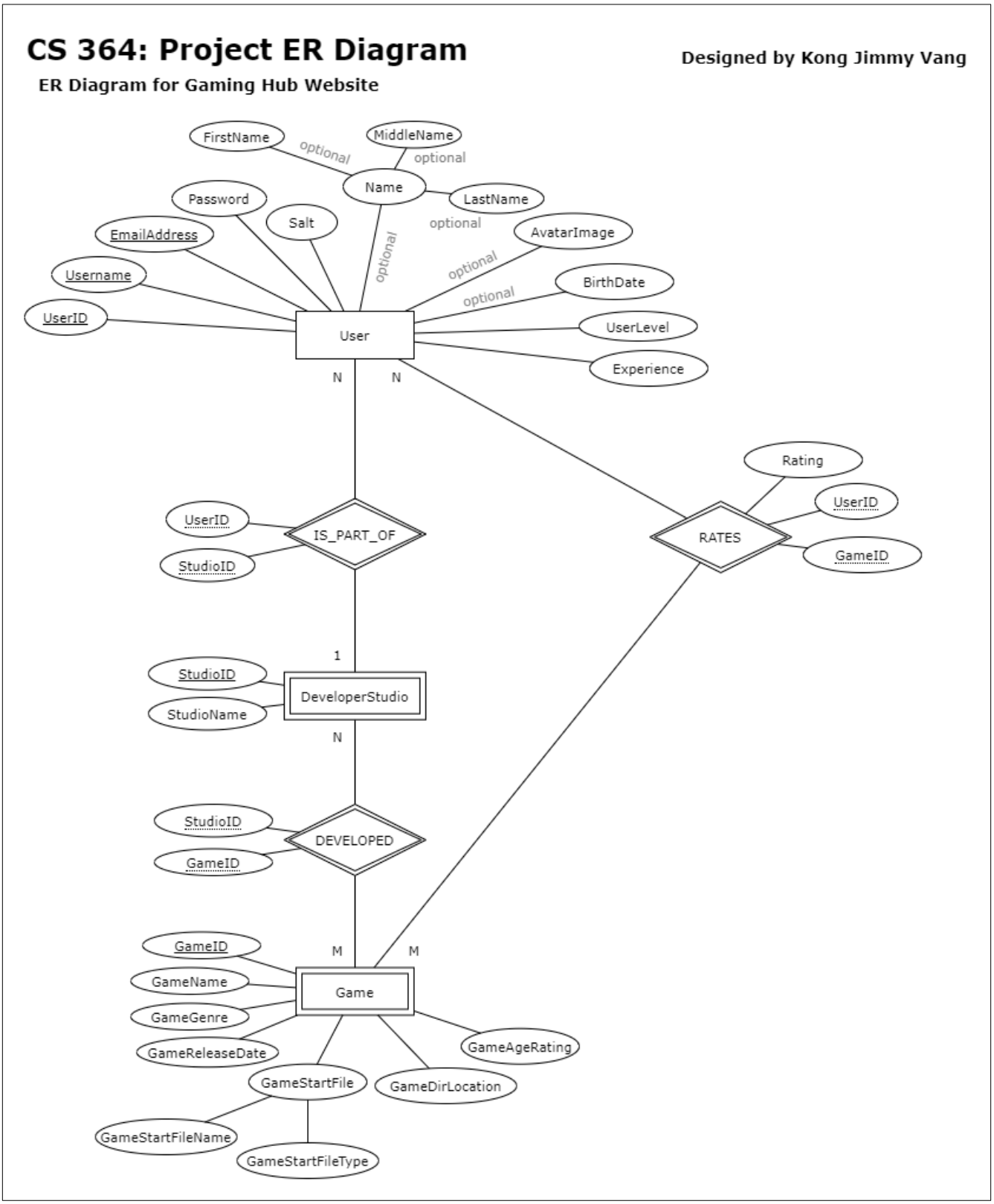
by Kong Jimmy Vang

## Synopsis

This project is based on a Free-To-Play (F2P) game hosting website named Kongregate (https://www.kongregate.com/). This website allows developers to create games and upload and host their games on the website for users to play. Developers are allowed to make money on their games through monetization, but their games are required to be free to play. In the past, the site allowed uploading of SWF files or Adobe Flash games. Nowadays, many browsers have deprecated the use of Flash. Most newer games on the site now use Unity WebGL, or other forms of integrated web application software. The site does allow for backwards compatibility with older games, but they require the use of Third Party software that are similar to Adobe Flash. The site allows searching for games to play with the use of an advanced search bar. The site shows top new games and the top rated games if the button is clicked on. At the bottom of each web application, comments are allowed. Comments can be upvoted or downvoted, they can also be replied to. Users are allowed to send messages to each other via PM or DM.

The site allows users to play without an account. They are also allowed to create an account, sign in, and play games. Users that sign in have information stored about each game they play. Users are allowed to chat with others during each game they play. Each game has rooms that players can socialize in during gameplay. Each game has it's own acheviement that the user can earn by playing the game. They are only earned if you own a user account. Overall, my project is based on a this type of design and attempt to store data in a database relevant to running a website similar to Kongregate.

# Database (ER Diagram)



## Tables/Attributes Details

- **User** - This table will store user information relevant to the F2P game hosting website. Users can send a Ticket, which Employees can review. Users can also be part of a Developer Studio and develope Games together.
  - **UserID** - (**PRIMARY**) Unique User ID.
  - **Username** - (**UNIQUE**) Unique Username.
  - **EmailAddress** - (**UNIQUE**) The Email Address of the User.
  - **Password** - Login password (Must be encrypted).
  - **Salt** - Cryptographic salt for password.
  - **Name** - (**OPTIONAL COMPOSITE**) Real name of the user. Optional for privacy reasons.
    - **FirstName** - (**OPTIONAL**) Real first name. Optional for privacy reasons.
    - **MiddleName** - (**OPTIONAL**) Real middle name. Optional for privacy reasons.
    - **LastName** - (**OPTIONAL**) Real last name. Optional for privacy reasons.
  - **AvatarImage** - (**OPTIONAL**) A path that points to a jpg, png, or gif file, which shows the user's avatar icon.
  - **BirthDate** - To verify if the user is 13+ years old.
  - **UserLevel** - The level of the user from earning experience points.
  - **Experience** - The experience points earned from playing games.
  - **StudioID** - (**FOREIGN**) Studio ID.
- **DeveloperStudio** - This table will store information about a developer studio team. The Developer Studio consists of multiple Users and can Develop multiple different games. Many studios can work on one game together also.
  - **StudioID** - (**PRIMARY**) Unique Studio ID.
  - **StudioName** - The Developer Studio's name.
  - **UserID** - (**FOREIGN**) User ID.
- **Game** - This table will store information about each game a Developer Studio has created. Each game can be rated by multiple users.
  - **GameID** - (**PRIMARY**) Unique Game ID.
  - **GameName** - The game's title/name.
  - **GameGenre** - The primary genre of the game.
  - **GameReleaseDate** - The date that the game was released on.
  - **GameStartFile** - (**COMPOSITE**) The game's start file.
    - **GameStartFileName** - The name of the start file.
    - **GameStartFileType** - The type (file extension) of the start file.
  - **GameDirLocation** - The folder directory path/location where the game is stored on the server.
  - **GameAgeRating** - The ESRB age rating for a game.
- **DEVELOPED** - This table stores a many-to-many relationship between the Developer Studio table and the Game table.
  - **StudioID** - (**FORIEGN**) Studio ID.
  - **GameID** - (**FORIEGN**) Game ID.
  - **UNIQUE (StudioID, GameID)** - Unique key constraint to prevent insertion duplication of the same exact StudioID and GameID.

## Tables/Attributes Details (Continued)

- **RATES** - This table stores a many-to-many relationship between the User table and the Game table. Many Users can rate many Games.
  - **Rating** - The star rating for a game (0.0 stars being the worst, 5.0 stars being the best).
  - **UserID** - (**FORIEGN**) User ID.
  - **GameID** - (**FORIEGN**) Game ID.
  - **UNIQUE (UserID, GameID)** - Unique key constraint to prevent insertion duplication of the same exact UserID and GameID.

## Functionality

### Main Functions of the Application

- INSERT INTO Queries
  - Allow the creation of a User account to play games. Users must be 13+ years or older to create an account.
  - Allow Users over 17+ years of age to create a Developer Studio.
  - Allow Developer Studios to upload games.
  - Allow Developer Studios to upload games with other developers.
- UPDATE Queries
  - Allow Users to customize their Account Settings, Profile Settings, and Profile Picture.
  - Allow Users to level up their accounts by earning points through rating games.
  - Allow Users over 17+ years of age to join a Developer Studio.
  - Allow Users to rate games.
  - Allow the owner/leader of the Developer Studio update/edit published games.
- DELETE FROM Queries
  - Allow the owner/leader of the Developer Studio to disband the Developer Studio.
  - Allow the owner/leader of the Developer Studio to remove published games.
  - Allow Users to remove their rating from a game.
- SELECT Queries
  - Allow Guests and Users to browse and play games.
  - Do not show 17+ age rated games to Guests.
  - Show appropriate games to the appropriate User's age based on the age rating of each game.
  - Allow Users to view their Developer Studio and the games that they have made.

## Advanced Queries

**Query #1 (Group 1)** [Find All Games Developed by the Studio with a given Studio ID]

Find Games that were developed by a Studio Developer with the given Studio ID of 1. Return the game's id, name, genre, release date, age rating, and directory location.

```
SELECT `Game`.`GameID`,
       `GameName`,
       `GameGenre`,
       `GameReleaseDate`,
       `GameAgeRating`,
       `GameDirLocation`
    FROM `DeveloperStudio`
        NATURAL JOIN `DEVELOPED`
        NATURAL JOIN `Game`
    WHERE `StudioID` = 1
    GROUP BY `Game`.`GameID`, `GameName`;
```

**Query #2 (Group 2)** [Display Popular Games on Pages]

Finds all popular games with or without an average rating on page #4 (each page returns 5 games). Games with no ratings are included so they can be displayed to a user (Must use left outer join). The user must only recieve games with an age rating less than or equal to 14. The average rating is calculated from all users that rated the game via the RATES table. Order by the average rating for each game in descending order, then order by the game's name in ascending order. Return the Game's average rating, number of ratings, id, name, genre, release date, age rating, and game directory location.

```
SELECT avg(`Rating`) AS `AvgRating`,
       count(`Rating`) AS `NumOfRatings`,
       `Game`.`GameID`,
       `GameName`,
       `GameGenre`,
       `GameReleaseDate`,
       `GameAgeRating`,
       `GameDirLocation`
    FROM `Game`
        NATURAL JOIN (SELECT *
                      FROM `DEVELOPED`
                      GROUP BY `GameID`) AS `UniqueDeveloped`
        NATURAL JOIN `DeveloperStudio`
        LEFT OUTER JOIN `RATES`
            ON `Game`.`GameID` = `RATES`.`GameID`
    WHERE `GameAgeRating` <= 14
    GROUP BY `Game`.`GameID`, `GameName`
    ORDER BY `AvgRating` DESC, `GameName` ASC
    LIMIT 5 OFFSET 15;
```

## Advanced Queries (Continued)

**Query #3 (Group 3)** [Advanced Boolean Mode Search Query with Total Average Rating]

Find all popular games that are related to the given search string input variable 'Studios'. For a display page, each page returns 5 games. Only return the games on page #2. Games with no ratings are included so they can be displayed to a user (Must use left outer join). The user must only recieve games with an age rating less than or equal to 16. The average rating is calculated from all users that rated the game via the RATES table. Games must have a rating higher than or equal to the average total rating of all games. Order by the most popular game to the least popular game, then order by the game's name in ascending order. Return the Game's average rating, number of ratings, id, name, genre, release date, age rating, and game directory location.

**Note:** I discovered that CONCAT is required because when the input 'Studios' is replaced with '?', this causes issues in NodeJS's npm mysql module when dynamically parsing input from the mysql module to MySQL. For example, "`LIKE CONCAT('%', '?', '%')`" will not cause an error when parsed by the mysql module, but replacing that line with "`LIKE '%?%'`" will throw an error for the mysql module.

```
SELECT avg(`Rating`) AS `AvgRating`,
       count(`Rating`) AS `NumOfRatings`,
       `Game`.`GameID`,
       `GameName`,
       `GameGenre`,
       `GameReleaseDate`,
       `GameAgeRating`,
       `GameDirLocation`
    FROM `Game`
        NATURAL JOIN (SELECT *
                      FROM `DEVELOPED`
                      GROUP BY `GameID`) AS `UniqueDeveloped`
        NATURAL JOIN `DeveloperStudio`
        LEFT OUTER JOIN `RATES`
            ON `Game`.`GameID` = `RATES`.`GameID`
    WHERE `GameAgeRating` <= 14
        AND (MATCH (`GameName`, `GameGenre`) AGAINST ('Studios' IN BOOLEAN MODE)
            OR MATCH (`StudioName`) AGAINST ('Studios' IN BOOLEAN MODE)
            OR `GameName` LIKE CONCAT('%', 'Studios', '%')
            OR `GameGenre` LIKE CONCAT('%', 'Studios', '%')
            OR `StudioName` LIKE CONCAT('%', 'Studios', '%'))
    GROUP BY `Game`.`GameID`, `GameName`
    HAVING `AvgRating` >= (SELECT avg(`AvgGameRating`) AS `AvgTotalGameRating`
                           FROM (SELECT avg(`Rating`) AS `AvgGameRating`
                                 FROM `Game`
                                     NATURAL JOIN `DEVELOPED`
                                     NATURAL JOIN `DeveloperStudio`
                                     LEFT OUTER JOIN `RATES`
                                         ON `Game`.`GameID` = `RATES`.`GameID`
                                 GROUP BY `Game`.`GameID`, `GameName`
                                ) AS `TableAvgGameRating`
                          )
    ORDER BY `AvgRating` DESC, `GameName` ASC
    LIMIT 5 OFFSET 5;
```

## Stakeholders:

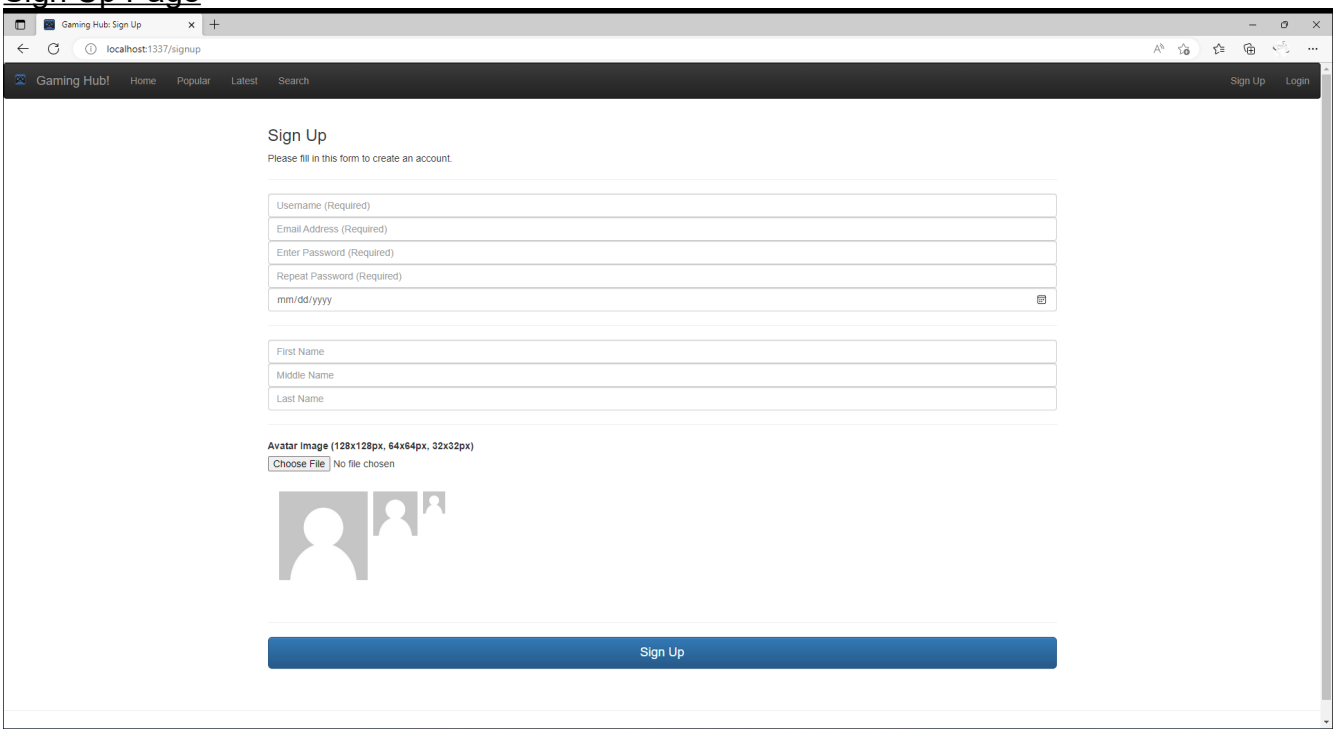
The users of the my application will be age 13 years and older who are interested in games. For 13 to 16 year olds, they are able to use the site to socialize with friends and play games together. Some may play singleplayer games for fun, or are just looking for leisure time. For 17+ year olds, I expect it to be similar. Otherwise, 17+ year olds may want more mature rated games (based on the ESRB rating system) and the site should be expected to not filter games rated for 17+ year olds.

## Technological Requirements:

The platform is developed as a web application. The front-end language for the applicaiton GUI is Pug.js (formerly known as Jade). The back-end languages are Node.js, Express.js, Unity WebGL. The database that stores all the table data is MySQL. To connect to and query from MySQL, the "mysql" module from Node.js is used. For the games, they are compiled and built with WebGL using the Unity Game Engine. To save a bunch of time with Unity WebGL development, all games except for my Modular Maze Demo were open-sourced from GitHub.com and Code-Projects.org.

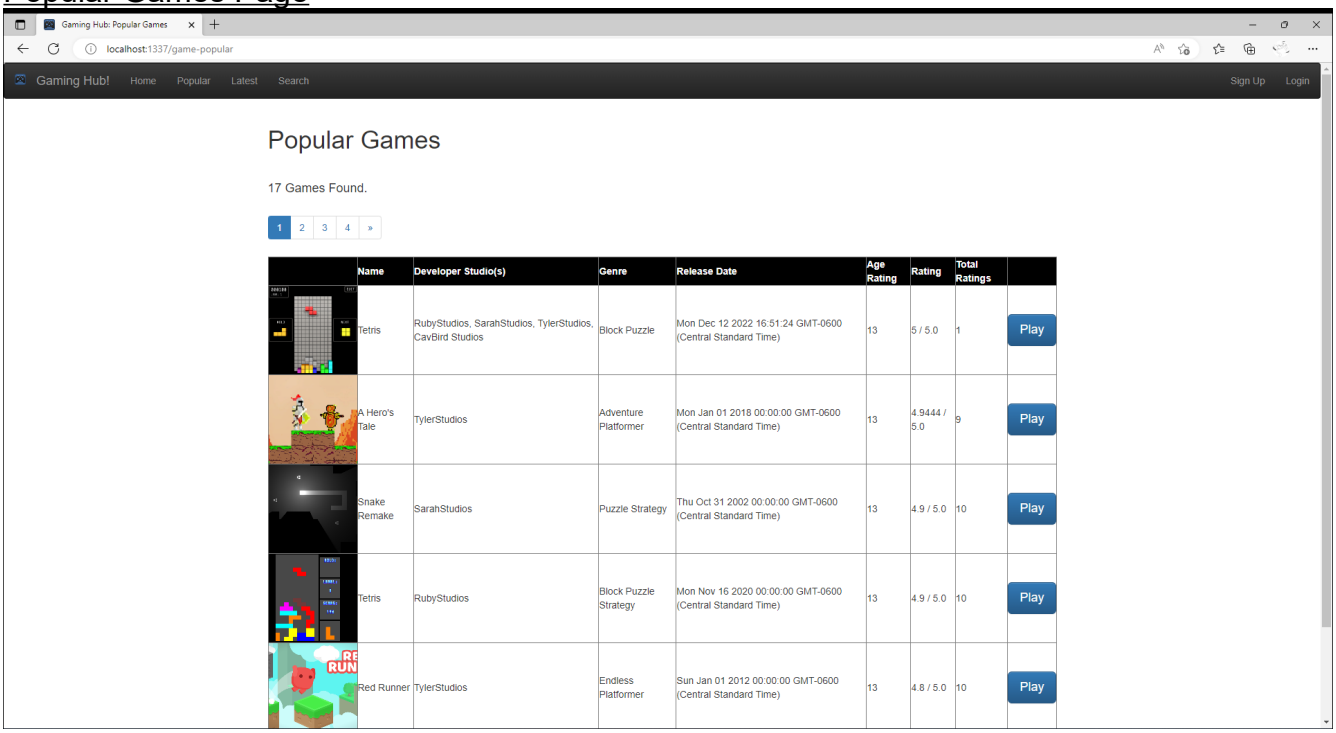
## Screenshots

Sign Up Page



# Screenshots (Continued)

## Popular Games Page




## Advanced Search Query Page