Lab 6 - Team-42

Ford St. John, Eduardo Rocha, Robert Arenas, Kyle Dean

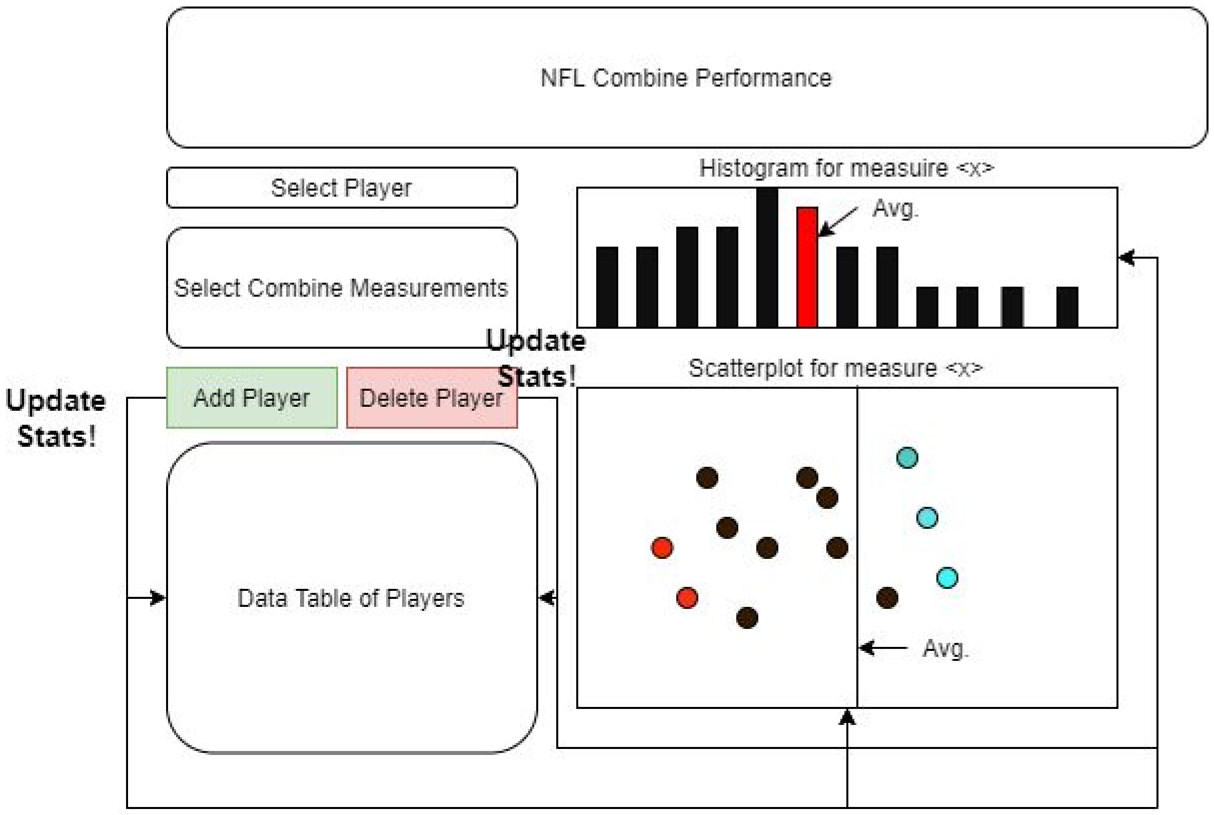
Features (user stories) to Implement in Next Sprint:

1. Combine
    a. Feature 1: As a user, I want to be able to add a player with particular combine performance statistics and have that player's new statistics show up in the aggregate performance statistics
    b. Feature 2: As a user, I want to be able to delete a player by name, and have that player's performance records removed from the data and no longer show up in the aggregate performance statistics

2. Rushing
    a. Feature 3: As a user, I want to be able to see an increase in 'all time' page performance .
    b. Feature 4: As a user, I want to be able to delete a player from the rushing statistics if he exists already.

3. Receiving
    a. Feature 5: as a user i want to add a receiving play/player to the receiver data and recompute statistics.
    b. Feature 6: as a user i want to delete a receiving play/player to the receiver data and recompute statistics.

4. Passing
    a. Feature 7: as a user, I want to be able to add a player to the 'passer' data and have that player's new statistics show up in the aggregate passing statistics.
    b. Feature 8: as a user, I want to be able to delete a player from the 'passer' data and have that player's passing records removed from the data and no longer show up in the aggregate passing statistics.



GUI also applies to Rushers, Receiving, and Passers application

Test Cases

1. Feature 1 test cases
    a. Test Case 1: Filter on combine year = 2017, position = Wide Receiver, measurement = “40-yard dash”. Add a new player with whatever name you want, and with a 40-yard dash time somewhere in the “3s” (e.g. 3.5, 3.4, etc.).
        i. Correct output: mean 40-yard dash time drops significantly, timing benchmark to compute the new statistics (mean, standard deviation, outliers) should take ~ 1 microsecond to run
    b. Test Case 2: Filter on combine year = 2017, position = Wide Receiver, measurement = “40-yard dash”. Add a new player with whatever name you want, and with a 40-yard dash time somewhere in the “6s” (e.g. 6.1, 6.2, etc.).
        i. Correct output: mean 40-yard dash time should increase significantly, and timing benchmark to compute the new statistics should be ~ 1 microsecond (vs. 55 ms on average)

2. Feature 2 test cases
    a. Test Case 1: Delete “Tom Brady” from the dataset, and review the 2000 QB 40-yard dash time

        i. <u>Correct output:</u> the mean 40-yard dash time should drop a few points, and the benchmark computation time for the statistics should be ~1 microsecond
    b. Test Case 2: Delete John Ross from the dataset. Review the 2017 Wide Receiver 40-yard dash times
        i. <u>Correct output:</u> the mean 40-yard dash time should increase a bit, and the benchmark time to compute the new statistics should be ~1 microsecond

3. Feature 3 test cases
    a. Test Case 1: As a user, I want to see incremental speed in the way a page loads when I click 'all time' players..
        i. <u>Correct Output:</u> When the user clicks 'all time' players for top rushers, the load speed is faster for the second time around.

4. Feature 4 test cases
    a. Test Case 1: As a user, I want to delete Frank Gore from being the top rusher because I'm tired of seeing him as the top rusher in the NFL
        i. <u>Correct Output:</u> When the user deletes Frank Gore as a player, he no longer pops up on the top rushing statistics page anymore.

5. Feature 5 test cases
    a. Test Case 1: As a user, I want to add a new player to receiving data
        i. <u>Correct Output:</u> player is added to receiving data store
    b. Test Case 2: As a user, I want to display updates to receiving stats, for a searched player, faster than having to re-calculate already present data.
        i. <u>Correct Output:</u> The time taken to compute additional receiving plays for a searched player is significantly less than the time taken to first look up the player and their stats.
    c. Test Case 3: User attempts to add a player that already exists
        i. <u>Correct Output:</u> message displays saying the player cannot be added because he already exists.

6. Feature 6 test cases
    a. Test Case 1: As a user, I want to delete a new player to rushing data
        i. <u>Correct output:</u> player is removed to receiving data store
    b. Test Case 2: User attempts to delete a player that does not exists
        i. <u>Correct Output:</u> message displays saying the player cannot be removed because they do not exist in the data store.

7. Feature 7 test cases

    a. Test Case 1: as a user on the passers statistics page, when I search a player using the search form they do not exist in the ‘passer’ data, I can add that player with particular passing data using an ‘add’ button.
        i. <u>Correct output:</u> when the player is added, their data is accounted for in the top n passing statistics (total, average, etc.).
    b. Test Case 2: as a user on the passers statistics page and after I used the top n form and used the ‘add’ button, the time to show the top n analytics should be decreased since analytics were calculated previously.
        i. <u>Correct output:</u> the passing page shows that the previous time to correctly output the top n analytics was greater than after using the ‘add’ button.

8. Feature 8 test cases
    a. Test Case 1: as a user on the passers statistics page, when I search a player (i.e. Tom Brady) using the search form and they already exist in the ‘passer’ data, I can delete Tom Brady using a ‘delete’ button.
        i. <u>Correct output:</u> when the player is removed, their data is not accounted for in the top n passing statistics (total, average, etc.).
    b. Test Case 2: as a user on the passers statistics page and after I used the top n form and used the ‘delete’ button, the time to show the top n analytics should be decreased since analytics were calculated previously.
        i. <u>Correct output:</u> the passing page shows that the previous time to correctly output the top n analytics was greater than after using the ‘delete’ button.

TODO List

<u>Done list of last sprint (week of 11/06 - 11/12)</u>

1. Implemented frontend histogram visual that reflects user selections on the combine page
   [Finished by Ford St. John, verified by team]
2. Implemented frontend scatterplot visuals that show selected combine performance against player weight and player height, based on filter criteria selected by the client on the combine page
   [Finished by Ford St. John, verified by team]
3. Updated combine page HTML file to correctly render plots, slightly change the page layout
   [Finished by Ford St. John, verified by team]
4. Implemented backend code that creates the histogram plot and converts the created image to HTML code to be rendered on the frontend
   [Finished by Ford St. John, verified by team]
5. Implemented backend code that creates the scatterplots and converts the created image to HTML code to be rendered on the frontend

[Finished by Ford St. John, verified by team]

6. Added scatter plot comparing the average receiving yards per play and total receiving plays [Completed by Robert Arenas]
7. Implemented average receiving yards per play for individual and top n plates pages [Completed by Robert Arenas]
8. Edited Html of top n receiving page to show scatter plot above top n players table. [Completed by Robert Arenas]
9. Implemented the average yards per play for the top 20 rushers in the NFL. [Completed by Eduardo]
10. Implemented a graph using plotly for visualization of stats on the Rushing page. [Completed by Eduardo]
11. Implemented graph of where the top 20 rushers stand against each other. (total rushing yards vs. total rushing plays)
    [Completed by Eduardo]
12. Implemented total times passed for each top n player in the passing statistics page. [Completed by Kyle Dean, verified by team]
13. Implemented average passing length for each top n players in the passing statistics page.
    [Completed by Kyle Dean, verified by team]
14. Implemented a bar graph in the passing statistics page to visually show top n passing statistics by total passing yards.
    [Completed by Kyle Dean, verified by team]
15. Implemented a scatter plot in the passing statistics page to visually show the relationship between total passing length and total passing times for each top n player.
    [Completed by Kyle Dean, verified by team]
16. Successfully updated the HTML for the passing statistics page to properly display the bar graph and scatter plot.
    [Completed by Kyle Dean, verified by team]

<u>ToDo task list for next sprint:</u>

1. Implement frontend “Add Player” and “Delete Player” buttons to the combine page so that clients can either add a new player or delete a player (if that player exists)
2. Implement backend functionality to check if the “Add Player” or “Delete Player” buttons were pressed, check the data passed from the forms, and either add the new player or delete the selected player from the dataset accordingly
3. Implement backend functionality to ensure performance statistics are correctly aggregated with the new player considered, or the existing player deleted
4. Implement “Add Player” for the Rushing statistics page that will add a player to the dataset if they do not exist.
5. Implement “Delete Player” for the Rushing statistics page that will delete a player if he exists.

6. Implement a form filter so that the user can select which team to get the top rushers from.
7. Implement add player/receiving play from receiver data.
8. Implement delete player/receiving play from receiver data.
9. Implement the “Add Player” button for the Passing statistics page that will add a player to the dataset if they do not exist.
10. Implement the “Delete Player” button for the Passing statistics page that will delete a player if he exists.