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)

User authentication including login and signup was tested for form validation, password hashing, and successful login/signup flows. We manually tested the frontend forms to ensure proper validation messages appeared and used the browser inspector on Firefox to test the backend endpoints with the MongoDB. We confirmed that JWT tokens were generated and used correctly for authenticated requests.

Anime browsing was tested by integrating the Jikan API. We interacted with and confirmed that the anime listings were updated according to the API query. We also monitored API calls in browser developer tools to ensure correct data was being fetched and displayed.

User reviews were tested by reviews tied to specific anime. We used frontend forms as well to verify that reviews were saved and retrieved correctly. We checked MongoDB Atlas to confirm the data was stored as expected.

Protected routes and authentication were tested by attempting to access secured endpoints with and without valid JWT tokens. We verified that access was correctly restricted or allowed based on the presence and validity of the token.

General UI functionality was tested by navigating through the application, checking page transitions, and testing responsiveness across different screen sizes. We ensured that all routes rendered the correct components and that the application behaved consistently across devices.

All features implemented so far passed their respective functional tests based on our initial requirements. Testing was primarily done manually using the browser interface, inspector, and the MongoDB Atlas dashboard.

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

Performance Testing

- Measured search response times using Firefox inspector to confirm results return within 3 seconds
- Tested API rate limiting by simulating concurrent requests to verify our caching prevents hitting Jikan API restrictions

Security Testing

- Verified bcrypt implementation with 10 salt rounds by examining database password hashes
- Tested JWT authentication by confirming tokens are properly issued, protected routes reject invalid tokens, and tokens expire after 24 hours

Compatibility Testing

- Tested the application on Chrome, Firefox, Safari, and Edge browsers
- Verified responsive design using device emulation for mobile, tablet, and desktop screens

Reliability Testing

- Simulated Jikan API outages to verify our caching system properly served cached anime data
- Tested error handling by triggering various error conditions to confirm appropriate error messages

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

Code Reviews

- Implemented mandatory peer code reviews before merging any pull requests
- Used Prettier to enforce consistent coding standards
- Conducted focused reviews for authentication-related code to identify security vulnerabilities

Architecture Reviews

- Performed architecture reviews to ensure proper separation between frontend, backend, and recommendation services
- Reviewed database schema design to verify support for current requirements and future expansion

Documentation Reviews

- Reviewed API endpoint documentation for completeness
- Evaluated React component documentation to ensure reusable components were well-documented

Code Walkthroughs

- For complex features like the recommendation system, conducted team walkthroughs to trace data flow and logic
- Identified potential edge cases and optimizations during collaborative code examination sessions