**SportsInfoAPI**

**Part I:**

The objective of this initiative is to create a streamlined method for retrieving detailed information about sports clubs or players, particularly in Major League Baseball (MLB) and the National Basketball Association (NBA). When users input a specific name or identifier, the system will furnish a plethora of information, including the latest statistics, recent achievements, status, and historical records. This solution aims to be versatile, employing APIs, web scraping, and other suitable data collection methods to ensure comprehensive coverage. This also means that the API can expand upon as new data sources become available. Critical evaluation metrics for this project include user interface simplicity, data retrieval speed, and the accuracy of the information provided. One of the most exciting applications is in the realm of Fantasy Sports. Users can leverage the data provided by your API to make informed decisions about player selection and team management. It can also be used to develop Sports Analytics Platforms, providing insights into player performance, team dynamics, and game outcomes, which can be invaluable to coaches, players, and sports enthusiasts. can power Sports Betting Applications, providing the necessary data for users to make educated bets.

A platform that comes close to the concept you’re describing is StatMuse. StatMuse focuses on sports data and leverages natural language processing to answer complex sports-related queries. While ESPN is a popular sports platform, it often lacks certain information, leading users to seek additional sources. This is where StatMuse shines. It aims to provide a one-stop solution for sports enthusiasts, offering a wide range of data from various sports leagues, including the NBA, MLB, and more.

To create this api, there are 7 tasks that need to be done. Task one is to decide what data that will be used to create these queries. The project will be for MLB and NBA stats. The second task that needs to be decided is the method of data collection. For MLB data, we can use API like pybaseball or the data files from Retrosheet.org. For the NBA, we can use nba\_api to get statistics. Data scrapers can be used to get more data or expand the scope of these api. If we decide to use data scrapers, the subtask will be to determine what website we will scrap from. The data retrieval system is the third task that would need to be done. We need to figure out what format the data will be collected in. The api will also need to ensure that when new data sources become available,they can be integrated. The fourth task is to develop the UI. Looking at how sites like baseball reference or ESPN would allow for ideas on what to do. The system needs to work with queries that allow for better stats lookups. The fifth task will be to create the functions that will do the queries that the users want to get. The sixth task is to conduct a massive amount of testing to ensure that the correct data is being retrieved and formatted correctly. Speed testing will also need to be done to make sure queries are done efficiently. The final task will be to document the project. Provide information on any challenges and solutions. There will also need to be instructions on how to use it.

The evaluation of this project will done by using the time python library to see how long it takes for each query is run and compare it to statmuse.com

The weekly plan for the project is as follows:

In the first week, the project scope and requirements will be defined, and the development environment will be set up. The second week will be dedicated to implementing the data fetching functions and writing unit tests for these functions. In the third week, the focus will shift to implementing the data analysis functions and writing unit tests for these functions. The fourth week will see the implementation of the data presentation functions and the beginning of integration testing. The fifth week will continue with integration testing and the start of performance benchmarking. In the sixth week, user testing will be conducted, and necessary adjustments will be made based on the feedback received.The seventh week will be spent finalizing the application and preparing for the project presentation. In the eighth and final week, the final project presentation will be delivered, and the project’s successes and areas for improvement will be discussed.

For the final project presentation preparation, start by outlining the key points you want to cover, such as the project’s goals, the challenges you faced, the solutions you implemented, and the results you achieved. Practice your presentation to ensure you are well-prepared.

**Part II:**

The design of this API is centered around providing a streamlined and comprehensive method for retrieving detailed information about sports clubs or players, particularly in Major League Baseball (MLB) and the National Basketball Association (NBA). The API is designed to be versatile, employing various data collection methods such as APIs, web scraping, and other suitable techniques to ensure comprehensive coverage. This versatility also allows the API to expand and incorporate new data sources as they become available. The API is structured around several key functions that retrieve specific types of data. For instance, functions like get\_player\_game\_logs, get\_team\_roster, get\_team\_schedule, get\_league\_standings, get\_player\_shot\_chart, and get\_team\_stats are designed to fetch data related to player game logs, team rosters, team schedules, league standings, player shot charts, and team statistics respectively.

Each function interacts with the NBA API pybaseball or Retrosheet database to parses the returned data, and formats it into a user-friendly dictionary. The functions will use a combination of web scraping, the apis, and the clingo logic library. The clingo library will be used to parse through a massive amount of data that is located within an excel file The API also includes a user interface that allows users to interact with these functions and retrieve the desired information. The user interface is designed to be simple and intuitive, providing a menu of options for the user to choose from and handling their input to call the appropriate API functions. There is a small version of this design located in the folder provided.

Retrosheet is a non-profit organization that collects and provides comprehensive play-by-play accounts of every major league baseball game. The data spans from 1912 to the present, with some accounts deduced from newspaper stories and box scores. They also compile data for Negro Leagues games. They provide the historical data portion of the api.The Retrosheet database uses the language R to allow for querying the data they have collected. The pybaseball allows users to get current games such as game logs and other analytics. This would provide the less technical to still get info but more advanced users to get interesting statistics. It allows the APi to get data from places like statcast, mlb.com and baseballreference.com. The nba\_api is a python library that allows users to get statistics from the NBA.com api. The NBA\_api uses the http protocol to retrieve data from the NBA website.

Documentation has been provided with this file and is in the folder. It is called SportsInfoAPI.NBA.html and SportsInfoAPI.MLB.html

Part III: Implementation:

The API for this system is done in three different systems of algorithms. In the NBA section of the API, some of the functions were done using the basic nba\_api which would send back a dataframe of the data that was collected. The second style was done by using web scrapers to gather tables from the website such as baseball-reference and basketball-reference. They would parse through the table that is collected and collect the requested information. The third style would use clingo to parse the data that is in an excel file that has the game logs of the 2014-2023 MLB Season.

The implementation of the system is two different systems. One is a driver program called driver that has a basic UI. This UI is a basic interface to allow a user to interface with the application in an easy manner. Another of the implementation of the system is two different query analyzers. The NBA one which is called NBA\_Query. The MLB one which is named MLB\_Query is done just like the NBA one. This was done by using the NLTK library and the spacy library. They would take a string based query and use an algorithm to figure out context clues to use the correct function.

This entire system was created using Python as the primary language. For libraries, I used the NTLK which is used for tokenization of the queries. Also using the spacy library for Named Entity Recognition or NER. NER is used to identify proper nouns that are present inside the query. Data was provided from the nba\_api, pybaseball, and the retrosheet databases that can be downloaded from Retrosheet.org. For logic queries, I decided to use the clingo python interface. This would allow for me to parse through thousands of rows of data to answer questions that the user has about sports. I tried to use prolog but I had issues with that due to the limited power of my laptop and it would not interface with the swi-prolog that was already installed on my laptop. I also tried to use the datalog library that we looked at in class. I was unable to get it to import itself into my python environment. The development of this application has taken many changes and still has a lot of work to do. With all the files that I have created for this application has about 2,000 lines of code all together. Something interesting that I have noticed during testing with the state of art that I have discussed in Part I. There are a lot of questions that I started to come up with looking at the data that I was unable to ask it but can using my application.

Part IV: Testing and Evaluation:

For testing and evaluation, I create two different files that have a sample version of every single function that I created for the application. Each of these has a sample of the queries that can be run by the individual on both the NBA and MLB. I made it easy for people to use this as testing to see what is given to the user when they ask for the information. To run the test cases, you can just run the files MLBTestcases and NBAtestcases. They are set up to tell the user the time it takes for a function to run and tell the user.

The results of the test were that everything was running the way I expected them to run. The output of these functions could be modified to be better output to the user but the function did say what I expected the value I expected. The overall outcome of the test cases is that the info that was being sent to the user was coming faster than the state of art that was mentioned in Part 1. The interesting thing is that some of the answers were faster than google and were unable to be found anywhere I looked for the answer.

Part V: References:

Data Sources and Libraries:

NBA API. (n.d.). Retrieved April 21, 2024: <https://github.com/swar/nba_api/tree/master>

Class NBA

Retrosheet. (n.d.). Retrieved April 21, 2024: <https://www.retrosheet.org/>

Class MLB

pybaseball: <https://github.com/jldbc/pybaseball>

Class MLB

Baseball Reference. (n.d.). Retrieved April 21, 2024: <https://www.baseball-reference.com/>

Class MLB

Basketball Reference. (n.d.). Retrieved April 21, 2024: <https://www.basketball-reference.com/>

Class NBA

Articles I have read for help:

Gebser, Martin; Kaminski, Roland; Kaufmann, Benjamin; Ostrowski, Max; Schaub, Torsten; Thiele, Sven. (2015). Clingo Guide. Retrieved April 21, 2024, from <https://wp.doc.ic.ac.uk/arusso/wp-content/uploads/sites/47/2015/01/clingo_guide.pdf>

**Appendix**:

Part I:

Tried to be more precise input,output and example uses

Added information about using Clingo within my code

Got rid of the word bunch and added more precise tasks

Added a weekly breakdown

Part II:

Added information about the clingo used

Part V:

I updated the references to include the date accessed and more information where it is used.