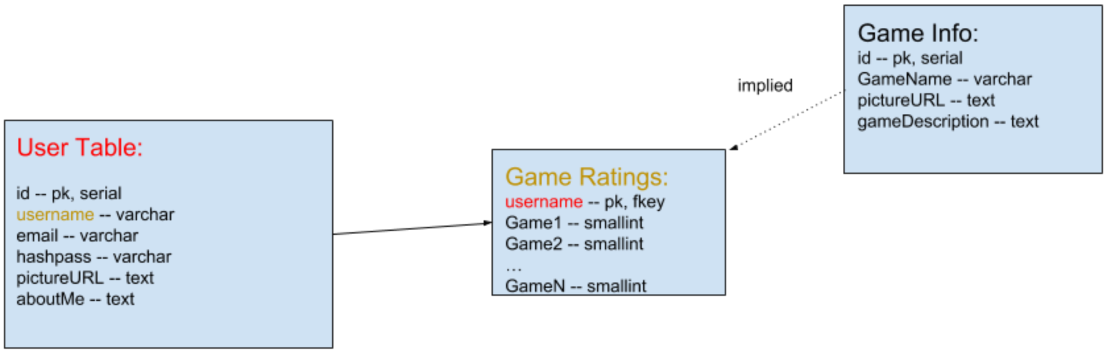
Milestone 4

Table diagram (large one included in folder)



User Table:

id -- pk, serial
username -- varchar
email -- varchar
hashpass -- varchar
pictureURL -- text
aboutMe -- text

Game Ratings:

username -- pk, fkey
Game1 -- smallint
Game2 -- smallint
…
GameN -- smallint

Game Info:

id -- pk, serial
GameName -- varchar
pictureURL -- text
gameDescription -- text

Logic behind tables:

Since we are making a game recommendation website, the schema needs to be organized in a way where the data needed for generating user recommendations is as accessible as possible. To recommend users games we essentially just need a matrix of game ratings where each row (in this case) corresponds to a specific user and each column corresponds to a specific game. So the game ratings table is organized in that manner.

The user table contains all the information about the user, it is also the parent table of game ratings through a foreign key (username) which should help facilitate joins.

The game info table is a standalone table, but implicitly it is linked to the ratings table. Each game column in the ratings table maps to a specific game in the game info table. The game's name will be used as the name for the column. This constraint will need to be enforced in the javascript handling database transactions rather than the database itself.

Management software. We will be using postgresql to manage the database.

Population scripts (commands.sql):
https://github.com/CSCI-3308-T5/milestones/tree/master/ProjectMilestone4_T5