Team 1:

Uveimar Sandoval: usandova@gmu.edu

Farith Bascope: fbascope@gmu.edu

Daniel Curtis: dvurtis7@gmu.edu

Rajeep Karki: rkarki6@gmu.edu

Sammy Miller: smille77@gmu.edu

Omar Naseem: [onaseem@gmu.edu](mailto:onaseem@gmu.edu)

Joshua Reyes: jreyes36@gmu.edu

October 15, 2023

CS 321: Software Engineering

Professor Wes Masri

GTA: Bhargavi Janga

Project Deliverable 3 – API & Unit Testing

**Methods and Classes**

**Methods**:

* extract\_RosterData(url):
  + Extracts roster from specific NFL team.
  + Param 1: url -> url of team roster
  + Returns: multi-dimensional array for each player in specific team.
  + Return format: [[name, position, age, height],…]
* extract\_PlayerStats(url)
  + Extracts players stats based on position.
  + Param 1: url -> url of full team stat breakdown.
  + Returns: multi-dimensional array of player stats.
  + Return format: [[playerName, yards, tackles, receptions],…]
* insertRoster\_DB()
  + Inserts roster data into MySQL database.
  + No return.
* insertStats\_DB()
  + Inserts player stats into MySQL database.
  + No return.
* queryPositions(position)
  + Queries database for players with designated position.
  + Param 1: position -> designated position to query for.
  + Return: Array of players with designated position.
* queryPlayer(player)
  + Queries database for the stats of designated player.
  + Param 1: player -> name of player to retrieve stats for.
  + Returns: Array with players stats or None if player not found.

**Classes:**

* Parser: parses web for team and player data.
* Database: creates, inserts, and queries data from database.
* Main: Back-End application of website.

**The following are the Pytest methods that we will use:**

**extract\_RosterData()**

* test\_first()
  + This method is for displaying the first player's name from the NFL database.
  + Gets player’s name using the SQL query: "SELECT Name FROM NFL."
  + Expected "Josh Allen."
* test\_second()
  + Method to display a list of players and verify the index of "Kyle Allen."
  + To get a list of player names using the SQL query: "SELECT Name FROM NFL."
  + Checks if "Kyle Allen" is at index 1 in the result list.
* test\_many()
  + Method to display list of players and verify the index of "Josh Allen" and "Shane Buechele."
  + Gets a list of player names using the SQL query: "SELECT Name FROM NFL."
  + Checks if "Josh Allen" is at index 0 and "Shane Buechele" is at index 2 in the result list.
* test\_defense()
  + This method corresponds to displaying a list of players and verifying the player at index 35.
  + It retrieves a list of player names using the SQL query: "SELECT Name FROM NFL."
  + It checks if the player at index 35 is "Kameron Cline."
* test\_specialTeams()
  + This method is to displaying a list of players and verifying the player at index 70.
  + Gets list of player names using the SQL query: "SELECT Name FROM NFL."
  + Checks if the player at index 70 is "Tyler Bass."

The following test cases confirm extract\_RosterData() is functional with multiple NFL teams.

* test\_teamMIA()
  + - Method to test the data extraction for the Miami Dolphins team.
    - It calls the **extract\_RosterData** function with the URL for the Miami Dolphins' roster.
    - It then asserts that the first item in the extracted data list matches the expected value: "Tua Tagovailoa QB 25 6' 1"
* test\_teamMIADef()
  + - Method corresponds to testing the data extraction for the Miami Dolphins' defense.
    - It calls the **extract\_RosterData** function with the URL for the Miami Dolphins' roster.
    - It asserts that the 41st item in the extracted data list matches the expected value: "Emmanuel Ogbah DE 29 6' 4".
* test\_teamNE(), test\_teamNEDef(), test\_teamNYJ(), test\_teamNYJDef, test\_teamBAL(), test\_teamBALDef()
  + - Methods to test data extraction for other NFL teams (e.g., New England Patriots, New York Jets, Baltimore Ravens) in a similar manner to above methods like **test\_teamMIA** and **test\_teamMIADef**.
    - These methods test different teams and positions by calling the **extract\_RosterData** function with the appropriate URLs and asserting the extracted data.
* test\_teamCIN(), test\_teamCINDef()
  + - Methods to test data extraction for the Cincinnati Bengals and their defense.
    - They follow the same pattern as the other test methods by calling the **extract\_RosterData** function with the respective URLs and asserting the extracted data.

**extract\_PlayerStats()**

* test\_playerOne()
  + Method to retrieve a players stats from the website
  + Tests that the received data is in the format
  + [James Cook, 507, 1, 21]
* test\_playerTwo()
  + Method to retrieve players stats from the website
  + Tests that the received data is in the format
  + [Damien Harris, 462, 0, 17]
* test\_playerThree()
  + Method to retrieve players stats from the website
  + Tests that the received data is in the format
  + [Ty Johnson, 160, 2, 12]

**queryPositions()**

* test\_position()
  + Method to display a player's position based on the SQL query: "SELECT Name, Position FROM NFL."
  + Gets the player's name and position.
  + Checks if the position is "QB" and asserts True or False.
* test\_QB()
  + Method to display a list of players with the position "QB."
  + Gets player names using the SQL query: "SELECT Name FROM NFL WHERE position ='QB'."
  + Checks if the list of players with the "QB" position matches the expected list: ["Josh Allen", "Kyle Allen", "Shane Buechele"].

**queryPlayer()**

* test\_Josh()
  + Method that will retrieve the data that is stored on Josh Allen
  + Querys’ the database using the SQL Query “SELECT \* FROM Stats WHERE name=”Josh Allen”
  + Asserts that the data is [762, 3, 0]
* test\_Kyle()
  + Method that will retrieve the data that is stored on Josh Allen
  + Querys’ the database using the SQL Query “SELECT \* FROM Stats WHERE name=”Kyle Allen”
  + Asserts that the data is [13, 0, 0]

**Web Application Implementation**

* The web application API will be generated by Flask.
* The web application consists of an HTML, CSS, and JavaScript frontend:
  + HTML defines the structure of the web page, including elements for displaying player information.
  + The CSS (styles.css) provides styling for the web page.
  + The JavaScript (script.js) handles the behavior of the web page, displaying player information.

**Contributions**

* Project Leader: Sammy Miller - 30%
  + Miller oversaw the formatting and coding of Pytest cases.
* Task Distribution:
  + Coding: Uveimar Sandoval - 20%
  + Coding: Farith Bascope - 15%
  + Documentation: Rajeep Karki - 20%
  + Front End research: 15%
    - Daniel Curtis
    - Omar Naseem
    - Joshua Reyes