THE OLD FASHIONED WAY

WHO:

Recitation 012 – Group 07

Ishan Gohil
Erik Hirschmann
Chengming Li
Nathan Mukooba
Kartik Sharma

PROJECT DESCRIPTION:

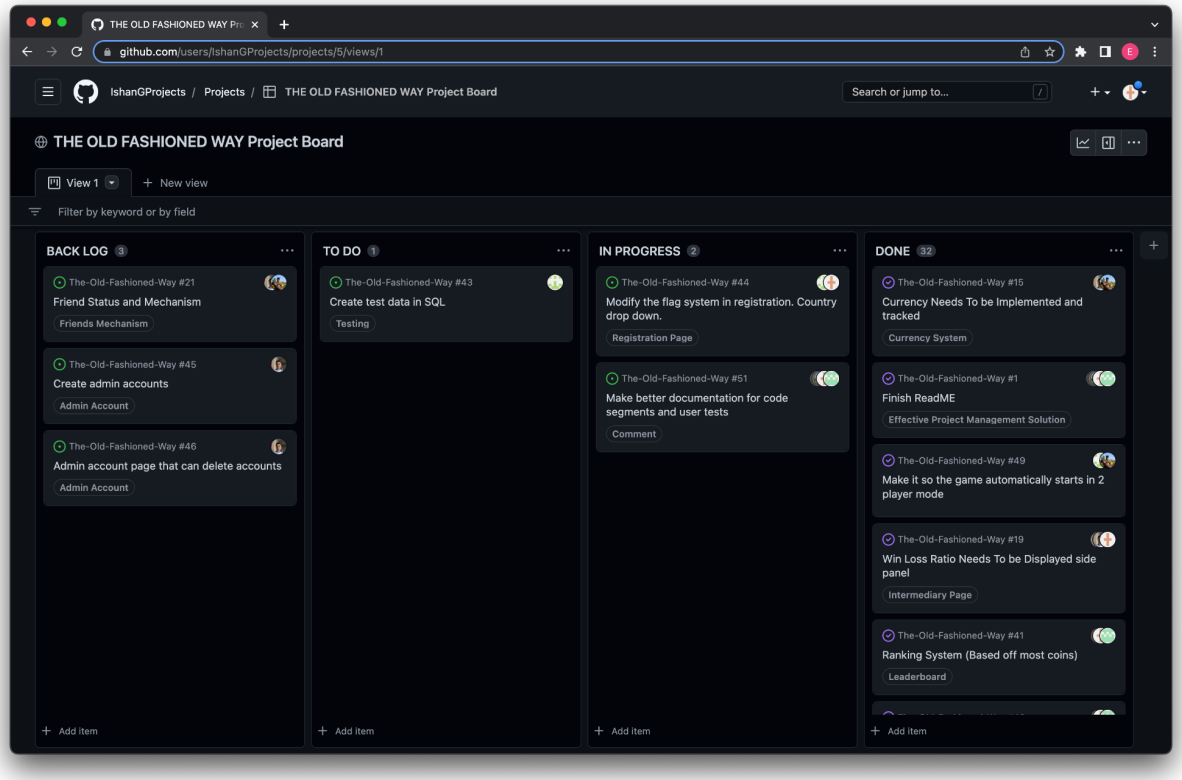
How do you settle a bet? The Old Fashioned Way! Our project provides a lightweight local gaming platform which allows two players to bet on and play the classic video game Pong. The application supports user registration and simultaneous login for two users, allowing for local multiplayer Pong gameplay. We also used NES.CSS LIBRARY to implement a pixel theme throughout the project which can be seen through the labels, text boxes, and buttons. We also customly designed the home page background and text as well.

Users will bet on matches using in-game currency, or “coins.” Additionally, users who earn the most wins or collect the most coins can compare their status with others on the worldwide leaderboard; which can rank all users based on their total wins or their coin balance. The application also supports a customizable profile system, which allows users to add unique profile pictures to their account via url. Ultimately, The Old Fashioned Way is designed as a fun and inviting way to resolve conflicts in a classy and respectable manner.

The Old Fashioned Way was built in Docker Containers, and uses a PostgreSQL database to store its users and user data. The web application was created using the NodeJS application server which allows the database to interact with the UI. The user interface was developed using a combination of EJS and HTML.

PROJECT TRACKER:

[Link to GitHub Project Board]



VIDEO:

[Link to Report Video.mov]

VCS:

[Link to GitHub Repository]

CONTRIBUTIONS:

Ishan Gohil:
My tasks included the Design and Architecture of our platform and database, working on the integration of the PONG game into the platform, going through the testing documentation to debug code, and writing backend database queries and API calls. As the primary architect of the group I designed the database that would be utilized by the group. I wanted to have a 3NF Normalized database for maximum usability and ease of access for the development team. In order to communicate the winning status of the game from the client side Java Script I had to utilize Axios to send data.

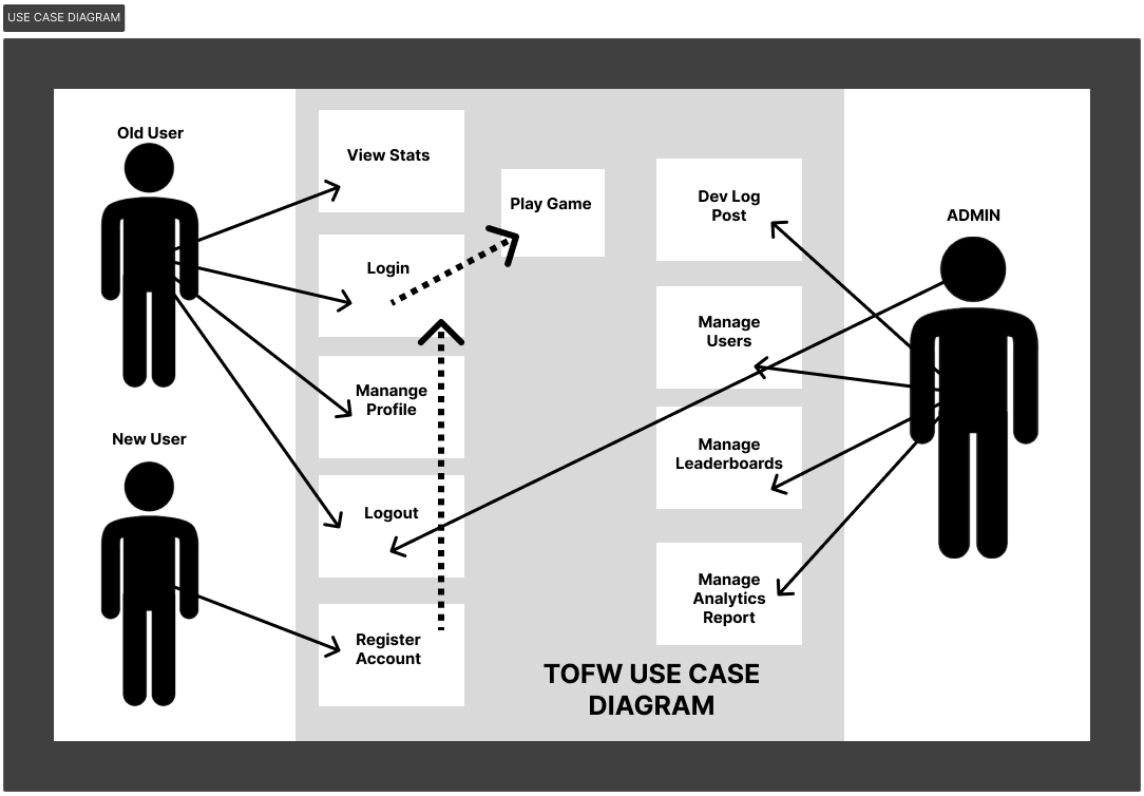
Erik Hirschmann:
My contributions to this project included both front end and back end implementations. I was responsible for using HTML and EJS to develop the user interface of the application—specifically the leaderboard, profile, game and main pages, as well as login, register and betting modals—while making sure that it was visually consistent across the platform. Additionally on the back end, I was responsible for developing the NodeJS functions that allowed for simultaneous login and profile editing, as well as developing the forms for individual login and register functions.

Chengming Li:
I've been working on creating the profile page, main page, and leaderboard with the filter in this project. I also made the NodeJS function for the button linking to the profile and leaderboard pages. And the NodeJS function for the filter button inside of the leaderboard page is created by me. I also worked on writing the test plan. The technologies I've been working on are EJS and NodeJS for software development. Git bash is also used to control the development flow.

Nathan Mukooba:
My contribution to the project pertained to more front end implementations towards the ite we created. I used my understanding of CSS and HTML in terms of developing the style feature for the game. In assistance with other team members I also held responsibilities such as helping with game integration as well as helping better format databases for the game interms of modifying the win to loss ratios with our game. Other aspects of my contributions included assistance with communications in terms of scribing for meetings and organiztions within our repository.

Kartik Sharma:
My tasks were both on the front end and back end, where I created the database, wrote queries to the database with the users information when registering an account and updating the database depending on if the user won a match or changed their profile picture using postgresql. I also worked on NodeJS functions by creating GET and POST by rendering EJS pages with a logged in users information for main page. On the other hand, I worked on the front end by creating the HTML input fields for registering a user and login, the main page, and the leaderboards.

USE CASE DIAGRAM:



TEST RESULTS:

Test Case – Login/Register Page

Register Page:

The user can create a unique identifier that allows him to sign onto the main page. The user needs to input their email address, username, password, and country to create a unique account.

Corner Case: the user will be told to sign into their account if the username has been used and is already defined in the database.

Login Page:

The user can log in to their account and will be led to the main page. The user needs to input the correct username and password to have access.

Corner Case: the user will be told to register an account if the username has not been found or is not defined in the database.

The user will be told to input the correct password for the account if the password does not match the password for this account in the database.

Test Case – Main Page/ Leaderboard Page

Main page:

The user can access the leaderboard page, his profile page, the pong game (two users needed), and the logout function.

The user has access to the leaderboard page by clicking the leaderboard button.
The user can access his profile page by clicking the first user's profile button if only one user has logged in. The second user can also be logged in from the main page by clicking the user2 login button. After the second user has logged in, he also has access to his profile page.
If two users have logged in, they will have access to start the pong game by clicking the start button.
The user can log out from the account by clicking the logout button.

Leaderboard Page:

The user can access the leaderboard page from the main page. The leaderboard page will display the first ten users with the highest wins by default. The name, country, portrait, and total wins will be displayed on the page by default. The user can filter the leaderboard by coins via the COINS button.
If the login user or both users are on the leaderboard, their names will be colored light yellow and dark yellow.
The user can go back to the main page from the leaderboard page by clicking the BACK button.

Test Case – Profile Page

Profile Page:

The user can access the profile page from the main page. On the main page, users can see their portrait, name, country, battle records, and coins.
On the profile page, users can change their portrait by clicking the original portrait on the profile page. The page will be rendered to the "changeUrl" page and ask the user to provide a valid URL. An error will be thrown if the user inputs an invalid URL. On the "changeUrl" page, the user can go back to the profile page if they don't want to change their portrait.

Test Case – Game

Game Page:

Two players are shown in the page (profile images work, Points updated, win loss history)
Separate Control Panels(q,a)(p,l) (must be functional and implemented
HP decrease if one player is attacked by the another,
Portraits of two player
Be able to determine the victor from each game (Higher HP wins)

DEPLOYMENT:

Follow These Steps to Deploy Our Project On Your Local Host:

1. Clone the Repository.
2. Create the .env file inside the mainProject folder following the sample env file guide from the repository.

3. Make sure you have docker installed on your computer and ensure that it is running.
   If you do not have docker you can download it here:
   https://docs.docker.com/get-docker/
4. After that run the command:
   `docker compose up`

5. Then on your browser please type in:
   `http://localhost:3000`