Open Waters Design Document

Project Name: Open Waters

TNPG: Gone Fishing

Roster: Jady Lei, Ankita Saha, Linda Zheng, Michelle Zhu

TARGET SHIP DATE: 1/17/2024

Program Overview:

Our game will be an ocean themed "choose your own adventure," Oregon-Trail-inspired game. It'll function on a turn-based system where a user must make decisions at different checkpoints within the journey. They will be faced with tasks during in-game days, in which resources (coins, food, etc.) must be managed. There will be a save/load and account mechanic. Games could be loaded based on variables stored under a user in a database table. A maritime weather API will allow for a more unpredictable and interesting twist to the game. The game can be educational too. We plan on using a recipe API to generate related recipes at the end of the game; whichever player makes it to the TBD end wins the game. If the resources are poorly managed, the ship sinks into the ocean and the player loses.

Program Components:

- Homepage
  - Load Saves or New Game
  - Login to access saved games and save games
- Login
  - Needed for users to login so they can access their game, etc.
  - Takes in username and password (sessions)
    - Hash password to the database (SQLite)
    - Requires certain parameters (check user's input for pass w/ if/else statements to verify) to create a password (length, characters, etc.)
    - If the user is not logged in, they should not be able to view anything; the user must be registered first
  - Usernames have to be unique
- Game
  - Using a marine weather api, flavor text is generated for the player who then makes (a) choice(s), increasing the day until the voyage is over and the game is done.
    - For example, if the waves are against them, no progress is made.
- Map
  - If we can get one to work, an updating map of the ship on its course would be cool. Worst case scenario it can just be a progress bar.
    - Updates when a user saves their progress (goes up to place user was last at)

        - If progress bar, uses locations of the courses to rack progress(if user gets up to a certain location on the map that is closer to the finish, progress bar goes up)
- Leaderboard
    - Ranks voyages by the length user has taken OR time taken for user to get across
        - Ex: The voyage is the same for all users, but if one user took a longer route than others, they are lower on the leaderboard.

APIS:

- Marine Weather API: returns wave height and wave direction to create a more realistic and random weather cycle in the game.
    - Need to make a card for this api. Free daily rate of 10,000 calls.
- Spoonacular: will hopefully be able to return fish related recipes to be shown throughout the game.
- Joke API: We can show a joke that contains relevant words such as ocean or fish at loading points, in the menu, or during a task.
- Fact API: If we dont show a joke, we’ll show an education fact related to the location, obstacle, or anything else in the user’s journey
- Google Font: Monospace pixel font used for the website.

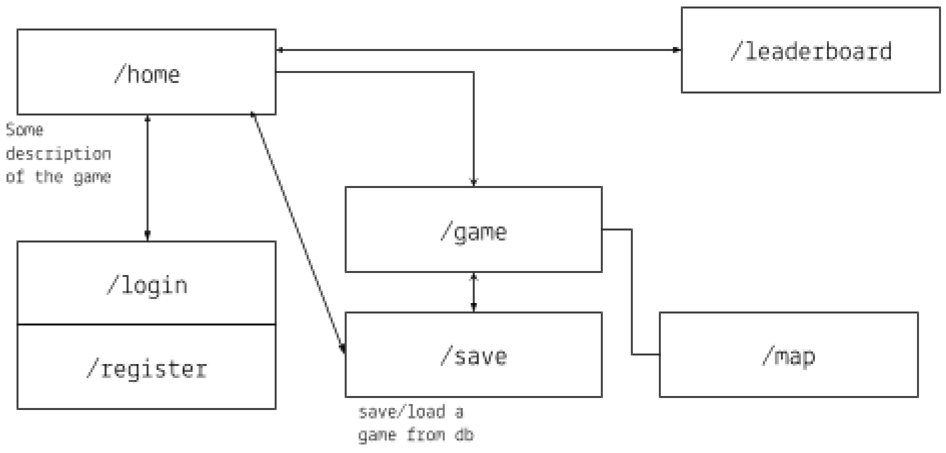
FEF:

Bootstrap: We decided to use this because it will fit best for our project components, especially those that require specific stylistic changes (color, border, layout, etc.), and is easily understandable.

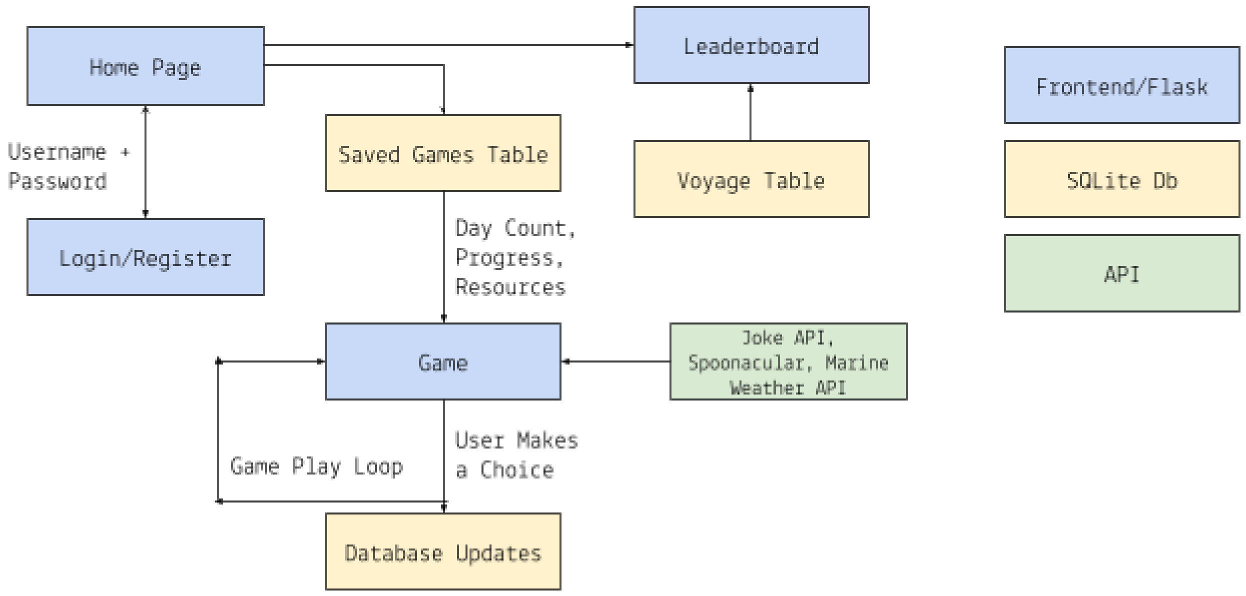
Database Organization:

- Users
    - Username - text
    - Password - text
    - Games - int
- Game Saves
    - Username - text
    - Day - int
    - Food - text
    - Money - int
    - Progress - int
    - Crew Mood - text
- CompletedJourneys
    - Username - text
    - VoyageLength - int

Site Map:



Component Map:



Task Breakdown:

- Jady - Project Manager & Flask
    - Oversee progress on each component and sets the pace for each component(communicate readily about deadlines, ensure each function, page, etc. is being done in orderly and timely manner)
    - Provides any additional help to any component that requires it.
    - Set up Flask routes and Game loop
- Linda - FEF & API
    - Create functions to access API and save data from API into the databases.

    - Work on incorporating API into frontend, including displaying information from each API into corresponding elements of the game.
- Michelle - FEF & API
    - Create functions to access API and save data from API into the databases.
    - Work on incorporating joke API and fact API into frontend, including displaying information from each API into corresponding elements of the game.
- Ankita - FEF & DB
    - Functions for the flask app to access/update the database
        - User Database: helper functions for inputting and accessing usernames and passwords.
        - Game Saves: store game components as user progresses (updateGame())
        - Completed Journeys → lengths complete, leaderboard, updateScore
    - Work on CSS for finished components (Bootstrap)