Software Design Document

for

JAX Front-end Project

Version 1.1 approved

Authored by Sarah Crane, Harshil Patel, Prince Safo, Salim Salim, Kate Stenberg

Edited by Sarah Crane

CS5500 – Foundations for Software Engineering, Professor Gary Cantrell

Northeastern University, Roux Institute

March 28, 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Harshil Patel, Prince Safo, Kate Stenberg, Sarah Crane, Salim Salim	3-21-25	Initial Draft	1.0
Harshil Patel, Prince Safo, Kate Stenberg, Sarah Crane, Salim Salim	3-28-25	Formatting & Revisions	1.1

Team Signatures

Name	Date
Prince Safo	03/28/2025
Harshil Patel	03/28/2025
Sarah Crane	03/27/2025
Kate Stenberg	03/28/2025
Salim Salim	03/28/2025

Section 1: Introduction

1.1: Purpose

This software design document (SDD hereafter) will lay out the design of the Jackson Laboratories Front-End project for Northeastern's Spring 2025 Portland campus CS5500: Software Engineering course. It will detail the structure and design for the project that the team intends to implement.

1.2: Scope

This project will be focused on implementing two of the GeneWeaver analysis tools in the web application. In a similar vein to the implementation of the GeneSet graphs, which is intended to show how closely linked gene sets are, this project will focus on the front-end UI/UX implementation of two analysis tools that are currently documented but missing from the web application. Those tools are: similar variant set and find variant. Specifically, this project will focus on the user interface for these tools, as the logic and algorithms have already been designed and coded. We will alter the GeneWeaver web app to include these two tools in the Analysis Tools section, mirror the existing UI/UX, and allow users to access them.

1.3: Definitions

Gene set: a list of genomic features, free text descriptive content, ontology annotations, and gene association scores [1]
Jax: Jackson Laboratories, the client on this project
MoSCoW Method: M - *Must have*, S - *Should have*, C - *Could have*, W - *Won't have*. [2]
SDD: Software Design Document
API: application programming interface
TBD: to be determined
UX/UI: user experience / user interface

Section 2: Requirements

2.1: Functional Requirements

2.1.1: The UI should provide a login page for users to enter their credentials.

- The landing page will have clickable buttons to enter login credentials.
- Login will be found in the upper right of the login page.

2.1.2: The UI should ensure smooth navigation between different pages.

- Use of larger text size and different text color will aid in navigation between pages.
- Menus will aid in organization of pages.

2.1.3: A responsive design should be implemented to support various devices.

2.1.4: The UI should support interactive charts and graphs for data analysis.

2.1.4.1: Find Variants Implementation

The system should allow for the following process.

- When a user selects “Mouse to Human”: the system should search the database for the given geneset under Mouse geneset and display the matching result from Human geneset (M)
- When a user selects “Human to Mouse”, the system should search the database for the given geneset under Human genesets and display matching results from Mouse genesets. (M)
- When a user selects an eQTL relationship, the system should query the database for gene sets with similar eQTL attributes and display the results.
- Transcript Relationship Search (M)
- When a user selects a transcript relationship, the system should query the database for gene sets with similar transcript attributes. (M)
- The application should log all failed login attempts and trigger an alert after a number of consecutive failed attempts. (C)
- The system should consistently provide accurate and reproducible gene set relationships, ensuring correctness in graph generation and analysis. It must handle errors gracefully without data corruption or unexpected failures (M).
- The user interface should be intuitive, allowing researchers to easily navigate, interpret, and manipulate gene set graphs with minimal training. Clear visualization and well-labeled elements should enhance the user experience. (S)

2.1.4.2: Similar Variant Set Implementation

The system should allow for the following processes (priority indicated in parentheses):

- Users should input a variant ID or select a variant from existing datasets to initiate the search for similar variants (M)
- Users can refine their search by applying various similarity criteria, such as sequence similarity, phenotype relevance, and disease association, ensuring that the retrieved variants align with their specific research needs (M)
- Once a query is submitted, the system should efficiently search the database and rank similar variants based on predefined metrics. To maintain efficiency, the system should retrieve and display results within a reasonable time frame, ideally under five seconds for most queries. (S)
- The application should log all failed login attempts and trigger an alert after consecutive failed attempts. (C)
- The system should consistently provide accurate and reproducible gene set relationships, ensuring correctness in graph generation and analysis. It must handle errors gracefully without data corruption or unexpected failures (M).
- The user interface should be intuitive, allowing researchers to easily navigate, interpret, and manipulate gene set graphs with minimal training. Clear visualization and well-labeled elements should enhance the user experience. (S)

2.1.5. Users should be able to share visualizations and insights via links or email.

- Under View Results section (share button)

2.1.6. Error systems should be clear and actionable.

- Error for invalid login credentials.
    - Redirection to create an account or try again.
- Error for timeout or delayed page loading.

2.2: Non-Functional Requirements

2.2.1: Performance

- The system should respond to user actions within 5 seconds under normal load conditions.
- The application should support at least 100 concurrent users without significant performance degradation.
- The system should support an increasing number of gene sets and complex graph structures without significant degradation in performance. It should be able to accommodate large-scale datasets used in genomics research.

2.2.2: Availability

- The system should have 99.9% uptime, ensuring high availability.
- In the event of a system crash, recovery should occur within 5 minutes and should be able to auto-restart in case of unexpected failures.

2.2.3: Security

- User authentication should be protected using OAuth 2.0 and all sensitive data should be encrypted using AES-256 before storage.
- The tool should protect gene set data from unauthorized access and ensure data integrity. If applicable, authentication mechanisms will be in place for restricted functionalities.

2.2.4: Portability and Compatibility

- The system should support the latest versions of Chrome, Firefox, Edge, and Safari.
- The application should be OS-independent and run on Windows, macOS and Linux.

Section 3: Component design

For this project, we will be building the website with React and since it is a component-based architecture, the project will be organized in a way that will promote reusability, separation of concern, and maintainability. The components will include a **Layout component** that will define the overall structure of the UI that is used across multiple pages, the **Page componen**t which will correspond to a route in the application and will render different content based on user navigation, the **UI component** which will contain small, reusable building blocks that will be

used across the application to ensure consistency and **Service layer component** which will contain files responsible for handling external requests, including authentication and fetching data.

Section 4: Interface design

The web application interface will have a front-end interface containing multiple system components that seek to interact with each other and provide the end user with a holistic experience. These will be outlined using Figma mockups. [4] The components (we can also call them frames/pages) include:

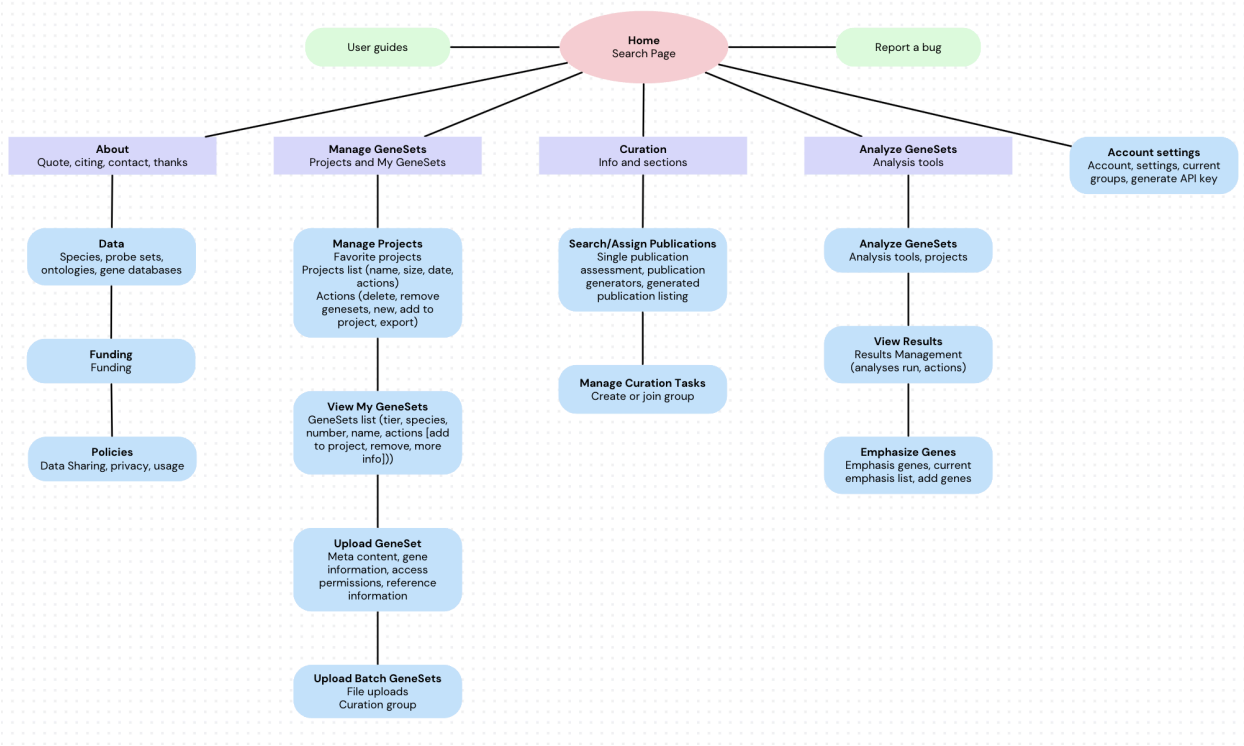
1. Frame 1: Home Page
    a. This page lays out the main pages (About, Analysis tools, login, help, etc.)
2. Frame 2: About Page
    a. Outline a bio about the company and the provided tools on the website, along with who developed it / contact for questions.
3. Frame 3: Analysis Tools
    a. Subpage for Similar Variant Set
    b. Subpage for Find Variant
4. Frame 4: Help Page
    a. Contact page
5. Frame 5: Login Page
6. Frame 6: Error page
    a. For when the web application runs into errors

Figma Mockup Link - https://www.figma.com/proto/cJMfrX8ru3BhBbzygAkhMb/GeneWeaver_HomePage?node-id=0-1&t=eimmwrKTke08II9P-1

Section 5: Pattern(s) used

We will be using a Model-View-Presenter design pattern. The MVP pattern is similar to the MVC pattern, with a “presenter” instead of a “controller”. [3] Otherwise, the architecture is similar. Input is sent to the View (which, in our case, will be the interface build). The View sends its information to the Controller, which communicates with the Model directly (this is the Geneweaver databases). The Model then returns information to the Controller, which alters the View. This is especially clear in the Geneweaver analysis tools. However, since our team is not working with the data, we will only be implementing the Controller and Presenter in this project.

Section 6: Processes



Section 7: Usability Testing

The usability testing component of this project will focus on the process in which we can assess the quality of the web application using qualitative and quantitative methodologies:

- Survey draft with closed and open-ended questions.
- Utilizing PageSpeed Insights (https://pagespeed.web.dev) for automated accessibility performance & load testing. [5]
- Internal team examination for overall readability/design development

Through the aforementioned methodologies, we would like to assess the web application through the following areas:

1. UX/UI
    a. Ease of Platform Use & Navigation – how easy/intuitive do the menu items & navigation feel?
    b. Readability / Design – How does the website perform when it comes to overall design/flow/Font Size?

c. Accessibility Compliance – WCAG (Web Content Accessibility Guidelines) standards, color contrast, keyboard navigation, screen reader compatibility.
d. Error Handling & Feedback – Are error messages clear and helpful?
e. User Engagement & Satisfaction – Surveys, feedback forms, and behavior tracking.

2. Performance

a. Can the application handle [insert number of users] at once?
b. How fast is it?

Usability testing is critical to any application, and specifically this web application because it will help the team gain metric-based feedback to improve on the website. While we do not have time to provide a fully detailed survey with 1000+ respondents to create confidence in the feedback from the user base, we would like to at least have a small sample of respondents who would be willing to test the prototype and provide feedback.

Section 8: Deployment model

For our prototype, we will develop the front end of the website using HTML, CSS, JavaScript, and React. Since this is an early-stage version (prototype), we will host it locally on our computers and run it on the localhost using a development server.

At this stage, there are no external hosting or deployment requirements, but in the future, we may consider deploying the website to a cloud platform like Netlify, Vercel, or GitHub Pages for easier access and testing.

For IP considerations, all code will be stored in a version-controlled repository (GitHub) to ensure proper tracking and ownership. We will also document key aspects of the project, including

• Setup Instructions – How to install dependencies and run the project locally.

• Maintenance Guide – Steps to update, modify, or expand the front end.

• Code Documentation – Explanation of major components and their functionality.

These documents will help future developers or stakeholders understand how to maintain and extend the website efficiently.

References

[1] “Genes and genesets,” Genes and GeneSets - GeneWeaver, https://thejacksonlaboratory.github.io/geneweaver-docs/concepts/genes-and-genesets/#genomic-features-genes (accessed Feb. 14, 2025).

[2] “MoSCow method.” Wikipedia, https://en.wikipedia.org/wiki/MoSCoW_method (accessed Feb. 16, 2025).

[3] GeeksforGeeks, “MVP (model view presenter) architecture pattern in Android with example,” GeeksforGeeks, https://www.geeksforgeeks.org/mvp-model-view-presenter-architecture-pattern-in-android-with-example/ (accessed Mar. 27, 2025).

[4] “Diagramming and chart templates.” Figma, https://www.figma.com/community/diagramming?editor_type=figjam (accessed Mar. 27, 2025).

[5] “About PageSpeed Insights,” PageSpeed Insights - Google, https://developers.google.com/speed/docs/insights/v5/about (accessed Mar. 27, 2025).

Appendices

Appendix A - Omitted Sections

Our team would like to note that the following sections have been omitted from the original outline of the SDD and their respective reasons for omission:

- Section 3: Data design (if appropriate)
    - As a front-end based project this section was not appropriate and therefore removed as it does not pertain to the scope of our project.
- Section 4: Architecture
    - The UX/UI elements are better addressed in the ‘Interface design’ section of our SDD, and where we are not focusing on back-end coding, this section was not pertinent to our project.
- Section 6: OO design
    - Again, due to the front-end nature of the project, our design specifications were better addressed in the ‘Component design’ and ‘Interface design’ sections of our SDD.
- Section 9: Design Concept Review (not standard), Section 10: Architectural design considerations (not standard), & Section 11: Component design principles (not standard)
    - After outlining and doing research for each of these sections of the SDD, the team determined that these sections did not align to the front-end nature of our project.
    - In the sake of applying the KISS principle, the team discussed these sections and firmly believe that our time would be better spent working on developing our project given that this SDD, and shift to a web development focus, presents a massive pivot in the scope of our project departing from what we had originally outlined in our SRS.
    - As ‘not standard’ sections these three sections were determined to be repetitious, and would need to address elements (like OO design, architectures, etc.) that are beyond the scope of this project, and therefore should be omitted.