Bilkent University

Department of Computer Engineering

Senior Design Project

T2522

CASSIE

Analysis and Requirement Report

Ege Şirvan, 22102289, ege.sirvan@ug.bilkent.edu.tr
Eren Aslan, 22102329, eren.aslan@ug.bilkent.edu.tr
Arda Kırcı, 22002031, arda.kirci@ug.bilkent.edu.tr
Emre Can Yoloğlu, 22102542, can.yologlu@ug.bilkent.edu.tr
Osman Baktır, 22002553, osman.baktir@ug.bilkent.edu.tr

Supervisor

Can Alkan

Course Instructors

İlker Burak Kurt

Mert Bıçakçı

19.12.2025

1. Introduction

CASSIE is a cloud-based genome assembly and annotation platform that aims to automate complex genome analysis processes and make large-scale genome assembly procedures more accessible to researchers. Modern genome assembly workflows require multiple complex tools to be integrated, highly compute-intensive steps to be managed, and proficiency in using the required tools.

CASSIE eliminates this complexity by presenting a streamlined environment where users can upload their data, select appropriate tools/steps, and run complete assembly workflows from a simple web interface with ease. CASSIE addresses a wide range of use cases by supporting both human and nonhuman genomic data and appropriate tools, and embedded privacy policies.

CASSIE automates the fundamental steps of a standard assembly process: Quality control, estimations for genomic properties, assembly, assembly quality assessment, and, optionally, fundamental repeat annotations and gene annotations. All required tools have been containerized using Docker, and the workflow is managed by Nextflow. This architecture allows reproducibility, modularity, flexible scalability, and adaptability.

CASSIE's infrastructure is designed to go with a scalable cloud architecture. During the first part of the development phase, a custom-built emulator is used to simulate an AWS environment. This emulator uses Docker for Virtual Machines (VMs), database, and genomics tools, MinIO for S3-compatible storage, and Minikube for the Kubernetes-compatible workflow orchestration and pod/namespace management. During the second part of the project, AWS and its tools will be used. EC2 instances of different compute and storage levels will be used as easily scalable VMs. Docker will be used for Dockerized genomics tools, S3 will be cloud-based storage, and Kubernetes will be used for workflow orchestration and pod/namespace management. CASSIE helps researchers focus directly on scientific results instead of dealing with calculation infrastructure by abstracting these operational complexities completely from the end user.

2. Current System

The rapid expansion of genomics and the increase in the need for biological analyses have necessitated the development of accessible computational platforms that are capable of processing large-scale biological datasets without requiring proficiency in the tools. To meet this demand, web-based workflow management systems like Galaxy[1] have become the industry standard. These general-purpose platforms have successfully opened the world of bioinformatics to everybody by providing a graphical interface for tools across various domains.

However, the one-platform-for-everything approach of these systems introduces significant complexities for specialized tasks like genome assembly. Since they are designed to support every pipeline, they place the burden of orchestration entirely on the user. Researchers often need to manage a tool salad, a confusing set of software options that must be manually selected, configured, and wired together. The lack of specialization requires users to have deep domain expertise to avoid compatibility errors. Furthermore, these platforms often operate as black boxes and can't provide transparency about the computational cost and duration of a job before execution, which leads to inefficient time management and budget uncertainty for the users.

CASSIE addresses these critical gaps by moving the paradigm from a general-purpose workbench to a specialized, automated workflow manager for genome assembly. Unlike the traditional systems that require manual pipeline construction, CASSIE changes this by utilizing a goal-oriented interface where users can define their biological objectives. As CASSIE is based on a cloud-native architecture built on Docker and Nextflow, it automates pipelines including quality control, assembly, assembly analysis, and annotation while ensuring full reproducibility. Uniquely, it introduces a new time and price estimation step before the workflow execution, which makes the genome assembly pipeline more accessible and predictable.

CASSIE's infrastructure is designed to operate on a scalable cloud architecture. During the early development phase, a custom-built emulator is used to simulate cloud behavior and execution environments, enabling controlled experimentation and testing. In later stages, the system is intended to be deployed on a real cloud platform, utilizing scalable compute, storage, and orchestration services. By abstracting infrastructure and resource management

from the end user, CASSIE allows researchers to focus directly on scientific analysis rather than computational complexity.

3. Proposed System

3.1. Overview

Cassie is a cloud-native bioinformatics platform designed to automate, standardize, and simplify genome assembly and annotation execution workflows for researchers. It is proposed to solve the needs of the growing complexity of modern genomic analysis pipelines, which require users to manually install, configure, and orchestrate multiple command-line tools, manage computational resources, and ensure reproducibility across their experiments. These manual tasks are time-consuming, error-prone, and make it much harder for researchers who want to focus on biological inference rather than software tools infrastructure management.

The primary objective of CASSIE is to provide an integrated environment to its users, where sequencing data can be uploaded, and complex bioinformatic workflows can be executed through a unified interface. CASSIE supports common genomic data formats, and the system is designed to handle large datasets generated by next-gen sequencing technologies. Abstracting away low-level operational details allows CASSIE to help initiate analyses without requiring extensive knowledge of cluster management, containerization, or workflow engines for the users.

The proposed system is an automated workflow orchestration mechanism that coordinates genome assembly, quality control, and annotation tasks. Workflows are executed using industry-standard bioinformatics tools that are packaged and managed in a reproducible manner. CASSIE does not provide any novel assembly or annotation algorithms. It is basically a system for building and executing bioinformatics pipelines. With this approach, it tries to keep the scientific validity intact while trying to reduce setup complexity and execution errors for the users.

The system utilizes Docker to encapsulate the bioinformatics tools and their dependencies. Workflows are handled by a workflow engine, Nextflow, enabling consistent execution across different environments. CASSIE is designed to support both development and production deployments, where workflows can be tested locally and later executed at scale in a cloud environment using orchestration platforms such as Kubernetes without modification. By

separating workflow definitions and execution environments, CASSIE aims to enhance portability and reproducibility.

CASSIE is a web-based platform. Users can interact with the system through a browser interface to upload data, configure workflows, monitor execution progress, and retrieve their results. The backend services, implemented primarily using Python, are designed to manage job scheduling, data storage, and communication with the cloud computation service, AWS. This architecture is suitable for concurrent use of multiple users while maintaining isolation between individual workflow executions.

The output of workflow executions may include assembled genomes, annotation files, logs, and summary reports based on the analysis goals chosen by the user. These results are stored in a structured manner using S3 and can be downloaded by the user for further offline analyses. By preserving workflow configurations and execution metadata, the system supports reproducibility and allows users to re-run analyses with identical or modified parameters.

Considering the privacy-sensitive nature of genomic data, security and data protection are very crucial for CASSIE. Therefore, it includes access control mechanisms to ensure that users can only view and manage their own data. Data transfers and storage are handled in a controlled manner, and the system is designed to support deployment configurations that comply with institutional and regulatory data governance requirements.

In addition to workflow configuration and execution, CASSIE provides a community-driven workflow sharing mechanism. Users can build workflows interactively with building components and publish them to a shared community space, allowing other users to discover, reuse, and adapt validated pipelines. Community workflows share only workflow structure and configuration parameters, while user datasets and execution environments remain isolated. This feature promotes reproducibility, collaboration, and knowledge reuse without compromising data privacy.

CASSIE is designed to be modular and extensible by using containerized tools and workflow-based execution models. New bioinformatics tools or research norms can be incorporated without the need for significant changes in the system. With this flexibility, the

platform can adapt novel methodologies in sequencing technologies and bioinformatics methodologies.

While the system automates workflow execution and management, it is intended as a mere support tool. Interpretation of results and scientific decision-making remain entirely the responsibility of the researcher.

3.2. Functional Requirements

Requirements below define the core functionality that the CASSIE system must fulfill. Requirements are delivered in a numbered, transparent, and traceable way.

User and Access Requirements

1. The system should provide an authentication mechanism in the web app that allows users to log in securely

2. Each submitted job should be executed within its own dedicated Kubernetes namespace to isolate it from other processes and data from other jobs.

3. Each user's data storage space on S3 should be configured independently and isolated from other users.

4. The system should allow users to cancel or terminate a running workflow execution.

Data Upload and Management Requirements

1. The system should support uploading genomic sequencing files in formats such as FASTQ and FASTA

2. Uploaded data should be automatically stored in the isolated storage area assigned to the corresponding user.

3. Formats of the uploaded files should be verified by the system, and the user should be informed when unsupported or invalid formats are detected.

4. The system should allow users to transfer data directly from their own cloud storage services into CASSIE's storage without them having to download the data to their local machines.

Workflow Configuration Requirements

1. Users should be able to choose which bioinformatics tools will be used in a workflow.

2. The system should automatically display the required parameter options, based on the selected tools.

3. User-provided parameters should be checked for validity, and warnings should be issued by the system for missing/incorrect inputs.

4. The system should allow multiple workflows with different configurations to be executed on the same data.

5. The system should be capable of asking high-level questions about the intended analysis goals and automatically selecting appropriate tools.

6. Users should be able to manually construct custom workflows without relying on automated tool selection.

7. The system should be able to estimate expected execution time and total cost for all available EC2 instance classes, each offering different CPU and memory capacities.

8. Based on estimated requirements and execution time, the system should recommend a suitable EC2 instance class.

9. Users should be able to manually select an EC2 instance class if desired, by indicating time and price estimations for their EC2 instance class choice.

10. The finalized workflow should be executed on the EC2 instance class selected by the user.

Workflow Execution Requirements

1. The system should submit configured workflows to Nextflow for execution using Kubernetes.

2. Each workflow step (e.g., quality control, assembly, assessment) should be executed as a separate Kubernetes pod.

3. Kubernetes should schedule pods according to defined CPU and memory requests and limits, as well as namespace-specific resource quotas.

4. Temporary and intermediate files, except the tool outputs, should be managed carefully and removed after execution.

5. Nextflow should be configured to automatically retry failed tasks according to a predefined retry policy.

6. The system should record metadata such as execution duration and CPU/memory usage levels.

Post-Execution Requirements

1. The system should record and display the execution status (started, running, completed, failed) of each workflow step and notify the user accordingly.

2. All final output files should be written to the user's designated storage directory.

3. Upon completion, users should be able to download result files individually or as a single compressed archive.

4. The system should provide users with readable and accessible workflow execution logs.

Tool Requirements

1. The system should support quality control analyses for sequencing data.

2. The system should support estimation of genome size, heterozygosity, and repeat content.

3. The system should support genome assembly operations.

4. The system should support scaffolding, polishing, and gap-filling steps.

5. The system should support the assessment of assembled genomes.

6. The system should support repeat and segmental duplication analysis.

7. The system should support gene annotation workflows.

Community Workflow Requirements

1. The system should allow users to publish workflows they have created to the community.

2. Published workflows should include descriptive metadata such as name, description, tags, and author information.

3. The system should ensure that publishing a workflow does not expose any user datasets or execution outputs.

4. The system should provide a community catalog where users can browse shared workflows.

5. The system should allow users to search and filter community workflows based on metadata.

6. The system should display popularity indicators such as vote count and usage count for community workflows.

7. The system should allow users to import a community workflow into their private workspace.

8. Imported community workflows should be editable independently of the original workflow.

9. The system should allow users to vote for community workflows.

10. The system should maintain attribution to the original workflow author.

System Integration Requirements

1. In a production deployment, the system should integrate with AWS S3 for storage and AWS EKS for Kubernetes-based execution.

2. The system should be capable of retrieving container images from Docker Hub or a private container registry

3. The system should enforce both namespace-level and storage-level isolation to support a secure multi-tenant architecture.

3.3. Nonfunctional Requirements

This section defines the criteria that CASSIE must meet in terms of usability, reliability, and scalability. These requirements include the core features that impact the system's user experience and operational stability.

Usability

1. The system should provide a clear and simple web interface that enables users to run genome assembly workflows step-by-step with ease.

2. The interface should only show the parameters for the related tools and reduce unnecessary complexity.

3. The error messages should be presented in a way that is readable, clear, and guides the user towards the correct solution.

4. Workflow status (ready, working, completed, error) should be output to the user via real-time or near-real-time updates.

5. The users should be able to access previous jobs, results, and logs from a single panel.

6. The users should be able to access and execute workflows from other users if they have shared them.

7. The system should provide contextual help and information about tools and parameters.

Reliability

1. The system should give readable error notifications in the event of a failure in the workflow steps.

2. User data shall be stored in a durable storage layer to prevent data loss in the event of network outages or system failures.

3. Critical system services shall run under Kubernetes controllers that automatically recreate pods if they terminate unexpectedly.

4. Error handling and retry mechanisms should be present when a connectivity error occurs between system components (ex., Nextflow → S3 access).

5. All workflows shall be designed to produce consistent and reproducible results when re-executed with the same parameters, tool versions, reference data versions, and random seeds.

Scalability

1. The system should support multiple workflows started by multiple users.

2. The Kubernetes cluster should provide horizontal scalability by adding new nodes when system load increases.

3. User-specific namespace architecture should guarantee resource isolation even under heavy load.

4. The storage system (S3) should be configured in a way that allows high volume and large numbers of concurrent read/write operations.

5. The system architecture should be designed with the flexibility to increase the number of tools if needed.

6. The system should be able to work with both small and large data.

3.4. Pseudo Requirements

This section outlines pseudo-requirements that describe technologies and tools that will be used during the implementation process of the CASSIE project. These technologies and tools are mostly set in stone; however, they are still open to change and could evolve as the project progresses.

1. Python will be used to implement backend services, including workflow orchestration, client-server communication, and the custom-built emulator.
2. PostgreSQL will be used for AWS-compatible storage of user information, workflow configuration, and job management.
3. A JavaScript-based frontend framework, such as React, will be used for a web-based UI.
4. Nextflow will be used to manage and execute the genomics workflows in a reproducible manner.
5. Docker will be used to containerize the bioinformatics tools, which will provide scalability and reproducibility. Also, Docker will be used for containerizing the parts of the custom-built emulator.
6. Kubernetes will be used to orchestrate the workflow execution, manage computational resources, and provide isolation through namespaces and pods.
7. AWS will be used as the cloud architecture service of the project, which will provide scalable compute (EC2) and storage (S3) resources.
8. MinIO will be used to emulate an S3-compatible object storage tool in the custom-built emulator.
9. Machine learning libraries such as PyTorch and Google Colab will be used to develop, train, and evaluate time and price estimator models.

3.5. System Models

3.5.1. Scenarios

Scenario 1: Uploading Sequencing Files and Isolated Storage Placement

a) Actors: User (Researcher), CASSIE system, AWS S3

b) Steps:

1) The user logs into CASSIE through the authentication interface

2) The user uploads the required files.

3) CASSIE validates the file formats and rejects invalid uploads with feedback.

4) CASSIE stores the uploaded data in the user's isolated S3 storage area.

5) The user confirms that the data they uploaded appears under their data list and is ready for workflow execution.

Scenario 2: Direct Cloud-to-CASSIE Data Transfer

- **a)** Actors: User (Researcher), CASSIE system, External Cloud Storage Provider, AWS S3
- **b)** Steps:

  1) The user logs into CASSIE and chooses the "Import from Cloud Storage" action.

  2) The user provides the required access method.

  3) CASSIE transfers selected data from the user's cloud storage to CASSIE's storage.

  4) The system displays the transfer progress.

  5) After completion, the imported data is stored in the user's isolated S3 directory and becomes available.

Scenario 3: Manual Workflow Construction and Parameter Validation

- **a)** Actors: User (Researcher), CASSIE system
- **b)** Steps:

  1) The user selects a dataset and chooses to build a workflow manually (choose without automation action)

  2) The user selects the desired tools (e.g., QC, assembly, assessment, annotation)

  3) Based on selected tools, CASSIE automatically displays the required parameter fields

  4) The user enters parameters and submits the configuration for validation

  5) CASSIE detects/invalid parameters and notifies the user until the configuration is acceptable.

  6) The user saves the workflow configuration and proceeds to execution planning

Scenario 4: Automated Tool Selection via High-Level Questions

- **a)** Actors: User (Researcher), CASSIE system
- **b)** Steps:

  1) The user provides data to be used, workflow name, and workflow description.

  2) CASSIE asks high-level questions about the user's analysis goals.

  3) The user answers the questions without selecting tools directly.

  4) CASSIE selects an appropriate set of tools and proposes a workflow plan.

  5) The user reviews the selected tools and parameters and confirms or adjusts them.

  6) The system moves to resource estimation and execution planning.

Scenario 5: Resource Estimation and EC2 Instance Class Recommendation

- **a)** Actors: User (Researcher), CASSIE system, AWS EC2 metadata/pricing source
- **b)** Steps:

  1) The user finalizes a workflow configuration for a chosen dataset.

  2) CASSIE estimates the expected execution time and total price for available EC2 instance classes.

  3) CASSIE recommends a suitable instance class based on estimated requirements and expected runtime/cost.

  4) The user selects either the recommended class or a different class manually.

  5) CASSIE confirms the selected execution class and prepares the workflow submission

Scenario 6: Workflow Execution with Namespace Isolation and Pod-Based Tasks

- **a)** Actors: User (Researcher), CASSIE system, Nextflow, Kubernetes (EKS), AWS S3
- **b)** Steps:

  1) The user starts the workflow execution

  2) CASSIE assigns a dedicated Kubernetes namespace to the job to ensure isolation

  3) Nextflow launches each workflow step as one Kubernetes pod.

  5) Kubernetes schedules pods according to resource requests/limits and namespace quotas.

  6) Intermediate outputs are written to a temporary storage, workflow-specific, and associated with the namespace

  7) If a pod fails, Nextflow retries the failed task according to the retry policy.

  8) The user monitors the job status through CASSIE during execution

Scenario 7: Completion, Result Delivery, and Download Options

- **a)** Actos: User (Researchers), CASSIE system, AWS S3
- **b)** Steps:

  1) The workflow finishes successfully, and CASSIE marks each step as completed.

  2) All output files are written to the user's storage directory.

  3) CASSIE cleans up temporary and intermediate files present in the workflow namespace

  4) CASSIE shows the user the result files alongside logs created by the tools for transparency.

  5) The user can download outputs or see visual diagrams related to the executed analyses.

Scenario 8: Failure Handling, Logs, and Retry/Resubmission

- **a)** Actors User (Researcher), CASSIE system, Nextflow, Kubernetes (EKS)
- **b)** Steps:
  1) During execution, one workflow step fails.
  2) Nextflow automatically retries the failed task based on its retry policy.
  3) If a maximum number of retries is exceeded, CASSIE marks the workflow as failed and records its error status.
  4) The user accesses readable workflow logs and error messages through the interface.
  5) The user adjusts parameters and/or selects a different EC2 instance class.
  6) The user resubmits the workflow using the same data without re-uploading the data.

Scenario 9: Publishing a Workflow to the Community

- **a)** Actors: User(Researcher), CASSIE System
- **b)** Steps:
  1) The user creates or loads a workflow using the Pipeline Builder.
  2) The user selects the option to publish the workflow to the community.
  3) The system prompts the user to provide information about the created workflow.
  4) The system validates that no private data is included.
  5) The system publishes the workflow to the community site.
  6) Workflow becomes visible to other users.

Scenario 10: Reusing a Community Workflow

- **a)** Actors: User (Researcher), CASSIE System
- **b)** Steps:
  1) The user opens the Community page.
  2) The system displays a list of community workflows with workflow information and popularity indicators.
  3) The user selects a workflow and views its details.
  4) The user chooses to import the workflow.
  5) The system creates a private copy of the workflow.
  6) The user customizes and executes the workflow.

3.5.2. Use Case Model

Figure 1. System-Level Use Case Diagram for the CASSIE Platform

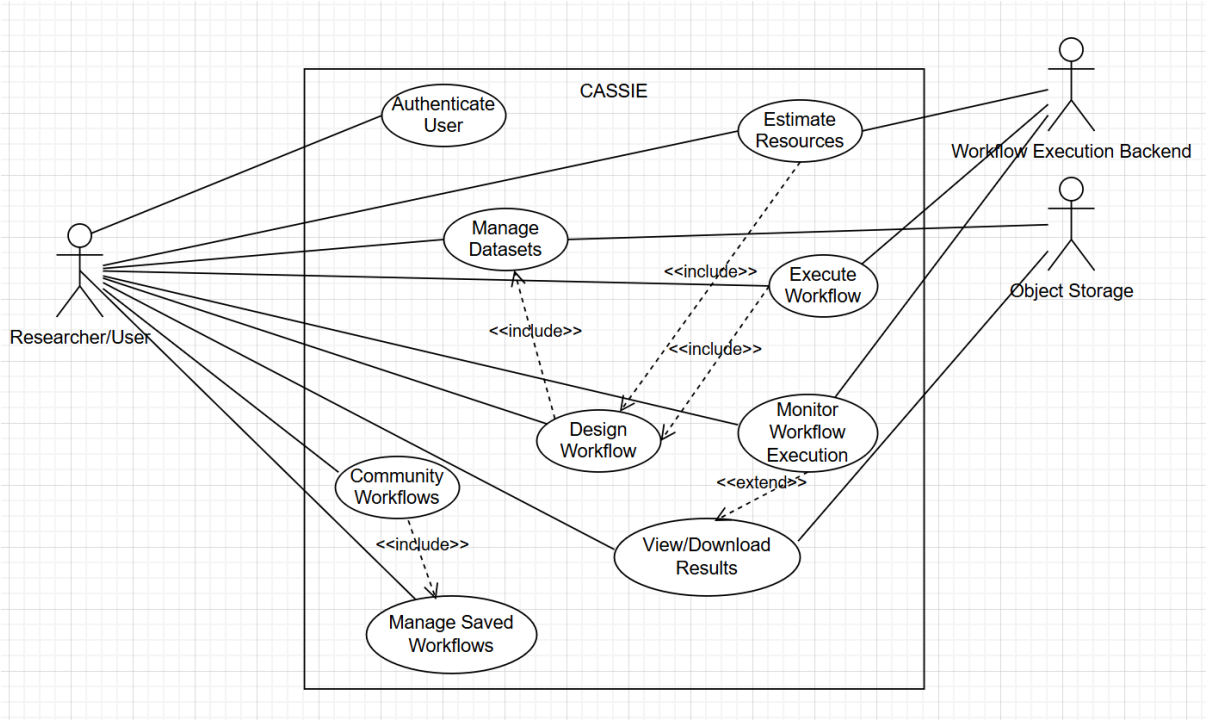
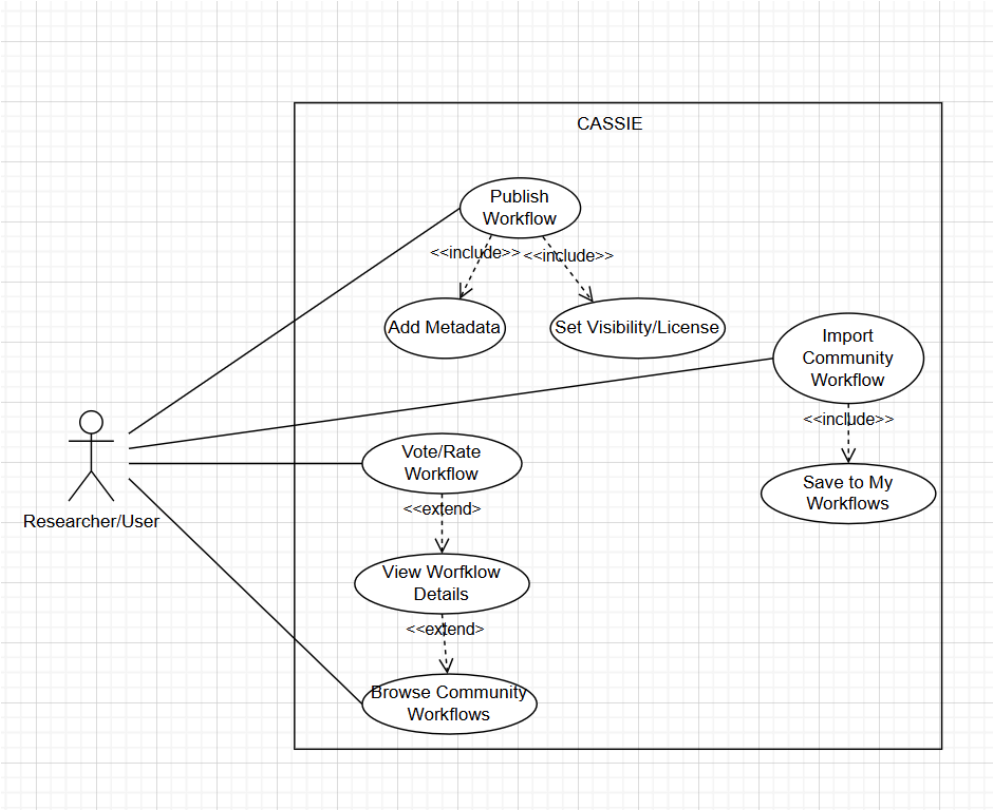
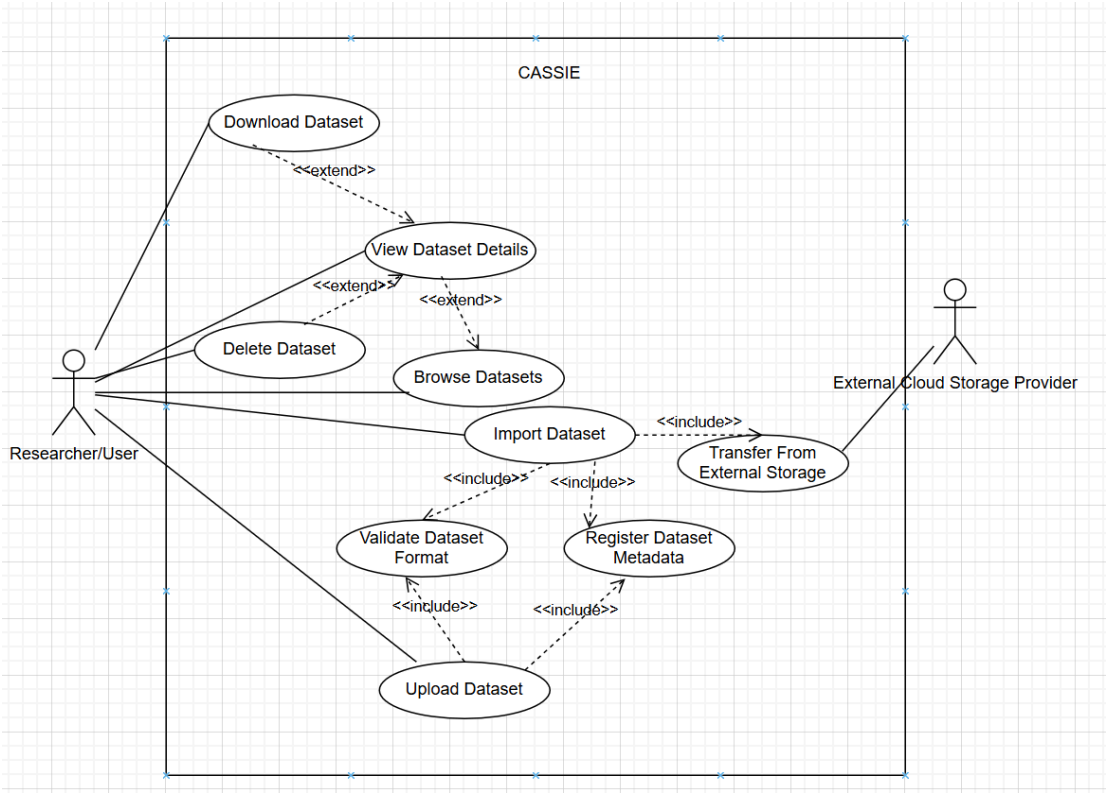


Figure 2. Detailed Use Case Diagram for Managing Community Workflows

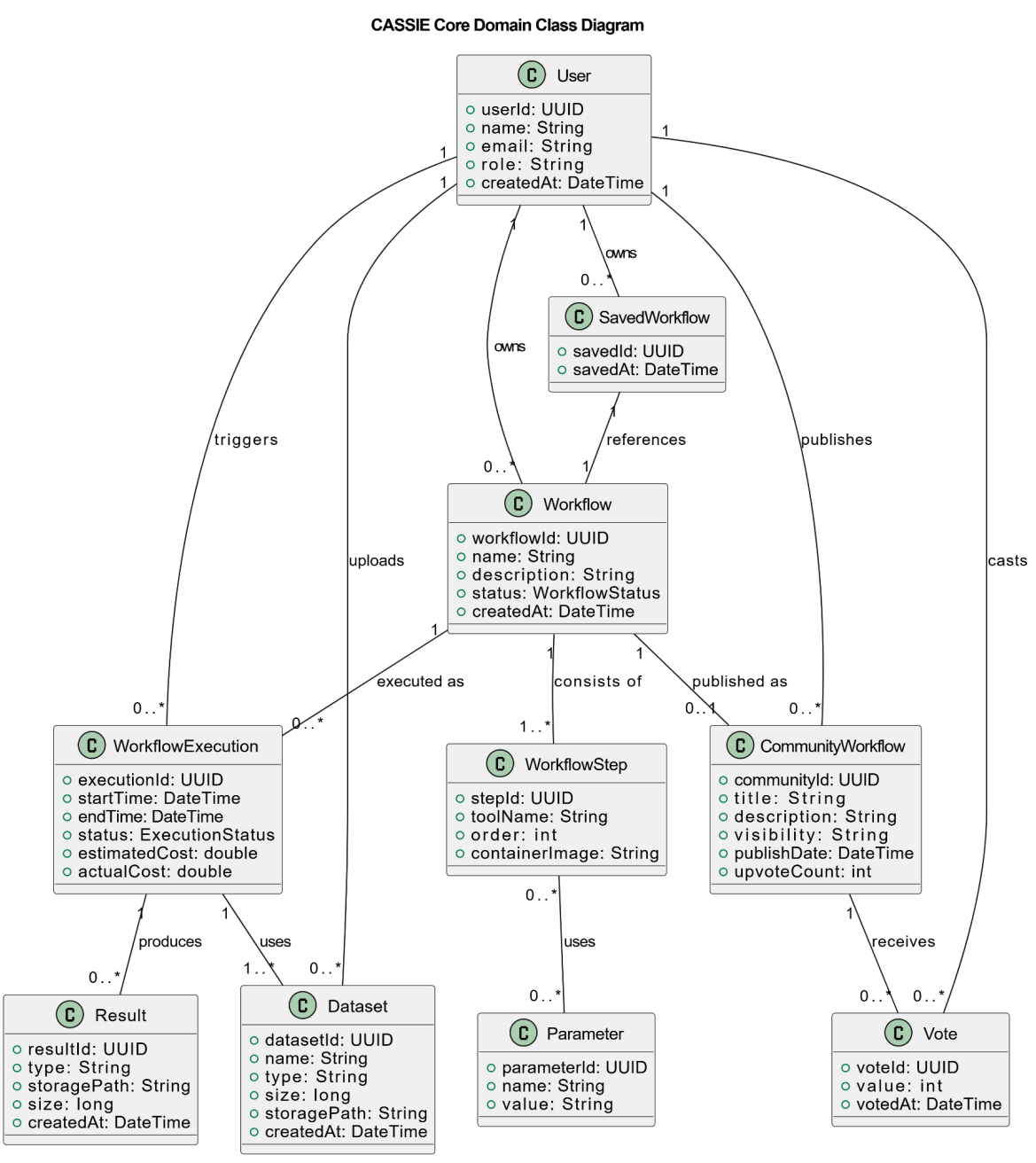


Note: This diagram focuses on user interactions for community workflow management; execution and storage infrastructure are outside the scope of this use case detail.

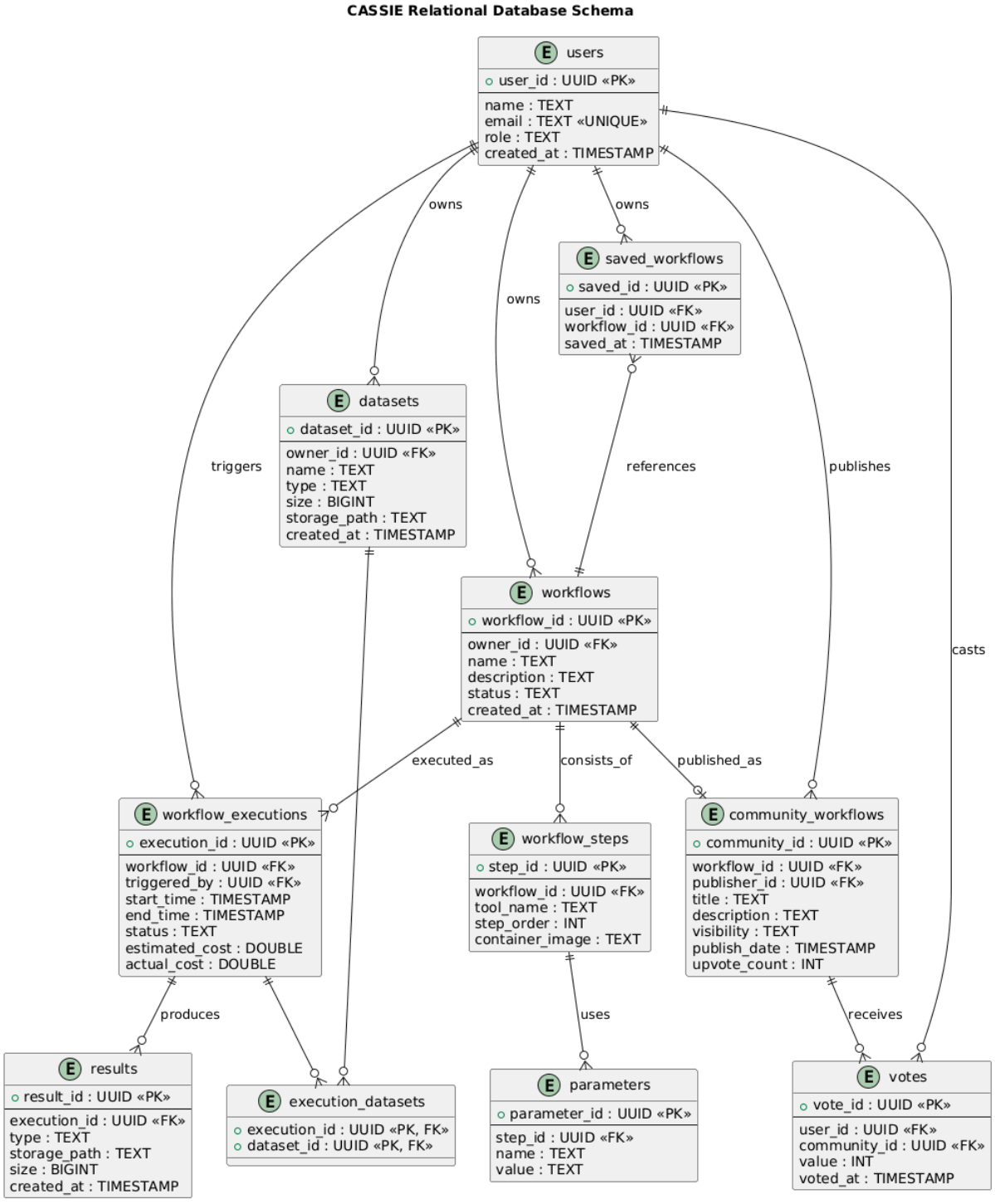
Figure 3. Detailed Use Case Diagram for Managing User Datasets



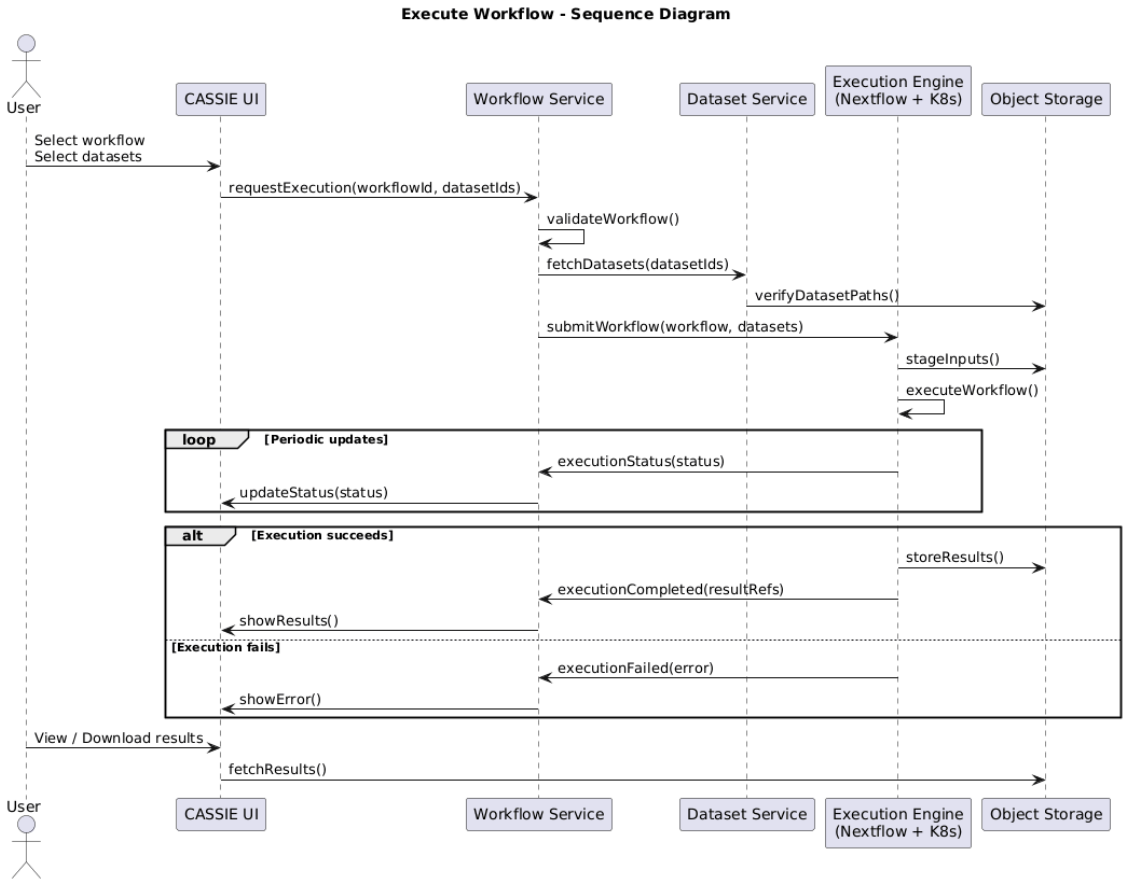
3.5.3. Object and Class Model

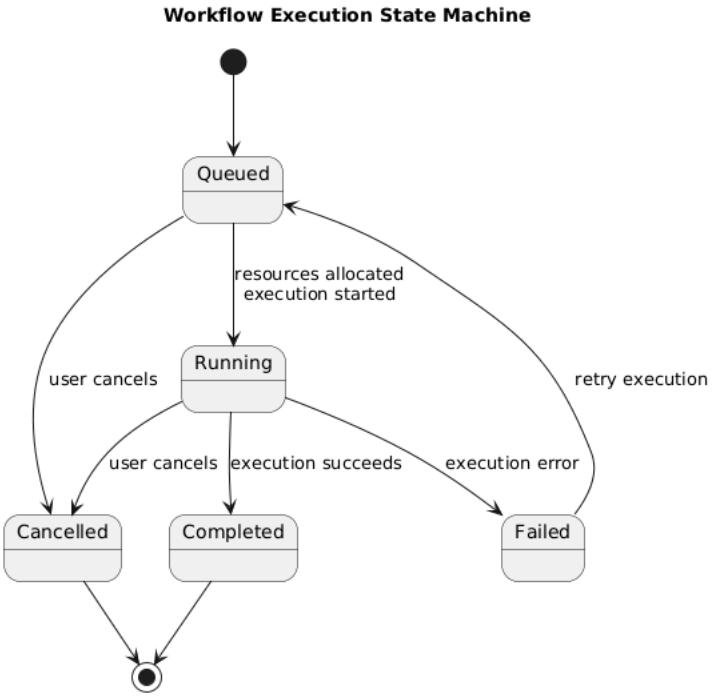


3.5.3.1. SQL Tables



3.5.4. Dynamic Models



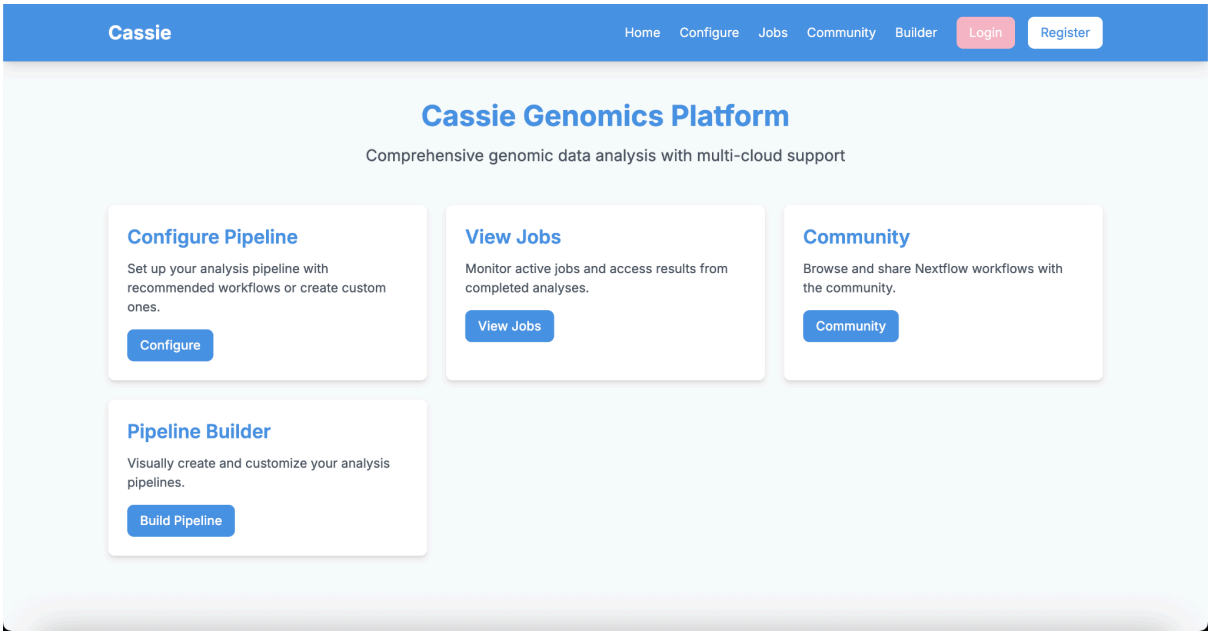


3.5.5. Endpoint Model

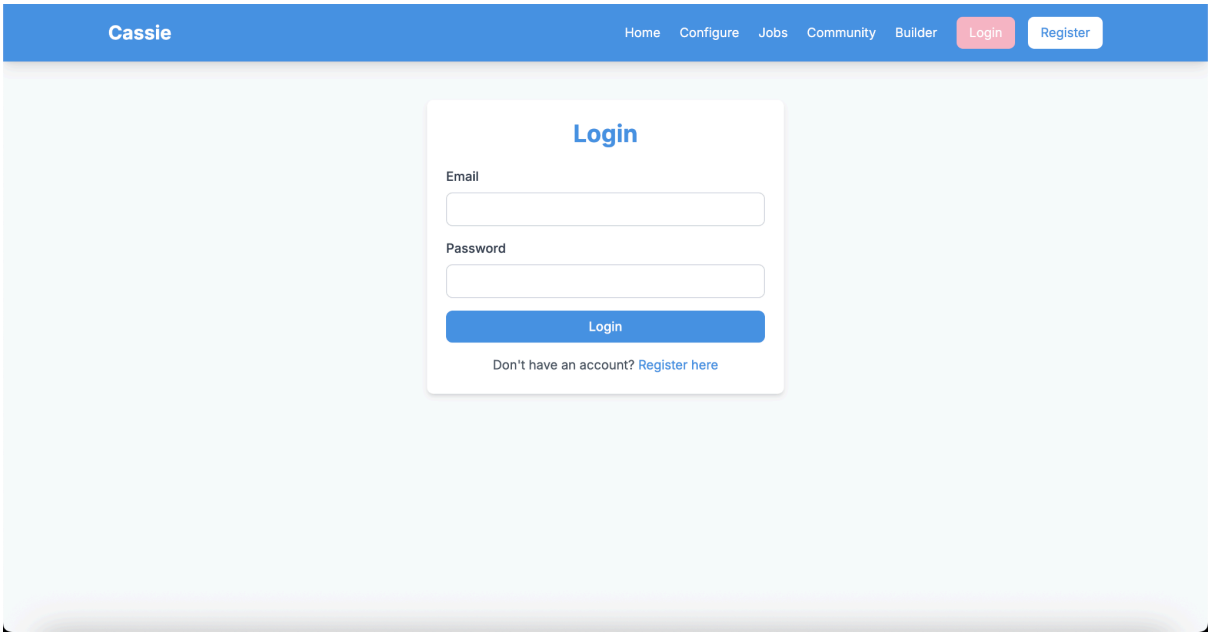
Endpoint	Primary Use Case(s)	Main DB Tables
POST /auth/login	Authenticate User	users
GET /auth/me	Authenticate User	users
GET /workflows	Manage Workflows	workflows, workflow_steps, parameters
POST /workflows	Design Workflow	workflows, workflow_steps, parameters
PUT /workflows/{id}	Edit Workflow	workflows, workflow_steps, parameters
DELETE /workflows/{id}	Delete Workflow	workflows, workflow_steps, parameters
GET /datasets	Manage Datasets	datasets
POST /datasets/upload	Upload Dataset	datasets
POST /datasets/import	Import Dataset	datasets
POST /executions	Execute Workflow	workflow_executions, execution_datasets
GET /executions/{id}	Monitor Workflow	workflow_executions
POST /executions/{id}/cancel	Cancel Workflow	workflow_executions
POST /executions/{id}/retry	Retry Workflow	workflow_executions
GET /results/{executionId}	View Results	results
GET /results/{resultId}/download	Download Results	results
GET /community/workflows	Browse Community Workflows	community_workflows, users
POST /community/workflows	Publish Workflow	community_workflows, workflows
POST /community/workflows/{id}/import	Import Community Workflow	saved_workflows, workflows
POST /community/workflows/{id}/vote	Vote Workflow	votes, community_workflows

3.6 User Interface - Navigational Paths and Screen Mock-ups

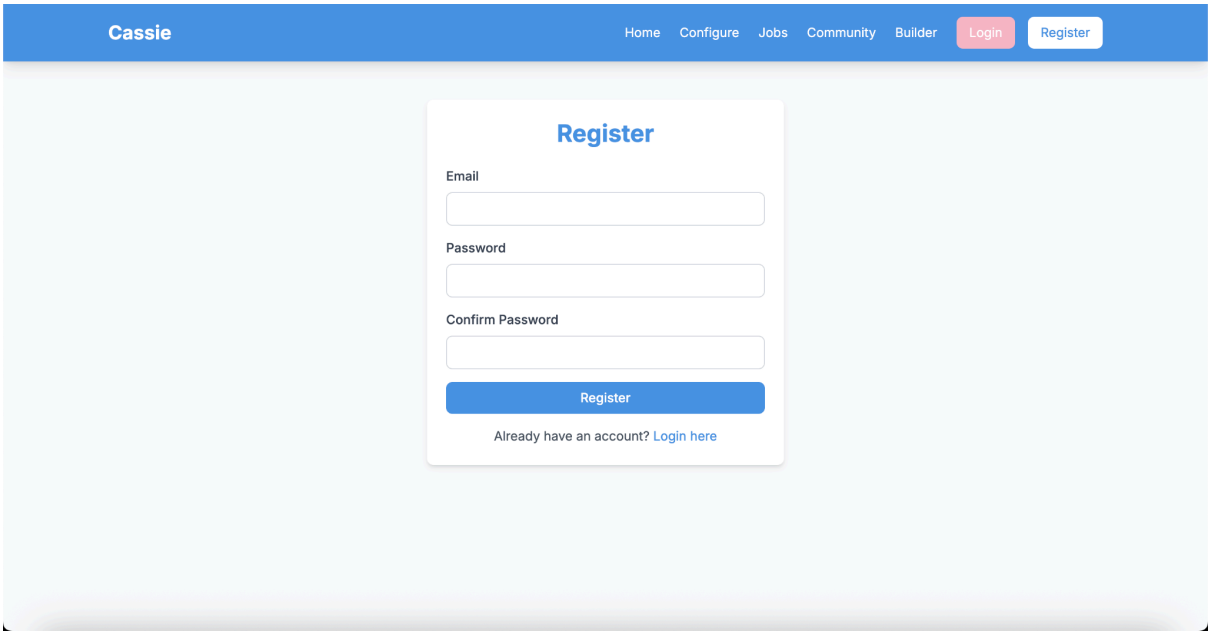
Home Page



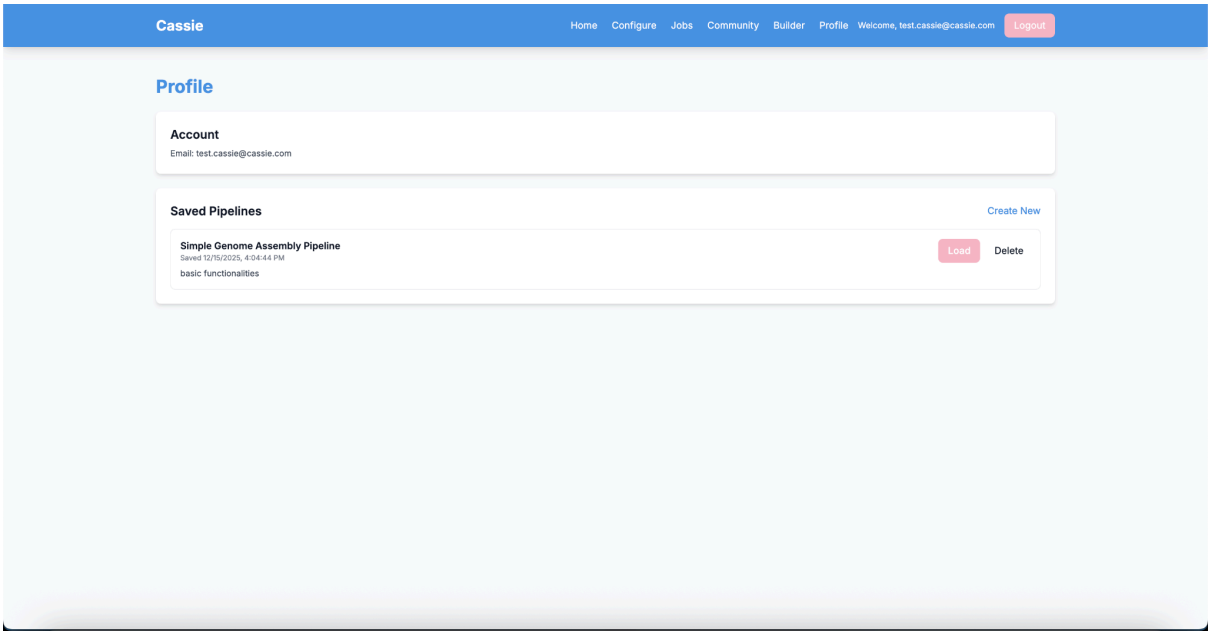
Authentication / Login



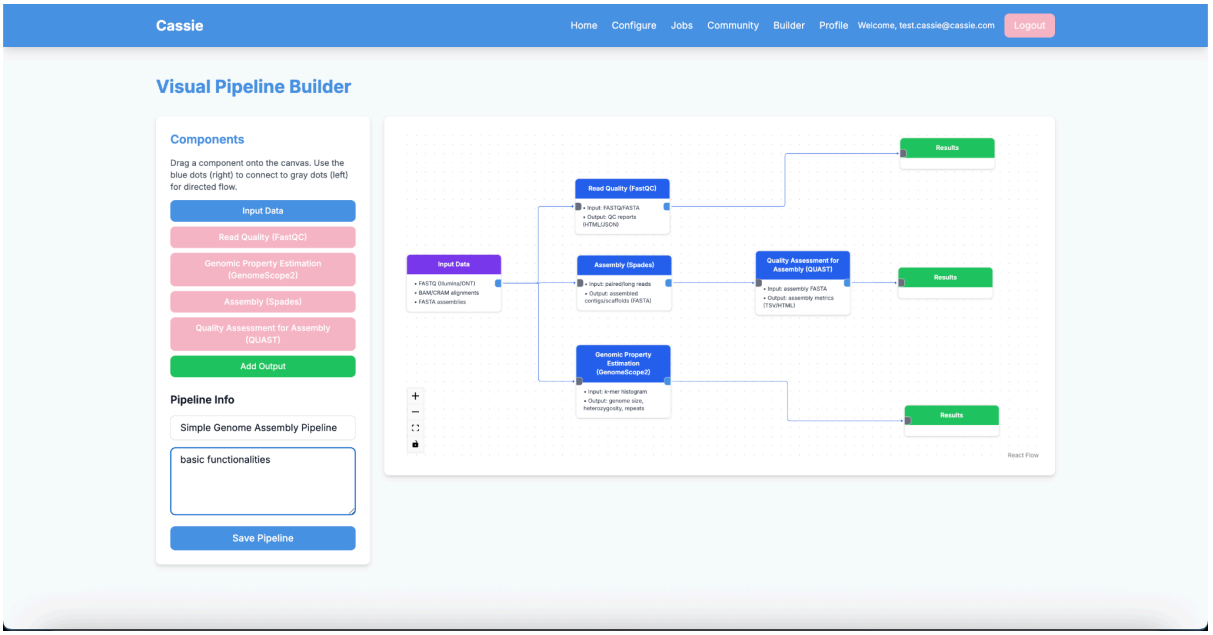
Authentication / Sign Up



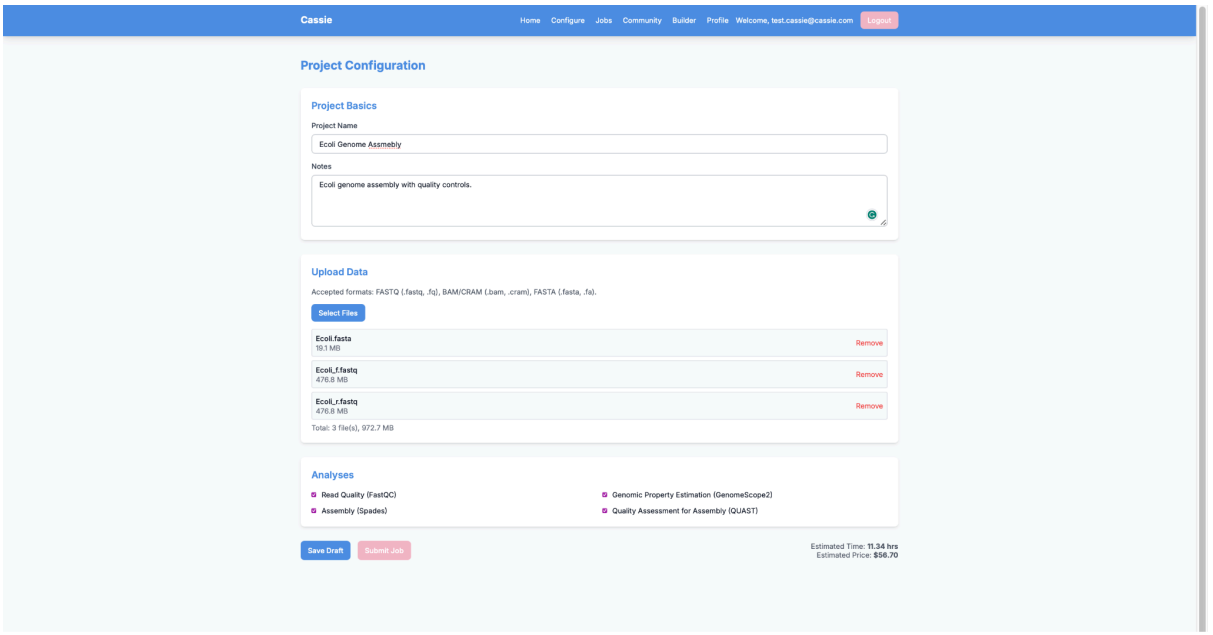
Profile Page



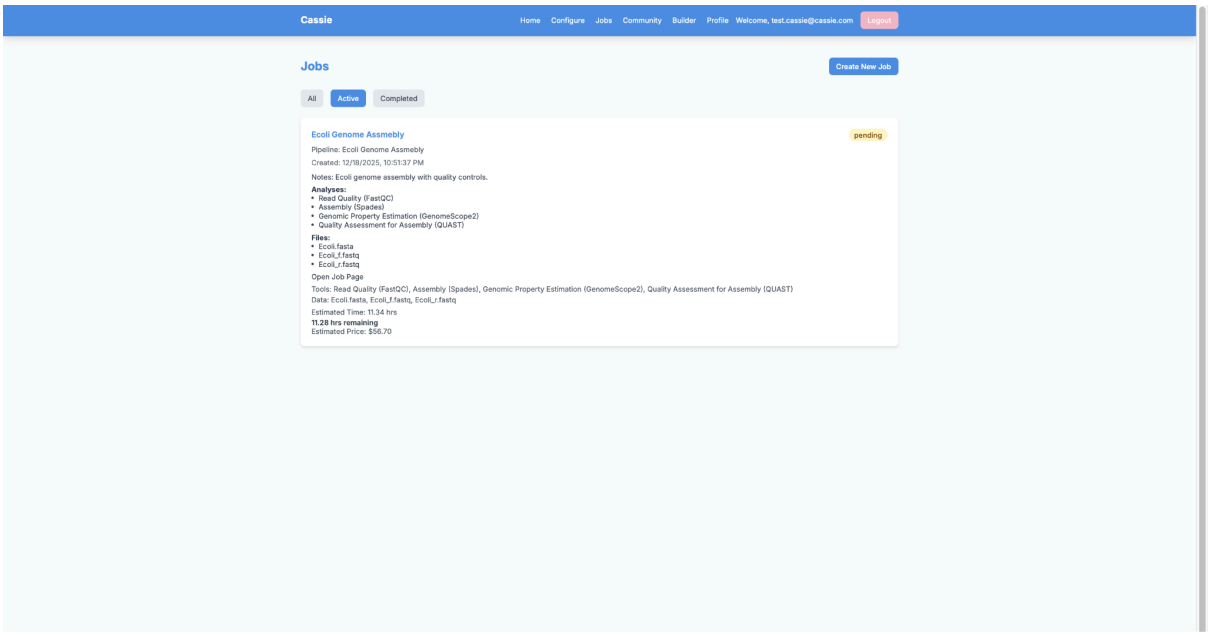
Pipeline Builder



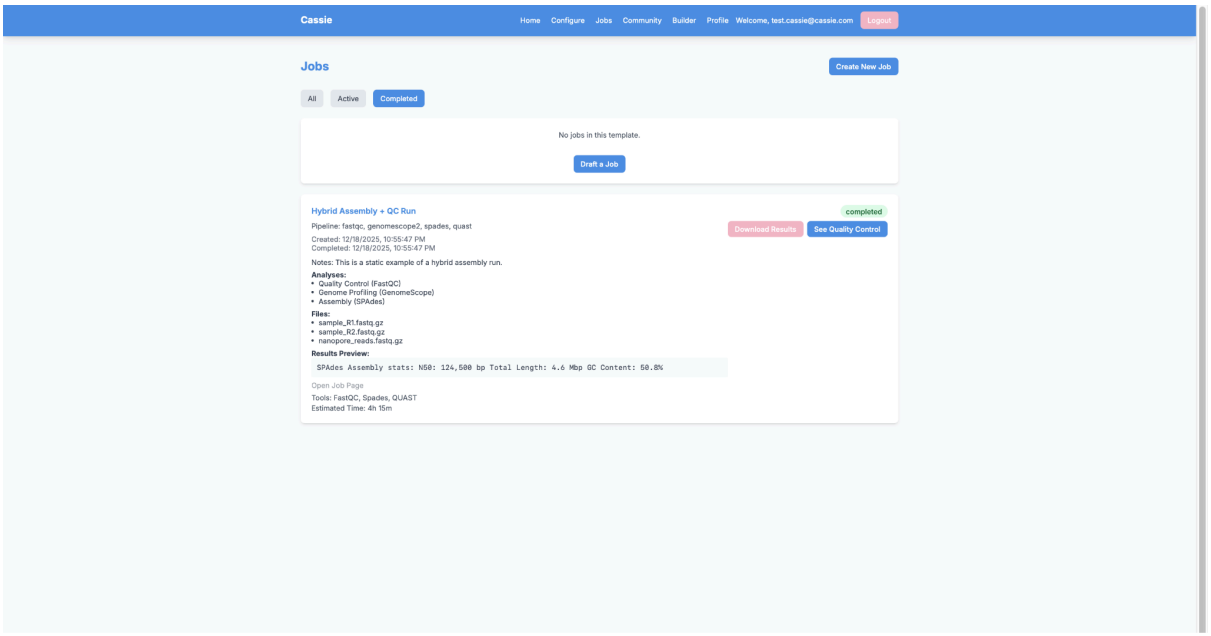
Workflow Configuration



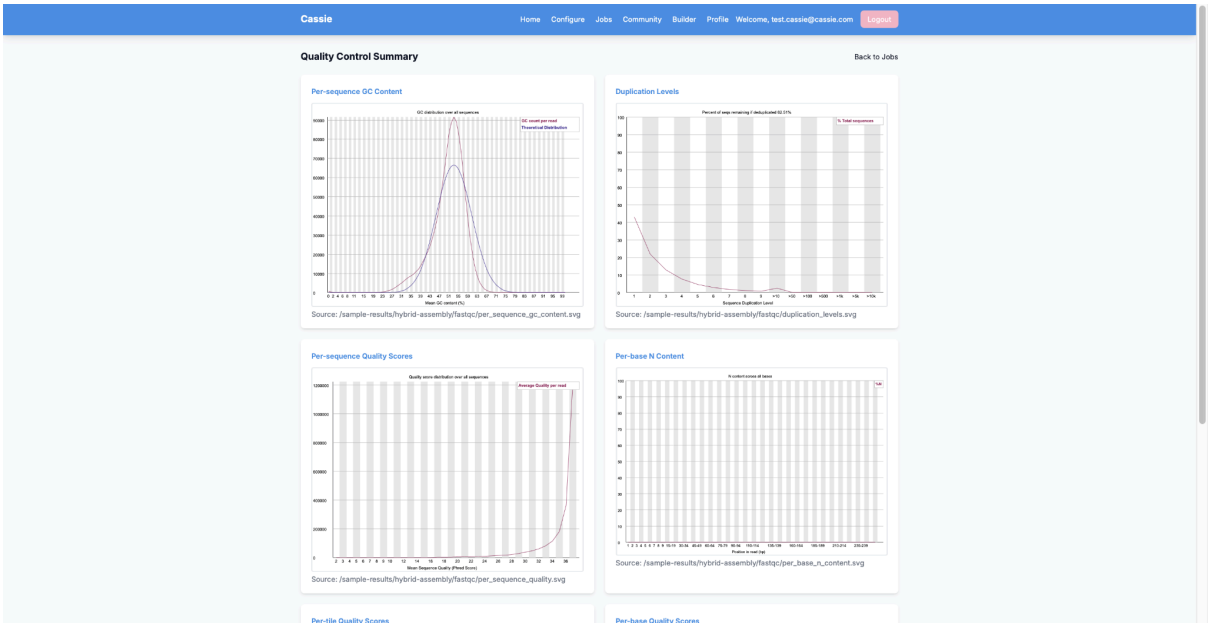
Jobs Page / Active Jobs



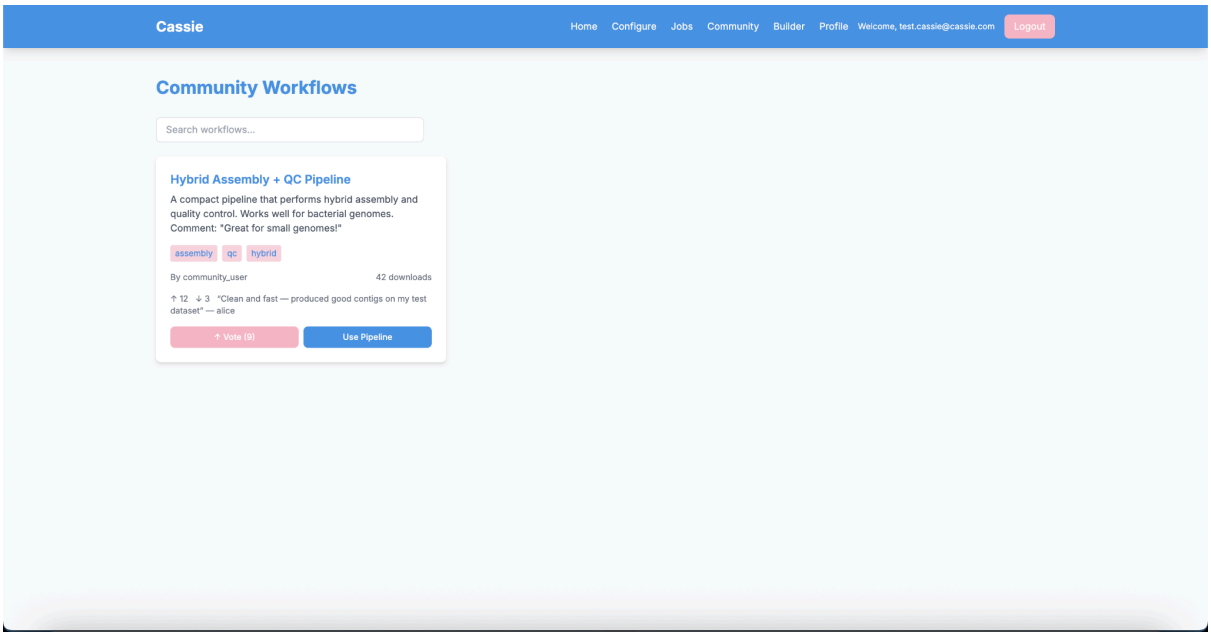
Jobs Page / Completed Jobs



Jobs Page / Quality Control Results



Community Page



4. Other Analysis Elements

4.1. Consideration of Various Factors in Engineering Design

4.1.1. Constraints

Public Health Considerations

CASSIE does not directly affect public health outcomes as it is proposed solely as a tool to support researchers in analyzing their data effectively and efficiently. Any scientific decision-making is left up to the researchers.

Public Safety Considerations

CASSIE's public safety considerations are addressed through design choices covering the secure execution of computational workflows and prevention of malicious or unintended system behaviour. Since CASSIE is a system that enables users to execute complex pipelines, safety measures regarding resources, unauthorized access, or execution of unsafe code must be implemented. The platform provides isolation between workflow executions, controlled access to computational resources, and strict permission boundaries to reduce the risk of system misuse.

Public Welfare Considerations

Public welfare is addressed through equitable and responsible access to computational resources. CASSIE is designed to support scientific research by simplifying individual software infrastructure work performed by researchers. The system imposes constraints on resource usage and execution scheduling to prevent a small number of users from establishing monopolies over the platform and disrupting the shared infrastructure. Thereby, CASSIE ensures fair access and maintains service availability for all users.

Global Considerations

Global considerations are limited, as CASSIE is designed to be used for educational or research purposes and does not impose the use of unfamiliar or poorly established tools in the analysis process.

Cultural Considerations

Cultural considerations are minimal for CASSIE, as the platform provides technical functionality. There exists no visible risk of any kind of cultural violations, as the system does not include any culturally sensitive content.

Social Considerations

Social interactions are only possible via the community page, and constraints and monitoring are required to prevent any plagiarism, misleading workflow information, manipulation in the popularity metrics, and the language used in the workflow details. CASSIE has a dedicated reporting system and communication channels to prevent any kind of misconduct.

Environmental Considerations

Environmental considerations are addressed through efficient use of computational resources. Introducing time and cost estimation helps prevent unnecessary use of computation units and helps reduce the energy consumption during executions.

Economic Considerations

Economic constraints are addressed by time and cost estimations. The proposed estimation functionalities help users to estimate the cost of execution beforehand. Also, different pricing based on different instance classes encourages users to select more cost-efficient options, which reduces redundant computation of the system. Therefore, lower operational costs for both the user and the system can be achieved.

4.1.2. Standards

During CASSIE's development, both software engineering standards and bioinformatics-specific data format standards are considered to achieve a robust and transparent system. These standards ensure sustainability, reproducibility, and compatibility with different platforms.

Software and Engineering Standards:

- **IEEE 830 – Software Requirements Specification Standard:**
  Principles of this standard are taken into account within the requirements section of the project documentation [2].
- **UML 2.5.1 – Modeling Standards:**
  UML guidelines are adopted for any diagrams utilized as needed (class diagrams, component diagrams, etc.) [3].
- **OCI (Open Container Initiative) Standards:**
  Creation, portability, and execution of Docker containers are performed in compliance with OCI standards [4], [5].
- **Kubernetes API ve Configuration Standards:**
  Component definitions such as Pod, Deployment, Namespace, and Resource quota are configured in compliance with official Kubernetes API conventions [6].
- **Nextflow DSL2 Standard:**
  Nextflow's DSL2 syntax is used to make sure the workflow is defined in a modular, reusable, and transparent structure [7], [8].

Bioinformatic Data Format Standards:

Widely accepted data formats are integrated into CASSIE. By providing compatibility with the following data formats, it is ensured that data flow within different tools without any problems.

- **FASTQ:** Stores raw character data produced by the sequencer along with confidence values for each character [9].
- **FASTA:** Stores raw character data produced by the sequencer [10].
- **GFA (Graphical Fragment Assembly):** A graph format where nodes represent sequence fragments and edges represent connections between them [11].
- **GFF3 / GTF:** Standard format used to store annotation data. Standard format used to store annotation data [12], [13].
- **BAM / CRAM / SAM:** Log-like formats showing where each read maps onto a reference sequence; BAM/CRAM are compressed versions of SAM [14], [15].
- **BED:** A simple tabular interval format specifying start–end positions of regions on a long sequence [16].

Compliance with these standards allows CASSIE to work in full compatibility with both modern cloud-based infrastructures and the existing tools and data structures within the bioinformatics ecosystem.

4.2. Risks and Alternatives

CASSIE is being implemented similarly to building blocks: an interactive Frontend, a backend capable of both communicating with the computation units and the frontend, a machine learning model for time and cost estimations, a workflow manager, and a cloud manager. Among these small containers, the primary risks are the integration of these building blocks and training the machine learning model without any available dataset.

Firstly, to solve the integration problem, we divided and assigned the integration process equally to the group members. This way, the burden of the integration won’t be on a single group member, and everyone will be able to do the integration of the block they implemented.

Secondly, to overcome the data scarcity problem with the training of the machine learning model, we decided to adopt a synthetic data generation approach based on operation counts. We will benchmark all the presented tools in CASSIE using biological datasets with various biological properties to collect the low-level operation counts. Then, we will estimate the number of low-level operations each instance class in our system performs, and assign time values to each benchmark based on the number of operations. Finally, we will train our estimation model based on the synthetic dataset we have generated.

4.3. Project Plan

WP 1: Market/Customer Analysis			
Start date: July 17, 2025		**End date: September 25, 2025**	
Leader:	Ege Şirvan	**Members involved:**	Ege Şirvan Eren Aslan Arda Kırcı
Objectives: Deciding the market and practical feasibility of the project.			
Tasks: **Task 1.1 Customer Analysis: Understanding customer needs and requests.** **Task 1.2 Market Analysis: Evaluating the market demand for the product and competitors.**			
Deliverables **D1.1: Summary of T2T consortium** **D1.2: Cloud Design Plan**			

WP 2: Custom-built Emulator			
Start date: September 26, 2025		**End date: December 11, 2025**	
Leader:	Eren Aslan	**Members involved:**	Ege Şirvan Eren Aslan
Objectives: Building an AWS emulation.			
Tasks: **Task 2.1 Server-Client System: Creating a Server-Client system where the Server is an AWS instance, and the Client can connect via sockets.** **Task 2.2 Adaptation for Tools: Making tools work in the emulator.**			
Deliverables **D2.1: AWS Emulator**			

WP 3: Genomic Tools Research and Selection			
Start date: October 7, 2025		**End date: October 31, 2025**	
Leader:	Ege Şirvan	**Members involved:**	Ege Şirvan Eren Aslan
Objectives: Selecting tools to be supported by the system.			
Tasks: **Task 3.1 Customer Analysis: Selecting which tools to be integrated into the system.**			
Deliverables **D3.1: Bioinformatic Tools List**			

WP 4: Dockerization of Demo Tools			
Start date: November 1, 2025		**End date: November 18, 2025**	
Leader:	Ege Şirvan	**Members involved:**	Ege Şirvan Eren Aslan
Objectives: Deciding the market and practical feasibility of the project.			
Tasks: **Task 4.1 Dockerization of the Tools: Dockerizing necessary tools for the demo.**			
Deliverables **D4.1: Dockerized tools.**			

WP 5: Backend-Emulator Integration			
Start date: November 1, 2025		**End date: December 25, 2025**	
Leader:	Eren Aslan	**Members involved:**	Ege Şirvan Eren Aslan Emre Can Yologlu
Objectives: Making the backend able to communicate with the AWS emulator.			
Tasks: **Task 5.1 Backend-Emulator Integration: Connecting Backend to AWS emulator.** **Task 5.2 Modification in Emulator: Modifying the emulator so that it can efficiently speak with the backend.**			
Deliverables **None**			

WP 6: Frontend-Backend Integration			
Start date: November 1, 2025		**End date: May 1, 2026**	
Leader:	Ege Şirvan	**Members involved:**	Ege Şirvan Eren Aslan
Objectives: Connecting the backend to the frontend.			
Tasks: **Task 6.1 Creating the Required API Endpoints: API endpoints should be created and selected properly.** **Task 6.2 API Placement into the Frontend: APIs should be placed into the Frontend so that they can efficiently communicate with the backend.**			
Deliverables **None**			

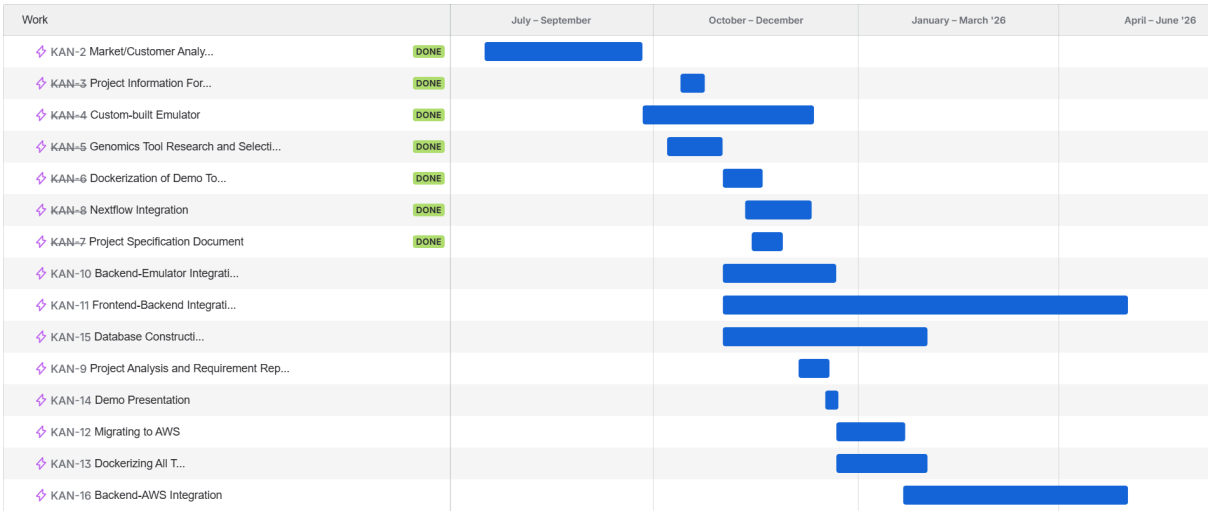
WP 7: Database Construction			
Start date: November 1, 2025		**End date: January 31, 2026**	
Leader:	Ege Şirvan	**Members involved:**	Ege Şirvan Emre Can Yologlu
Objectives: Designing and implementing a persistent relational database to store user information, workflow configurations, execution metadata, and community-related entities in a structured and scalable manner.			
Tasks: **Task 7.1 Database schema design based on the object and class model** **Task 7.2 Implementation of PostgreSQL schema and relational constraints** **Task 7.3 Integration of database access layer into backend services** **Task 7.4 Testing data integrity, isolation, and query performance**			
Deliverables **D7.1: Fully implemented PostgreSQL database schema** **D7.2: Database integration with backend API services**			

WP 8: Migrating to AWS			
Start date: December 22, 2025		**End date: January 21, 2026**	
Leader:	Eren Aslan	**Members involved:**	Ege Şirvan Eren Aslan Emre Can Yologlu Arda Kırcı
Objectives: Transitioning the system from the local emulator environment to a real AWS-based cloud infrastructure.			
Tasks: **Task 8.1 Deployment of Kubernetes cluster on AWS (EKS)** **Task 8.2 Configuration of AWS S3 for persistent object storage** **Task 8.3 IAM role and permission configuration** **Task 8.4 Validation of namespace isolation and storage security in production**			
Deliverables **D8.1: Deployed AWS-based execution environment** **D8.2: Verified production-ready cloud infrastructure**			

WP 9: Dockerizing All Tools			
Start date: December 22, 2025		**End date: January 31, 2026**	
Leader:	Ege Şirvan	**Members involved:**	Ege Şirvan Eren Aslan
Objectives: Ensuring that all bioinformatics tools supported by CASSIE are containerized and reproducible across environments.			
Tasks: **Task 9.1 Dockerization of all selected genome assembly, QC, and annotation tools** **Task 9.2 Version pinning and dependency isolation** **Task 9.3 Validation of container compatibility with Nextflow and Kubernetes**			
Deliverables **D9.1: Fully Dockerized and validated tool set**			

WP 10: Backend-AWS Integration			
Start date: January 21, 2026		**End date: May 1, 2026**	
Leader:	Eren Aslan	**Members involved:**	Ege Şirvan Eren Aslan Arda Kırcı Emre Can Yologlu
Objectives: Connecting backend services to AWS infrastructure and enabling full end-to-end execution in production.			
Tasks: **Task 10.1 Backend adaptation for AWS S3 and EKS** **Task 10.2 Workflow submission and monitoring via AWS infrastructure** **Task 10.3 Performance testing under concurrent user workloads** **Task 10.4 End-to-end system validation**			
Deliverables **D10.1: Fully functional cloud-native CASSIE deployment** **D10.2: Final integrated system ready for demonstration**			

Timeline



4.4. Ensuring Proper Teamwork

To ensure an inclusive, creative, and collaborative development environment, our team has established rules for task management, healthy communication, and shared leadership.

Collaboration and Management Tools

Use of top-notch tools to track individual contributions and ensure easy collaboration:

- **GitHub**: Used for source code management. Every feature's test is performed on separate branches and merged with pull requests. This ensures code changes, additions, and deletions are visible, traceable, and can be reviewed by other team members before merging.
- **Jira**: We use a board to break down work into tasks that are more manageable. This helps us to visualize the project status and helps to see if a team member is overloaded or left without a task.

4.5. Ethics and Professional Responsibilities

Developing a platform for genomic analysis brings specific ethical and professional responsibilities, particularly regarding data privacy, scientific integrity, and system reliability. Since what is being developed is a platform for genomic analysis, this brings specific ethical and professional responsibilities, especially regarding data privacy and scientific integrity.

Data Privacy and Security

Genomic data is one of the most highly sensitive pieces of information. If it belongs to a person, it is the most personal information about their health. Even if it does not belong to a

human being, it is still the property of the researcher. As the architects and engineers of this platform, we have obligations to protect this data. CASSIE handles this by:

- **Isolation:** CASSIE enforces namespace isolation with Kubernetes and bucket-level isolation in S3. These precautions are taken to ensure that no user can access another researcher's data.
- **Access Control:** Industry-standard measurements are taken for user authentication, user data integrity, and general security of the CASSIE platform. We implement multi-layer access control covering both the web application and the Kubernetes environment underlying it. JSON Web Tokens (JWT) are utilized in the application layer for stateless user authentication. Authorization in the execution environment is done using Kubernetes Role-Based Access Control (RBAC). RBAC defined using Kubernetes declarative configuration files ensures access to computational resources is done with the Principle of Least Privilege in mind.

Scientific Integrity and Reproducibility

A platform that makes its debut with a claim that it will solve a certain problem regarding scientific tools, of course, must ensure the scientific integrity, meaning results produced are valid, trustworthy, and produced with rigorous standards. Its results should also be reproducible. CASSIE uses containerized tools that are locked to specific versions. This guarantees that results are reproducible. These containerized tools are chosen from peer-reviewed bioinformatics tools. This makes sure all output is derived from proven and established methods.

4.6. Planning for New Knowledge and Learning Strategies

The CASSIE project requires knowledge of advanced technologies that we haven't encountered in any of our past courses. These include Container and Cloud Orchestration, Workflow Management, and Bioinformatics knowledge. We followed some strategies to fill in knowledge gaps:

Identify the gap:

- **Bioinformatic Pipelines**: Understanding formats of input and output. Then, understanding the biological significance and logic of each step.
- **Workflow Management:** Learning the configuration of Nextflow. Learning how it handles data flow and resource usage.

- **Container and Cloud Orchestration (Kubernetes & AWS):** Learning to configure Kubernetes for container orchestration and configuring cloud instances (EC2, S3) securely.

Execute Learning Strategies:

- **Literature & Documentation:** Consulting official documentation of Kubernetes and Nextflow, as well as user-created guides.
- **Specialization of team members:** Identified the main topics as backend, frontend, database, containerized tools, workflow management, and container orchestration. Those main topics are then divided among members.
- **Prototypes:** We test small-scale pipelines with few tools before merging them into the full system.
- **Use of Generative AI:** We use generative AI for both closing the knowledge gap we have and debugging our implementation.

5. References

[1] The Galaxy Community, The Galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update, *Nucleic Acids Research*, vol. 52, pp. 83-84, 2024, https://doi.org/10.1093/nar/gkae410 [Accessed Nov. 17, 2025].

[2] IEEE Computer Society, *IEEE Recommended Practice for Software Requirements Specifications (IEEE 830-1998)*, IEEE Standards Association, 1998. Available: https://doi.org/10.1109/IEEESTD.1998.88286 [Accessed Dec. 15, 2025].

[3] Object Management Group, *Unified Modeling Language (UML) Version 2.5.1 Specification*, OMG, 2017. Available: https://www.omg.org/spec/UML/2.5.1 [Accessed Dec. 15, 2025].

[4] Open Container Initiative, *OCI Image Format Specification v1.0.0*, The Linux Foundation, 2017. Available: https://github.com/opencontainers/image-spec [Accessed Dec. 15, 2025].

[5] Open Container Initiative, *OCI Runtime Specification v1.0.0*, The Linux Foundation, 2017. Available: https://github.com/opencontainers/runtime-spec [Accessed Dec. 15, 2025].

[6] The Linux Foundation and Cloud Native Computing Foundation, *Kubernetes API Reference Documentation*, Kubernetes Project, 2024. Available: https://kubernetes.io/docs/reference/kubernetes-api [Accessed Dec. 15, 2025].

[7] P. Di Tommaso et al., "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, vol. 35, pp. 316–319, 2017, https://doi.org/10.1038/nbt.3820 [Accessed Dec. 15, 2025].

[8] Seqera Labs, *Nextflow DSL2 Documentation*, Seqera Labs, 2024. Available: https://www.nextflow.io/docs/latest/dsl2.html [Accessed Dec. 15, 2025].

[9] P. J. Cock et al., "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants," *Nucleic Acids Research*, vol. 38, no. 6, pp. 1767–1771, 2010, https://doi.org/10.1093/nar/gkp1137 [Accessed Dec. 16, 2025].

[10] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proceedings of the National Academy of Sciences*, vol. 85, no. 8, pp. 2444–2448, 1988, https://doi.org/10.1073/pnas.85.8.2444 [Accessed Dec. 16, 2025].

[11] The GFA Specification Team, “The Graphical Fragment Assembly (GFA) Format Specification, Version 1.0,” GitHub Documentation, 2016. Available: https://github.com/GFA-spec/GFA-spec [Accessed Dec. 16, 2025].

[12] L. Stein, “GFF3 Specification: Generic Feature Format Version 3,” Sequence Ontology Project, 2013. Available: https://github.com/The-Sequence-Ontology/Specifications [Accessed Dec. 16, 2025].

[13] The GENCODE Consortium, “GTF Format Specification,” EMBL-EBI/GENCODE Documentation, 2024. Available: https://www.gencodegenes.org/pages/data_format.html [Accessed Dec. 16, 2025].

[14] H. Li et al., “The Sequence Alignment/Map format and SAMtools,” *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009, https://doi.org/10.1093/bioinformatics/btp352 [Accessed Dec. 16, 2025].

[15] J. Bonfield et al., “CRAM 3.1: advances in reference-based compression of high throughput sequencing data,” *Bioinformatics*, vol. 37, pp. 456–458, 2021, https://doi.org/10.1093/bioinformatics/btaa1085 [Accessed Dec. 16, 2025].

[16] J. Kent et al., “The Human Genome Browser at UCSC,” *Genome Research*, vol. 12, no. 6, pp. 996–1006, 2002, https://doi.org/10.1101/gr.229102 [Accessed Dec. 16, 2025].