QA BUG TEST PLAN

**Test Plan**

Objective: To thoroughly test the game and identify any bugs or issues because bugs suck and no one likes em.

Scope

The testing will focus on the functionality and user interface of the game. It will cover features such as drawing random bots, choosing a player duo, initiating duels, tracking win/loss count, and the shuffle functionality. The testing will be performed manually in a local development environment.

Test Environment

- Operating System: Windows 10

- Browser: Google Chrome, Mozilla Firefox, Edge because I’m edgy like that.

- Node.js:

- Dependencies: Express, Jest, Nodemon, BrowserSync

- Code Editor: VS

Testing Approach

The testing will follow a manual testing approach. Test cases will be executed manually to simulate user interactions and verify the expected behavior of the game.

Test Scenarios

1. Player Duel Outcome

2. Shuffle Functionality

3. Shuffle Endpoint Accessibility

**Test Cases**

**Test Case 1: Player Duel Outcome**

Description: Test the outcome of a player duel and record the win/loss count.

Test Steps:

1. Start a new game and draw five random bots.

2. Choose two bots to form the player's duo.

3. Initiate a duel against the computer duo.

4. Observe the duel outcome (win or loss) and the displayed message.

5. Check the updated win/loss count in the player's record.

Expected Result:

- The win/loss count should be incremented correctly based on the duel outcome.

- The displayed message should reflect the correct outcome (e.g., "You won!" or "You lost!").

**Test Case 2: Shuffle Functionality**

Description: Test the functionality of the shuffle feature in the `/api/robots/shuffled` endpoint handler.

Test Steps:

1. Send a GET request to the `/api/robots/shuffled` endpoint.

2. Retrieve the response and examine the order of the bots.

Expected Result

- The response should contain a shuffled order of bots, showing that the shuffle functionality is working as expected.

**Test Case 3: Shuffle Endpoint Accessibility**

Description:Test the accessibility of the `/api/robots/shuffled` endpoint.

Test Steps:

1. Send a GET request to the `/api/robots/shuffled` endpoint.

2. Check the HTTP status code of the response.

Expected Result:

- The response should have a status code of 200, indicating that the endpoint is accessible.

**Bug Report**

Bug Report 1: Incorrect Incrementing of Player Wins/Losses

Description: The player's win/loss count is incorrectly incremented in the `/api/duel` endpoint handler.

Steps to Reproduce:

1. Start a new game and draw five random bots.

2. Choose two bots to form the player's duo.

3. Initiate a duel against the computer duo.

4. Observe the duel outcome and the displayed message.

5. Check the updated win/loss count in the player's record.

Actual Result:

- The win/loss count is goes up for both wins and losses, resulting in incorrect tracking of player statistics.

Expected Result:

- The win/loss count should be incremented correctly based on the duel outcome (wins increment wins, losses increment losses).

Code to fix

if (compHealth > playerHealth) {

playerRecord.losses += 1;

res.status(200).send("You lost!");

} else {

playerRecord.wins += 1;

res.status(200).send("You won!");

}

Bug Report 2: Shuffle Functionality Not Working

Description: The shuffle functionality in the `/api/robots/shuffled` endpoint handler is

not shuffling the bots correctly.

Steps to Reproduce:

1. Send a GET request to the `/api/robots/shuffled` endpoint.

2. Retrieve the response and examine the order of the bots.

Actual Result:

- The order of the bots in the response is not shuffled, showing that the shuffle functionality is not working as expected.

Expected Result:

- The response should contain a shuffled order of bots, indicating that the shuffle functionality is working correctly.

Bug Report 3: Typo in Variable Name in `/api/robots` Endpoint Handler

Description: There is a typo in the variable name used in the `/api/robots` endpoint handler, causing an error when the endpoint is accessed.

Steps to Reproduce:

1. Send a GET request to the `/api/robots` endpoint.

2. Check the HTTP status code of the response.

Actual Result:

- The server throws an error when accessing the `/api/robots` endpoint due to the incorrect variable name.

Expected Result:

- The response should have a status code of 200, indicating that the endpoint is accessible without any errors.