Software Requirements and Design Document

For

Group 13

Version 1.0

Authors:
Luis Ferrer
Ethan Anderson
Giancarlo Franco
Leo Ribera

1. Overview (5 points)

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

We are making a social platform for users to list and review their favorites animes, have our system suggest new ones to watch, as well as display these favorites animes off to users they have added. We are creating this on a MERN stack (MongoDB, Express.JS, React.JS, and Node.JS). Our vision is similar to a website like MyAnimeList, but with a modern feel like CrunchyRoll's layout. We felt as if there are no modern feeling websites in which users can get new anime recommended to them, and show their favorites off to their friends. LetterBoxd is a great inspiration to what we want to make. Simplicity would be key to it, since we feel as if many sites currently have an overabundance of features.

2. Functional Requirements (10 points)

1. The system shall allow users to register with an email, password, and display name. (High Priority)
2. The system shall authenticate users via JWT tokens to maintain secure sessions. (High Priority)
3. The system shall enable users to log in using their registered email and password. (High Priority)
4. The system shall enforce password requirements: minimum 8 characters with at least one uppercase letter, one lowercase letter, and one number. (Medium Priority)
5. The system shall validate display names to be 3-20 characters without spaces. (Medium Priority)
6. The system shall fetch, display, and cache anime data from the Jikan API. (High Priority)
    - Rationale: Caching reduces API calls and improves performance while ensuring data availability.
7. The system shall provide search functionality for users to find anime by title. (High Priority)
8. The system shall display detailed information for each anime including synopsis, genres, and trailer URLs. (High Priority)
9. The system shall show streaming links where users can watch the selected anime. (Medium Priority)
10. The system shall allow users to maintain a "watched" list of anime they have completed. (High Priority)
11. The system shall allow users to maintain a "plan to watch" list for anime they intend to watch. (High Priority)
12. The system shall enable users to rate anime on a 1-5 scale. (High Priority)
13. The system shall allow users to write reviews of up to 350 characters for anime they have watched. (Medium Priority)
14. The system shall permit users to select their top three favorite anime to display on their profile. (Medium Priority)
15. The system shall display general recommendations on the homepage organized into horizontal rows. (High Priority)
16. The system shall provide a "Popular Picks" section showing trending anime. (High Priority)
17. The system shall display a "Seasonal Picks" section showing currently airing anime. (High Priority)
18. The system shall generate personalized recommendations based on user watch history and ratings. (High Priority)
    - Rationale: This core functionality differentiates the platform from basic anime databases.
19. The system shall offer "Sleeper Picks" featuring highly-rated but less popular anime. (Medium Priority)
20. The system shall allow users to add other users as friends. (Medium Priority)

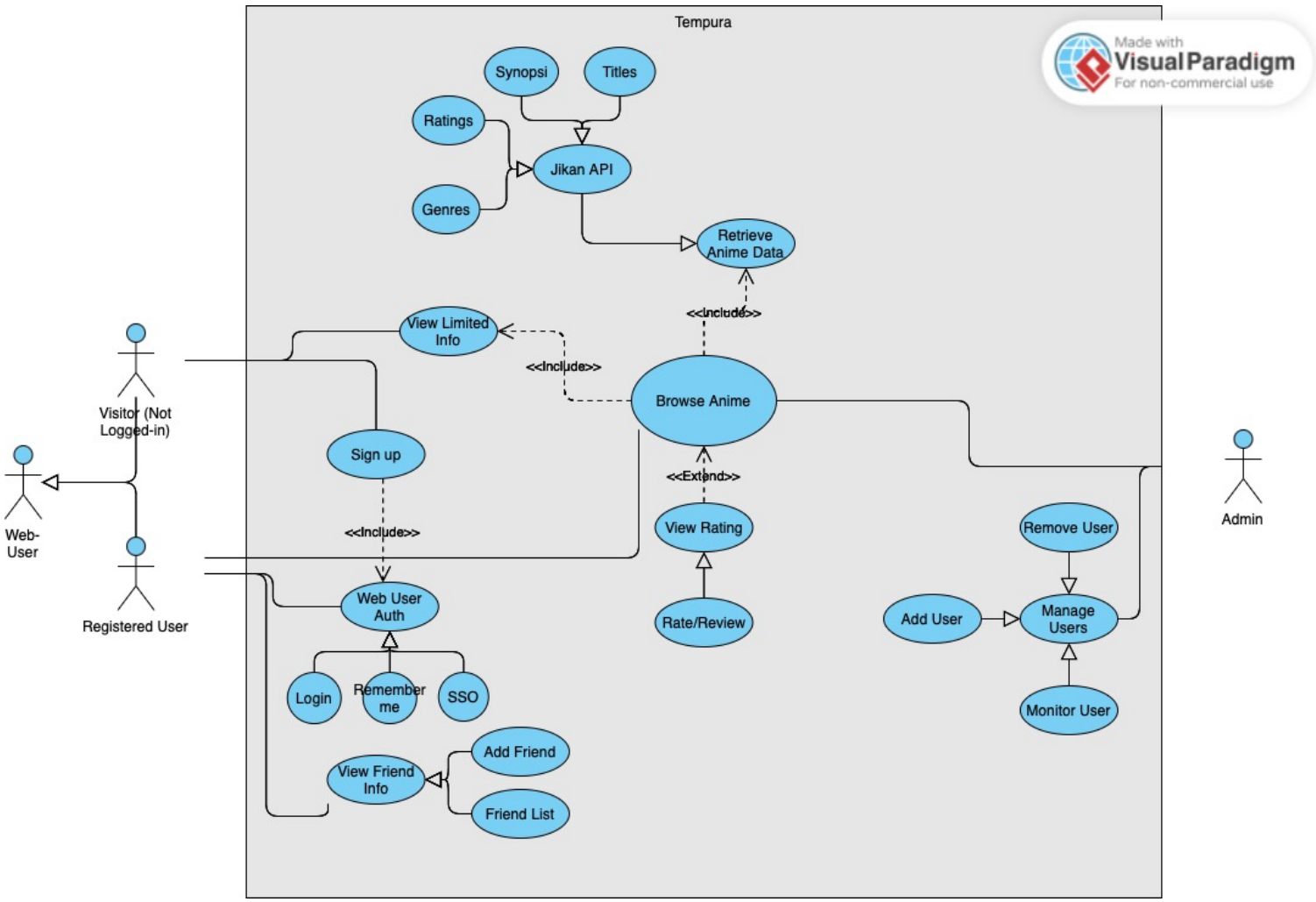
21. The system shall display an activity feed showing recent actions by friends. (Medium Priority)
22. The system shall track and display when users add anime to their watched list. (Medium Priority)
23. The system shall track and display when users post reviews. (Medium Priority)
24. The system shall track and display when users update their top three favorite anime. (Low Priority)
25. The system shall allow users to view their friends' anime lists and reviews. (Medium Priority)
26. The system shall provide a navigation bar with search, friends, and account buttons. (High Priority)
27. The system shall display anime in visually appealing card components with images. (High Priority)
28. The system shall allow users to send anime recommendations directly to friends with a personalized message. (Medium Priority)
    - Rationale: This enhances social engagement by enabling users to directly share discoveries with specific friends rather than only through passive activity feeds.

3. Non-functional Requirements (10 points)

1. The system shall respond to user searches within 3 seconds under normal load conditions.
2. The system shall cache anime data to reduce API calls and ensure operation during Jikan API outages.
3. The system shall encrypt all passwords using bcrypt with a minimum of 10 salt rounds.
    a. Rationale: Bcrypt with sufficient salt rounds protects user credentials against brute force attacks.
4. The system shall use JWT tokens with 24-hour expiration for maintaining authenticated sessions.
5. The system shall validate all user inputs to prevent injection attacks and XSS vulnerabilities.
6. The system shall implement proper authorization checks to ensure users can only access and modify their own data.
7. The system shall store sensitive information like API keys and database credentials in environment variables, not in source code.
8. The system shall maintain 95% uptime during peak usage hours.
9. The system shall implement error handling for API failures to provide graceful degradation.
10. The system shall have automated backups of the database daily.
11. The system shall be designed with a modular architecture (separate server, client, and recommendation services) to facilitate scaling of individual components.
12. The system shall use efficient database indexing to maintain performance as the user base grows.
13. The system database schema shall support future expansion of features without major restructuring.
14. The system shall have an intuitive UI that requires no tutorial for basic operations like searching and adding anime to lists.
15. The system shall be responsive and functional across desktop and mobile devices.
16. The system shall provide clear error messages that help users understand and resolve issues.
17. The system shall limit API rate usage to prevent hitting Jikan API restrictions.
    a. Rationale: Jikan API has strict rate limits that could disrupt service if exceeded.
18. The system code shall follow consistent styling and documentation standards to facilitate maintenance.

19. The system shall separate concerns between frontend, backend, and recommendation logic for easier updates.
20. The system shall use version control to track changes and facilitate collaboration.
21. The system shall function correctly on modern browsers including Chrome, Firefox, Safari, and Edge.
22. The system shall use responsive design to support various screen sizes and orientations.
23. The system shall comply with data protection regulations for user information.

4. Use Case Diagram (10 points)



Textual Descriptions:
Each use case present in the diagram is described below. With each use case, it will show what Actor it is associated with, and what the description/function of the use case is. Relationships are also described

View Limited Info
Actor: Visitor
Description: A visitor can browse a limited set of anime information such as basic details from the Jikan API without needing to log in or register. This helps users explore the platform before deciding to sign up.

Sign Up
Actor: Visitor
Description: A visitor can create an account by providing necessary information like email and password. Upon successful registration, the visitor becomes a registered user.

Web User Auth
Actor: Registered User
Description: The registered user can authenticate themselves using login credentials. The system verifies the user and grants access to personalized features.

Login
Actor: Registered User
Description: A registered user inputs their login credentials. If valid, the system authenticates the user and allows access to the platform's features.

Remember Me
Actor: Registered User
Description: The system can optionally remember the user on future visits using local/session storage to keep them logged in. This is enabled by default, but we will be implementing a feature to turn this off.

SSO
Actor: Registered User
Description: Users can log in using a third-party single sign-on service like Google or Facebook, simplifying the login process.

View Friend Info
Actor: Registered User
Description: After logging in, the user can view details of their added friends such as usernames, watchlists, or ratings.

Add Friend
Actor: Registered User
Description: A logged-in user can add another user as a friend by searching or selecting from existing users.

Friend List
Actor: Registered User
Description: Displays a list of all the user's added friends for easier access and interaction.

Browse Anime
Actor: Visitor, Registered User
Description: Users can browse anime titles using the integrated Jikan API. This includes listing, searching, and filtering anime content.

Retrieve Anime Data
Actor: System (via Jikan API)
Description: The system fetches anime data such as titles, synopsis, genres, and ratings from the Jikan API for display on the platform.

View Rating
Actor: Registered User
Description: While browsing, a user can view ratings of anime titles provided either by other users or fetched from the API.

Rate/Review
Actor: Registered User
Description: A user can submit their own rating or review for an anime. These are stored in the system and may be displayed to other users.

Database Related:

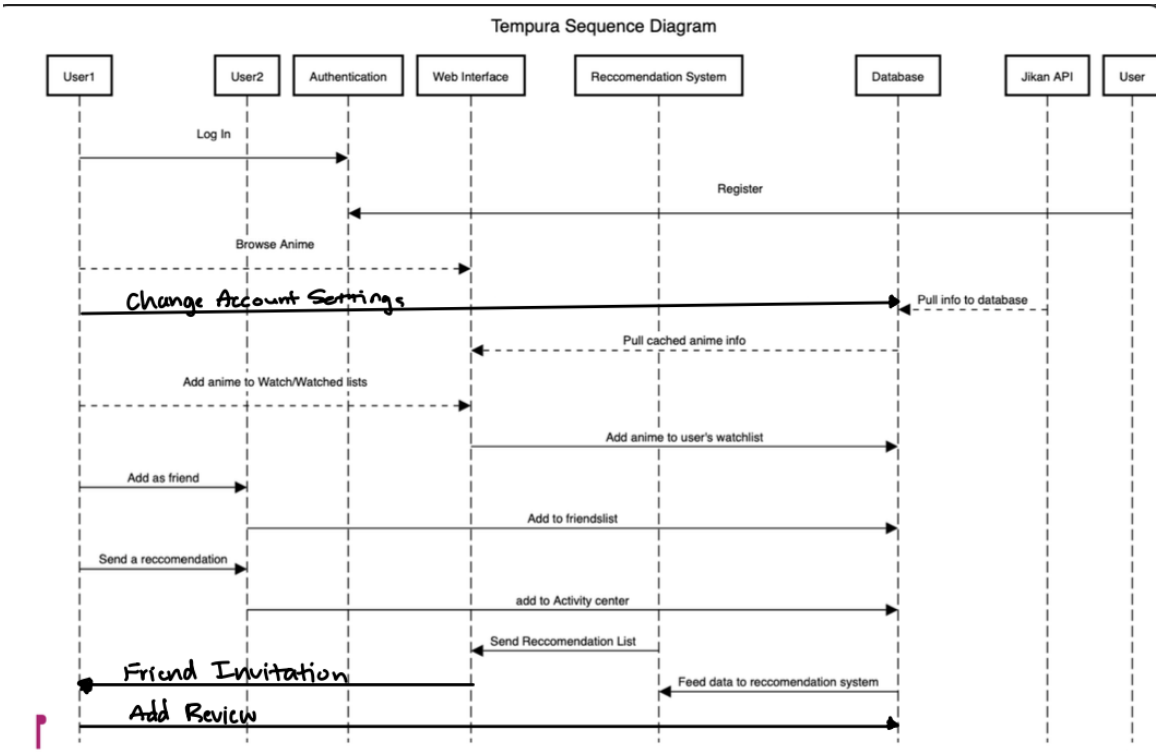
Manage Users
Actor: Admin
Description: Admin has full access to user management, including viewing user activity, modifying user roles, or accessing administrative tools.

Add User
Actor: Admin
Description: Admin can manually add a new user to the system, for example in cases of special roles or moderation purposes.

Remove User
Actor: Admin
Description: Admin can delete or deactivate a user's account based on activity or policy violations.

Monitor User
Actor: Admin
Description: Admin can view user behavior, such as posting history, login activity, and interaction logs, to monitor the health of the community.

5. Class Diagram and/or Sequence Diagrams (15 points)



6. Operating Environment (5 points)

Tempura is designed as a web-based system with a standard three-tier architecture (frontend, backend, database) plus a Python recommendation engine. The frontend is built with Next.js/React, making it compatible with all modern web browsers, while the backend uses Node.js, Express, and MongoDB to handle data storage and API requests. The system relies on the Jikan API for anime information and will run on standard development environments and hosting services. Requires internet connection as well.

7. Assumptions and Dependencies (5 points)

We assume Jikan API will handle all the requests we need. It is our primary backbone because it provides almost all our data on anime, genres, etc. We assume the MongoDB connection works. Assuming our server is secure with the Mongoose system. We assume YouTube links will automatically play when an anime card is clicked, this is a third party system.