Database Design

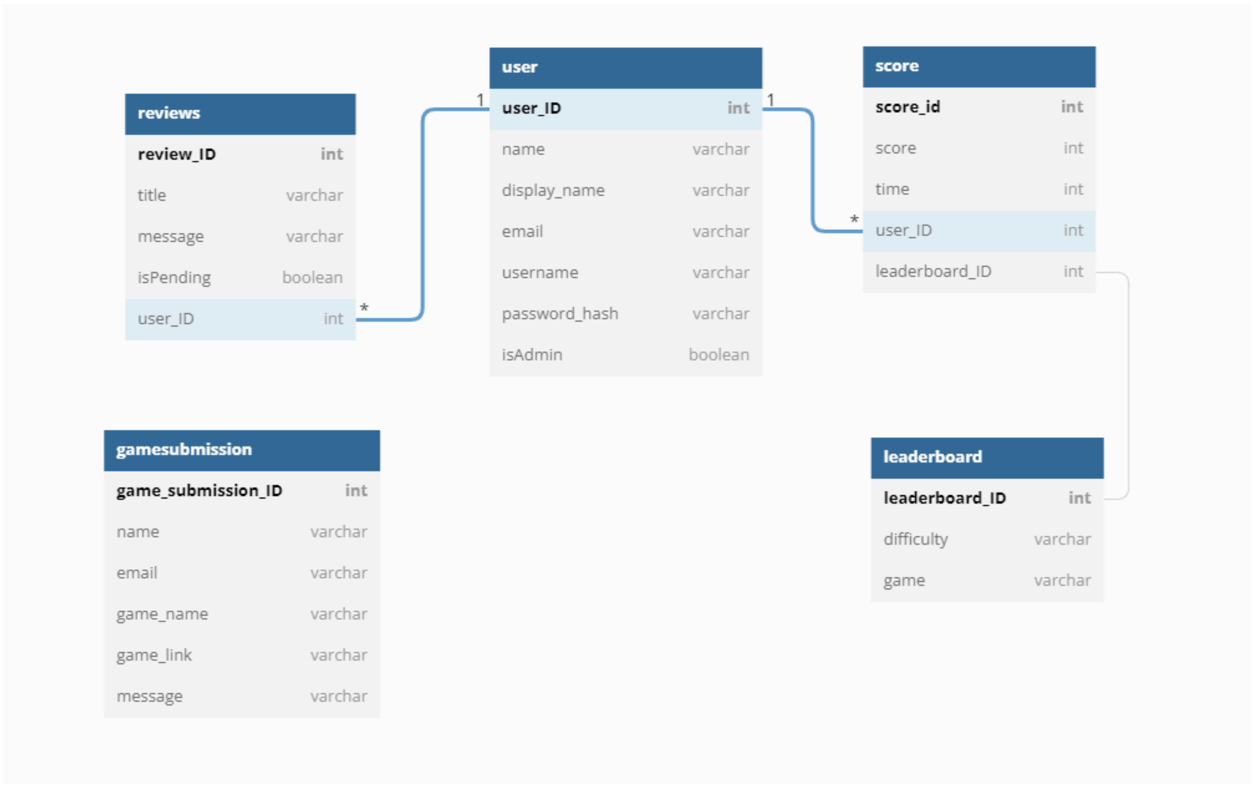
Alexei Cogdill

OVERVIEW

I will be utilizing SQLite to store all the data that is relevant to my web application. I chose this database due to how my web application is built and functions. The relationships between the tables created in SQLite will allow any user to render the data in the appropriate webpages.

The web application will interact with the database through a REST API that is accessed by the user on the frontend of the web application. All requests made by the user involving data from the database will transmit through the API instead of interacting with a database layer on the frontend of the application.

Table Design



User

The users table contains an id primary key for identification within the database. There are also fields that would attribute to the user’s personal information, being the name, display_name, email, username, and password_hash fields. This information will be used to identify different users. Lastly a field called isAdmin is passed as a Boolean to identify if the user is an admin of the application or not. This data will be used to track user information.

Below is how the table is broken up.

```
Table user as U {

  user_ID int [pk, increment]

  name varchar

  display_name varchar

  email varchar

  username varchar

  password_hash varchar

  isAdmin boolean

}
```

Score

The score table contains an id primary key for identification, a score field for tracking the score, a time field for getting the amount of time the user took to finish the game, a foreign key connection to my users table, and a foreign key connection to my leaderboards table. A one-to-many relationship exists between the users and score tables as a user can have multiple scores for a single game. A many-to-one relationship exists between the score and leaderboard table because multiple scores can exist within one leaderboard for a game. This table functions as a bridge between the user and leaderboard tables as it identifies which score belongs to who and to which leaderboard it goes to.

Below is a breakdown of the score table.

```
Table score as S {

  score_ID int [pk, increment]

  score int

  time int

  user_ID int

  leaderboard_ID int

}


Ref: U.user_ID < S.user_ID

Ref S.leaderboard_ID > L.leaderboard_ID
```

Leaderboard

My leaderboard table will contain an id primary key for identification, a difficulty field to show the user which level of game difficulty leaderboard, and a game field that is used to identify which leaderboard belongs to which game. This table allows for multiple leaderboards to exist and represent different games with multiple difficulties.

Below is the breakdown of this table.

```
Table leaderboard as L {

  leaderboard_ID int [pk, increment]

  difficulty varchar

  game varchar

}
```

Reviews

My reviews table will contain an id primary key for identification, a title field for the review title displayed, a message field for the contents of the review left by the user, a Boolean value called isPending to check if the review is still pending for an admin to accept, and a foreign key connection to the user table. A one-to-many relationship exists between the user and reviews table since a user can create multiple reviews. This table functions as a storage for the data for each review created by a user.

Below is a breakdown of this table.

```
Table reviews as R {

  review_ID int [pk, increment]

  title varchar

  message varchar

  isPending Boolean

  user_ID int

}


Ref: U.user_ID < R.user_ID
```

Game Submission

My gamesubmission table will be apart of the stretch feature that is planned after building the requirements for the MVP. This table will include an id for the primary key for identification, a name field for your name, an email field so that I can email you if the game gets accepted, a game name field that will be used to indicate what their game name would be, a game_link field so that I can utilize and embed into my application, and lastly, a message field indicating why you think your game should be implemented into my application. This table does not have any relationships as you are not required to have an account to submit an application. Again, this would be built for the stretch feature.

Below is a breakdown of this table.

```
Table gamesubmissions as GS {
  game_submission_ID int [pk, increment]
  name varchar
  email varchar
  game_name varchar
  game_link varchar
  message varchar
}
```