EECS 581 – Project 3 Architecture

Team Details

Group

Group 20

Members

- Aiden Burke
- Andrew Reyes
- Brett Suhr
- Nicholas Holmes
- Riley Meyerkorth
- Ty Farrington

Project Details

Name

PokerCavern

Synopsis

A browser-based game based in a casino that hosts several various games and interesting interactions.

Architecture Description

Base

At the base level, the game will utilize a React Vite web application. The React Vite library was chosen due to its modern appeal, our team's familiarity with the framework, and the simplicity in starting web applications from scratch using it.

User Data

User data for the player will be stored locally. This is because it an effective way to store data for us and is much easier than utilizing a backend database.
There are various pieces of user data that are to be stored, including:

- Username
- Color/cosmetic details
- Gameplay statistics
- Achievements unlocked and progress

Statistics and Achievements

We plan on utilizing statistics and achievements to help give the users more incentive than just simply playing the game. Some example statistics could be:

- Play time
    - In total
    - Per game
    - Per shop
- Account age
- Wins/losses
    - Per multiplayer game
    - Per singleplayer game
    - In total
- Money spent
    - In shops
    - In games
    - In total
- Total users met

Multiplayer

To implement multiplayer, we thought of many different avenues we could take to create a connection between clients. In the end, we decided to focus on LAN-based connections through a tiny Node.js backend server to handle WebSocket connections. This way, we could save a lot of time and effort by favoring LAN connections over WAN connections, and our server implementation could be as small as possible.

Player Navigation

For the application, we plan on implementing a single-page approach to host the game world with as little page routing as possible. This allows us to keep a lot of the gameplay centralized, as well as helps us mitigate potential routing issues that may arise.

Main Gameplay Loop

Overall, the generic gameplay loop is as follows:

1. The player opens the game.
    a. If this is the first time, the player will be prompted to enter a name and potentially other details.
2. The player is connected to the backend server.
    a. If there is an issue with the connection, an error message will be displayed.
3. The player will spawn in the main hub (the main page).

    a. From the hub, the player is able to navigate around by clicking their mouse on the location they would like to go.
    b. The player is also able to enter certain sub-loops of gameplay. These include:
        i. Various casino games (poker, blackjack, slots, etc.)
        ii. Shop(s)
        iii. Menus (settings, cosmetics, achievements, stats, etc.)

Casino Games Gameplay Loop(s)

These gameplay loops begin when the player navigates their character to the game's specified location on the map.

1. If it is a multiplayer game…
    a. The player is connected to the lobby (if it is a multiplayer game)
    b. The first player who connects to the lobby is considered the "host" and can choose when the game begins
2. Once the game begins, the standard gameplay loop for each game takes place
    a. Ex: for slots, the player will spin over and over. For most other games, it will be turn-based.
3. Once the game ends, the user can choose to leave the lobby or stay.

Shop(s)

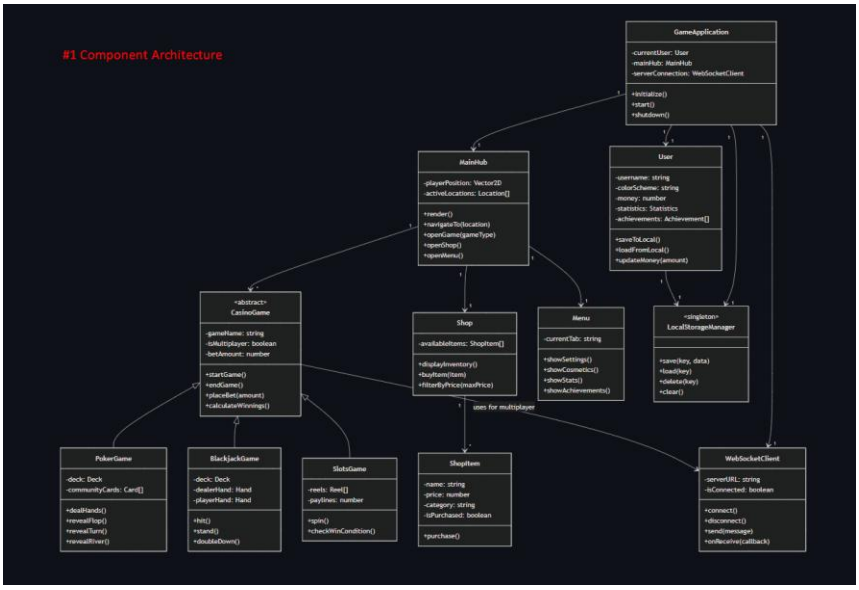
A shop is another way the player can use their accumulated money from their time playing casino games. The items purchased from the shop are primarily cosmetic-only.
The basic loop of shops is as follows after the player enters the shop on the map:

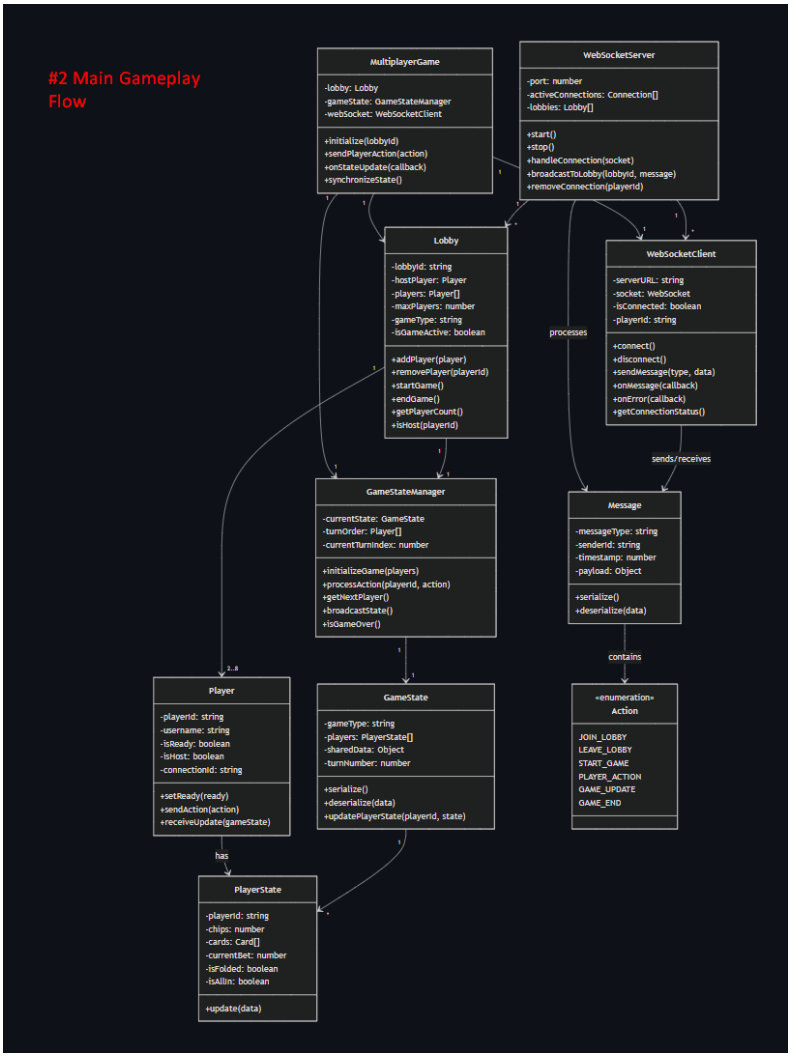
1. The player is prompted with a list/grid of items to choose from.
2. The player chooses the item(s) they want/can afford.
3. Once finished, the player can leave or continue shopping.

Menu(s)

Menus are for the more meta-aspect of the game. They are for things like settings, cosmetic changes, statistic/achievement viewing, etc. They are really nothing special; just simple UI elements.

UML Diagrams





#3 Multiplayer connection sequence