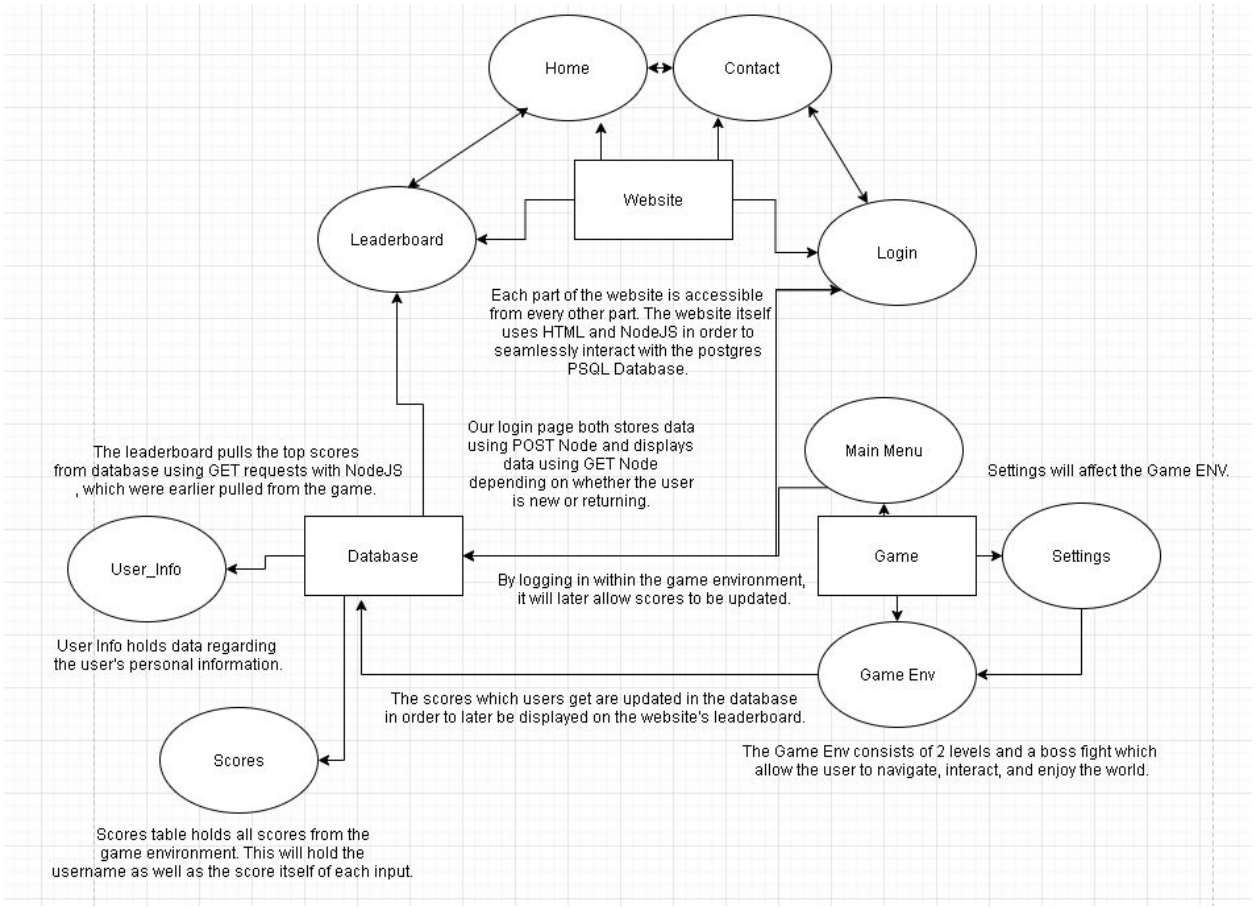
Revised List of Features

- Game Popups - These would include a respawn screen when the character dies, a beat level screen, and a beat game screen.
    - **Priority: 4**
    - "As an experienced player, I want to be able to see graphics displaying respawn options and level completion so that I can understand how much progress I have made in the game."
    - Functional Requirements
        - The game popups display a graphic when the player dies and gives the option for respawning or exiting the game
        - The game popups display a graphic when a player beats a level
        - The game popups display a graphic when a player finishes the game
    - Non-functional Requirements
        - The game popups display the FPS of the game.
- Back End Game Storage - Back end character storage would allow the user to save the game state and their character stats / location for use at a later time. (This will most likely end up being hosted online.)
    - **Priority: 1**
    - "As an experienced player, I want to be able to save my game state so that I can load my progress in the future."
    - "As an experienced player, I want to be able to see top scores from other players on a leaderboard so that I can compare my scores to their scores."
    - Functional Requirements
        - The game saves a previous game state as directed by the user, including the current level and any customized character traits, in the back end technology stack
        - Top scores are saved to a global leaderboard
        - Back end has to use server (can not be local)
    - Non-functional Requirements
        - When the player chooses to save their game, the game saves necessary data in under 5 seconds
        - Provide multiple saves for a single player
- Loading Procedures - These procedures would use the back end character storage to link the back end and front end while allowing the user to load a previous game state.
    - **Priority: 2**
    - "As an experienced player, I want to be able to load a previous game state so that I can continue playing with my previous progress."
    - Functional Requirements

        - The game loads a previous game state as directed by the user through the integration between the back end and front end technology stacks
        - Current leaderboard stats are also loaded and can be viewed by the experienced player
    - Non-functional Requirements
        - If there is no previous file to load, this should not be possible
        - The procedures load necessary data in under 5 seconds
        - The procedures load a history of saves.
- Main Menu to Game connection - This includes allowing each button in the main menu to connect to the game itself, a previous game state or settings, and for the player to be able to leave the game and go to the menu.
    - **Priority: 3**
    - "As an experienced player, I want to be able to switch between my game state and the main menu."
    - Functional Requirements
        - The player can click a button in-game that prompts them if they would like to access settings or the main menu, or if they would like to return to the game.
    - Non-functional Requirements
        - The popup should stop the game from being played while open.
- Game Environment
    - **Priority: 0 (Functional requirements complete)**
    - "As an experienced player, I want to be able to see my game environment through graphics so that I can play the game."
    - Functional Requirements
        - The back end game storage loads level layouts and the front end technology stack displays graphics for the level
        - The character sprite changes depending on what the player is doing (moving, interacting, etc.)
    - Non-functional Requirements
        - The game environment loads in under 5-10 seconds
        - More than two levels (not including boss)
- Character Interactions
    - **Priority: 0 (Functional requirements complete)**
    - "As an experienced player, I want to be able to move my character so that I can play the game."
    - Functional Requirements
        - The player should be able to move their character using the keyboard.
        - The player should be able to interact with the game environment.
    - Non-functional Requirements

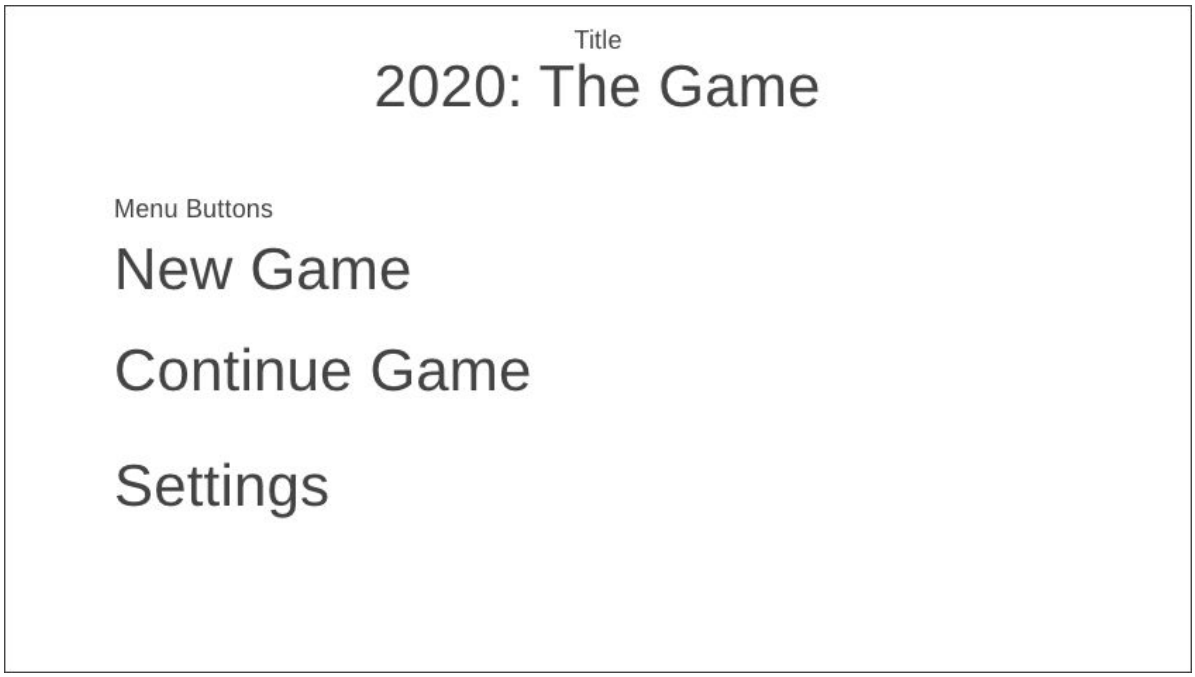
- A player is able to interact with other players.

Architecture Diagram



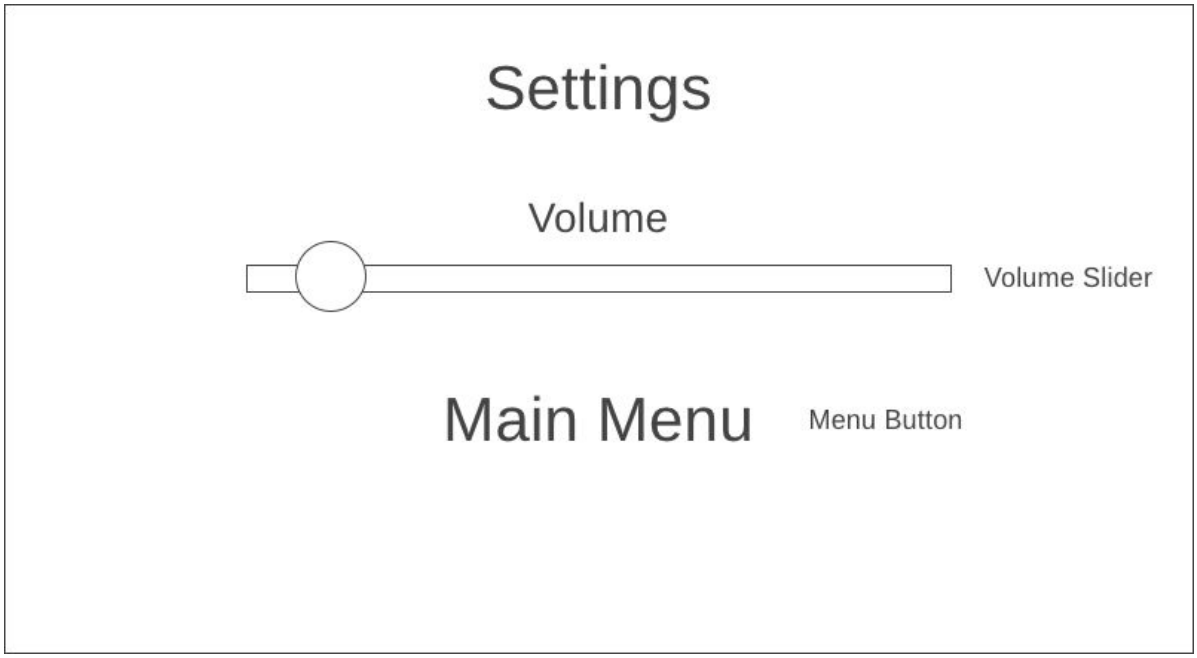
Front End Design

Main Menu Diagram



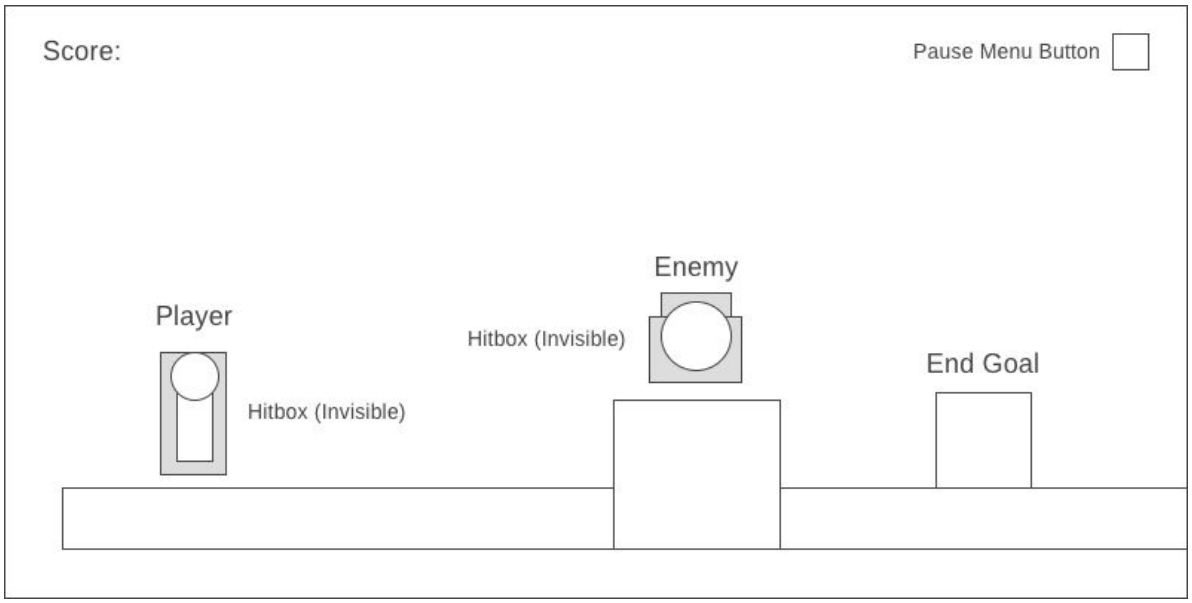
https://wireframe.cc/qsVXsu

Settings Diagram



https://wireframe.cc/nBV9Di

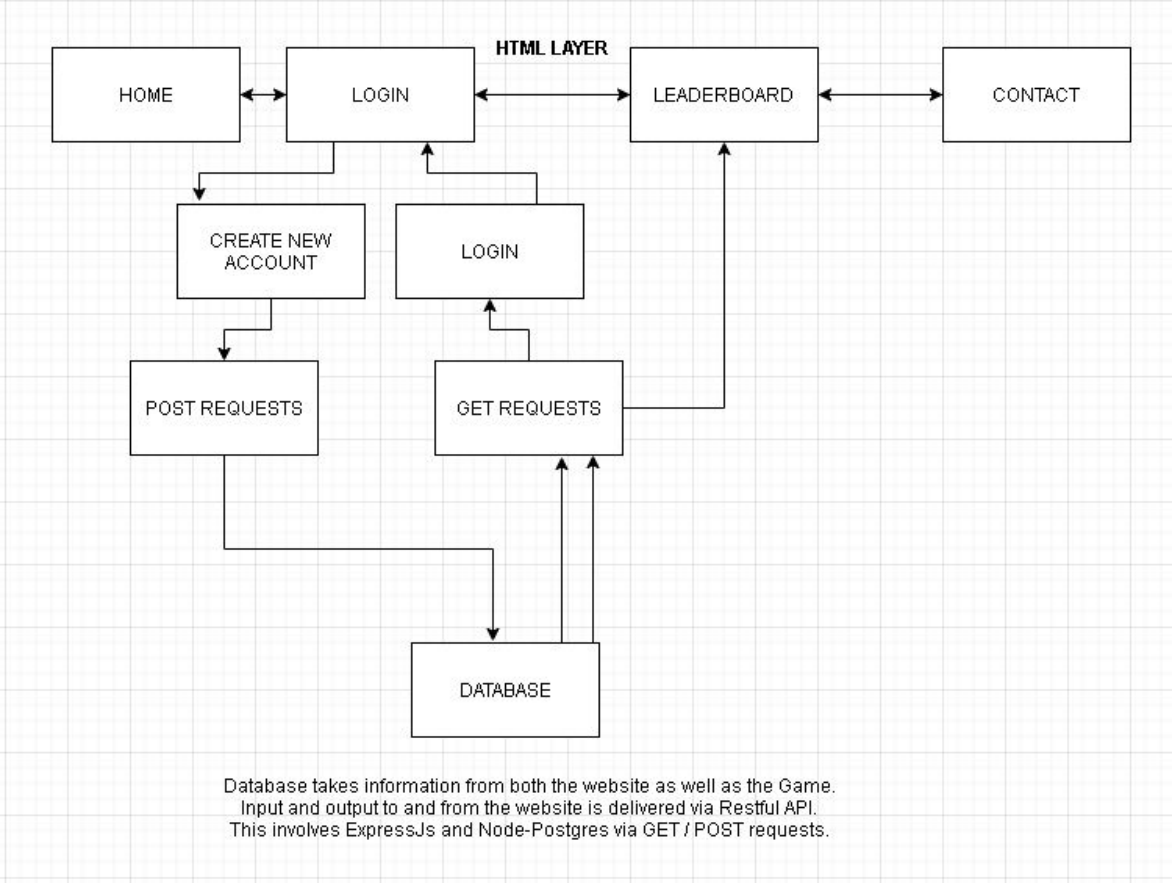
Game Environment Diagram



https://wireframe.cc/XrQLs5

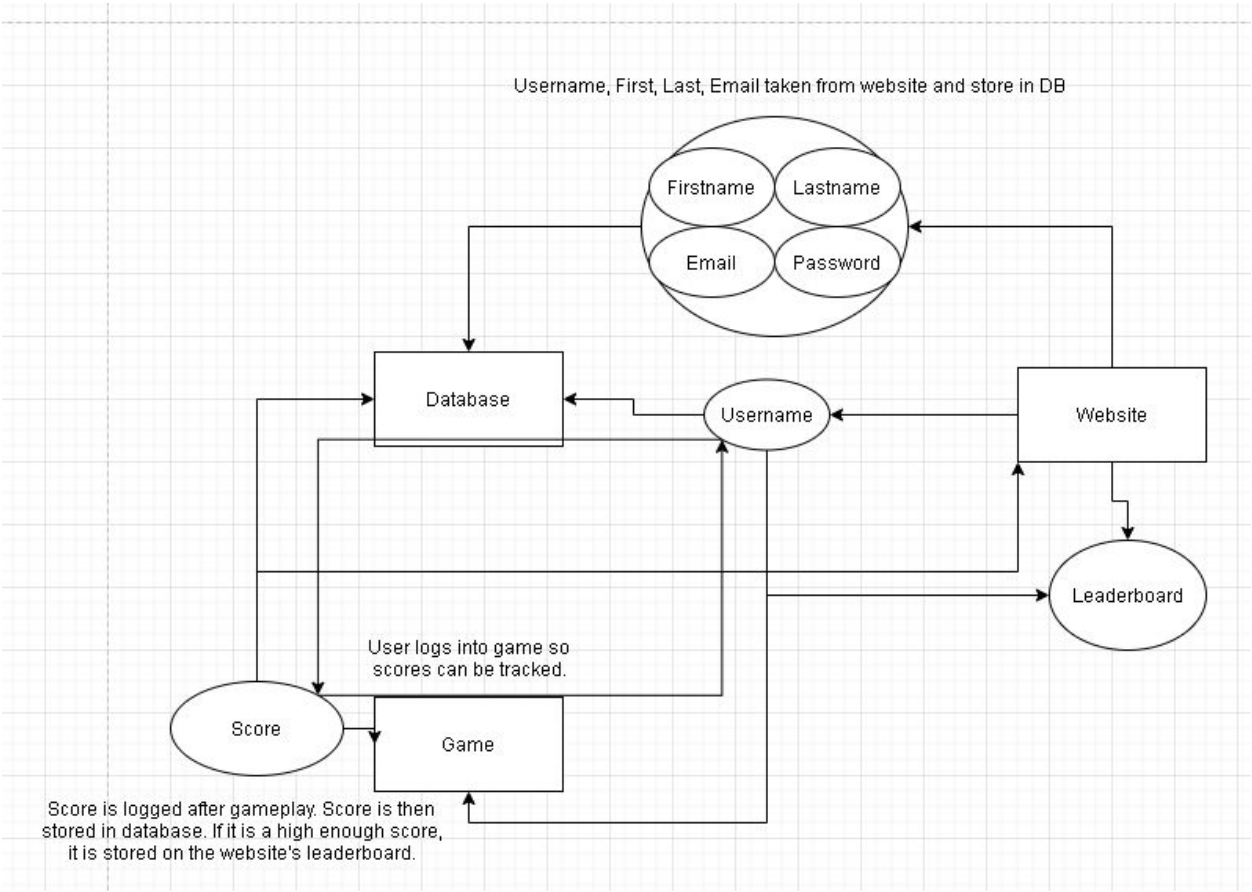
Web Service Design

- HTML
- Postgres PSQL
- Restful API
- Express JS
- Node-Postgres
- The database acts as a middle man between the game and the website. The website allows users to create accounts and login, as well as view the leaderboard. The database stores user information such as username and scores which are then logged in the game's leaderboard.



Database Design

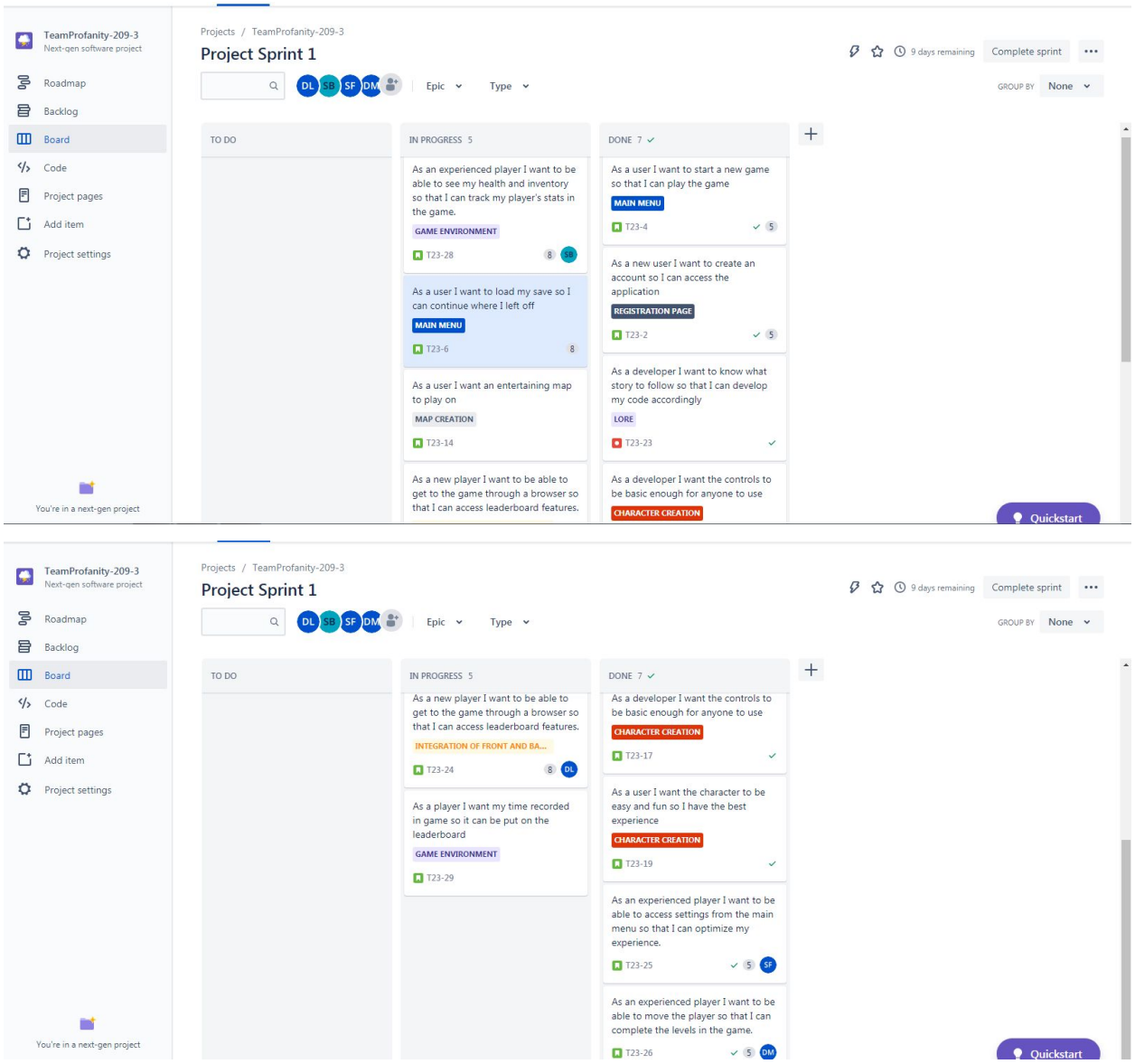
- Postgres PSQL
- Restful API
- Express JS
- Node-Postgres
- Our database uses postgreSQL. This holds our data involving registration and login information in a table called user_info. This table holds the first and last name, username, and email of the visitor. Another table will soon be implemented which will hold a username as well as their best performance (time) in the game thus far. GET and POST are used to transfer data between the website itself and the database.



Individual Contributions

Github Commits

- Mac - Switched database and web formatting to RESTFUL API (nodeJS) from php. Created diagrams / summaries for milestone4.
- Dan - Worked on new tile map and final layout for levels
- Sam B - Worked on developing a score counter in the game along with a few other game features
- Sam F - Researched ways to script in-game audio, worked on Milestone 4 writeup
- Dom - Worked on Revised List of Features of Milestone 4, researched how to connect database to website.
- Matt - Worked on Front End Design wireframes, Challenges, and Revised List of Features part of Milestone 4. Starting working on implementing score and time functionality in game



Link since screenshots are poor quality:
https://teamprofanity.atlassian.net/jira/software/projects/T23/boards/1

Challenges

- We struggled with the initial integration of our website to our database. After using php and postgres, we decided to switch (with some input) to NodeJS and postgres.
    - Our backup/risk mitigation plan is to not use a local database and have our database on a hosted server.
- One future risk is that there won't be an effective way to connect the ingame scores generated by players to the online leaderboard via a database.

    - To mitigate this risk, those on the backend and game design teams should communicate to see how scores in game should be implemented in order to connect them to the leaderboard.
- One future challenge is that the game may become too monotonous after the player completes a number of levels.
    - To prevent this, each level layout should be unique, with new kinds of enemies and challenges possibly being added in with each level to make gameplay more interesting.