Milestone 3

What features were completed?

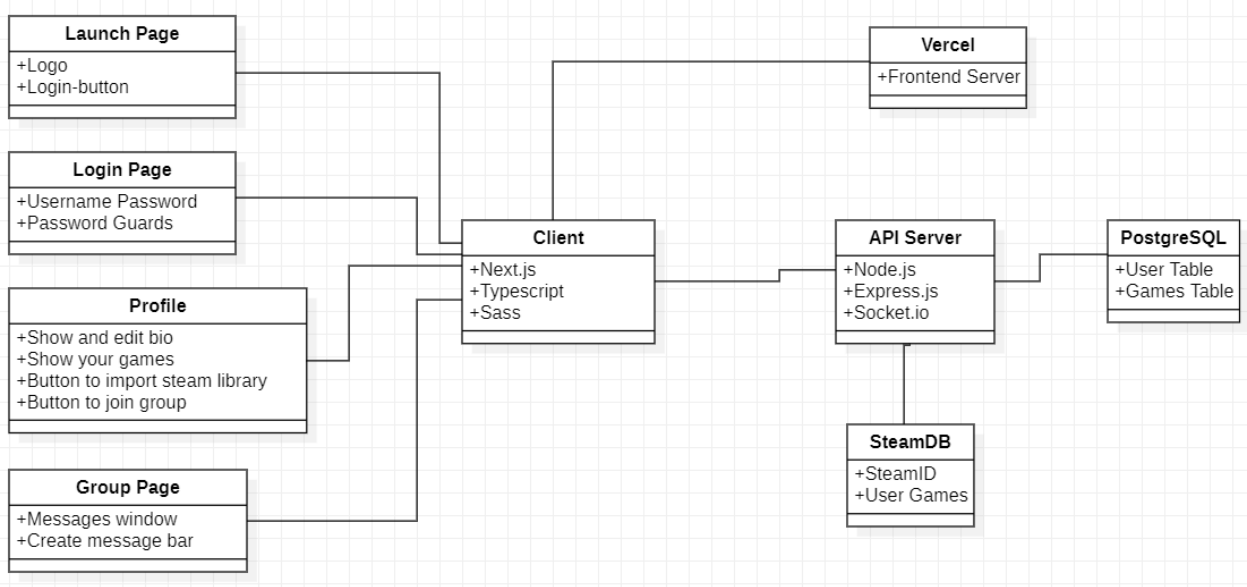
- Launch Page
  - Simple page showing information about the app as well as a prompt to Sign in
- Sign in
  - Created page with forms for inputting username and password
  - Included password guards of the same form as Lab 5
  - Created SQL commands to create tables that hold user authentication information
- Profile Page Skeleton
  - Profile page containing the user's username, bio, and Steam Library
- Import Steam Button
  - On the profile page, the user clicks "Import Steam Library" which takes them to the Steam sign-in page. Once the user signs in, our server captures information necessary to fetch the user's library of games.
- Group Messaging
  - In the group page, users can send chat messages to one another using a simple "Discord-like" interface.
  - Realtime messaging is handled with socket.io and express.

What worked during the demo?

All the features mentioned above worked during the demo. The Group Messaging feature is still quite minimalistic, but it is functional. Also note that we haven't created any of the database integrations yet given that SQL was only recently covered in class.

See the next page for a high-level diagram of the project.

Project UML



What issues were faced during development or during the demo?

The demo actually went quite smoothly, we didn't face any significant hurdles.

Most of development has been pretty standard in terms of building the frontend and express server. The most challenging feature so far was the Steam integration. First, we had to figure out how to even interact with the Steam service properly. However, we ultimately found a Steam Library for JavaScript that can be used quite easily as an express middleware. After that point, it was a matter of handling client-side redirects properly.

Though it wasn't particularly difficult, the only other particularly complex feature is the Group Messaging service. This service relies on Socket.io, which is a simple WebSocket wrapper that allows developers to implement message passing with simple event handlers.

What were the suggestions offered by the TA?

We are quite up to date with the course material and project expectations, so the TA didn't feel the need to offer many suggestions. We did briefly discuss whether the WebSocket protocol uses the same level of encryption as https (it can and does). Aside from that, the TA encouraged us to simply keep moving in the direction we are already heading.

Individual contributions by each team member

Danny Geisz:

- Implemented the skeletons of the pages on the frontend. Integrated socket.io both on the frontend and in the express server.

Aaron Grissom:

- Created the integration between the express server and the Steam API.

Oliver Doig:

- Designed the database model, created SQL scripts to generate the necessary tables in PostgreSQL

Daniel Pearsall:

- Created the basic outline of the express server used as our projects API server.