Software Requirements Specification

for

The Jackson Laboratory GeneWeaver Project

Version 2.0

Prepared by

The Backend GeneUses Team:

Daniel Hayes

Harshit Khattar

Anushka Ashishkumar Doshi

Kishan Lakhshman

March 17th 2025

Table of Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Anushka & Harshit	2/12/25	Added Introduction and Overall Description Sections	0.1
Dan	2/13/25	Added System features and started Interfaces	0.2
Anushka & Harshit	2/13/25	Modified Section 1 & 2 based on the client meeting	0.3
Kishan	2/14/25	Section 3 has been added, along with Appendix A: Glossary whose Content updates for section-3 have been implemented	0.4
Kishan	2/16/25	Addressed review comments by Alex	0.5
Harshit	2/16/25	Formatted the specification and resolved comments	0.6
Anushka	2/16/25	Enhanced and improved parts of SRS making it in-lined with mentioned comments from Alex.	0.7
Kishan & Harshit	2/16/25	Added Appendix B: Swim Lane Diagram	1.0
Dan	3/10/25	Updated System features off from feedback	1.1
Anushka & Harshit	3/11/25	Updated Sections 1, 2, & 5 based on the feedback	1.2
Kishan	3/16/25	Updated Section 3 based on feedback	1.3
Harshit	3/16/25	Formatted the specification and updated section 5	2.0

Team Signatures

Member Signature	**Signed On**
Harshit Khattar	March 16th 2025
Daniel Hayes	March 16th 2025
Kishan Lakshman	March 16th 2025
Anushka Ashishkumar Doshi	March 16th 2025

1. Introduction

1.1 Purpose:

The purpose of this document is to define the software requirements for the implementation of GeneWeaver Tools (via a Plugin) integrated with the Asynchronous Task Service (ATS). This document will outline the functionalities and the features of the system, as well as its constraints while defining how it will interact with users and existing infrastructure.

This specification will serve as a reference for The Jackson Laboratory team, the development team, as well as all the product stakeholders. It ensures that all functional and non-functional requirements are clearly specified to guide the software development process.

1.2 Product Scope:

The software system described in this document is designed to enhance the GeneWeaver platform by enabling asynchronous execution of multiple gene analysis tools. Its primary purpose is to streamline the processing of GeneSets, allowing researchers and scientists to submit analysis tasks without having to wait for each to complete before initiating another. This improves the efficiency and scalability of gene data analysis workflows.

The software will allow the users to:

1. Perform analysis on GeneSets asynchronously by making use of the implemented GeneWeaver Tools and the plugin developed to handle the tool logic, input processing, and result generation.
2. Receive error log and feedback on how the task is being performed.
3. Receive the dataset to be computed either through GeneSet ID or the complete GeneSet input.
4. Process the input data, execute the corresponding tool logic, and return the computed results in a serializable format (JSON)

The following is out of scope for the current product:

1. User experience / Output GUI is not in scope of current implementation and the current implementation is purely focused on the backend execution of gene analysis tools.
2. Implementation of tools in the current geneweaver library is contingent upon time constraints and tool complexity.

The system will enable users to process and analyze the output asynchronously, without waiting for each task to complete before starting another. It will make use of the GeneWeaver REST API to process the gene data instead of directly accessing the database, ensuring compliance with privacy constraints.

1.3 Document Conventions:

ATS	Asynchronous Task Service

1.4 Intended Audience :

This requirements specification document is intended for the Software Managers, Technical & Architectural Leads, Front-end Team and Developers at The Jackson Laboratory, as well as at other organizations who intend to modify or develop future iterations of the product.

1.5 Clients and Stakeholders :

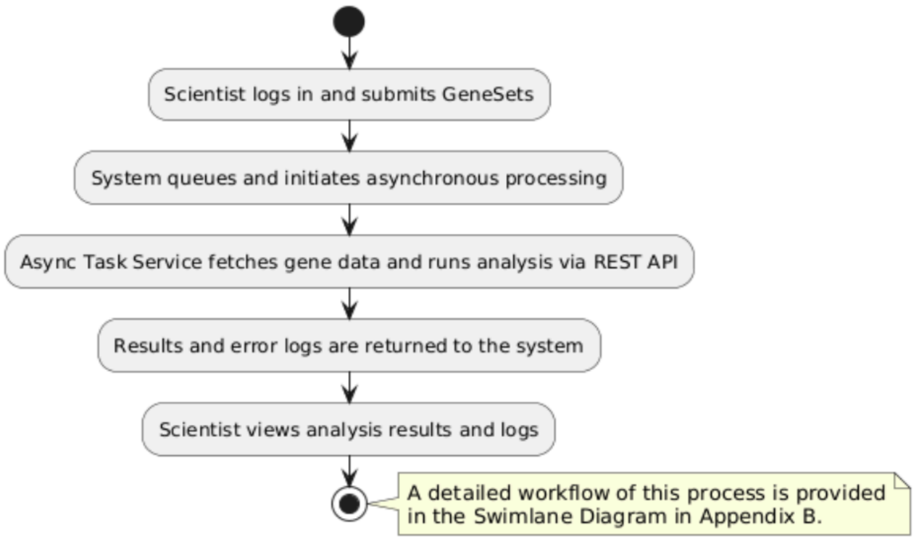
The clients and stakeholders for the project are:

1. The Jackson Laboratory team, including John Bluis - Software Manager, and Alexander Berger - Software's Technical and Architectural Lead.
2. Geneticists, Scientists, and Researchers, who will be using the software to analyze the genomic data. This includes both those at The Jackson Laboratory as well as at other organizations.
3. The Frontend Team, who will be integrating the analysis results with the UI.

2. Overall Description

2.1 Product Description

The product process orchestration is as follows -


Scientist logs in and submits GeneSets
System queues and initiates asynchronous processing
Async Task Service fetches gene data and runs analysis via REST API
Results and error logs are returned to the system
Scientist views analysis results and logs
A detailed workflow of this process is provided in the Swimlane Diagram in Appendix B.

1. The system will provide an interface to submit GeneSets for analysis.
2. The analysis will take place asynchronously for a faster runtime and concurrent processing, which will allow researchers and bioinformaticians to analyze GeneSets efficiently without waiting for each individual task to complete.
3. For better security, the software will interact with the GeneWeaver REST API instead of directly accessing the database.

2.2 Product Perspective:

The software is aimed at improving the execution of GeneWeaver tools by implementing them from scratch and enabling asynchronous execution. The system will integrate with the GeneWeaver platform by providing a plugin-based architecture. Each implemented tool will conform to a standard interface. This will allow it to process gene data and return results in a serializable format.

The asynchronous execution of tasks using the developed software will provide Geneticists and Scientists with a more efficient way to process the genomic data and view the result in a serialized format. This will reduce execution bottlenecks and improve the system scalability, ultimately enhancing the performance and usability of the GeneWeaver tools.

2.3 Constraints and Operating Environment

Constraints:

1. The tool's input and output must be serializable to ensure smooth data exchange and integration.
2. All GeneWeaver data must be accessed exclusively through the REST API; direct database access is not permitted.

Operating Environment:

1. The software will be deployed in a microservice-based environment, ensuring scalability and modularity.
2. The tool will be implemented as a plugin in the form of a Python package, enabling easy integration and extension in future.

2.4 User Documentation Requirements

The user documentations to be delivered along with the software are:

1. README.md File: This will contain instructions on steps for setting up the environment, and how to run the tool and, both with and without the ATS.
2. Code Comments: The module, class and function docstrings should follow the reStructureText (reST) format. The docstrings should include :param, :return:, and

:raises:. Inline code comments should be used liberally to describe the code and python type hints should be included to document arguments and return values.
3. Project Writeup: This document should describe the design and implementation of the project.

2.5 Assumptions & Dependencies

Assumptions:
1. The system is compatible with Python version >= 3.10
2. There is no direct database access, all data retrieval will be done through GeneWeaver REST API.

Dependencies:
1. Each tool may have specific processing and computational requirements, which will need to be considered while developing them.

2.6 User Classes and Characteristics

Scientists & Geneticists -
1. Upload GeneSets for analysis
2. View computed results for their research
3. Use JSON to interpret results & derive insights.

Frontend Development Team -
1. Use API results to render in a user friendly, easy to interpret frontend
2. Seamless integration between json responses & UI
3. Implement user friendly result interpretation.

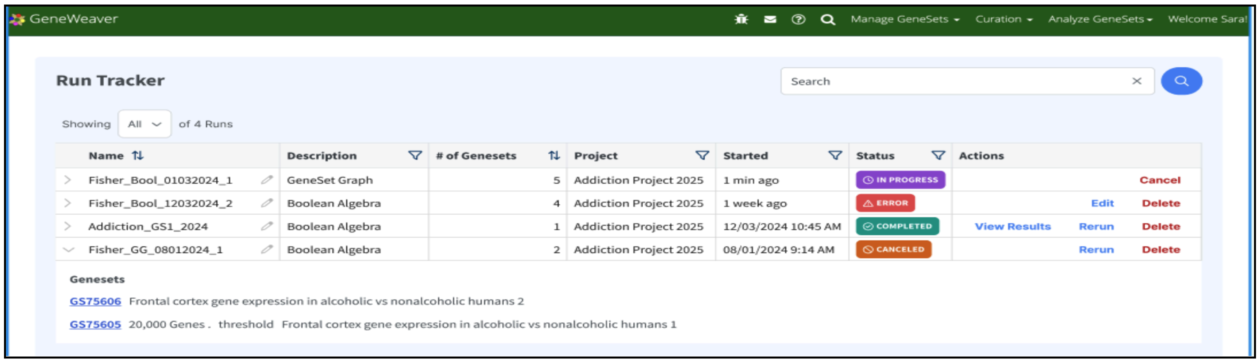
Developers -
1. Ensure efficient execution of multiple GeneWeaver tools asynchronously.
2. Maintain serialization and API constraints.
3. Debug and enhance the tool’s functionality.

3. External Interface Requirements

3.1 User Interfaces

The plugin itself lacks a visual interface, and the output is evaluated directly by the ATS. The output from the plugin is transmitted to the ATS, which then sends it to the frontend.

1. **GeneWeaver Web App**: Users interact with the UI and to submit tasks and view results. The results come from an Asynchronous task service- which calls the plugin to perform an Asynchronous task.
2. **Tool Results and Error Logs(Optional)**: Tool Results are fetched by the ATS and errors are shown in the geneweaver UI similar to how the screenshot below is represented



Run Tracker

Showing All of 4 Runs

Name	Description	# of Genesets	Project	Started	Status	Actions
Fisher_Bool_01032024_1	GeneSet Graph	5	Addiction Project 2025	1 min ago	IN PROGRESS	Cancel
Fisher_Bool_12032024_2	Boolean Algebra	4	Addiction Project 2025	1 week ago	ERROR	Edit Delete
Addiction_GS1_2024	Boolean Algebra	1	Addiction Project 2025	12/03/2024 10:45 AM	COMPLETED	View Results Rerun Delete
Fisher_GG_08012024_1	Boolean Algebra	2	Addiction Project 2025	08/01/2024 9:14 AM	CANCELED	Rerun Delete

Genesets

GS75606 Frontal cortex gene expression in alcoholic vs nonalcoholic humans 2

GS75605 20,000 Genes . threshold Frontal cortex gene expression in alcoholic vs nonalcoholic humans 1

Status of the asynchronous tasks are shown based on its progress in the backend

3.2 Hardware Interfaces

1. **Cloud Infrastructure**:
   ATS runs on cloud servers where Temporal is used for orchestration
2. **Local Execution Requirements**: For running the plugin on ATS, no specialized hardware required

3.3 Software Interfaces

The plugin must conform to an interface that allows it to receive input data and provide results data. The interface is for the ATS to use the functionalities of the plugin. The plugin itself is installed onto the ATS.
The plugin will implement a python-based interface that allows ATS to:

1. **Execute a primary function** that performs the main task (Eg. Hisem)
   Input: Contains necessary data
   Output: A serialized output(Must match the error/results schema)
2. **Retrieve status/progress**(Optional) from a secondary function during or after execution

GeneWeaver API**:**
It is an interface used for accessing, analyzing, and managing gene sets, genes, publications, and species-related data from the GeneWeaver platform. We either obtain data from this GeneWeaver API or elsewhere, depending on the specific tool we are developing.

Key Endpoint example:

1. /api/genesets/{geneset_id}: Retrieve detailed information about a specific gene set by its unique ID

3.4 Communications Interfaces

3.4.1 Protocols:

1. gRPC: Used by Temporal for task scheduling (ATS). Direct interaction with gRPC isn't required.
2. HTTP/HTTPS: For GeneWeaver API interactions.(For plugin)

3.4.2 Data Format for Communication:

1. Input/Output: JSON
2. Errors follow a particular Schema

3.4.3 Security:

Uses HTTPS for API calls.

4. System Features

4.1 Plugin for Async Task Service

4.1.1 Description and Priority

This plugin will take in data as GeneSets, or GeneSet IDs and run the GeneWeaver tools on the ATS platform. It will take in the data via an interface and will output the results via an interface. This is the main feature of our project, it has our top priority. We will be taking an already working ATS and adding a python plugin to it which will run the GeneWeaver tool(s) and return their solutions.

4.1.2 Stimulus/Response Sequences

Intake data interface function called with input GeneSets data and tool as arguments in the specified schema and structure needed. Each tool will have its own function in the interface. Inside this function the tool will interact with the input and return the output information from the tool in serializable form.

1. Implement input Interface in python plugin on ATS
2. Receive serializable input data from input interface
3. Complete tool analysis of data
4. Return completed tool analysis

4.1.3 Functional Requirements

REQ-1.1: Receive Serializable Data via tool function in intake interface with defined schema and structure.
Priority: High
Stimulus/Response Sequence: This initializes the instance of the tool in the ATS. The front end calls the function in the interface with the serialized data and the plugin in the ATS runs on that data.

REQ-1.2: Realtime tracker of process state in intake interface.
Priority: High
Stimulus/Response Sequence: Once the tool is initialized a follow up function associated with the first can be called to check the current state of the function. It will return either in process, or finished.

REQ-1.3: Return information on how the GeneWeaver tool built the process.
Priority: Low
Stimulus/Response Sequence: After the tool has run it could potentially return information on how it set up the process of running the tool.

REQ-1.4: Return completed GeneWeaver serializable process data.
Priority: High
Stimulus/Response Sequence: The tool has received the serializable data, completed the tool algorithm on the gene sets and returns the pertinent information with specified schema and structure to output interface.

REQ-1.5: Implement retrieval of GeneSets from GeneSet IDs
Priority: Medium
Stimulus/Response Sequence: When receiving input data, take in either the complete gene set themselves or the GeneSet IDs and retrieve those genesets from GeneWeaver.

REQ-1.6: Create template for use on future Tools
Priority: High
Stimulus/Response Sequence: When we have achieved the other requirements for the plugin we will set it up as a template to be used for future GeneWeaver Tools.

4.2 GeneWeaver Tools

4.2.1 Description and Priority

The plugin will have specific functions for each implemented GeneWeaver Tool which we implement from the current GeneWeaver tools. These tools have a secondary priority to the ATS.

The primary tool we will implement is the MSET tool. This tool retrieves input GeneSets, calculates the number of shared genes, and for the number of trials to compute it creates random sets without replacement out of the intersection of the first set and the gene background as well as the intersection of the second set and gene background. It compares these trial sets and calculates 'p' or how likely it is the two gene sets shared are due to randomness. The value of each trial and the 'p' value are returned.

4.2.2 Stimulus/Response Sequences

This function will be called with the input data as arguments. It will then either return failure information and reasons why, or information on how the tool was set up and the output data.

1. Function MSET Called
2. Tool receives GeneSet 1, GeneSet 2, their intersections with the gene Universe in questions as well as the number of simulations.
3. For the number of simulations these intersections are sampled without replacement, and compared.
4. A 'p' value is calculated from these comparisons.
5. Tool returns output information / potential for setup information

4.2.3 Functional Requirements

REQ-2.1: Be called with Serializable Data inputs matching tool requirements.
Priority: High
Stimulus/Response Sequence: This initializes the MSET tool with the information required for it to run. Including either the gene sets or their intersections with the gene universe.

REQ-2.2: Run analysis algorithms on the input data.
Priority: High
Stimulus/Response Sequence: For the number of simulations the intersections of the gene sets and the universe are sampled without replacement and compared. These values are saved and used to compute a 'p' value.

REQ-2.3: Return trial data matching expected schema and structure.
Priority: High
Stimulus/Response Sequence: After the algorithm has completed its calculations the p value and the trial data is made serializable and returned in a format expected by the front end.

REQ-2.4: Return error information
Priority: High
Stimulus/Response Sequence: If there are any errors in the calculations return a detailed reason why in the correct schema.

REQ-2.5: Testing using Unit Tests
Priority: High
Stimulus/Response Sequence: Implement Unit testing for the MSET and other tools that we potentially create.

REQ-2.6: Include other Geneweaver tools
Priority: Low
Stimulus/Response Sequence: After MSET has been implemented we may include other tools in this project.

5. Nonfunctional Requirements

5.1 Performance Requirements:

1. The system should be optimized for memory, storage, and CPU usage, even at the expense of execution time.
2. Each task execution of a GeneWeaver tool should not exceed 1 GB of memory usage.
3. The software should implement rate limiting controls when it sends requests to GeneWeaver REST API to prevent going over usage thresholds.

Appendix A: Glossary

Term	Definition
ATS (Asynchronous Task Service)	The system responsible for managing and executing plugins or tasks that run asynchronously.
Plugin	A self-contained module that extends or adds specific functionality to the ATS. It typically has a defined interface that the ATS can call.
Interface	The point of interaction between the plugin and the ATS. It defines how data is passed to the plugin and how results/status are returned

Appendix B: Swim Lane Diagram

The use case for the attached swim lane diagram is as follows:

Use Case:
Analyzing a GeneSet through a GeneWeaver tool

Primary Actor:
A Geneticist who wants to run gene analysis.

Goal in Context:
The geneticist wants to execute a GeneWeaver tool on selected GeneSet asynchronously and retrieve the results.

Preconditions:
1. The geneticist must be authorized to log into the GeneWeaver system
2. The Plugin, ATS, and REST API should be operational.

Trigger:
The geneticist logs into the GeneWeaver system and wants to analyze genomic data on a tool through our plugin.

Scenario:
1. The scientist logs into GeneWeaver.
2. If login is successful, the scientist selects GeneSets and a tool.
3. The scientist clicks 'Run Tool.'
4. The GeneWeaver UI submits an async request to ATS with the required data.
5. ATS checks if it is available:
    - if available, it acknowledges the request ; else, it returns an error message.
6. ATS invokes the GeneWeaver Plugin to process the request.
7. The GeneWeaver Plugin fetches gene data from the GeneWeaver REST API.
8. The GeneWeaver Plugin executes the selected tool logic.
9. If execution is successful, the plugin sends the results to ATS.
10. If execution fails, the plugin sends error details to ATS.
11. ATS stores the results.
12. The GeneWeaver UI continuously polls ATS for updates.
13. Once results are available, they are returned to the scientist.
14. The scientist views the final results or an error log.

Exception:
1. If the login fails, an error message is shown, and the geneticist must retry.
2. If ATS is unavailable, an error message is displayed, and the geneticist must retry later.

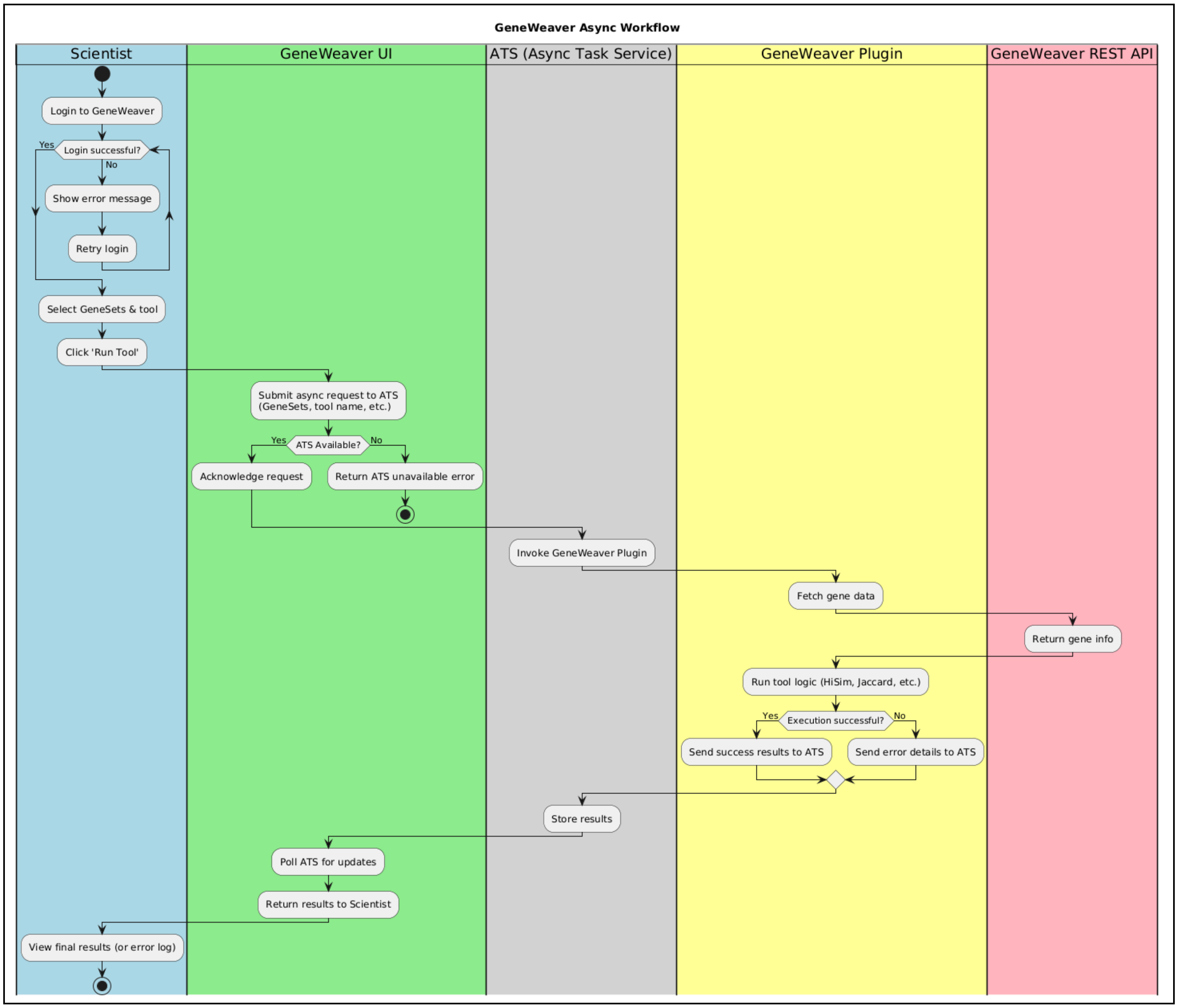
Priority:
High - since this is the primary use case of the software.

Frequency of Use:
Frequent.

Channel to Actor:
GeneWeaver UI (accessible via browser)

Open Issues:
The process takes up more than expected resources.

Swim Lane Diagram


GeneWeaver Async Workflow
Scientist
GeneWeaver UI
ATS (Async Task Service)
GeneWeaver Plugin
GeneWeaver REST API
Login to GeneWeaver
Login successful?
Yes
No
Show error message
Retry login
Select GeneSets & tool
Click 'Run Tool'
Submit async request to ATS (GeneSets, tool name, etc.)
ATS Available?
Yes
No
Acknowledge request
Return ATS unavailable error
Invoke GeneWeaver Plugin
Fetch gene data
Return gene info
Run tool logic (HiSim, Jaccard, etc.)
Execution successful?
Yes
No
Send success results to ATS
Send error details to ATS
Store results
Poll ATS for updates
Return results to Scientist
View final results (or error log)