Journal Comments

Tommy:

- Created a search page, which would then be called in the App.js
- Issue with dynamically fetching data from API
    - I resolved it by creating an array to pass the data to inorder to access it
- Issue with getting image of player from the initial API we had chosen
    - We solved the issue by fetching the image from another API
- Fetched the players physical data and put it into a separate cell under the player's picture.
- After getting the functionality that I wanted, I did some styling around on the search page to make it look simple and easy to view for users.
- Added a simple footer component that has the github linked.
- Created a react chart component which is called within the search page
- With the chart a component I had to figure out a way to pass the data from the search page
    - to solve the issue I utilized "states and props", the data that was fetched in the search page was then passed as a prop in the chart component
    - With the data from the API taking form of an array holding an object I had to change the way the data was formed so I took the data and put it into an array inorder to pass it as a prop which was then used to make the graph function properly
- With the chart working properly some of the styling that was made previously was not accounting for the chart so some adjustments were made
- Created the simple developers cards as a final touch to make the website more personalized
- We had a merge problem in the beginning since I didn't consider making small incremental pushes and so we had to reconfigure our repos and spent a good amount of time trying to figure out source control stuff.

Cesar:

- Created Navbar and pages for all of pages and features we were gonna implement
- Styled Navbar
- Issue with getting image of player from the initial API we had chosen
    - We solved the issue by fetching the image from another API
- First task after Navbar was working on the comparison page. In the comparison page the user searches two nba players and it compares the stats for both, highlighting higher scores in green and lower scores in red.
- Created the table which will hold all the data surrounding compare and created specific table td elements for each stat to display.
- I ended up using another API to grab the NBA player headshots since the API that we were using did not have that on hand. This was fairly straight forward, using each players unique name and ID I passed those variables to the headshot API which returned the correct headshot for each player. The headshot API though only has headshot for players in the 2018-2019 season and not new players that have joined since then.

Journal Comments

- Issue with having the user enter a second player name before the first player which wouldnt populate the array correctly. Solved this by not displaying the second button until the user enters the first player name.
- Used the NBA API to get daily games and ticket links to display on the daily schedule. No API had team logos that was working for the structure that I had in place so I had to manually grab the links for each NBA teams logo and store them into an array then based off of the TRIcode of each NBA Team (ex. Portland Trail Blazers TRIcode would be POR) it would loop though that array and once a match was found it would return that image url of the logo to be displayed into the daily schedule.
- Styling was handled after the compare and daily schedule were working on barebones table.

Links that helped me out:

1. https://javascript.plainenglish.io/creating-a-dynamic-html-table-through-javascript-f554fba376cf
2. https://dev.to/dcodeyt/creating-beautiful-html-tables-with-css-428l