Software Implementation and Testing Document

For

Group 3

Version 1.0

Authors:

Kevin R

Luke M

Daniel T

Christopher T

1. Programming Languages (5 points)

Python backend and Javascript frontend, React for the framework.

2. Platforms, APIs, Databases, and other technologies used (5 points)

- pandas
- matplotlib
- numpy
- scikit-learn
- cartopy
- keras
- keras-tuner
- Tensorflow
- Juypter Notebook
- FireBase
- balldontlie API

3. Execution-based Functional Testing (10 points)

For this increment functional testing involved:

- Testing to make sure FireBase stored the information after logging in and signing up
- Adding API calls allowing us to see current NBA games going on and making sure the information is reflected correctly.
- The AI was tested and trained in order to accurately predict stats better than humans.

4. Execution-based Non-Functional Testing (10 points)

Non-functional testing in this increment focused on expanding existing checks from Increment 1 while introducing new functionality:

- **Scalability Testing:** We continued evaluating scraper performance over varying time periods (e.g., single vs. multiple seasons) to ensure consistent handling of larger datasets.
- **Live Score Updates:** On the frontend, we implemented real-time score polling using setInterval and tested its refresh rate and accuracy. Games and scores were confirmed to update every 60 seconds without requiring manual reloads.
- **Timezone Accuracy:** We addressed and tested a timezone-related bug that caused early date shifts, ensuring Eastern Time is correctly reflected when displaying daily games.
- **Authentication Flow Testing:** Firebase login and signup functionality was integrated and verified through manual testing across devices and browsers to ensure successful user account creation and access control.
- **Collaboration & Development Environment Testing:** Using VS Code Live Share and Git version control, all members worked in sync, validating consistent builds across machines.

Model training pipelines were initialized on the backend, and initial performance checks were conducted to verify compatibility with the cleaned data.

5. Non-Execution-based Testing (10 points)

We conducted structured non-execution-based testing to maintain quality, detect bugs early, and ensure feature stability:

- **Code Reviews:** Before merging new features (e.g., Firebase auth, real-time scores, UI updates), team members conducted thorough code reviews to improve clarity, reduce redundancy, and enforce best practices.
- **Walkthroughs & Team Discussions:** We hosted multiple code walkthroughs and reviewed our overall architecture—including the scraper, proxy server, and frontend API integration—to address potential edge cases.
- **UI Consistency Audits:** Visual and usability audits ensured that dark theme styles, layouts, and routing logic (navbar links, page states) stayed consistent across views and screen sizes.
- **Issue Tracking & Documentation:** We continued using GitHub Issues to document bugs, improvements, and development milestones. Closed issues reflected completed features and testing efforts across the full stack.

These efforts allowed us to iteratively test and refine the system as new frontend functionality was layered in and backend infrastructure matured.