CS491 SENIOR DESIGN PROJECT

T2522

CASSIE

Detailed Design 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

13.03.2026

1. Introduction

1.1 Purpose of the System

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.

1.3 Overview

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 Software Architecture

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. 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 Software Architecture

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.

3.2 Subsystem decomposition

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 execute workflows at scale in a cloud environment using Kubernetes. 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, coordinate communication between the user interface, workflow manager, container orchestrator, authentication manager, and cloud manager. This architecture is suitable for concurrent use of multiple users while maintaining isolation between individual workflow executions.

3.2.1 Dynamic User Interface

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. It provides a workflow-building interface that allows users to create complex workflows with ease. It also allows users to access readable workflow results, workflow logs, and error messages through the interface without downloading them.

3.2.2 Container Orchestrator

CASSIE uses Kubernetes as its container orchestrator. It orchestrates the workflow execution, manages computational resources, and provides isolation through namespaces and pods. It automatically creates and destroys Docker containers of genomic tools, which are created from their Docker images, as needed. The Kubernetes cluster provides horizontal scalability by adding new nodes when system load increases. One of the most important functionalities Kubernetes provides is namespace isolation, which prevents both over-usage of resources by a user and possible data leaks.

3.2.3 Workflow Manager

CASSIE handles workflows using a workflow engine, Nextflow, enabling consistent execution across different environments. It automates pipelines, including quality control, assembly, assembly analysis, and annotation, while ensuring full reproducibility. It decides the execution order and whether to execute multiple tools at once, and if it will, which ones to execute automatically by checking the data and resource requirements of each tool in the pipeline.

3.2.4 Cloud Manager

CASSIE is a cloud-based website. AWS will be used as the cloud architecture service of the project, which will provide scalable compute (EC2) and storage (S3) resources. All other systems will work in the cloud subsystem. It manages the interaction between the other subsystems of CASSIE and the cloud infrastructure. It distributes compute resources for workflow executions, manages data storage operations, and ensures that workflows are executed appropriately. It also coordinates resource allocation, instance selection, and communication with the Kubernetes cluster running on the cloud infrastructure.

3.2.5 Authentication Manager

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. The Authentication Manager handles user authentication, session validation, and authorization to ensure secure access to the CASSIE components.

3.3 Persistent Data Management

CASSIE is designed to process large genomic datasets that must be stored reliably and accessed by different subsystems during workflow execution. Persistent data management is therefore a critical component of the system architecture. CASSIE uses a database (PostgreSQL for permanent/long-term data, while it utilizes object storage (S3 buckets) during workflow executions.

3.3.1 Database (PostgreSQL)

CASSIE uses a PostgreSQL database for user accounts and authentication information, workflow configurations, workflow execution metadata, references to stored datasets, job status, logs, and community workflow metadata. It allows efficient querying and management of system information.

3.3.2 Object Storage (S3 Bucket)

CASSIE uses S3 buckets as object storage. Genomic data, workflow outputs, and intermediate results are stored using S3 object storage. S3 provides scalable and durable storage that can

handle the large file sizes typical for genomic data. Each user is assigned an isolated storage space to ensure data privacy and prevent access to other users' private datasets. Uploaded datasets are accessed during workflow executions, and the results of these workflows are stored there as well.

3.4 Access Control and Security

Considering the privacy-sensitive nature of genomic data, CASSIE incorporates access control systems and security mechanisms to protect user data, computational resources, and workflow executions.

3.4.1 User Authentication

CASSIE requires all users to authenticate before accessing the system. User authentication is handled in the backend services using JSON Web Tokens (JWT). When a user successfully logs into the platform, the authentication service generates a JWT with the user's identity data and authorization information. This token then gets used when the user makes an API call. The backend validates the token for each request, ensuring that only authenticated users can access system services. CASSIE also encodes the user passwords to prevent liabilities in the case of a database breach and provides two-factor authentication to achieve higher security.

3.4.2 Data Isolation

Workflow executions are isolated using Kubernetes namespaces. Each workflow execution is assigned to its own namespace, ensuring that compute resources, intermediate files, and workflow results remain private.

3.4.3 File Control

Uploaded data is checked before workflow executions according to the pipeline's file requirements, and only if all required files are satisfied, the workflow is executed. Temporary execution data and intermediate files are removed after workflow completion to minimize unnecessary data exposure.

4. Subsystem Services

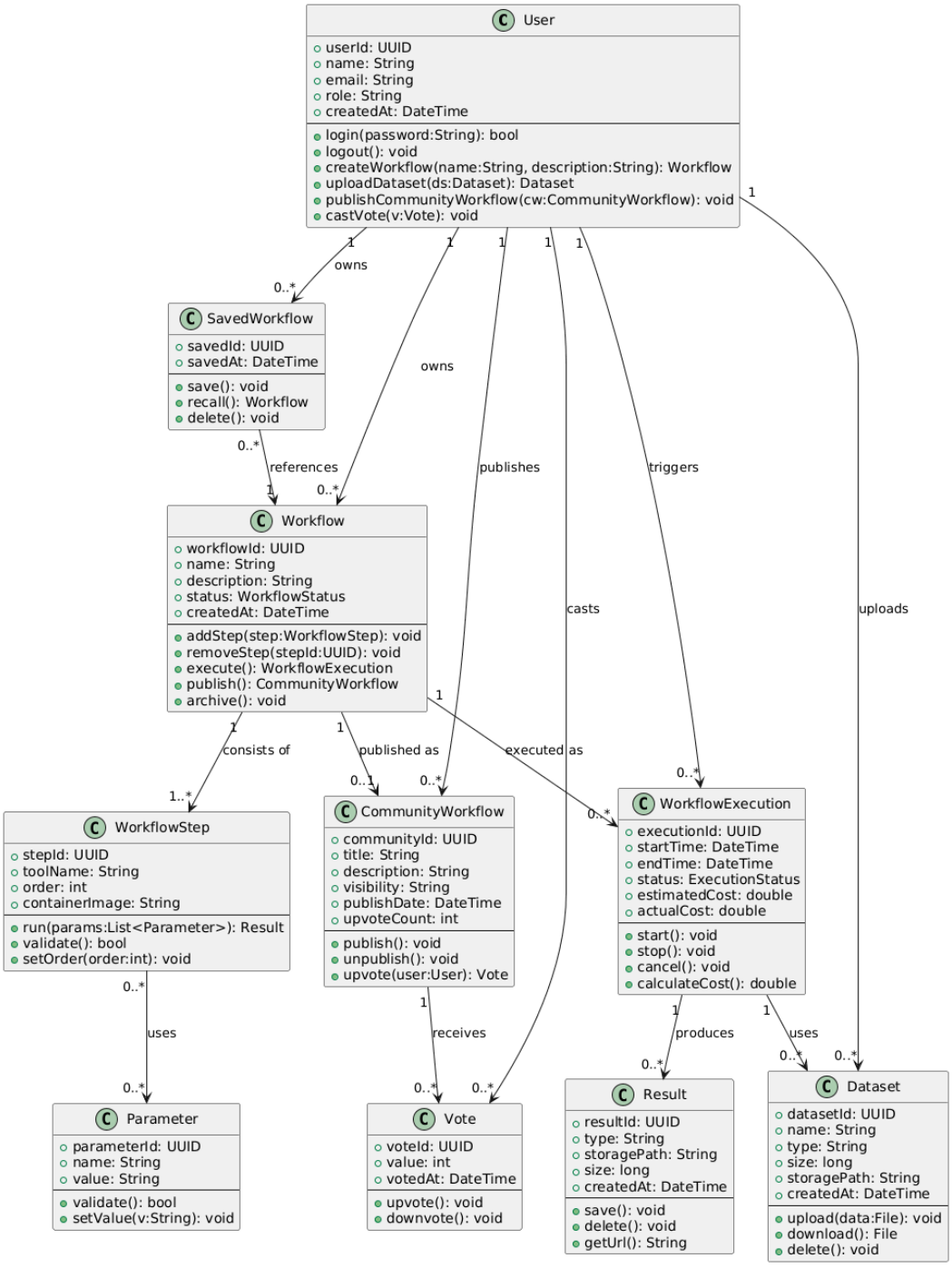

Fig. 1: Class Diagram of the CASSIE System

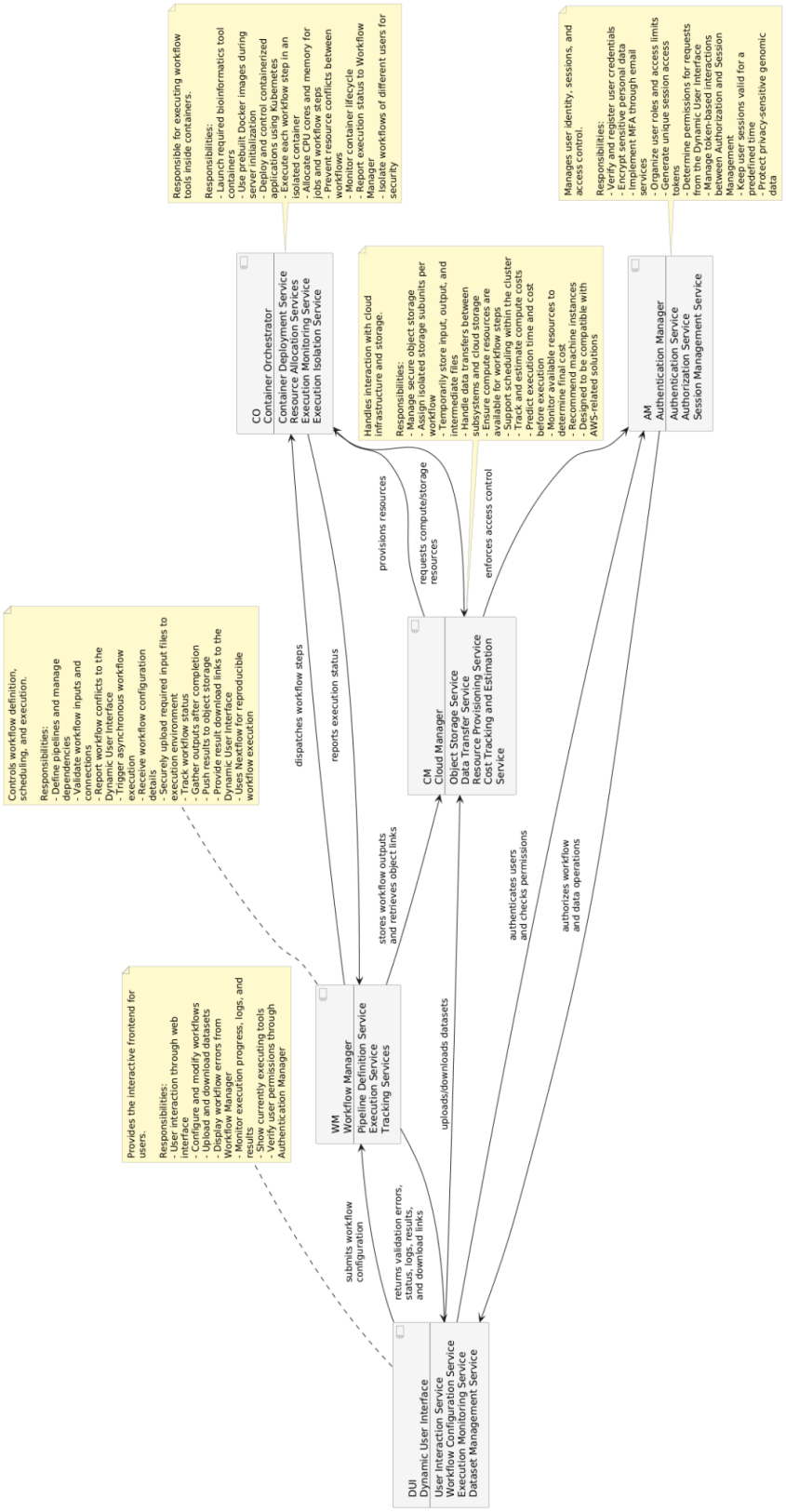

Fig 2: Subsystem Diagram of the CASSIE System

4.1 Dynamic User Interface

The Dynamic User Interface subsystem provides the interaction layer between users and the CASSIE platform. It enables users to configure workflows, upload datasets, monitor execution progress, and access analysis results. The subsystem is implemented through interface components and controller classes responsible for handling user requests (communicating with the backend).

4.1.1 User Interaction Service

This subsystem allows users to interact with the platform through a web interface. These services allow users to configure workflows, manage datasets, start various tasks, and interact with other users.

4.1.2 Workflow Configuration Service

Users can create and modify workflows through the interface. The subsystem collects workflow parameters and configuration details and forwards them to the Workflow Manager subsystem. It also shows the error messages (if any exist) that were received from the Workflow Manager in case of a parameter mismatch.

4.1.3 Execution Monitoring Service

This subsystem retrieves workflow execution status, logs, and results from the Workflow Manager and presents them to users in a readable format. It also shows which tools are currently executing in their workflow session.

4.1.4 Dataset Management Services

The interface supports dataset upload and download operations. Data transfers are coordinated with the Cloud Manager subsystem. The Dynamic User Interface subsystem utilizes the Authentication Manager to verify user access permissions before allowing any workflow or data operations.

4.2 Container Orchestrator

The Container Orchestrator subsystem manages the execution of genomic tools inside containerized environments. In CASSIE, this functionality is implemented using Kubernetes to deploy and control containerized applications.

4.2.1 Container Deployment Service

This subsystem launches containers of the required bioinformatics tools. Containers get their image instances from the already existing Docker images of genomic tools in the system, which are built during server initialization. Genomic tool images are built using publicly available links. Each workflow step is executed inside an isolated container environment.

4.2.2 Resource Allocation Services

Maximum computational resources, such as the number of available CPU cores and the amount of available memory, are allocated for each job. This service also allocates an appropriate number of CPU cores and amount of memory for each workflow step according to the maximum computational resources. This ensures efficient use of system resources and prevents resource use conflicts between workflows.

4.2.3 Execution Monitoring Service

This subsystem monitors the lifecycle of running containers and reports execution status to the Workflow Manager subsystem. So that the Workflow Manager Subsystem can redirect the current workflow status to the Execution Monitoring Services in the Dynamic User Interface Subsystem.

4.2.4 Execution Isolation Service

Different users' workflows are executed in separate, isolated containers, ensuring security and preventing interference between workflows.

4.3 Workflow Manager

This subsystem is responsible for handling workflows using Nextflow, which provides consistent executions across different environments. While ensuring full reproducibility, it automates complex pipelines, from quality control to assembly, analysis, and annotation.

4.3.1 Pipeline Definition Service

This service defines workflows and manages the dependencies required for biological analyses. To establish a compatible pipeline between Workflow Configuration Services of the Dynamic User Interface Subsystem and the Workflow Manager Subsystem, it supports a workflow creation system that utilizes text-based inputs. It also evaluates the inputs and connections that are defined in workflows and reports any conflicts to the Dynamic User Interface Subsystem.

4.3.2 Execution Service

This subsystem triggers the workflow execution. To ensure a responsive user interface, tasks are executed asynchronously in the background. It receives workflow configuration details from the Pipeline Definition Services. It securely uploads required input files from the user into the execution environment.

4.3.3 Tracking Services

This subsystem tracks the status of workflows. After a successful completion of a workflow, it locates the result directory, gathers the generated outputs, and pushes them to the object storage. It provides access links to download workflow results to the Execution Monitoring Service of the Dynamic User Interface Subsystem.

4.4 Cloud Manager

This subsystem manages the interactions between the other CASSIE subsystems and the cloud infrastructure on which it runs. The Cloud Manager Subsystem is designed to be compatible with AWS and related cloud solutions.

4.4.1 Object Storage Service

This subsystem is responsible for managing secure object storage. Each workflow will be assigned to an isolated storage subunit, which will be used to temporarily store input/output/intermediate files.

4.4.2 Data Transfer Service

This subsystem is responsible for handling the data transfers between the other CASSIE subsystems and the cloud storage services.

4.4.3 Resource Provisioning Service

This subsystem manages the cloud resources. When a workflow step is ready to execute, the service ensures that appropriate compute resources are available and that the task can be scheduled within the cluster.

4.4.4 Cost Tracking and Estimation Service

This subsystem will track and estimate compute costs. Workflows will be evaluated before execution to give predictions on time and execution costs. It will monitor available resources to determine the final cost. It will also recommend a machine instance according to its estimations.

4.5 Authentication Manager

This subsystem is responsible for handling authentication-related issues in the system. It manages user-access protocols, password/MFA actions, and user roles. It grants secure access to and control of privacy-sensitive genomic data.

4.5.1 Authentication Service

This subsystem verifies and registers user credentials in compliance with the existing user data. It encrypts sensitive personal data using novel methods and implements Multifactor Authentication to ensure personal access security via email services.

4.5.2 Authorization Service

This subsystem organizes in-system user-roles and their access limits. Each user is given a unique access token in each session by the Session Management Service, which is then used to determine user permissions whenever a request is sent from the Dynamic User Interface

Subsystem. The Authorization Service manages the access token-related interactions between the Dynamic User Interface and the Session Management Service.

4.5.3 Session Management Service

This subsystem generates unique access tokens whenever a user logs in, which are valid for a predetermined amount of time. These tokens are used to provide a continuous experience between different pages of the website until they expire. They are also used by the Authorization Service to decide user roles.

5. Test Cases

Test ID	T-01	Category	Functional	Severity	Major
Objective	Successful User Registration				
Steps	1. Navigate to Sign Up page. 2. Enter valid email address. 3. Enter and confirm a strong password. 4. Click 'Register' button.				
Expected	User account is created; user is redirected to the Home Page.				
Date-Result					

Fig. 3: Test Case 1

Test ID	T-02	Category	Functional	Severity	Critical
Objective	User Login with Valid Credentials				
Steps	1. Navigate to Login page. 2. Enter registered email. 3. Enter correct password. 4. Click 'Login' button.				
Expected	Successful authentication; user is redirected to the Home Page with active session.				
Date-Result					

Fig. 4: Test Case 2

Test ID	T-03	Category	Functional	Severity	Major
Objective	Login Validation - Incorrect Password				
Steps	1. Navigate to Login page. 2. Enter valid email. 3. Enter an incorrect password. 4. Click 'Login'.				
Expected	Authentication fails; clear error message 'Invalid email or password' is displayed.				
Date-Result					

Fig. 5: Test Case 3

Test ID	T-04	Category	Security	Severity	Major
Objective	JWT Session Persistence				
Steps	1. Log into CASSIE. 2. Close the browser tab. 3. Re-open the browser and navigate back to CASSIE. 4. Check if session is active.				
Expected	User remains logged in.				
Date-Result					

Fig. 6: Test Case 4

Test ID	T-05	Category	Functional	Severity	Minor
Objective	Password Confirmation Mismatch				
Steps	1. Navigate to Sign Up page. 2. Fill email and password. 3. Enter a different password in 'Confirm Password'. 4. Click 'Register'.				
Expected	Error message displayed stating passwords do not match; account not created.				
Date-Result					

Fig. 7: Test Case 5

Test ID	T-06	Category	Functional	Severity	Critical
Objective	Upload Valid Genomic Files (FASTQ)				
Steps	1. Navigate to Workflow Configuration. 2. Click 'Upload Data'. 3. Select a valid .fastq file. 4. Wait for upload to finish.				
Expected	File is uploaded successfully and appears in the user's data list.				
Date-Result					

Fig. 8: Test Case 6

Test ID	T-07	Category	Functional	Severity	Major
Objective	Upload Invalid File Format Validation				
Steps	1. Navigate to 'Upload Data'. 2. Select a txt or jpg file. 3. Attempt to upload. 4. Observe system feedback.				
Expected	System rejects the file; error message 'Unsupported or invalid format' appears.				
Date-Result					

Fig. 9: Test Case 7

Test ID	T-08	Category	Integration	Severity	Major
Objective	Direct Cloud-to-S3 Data Transfer				
Steps	1. Select 'Import from Cloud Storage'. 2. Provide External Cloud credentials/link. 3. Select target genomic files. 4. Click 'Transfer'.				
Expected	Files are transferred directly to CASSIE's S3 isolated directory without local download.				
Date-Result					

Fig. 10: Test Case 8

Test ID	T-09	Category	Security	Severity	Critical
Objective	Data Isolation - Bucket Access Control				
Steps	1. Log in as User A. 2. Upload a file. 3. Log out and log in as User B. 4. Navigate to Data list.				
Expected	User B cannot see or access User A's uploaded files.				
Date-Result					

Fig. 11: Test Case 9

Test ID	T-10	Category	Functional	Severity	Major
Objective	Delete Uploaded Dataset				
Steps	1. Navigate to Profile/Manage Datasets. 2. Click 'Delete' next to a file. 3. Confirm deletion in the pop-up. 4. Refresh page.				
Expected	File is removed from the UI and from the underlying S3 storage.				
Date-Result					

Fig. 12: Test Case 10

Test ID	T-11	Category	Functional	Severity	Major
Objective	Manual Workflow Construction				
Steps	1. Click 'Build Pipeline'. 2. Drag 'QC' component to canvas. 3. Drag 'Assembly' component and connect. 4. Click 'Save Pipeline'.				
Expected	Workflow structure is saved and visible in 'Saved Pipelines' on Profile Page.				
Date-Result					

Fig. 13: Test Case 11

Test ID	T-12	Category	Functional	Severity	Major
Objective	Automated Tool Selection via Goals				
Steps	1. Select 'Configure' with automation. 2. Provide analysis goals. 3. Answer high-level questions. 4. Review proposed tools.				
Expected	System automatically selects appropriate tools based on user answers.				
Date-Result					

Fig. 14: Test Case 12

Test ID	T-13	Category	Functional	Severity	Major
Objective	Dynamic Parameter Display				
Steps	1. Go to Workflow Configuration. 2. Select 'Tool A' from the list. 3. Select 'Tool B'. 4. Check parameter fields.				
Expected	UI only shows parameters specific to Tool A and Tool B; unrelated parameters are hidden.				
Date-Result					

Fig. 15: Test Case 13

Test ID	T-14	Category	Functional	Severity	Major
Objective	Validation of Missing Parameters				
Steps	1. Configure a workflow. 2. Leave a required parameter empty. 3. Click 'Validate/Execute'. 4. Check error markers.				
Expected	Validation fails; UI highlights missing field with a 'required' warning.				
Date-Result					

Fig. 16: Test Case 14

Test ID	T-15	Category	Functional	Severity	Major
Objective	Resource Estimation Accuracy				
Steps	1. Finalize workflow config. 2. View Resource Estimation section. 3. Change the dataset size. 4. Observe the estimated time/price.				
Expected	Estimated cost and time update dynamically based on the ML model's prediction.				
Date-Result					

Fig. 17: Test Case 15

Test ID	T-16	Category	Functional	Severity	Major
Objective	Manual Machine Selection				
Steps	1. View recommended Machine instance. 2. Open dropdown to select a different class. 3. Click 'Confirm Selection'.				
Expected	Workflow is updated to run on the user-selected instance class.				
Date-Result					

Fig. 18: Test Case 16

Test ID	T-17	Category	Integration	Severity	Critical
Objective	Workflow Submission to Nextflow				
Steps	1. Complete configuration. 2. Click 'Execute Pipeline'. 3. Navigate to 'Active Jobs'. 4. Verify job status is 'Queued/Running'.				
Expected	Backend sends parameters to Nextflow; Nextflow initiates the pipeline on Kubernetes.				
Date-Result					

Fig. 19: Test Case 17

Test ID	T-18	Category	Security	Severity	Critical
Objective	Kubernetes Namespace Isolation				
Steps	1. Start Job One. 2. Start Job Two as a different user. 3. Check Kubernetes API for namespaces. 4. Verify separate namespaces.				
Expected	Each job runs in a dedicated, isolated Kubernetes namespace.				
Date-Result					

Fig. 20: Test Case 18

Test ID	T-19	Category	Reliability	Severity	Major
Objective	Automatic Task Retry on Failure				
Steps	1. Simulate a transient pod failure. 2. Observe Nextflow logs. 3. Check retry count for the task.				
Expected	Nextflow automatically retries the failed task according to the predefined retry policy.				
Date-Result					

Fig. 21: Test Case 19

Test ID	T-20	Category	Functional	Severity	Major
Objective	Job Termination/Cancellation				
Steps	1. Start a running job. 2. Navigate to Jobs Page. 3. Click 'Cancel Workflow'. 4. Confirm action.				
Expected	Workflow stops; Kubernetes pods for that job are terminated; status changes to 'Cancelled'.				
Date-Result					

Fig. 22: Test Case 20

Test ID	T-21	Category	Reliability	Severity	Minor
Objective	Intermediate File Cleanup				
Steps	1. Run a workflow to completion. 2. Access the Kubernetes namespace/storage. 3. Verify presence of final outputs vs intermediate files.				
Expected	Temporary/intermediate files are removed; only final tool outputs remain.				
Date-Result					

Fig. 23: Test Case 21

Test ID	T-22	Category	Functional	Severity	Major
Objective	Real-time Execution Status Updates				
Steps	1. Start a job. 2. Stay on the Jobs Page. 3. Watch the progress bar/status. 4. Wait for step change.				
Expected	Status updates from 'Working' to 'Completed' for each step in near-real-time via UI.				
Date-Result					

Fig. 24: Test Case 22

Test ID	T-23	Category	Functional	Severity	Major
Objective	Download Results as Compressed Archive				
Steps	1. Navigate to Completed Jobs. 2. Click 'Download All Results'. 3. Open the downloaded .zip file. 4. Check contents.				
Expected	Download is successful; archive contains all relevant output files.				
Date-Result					

Fig. 25: Test Case 23

Test ID	T-24	Category	Functional	Severity	Major
Objective	View Workflow Execution Logs				
Steps	1. Go to a completed job. 2. Click 'View Logs'. 3. Scroll through tool output. 4. Verify readability.				
Expected	Full Nextflow and tool logs are accessible for debugging and transparency.				
Date-Result					

Fig. 26: Test Case 24

Test ID	T-25	Category	Functional	Severity	Major
Objective	Quality Control Visualization				
Steps	1. Click 'Quality Control Results' for a job. 2. Observe displayed diagrams. 3. Verify data axes/labels.				
Expected	UI displays visual diagrams as shown in mock-ups.				
Date-Result					

Fig. 27: Test Case 25

Test ID	T-26	Category	Functional	Severity	Major
Objective	Resubmit Failed Job with Adjusted Params				
Steps	1. Identify a 'Failed' job. 2. Click 'Adjust & Resubmit'. 3. Change a parameter. 4. Execute again.				
Expected	System re-runs workflow without requiring data re-upload; new job ID created.				
Date-Result					

Fig. 28: Test Case 26

Test ID	T-27	Category	Functional	Severity	Major
Objective	Publish Workflow to Community				
Steps	1. Open a saved workflow. 2. Click 'Publish to Community'. 3. Enter metadata (tags, description). 4. Submit.				
Expected	Workflow appears in the public Community catalog; no private datasets are linked.				
Date-Result					

Fig. 29: Test Case 27

Test ID	T-28	Category	Functional	Severity	Minor
Objective	Search/Filter Community Workflows				
Steps	1. Navigate to Community Page. 2. Type 'Assembly' in search. 3. Apply a 'Highly Voted' filter. 4. Review results.				
Expected	List updates to show only workflows matching the keyword and filter criteria.				
Date-Result					

Fig. 30: Test Case 28

Test ID	T-29	Category	Functional	Severity	Major
Objective	Import Community Workflow				
Steps	1. Select a workflow from Community. 2. Click 'Import Pipeline'. 3. Navigate to 'My Workflows'. 4. Check for the copy.				
Expected	A private, editable copy of the workflow is added to the user's workspace.				
Date-Result					

Fig. 31: Test Case 29

Test ID	T-30	Category	Functional	Severity	Minor
Objective	Voting on Community Workflows				
Steps	1. Open a community workflow detail. 2. Click 'Upvote'. 3. Refresh page. 4. Check vote count.				
Expected	Vote count increases by 1; system maintains attribution to original author.				
Date-Result					

Fig. 32: Test Case 30

Test ID	T-31	Category	Usability	Severity	Minor
Objective	Contextual Help Tooltips				
Steps	1. Navigate to Workflow Config. 2. Hover over a '?' icon next to a tool parameter. 3. Read the popup.				
Expected	Help text explains the parameter's purpose and expected values.				
Date-Result					

Fig. 33: Test Case 31

Test ID	T-32	Category	Performance	Severity	Major
Objective	Concurrent Job Handling				
Steps	1. Open multiple tabs. 2. Initiate 5 different workflows simultaneously. 3. Monitor 'Active Jobs'.				
Expected	System handles all requests without crashing; Kubernetes schedules pods per resource availability.				
Date-Result					

Fig. 34: Test Case 32

Test ID	T-33	Category	Reliability	Severity	Major
Objective	Data Durability - Storage Outage				
Steps	1. During an upload, simulate a brief S3 connection loss. 2. Check if system retries. 3. Verify file integrity after reconnect.				
Expected	Retry mechanism prevents data loss; partial uploads are handled gracefully.				
Date-Result					

Fig. 35: Test Case 33

Test ID	T-34	Category	Security	Severity	Critical
Objective	Unauthorized Endpoint Access				
Steps	1. Copy a Job ID from User A. 2. Log in as User B. 3. Manually enter the URL for Job A's result. 4. Observe response.				
Expected	System redirects to home; unauthorized access is blocked.				
Date-Result					

Fig. 36: Test Case 34

Test ID	T-35	Category	Functional	Severity	Major
Objective	Reproducibility with Identical Parameters				
Steps	1. Run a workflow. 2. Record results. 3. Re-run identical workflow with same dataset and parameters. 4. Compare outputs.				
Expected	Outputs are scientifically identical, ensuring reproducibility.				
Date-Result					

Fig. 37: Test Case 35

Test ID	T-37	Category	Functional	Severity	Minor
Objective	Empty Dataset Selection Validation				
Steps	1. Create a workflow. 2. Click 'Execute' without selecting any data. 3. Observe UI behavior.				
Expected	Execution is blocked; user is prompted to 'Select at least one dataset'.				
Date-Result					

Fig. 39: Test Case 37

Test ID	T-38	Category	Functional	Severity	Major
Objective	Large Data Volume Support				
Steps	1. Upload a 50GB genomic dataset. 2. Configure a workflow. 3. Initiate execution. 4. Monitor storage and pod status.				
Expected	System handles high-volume data and allocates appropriate disk space in Kubernetes.				
Date-Result					

Fig. 40: Test Case 38

Test ID	T-39	Category	Usability	Severity	Major
Objective	Readable Error Messages for CLI Failures				
Steps	1. Provide invalid input that triggers a tool-level error (not UI). 2. View the job error report. 3. Read the message.				
Expected	Error is translated into a user-friendly explanation with guidance.				
Date-Result					

Fig. 41: Test Case 39

Test ID	T-40	Category	Functional	Severity	Minor
Objective	Save/Edit User Profile Information				
Steps	1. Go to Profile Page. 2. Edit 'Name' field. 3. Click 'Update Profile'. 4. Refresh page.				
Expected	Database updates; new name is displayed correctly in the UI.				
Date-Result					

Fig. 42: Test Case 40

Test ID	T-41	Category	Functional	Severity	Major
Objective	Component Connection Validation				
Steps	1. In Pipeline Builder, try to connect 'Annotation' output to 'QC' input. 2. Observe the canvas behavior.				
Expected	Invalid logic connections are blocked.				
Date-Result					

Fig. 43: Test Case 41

Test ID	T-42	Category	Performance	Severity	Minor
Objective	Database Query Performance (Jobs List)				
Steps	1. Populate a test account with multiple previous jobs. 2. Navigate to Jobs Page. 3. Measure load time.				
Expected	Jobs list loads in < 2 seconds through efficient SQL pagination.				
Date-Result					

Fig. 44: Test Case 42

Test ID	T-43	Category	Functional	Severity	Minor
Objective	Handling of Concurrent Same-File Uploads				
Steps	1. Start uploading a FASTQ file. 2. Immediately start another upload of the same file. 3. Check for duplicates.				
Expected	System handles the conflict.				
Date-Result					

Fig. 45: Test Case 43

Test ID	T-44	Category	Integration	Severity	Major
Objective	Kubernetes Horizontal Pod Autoscaling				
Steps	1. Flood the system with 20 heavy compute jobs. 2. Monitor cluster nodes. 3. Check if new nodes are added.				
Expected	Cluster scales out automatically to provide resources for the load.				
Date-Result					

Fig. 46: Test Case 44

Test ID	T-45	Category	Compatibility	Severity	Major
Objective	Browser Compatibility - Safari/Firefox/Chrome				
Steps	1. Open Pipeline Builder in 3 different modern browsers. 2. Test drag-and-drop functionality in each.				
Expected	Core functionalities work identically across all supported browsers.				
Date-Result					

Fig. 47: Test Case 45

Test ID	T-46	Category	Functional	Severity	Major
Objective	Reset Password via Email				
Steps	1. Click 'Forgot Password' on Login page. 2. Provide valid email. 3. Follow the link in the simulated email. 4. Change password.				
Expected	Password is updated successfully; user can log in with the new password.				
Date-Result					

Fig. 48: Test Case 46

Test ID	T-47	Category	Functional	Severity	Minor
Objective	Export Workflow Metadata as JSON				
Steps	1. Open a saved workflow. 2. Click 'Export as JSON'. 3. Save the file. 4. Open and verify structure.				
Expected	JSON file contains correct tool sequence and parameter configurations.				
Date-Result					

Fig. 49: Test Case 47

Test ID	T-48	Category	Documentation	Severity	Minor
Objective	Verify README/Instructions Link				
Steps	1. Locate 'Help/Docs' link in the sidebar. 2. Click and navigate to documentation page. 3. Check for 'Getting Started' guide.				
Expected	Documentation is accessible and provides relevant setup instructions for new researchers.				
Date-Result					

Fig. 50: Test Case 48

Test ID	T-49	Category	Security	Severity	Major
Objective	Multi-Factor Authentication (MFA) Setup				
Steps	1. Navigate to 'User Profile'. 2. Select 'Security Settings'. 3. Click 'Enable 2FA'. 4. Enter the security code sent via email.				
Expected	MFA is successfully linked; system prompts for code on next login.				
Date-Result					

Fig. 51: Test Case 49

Test ID	T-50	Category	Functional	Severity	Minor
Objective	Batch Workflow Deletion				
Steps	1. Go to 'My Workflows'. 2. Select five different workflows using checkboxes. 3. Click the 'Bulk Delete' button. 4. Confirm the action in the pop-up modal.				
Expected	All five workflows are removed from the list and the database.				
Date-Result					

Fig. 52: Test Case 50

Test ID	T-51	Category	Integration	Severity	Major
Objective	External S3 Bucket Mounting				
Steps	1. Navigate to 'External Storage'. 2. Input AWS Access/Secret keys. 3. Provide a valid Bucket Name. 4. Click 'Mount' and browse files.				
Expected	External files are visible and selectable within the CASSIE interface.				
Date-Result					

Fig. 53: Test Case 51

Test ID	T-52	Category	Security	Severity	Major
Objective	Account Session Revocation				
Steps	1. Log into CASSIE on two different browsers. 2. In Browser A, go to 'Active Sessions'. 3. Click 'Log out of all other sessions'. 4. Attempt to navigate in Browser B.				
Expected	Browser B is automatically redirected to the login page.				
Date-Result					

Fig. 54: Test Case 52

Test ID	T-53	Category	Reliability	Severity	Critical
Objective	DB Failover Recovery				
Steps	1. Start a workflow job. 2. Simulate a database primary node failover (admin action). 3. Wait for the failover to complete. 4. Refresh the job status page.				
Expected	The job status remains accurate and the UI recovers connection to the DB.				
Date-Result					

Fig. 55: Test Case 53

Test ID	T-36	Category	Integration	Severity	Major
Objective	Docker Container Version Locking				
Steps	1. Inspect a 'Workflow Step'. 2. Check the Docker image tag. 3. Ensure it's not using 'latest'. 4. Run job.				
Expected	System uses specific version tags to prevent changes in results over time.				
Date-Result					

Fig. 38: Test Case 36

Test ID	T-54	Category	Functional	Severity	Major
Objective	Custom Tool Container Import				
Steps	1. Go to 'Admin Tool Settings'. 2. Click 'Add Custom Docker Image'. 3. Enter the Docker Hub URI. 4. Define the command-line interface (CLI) mapping.				
Expected	Tool is successfully indexed and appears in the workflow builder.				
Date-Result					

Fig. 56: Test Case 54

Test ID	T-55	Category	Functional	Severity	Major
Objective	Automated Storage Cleanup Policy				
Steps	1. Go to 'Settings'. 2. Set 'Auto-delete intermediate files' to '7 days'. 3. Run a workflow. 4. Check file availability after the specified period.				
Expected	Intermediate files are automatically purged from S3 after 7 days.				
Date-Result					

Fig. 57: Test Case 55

Test ID	T-56	Category	Functional	Severity	Major
Objective	User Account Deletion				
Steps	1. Go to 'Profile Settings'. 2. Click 'Delete My Account'. 3. Confirm identity via password. 4. Click 'Permanently Delete'.				
Expected	Account is deactivated; personal data and files are queued for purging.				
Date-Result					

Fig. 58: Test Case 56

Test ID	T-57	Category	Performance	Severity	Minor
Objective	Export Large Result Logs				
Steps	1. Locate a job with a 100MB+ log file. 2. Click 'Download Full Log'. 3. Monitor the download speed. 4. Verify the integrity of the downloaded file.				
Expected	Log file downloads without truncation or connection timeout.				
Date-Result					

Fig. 59: Test Case 57

Test ID	T-58	Category	Functional	Severity	Minor
Objective	Notification Preferences				
Steps	1. Go to 'Notifications Settings'. 2. Toggle 'Email on Job Completion' to OFF. 3. Run a workflow. 4. Verify that no email is received.				
Expected	System respects the user preference and only sends requested alerts.				
Date-Result					

Fig. 60: Test Case 58

Test ID	T-59	Category	Functional	Severity	Minor
Objective	Resetting Workflow to Default				
Steps	1. Open a heavily modified workflow. 2. Click 'Reset to Defaults'. 3. Confirm the action. 4. Check the parameter values.				
Expected	All tools revert to their original factory-set parameters.				
Date-Result					

Fig. 61: Test Case 59

Test ID	T-60	Category	Security	Severity	Major
Objective	Password Complexity Enforcement				
Steps	1. Go to 'Change Password'. 2. Attempt to set password to '123456'. 3. Attempt with no special characters. 4. Enter a valid complex password.				
Expected	Weak attempts are rejected with clear requirements; valid one is accepted.				
Date-Result					

Fig. 62: Test Case 60

Test ID	T-61	Category	Functional	Severity	Major
Objective	Partial Job Resubmission				
Steps	1. Identify a multi-step job where step 3 failed. 2. Click 'Resume from Failed Step'. 3. Update the faulty parameter. 4. Click 'Restart'.				
Expected	System skips steps 1 and 2, restarting only from step 3 onwards.				
Date-Result					

Fig. 63: Test Case 61

Test ID	T-62	Category	Performance	Severity	Major
Objective	UI Responsiveness during Heavy Upload				
Steps	1. Start a multi-file batch upload (20+ files). 2. While uploading, navigate to the 'Jobs' page. 3. Edit a workflow. 4. Return to the 'Data' page.				
Expected	The UI remains responsive and does not lag during background uploads.				
Date-Result					

Fig. 64: Test Case 62

Test ID	T-63	Category	Functional	Severity	Minor
Objective	Interactive Tool Modal Validation				
Steps	1. Open a tool configuration modal. 2. Enter a string into a numeric-only field. 3. Attempt to click 'Apply'. 4. Correct the value to a number.				
Expected	Apply' is disabled and an error message appears until input is valid.				
Date-Result					

Fig. 65: Test Case 63

Test ID	T-64	Category	Functional	Severity	Minor
Objective	Search by File Extension				
Steps	1. Go to the Data Manager. 2. Type ".vcf" in the search bar. 3. Verify the results list. 4. Clear search and type ".fastq".				
Expected	Only files matching the specific extension are displayed in each case.				
Date-Result					

Fig. 66: Test Case 64

Test ID	T-65	Category	Functional	Severity	Minor
Objective	Rename Active Project				
Steps	1. Select an existing project. 2. Click 'Rename'. 3. Enter a name. 4. Verify the name updates in the breadcrumb and sidebar.				
Expected	Project name updates globally without breaking file paths.				
Date-Result					

Fig. 67: Test Case 65

Test ID	T-66	Category	Functional	Severity	Minor
Objective	Sorting Jobs by Duration				
Steps	1. Go to the 'Job History' table. 2. Click the 'Duration' column header. 3. Click it again to reverse the order. 4. Verify the top/bottom results.				
Expected	Table correctly sorts jobs from longest to shortest and vice versa.				
Date-Result					

Fig. 68: Test Case 66

Test ID	T-67	Category	Performance	Severity	Minor
Objective	Large JSON Metadata Export				
Steps	1. Create a workflow with 50+ tool nodes. 2. Click 'Export as JSON'. 3. Open the file in a text editor. 4. Validate the JSON schema structure.				
Expected	Export is generated quickly and contains all 50+ tool definitions.				
Date-Result					

Fig. 69: Test Case 67

Test ID	T-68	Category	Reliability	Severity	Critical
Objective	Kubernetes Node Scaling				
Steps	1. Submit 10 heavy genomic pipelines simultaneously. 2. Monitor the cluster via Admin dashboard. 3. Observe new nodes spinning up. 4. Verify jobs complete.				
Expected	The cluster scales out to meet demand and scales back in after completion.				
Date-Result					

Fig. 70: Test Case 68

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

During the development of CASSIE, various subjects affecting the public are considered, and each one is given a priority level between 1 (lowest) and 10 (highest), indicating the effects of CASSIE on that particular consideration.

Public Health Considerations - Priority 3

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 - Priority 6

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 - Priority 5

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 - Priority 7

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 - Priority 2

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 - Priority 4

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 - Priority 5

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 - Priority 7

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.

6.2 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.

7. Teamwork Details

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.

7.1 Contributing and Functioning Effectively on the Team

Ege Şirvan contributed mainly to backend development and system integration tasks. He focused on the communication between different subsystems and designed the overall system architecture. He also selected and configured the genomic tools.

Eren Aslan contributed to the implementation of the cloud-side of the project and container orchestration. He worked on integrating Docker-based tools and Kubernetes orchestration into

the platform. He worked on the workflow orchestration logic and integration between the workflow manager and the cloud infrastructure.

Arda Kırcı contributed to the backend development and system architecture design of the project. He participated in system design decisions and helped improve workflow execution management. He also contributed to documentation and design discussions.

Emre Can Yoloğlu mainly worked on the user interface of the project. He focused on building an interface that simplifies complex bioinformatics operations for the average user. In addition to UI implementation, he also contributed to debugging, testing, and integration tasks to ensure communication between the frontend and backend services.

Osman Baktır contributed to documentation tasks. He also contributed to writing the reports and ensuring that the project documentation accurately reflects the implemented system.

7.2 Helping to Create a Collaborative and Inclusive Environment

To maintain a collaborative and inclusive environment, Ege Şirvan is actively involved in communication channels, including the team’s messaging group and regular meetings. He ensured everyone was informed about the project's progress and scheduled meetings as needed.

Eren Aslan maintained active communication with the team through meetings and online messaging platforms. He regularly shared updates about the development progress and regularly discussed the workflow and cluster structure so that every team member could contribute to them.

Arda Kırcı supported the collaborative environment by actively participating in both online and in-person meetings. He also provided his ideas on various technical problems.

Emre Can Yoloğlu contributed to maintaining a positive and inclusive team atmosphere by openly sharing ideas and participating in brainstorming sessions. He also helped other team members when technical challenges arose during tool integration or workflow configuration.

Osman Baktır contributed to team collaboration by supporting task coordination. He participated in discussions regarding task assignments and helped maintain a cooperative and respectful working environment.

7.3 Taking Lead Role and Sharing Leadership on the Team

Throughout the project, leadership responsibilities were shared among the team members depending on the tasks. Different members took the lead in areas where they had more experience and were responsible for resulting in a balanced workload for each member.

Ege Şirvan took the lead role in designing the overall software architecture and selection/implementation of genomic tools.

Eren Aslan assumed the leading role in the cloud-side infrastructure of the project. He led the implementation of container orchestration mechanisms and coordinated the integration of Docker-based genomic tools with Kubernetes-based workflow execution.

Arda Kırcı took a leading role in backend development and workflow execution management. He also contributed to improving workflow execution logic.

Emre Can Yoloğlu took the leading role in the development of the user interface. He coordinated the design and implementation of frontend components and ensured that the interface allowed users to easily configure workflows, upload data, and monitor execution results.

Osman Baktır assumed a leading role in the documentation of the project. He coordinated documentation tasks and ensured that the content of the project reports reflected the real system architecture, design decisions, and development progress.

8. 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 March 6, 2026].

[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 March 6, 2026].

[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 March 6, 2026].

[4] Open Container Initiative, *OCI Image Format Specification v1.0.0*, The Linux Foundation, 2017. Available: https://github.com/opencontainers/image-spec [Accessed March 7, 2026].

[5] Open Container Initiative, *OCI Runtime Specification v1.0.0*, The Linux Foundation, 2017. Available: https://github.com/opencontainers/runtime-spec [Accessed March 7, 2026].

[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 March 7, 2026].

[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 March 7, 2026].

[8] Seqera Labs, *Nextflow*, Seqera Labs, 2026. Available: https://www.nextflow.io/docs/latest/ [Accessed March 5, 2026].

[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 March 6, 2026].

[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 March 8, 2026].

[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 March 8, 2026]

[12] L. Stein, “GFF3 Specification: Generic Feature Format Version 3,” Sequence Ontology Project, 2013. Available: https://github.com/The-Sequence-Ontology/Specifications [Accessed March 8, 2026].

[13] The GENCODE Consortium, “GTF Format Specification,” EMBL-EBI/GENCODE Documentation, 2024. Available: https://www.gencodegenes.org/pages/data_format.html [Accessed March 8, 2026].

[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 March 8, 2026].

[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://academic.oup.com/bioinformatics/article/38/6/1497/6499262 [Accessed March 9, 2026].

[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 March 9, 2026].