Software Implementation and Testing Document

For

Group <13>

Version 1.0

Authors:
Ethan Anderson
Luis Ferrer
Giancarlo Franco
Leo Ribera

1. Programming Languages (5 points)

List the programming languages use in your project, where you use them (what components of your project) and your reason for choosing them (whatever that may be).

TypeScript/JavaScript (TSX/JSX)

- ***Where Used: Client-side React components (App.tsx, Login.tsx, Signup.tsx, etc.)***
- ***Reason: Type safety with TypeScript provides better developer experience, reduces runtime errors, and improves maintainability for the frontend application.***

JavaScript (Node.js)

- ***Where Used: Server-side logic (server.js, routes, models, middleware)***
- ***Reason: Enables sharing code/models between frontend and backend, has excellent package ecosystem for web servers, and provides non-blocking I/O for handling concurrent requests.***

CSS

- ***Where Used: In index.css and via Tailwind utility classes throughout components***
- ***Reason: Necessary for styling the UI components; using Tailwind for rapid development with utility-first approach.***

Python (Planned)

- ***Where Used: Recommendation engine (currently empty files in recommendation directory)***
- ***Reason: Powerful language for data processing and machine learning, ideal for implementing anime recommendation algorithms.***

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

List all the platforms, APIs, Databases, and any other technologies you use in your project and where you use them (in what components of your project).

Frontend

- ***React****: Component-based UI library for building the interface*
- ***React Router****: Client-side routing between different pages/views*
- ***Tailwind CSS****: Utility-first CSS framework for styling*
- ***PostCSS****: Tool for transforming CSS with JavaScript plugins*
- ***Webpack*** *(via React Scripts): Module bundler for the application*

Backend

- ***Node.js****: JavaScript runtime for server-side code*
- ***Express.js****: Web framework for building the REST API endpoints*
- ***JWT (JSON Web Tokens)****: Authentication mechanism for securing endpoints*
- ***bcryptjs****: Password hashing library for secure user authentication*

Database

- ***MongoDB****: NoSQL database for storing user data, anime details, reviews, etc.*
- ***MongoDB Atlas****: Cloud hosting platform for the MongoDB database*
- ***Mongoose****: ODM (Object Document Mapper) for MongoDB schema definition and validation*

External APIs

- ***Jikan API****: Third-party API for fetching anime data from MyAnimeList*

Development Tools

- ***dotenv****: For managing environment variables*
- ***nodemon****: For automatic server restarts during development*

3. Execution-based Functional Testing (10 points)

*Describe how/if you performed functional testing for your project (i.e., tested for the **functional requirements** listed in your RD).*

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

*Describe how/if you performed non-functional testing for your project (i.e., tested for the **non-functional requirements** listed in your RD).*

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

Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).