Julian Panici

CIS 611

Final Project

Requirements

1. Startup / Design

a. User Interface

i. Initially show a popup window which explains the rules of the game, along with giving credit to Immaculate Grid, which is the inspiration for the project.

ii. Create a 4x4 game board which is centered in the middle of the screen, where the top left box holds the number of guesses remaining, the top row and furthest left column each hold 3 random teams chosen when the game is started, and the middle 9 boxes hold clickable buttons.

iii. Create a logo in the top left corner which displays the game logo and title of the game “Footy Grid,” along with a button in the top right corner which displays a github logo and when clicked takes the user to the github profile which holds the game’s repository.

b. Game Startup

i. Run a function which initializes and starts the game. This includes:

1. Initializing number of guesses to 9 and number of correct guesses to 0.
2. Choose 6 random teams to be displayed, not allowing duplicates. Display the corresponding team images chosen from the images folder.
3. Each team has a unique team id. Create a list containing the team id combination for each square (ex. The top left square will hold the team ids’ for the corresponding row and column it resides in).

2. Game Logic

a. Guessing

i. When a user clicks a box, display a popup window which allows the user to search for a corresponding player name. Include a “Search Player” button which will send the string entered by the user to the api which holds the player database.

b. Searching
   i. When the user clicks the “submit guess” button, the string entered is sent to the api, which queries the player database and returns a list of json objects which hold the data of players whose name contains the search string.
   ii. Display another popup window which holds a scrollable select box holding the names of each player in the list of search results, along with a button which allows the user to submit their guess.

c. Submitting Guess
   i. Once the user chooses a player and clicks the “submit guess” button, store the chosen player’s json object data into a variable.
   ii. Retrieve the player’s team ids, and cross reference them with the team ids of the chosen box in order to determine whether the chosen player has been a member of both teams.

d. Correct Guess
   i. If the chosen box’s team ids are both found within the player’s team ids, the guess is correct. Do the following:
      1. Increment the number of correct guesses by 1.
      2. From the player’s json object, access the url which holds his headshot, and display that on the chosen box/button.
      3. Disable the chosen box/button.
      4. Decrement the number of guesses remaining by 1.

e. Incorrect Guess

i. If the chosen box's team ids are not both found within the player's team ids, the guess is incorrect. In this case, only decrement the number of guesses remaining by 1.

3. Win Conditions / Reset
    a. After each guess, if the number of guesses remaining is 0, the game has ended. Check the number of correct guesses and do the following depending on how many there are:
        i. Win State
            1. If the number of correct guesses is 9, the player has won the game. Display a popup window with a message congratulating the player on winning. Also include a "Reset" button which allows the player to restart the game.
    b. Lose State
        i. If the number of correct guesses is less than 9, the player has lost the game. Display a popup window with a message consoling the player for losing. Also include a "Reset" button which allows the player to restart the game.
    c. Reset
        i. Upon clicking the "Reset" button, restart the game by running the functions used in the "Game Startup" portion of the requirements. This will do the following:
            1. Create a new game board with 6 new teams.
            2. Reset the number of guesses remaining to 9.
            3. Reset the number of correct guesses to 0.