Final Project Write Up

**URL:** [**http://13.57.39.35/**](http://13.57.39.35/)

This web application is a multiplayer card game that allows four players to join a game room and has a real-time update of the player’s moves. The objective game is to select three cards, and the player that chooses the highest of their cards win the game. The first screen encountered is a login screen where each player must log in or register before being directed to the lobby. After logging in, the player will then be directed to the lobby where they wait for other players to join. Once four players join they are directed to the game room to play the card game. Those who select the highest card win and their total wins are updated and displayed on the login page under top scores.

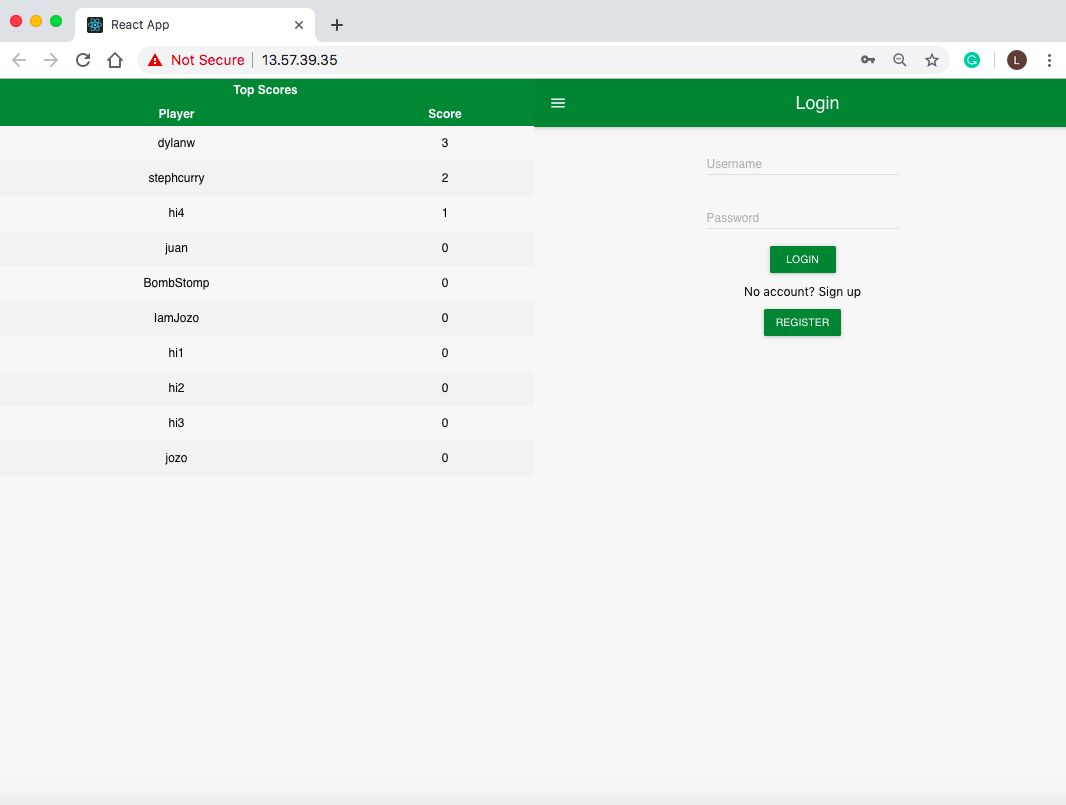
In order to effectively complete this project, we had two teams a front end team that handled the styling, user interface and utilizing redux and assisting with connecting it with the backend. The backend team was responsible for setting up express, the microserver architecture with nginx, Redis, websockets, and the mongodb database. The first step we took was to create the express server, a gateway route, and then the mongodb database. Once the database was created and connected, we began with the first screen, the login and registration screen, and then progressed to the lobby and then the game room. This project was done on a page by page basis.

The technologies used for the front end of this application were React.js and Redux with routes. We used Material-UI, a react UI framework for the login page and HTML and CSS for the display of the top scores, the game room, and the lobby. Redux was also used for the front end, redux accesses the state which was essential to have in our game. Redux was mainly used to keep track of the room of the current game.

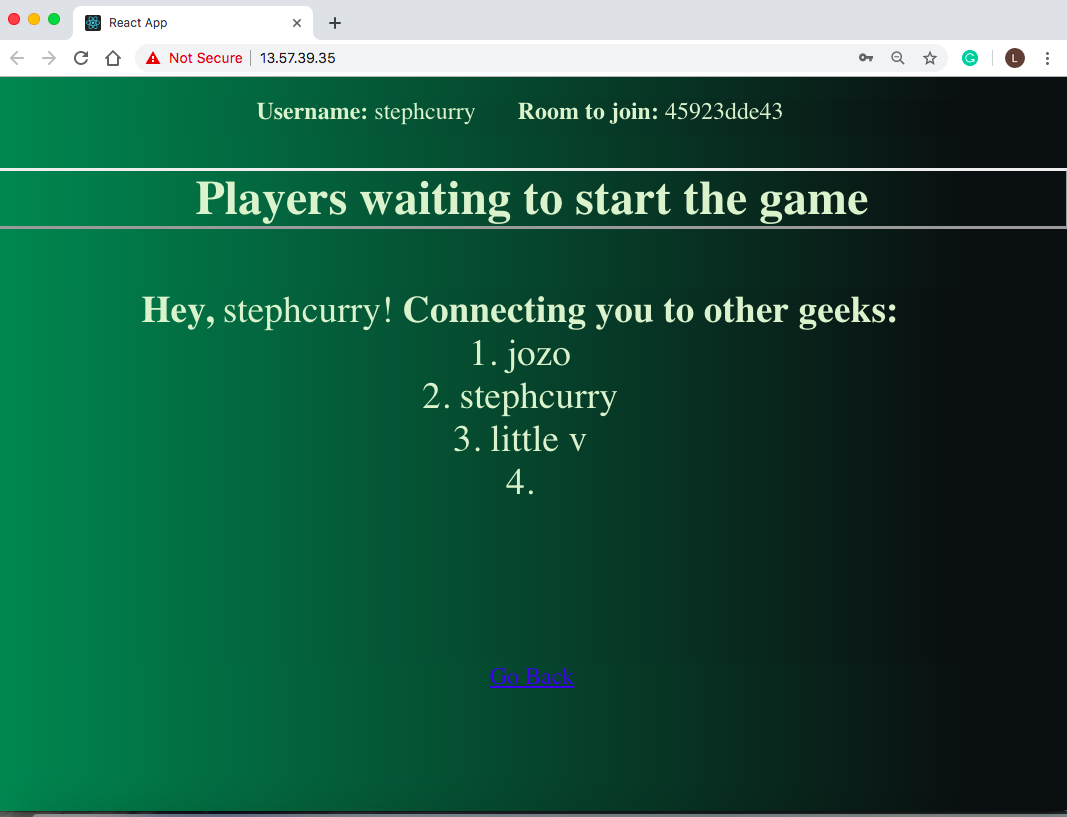
The back end used expess that had get and post endpoints using axios to get and post the scores, winners, usernames, passwords from login and registration. The database used was mongodb to load and store the usernames of the users and the username and the score of the winner of that game. The winner is stored so that the player that has the most wins has a top score that is displayed under top scores on the login page. The passwords are also stored, however they are hashed using bcrypt, a password hashing function. Websockets were heavily used in our game, socket.io was used. They were used in the lobby to tell each player who is in the lobby as they are added. Websockets were also used in the game room to manage which player’s turn it is, what card they got dealt and who won that game. Redis was also used to keep track of the current room. The microserver architecture was completed using nginx, and the gateway was used only to route requests by topics, such as users and winners.

There were several hardships encountered upon trying to complete this assignment. A big issue we had was that we were unable to put it on the AWS. This issue stemmed from setting up websockets, we used socket.io. In order to solve this issue, we had to include the route /socket.io in the gateway. We then got rid of gateway.js and made a switch from an express gateway to an nginx gateway. Another issue we had was styling the gameboard using MaterialUI, this issue was solved by using CSS, particularly the Flexbox layout module.

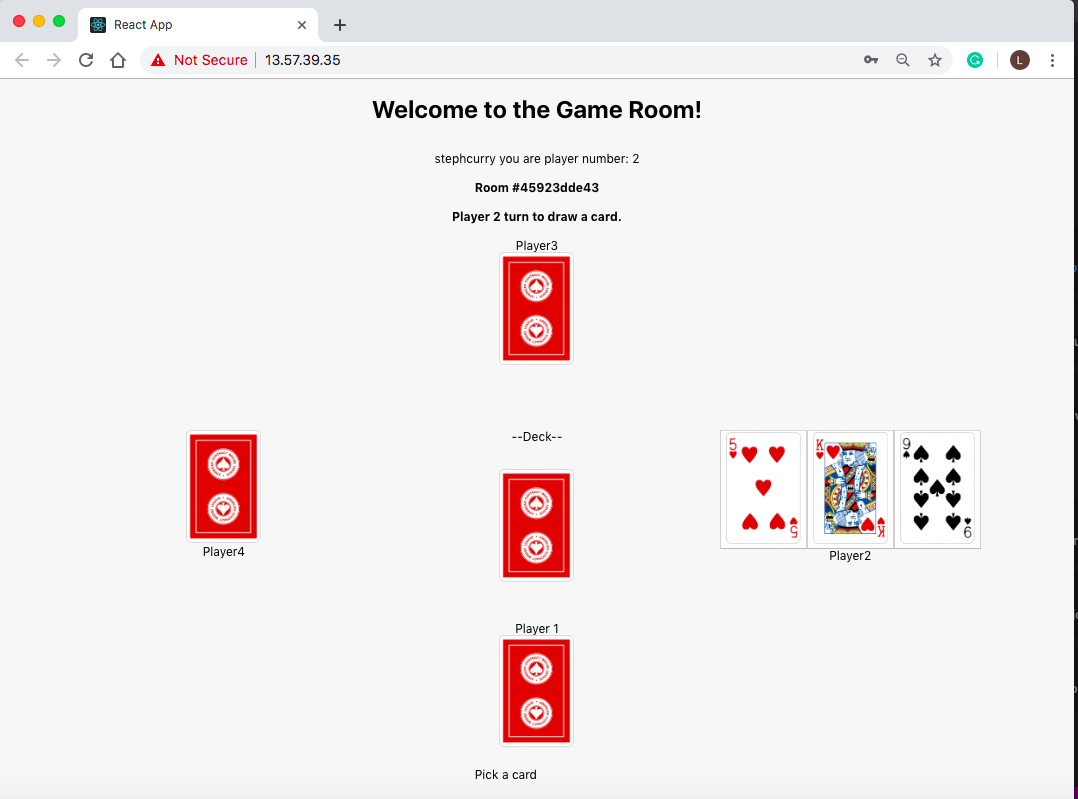
User Interface Screens:



*Login and Top Scores*



*Lobby*



*Game Room*