**Software Implementation and Testing Document**

**For**

**Group 31**

Version 2.0

**Authors**:

Aiden Livingston

Dustin Haire

Maria Penalver

McKenna Warrick

Maria Penalver

# Programming Languages (5 points)

* JavaScript
* CSS
* HTML

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

***Technologies***

* **React**: JavaScript library for building user interfaces, particularly single-page applications
* **Node.js:** A JavaScript runtime built on Chrome's V8 JavaScript engine, allowing you to run JavaScript on the server
  + **react-router:** A standard routing library for React, used to handle navigation in single-page applications
  + **firebase admin:** A server SDK that lets you interact with Firebase from environments for various administrative tasks
  + **axios:** A promise-based HTTP client for making requests to APIs from both the browser and Node.js
  + **nodemon:** A utility that starts a backend server and monitors changes in your source and automatically restarts your server
  + **cors:** A Node.js package for providing a Connect/Express middleware that can be used to enable CORS (Cross-Origin Resource Sharing)
* **Express.js:** A flexible Node.js web application/ backend framework
* **SCSS:** Sassy CSS, a preprocessor scripting language that is interpreted or compiled into CSS
* **Firebase:** A platform developed by Google for creating mobile and web applications
  + **Firestore NoSQL Database:** A flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform

# Execution-based Functional Testing (10 points)

At this stage of the project, we have not yet conducted functional testing. However, we plan to implement execution-based functional testing by the third increment of development. During this phase, we will validate the functional requirements specified in our RD Document by executing test cases for user and API interactions.

# Execution-based Non-Functional Testing (10 points)

**Performance Testing:** Examined the functionality of the site to handle vast content in flashcards and folders. Created test scenarios where hundreds of flashcards were nested into folders to establish whether the application responds. The developers utilized the browser developer tools to find out how long it takes to load a web server’s critical pages (e.g., home, folder views and study mode). The large folder rendering was slow, with about 7.5 seconds at 400 flash cards (This is an issue we aim to tackle in the next increment).

**Scalability Testing:** Simulated multiple users interacting with the app simultaneously using Postman to stress test the Firebase backend for simultaneous read/write operations.

**End Users Testing:** Conducted usability testing with potential users to evaluate ease of navigation, clarity of UI, and the learning curve for new users. Legitimate feedback directed towards the minor adjustments spent on rearrangement of the screen with some including movement of button locations to all corners and taking of the equivalent screen shots for further categorization.

**Compatibility Testing:** Tested Flashcardio on multiple browsers (e.g., Chrome, Firefox, Edge) and devices (desktop and tablet) to ensure consistent functionality and responsiveness. Checked cross-browser compatibility and optimized mobile responsiveness using media queries in SCSS.

# Non-Execution-based Testing (10 points)

**Code Reviews:** Conducted peer code reviews with team members for critical components such as Firestore utilities and SCSS styles. Focused on adherence to best practices, modularity, and avoiding redundant code patterns or styles. Improved code readability and identified opportunities to refactor complex logic (e.g., simplifying folder deletion logic).

**Static Analysis:** Used static analysis tool ESLint to identify coding errors, enforce consistent styling, and detect potential security violations. Resolved multiple warnings related to unused variables, potential null references, and formatting inconsistencies.

**Walkthroughs:** Conducted team walkthroughs of the project architecture, discussing the rationale behind key decisions like folder-path-based routing and Firestore query structure.Helped ensure alignment on design decisions and identified opportunities for optimization, such as caching frequently accessed folders.

**Inspection of Requirements and Design Documents:** Reviewed the Requirements Document (RD) to ensure the application aligns with the specified non-functional requirements. Checked that the current implementation meets outlined requirements and flagged areas for future improvement.