Unit Tests

Alfred Lam:

- Bar Chart - createBarChart()
    - Display
        - The colors of the bars should match the colors in the legend, based on a player's team
            - Tested by passing in different mock team data with different colors to the visualization
        - The height of a bar should correspond to the selected statistic of a player
            - Tested by passing in mock statistics for players to the visualization, and ensuring the values were correct.
    - Interaction
        - Clicking one of the leagues should make the correct teams show up
            - Tested with real data by clicking on all 4 leagues and checking that the correct teams show up.
        - Clicking one of the team selections should toggle that team's players on/off on the bar chart
            - Tested by verifying that every team can show up in chart on click, and cumulative clicks result in multiple teams showing.
        - Clicking one of the statistics should make that statistic show up on the bar chart
            - Check by injecting mock statistics in the data, and ensuring the values were correctly displaying.
- Parallel Coordinates Line Chart - createLineChart()
    - Display
        - The colors of the lines should match the colors in the legend, based on a player's team
            - Tested by passing in different mock team data with different colors
        - The position of the line on each axis should correspond to that statistic of the player
            - Tested by using real data, and ensuring that the correct data was being displayed
    - Interaction
        - User should be able to select/highlight lines based on a statistic, and only the selected statistics should show up colored.
            - Tested by interacting with the chart and ensuring that selection of lines would make those lines highlighted.

Gus Person:

- Mongodb - Flask Connection:
    - List of names stored in MongoDB as {“name”: [name]}
    - Query the list of names to printed out all names which are stored in mongodb.
    - Display names in an HTML page, verify that each name is listed on the HTML page.
    - This should run upon opening the server page (endpoint is ‘/’).
- Login System:
    - Registration:
        - POST request should send user input field data to flask.
        - Flask hashes the password and stores it within mongodb as a bytes literal (b’*hash here*’).
        - Return error if username is already in database.
        - Verify hashing with variety of unicode characters for edge cases(“”, <,>.?/!@#$%^&*()_+-=) and all lower/uppercase characters and numbers.
        - Verify no duplicates statement by registering same username twice.
    - Login:
        - POST request should send user input field data to flask.
        - Output POST request status code, should return “200: OK”.
        - Flask requests the data from MongoDB using username to retrieve hashed password, returns “User not found!” if username does not exist.
        - Input usernames that are not stored within MongoDB to verify.
        - Input correct and incorrect user/password combos with information entered from Registration to verify login function, should output “success!” if successful, return “false!” and output the retrieved password hash, the POSTed password hash with the corresponding usernames to the console.
- Scrape player backgrounds:
    - geturl() should return an assortment of different working URLs that access individual player pages from different rows in the table.
        - Run an assert true mock object that checks if the url in question is the format below: 'https://lol.gamepedia.com/{PlayerName}
        - Run an assert true mock object that checks if the table retrieved from the url contains {‘class’: ‘infobox InfoPlayer’} HTML object.
        - Verify that retrieved username matches with player username from LOLesports database within MongoDB.

Jake Hwang:

- Scrape Stats:
  - geturl()
    - should return an assortment of different working URLs that access tables from different series.
    - Run an assert true mock object that checks if the url in question is the format below: 'https://lol.gamepedia.com/{-}/{-}_Season/{-}_Season/Player_Statistics'
  - getdf()
    - should return a Pandas df of exactly 19 columns and a non-zero amount of rows.
    - Run a stub that inputs fake data into the df and checks if the shape of the dataframe asserts true to the following conditionals: `df.shape[0] > 0 and df.shape[1] == 19`
  - __main__()
    - should push the df into the mongoDB database, regardless of file type
    - Run a test stub that inputs fake indirect input data into a temporary data frame and insert a test spy that checks for all the data loaded into the database
- Compare Page:
  - League and player input
    - should be able to access data and flow into the correct color object in visualizations
    - Run a test spy that console outputs the data that is trying to be accessed
  - Graphs
    - should be able to react to changes in the league and player input and display statistics
    - Insert a test stub that changes indirect inputs into the visualizations class periodically and see the graph animate to reflect player statistics changing

Michael Hsieh:

- Data Page
    - Table
        - Test that the table can render by giving the column prop in React Table an array of objects where each object contains a Header and accessor.
        - Once the table can render with the correct Headers for each column, add in the correct headers and accessors corresponding to the data in the database.
        - Input the data into the table by passing it to React Table as props
        - Test if data is being retrieved properly by opening sidebar, selecting a league, and observing if the table is being filled with the correct data
    - Individual Player Cards
        - Test that the cards can render by first inputting filler data for where the data should go
        - Click on player info tab to observe that the cards are rendering properly
        - Once the cards can render properly, the data can be retrieved and inputted into the card
    - Sidebar
        - Test that the sidebar can render by passing the component a list of regular buttons
        - Test the sizing of the sidebar so that it does not cover too much of the page by adjusting the width in the css
        - Once sidebar can render with the correct size, the actual button with images corresponding to each league can be implemented
    - Autocomplete search bar
        - Test that search bar can render by passing in an array of objects with dummy values
        - Test that the search bar filters out the list based on inputted text by searching for one of the dummy values
        - Test that selecting a value in the search bar console logs the object that is selected by going to inspect/console and observing outputs
        - Once the autocomplete search bar works for dummy values, list of players can be inputted as the array of objects

Perry Yang:

- Server - Flask Route Endpoints:
    - /<tourney>/players => get_players(tourney) to get all players' statistics in the selected tourney
    - /<tourney>/teams => get_teams(tourney) to get all teams in the selected tourney
    - /<tourney> => get_tourney(tourney) to retrieve the whole league in the selected tourney
    - /<tourney>/<player> => find_one(tourney, player) to retrieve single player's performance in the selected tourney
    - /<league>/<player_name>/page => get_player(league, player_name) to retrieve a players information for Individual Card
    - How to Test: used Postman to manually test each endpoint with random tourneys and player names. Verify Postman response with data in MongoDB
- Header Icon:
    - When clicking the header icon, it directs to the homepage.
    - How to Test: manually click on the icon to see if it jumps to the homepage. If yes, the icon is functional.
- Data Page:
    - Table:
        - Completed the name of each column and send GET requests to retrieve data from MongoDB
        - How to Test: on Data page, select league, and click on Table tab, data of the selected league are displayed as stored in MongoDB
    - Individual Player Cards:
        - Sent requests from React to Flask to retrieve an individual's information from MongoDB
        - How to Test: select a random player each time, and then click Card tab, observe if the card display the information of that player as stored in MongoDB
- Login:
    - Login page:
        - Implement and render the login page
        - How to Test: Login page is rendered containing imperative input spaces to fill in and buttons to click on; type in usernames and passwords and click Sign In, observe if Flask returns the correct status.
    - Register page:

- Implement and render the register page
- How to Test: Login page is rendered containing imperative input spaces to fill in and buttons to click on; type in usernames and passwords and click Register, observe MongoDB if the username and password exist.