CSC 337 Final Project

Team Members:

1. Zachary Hansen
2. Sameeka Maroli
3. Jordan Demler

Sadly, a group member dropped the class, Alex Roy.

Overview:

Pocket Pond (FKA Fish Kingdom) is a web game that allows players to:

1. Make an account (supported by the code in index.html)
2. Go to the shop (a modal powered by shopScript.js) and purchase their first fish, a small starter fish
3. Pet the fish, feed the fish, and earn coins to buy more fish from the shop to upgrade their aquarium
4. View a leaderboard of the person with the most coins and fish
5. View other users' aquariums!

Players begin with a single little fish and can interact with it through feeding and petting, all while collecting coins along the way. Coins can be used in the games shop to purchase new fish. Larger fish earn more coins, encouraging the user to grow their tanks.

A leaderboard can be opened that displays the users with the most valuable fish. Once players unlock their second fish, they are able to visit the aquariums of the top three users on the leaderboard.

The program has cute sprite work and dynamic animations, including a health indicator to monitor the fish's health, and a video tutorial for beginners.

Frontend:

The frontend is implemented using HTML, CSS, and JavaScript. It includes the following screens and features:

Screens

1. Opening Screen/ Login Screen

        - A welcoming splash screen with animated bubbles and a mer-russ.
        - Options to log in, sign up, or access the help screen.
            - Help option for the login screen explains the game mechanics, including feeding, earning coins, using the shop, and understanding the leaderboard.
2. Fish Tank Screen
        - A cute little aquarium graphic
        - The main gameplay area where users:
            - Feed and pet their fish (to earn coins)
            - View the coin counter and health bar.
3. Shop Screen (an html page powered by shopScript.js)
        - Includes a mer-russ graphic
        - Allows users to buy new fish
4. Leaderboard
        - Shows the top players ranked by the value of their fish.
        - Allows users to visit the aquariums of the top 3 players.

Key Features

- Dynamic DOM Updates: Actions like feeding and petting update the health bar and coin counter without reloading the page.
- Animations: Include basic animations for fish movements and user interactions.

(A single sprite for an unlockable fish in the game store)

Backend

The backend is implemented using NodeJS, Express, MongoDB, and Mongoose. It will handle API requests, user data, and gameplay logic.

NPM Modules

- express: For server and routing.
- mongoose: For database interactions with MongoDB.

- body-parser: For parsing incoming requests.

Routes : The server will handle the following API routes:

1. **User Routes**:
    - POST /api/users/login: Log in an existing user.
    - POST /api/users/signup: Register a new user.
2. **Gameplay Routes**:
    - POST /api/fish/feed: Feed a fish and update its health.
    - POST /api/fish/cuddle: Cuddle a fish and earn coins.
3. **Leaderboard Routes**:
    - GET /api/leaderboard: Fetch top players and their scores.
4. **Trade Routes**:
    - POST /api/trade: Trade fish between two users.

Database Structure

The MongoDB database:

1. **Users**:
    - ➢ username: String
    - ➢ last accessed: String
    - ➢ coins: Number
    - ➢ level: Number
    - ➢ fish: Array of fish models
    - ➢ Decorations: Array of string names of Decorations
- ❖ **Fish**:
    - ➢ id: Unique identifier
    - ➢ name: String
    - ➢ type: String (e.g., small, medium, large)
    - ➢ health: Number(0-2)
    - ➢ beenFed: Number(0-2) I think fish should be fed 3 times before the are full
    - ➢ beenPet: Boolean
    - ➢ isHungry: Boolean
    - ➢ value: Number
    - ➢ accessories: Array of Strings (e.g., hats, sunglasses)
- ❖ **Leaderboard**:
    - ➢ username: String
    - ➢ fishValue: Number (sum of all fish values)

Reflections:

So much went wrong!!! We lost a group member who was responsible for a giant part of the front end, which was getting all the petting, feeding, and coin drop stuff to work. The rest of the group had their respective portions done, when on December 10th we found out the member who went MIA would be dropping the class, and had not been able to work on this section of the front end. After immediately meeting up, we realized getting over 2 weeks of missing work done in 48 hours would not be possible, and we split the work and tried our best to make a functioning web game.

We had to drop a significant amount of the original design, which included a trading feature, and the ability to put hats and accessories on the fish.

Along with this, while getting the server set up we ran into many, many errors. This included CORS errors, favicon errors, and errors due to a member's laptop breaking and being unable to download things like npm.

Overall, this project ended up differently than originally designed. We ran into many problems, but I still believe we all truly expanded our knowledge of HTML, JavaScript, and CSS. With this, Zachary also spent a very large amount of time dealing with a broken server, and getting the server and database to work and interact with the rest of the code correctly (all while working on an unfamiliar laptop), a true MVP.