Software Implementation and Testing Document

For

Group <6>

Version 3.0

Authors:
Frankie Messina
Zach Porcoro
Raul Rodriguez
Andrew Stade
Peter Vasiljev

1. Programming Languages (5 points)

- Javascript (with JSX)
    - Required for building React components and handling web behavior. Used in the majority of the source files.
- Additional languages (not exactly programming languages)
    - HTML
        - Required for web development. The elements that make up the content of our website are defined within JSX elements that are very similar to HTML syntax. Although we aren't developing too heavily within any html files, we are still required to utilize many of the same principles needed to write HTML.
    - CSS
        - Required for styling the content of the website. We have external stylesheets that will hold any CSS styling rules so that they can be applied to various elements as needed.

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

- Platforms
    - Web-based
- Front-end Frameworks
    - React
- APIs
    - Firebase Authentication: Used in the Login and Registration components to create and log into user's accounts. Also used in every page to make sure that the user is actually logged in.
- Databases
    - Firestore: Used in any components where they have to retrieve or save the user's data.
- Libraries
    - MUI (Material-UI) for our React components such as buttons, text fields, etc.
    - React router
    - Jest testing library for React
- Other technologies
    - VSCode with git source control
    - Node package manager
    - Node.js

3. Execution-based Functional Testing (10 points)

All currently implemented functional requirements have been tested manually as they were implemented. We implemented a small amount of unit tests specifically for the nav bar component, but the tests are not very comprehensive. Therefore, the testing that we did perform was more influenced by the idea of black-box testing in the sense that we focused on making sure our components performed the expected functionality.

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

The execution-based non-functional testing that we performed for this increment did not differ significantly from the previous increment. We mainly focused on ensuring that the application performance did not degrade with any of the additional features that were added. This process did not really involve measuring any performance metrics. Our application continues to check for valid user authentication on each page that requires a login to access, and we were still unable to get around these security checks.

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

Whenever we created a pull request, we made sure to have at least one other group member review the changes and approve them. We would also frequently check out other member's branches and view the progress that they had made. Finally, every once and a while we would get in a call together and quickly walk through what features we implemented and how. This allowed us the opportunity to make adjustments to our code if necessary.