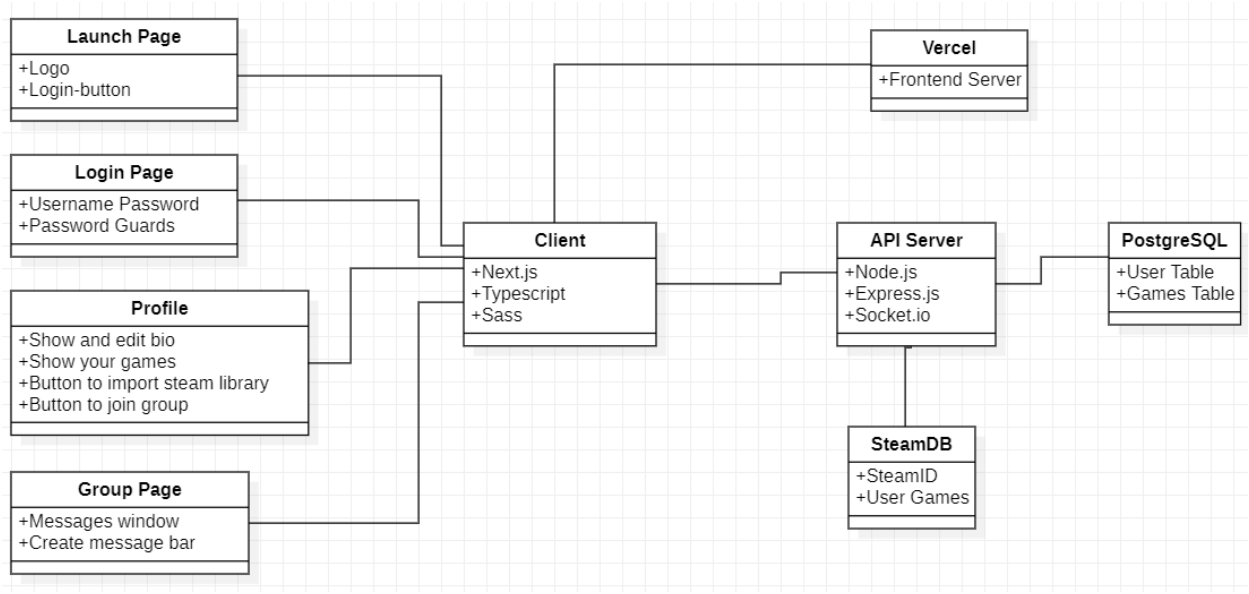
Project Milestone 4

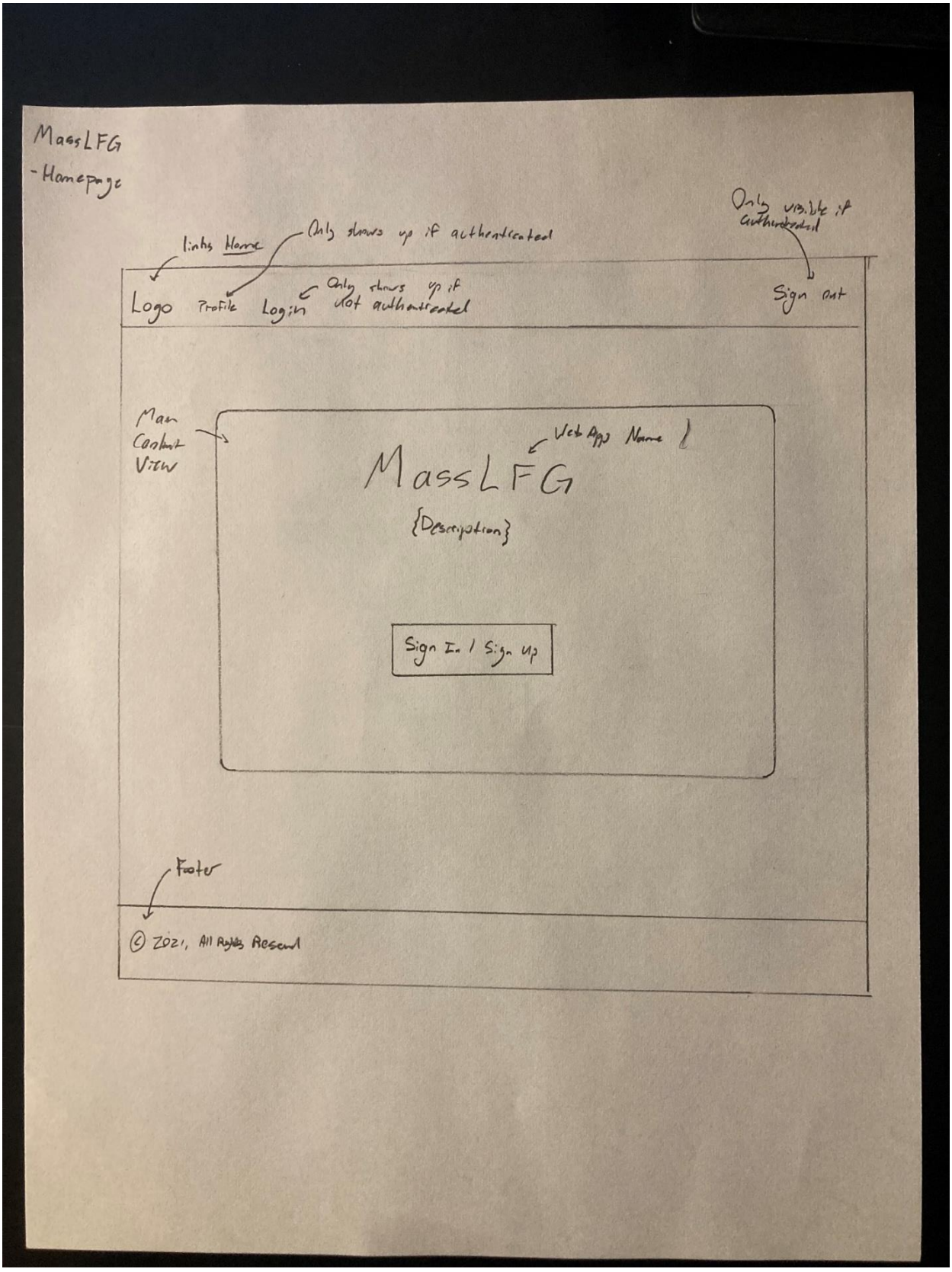
Feature List (features with highest priority are listed first):

- Import Steam Library
    - Description: MassLFG users are able to log into their Steam account to import their steam library. MassLFG uses these games to pair users into groups
- Group Chat
    - Description: When MassLFG users have been paired with a group, they are able to communicate with one another by means of a “group chat” style messaging window.
- Customize User Profile
    - Description: MassLFG users can easily customize their bio and the look and feel of the client interface.
- Account Registration
    - Description: Users will be able to create and log into dedicated accounts which hold information about their gaming preferences.
- Instant Group Finding
    - Description: When MassLFG users are ready to game, they simply hit the “Find Group” button which instantly pairs users with groups of gamers who are playing the same games.

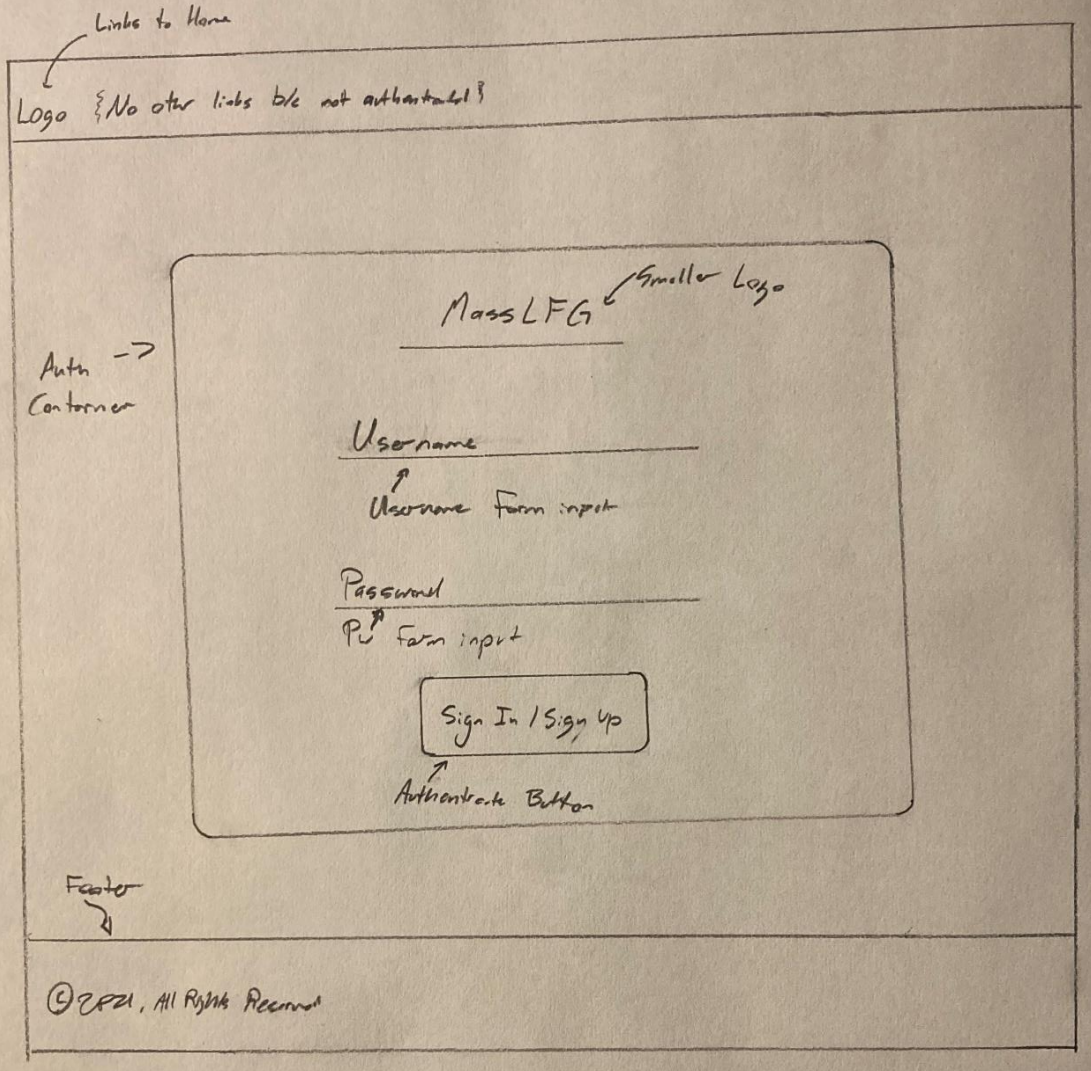
Architecture Diagram



Front End Design

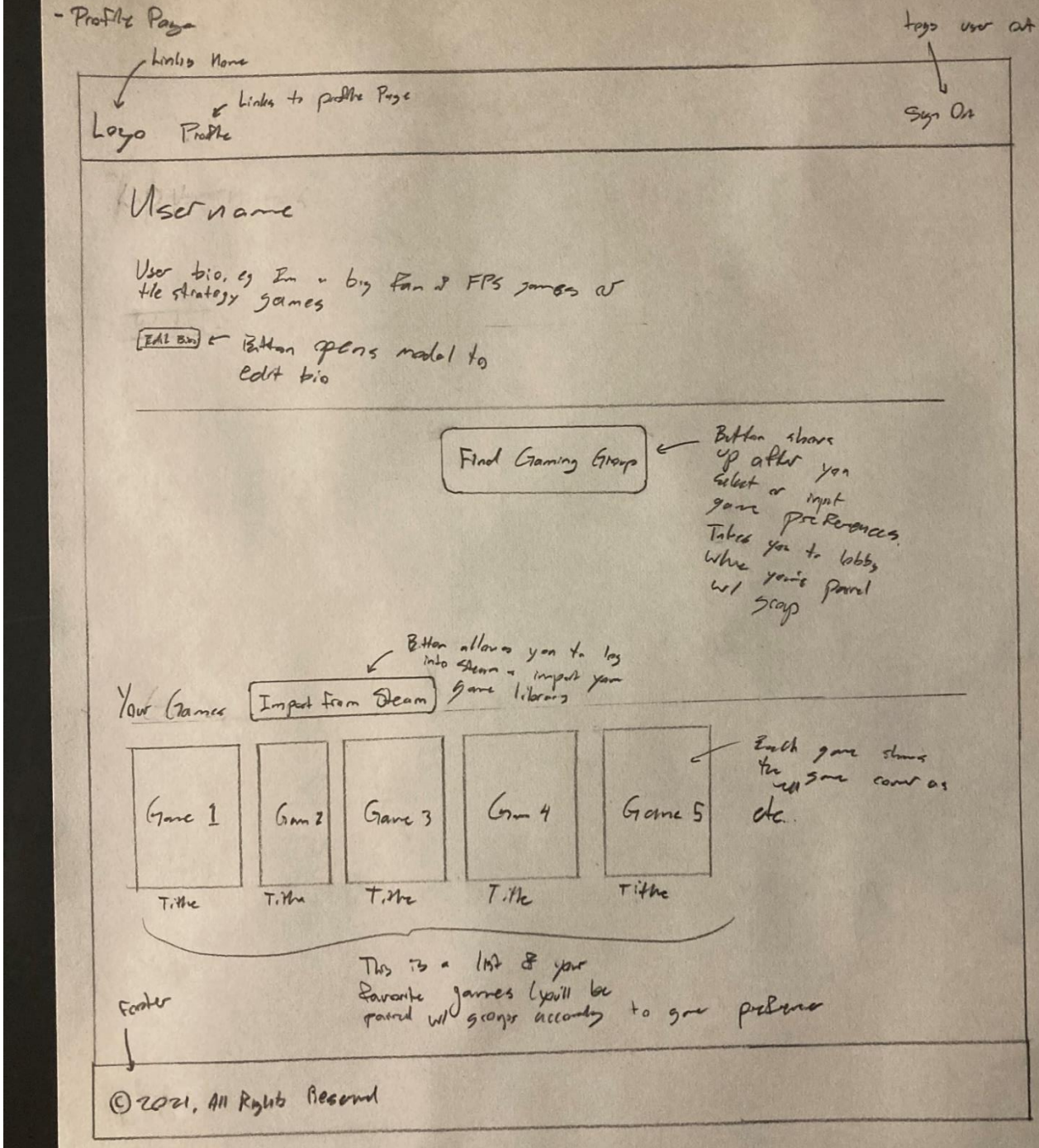


Mass LFG
- Login



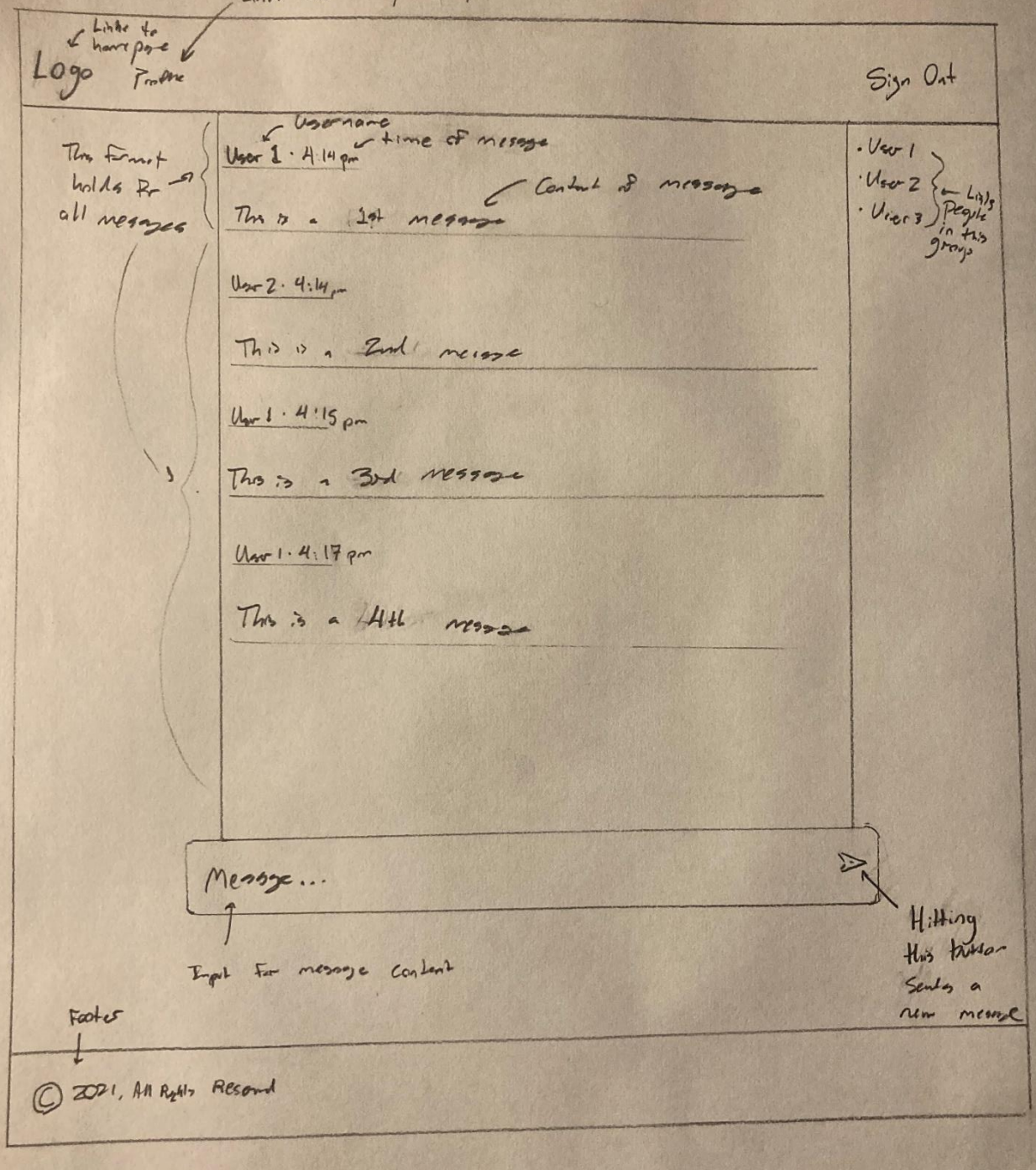
Moss LFG

- Profile Page



MessInFG

- Group Chat



Web Service Design

Steam: we're using SteamKit to allow users to log into their Steam accounts and then subsequently capture their Steam Account information in our Express server. We then use this information (specifically the user's SteamId) together with our Steam API key to query Steam's API for the user's public game library.

Database Design

We are using Postgres together with a docker configuration very similar to what we have been working with in the labs.

We will be using two main tables defined as below:

```
user_info (
        username VARCHAR(200) PRIMARY KEY,
        steamid VARCHAR(200),
        bio VARCHAR(200),
        password VARCHAR(200)
);

games_info (
        steamid VARCHAR(200) PRIMARY KEY,
        gameid  VARCHAR(200)
);
```

The first table holds information about each individual user (including their steamid). We'll use this table for anything that requires user-specific actions.

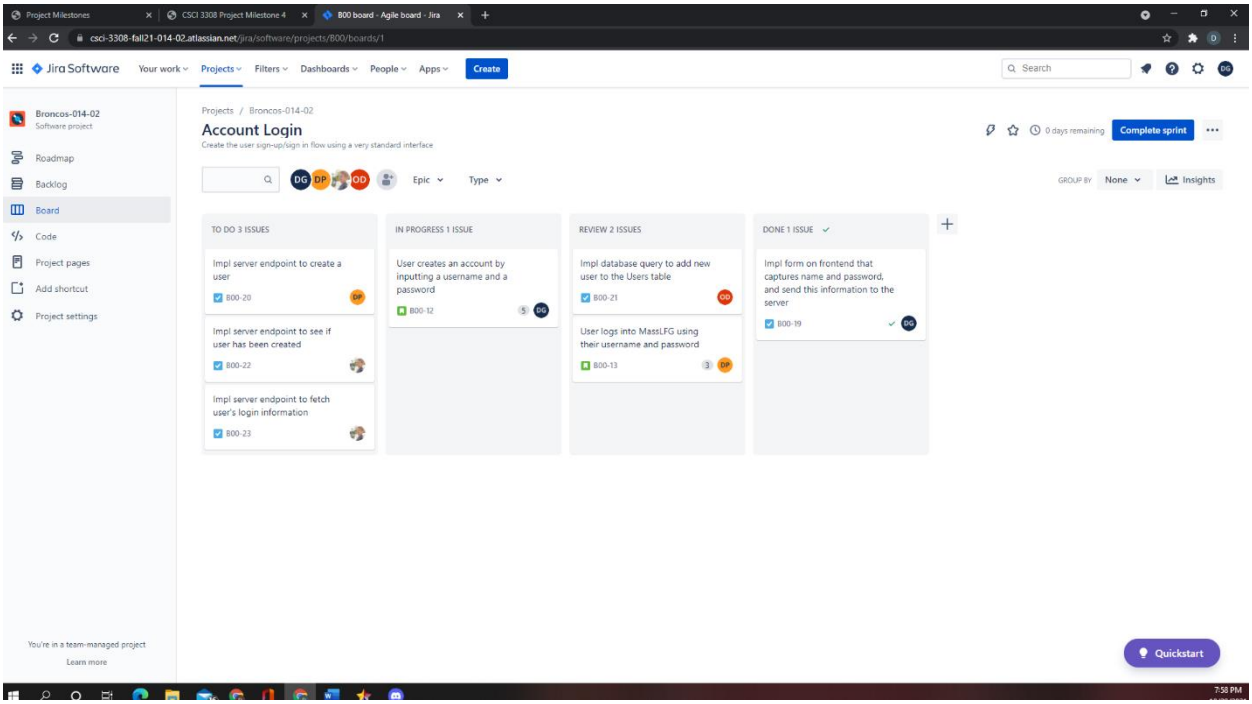
The second table is a one-to-many table which associates users (by steamid) with the ids of the games in their steam library. We'll use this information to pair users into gaming groups.

Challenges

1. Steam Api: The Steam API is fairly finicky, and one of our group members got denied service to the API. As such, we only have limited access to information about games in the user's steam library
    a. Backup Plan: Use the limited information for now (which does include game name) and look for other options to obtain more information about the games.
2. Pairing Algorithm: We're still trying to figure out how to efficiently pair users based on their preferred games.
    a. Backup Plan: If everything else fails, we can simply use a very inefficient SQL query against Postgres to literally just find other users who have the same games in their library. It will work, but it's certainly not particularly "smart."
3. User Authentication: We're trying to figure out how to securely hold user information during authentication, as it's very poor practice to physically store passwords in a database.
    a. Backup Plan: For now, we're just going to store the passwords themselves, and we get the correct behavior on the frontend, even if it is a very poor pattern.

Individual Contributions

- Danny Geisz:
    - Contribution: Used socket.io to complete messaging functionality on the frontend; added Express logic to handle Steam authentication and redirect; added components to show user games on the frontend.
    - Last Commit: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-014-02/commit/b647759b094c8eb504631c65985321635ab558ba
- Aaron Grissom
    - Contribution: Helped integrate Steam authentication and API access in the Express server
    - Last Commit: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-014-02/commit/8eda73c516e29ac47f2c1926e5e9a818e8e229ed
- Daniel Pearsall
    - Contribution: Planning docker deployment to Heroku
    - Last Commit: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-014-02/commit/647633f89b3019888ab0fbde05f89951c2c7f449
- Oliver Doig
    - Contribution: Creating Postgres architecture; planning pairing algorithm
    - Last Commit: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-014-02/commit/9d5758fdf3ee70f03fc804f476c736ccbed59e9d


Project Milestones
CSCI 3308 Project Milestone 4
B00 board - Agile board - Jira
csci-3308-fall21-014-02.atlassian.net/jira/software/projects/B00/boards/1
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Broncos-014-02
Software project
Roadmap
Backlog
Board
Code
Project pages
Add shortcut
Project settings
Projects / Broncos-014-02
Account Login
Create the user sign-up/sign in flow using a very standard interface
0 days remaining
Complete sprint
Epic
Type
GROUP BY
None
Insights
TO DO 3 ISSUES
Impl server endpoint to create a user
B00-20
Impl server endpoint to see if user has been created
B00-22
Impl server endpoint to fetch user's login information
B00-23
IN PROGRESS 1 ISSUE
User creates an account by inputting a username and a password
B00-12
REVIEW 2 ISSUES
Impl database query to add new user to the Users table
B00-21
User logs into MassLFG using their username and password
B00-13
DONE 1 ISSUE
Impl form on frontend that captures name and password, and send this information to the server
B00-19
You're in a team-managed project
Learn more
Quickstart
7:58 PM