Course-Work 1

CST 2120: Web Applications and Databases

Jake Sajth

Student ID: M0103357

Table of Contents

Project Overview

For this coursework, the game idea I have picked is going to be a simple Rock, Paper, Scissors game. It includes account creation & user validation, a leaderboard system using local storage to track account data, and a score tracker alongside guest functionality that allows you to play the game without signing up.

The rock, paper, scissors game is nothing short but a simple game. In this, I made it so there are 5 rounds. By pressing start round, whoever has the highest score by the final round will win and earn the score for the leaderboard. If the player does not win, the score is not counted. The round may also be ended early by clicking the end round button, but it will still check for who had the highest score. I had many ideas and future improvements for this project, but due to heavy time constraints, I was not able to add everything, like abilities and more. Each round has a timer of 3 seconds. If the player does not make a choice in time, they will automatically lose, and the whole match ends.

It is important to note that in order to run this project, make sure you do not change the folder names of any of the files and ensure they are located preferably in the desktop.

To open up the project, open the folder in VS Code and install Live Server. After doing so, run the live server in Chrome and open home_page.html as it will be the starting point. Due to a bug, it is recommended to run the game with a live server.

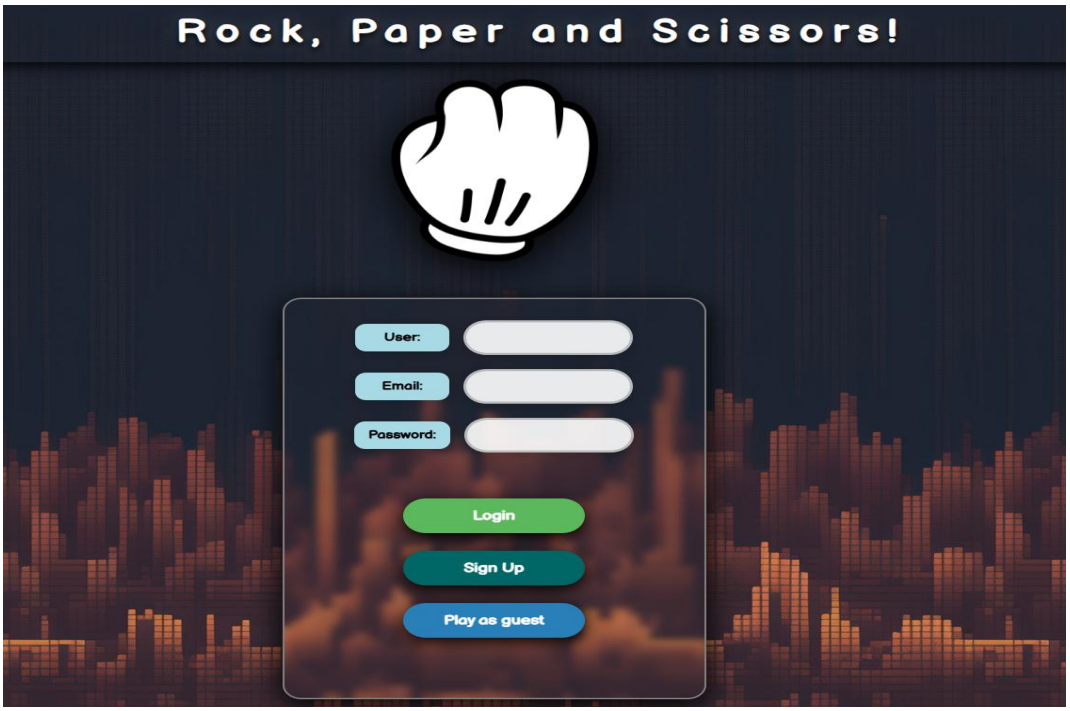
In case you would like to view the account data or see how it's managed, press the shortcut Ctrl + I on Chrome and head to the applications tab, where it will display local storage. Upon signing up, your newly created account will appear as your email with its index, which will display the username, password, phone, age, etc.

File Structure

The project consists of three main HTML pages:

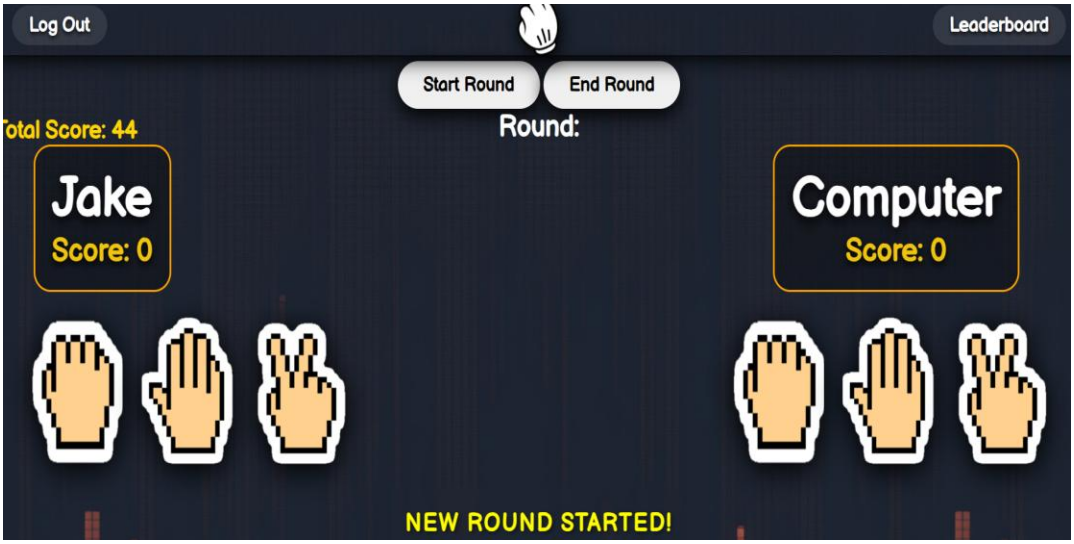
- **Home_Page.html** - Login/Signup interface (Figure 1.0)

Figure 1.0



- **Game_Page.html** - Main game interface (Figure 1.1)

Figure 1.1

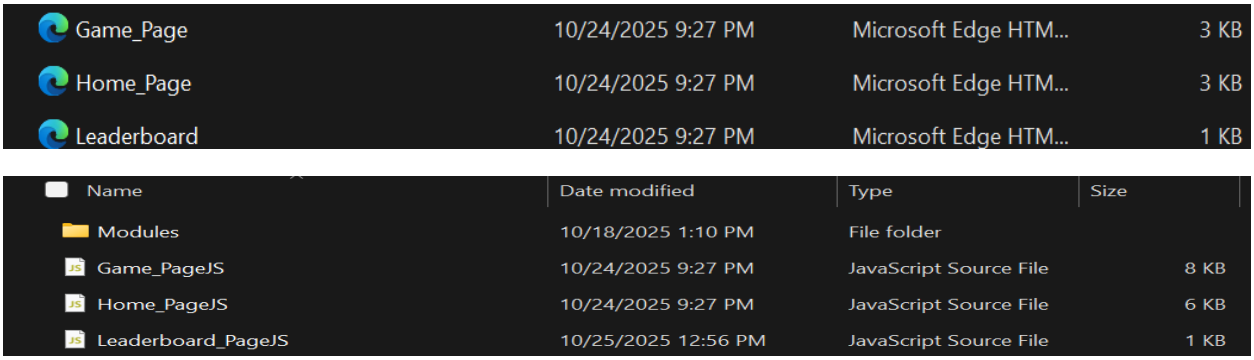


- **Leaderboard.html** - Score ranking display (Figure 1.2)

Figure 1.2

The pages of the HTML correspond with the names of the JavaScript files (which total to 3). (Figure 1.3)

Figure 1.3



Followed by one CSS file and assets that will include the images required for the game's visualization.

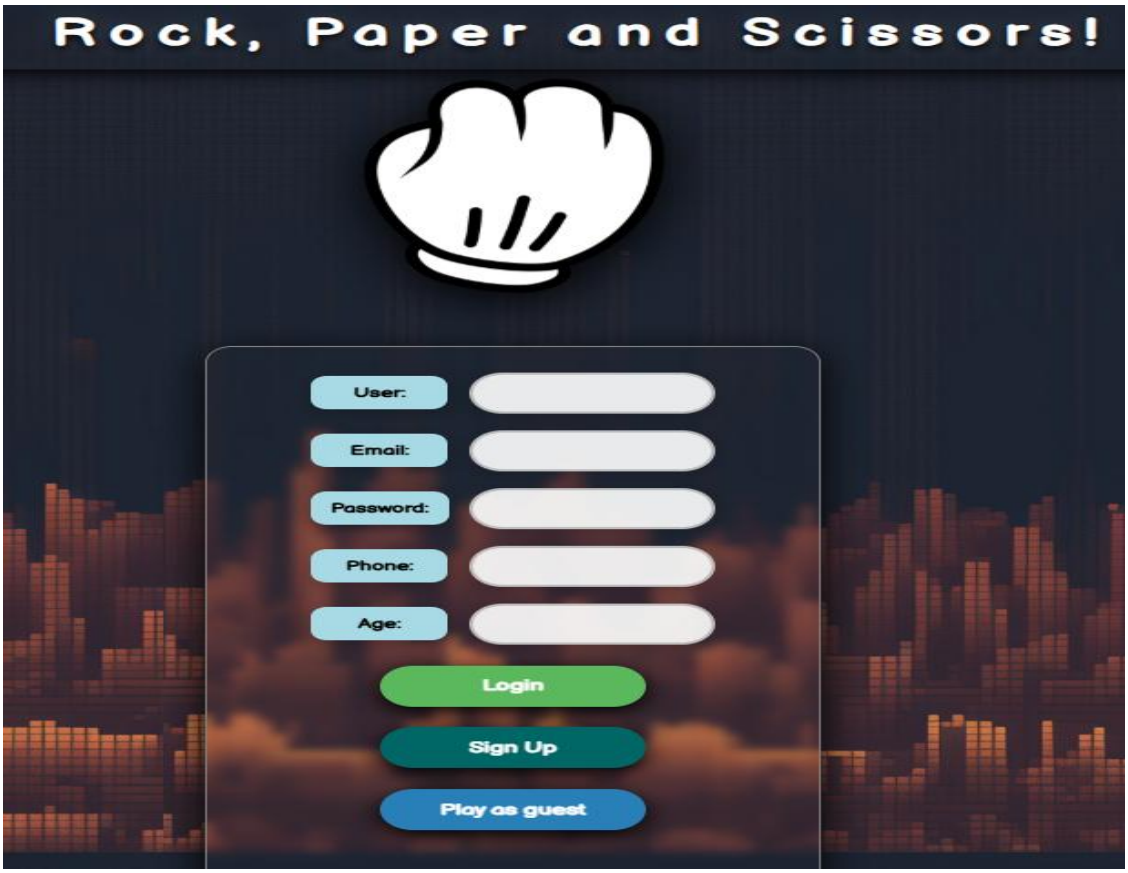
Core Features

1. User Class

To manage the players' account information, we use a User Class which takes username, password, phone, age, and a global score point as its parameters.

- Email-based user identification
- Password protection (6+ characters)
- Age Check (5-100 years) (Figure 1.4)
- Phone number validation (10-15 digits)
- Duplicate username/email/phone prevention

Figure 1.4



Note: The global score point tracks the score required for the leaderboard ranking (more on this soon.)

With User Class, when the player signs up, we validate with what they've provided when they try to log in, with user, email, and password, with user data.

For email and password, I have decided to use basic regex checks rather than manually having a button type for email and password, which, looking back, may have been a more ideal way.

It is also important to note that I have also decided to use the registered email as an index key for the class user for better readability, so that when checking the data on the player, you would only need to call it by the index key (which is the email).

---

2. Player Class

After the user has successfully signed up or logged in, we create a new class called “Player”. Which will take email, username, and the local score point (more about this in the next topic).

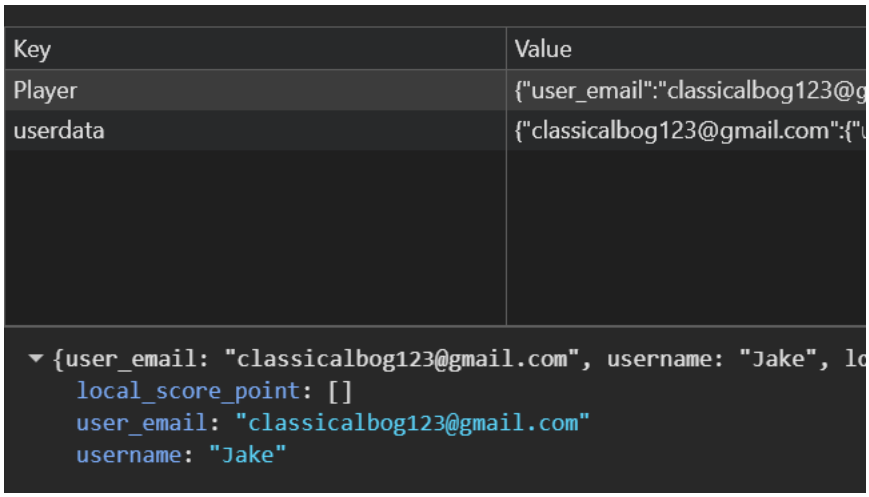
This will be important because it will be used to handle the session scores.

---

2. Local Storage & Session Storage

Working with local storage and session storage has provided me with new and developed insight into how data handling is often done with JavaScript. In this case, once the account has been created, a player class is used with session storage. The game consists of 5 rounds against a bot that will use math. random function to pick a random choice out of the 3: rock, paper, scissors. Whoever has the highest score when the maximum rounds have been reached will win the game, and the score, which is treated as a local score point (Figure 1.5), is then added to the global score point. In case the player wants to end the round early, they can do so by clicking end round, but it will still check for who has the highest points and then store the score if they win.

Figure 1.5



---

3. Game Logic

The game follows with 3 core functions: start round, end round and play round. After initializing everything, we use start round as a logic to begin the match with the press of a button as an event listener, end round, which the user can end the match at any given time (but the highest score will win), with local storage management, and play round, which contains the main logic for the matches.

Core Game Variables:

- PlayerScore and ComputerScore – Track Score
- Round_Number - Current round counter (max 5 rounds)
- Round_Status - Bool to control if the game is active or not.

Bot Choice:

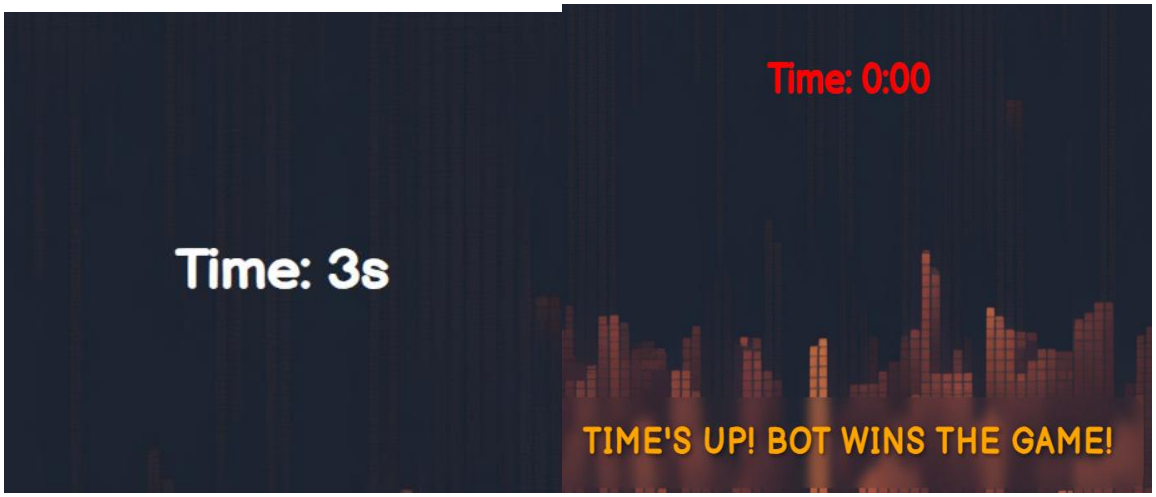
To avoid repeated code, I made the player and the bot share the same choices. Initially, I made it as rock, paper, scissors for the player and c_rock, c_paper, and c_scissors for the bot.

But I found that to be a little tedious and thought that using string concatenation, I could manipulate the computer choices directly without creating new IDs if the IDs shared the same as the player would.

4. Modules - Message Displayer & Timer

Due to time constraints, I was not able to add more of modules and classes. But when it comes to modules, and considering I don't think I could've done much with it other than using it for the game logic, the basic route I took was to use it with message displayer. Initially when I made the game, I had to define the function for message displayer always at the top. But after learning modules, I figured this could reduce redundant code and make the code cleaner. So, while yes, it's possible to go more with modules, given the current stance and time, I have decided to use modules with the message displayer, which is also shared across almost every page in this game.

I also included a module for the timer round, although this was last minute changes, I could not find the proper time to polish it. The way the timer works is, if the player does not make a choice in time, the bot will automatically win by default and the round ends.



5. Leaderboard System

If the account created is true, It takes the global score and then sorts it from highest to lowest to and displays the rankings with the user and the score accumulated.

Challenges

1. Round System Issues

For an unreasonably long time, The game had a critical bug where rounds would not only continue playing beyond the 5 round limit if you spammed the buttons fast enough, but the incrementation would also mess up and often the round would not end until it waited for a event click (since end round is the same function used for the button and the games automatic end). Since I used an event listener for it, I thought the core issue was because of the functionality was shared the same and the logic was messing up. Even after using AI to bug check on what was wrong, I couldn’t wrap my head around it. However, credits are due to my good friends, as they made me realize that the issue was just within the round logic itself and not the function.

These issues probably seem minor, but they made me realize how to deal with failsafes and bools, such as scores not resetting properly and the game state becoming inconsistent.

The game experienced critical bugs where rounds would continue beyond the 5-round limit, scores wouldn't reset properly, and the game state became inconsistent.

2. CSS Styling and Hover Effects

Unironically, the most trouble I really had was with CSS. JavaScript and html were really manageable thanks to the support of my professors and a few hours of learning to understand, but the positioning of the CSS elements and how things are handled, even with the help of AI, I struggled to achieve a good layout,

even with how the UI and design would turn out. Though I did manage to make it work, I feel that I could've done a better job with this.

AI Usage

Debugging

AI was mainly used to help me learn in this project, not just as a developer, but to ensure I use good code practices. Some of its assistance includes syntax errors, logic errors, and sorting with the leaderboard.

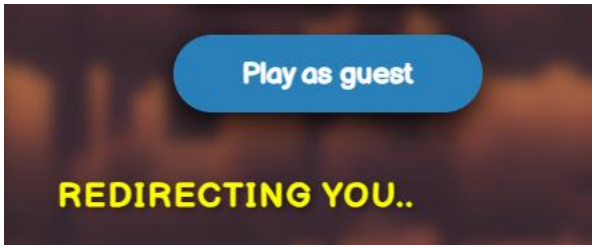
Help with Code Optimisation

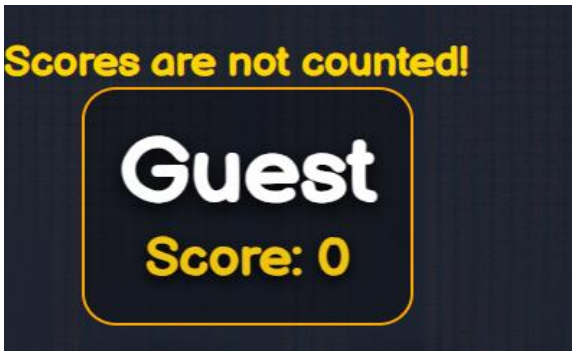
After making the project, I realized and thought of better ways of doing it and optimizing the code. One of the ways actually included the separate buttons of the player and the bot, when I can just use string concatenation to specify the choice, all the while the IDs would remain the same. But this is just an example, More ways were done to optimize the code as the lectures went on, which gave me ideas on how to further improve my skill and the code to the best of my knowledge as of developing the project.

Guest Mode

Incase the player does not want to sign up, they are free to play as a guest too but the score is not counted and will not add up to the leaderboard. (Figure 1.6)

Figure 1.6





HTML Validation

Leaderboard.html validation

Showing results for Leaderboard.html (checked with vnu 25.10.25)

Checker Input

Show ☐ source ☐ outline ☐ image report Options…

Check by file upload Choose File No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Check

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1. **Warning** **Consider adding a `lang` attribute to the `html` start tag to declare the language of this document.**
   From line 1, column 16; to line 2, column 6
   `TYPE html>↩<head>↩`
   For further guidance, consult Declaring the overall language of a page and Choosing language tags.
   If the HTML checker has misidentified the language of this document, please file an issue report or send e-mail to report the problem.
2. **Error** **The character encoding was not declared. Proceeding using `windows-1252`.**

Home_Page HTML Validation

Showing results for Home_Page.html (checked with vnu 25.10.25)

Checker Input

Show ☐ source ☐ outline ☐ image report Options…

Check by file upload Choose File No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Check

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1. **Warning** **Consider adding a `lang` attribute to the `html` start tag to declare the language of this document.**
   From line 1, column 16; to line 2, column 6
   `TYPE html>↩<head>↩`
   For further guidance, consult Declaring the overall language of a page and Choosing language tags.
   If the HTML checker has misidentified the language of this document, please file an issue report or send e-mail to report the problem.
2. **Error** **The character encoding was not declared. Proceeding using `windows-1252`.**
3. **Error** **A `script` element with a `defer` attribute must not have a `type` attribute with the value `module`.**
   From line 50, column 1; to line 50, column 68
   `</div>↩<script type="module" src="../JavaScript_file/Home_PageJS.js" defer></scri`

Game_Page.html validation

Showing results for Game_Page.html (checked with vnu 25.10.25)

Checker Input

Show ☐ source ☐ outline ☐ image report Options…

Check by file upload Choose File No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Check

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1. **Warning** **Consider adding a `lang` attribute to the `html` start tag to declare the language of this document.**
   From line 1, column 16; to line 2, column 6
   `TYPE html>↩<head>↩ <`
   For further guidance, consult Declaring the overall language of a page and Choosing language tags.
   If the HTML checker has misidentified the language of this document, please file an issue report or send e-mail to report the problem.
2. **Error** **The character encoding was not declared. Proceeding using `windows-1252`.**
3. **Warning** **Empty heading.**
   From line 20, column 9; to line 20, column 32
   `>↩ <h1 id = "player_name" ></h1>↩`
4. **Warning** **Empty heading.**
   From line 21, column 9; to line 21, column 32
   `>↩ <h2 id = "player_score"></h2>`
5. **Error** **A `script` element with a `defer` attribute must not have a `type` attribute with the value `module`.**
   From line 55, column 1; to line 55, column 70
   `>↩</div>↩↩<script type = "module" src="../JavaScript_file/Game_PageJS.js" defer></scri`

Conclusion

To conclude, I would like to give proper credits to not only AI for helping me understand the project and the best approaches, but also to my friends and my professors for giving me very clear guidance on the project. With the time constraints, they were reasonably understanding to support not only me but my fellow friends and students whenever we came across a bug or confusion that led us to a block when developing the project. The key point of the project helped me understand more about web applications and databases

Future Improvements

Given the tight schedule, there was a lot more I wanted to add to this simple game. I wanted to add a unique abilities feature where, in case the player was losing to the bot's RNG with a lower score during the match, the player could pick a random ability, like a hint choice, where it would cross out the choice the bot didn't pick and show which of the 2 it has picked. for example, if I have a score of 1 and the bot with 3, I will be given an ability where it'll show me what the bot has picked, so instead of having to guess between rock, paper, and scissors, if the bot picked rock, then it'll show a hint of either rock or scissors. There are still many unique ideas I'd like to add, but I think this is as far as I could've gotten with the project; however, it's taught me a lot about web applications in general.