Lab 3 (Oct. 15) #Team1

Sprint-3:

Retrospective meeting: any updates since last meeting? any issues?

Planning meeting:

- what are the fields of interest to search?
- breakdown the goals into the actionable tasks
- define task completeness criteria (test cases if needed)

Feature:

- Search player database (using specific attributes)

- Insert players(player name, unique ID) (TO DO sprint 3)

- Delete players(player name, unique ID) (TO DO sprint 3)
- Modify players(player rating, age, team, position) (TO DO sprint 3)
- Interactive world map listing players by nationality
- Average rating of all players in each club (player name, Overall rating, club)

- fields of interests to search

-Search by player name, nationality, position, overall rating, age, potential growth, player's team (if any.)

GUI design and User test cases:

- for testing the search feature

Test backend search function, where user inputs player name and our database then finds player.

Taskboard:

Done list of last sprint 2:

- Parsed csv file using our own python Parser.
Evan/Abraham
- Converted data into a Pandas DataFrame and displayed that on our search.html.
Evan/Mustafa
- Added 4 HTML pages and CSS for our web application (Homepage, Search, Map, ratings and Add/delete)
Hoda/Calvin

ToDo task list for the next sprint

TODO LIST KEY: 🌭 = Must be done during sprint 3. 😴 = For future sprints

BackEnd Tasks:

- 🌭 Create search function for each attribute in database i.e name, player id, nationality, team, etc
    - Function will take in attribute to search for and the search entry ex: search('name', 'Lionel Messi')
    - Function will return dataframe (which can be converted to html)
    - Acceptance Criteria: function returns row of data in the form of a dataframe
- 🌭 Create python database class to manage and manipulate database
    - Class will include python pandas import
        - Acceptance Criteria: pandas is imported without compiler errors
    - Class will include reference to newly created file to edit (so we can avoid changing fifa csv file)
        - Acceptance Criteria: file variable exists and can be called upon
    - Class will include search, add, modify, and delete functions
        - Acceptance Criteria: functions are callable from other classes/python files
    - Class will store all current data in mysite/fifa/fifaCS180.txt
        - Acceptance Criteria: text file is appropriately named and accessible from other classes/python files
- 🌭 Create database function to add new db entry
    - Acceptance Criteria: fifaCS180.txt adds new row after function call
- 🌭 Create database function to modify existing entry
    - Function will take in player id argument along with all other elements
    - All soccer player elements changed or unchanged should be sent to database
    - Acceptance Criteria: function changes existing row according to arguments on same line
- 🌭 Create database function to delete existing entry
    - Function will take in player id as argument and remove row accordingly
    - Acceptance Criteria: fifaCS180.txt no longer contains row with passed in player id
- 🌭 Create render functions to communicate with frontend
    - Acceptance Criteria: Intended data is sent and received to and from forms
- 😴 Create map and populate with player names and nationality
    - Acceptance Criteria: Map entity displays with all players
- 😴 Create average rating calculation function
    - Acceptance Criteria: Returns list of teams with rating values for each team

FrontEnd Tasks:

-
- 🌭 Create frontend form for search
    - Acceptance criteria: looks pretty, clicking on the button receives data from database, confirms update in backend, displays search result.
-

- 🌭Create frontend form for add
    - Acceptance criteria: looks pretty, clicking on the button receives data from database, confirms update in backend.
-
- 🌭Create frontend form for modify
    - Acceptance criteria: looks pretty, clicking on the button receives data from database, confirms update in backend.
-
- 🌭Create dropdown
    - Acceptance criteria: looks pretty, ll attributes available and selecting each attribute updates form label
-
- 🌭Form validation for add
    - Acceptance criteria: creates alert when some elements are not filled out/ not valid
-
- 🌭Form validation for modify
    - Acceptance criteria: creates alert when some elements are not filled out/ not valid
- 🌭Create frontend form for ratings
    - Acceptance criteria: looks pretty, clicking on the button receives data from database, confirms update in backend.
- 😴Form css
    - Acceptance criteria: looks pretty, perfectly centered, shadowing to create elevation
- 😴Create shared nav bar view
    - Acceptance criteria: nav bar works on all pages of the website, button links take you to each page
- 😴Fix button spacing
    - Acceptance criteria: button text is perfectly centered
- 😴Mobile view/desktop view
    - Acceptance criteria: Website looks pretty when the screen size is shrunk to less than 800px and no functionality of the website is hidden
- 😴Mobile view button
    - Acceptance criteria: Hamburger menu icon, drop down links display when clicked, click anywhere else on the page and the dropdown closes

Test Case:

- **Feature 1 (search): As a user I want to be able to search the Database, filtered by different attributes (name, nationality, team, player id).**
    - **Test case 1: User enters first and last name of player: 'Lionel Messi'**
        - **Correct output: Website displays player Lionel Messi with all his attributes**
    - **Test case 2: User enters nationality to search a list of players from that country 'portugal'**
        - **Correct output: List of players from that country with attribute of each player**
    - **Test case 3: User enters incorrect information filled with non alphanumeric characters**

        - **Correct output: The page throws an error that tells the user of incorrect input**

- **Feature 2 (Insert): As a user I want to be able to add a new player to the Database with the information needed (name, nationality, team, player id).**
    - **Test case 1: User enters all the necessary information of a player into the website**
        - **Correct output: Website successfully takes in new information and updates the database with the player**
    - **Test case 2: User enters incomplete information about a player (eg. missing age, rating, etc.)**
        - **Correct output: Website throws an error requiring the user to enter all fields with information**
    - **Test case 3: User inputs a non alphanumeric character in any of the fields**
        - **Correct output: An error is thrown to notify the user that an illegal character has been inputted for information**

- **Feature 3 (modify): User want to update the players' information and keep the data up to date.(team, rating, age, position)**
    - **Test case 1: Users want to change the age for one of the players in the data. Users will find the players who need to be changed. Then,click the modify button and select the age. The textbox will show up and users are required to enter the correct age in the textbox and click the submit button.**
        - **Correct output: Server will be updated and show the age that users changed.**
    - **Test case 2: Users want to change the position for one of the players in the data. Users will find the players who need to be changed. Then,click the modify button and select the position. The dropdown menu will show up and users are required to select the position from the menu and click the submit button.**
        - **Correct output: Server will be updated and show the position that users changed.**
    - **Test case 3: Users want to change the rating for one of the players in the data. Users will find the players who need to be changed. Then,click the modify button and select the age. The textbox will show up and users are required to enter the correct rating in the textbox and click the submit button.**
        - **Correct output: Server will be updated and show the rating that users changed.**

- **Feature 4 (World map): We want to display a world map that indicates where players are from each country**
    - **Test case 1: Display all players and country of origin**
        - **Correct output: World map is populated with 'points or dots' that shows each country and how many players are from there. Example Portugal has 14 players, so there will be 14 points on portugal.**
    - **Test case 2: Country with no players eg (Monaco)**
        - **Correct output: Monaco map will be displayed with no points as fifa 21 has not included any players in their database.**

- **Feature 5 (Average): User want to checkout the average rating for each club**
    - **Test case 1: Users want to know the average rating for Club A. Users will type the Club name in the text box. Then,click the average button.**
        - **Correct output: Server will bring to user and show all the player list from the Club that users choose and the bottom of the list will show the average rating.**

- **Feature 6 (Delete): User want to delete a player from Database**
    - **Test Case1: Player enters ‘Lionel Messi’ to delete him from database**
        - **Correct Output: Player ‘Lionel Messi’ and all his attributes are deleted from Database**
    - **Test Case 2: User enters incorrect information filled with non alphanumeric characters**
        - **Correct output: The page throws an error that tells the user of incorrect input**
    - **Test Case 3: User enters a player name that does not exist (eg ‘John Cena’)**
        - **Correct output: ‘Player does not exist in database’**