Software Requirements and Design Document

For

Group 3

Version 1.0

Authors:

Kevin R

Luke M

Daniel T

Christopher T

1. Overview (5 points)

The system will be a Sport Betting AI assistant to aid in helping our clients select profitable and likely picks for any given night of sports betting. Our application will hone in on the NBA, but we will create it in such a way that will make it easily scalable in case we want to switch sports or add various sports into the mix. Our application will have a UI which will allow the player to view the odds of any given pick.

2. Functional Requirements (10 points)

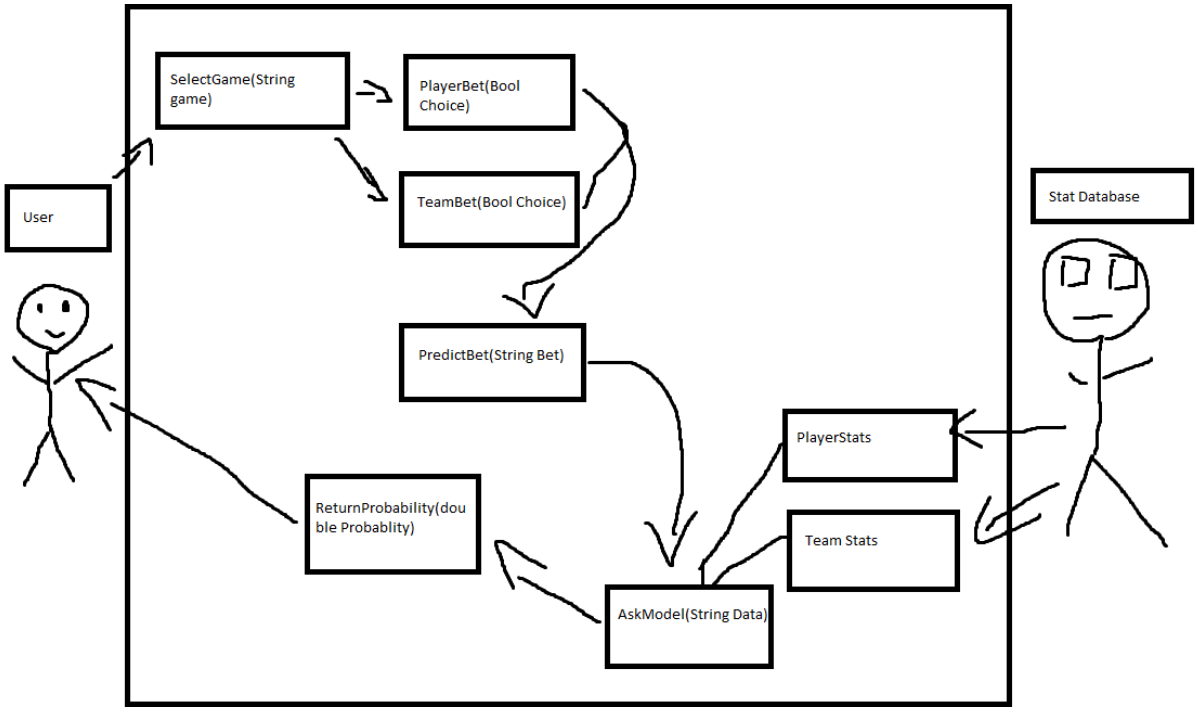
1. Proper data (10) : The system will collect, store, and validate all necessary data, including player statistics, team schedules, and historical performance records, to ensure accurate calculations and predictions. If we do not have the proper and necessary data for our algorithms, our entire system will not work as intended.
2. Code for data lookups(5): The system will be able to quickly find specific data points, such as a player's average points against a particular team or performance in home vs. away games. Although this is not crucial, it is certainly important.
3. Proper Front-End code(8): The system will provide a user interface that allows for seamless access to player stats, matchup predictions, and team schedules. We need a working and functioning front-end to ensure our clients can use the software and it is seamless to use for them.
4. Proper "middle-End" code(10): Arguably more important than the front-end itself, the system will ensure our back end correctly links up to our front end if not the entire application simply will not work. This includes handling user requests, retrieving relevant data, and delivering accurate outputs in real time.
5. A strong Machine Learning Model(10): The system will implement and continuously work on refining a machine learning model that analyzes past game data and player performance to generate accurate game predictions and betting recommendations. The model shall be trained on a dataset containing past game results, player statistics, and matchup trends.

3. Non-functional Requirements (10 points)

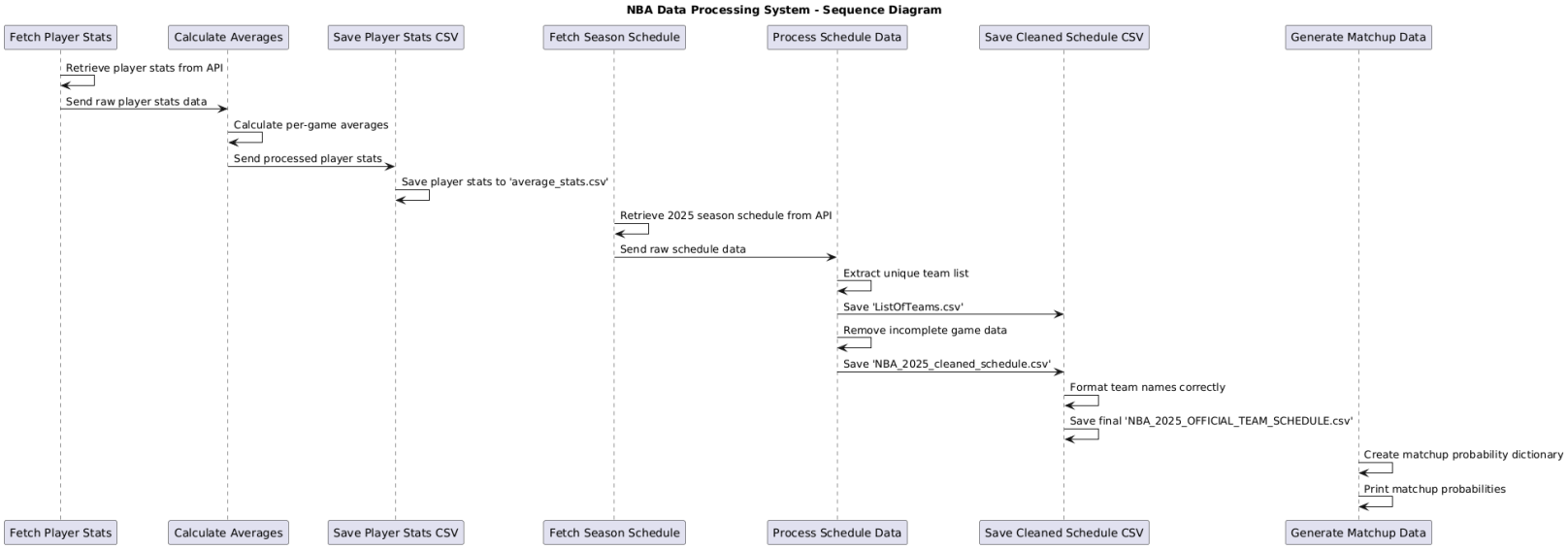
1. Performance: The system will process player statistics and game schedule data within 5 seconds for typical data sizes. This would be 30 teams and 82 games per team, roughly speaking. The system will also load and clean the game schedule in under 10 seconds for an entire season. This ensures the system is responsive and also efficient with datasets we might have in the future.
2. Scalability: The system will be capable of handling historical and future seasons without significant modifications to the code. The system needs to be adaptable in order to support future datasets and additional analytical features.

3. Reliability: The system will not lose any data during processing and file saving. In the case that an error occurs, the system will report an error message for missing/incomplete data. This approach prevents misleading our users with incorrect statistics and ensures the integrity of our data.
4. Security: The system will not expose any private data of the user, as everything it analyzes is publicly available data. The system will also store all api keys in an environment variable as opposed to in our code, to prevent unauthorized access.
5. Maintainability: The system's code will be formatted in a modular manner (Jupyter). All functions will be clearly named and documented to allow for easy modifications in the future. This ensures easy bug fixes down the road.
6. Data Accuracy: The system will ensure all statistical calculations are performed with precision (proper rounding, storing variables in the correct data types, etc.). The system will validate the correctness of the statistics as well, to ensure there are no inaccuracies in the metrics.

4. Use Case Diagram (10 points)



5. Class Diagram and/or Sequence Diagrams (15 points).



6. Operating Environment (5 points)

The NBA Data Processing System is designed to run in a collaborative development environment using VS Code Live Share and Jupyter Notebook, allowing multiple contributors to work on the code at once. Since it is primarily written in Python, it is platform-independent and can run on Windows, macOS, or Linux without major modifications. However, for optimal performance, the system should be used on a modern laptop or desktop with at least 8 gigabytes of RAM and a multi-core processor, though 16 gigabytes of RAM is recommended for handling the larger datasets efficiently. The system also requires a minimum of 5-10 gigabytes of free storage to accommodate data files, machine learning models, and dependencies. Additionally, it relies on various Python libraries, including Pandas, NumPy, Matplotlib, Seaborn, Scikit-Learn, TensorFlow, and Keras, as well as external data sources such as the basketball-reference-scraper API. This ensures compatibility with statistical analysis, visualization, and machine learning tasks. The development workflow is based on Jupyter Notebooks for interactive scripting and testing.

7. Assumptions and Dependencies (5 points)

Our project assumes that the basketball-reference-scraper and Balldontlie API's will remain available and accurate, as any changes could disrupt data retrieval. We also rely on Python libraries such as Pandas,

NumPy, Matplotlib, and TensorFlow, assuming they stay compatible and supported. One of the biggest challenges was deciding which data to keep and how to format team stats, and we settled on home vs. away performance and head-to-head matchups for better analysis. We assume all contributors will continue using VS Code Live Share and Jupyter Notebook for collaboration, and that most users will have computers capable of handling the data processing. Any major changes in these areas could require us to rethink parts of the project.