Bilkent University
Department of Computer Engineering

Senior Design Project
T2504
Pathogenius

Analysis and Requirement Report

Nazlı Apaydın 22202104
Ege Ateş 22201914
Yiğit Ali Doğan 22202329
Yunus Günay 22203758
Ata Uzay Kuzey 22203050

Supervisor: Can Alkan
Course Instructors: Mert Bıçakçı, İlker Burak Kurt

19.12.2025

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Contents

Analysis and Requirement Report

Project Short-Name: Pathogenius

1 Introduction

The rapid advances in sequencing technologies have enabled the analysis of genetic material obtained from clinical and environmental samples. Despite this progress, the software ecosystems required to interpret sequencing data remain complex, computationally expensive, and often dependent on high-performance servers or continuous cloud connectivity. These requirements pose a significant barrier in field settings, emergency response scenarios, and resource-limited environments, where the lack of access to metagenomic analysis can hinder timely pathogen identification and critical decision making.

Pathogenius addresses this challenge by providing a portable, offline metagenomic analysis platform capable of operating on modest local hardware such as consumer grade laptops. The system transforms raw long-read sequencing data into clear, species-level identification results through a structured and automated analysis workflow. At its core, Pathogenius employs Snakemake to coordinate a modular pipeline including quality control, preprocessing and taxonomic classification. The classification stage relies on Kraken2, a memory efficient k-mer based classifier that compares read fragments against a locally stored reference database, enabling fast and reproducible analysis without reliance on external services.

To ensure accessibility to non-expert users, Pathogenius provides a locally running Electron.js based graphical interface that abstracts complex command-line operations and offers real time feedback on system resources and workflow progress. A key innovation of the platform is its integration of a local AI assistant, implemented using small or quantized language models, which translates technical analysis results into readable and actionable summaries while preserving full offline operation. This design ensures that sensitive data, including potential human host DNA, remains transient and securely contained within the local environment.

Although the current implementation is optimized for CPU-based execution, the system architecture supports future integration of NVIDIA CUDA acceleration, enabling performance scaling on compatible hardware. Positioned as a discovery-oriented decision support tool rather than a definitive diagnostic service, Pathogenius is intended for use especially in water based pathogen surveillance, emergency response operations, field hospitals, and clinics operating under infrastructure constraints. By combining unbiased metagenomic analysis, offline capability, and intuitive visualization, the platform bridges the gap between advanced genomic methods and practical field deployment.

2 Current System

Traditional pathogen detection approaches suffer from structural limitations that restrict their suitability for on-site and discovery oriented use. PCR is highly sensitive but laboratory-bound, qPCR and ddPCR are powerful yet costly and operationally complex, and NGS-based metagenomics enables unbiased discovery but typically requires substantial computational resources and expert interpretation [1]. As a result, existing systems built on these methods tend to optimize for a single dimension while failing to provide a portable and user-friendly solution. Pathogenius addresses these limitations by combining portable

long-read sequencing with an offline workflow, automated analysis workflow and an integrated user interface that enables pathogen identification without reliance on advanced bioinformatics expertise.

Commercial solutions such as Bio-Rad's IQ-Check tests [2], for example, rely on RT-qPCR workflows with carefully designed primers and probes to detect predefined pathogens, delivering results within 12-24 hours depending on the sample type. However, their primary weakness lies in their target specific design. Only anticipated pathogens can be detected, and assay redesign is required when new organisms are involved. Pathogenius overcomes this limitation by employing an unbiased metagenomic approach, in which raw FASTQ reads are classified against a locally stored reference database. This enables simultaneous screening of multiple organisms without prior assumptions about pathogen identity, making Pathogenius suitable for discovery driven field scenarios.

Similarly, Norgen Biotek's pathogen detection kits [3] follow a similar PCR-centered paradigm and depend on predefined target pathogens. These methods require thermal cycling equipment and rely on a steady supply of reagents, which restricts their flexibility in field settings. In contrast, Pathogenius relies on a locally built and versioned reference index, thereby recurring consumable costs and logistical dependencies after initial deployment.

Nanometa Live [4] represents one of the most advanced solutions for real-time metagenomic visualization. However, it still exhibits key limitations when considered for field deployment. Its operation assumes complex toolchain setup and command line driven workflows, and its real-time analysis is constrained by batch processing and BLAST-based validation, leading to increased latency on modest hardware. While highly effective for monitoring and visualization, Nanometa Live primarily functions as an analysis dashboard rather than a decision support system, and the interpretation of its analytical outputs remains challenging for non-expert users. Pathogenius addresses these gaps by providing structured result objects, confidence aware outputs, and intuitive visualizations, complemented by a local AI module that converts technical outputs into readable summaries for end users.

Academic work based on Oxford Nanopore Technologies' MinION device [5] has demonstrated that portable long-read sequencing can be used for offline, on-site pathogen detection with locally curated reference databases. However, such pipelines remain research oriented, requiring expert driven database curation, command-line execution, and manual interpretation of raw results. Pathogenius builds on these validated foundations but shifts the emphasis from technical feasibility to operational usability by integrating automated workflow execution, efficient k-mer based classification, persistent analysis management, and user-friendly interface.

3 Proposed System

3.1 Overview

Pathogenius is a modular metagenomic analysis platform that can run offline, designed to transform raw sequencing output into actionable, species-level pathogen identification through an automated local workflow. The system is organized as a set of cooperating layers, including the sequencing environment, reference data management, workflow execution, and frontend interaction. All components operate entirely within the local environment, ensuring independence from external infrastructure and network connectivity.

At the core of Pathogenius lies a Snakemake based workflow layer that manages all analysis steps through a reproducible pipeline. Raw FASTQ files generated in the sequencing environment are first processed by a preprocessing component that performs quality control and cleanup. In parallel, curated reference genomes stored locally in FASTA format are processed by Kraken2's built-in database construction utilities to generate a Kraken2-compatible k-mer index, which is managed as a dedicated index-building step within the workflow. This locally generated index is stored and reused across analyses. Cleaned reads are then classified using the Kraken2 classifier against the local index, producing taxonomic assignments that are subsequently aggregated and processed to compute confidence metrics and structured result files.

Pathogenius utilizes Electron.js to provide a locally running graphical interface that enables users to select local input files, monitor real-time workflow execution, and interpret analysis results. Processed outputs are transformed by an analyzer component into structured result objects, which are then presented through visualization modules displaying species-level findings using multiple chart types. To further ease interpretation for non-expert users, Pathogenius integrates a local AI component that converts structured analytical results into readable explanations and summaries, supporting informed interpretation while ensuring that all data remains confined to the local system.

3.2 Functional Requirements

These requirements define the specific behaviours and services the system must provide.

3.2.1 Main Requirements

3.2.1.1 Analysis & Workflow Management

- **FR-AM-001:** The system shall store all completed analyses in a persistent local database for authenticated users including analysis name, date and status.
- **FR-AM-002:** The system shall display a comprehensive analysis history table for authenticated users, showing analysis name, date, input file, and processing status.
- **FR-AM-003:** The system shall allow users to assign custom names and descriptions to analyses.
- **FR-AM-004:** The system shall provide search and filter capabilities, allowing users to locate specific past analyses by criteria such as name, date or detected pathogens.
- **FR-AM-005:** The system shall enable users to reopen completed analyses to view results.
- **FR-AM-006:** The system shall allow users to delete analyses with confirmation prompts.

- **FR-AM-007:** The system shall support export of results and archiving multiple analyses in different formats including PDF, JSON and CSV.
- **FR-AM-008:** The system shall use an offline large language model to further elaborate on the complex classification metrics into easy to read, human readable summary paragraphs.

3.2.1.2 FASTQ Processing & Classification

- **FR-FP-001:** The system shall accept long read FASTQ format files (compressed and uncompressed) as primary input.
- **FR-FP-002:** The system shall validate input files and display file metadata before processing.
- **FR-FP-003:** The system shall calculate and report confidence scores for each species identification, distinguishing between the high and low confidence scores for the analyses.
- **FR-FP-004:** The system shall coordinate the analysis pipeline using the Snakemake workflow engine, ensuring deterministic execution of preprocessing, classification and output stages.
- **FR-FP-005:** The system shall support checkpoint and resume functionality, allowing analyses interrupted by the system events to continue from the last successful workflow stage.
- **FR-FP-006:** The system shall generate preprocessing and classification quality reports.
- **FR-FP-007:** The system shall support batch processing of multiple FASTQ files to analyse and depict the results of all the input files.

3.2.1.3 Dataset Management

- **FR-DM-001:** The system shall maintain a local reference database composed of species specific FASTA files curated from the NCBI resources.
- **FR-DM-002:** The system shall provide functionality to update the database by querying an update server when network is available or importing from custom FASTA files locally.
- **FR-DM-003:** The system shall display information about the current database version including the number of species, last update date, and database size.

- **FR-DM-004:** The system shall support manual database curation, allowing users to add, remove, or modify taxonomic information and associated genomic data for specific species.

3.2.1.4 Notifications

- **FR-NT-001:** The system shall provide immediate visual notifications upon the successful completion of an analysis workflow. This includes updating the status in the analysis history and triggering a completion banner on the dashboard.
- **FR-NT-002:** In the event of a processing failure, the system shall alert the user with descriptive actionable error messages.
- **FR-NT-003:** For long running operations like k-mer classification or index building, the system shall display real time progress indicators. These updates must include the current workflow stage and the completion percentage.
- **FR-NT-004:** The system shall maintain a persistent history of all system notifications, accessible via a dedicated section.

3.2.1.5 User Registration and Authentication

- **FR-UA-001:** The system shall provide user registration functionality allowing new users to create accounts with username and password.The system must validate that the username is unique and the password is secure.
- **FR-UA-002:** User authentication is not mandatory. The system will support a 'Guest Mode' for immediate offline use. Authenticated users shall gain access to their local history.
- **FR-UA-003:** The system shall authenticate users through a login interface with secure credential validation.
- **FR-UA-004:** The system shall store user credentials securely using encryption.
- **FR-UA-005:** The system shall maintain separate user workspaces ensuring data privacy between users.
- **FR-UA-006:** The system shall accommodate multiple users in a single device with encryption to ensure that the users can keep their analysis secure.

3.2.2 Secondary Features

3.2.2.1 Realtime Data Display

- **FR-RT-001:** The system shall provide a real time progress bar indicating active Snakemake stage and the completion percentage.
- **FR-RT-002:** The interface shall dynamically update a preliminary results table, showing species names and use counts as they are identified.
- **FR-RT-003:** The system shall display live telemetry cards displaying real time utilization metrics for the host CPU/GPU usage and available RAM.
- **FR-RT-004:** The system shall provide estimated time remaining for ongoing analyses, adjusting the prediction based on the analyses.
- **FR-RT-005:** The system shall update classification statistics dynamically as processing progresses.
- **FR-RT-006:** The system shall display the number of reads processed per minute during analysis to provide feedback on the analysis efficiency.
- **FR-RT-007:** The system shall show preliminary results for completed workflow stages while subsequent stages continue processing.

3.2.2.2 GPU Acceleration

- **FR-GPU-001:** In later stages the system shall detect available NVIDIA CUDA-compatible GPUs and their available VRAM.
- **FR-GPU-002:** The system shall provide an user controlled option to enable or disable GPU acceleration for classification tasks.
- **FR-GPU-003:** The system shall automatically optimize workload distribution between CPU and GPU when acceleration is enabled.
- **FR-GPU-004:** The system shall fallback to CPU-only processing if GPU acceleration fails or is unavailable.
- **FR-GPU-005:** The system shall display GPU utilization metrics during accelerated processing.
- **FR-GPU-006:** The system shall support CUDA-accelerated k-mer matching operations for improved performance.

3.3 Non-Functional Requirements

These requirements define the quality attributes and operational constraints of Pathogenius, categorized according to the **ISO/IEC 25010** software quality standard.

3.3.1 Usability

- **NFR-USE-01**: The system shall provide a 100% graphical user interface (GUI) via Electron.js for all operations, ensuring that field personnel never need to interact with the underlying command-line or terminal.
- **NFR-USE-02**: The interface shall provide continuous visual feedback, including dynamic progress bars and stage-specific status indicators during long-running metagenomic workflows.
- **NFR-USE-03**: The results dashboard must visually differentiate between high and low-confidence species identifications using color-coding or uncertainty markers to support accurate clinical interpretation.
- **NFR-USE-04**: The system shall deliver descriptive, actionable error messages that specify the nature of the failure to facilitate rapid user-level troubleshooting.

3.3.2 Reliability

- **NFR-REL-01**: The system must maintain full functionality for preprocessing, k-mer classification, and AI summarization in environments with no internet connectivity.
- **NFR-REL-02**: The deployment package must bundle all necessary bioinformatics binaries, reference indices, and software dependencies to ensure complete autonomous operation upon installation.
- **NFR-REL-03**: The system shall handle malformed or truncated FASTQ entries by logging the specific error and skipping only the affected read to prevent entire analysis termination.
- **NFR-REL-04**: The system must ensure deterministic execution, producing bit-identical identification reports when provided with identical input data and reference databases.
- **NFR-REL-05**: The system shall strictly treat original raw FASTQ files as read-only, ensuring no modification or deletion occurs during any stage of the Snakemake pipeline.

3.3.3 Performance

- **NFR-PER-01**: The analysis engine must be optimized to operate on mid-range, consumer-grade laptops without requiring high performance computing systems.
- **NFR-PER-02**: The system shall allow users to set memory caps for the classification algorithms to prevent the process from exceeding the host machine's physical RAM.
- **NFR-PER-03**: Preprocessing and classification must be completed within a timeframe that allows for rapid field decision making, minimizing latency.
- **NFR-PER-04**: Computationally intensive tasks shall be executed as background processes to ensure the Electron frontend remains responsive to user navigation during the analysis.

3.3.4 Supportability

- **NFR-SUP-01**: The analysis pipeline must be implemented using modular Snakemake rules to facilitate the independent update or replacement of individual bioinformatics tools without system restructuring.
- **NFR-SUP-02**: The system shall generate persistent, human readable session logs that record timestamps, analysis parameters, and hardware utilization for diagnostic purposes.
- **NFR-SUP-03**: The system shall be deployable as a containerized application or managed environment to ensure consistent performance across different OS platforms like Windows or Linux.

3.3.5 Scalability

- **NFR-SCA-01**: The platform must support processing FASTQ files ranging from 1 MB to 10 GB, scaling its resource usage based on available disk space and VRAM.
- **NFR-SCA-02**: Upon initialization, the system must automatically detect available hardware resources, including the number of CPU cores and NVIDIA CUDA compatible GPUs.
- **NFR-SCA-03**: The classification module shall dynamically utilize multi threaded CPU processing or GPU accelerated kernels depending on detected hardware capability.
- **NFR-SCA-04**: The interface must allow users to import and index new FASTA files into the reference database to detect emerging pathogens without requiring core software updates.

3.4 Pseudo Requirements

A. Version Control and Data Management:

- Git will be used as the primary version control system to manage source code and documentation.
- GitHub will serve as the central repository for the project and will also host the project website via Github Pages (https://patho-genius.github.io).
- GitHub Issues will be used to track tasks, bugs, feature requests, and overall project progress.

B. Technology Stack and Development Tools

- Snakemake will be employed to implement the analysis workflow, ensuring reproducibility and modularity.
- Kraken2 will be used as the primary taxonomic classification engine.
- Local reference databases will be built from NCBI FASTA files using an index builder component.
- Electron.js will be used to develop the frontend.
- NVIDIA CUDA will be used to support GPU-accelerated classification in later stages of the project.

C. Offline Execution

- The system will be designed to operate fully offline after installation, without requiring internet connectivity for core functionality.
- Reference databases and generated indexes will be stored locally and reused across analyses.

D. Artificial Intelligence and Result Interpretation

- An open-source local language model will be integrated to generate readable summaries and explanations from structured analysis outputs.
- All language model inference will be performed locally and will not rely on external APIs or cloud-based services. Internet connectivity, when available, may be used only to select or update which local model is used.
- Generated explanations will be derived solely from locally produced analysis results to ensure data privacy.

E. Testing and Validation

- Unit testing will be performed for core components such as preprocessing, indexing, classification, and result parsing.
- Integration testing will be conducted to validate correct interaction between workflow steps and the frontend.
- Sample FASTQ datasets will be used to verify deterministic behavior and reproducibility of analysis results.

F. Collaboration and Communication

- Team communication will be conducted using platforms such as WhatsApp for asynchronous discussions.

- Online meeting tools such as Zoom or Google Meet will be used for synchronous project meetings and reviews.

3.5 System Models

3.5.1 Scenarios

Scenario 1:	User Login Process
Actor	User
Entry Condition(s)	The actor opens the Pathogenius application and views the login page. No user should be logged in.
Exit Condition(s)	The actor is successfully logged into the system and redirected to the dashboard.
Flow of Events	1. The actor enters their username in the username field. 2. The actor enters their password in the password field. 3. The actor clicks the “Login” button. 4. The system validates the provided credentials against the authentication service. 5. The system creates a session with appropriate permissions based on the user’s role. 6. The actor is redirected to the dashboard with their profile information displayed in the sidebar.
Alternative Flow	1. The actor provides incorrect username or password. 2. The system denies access and displays an error message. 3. The form remains on the login page, allowing retry.

Scenario 2:	User Sign in Process
Actor	User
Entry Condition(s)	The computer is connected to the Internet and the actor opens the sign in page.
Exit Condition(s)	The actor successfully creates a new user.
Flow of Events	1. The actor enters their username in the username field. 2. The actor enters their password in the password field. 3. The actor enters their password in the enter your password again field. 4. The actor clicks the “Sign in” button. 5. The system validates that the username does not exist and the passwords are identical. 6. The system registers the new user to the system.

	7. The actor is redirected to the login page.
Alternative Flow	1. The actor provides an already existing username or enters different passwords. 2. The system denies the action and displays an appropriate error message. 3. The form remains on the sign in page, allowing retry.

Scenario 3:	Guest Mode Login
Actor	User
Entry Condition(s)	The actor opens the Pathogenius application and needs offline/emergency access.
Exit Condition(s)	The actor enters the system in guest mode with limited permissions.
Flow of Events	1. The actor clicks the “Continue as Guest” button on the login page. 2. The system creates a guest session with read and analyze permissions. 3. The actor is redirected to the dashboard as “Guest User.” 4. The system tracks analyses created during the guest session locally.

Scenario 4:	User Logout
Actor	User
Entry Condition(s)	The actor is logged into the system.
Exit Condition(s)	The actor is successfully logged out and returned to the login page.
Flow of Events	1. The actor clicks the logout button in the sidebar. 2. The system terminates the current session. 3. The system clears user state and session data. 4. The actor is redirected to the login page.

Scenario 5:	View Dashboard
Actor	User
Entry Condition(s)	The actor is logged into the system.
Exit Condition(s)	The actor views the system overview with resource statistics and running analysis status.

Flow of Events	1. The actor navigates to the “Dashboard” section via the sidebar. 2. The system retrieves and displays current system statistics (RAM usage, CPU usage). 3. The system checks for any running analyses. 4. If an analysis is running, the system displays a progress banner with analysis name, status, and completion percentage.

Scenario 6:	Create New Analysis - Select Input Files
Actor	User
Entry Condition(s)	The actor is logged in and navigates to the “New Analysis” section.
Exit Condition(s)	The actor has selected one or more FASTQ files for analysis.
Flow of Events	1. The actor clicks the “Select Files” button or drags files into the upload zone. 2. The system opens a file selection dialog filtered for FASTQ files. 3. The actor selects one or more FASTQ files (Nanopore-style long reads). 4. The system displays the selected files in a list with filenames. 5. The actor proceeds to the next configuration step.
Alternative Flow (Drag and Drop)	1. The actor drags FASTQ files from their file system into the upload zone. 2. The system highlights the drop zone during the drag operation. 3. The system processes the dropped files and displays them in the file list.
Alternative Flow (No Files Selected)	1. The actor attempts to proceed without selecting files. 2. The system displays an alert: “Please select at least one FASTQ file.” 3. The actor remains on the file selection step.

Scenario 7:	Import Custom Database
Actor	User
Entry Condition(s)	The actor is logged in with admin privileges and navigates to the Database section.
Exit Condition(s)	A custom reference database is imported into the system.
Flow of Events	1. The actor navigates to the “Database” section. 2. The actor clicks the “Import Custom Database” button. 3. The system opens a folder selection dialog. 4. The actor selects a folder containing FASTA files.

	5. The system validates the selected folder structure. 6. The system begins the database import process. 7. The system displays a success message upon completion.
Alternative Flow	1. The selected folder does not contain valid database files. 2. The system displays an error. 3. The actor is prompted to select a different folder.

Scenario 8:	Configure Analysis Parameters
Actor	User
Entry Condition(s)	The actor has selected input files in Step 1 of the analysis wizard.
Exit Condition(s)	The actor has configured all analysis parameters.
Flow of Events	1. The actor enters an analysis name (e.g., “Patient_001_Sample”). 2. The actor selects a sample type from the dropdown (Clinical Sample, Environmental, Food Safety, Other). 3. The actor selects a reference database (NCBI RefSeq 2025, Bacteria Only, Viral Only, Custom Database). 4. The actor adjusts the confidence threshold using the slider (default: 0.7). 5. The actor proceeds to the review step.
Alternative Flow	1. The actor attempts to proceed without entering an analysis name. 2. The system displays an alert: “Please enter an analysis name.” 3. The actor remains on the configuration step.

Scenario 9:	Review and Start Analysis
Actor	User
Entry Condition(s)	The actor has completed configuration in the analysis wizard.
Exit Condition(s)	The pathogen analysis is successfully started and the Snakemake workflow begins.
Flow of Events	1. The system displays a summary of the analysis configuration (name, files, sample type, database). 2. The actor reviews the configuration details. 3. The actor clicks the “Start Analysis” button. 4. The system creates an output directory for the analysis results. 5. The system generates a Snakemake configuration file. 6. The system initiates the Snakemake workflow with Kraken2 classification.

	7. The actor is redirected to the Results page to monitor progress.

Scenario 10:	Monitor Running Analysis
Actor	User
Entry Condition(s)	An analysis has been started and is currently running.
Exit Condition(s)	The actor monitors the analysis progress in real-time.
Flow of Events	1. The actor views the running analysis banner on the Dashboard or Results page. 2. The system displays the analysis name, current status, and progress percentage. 3. The system updates the progress bar as the workflow progresses through stages: - Preprocessing (Quality filtering reads) - Classifying (Running Kraken2 classification) - Processing (Processing classification results) - Finalizing (Generating reports) 4. Upon completion, the system updates the status to “Completed” and moves the analysis to history.
Alternative Flow	1. The analysis encounters an error during processing. 2. The system updates the status to “Failed” and displays the error message. 3. The analysis is moved to history with failed status.

Scenario 11:	Cancel Running Analysis
Actor	User
Entry Condition(s)	An analysis is currently running.
Exit Condition(s)	The running analysis is cancelled.
Flow of Events	1. The actor navigates to the Results page and locates the running analysis. 2. The actor clicks the “Cancel” button next to the running analysis. 3. The system prompts for confirmation: “Are you sure you want to cancel this analysis?” 4. The actor confirms cancellation. 5. The system terminates the Snakemake workflow process. 6. The analysis is marked as “Cancelled” and moved to history.

Scenario 12:	View Analysis History
Actor	User
Entry Condition(s)	The actor is logged in.
Exit Condition(s)	The actor views the list of all past analyses.
Flow of Events	1. The actor navigates to the “Results” section via the sidebar. 2. The system retrieves and displays a table of completed, failed, and cancelled analyses. 3. Each row shows: Analysis Name, Date, Sample Type, Status, Pathogens Detected, and Action buttons. 4. The actor can search/filter analyses using the search input.

Scenario 13:	View Detailed Analysis Results
Actor	User
Entry Condition(s)	The actor is logged in and an analysis has been completed successfully.
Exit Condition(s)	The actor views the detailed pathogen identification results.
Flow of Events	1. The actor clicks “View Report” on a completed analysis in the history table. 2. The system retrieves the analysis results from the stored JSON file. 3. The system displays the detailed result view including: - Analysis summary (total reads, classified reads, classification rate) - Quality metrics (average quality, high-quality rate, mean coverage) - Pathogen cards showing detected pathogens with abundance, confidence, and risk level - Taxonomy breakdown (bacteria, viruses, etc.) 4. The actor can switch between tabs.

Scenario 14:	Export Analysis Results
Actor	User
Entry Condition(s)	The actor is viewing detailed results of a completed analysis.
Exit Condition(s)	The analysis results are exported in the selected format.

Flow of Events	1. The actor clicks the “Export” button on the results detail view. 2. The actor selects the export format (PDF, JSON, CSV). 3. The system generates the export file with all relevant data. 4. The system prompts the actor to select a save location. 5. The file is saved to the specified location.
Alternative Flow	View Detailed Analysis Results

Scenario 15:	Check for Database Updates
Actor	User
Entry Condition(s)	The actor is logged in and the system has network connectivity.
Exit Condition(s)	The system checks for and reports available database updates.
Flow of Events	1. The actor navigates to the “Database” section. 2. The actor clicks the “Check for Updates” button. 3. The system queries the update server for newer database versions. 4. The system displays update availability status and version information. 5. If updates are available, the actor can initiate the download.
Alternative Flow	1. The system cannot reach the update server. 2. The system displays: “Update check requires network connection.”

Scenario 16:	Add New Species to The Database
Actor	User
Entry Condition(s)	The actor is logged in and navigates to the Database section.
Exit Condition(s)	A new species and its associated genomic data are successfully integrated into the reference database.
Flow of Events	1. The actor navigates to the “Database” page from the sidebar. 2. The actor scrolls to the “Custom Species” section. 3. The actor clicks the “Add FASTA” button. 4. The system opens a file selection dialog. 5. The actor selects a FASTA file (.fasta or .fa) from their local filesystem. 6. The system prompts the actor to enter species metadata (species name, taxonomic ID, type). 7. The actor fills in the required metadata fields.

	8. The system validates the FASTA file format and metadata. 9. The system indexes the sequences and adds the species to the local database. 10. The system displays the new species in the “Custom Species” list with its metadata.
Alternative Flow (Invalid File Format)	1. The actor selects a file that is not a valid FASTA format. 2. The system displays an error message indicating the file format is invalid. 3. The system prompts the actor to select a valid FASTA file.
Alternative Flow (Duplicate Species)	1. The actor attempts to add a species that already exists in the database. 2. The system warns the actor about the duplicate entry. 3. The actor can choose to update the existing entry or cancel the operation.

Scenario 17:	Remove Species From Database
Actor	User
Entry Condition(s)	The actor is logged in and navigates to the Database section. At least one custom species entry exists in the database. There is no ongoing analysis.
Exit Condition(s)	The selected species entry is successfully removed from the local reference database.
Flow of Events	1. The actor navigates to the “Database” page from the sidebar. 2. The actor scrolls to the “Custom Species” section. 3. The actor locates the species they want to remove from the list. 4. The actor clicks the “Remove” button next to the species entry. 5. The system displays a confirmation dialog asking if the actor wants to remove the species. 6. The actor confirms the removal. 7. The system removes the species entry from the database index. 8. The system updates the species list to reflect the removal. 9. The system displays a success notification.
Alternative Flow	1. The actor clicks “Cancel” in the confirmation dialog. 2. The system closes the dialog and the species remains in the database.

Scenario 18:	Edit Taxonomic Information
Actor	User
Entry Condition(s)	The actor is logged in and navigates to the Database section. At least one custom species entry exists in the database. There is no ongoing analysis.
Exit Condition(s)	The taxonomic information for the selected species is successfully updated.
Flow of Events	1. The actor navigates to the “Database” page from the sidebar. 2. The actor scrolls to the “Custom Species” section. 3. The actor locates the species they want to edit. 4. The actor clicks the “Edit” button next to the species entry. 5. The system displays an edit dialog with the current species information. 6. The actor modifies the desired fields (species name, taxonomic ID, or type). 7. The actor confirms the changes. 8. The system validates the updated information. 9. The system saves the updated metadata to the database. 10. The system displays the updated species information in the list. 11. The system shows a success notification.
Alternative Flow (Invalid Taxonomic ID)	1. The actor enters an invalid taxonomic ID format. 2. The system displays a validation error. 3. The actor corrects the taxonomic ID and resubmits.
Alternative Flow (Cancel Edit)	1. The actor clicks “Cancel” or closes the edit dialog. 2. The system discards the changes and retains the original information.

Scenario 19:	View System Resources
Actor	User
Entry Condition(s)	The actor is logged in to the application.
Exit Condition(s)	The actor views the current system resource usage including CPU, RAM, GPU, and storage.
Flow of Events	1. The actor navigates to the “Dashboard” page from the sidebar. 2. The system automatically loads and displays current system resource statistics. 3. The actor views the “System Resources” card.
Alternative Flow (Resource Warning)	1. A system resource exceeds the warning threshold (e.g., storage below 10%).

	2. The system highlights the resource in red/warning color. 3. The system displays a warning notification to the actor.
Alternative Flow (GPU Not Available)	1. The system detects that the GPU is not available. 2. The system displays “Not Available” in the GPU status. 3. The system notifies the actor that analyses may run slower without GPU acceleration.

Scenario 20:	Search and Filter Analyses
Actor	User
Entry Condition(s)	The actor is logged in and on the Results page with multiple analyses in history.
Exit Condition(s)	The actor successfully filters and finds specific analyses based on search criteria.
Flow of Events	1. The actor navigates to the “Results” page from the sidebar. 2. The system displays the Analysis History table with all completed analyses. 3. The actor locates the search box above the history table. 4. The actor types a search term (analysis name, sample type, or date). 5. The system filters the table in real-time as the actor types. 6. The system displays only analyses matching the search criteria. 7. The actor can view the filtered results and select one to view details.
Alternative Flow	1. The actor enters a search term that matches no analyses. 2. The system displays an empty table with a message “No analyses found.” 3. The actor can clear the search to view all analyses again.

Scenario 21:	Change User Password
Actor	Registered User
Entry Condition(s)	The actor has a registered username in the system
Exit Condition(s)	The actor changes their password.
Flow of Events	1. The actor clicks on the change password button 2. The actor enters their old password, and enters their new password twice 3. The system checks the validity of the credentials and checks that the two passwords are the same.

	4. The password is changed in the database.
Alternative Flow	1. The actor enters their old password incorrectly. 2. The system denies changing of the password.

Scenario 22:	Admin Changes User's Password
Actor	Admin
Entry Condition(s)	The actor is logged in with administrator privileges. The actor has navigated to the User Management section and at least one registered user exists in the system.
Exit Condition(s)	The selected user's password is successfully changed by the administrator.
Flow of Events	1. The actor accesses the "User Management" section. 2. The system displays a list of all registered users. 3. The actor locates the user whose password needs to be changed. 4. The actor clicks the "Reset Password" button next to the user entry. 5. The actor enters a new temporary password for the user. 6. The actor confirms the new password by re-entering it. 7. The system validates the password meets security requirements. 8. The system updates the user's password in the database. 9. The system optionally sends a notification to the affected user.
Alternative Flow	1. The actor enters a password that does not meet security requirements. 2. The system displays an error message indicating the password requirements. 3. The actor enters a valid password that meets the requirements.

Scenario 23:	Download Results from The Cloud
Actor	Registered User
Entry Condition(s)	The actor is logged in with a registered account. The system has an active internet connection. The actor has completed analyses stored on a remote server/cloud.
Exit Condition(s)	The analysis results are successfully downloaded from the internet to the local system.
Flow of Events	1. The actor navigates to the "Results" page from the sidebar. 2. The actor clicks on the "Cloud Results" tab.

	3. The system connects to the remote server and retrieves the list of available results. 4. The system displays a list of analysis results stored remotely. 5. The actor selects one or more results to download. 6. The actor clicks the “Download” button. 7. The system displays a download progress indicator. 8. The system downloads the selected results to the local storage. 10. The system adds the downloaded results to the local Results history. 11. The system displays a success notification with the download location.
Alternative Flow	1. The download process is interrupted due to network issues. 2. The system displays an error message with the failure reason. 3. The system offers the option to retry the download.

Scenario 24:	Upload Result to The Cloud
Actor	Registered User
Entry Condition(s)	The actor is logged in with a registered account. The system has an active internet connection. The actor has completed analyses on the local machine.
Exit Condition(s)	The encrypted analysis results are successfully uploaded to the internet.
Flow of Events	1. The actor navigates to the “Results” page from the sidebar. 2. The actor clicks the “Upload to Cloud” button of a completed analysis. 3. The system prompts for confirmation and displays data privacy notice. 4. The actor confirms the upload. 5. The system displays an upload progress indicator. 6. The system encrypts the data before transmission. 7. The system uploads the results to the remote server. 8. The system verifies the upload was successful. 9. The system marks the result as “Synced” in the local history. 10. The system displays a success notification.
Alternative Flow	1. The upload process is interrupted due to network issues. 2. The system displays an error message with the failure reason. 3. The system offers the option to retry the upload. 4. The local copy of the results remains intact.

Scenario 25:	Forgot Password
Actor	Registered User
Entry Condition(s)	The actor is on the login page. The actor has a registered account with a valid email address.
Exit Condition(s)	The actor successfully resets their password and can log in with the new credentials.
Flow of Events	1. The actor navigates to the login page. 2. The actor clicks the “Forgot Password?” link below the login form. 3. The system displays the password recovery form. 4. The actor enters their registered email address or username. 5. The actor clicks the “Send Recovery Link” button. 6. The system validates if the email/username exists in the database. 7. The system generates a secure password reset token. 8. The system sends a password reset email to the registered email address. 9. The system displays a confirmation message that the email has been sent. 10. The actor opens the email and clicks the password reset link. 11. The system validates the reset token and displays the password reset form. 12. The actor enters a new password. 13. The actor confirms the new password by re-entering it. 14. The system validates the new password meets security requirements. 15. The system updates the password in the database. 16. The system displays a success message and redirects to the login page. 17. The actor logs in with the new password.
Alternative Flow (Email/Username Not Found)	1. The actor enters an email or username that is not registered. 2. The system displays a generic message “If this account exists, a recovery email has been sent.” 3. No email is sent, but no specific error is shown (security best practice).
Alternative Flow (Link Expired)	1. The actor clicks a password reset link that has expired (e.g., after 24 hours). 2. The system displays a message that the link has expired. 3. The system prompts the actor to request a new password reset link.

3.5.2 Use Case Model

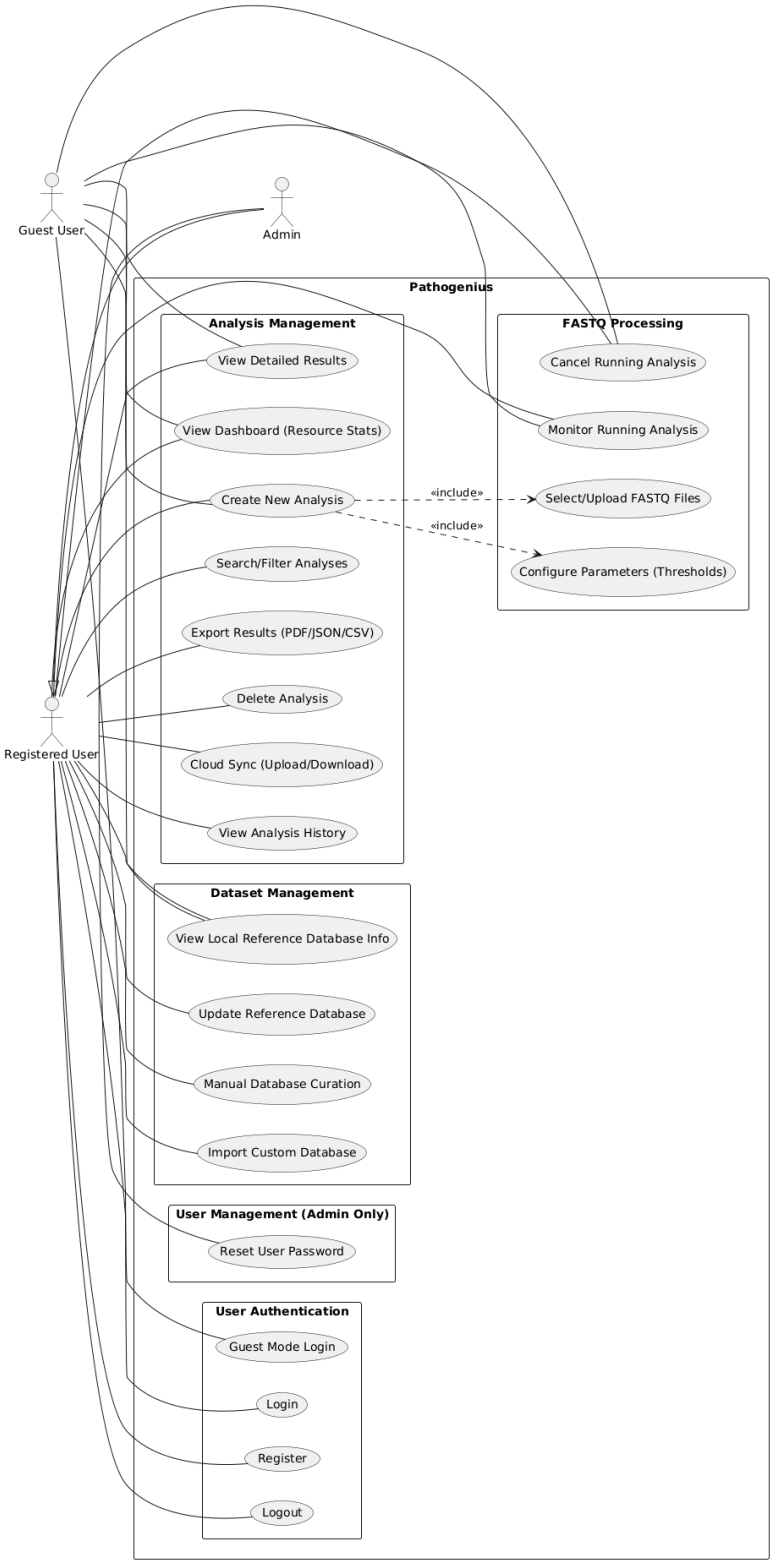

Figure 1. *Use Case Model of Pathogenius.*

3.5.3 Object and Class Model

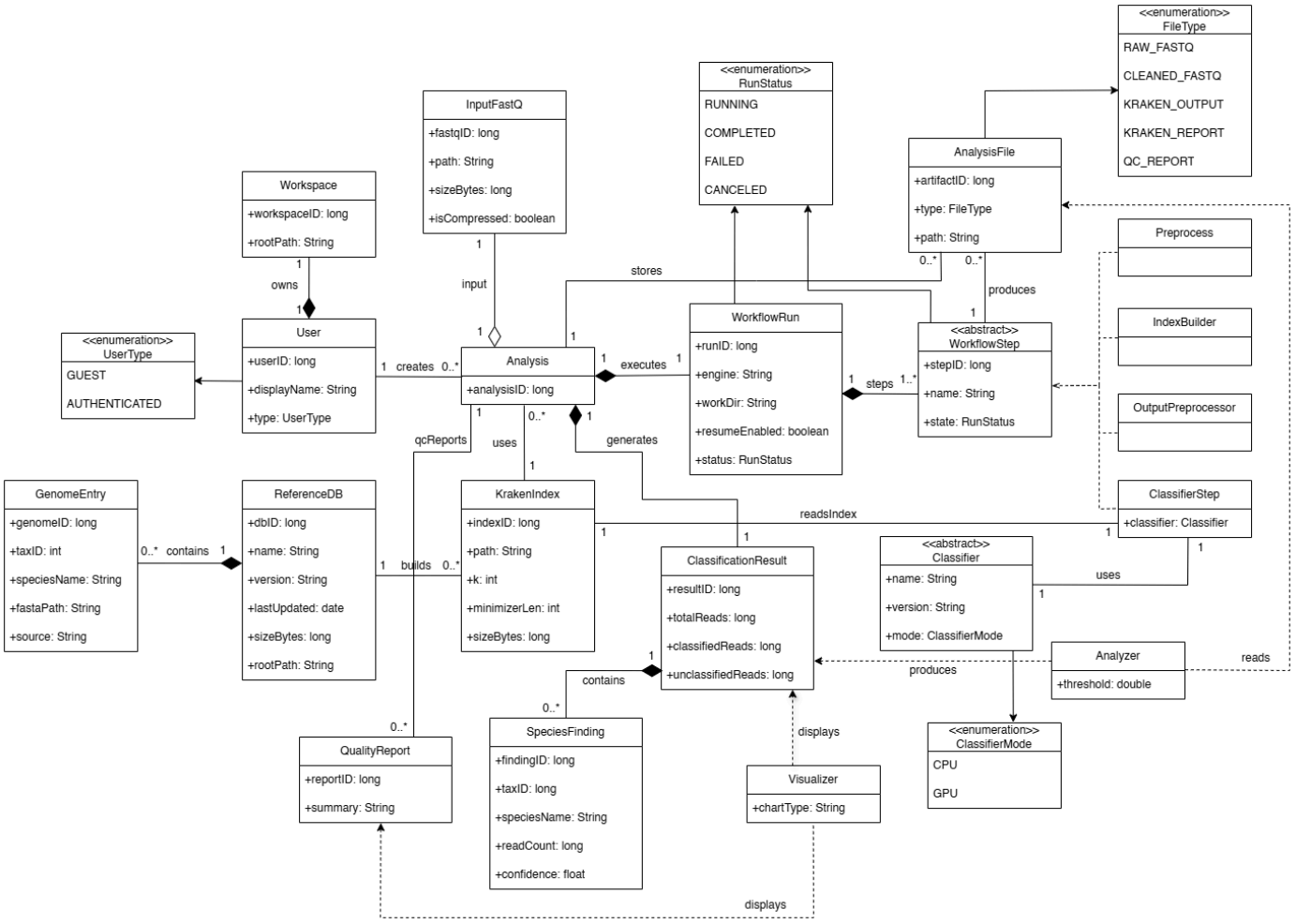

Figure 2. *Object & Class Model of Pathogenius.*

3.5.4 Dynamic Models

3.5.4.1 Activity Diagrams

3.5.4.1.1 Authentication

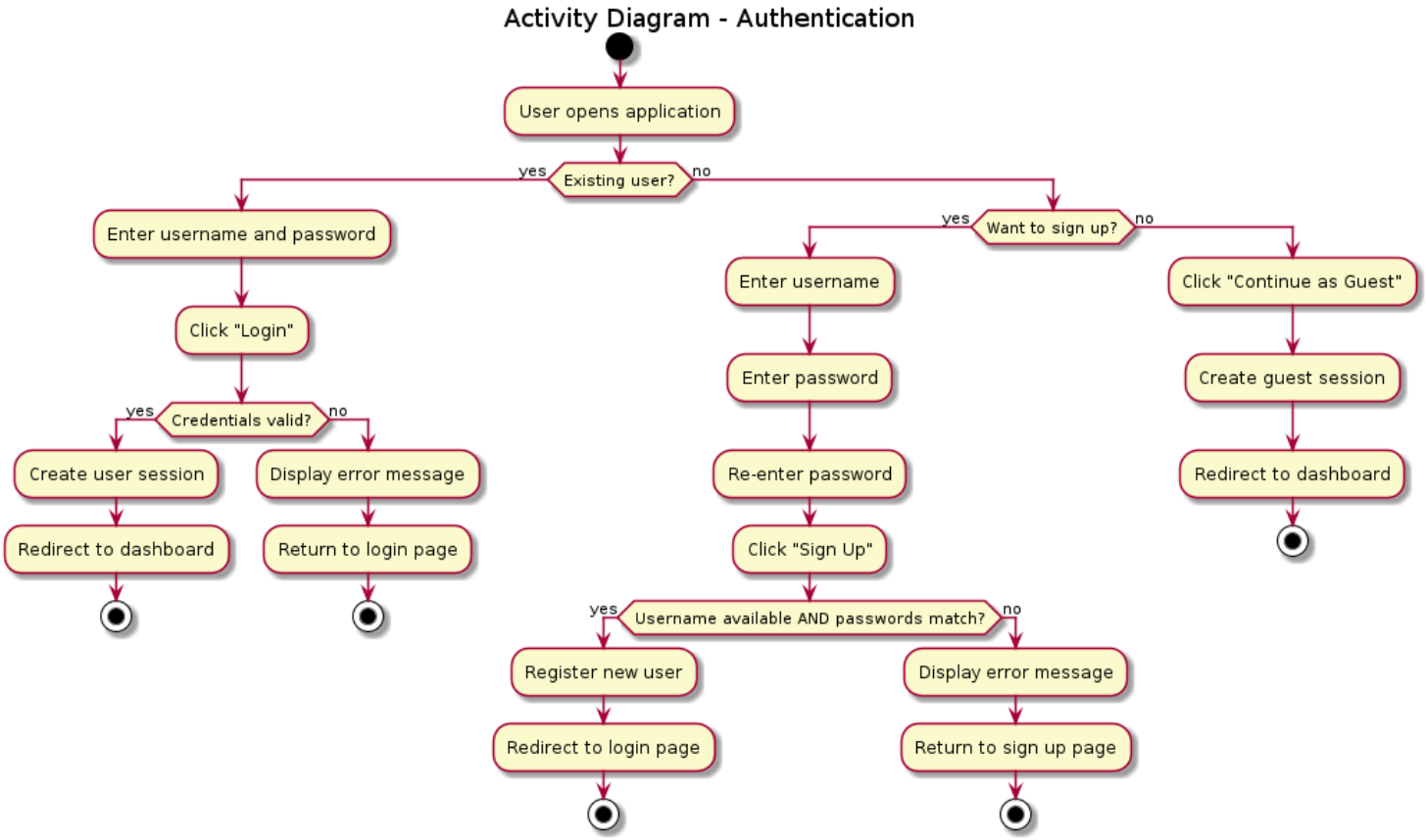

Figure 3. *Activity Diagram of Authentication.*

3.5.4.1.2 Analysis Activity Diagram

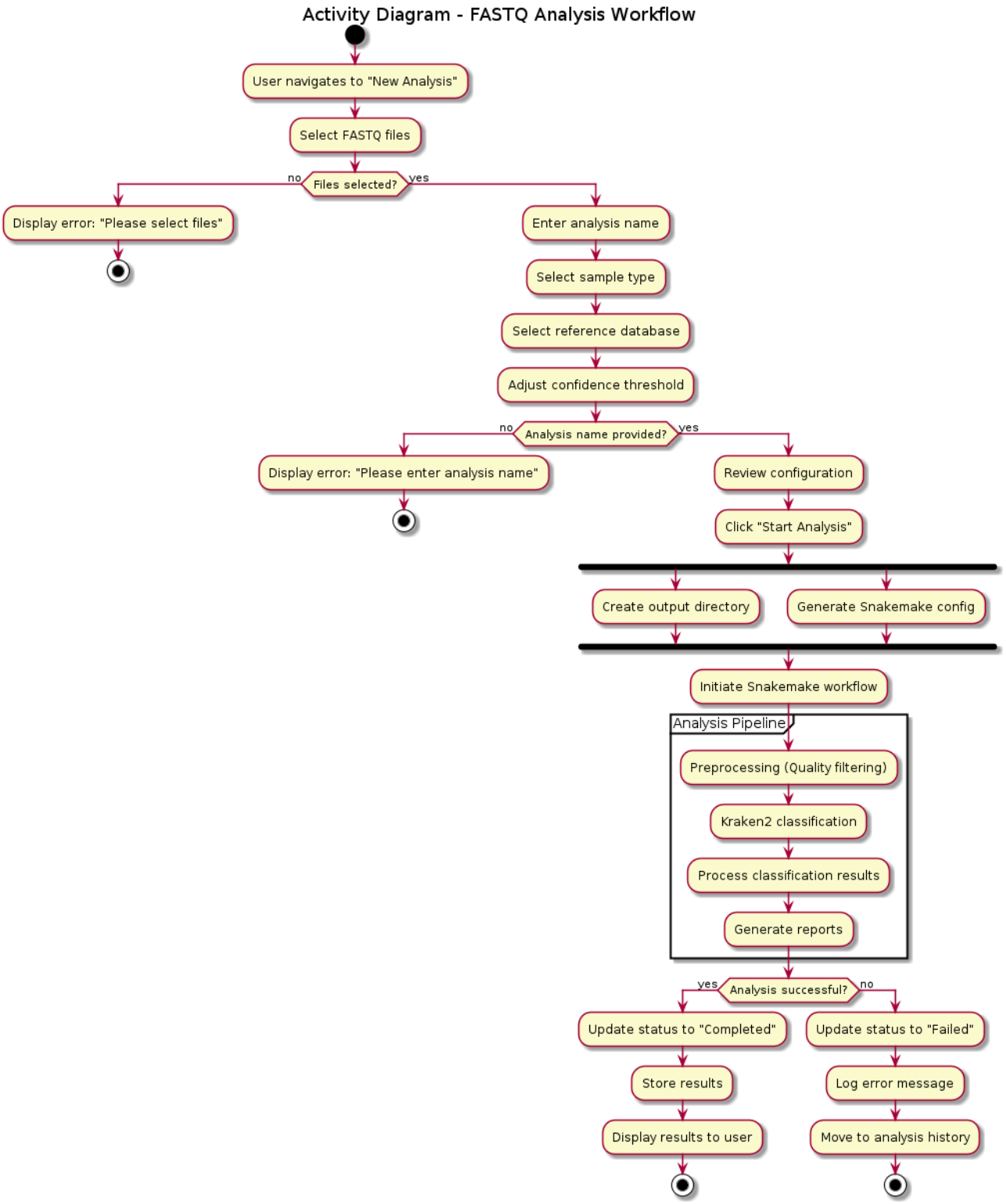

Figure 4. *Activity Diagram of FASTQ Analysis Workflow.*

3.5.4.1.3 Database Update Diagram

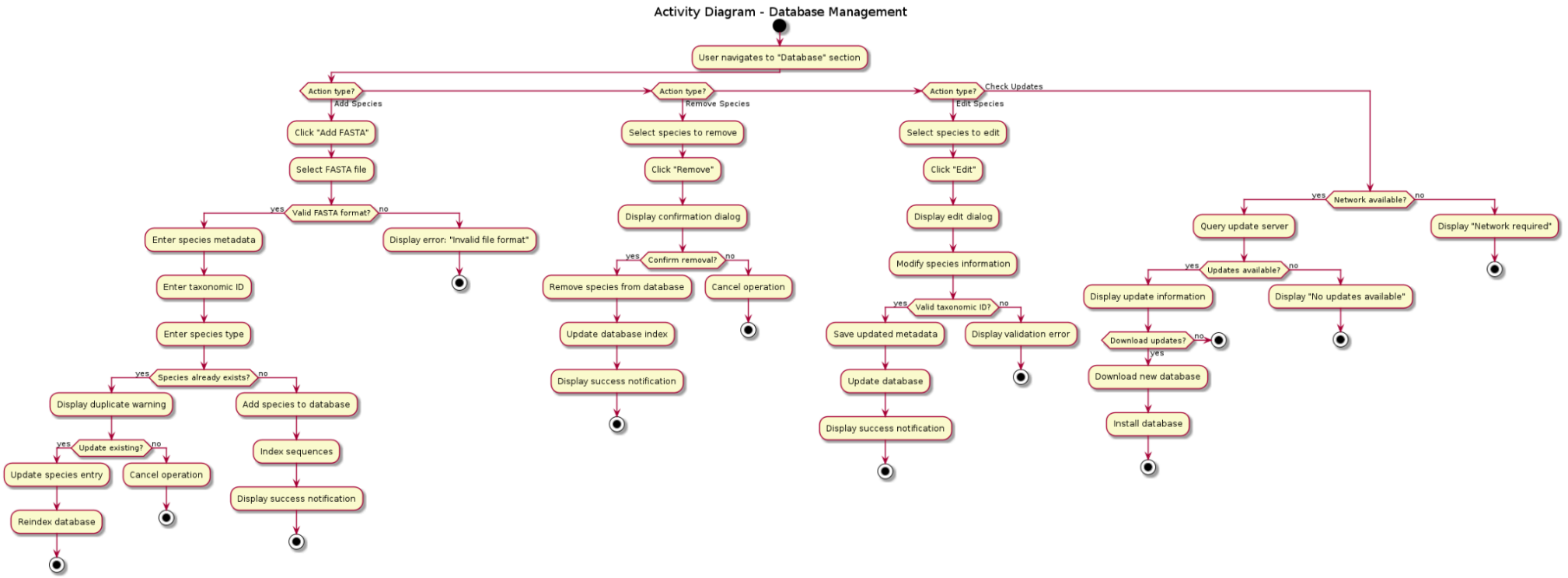

Figure 5. *Activity Diagram for Database Management.*

3.5.4.2 State Diagrams

3.5.4.2.1 Analysis State Diagram

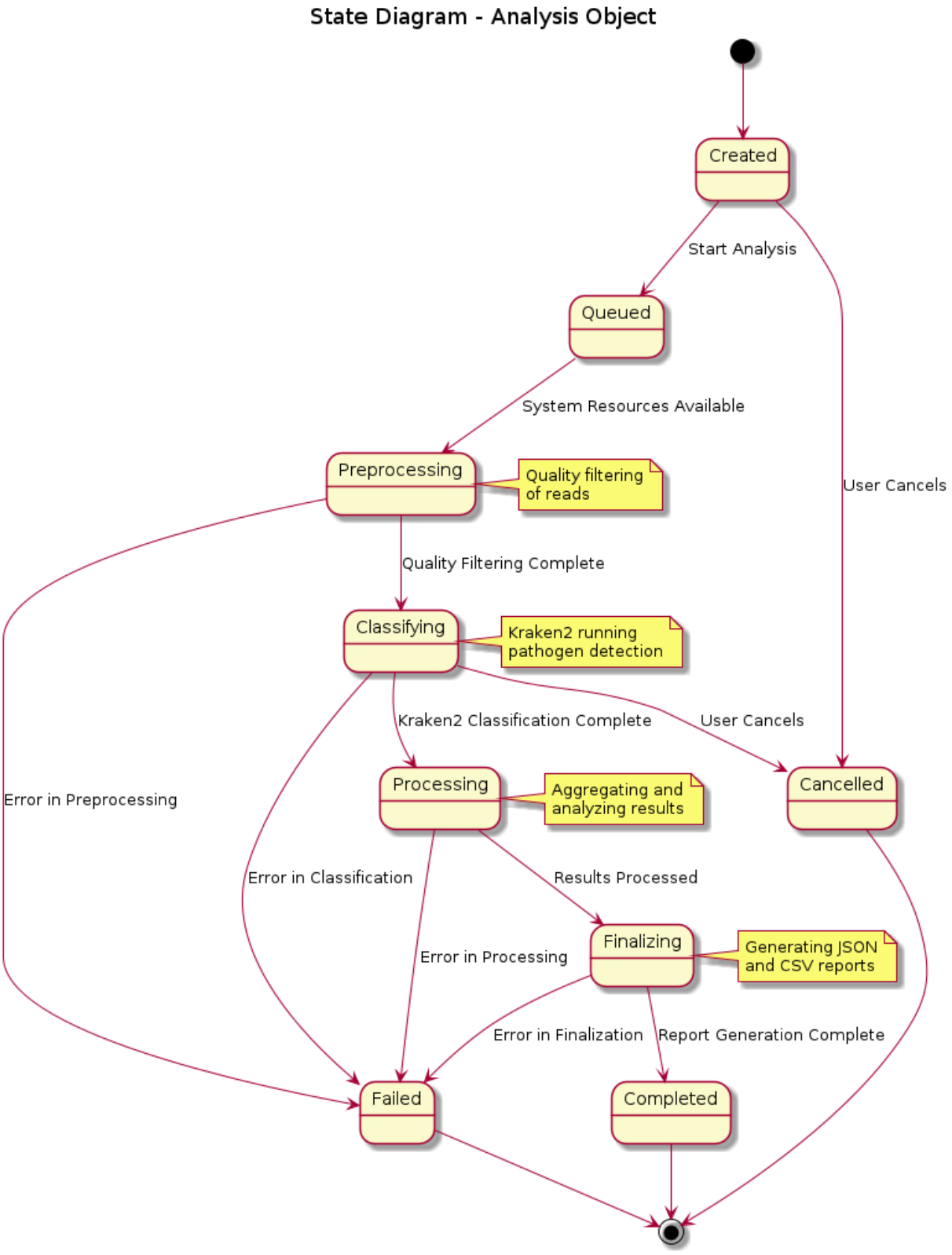

Figure 6. *Analysis Object State Diagram.*

3.5.4.2.2 Database State Diagram

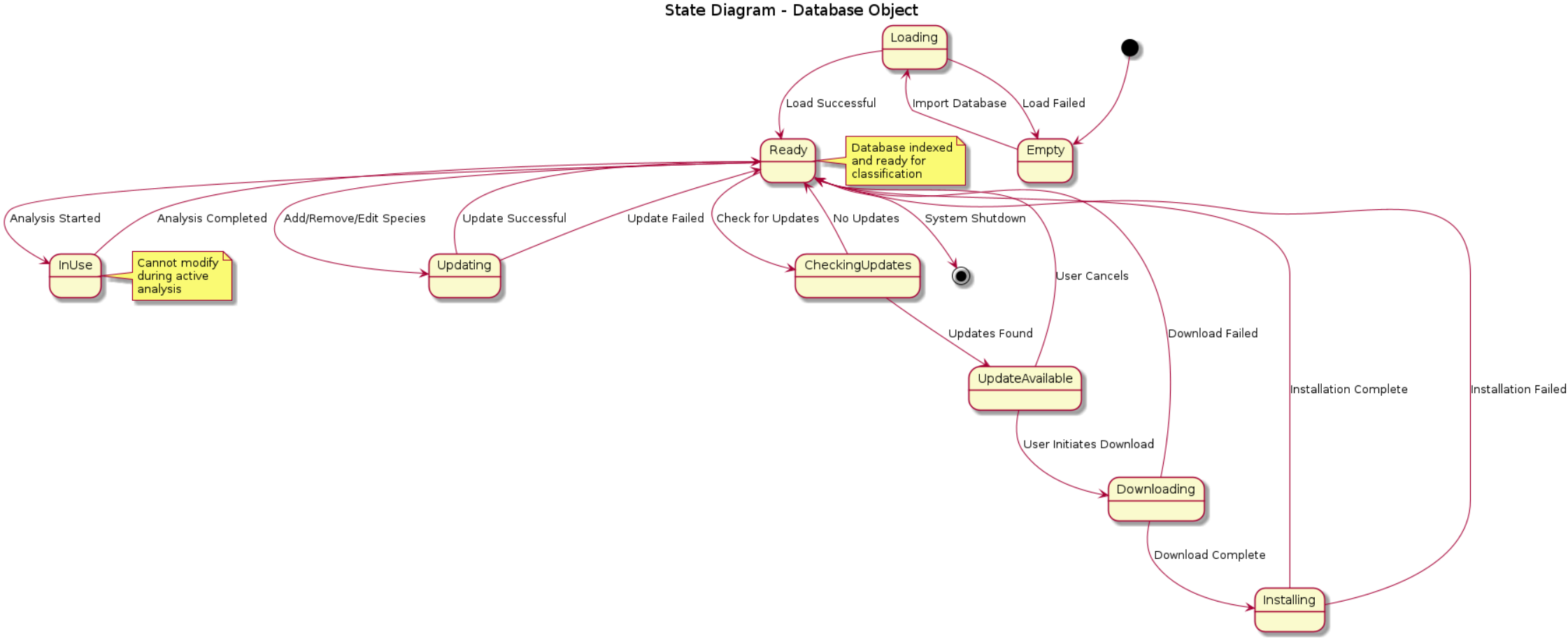

Figure 7. *Database Object State Diagram.*

3.5.4.2.3 User Session State Diagram

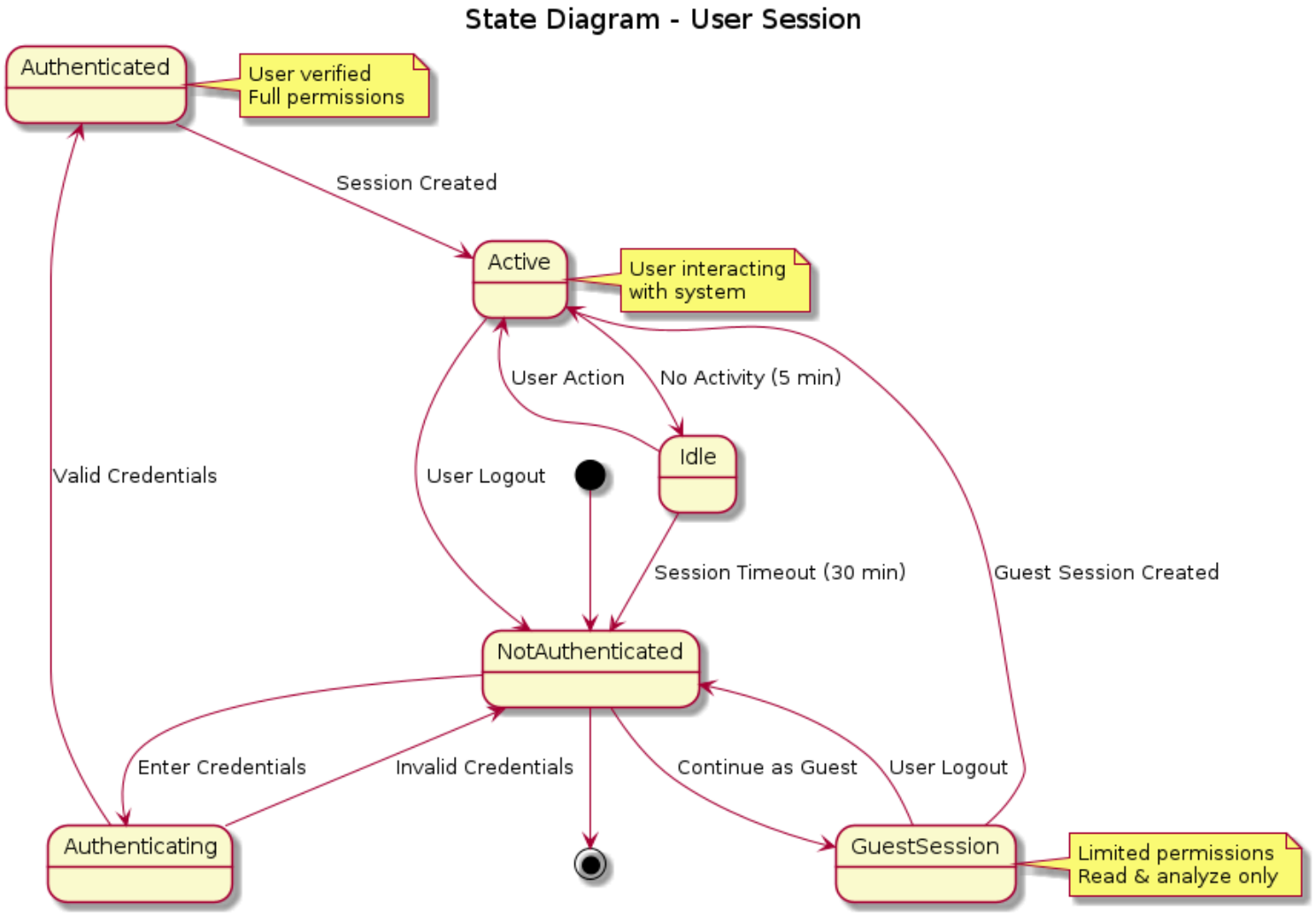

Figure 8. *User Session State Diagram.*

3.5.4.3 Sequence Diagrams

3.5.4.3.1 FastQ Analysis Sequence Diagram

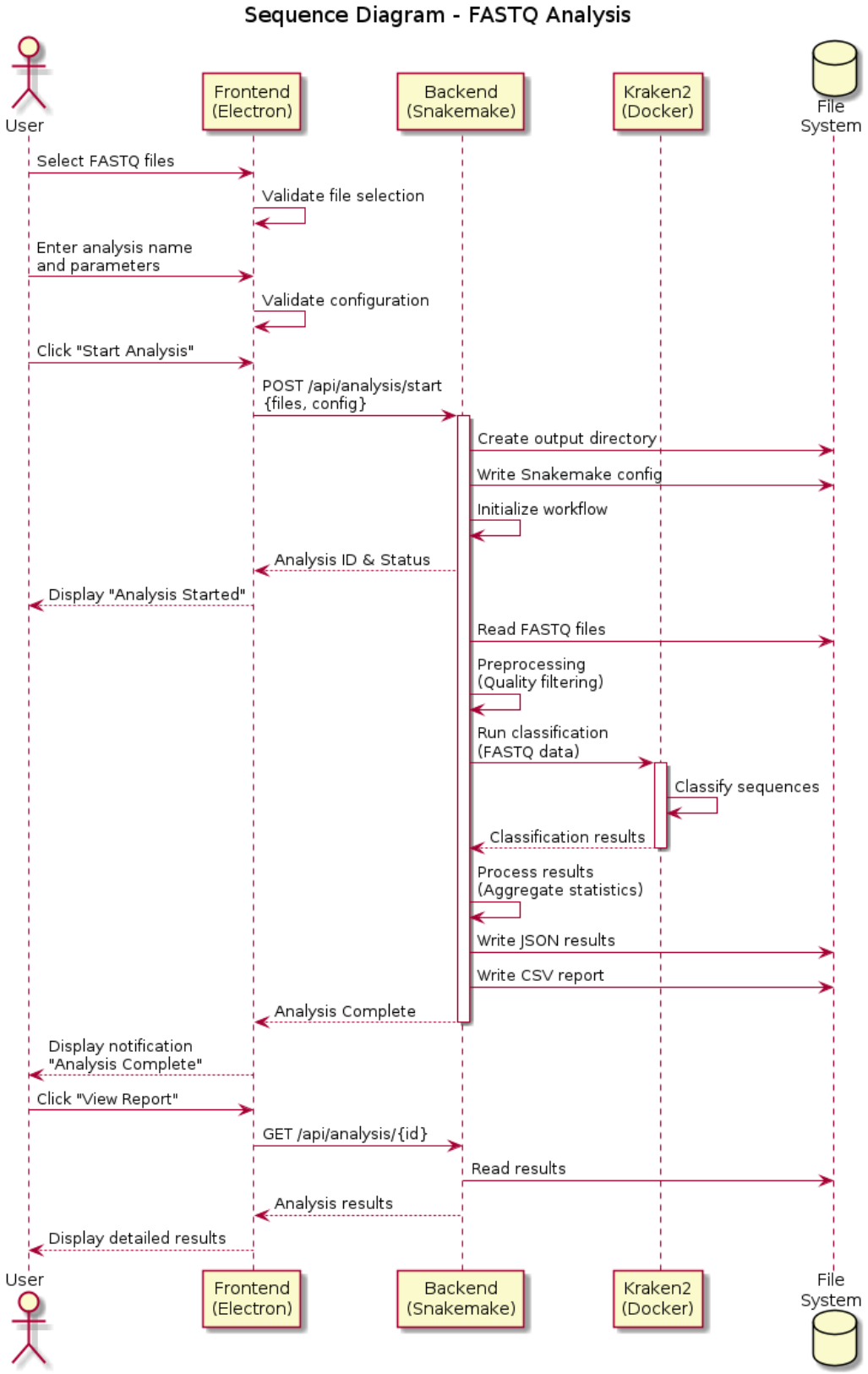

Figure 9. *FASTQ Analysis Sequence Diagram.*

3.5.4.3.2 User Authentication Sequence Diagram

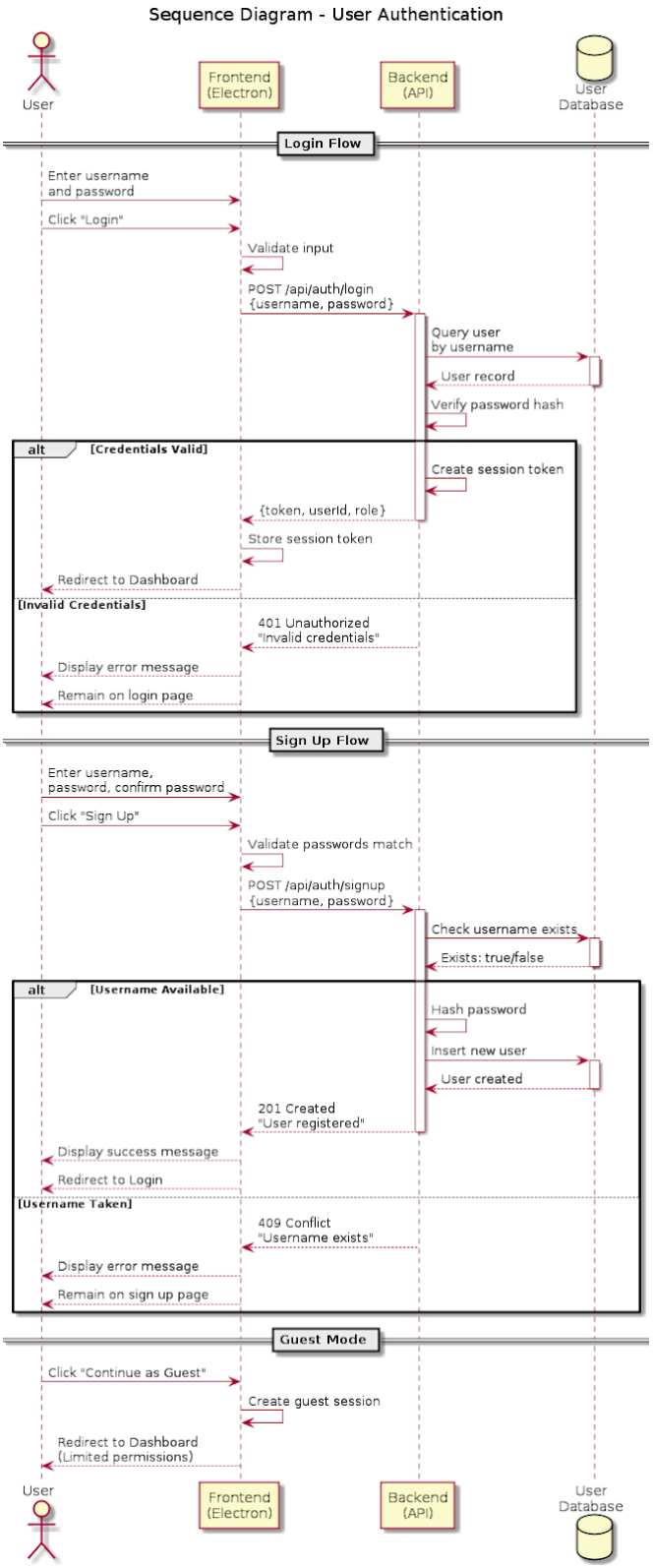

Figure 10. *User Authentication Sequence Diagram.*

3.5.4.3.3 Database Update

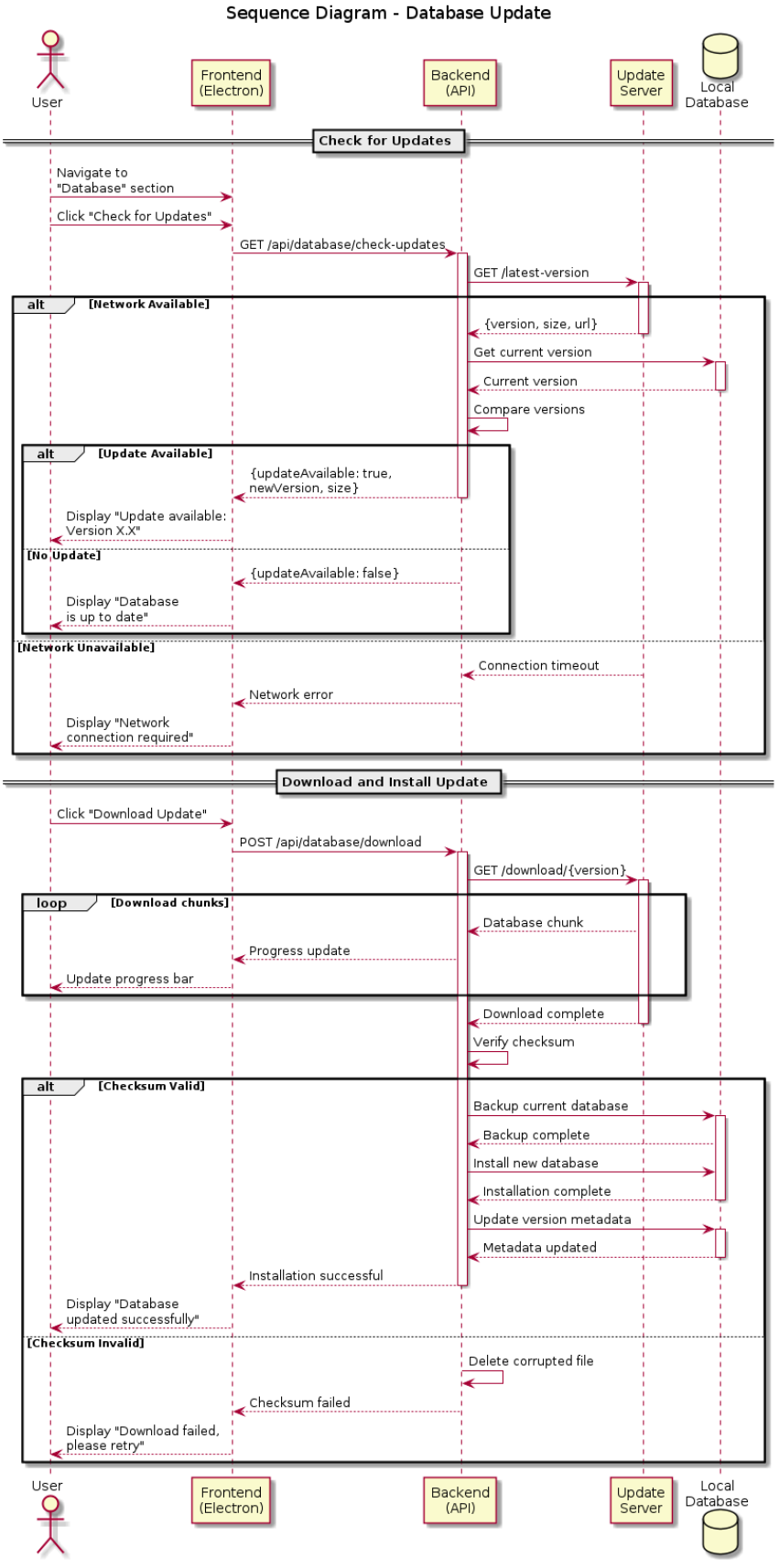

Figure 11. *Database Update Sequence Diagram.*

3.5.5 User Interface

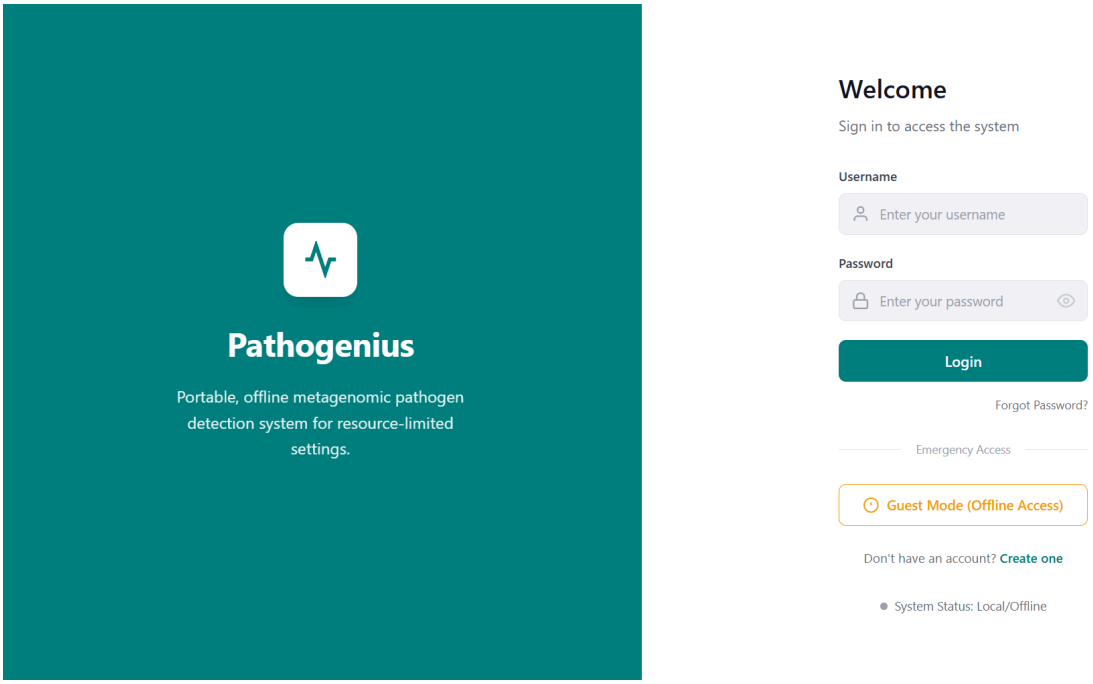

Figure 12. *Login Page of Pathogenius.*

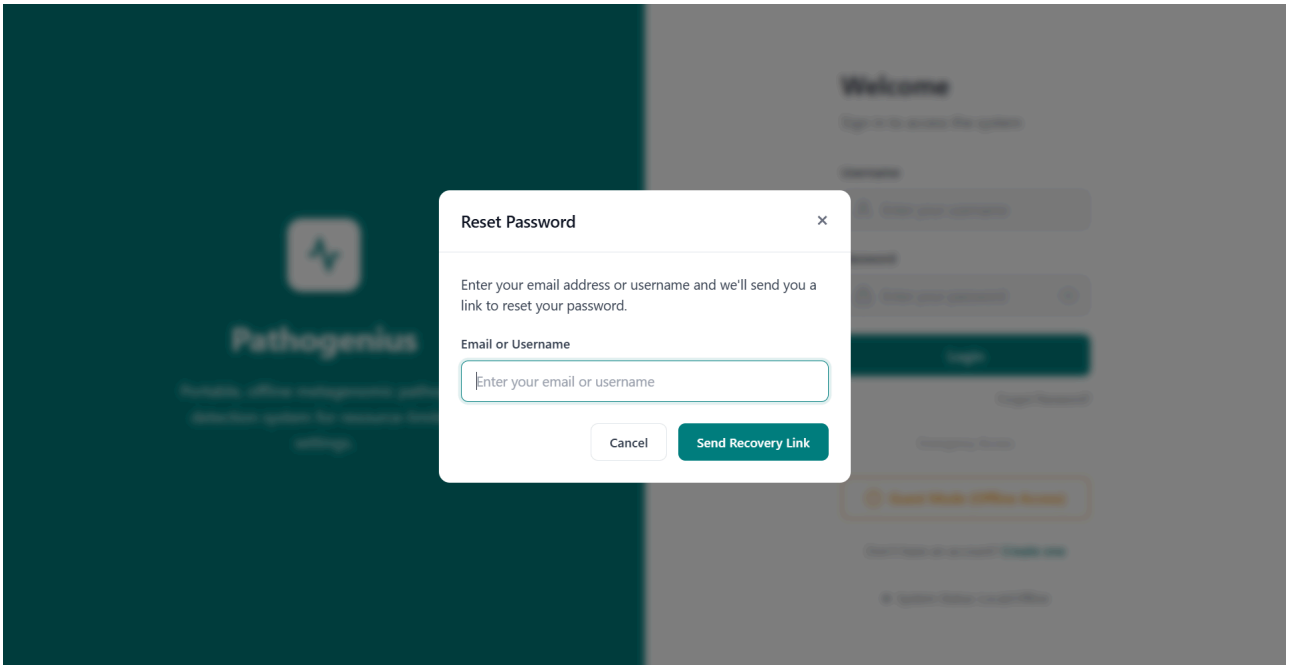

Figure 13. *Reset Password Pop-up.*

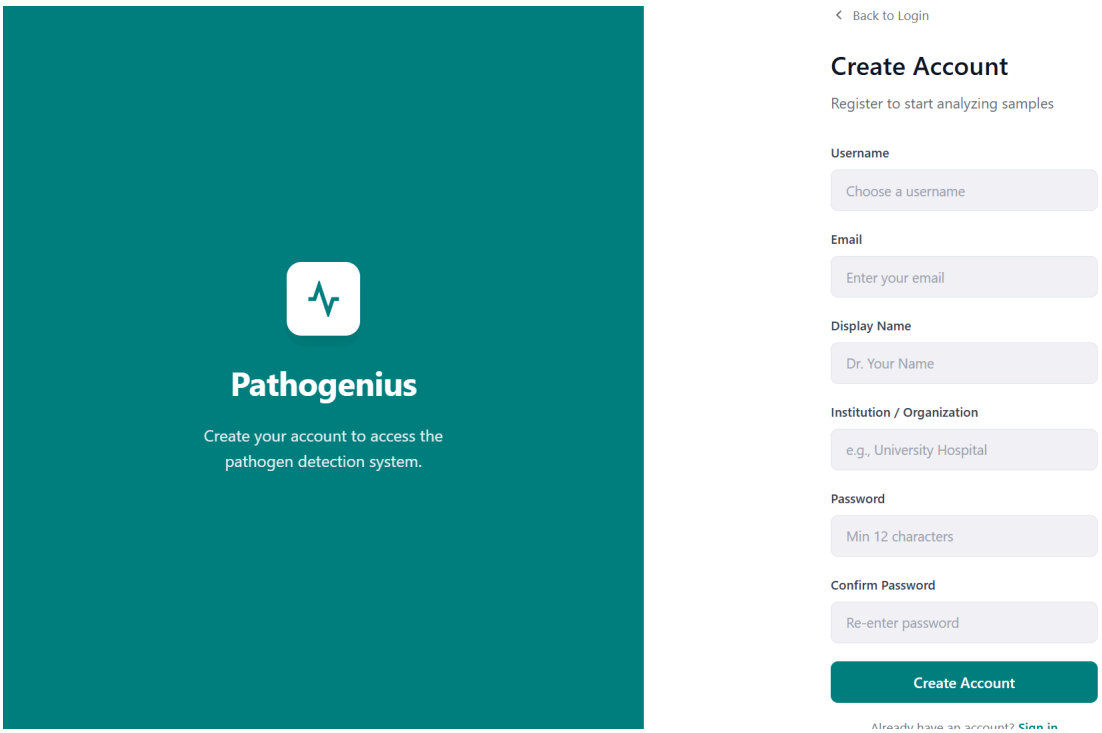

Figure 14. *Sign Up Page.*

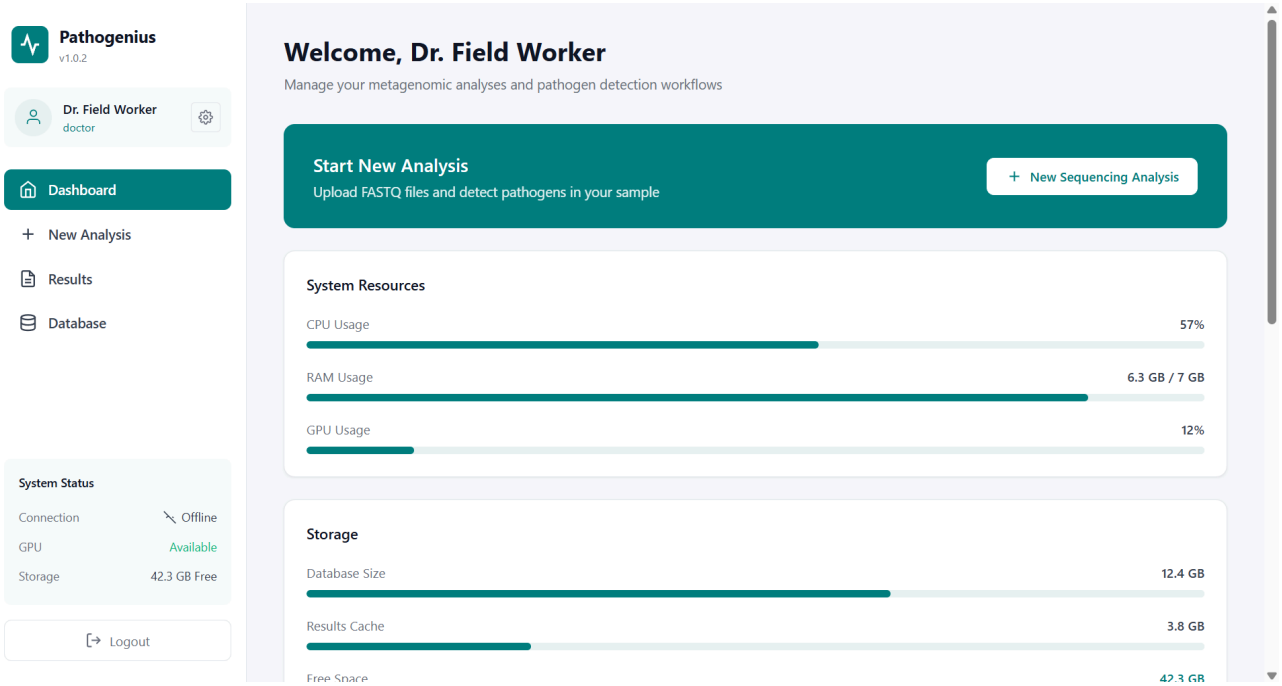

Figure 15. *Dashboard of Pathogenius.*

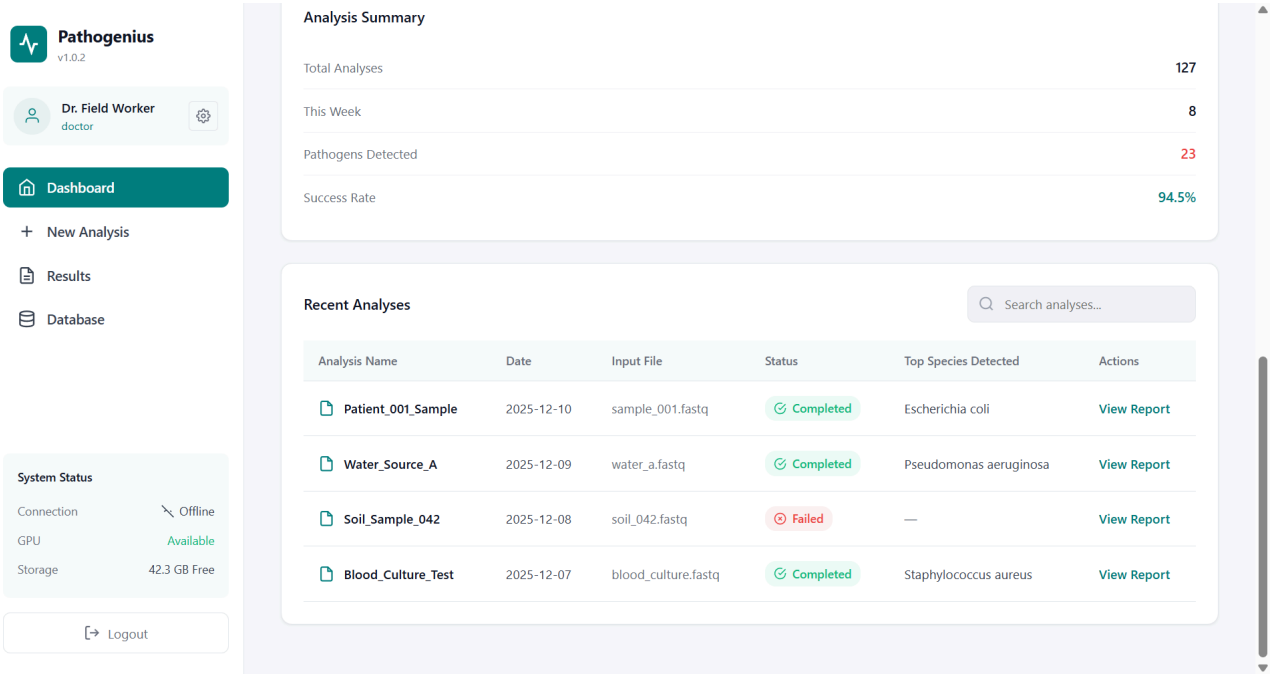

Figure 16. *Recent Analysis in Dashboard Page.*

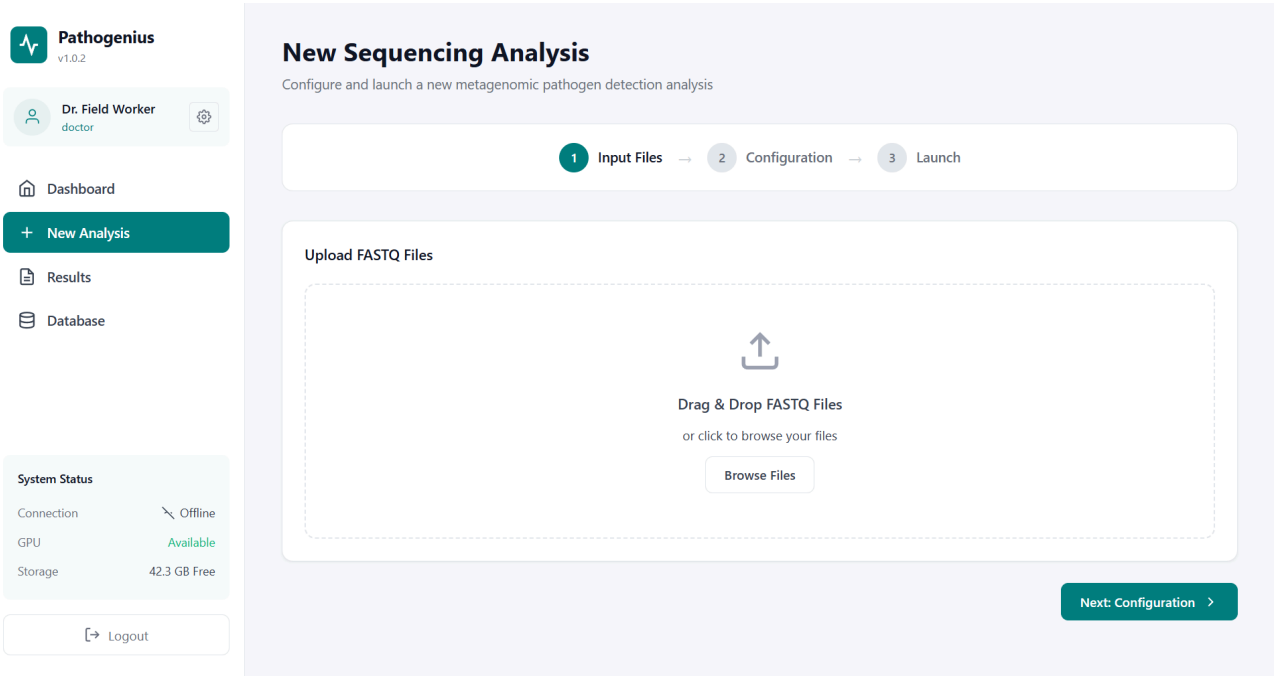

Figure 17. *New Analysis Page.*

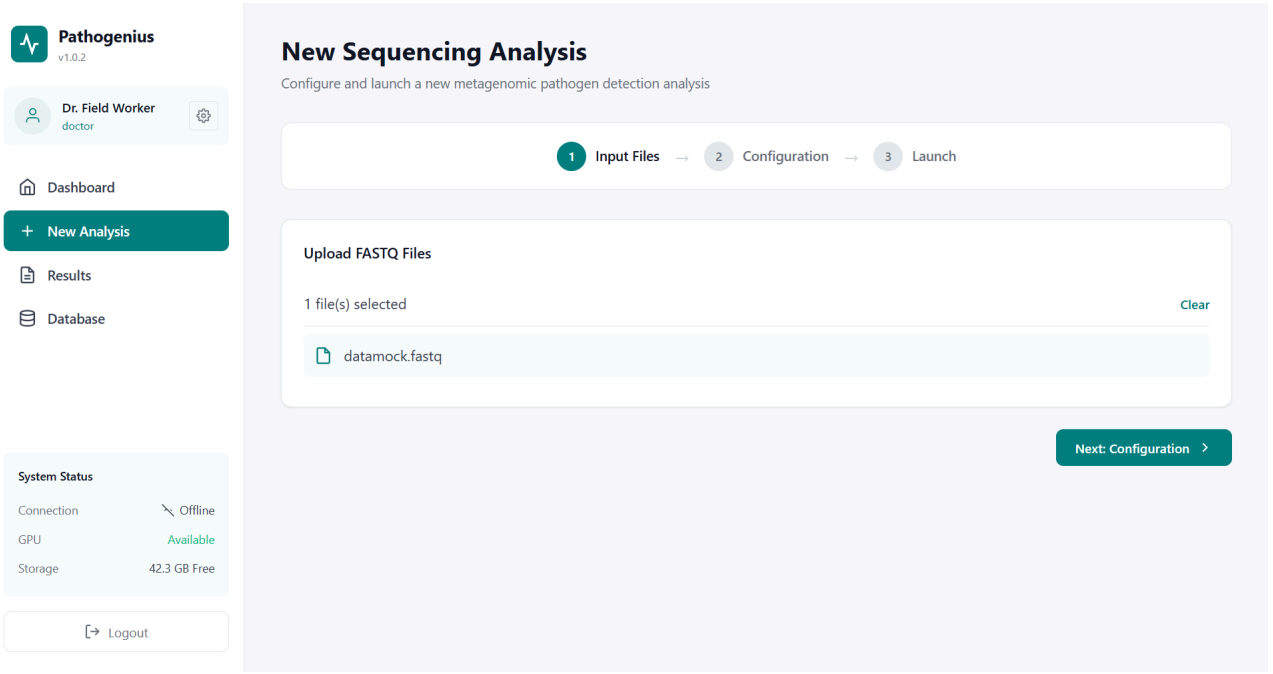

Figure 18. *New Analysis Page After File Selection.*

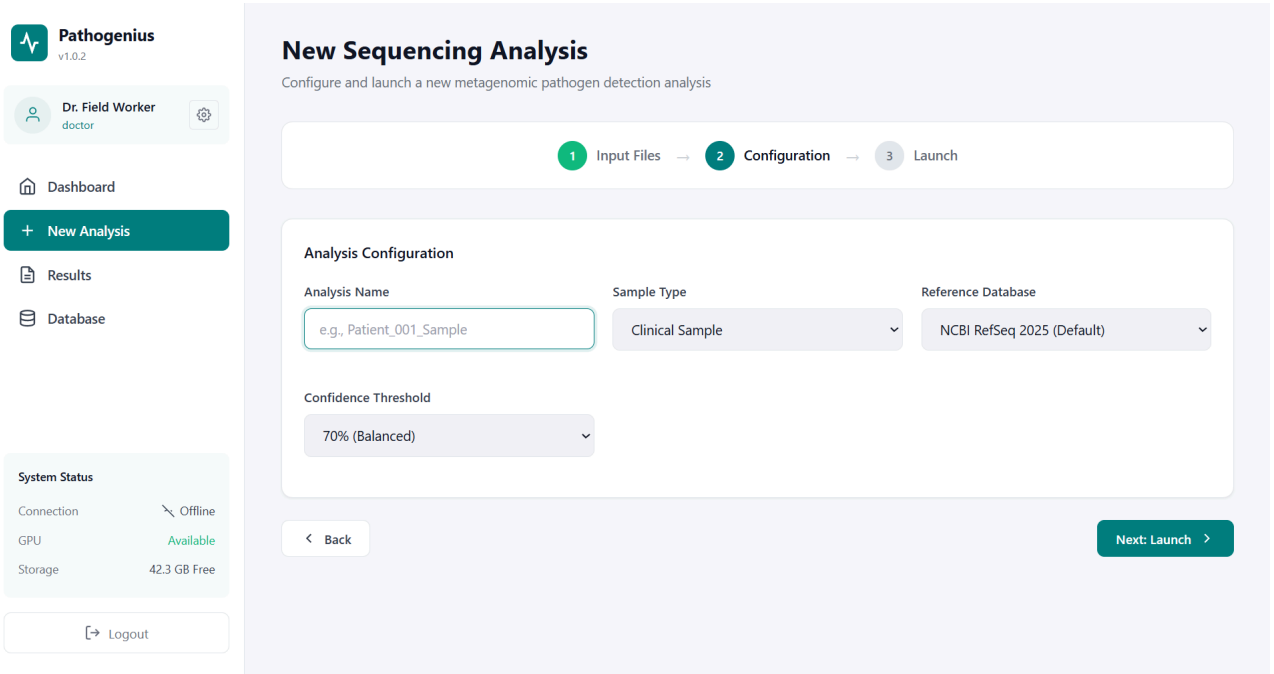

Figure 19. *New Analysis Page Second Step, Configuration.*

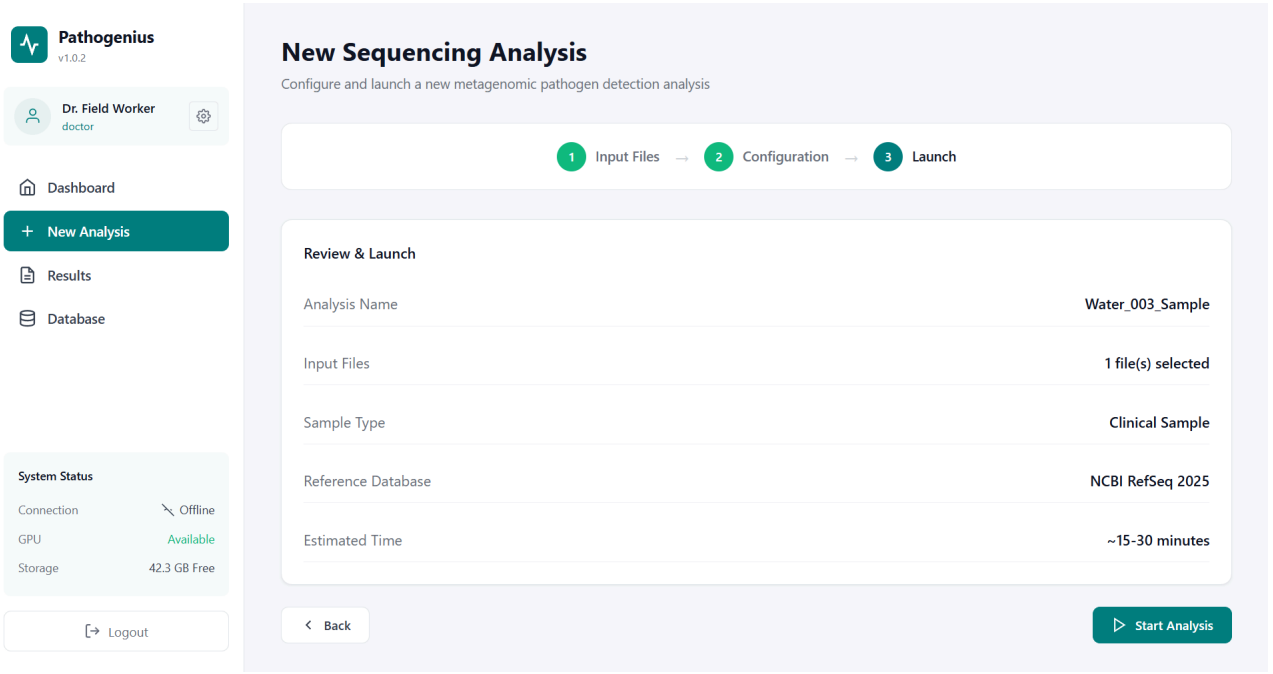

Figure 20. *New Analysis Page Last Step, Launch.*

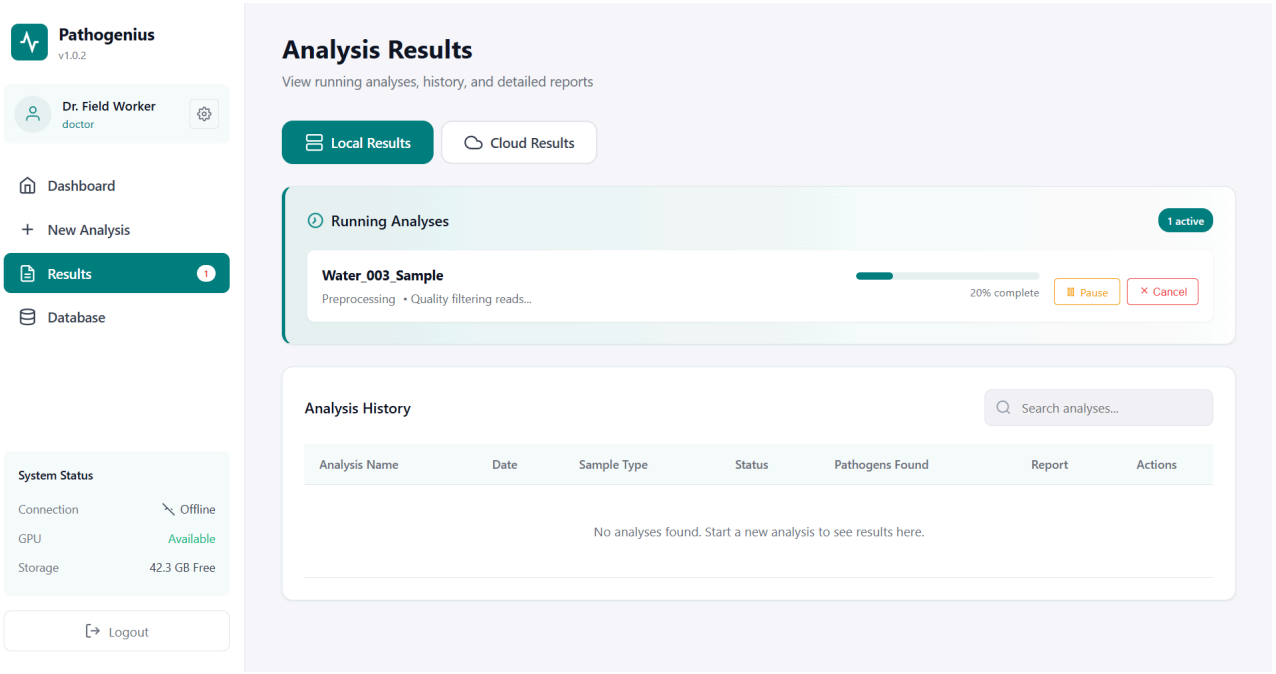

Figure 21. *Results Page with Running Analysis.*

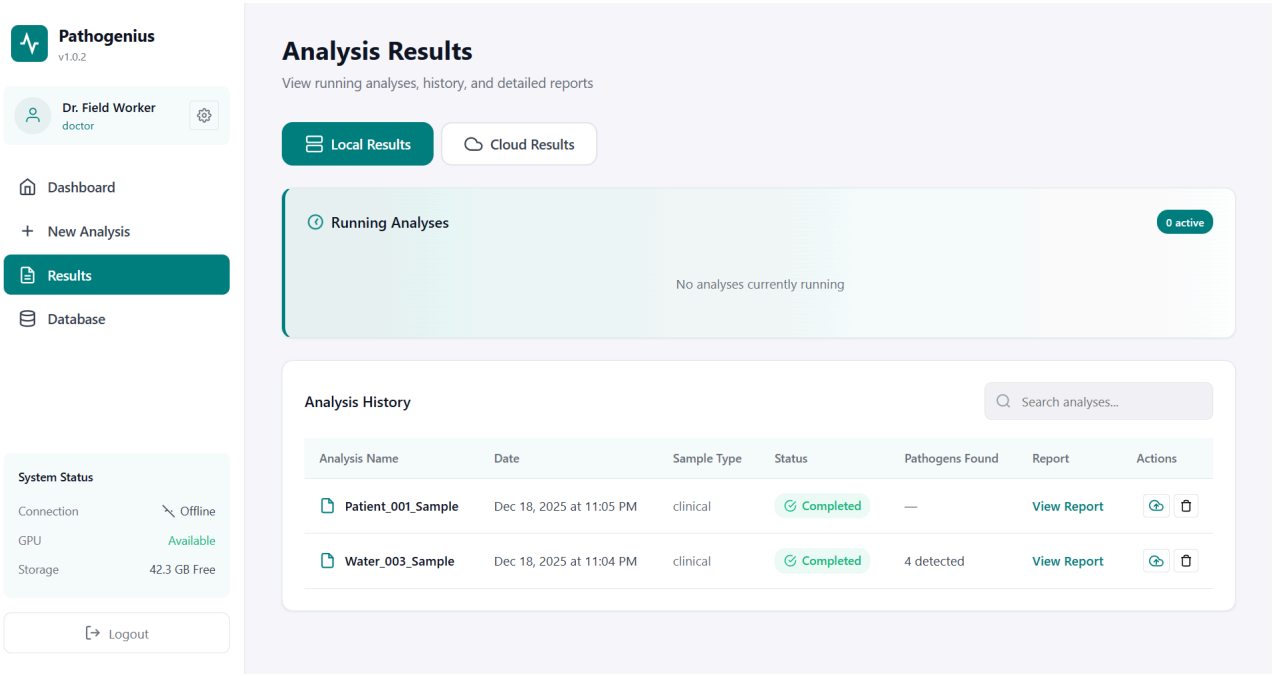

Figure 22. *Results Page with Analysis History.*

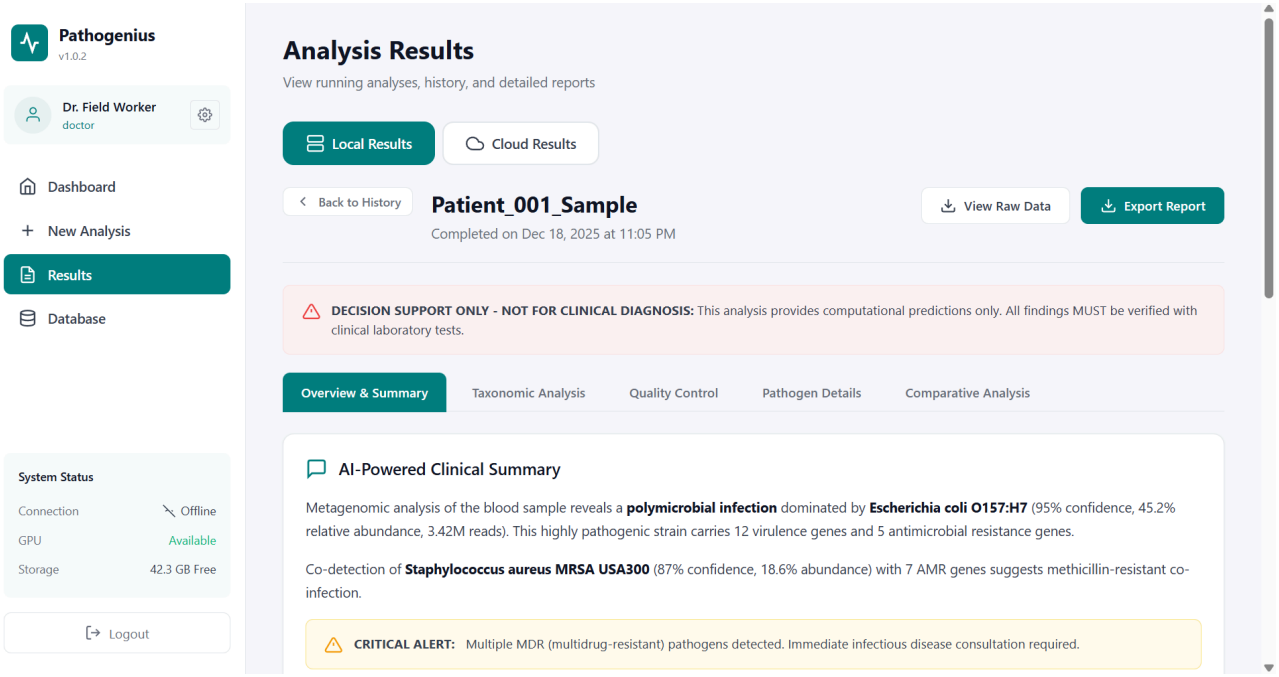

Figure 23. *Detailed Results Page, AI Summary.*

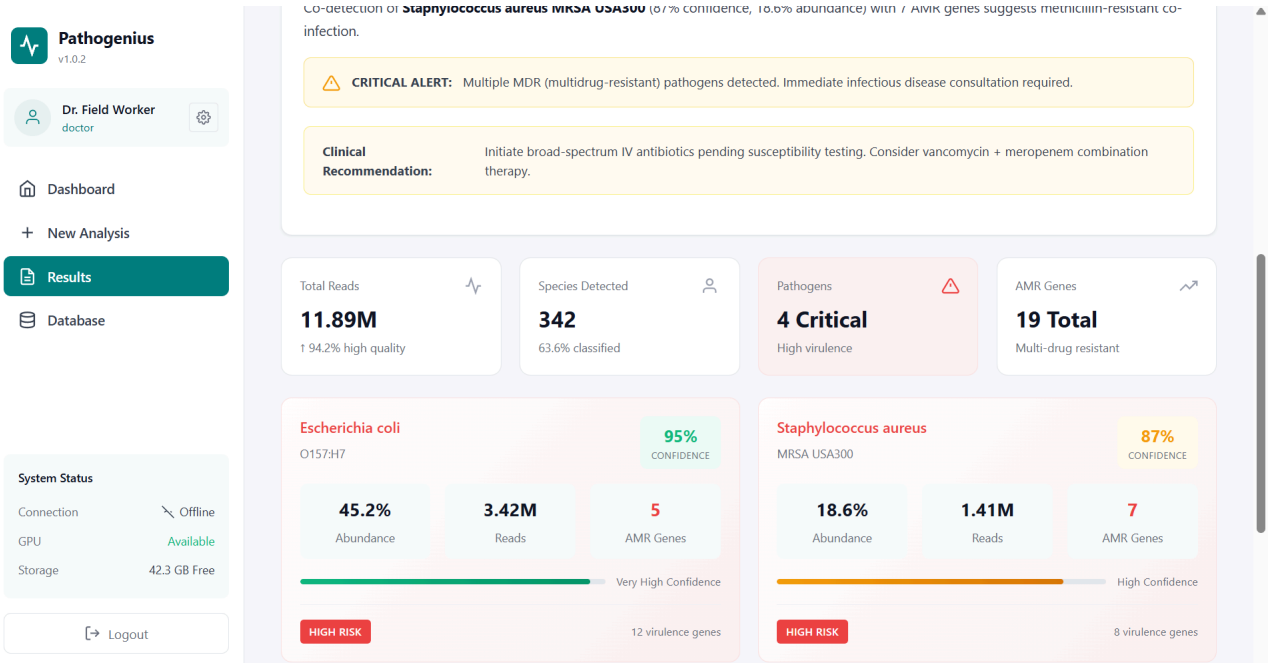

Figure 24. *Detailed Results Page.*

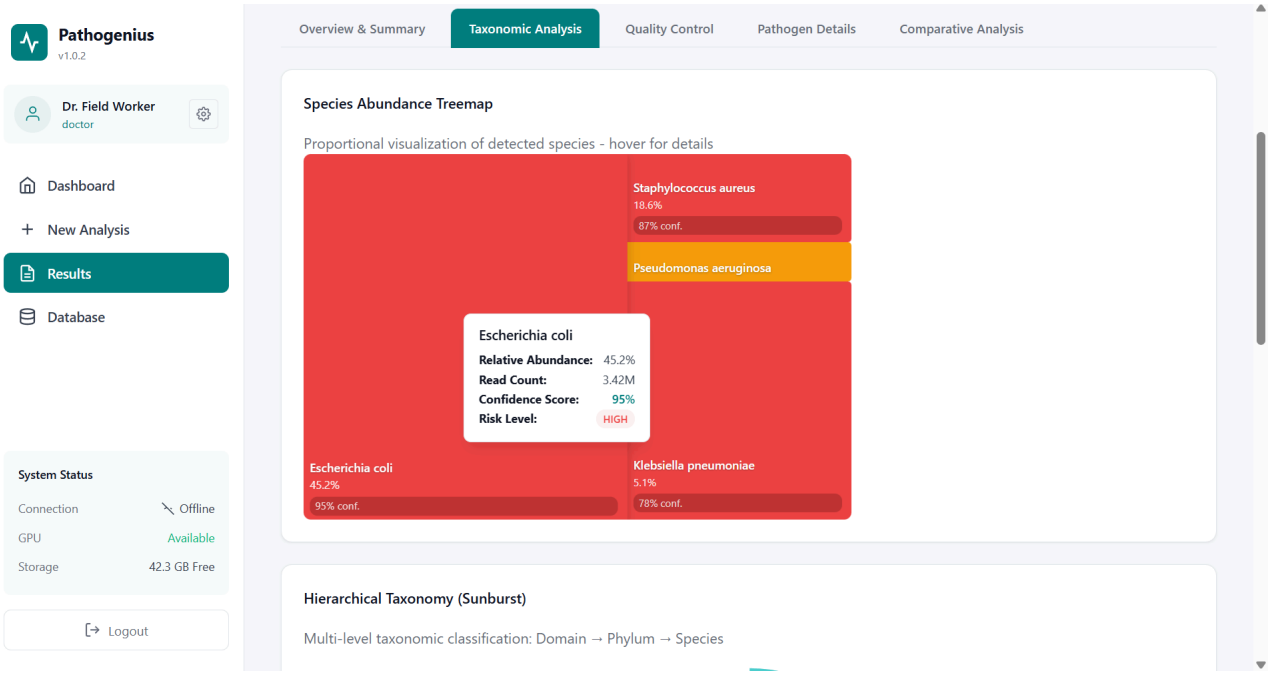

Figure 25. *Detailed Results Page, Species Abundance Treemap.*

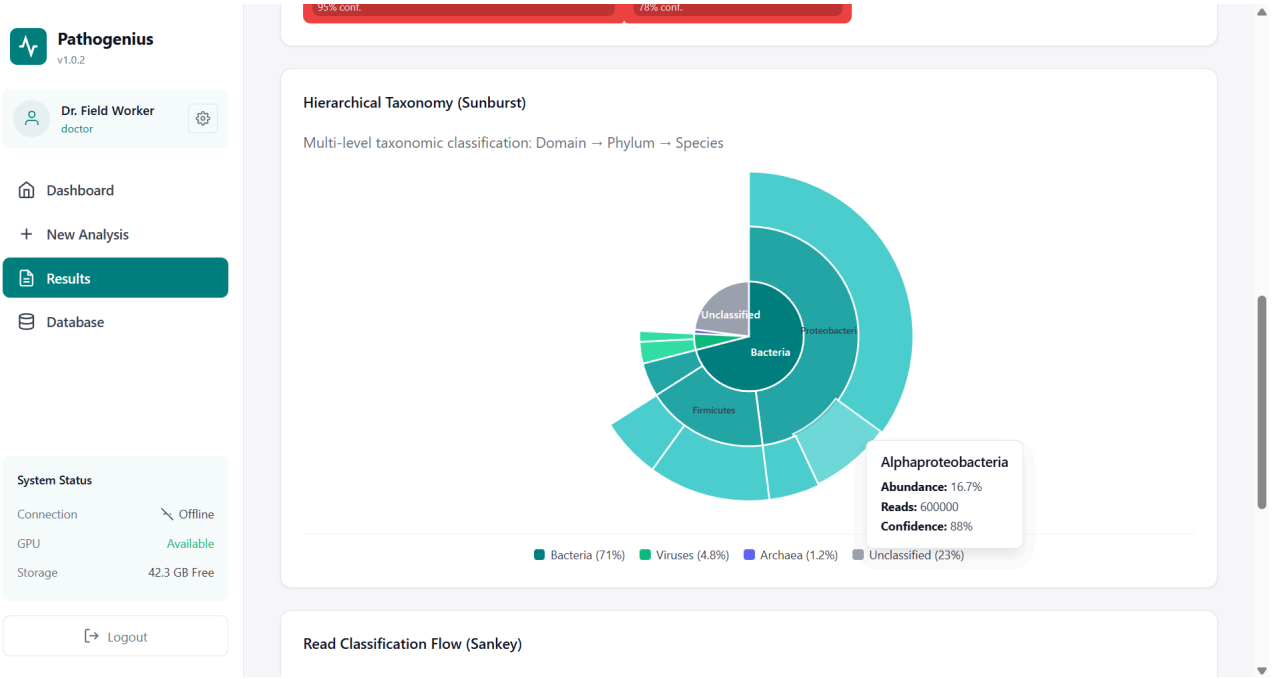

Figure 26. *Detailed Results Page, Sunburst Chart of Taxonomy.*

Figure 27. *Detailed Results Page, Classification Sankey Plot.*

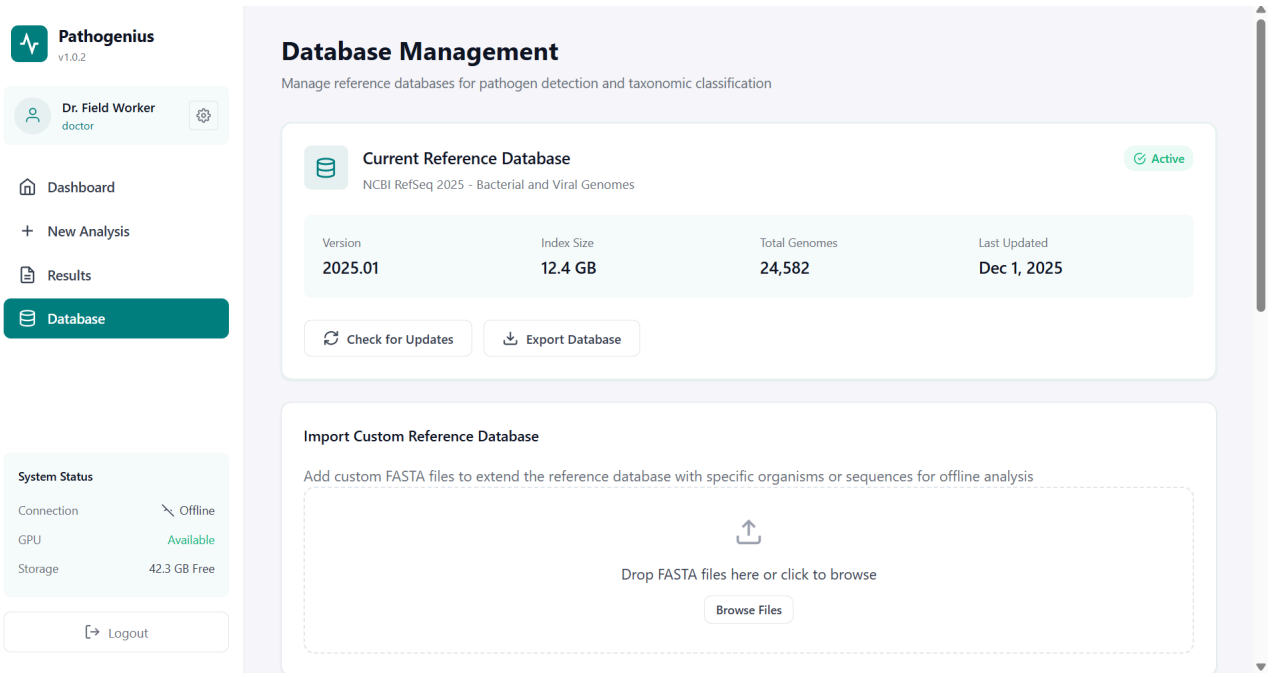

Figure 28. *Database Management Page*

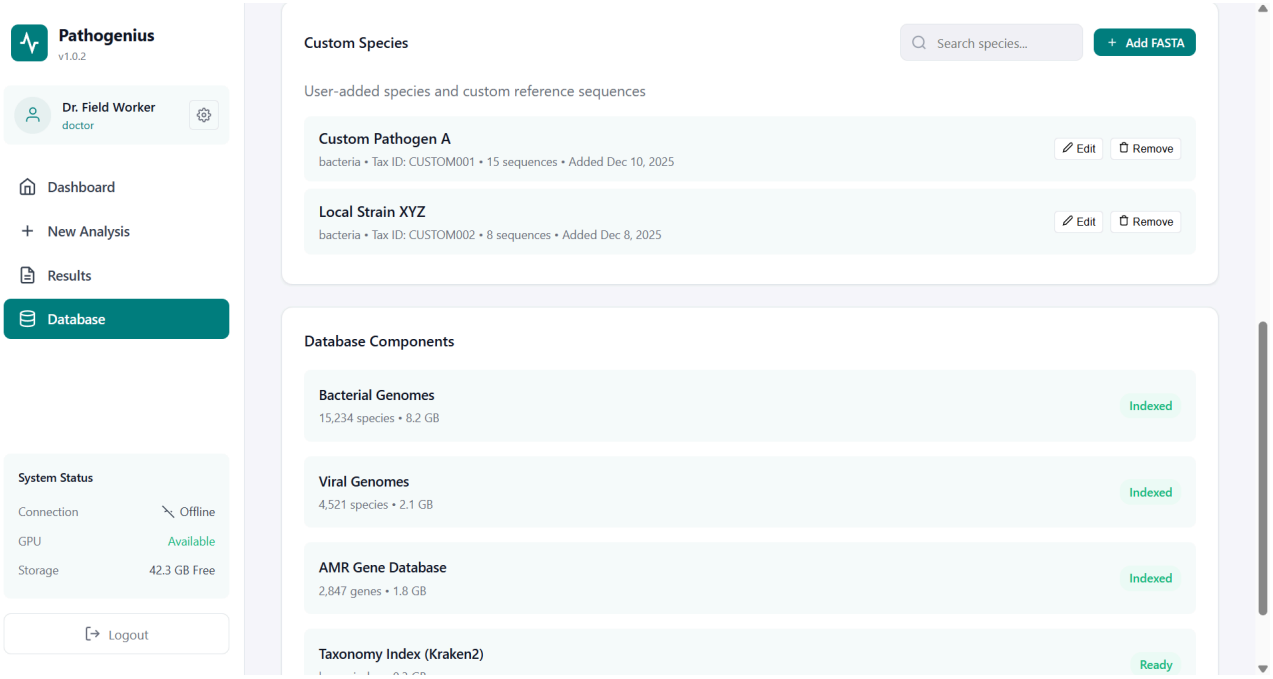

Figure 29. *Database Management Page, Continued.*

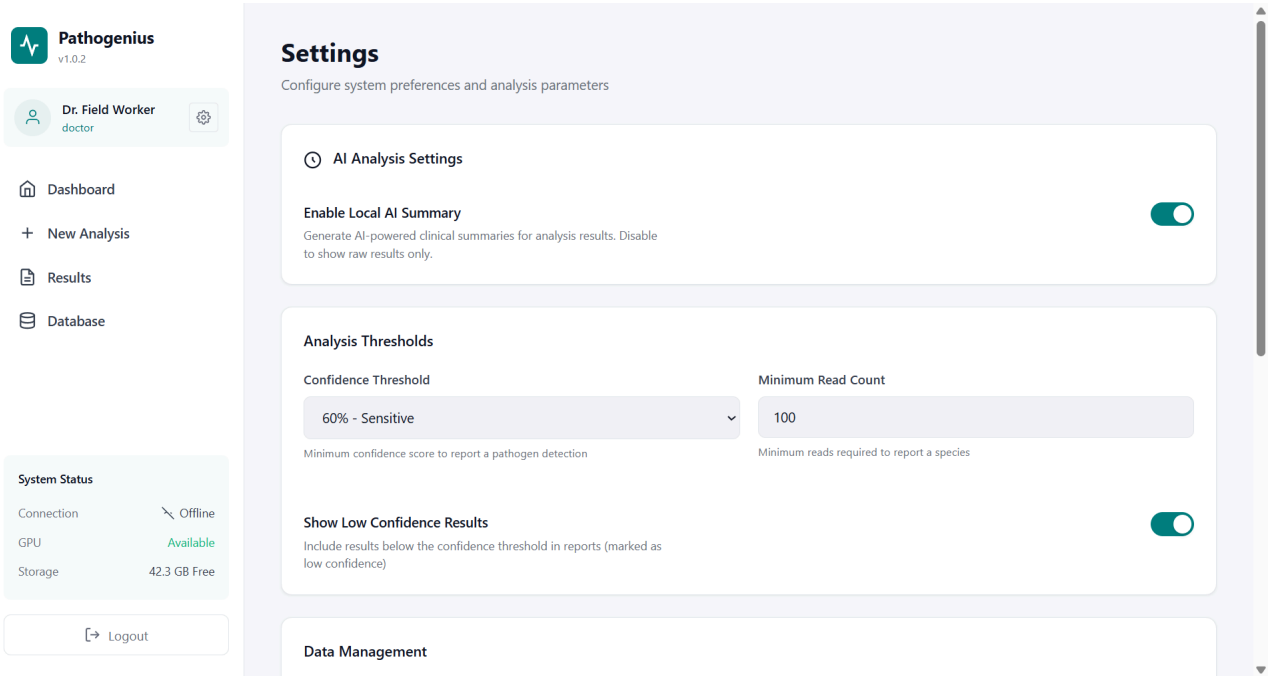

Figure 30. *Settings Page.*

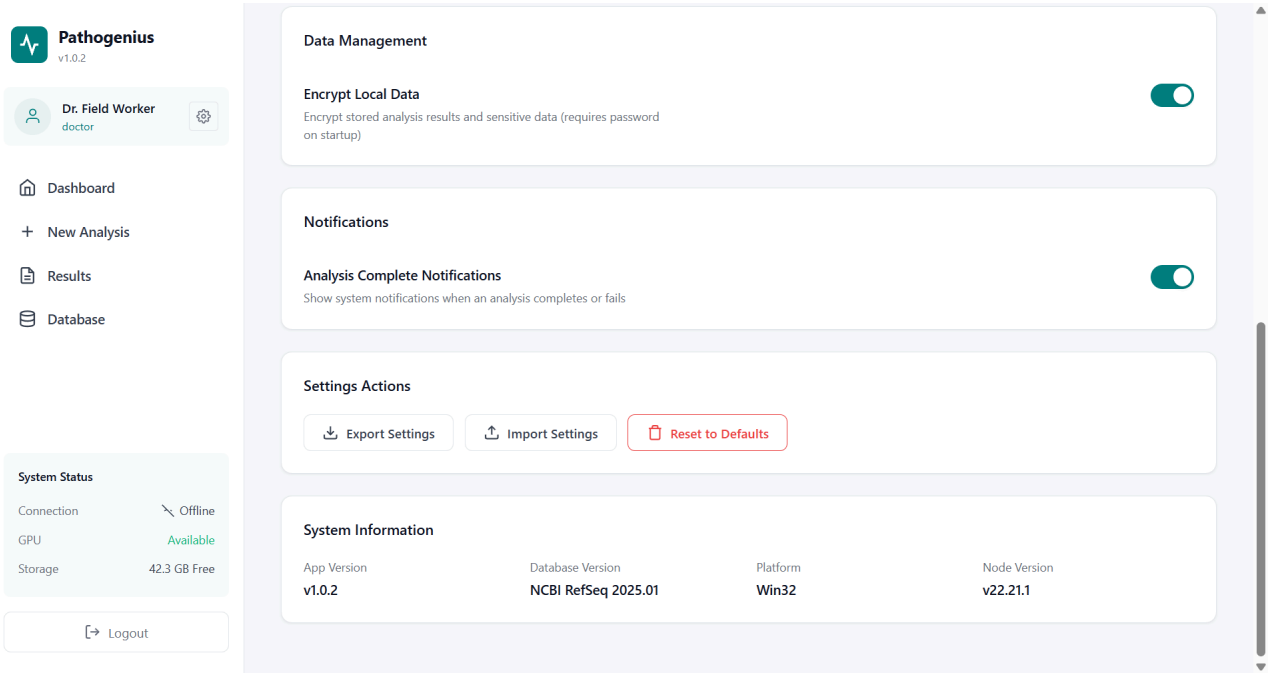

Figure 31. *Settings Page, Continued.*

4 Other Analysis Elements

4.1 Consideration of Various Factors in Engineering Design

4.1.1 Constraints

Implementation Constraints:

- Version control and collaboration are managed through Git and GitHub.
- Bioinformatics pipeline implemented using Snakemake workflow management for modularity, reproducibility, and fault tolerance.
- GUI developed with Electron.js for cross-platform compatibility and offline functionality.
- System employs phased development: initial CPU-based Kraken2 implementation with NVIDIA CUDA-compatible hardware for future GPU acceleration.
- AI assistant functionality must use locally-operated Small Language Models (SLM) or quantized LLMs for privacy and offline capability.
- Complete offline operation during the analysis phase with all dependencies, databases, and tools bundled locally.
- Accepts raw sequencing data in standard FASTQ format; development uses simulated reads from Icarus simulator.

Hardware and Software Specifications:

- Must operate on mid-range commercial laptops without requiring HPC clusters.
- Designed for low-power hardware to minimize energy consumption and maximize battery life in off-grid scenarios.

Economic Constraints:

- Software must be completely free of charge.
- No recurring per-use costs to ensure accessibility in resource-limited settings.

Ethical Constraints:

- Transient local processing of clinical samples containing human DNA.
- No exposure of sensitive human genomic information.
- Functions as a decision support system, not a diagnostic device.
- Results presented as probabilistic evidence with appropriate uncertainty markers.

Environmental Constraints:

- Minimized environmental footprint through local analysis on low-power hardware.
- Performance-per-watt optimization for battery preservation.
- Avoids energy-intensive HPC clusters and continuous cloud connectivity.

Usability Constraints:

- Designed for field personnel without bioinformatics expertise.
- Command-line operations abstracted through GUI.
- No terminal or script interaction required.

Legal and Regulatory Considerations:

- Compliance with data privacy requirements for human genomic information.
- Operation within medical device regulations as decision support (not diagnostic) tool.

Maintainability and Extensibility:

- Modular architecture enabling future GPU acceleration migration.
- Workflow management supports reproducibility and fault tolerance.

Interoperability:

- Standard FASTQ format compatibility.
- Integration with Oxford Nanopore sequencing devices.

Global, Cultural, Social, Environmental, and Economic Factors:

The Pathogenius system addresses critical healthcare disparities in resource-limited global settings where traditional laboratory infrastructure is unavailable. Culturally, the system respects data sovereignty by maintaining complete offline operation, which is particularly important in regions with concerns about external data control. Socially, it democratizes access to advanced pathogen detection by removing cost barriers and simplifying technical complexity, enabling field personnel without specialized training to perform sophisticated analyses. Environmentally, the platform's low-power operation and elimination of cloud dependency significantly reduce carbon footprint compared to HPC-based alternatives, making it sustainable for deployment in off-grid locations. Economically, the zero-cost model and modest hardware requirements remove financial barriers that typically exclude resource-constrained healthcare facilities from accessing cutting-edge diagnostic technology.

Table 1: Factors that can affect analysis and design.

Factor	Effect Level	Effect Description
Public Health	High	Enables rapid pathogen identification in outbreak scenarios and resource-limited settings; improves disease surveillance capabilities; reduces time-to-diagnosis for infectious diseases.
Public Safety	High	Mitigates risks of disease spread through faster pathogen detection; however, misinterpretation of probabilistic results could lead to inappropriate treatment if confidence indicators are ignored.

Public Welfare	High	Increases healthcare accessibility in remote areas; removes cost barriers to advanced diagnostics; empowers local healthcare workers with sophisticated tools.
Global Factors	High	Addresses global health inequities via portable diagnostics independent of central labs; supports pandemic preparedness; enables data sovereignty through offline operation.
Cultural Factors	Medium	Respects data privacy concerns; offline operation addresses trust issues regarding cloud storage; requires consideration of local healthcare practices and workflows.
Social Factors	High	Democratizes access to advanced molecular diagnostics; reduces dependency on bioinformatics expertise; empowers non-expert users through intuitive interface design.
Environmental Factors	Medium-High	Minimizes carbon footprint by eliminating cloud infrastructure; optimizes energy efficiency for battery-powered/off-grid operation; reduces impact vs. energy-intensive HPC alternatives.

4.1.2 Standarts

4.1.2.1 IEEE 830

IEEE 830 provides the systematic framework for defining and documenting functional and non-functional requirements. The standard ensures clarity, completeness, and unambiguous specification, preventing misinterpretation among team members and supporting accurate validation and testing. This contributes to producing well-defined, traceable, and maintainable specifications for Pathogenius.

4.1.2.2 ISO/IEC 25010

This standard provides a comprehensive quality model encompassing performance efficiency, reliability, usability, security, maintainability, and compatibility. ISO/IEC 25010 [6] guides the assessment of whether Pathogenius meets user expectations and operational requirements, particularly ensuring reliable function in resource-constrained settings and producing trustworthy results. The project aligns software quality goals with these internationally recognized guidelines.

4.1.2.3 UML 2.5.1 - Unified Modeling Language

UML 2.5.1 [7] is employed for describing system structure, behavior, and component interactions through standardized graphical representations. Use-case diagrams illustrate user interactions, while activity diagrams describe workflow execution through tools like Kraken2. Adherence to UML ensures systematic documentation of architectural decisions that can be easily understood, reviewed, and maintained, contributing to clearer design discussions and more accurate implementation.

4.1.2.4 ISO 9241-210

This standard emphasizes designing software by prioritizing user needs, capabilities, and limitations, ensuring systems are usable and useful in real-world contexts. ISO 9241-210 [8] is particularly relevant for Pathogenius as it targets users without bioinformatics expertise. The standard encourages development of a clear, intuitive, error-reducing interface that supports correct interpretation of species-level outputs without overwhelming users with technical complexity, incorporating iterative evaluation and refinement.

4.2 Risks and Alternatives

4.2.1 Insufficient Classification Accuracy Due to Limited FASTQ Size

Description: The classification accuracy of the system may be reduced when the input FASTQ file contains an insufficient number of reads or limited sequencing depth. In such cases, rare or low-abundance pathogens may not be detected reliably, and accurate species-level identification may require larger sequencing datasets, potentially exceeding 10 GB in size.

Contributing Factors:

- Low sequencing coverage or short sequencing runs, reducing the probability of sampling rare organisms
- Presence of low-abundance pathogens within complex samples
- Hardware limitations restrict processing of very large FASTQ files in a single execution

Mitigation:

- Integrate a FASTQ splitting utility such as SeqKit to divide large FASTQ files into smaller, manageable chunks that can be processed sequentially or incrementally.

4.2.2 Limited Interpretive Quality of Local Language Model

Description: Pathogenius integrates a local language model to generate readable summaries and explanations from structured analysis results. Due to hardware constraints and the requirement for local execution, the selected model may be relatively low-parameter, potentially limiting the depth, nuance, or contextual richness quality of generated answers.

Contributing Factors:

- Absence of large-scale pretrained models that typically require cloud-based inference
- Variability in analysis complexity that may exceed the expressive capacity of smaller models

Mitigations:

- Implement a deterministic, rule-based text generation layer that converts structured analysis outputs into clear, generalized explanatory text.
- When internet connectivity is available, optionally allow the selection of higher-capacity language models while keeping inference local.

4.2.3 GPU Memory Limitations

Description: When GPU acceleration is implemented, large FASTQ inputs may exceed available GPU memory (VRAM), even if total file size does not exceed typical storage limits such as 10 GB. This may cause runtime failures or require fallback to CPU execution, reducing expected performance gains.

Contributing Factors:

- Limited VRAM capacity on consumer-grade GPUs
- High read counts or long-read lengths increasing memory footprint during classification

Mitigations:

- If supported, enable batching or chunked processing options provided by GPU-accelerated tools.
- Apply FASTQ splitting using utilities such as SeqKit to ensure per-batch memory usage remains within VRAM limits.

4.2.4 Incomplete or Outdated Reference Databases

Description: Pathogen detection accuracy is dependent on the completeness and correctness of the local reference database. Missing, outdated, or poorly annotated genomes may result in false negatives or ambiguous classifications.

Contributing Factors:

- Limited availability of high-quality reference genomes for emerging or rare pathogens
- User-curated databases that unintentionally omit relevant taxa

Mitigations:

- Maintain versioned reference databases with clear provenance data.
- Clearly report database version and coverage information in analysis outputs.

4.2.5 False Positives Due to Shared k-mers and Taxonomic Ambiguity

Description: k-mer based classification methods such as Kraken2 may assign reads to incorrect or overly specific taxa when closely related pathogens share a large proportion of genomic sequences. This can lead to false positives or inflated confidence at the species level.

Contributing Factors:

- Closely related species with highly similar genomic content
- Short or noisy reads increasing classification ambiguity
- Overly permissive classification thresholds

Mitigations:

- Apply confidence thresholds and minimum read-count filters during the output post-processing.
- Clearly label low-confidence classifications in the interface and reports.

Table 2: Risks

Risk	Likelihood	Effect on the project	B Plan Summary
Insufficient classification accuracy due to limited FASTQ size	Medium	Low-abundance or rare pathogens may not be detected reliably, reducing confidence in results	Split FASTQ files using SeqKit
Limited interpretive quality of local language model	High	Generated explanations may be overly simplistic or lack contextual depth	Rule-based text generation or enable optional higher-capacity local models when available
GPU memory limitations during accelerated processing	High	GPU acceleration may fail or fall back to CPU, increasing analysis time	Enable GPU batching options or split FASTQ files using SeqKit to fit VRAM constraints
Incomplete or outdated reference databases	Medium	Relevant pathogens may not be identified or may be reported with reduced taxonomic resolution	Maintain versioned reference databases with clear provenance

False positives due to shared k-mers and taxonomic ambiguity	Medium	Incorrect or overly specific species assignments may be reported, reducing the reliability of classification outputs	Apply confidence thresholds, minimum read-count filters, and taxonomic rank fallback strategies

4.3 Project Plan

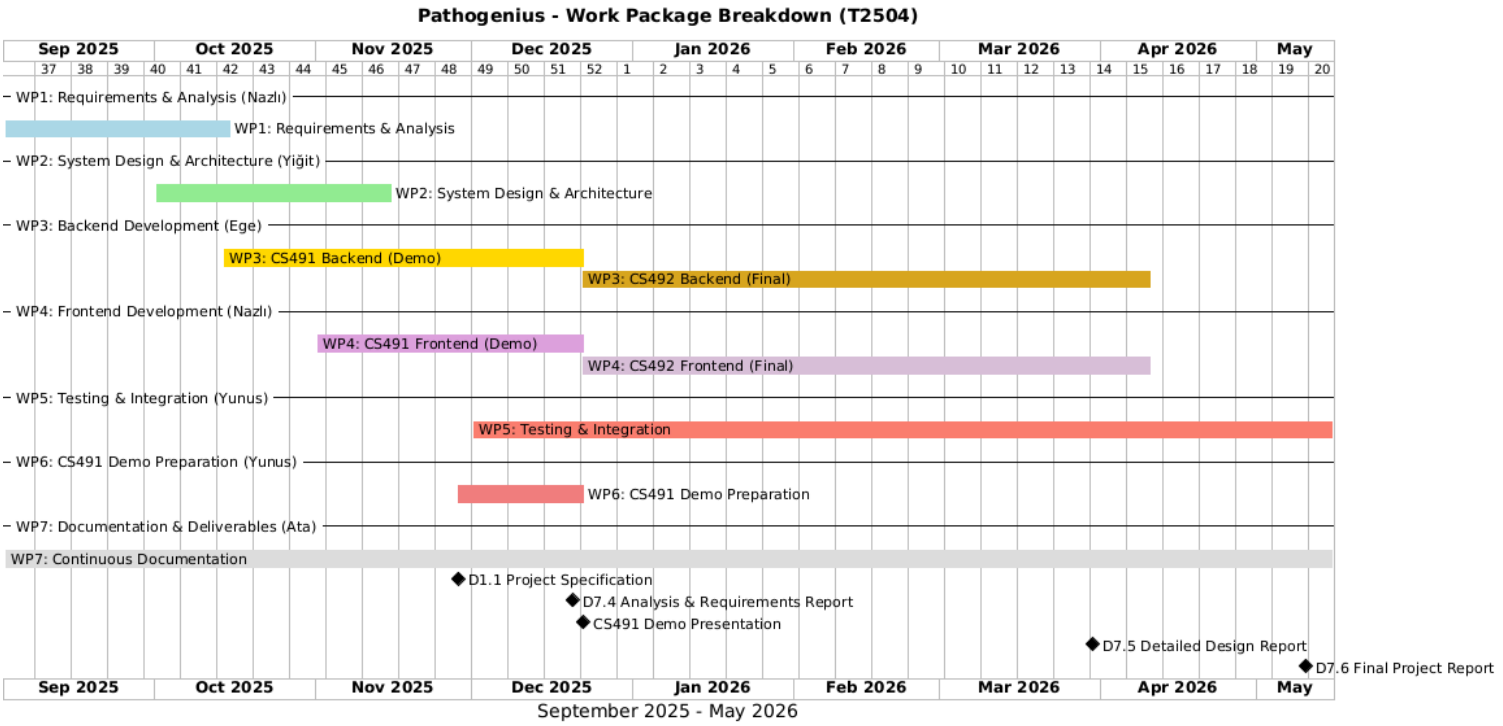

Figure 32. *Gantt Chart of the Work Packages.*

Pathogenius employs a work package-based planning methodology spanning September 2025 through May 2026, systematically breaking down the complex development effort into seven manageable, interconnected packages. This structured approach enables clear assignment of responsibilities with designated leaders for each package while ensuring all team members contribute across multiple areas, fostering shared leadership and collaborative ownership. By organizing work packages with explicit start/end dates, objectives, tasks, and deliverables, the team establishes transparent accountability and progress tracking mechanisms. The parallel execution of development work packages (backend and frontend) maximizes efficiency while dedicated packages for testing, demo preparation, and documentation ensure quality and course compliance without disrupting core development. Each work package is tracked as a set of GitHub Issues, providing real-time visibility into task status, blockers, and individual contributions, while the accompanying Gantt chart visualizes dependencies, critical paths, and

timeline adherence. This planning methodology directly addresses the iterative nature of the project. The CS491 demo milestone provides early validation and feedback, informing refinements in CS492, rather than treating the project as a single linear effort. The work package structure also facilitates risk management by identifying dependencies early and allowing the team to adjust resource allocation based on progress and challenges encountered. Most importantly, this approach ensures continuous documentation throughout development rather than retroactive report writing, as the Documentation package runs parallel to all technical work, systematically capturing decisions, models, and results as they occur. The result is a realistic, achievable plan that balances academic requirements with engineering best practices while maintaining flexibility to adapt to challenges discovered during implementation.

Table 3: List of work packages

WP#	Work package title	Leader	Members involved
WP1	Requirements & Analysis	Nazlı Apaydın	Ege Ateş, Yiğit Ali Doğan, Yunus Günay, Ata Uzay Kuzey
WP2	System Design and Architecture	Yiğit Ali Doğan	Nazlı Apaydın, Ege Ateş, Yunus Günay, Ata Uzay Kuzey
WP3	Backend Development	Ege Ateş	Yiğit Ali Doğan
WP4	Frontend Development	Nazlı Apaydın	Ata Uzay Kuzey, Yunus Günay
WP5	Testing & Integration	Yunus Günay	Ege Ateş, Yiğit Ali Doğan, Nazlı Apaydın, Ata Uzay Kuzey
WP6	Demo Preparation	Yunus Günay	Ege Ateş, Yiğit Ali Doğan, Nazlı Apaydın, Ata Uzay Kuzey
WP7	Documentations & Deliverables	Ata Uzay Kuzey	Nazlı Apaydın, Ege Ateş, Yiğit Ali Doğan, Yunus Günay,

WP 1: Requirements & Analysis			
Start date: 2 September 2025 **End date:** 15 October 2025			
Leader:	Nazlı Apaydın	**Members involved:**	Ege Ateş, Yiğit Ali Doğan, Yunus Günay, Ata Uzay Kuzey

Objectives: Establish foundational project requirements through comprehensive analysis. Define functional and non-functional requirements following IEEE 830 standards. Conduct feasibility studies across technical, economic, and ethical dimensions. Address consideration of various engineering factors, identify constraints and standards, document risks with alternatives, create detailed project plans, establish teamwork strategies, and plan for new knowledge acquisition.

Tasks:

Task 1.1 Market and Competitive Analysis: Research existing pathogen detection solutions. Identify gaps Pathogenius can address.

Task 1.2 Academic Literature Review: Review metagenomic sequencing literature. Validate Kraken2 technical approach.

Task 1.3 Functional Requirements Specification: Define detailed functional requirements for core features. Ensure testability and alignment with goals.

Task 1.4 Non-Functional Requirements Definition: Establish usability, reliability, performance, supportability, and scalability requirements following ISO/IEC 25010.

Task 1.5 Consideration of Various Factors: Analyze impact of public health, safety, welfare, global, cultural, social, environmental, and economic factors. Rate each factor 0-10 and create a summary table.

Task 1.6 Constraints and Standards Documentation: Document all constraints and engineering standards utilized (IEEE 830, ISO/IEC 25010, UML 2.5.1, ISO 9241-210).

Task 1.7 Risk Analysis and B Plan: Identify project risks and develop alternative plans. Create a risk summary table.

Task 1.8 Project Planning: Define work packages with leaders, members, dates, milestones, and deliverables. Create a Gantt chart.

Task 1.9 Teamwork Strategy: Document strategies for shared leadership, inclusive collaboration, and equal contribution. Plan evidence collection.

Task 1.10 Ethics and Professional Responsibilities: Identify ethical responsibilities to be fulfilled throughout the project.

Task 1.11 Learning Strategy Planning: Plan new knowledge acquisition and identify learning strategies.

Deliverables

D1.1: Project Specification Document

D1.2: Functional Requirements Specification

D1.3: Non-Functional Requirements Document

D1.4: Market Analysis Report

D1.5: Feasibility Study Report

D1.6: Constraints and Standards Compliance Document

WP 2: System Design and Architecture			
Start date: October 1, 2025 **End date:** November 15 2025			
Leader:	Yiğit Ali Doğan	**Members involved:**	Nazlı Apaydın, Ege Ateş, Yunus Günay, Ata Uzay Kuzey

Objectives: Translate requirements into detailed system architecture with complete UML modeling. Design four-layer architecture, decompose into subsystems, create all system models for Analysis Report. Develop use cases, class, activity, sequence, and state diagrams. Design UI wireframes and navigation. Define interfaces and data management strategy.

Tasks:

Task 2.1 Design Goals Establishment: Define usability, performance, reliability, maintainability, scalability, security goals. Prioritize based on requirements.

Task 2.2 High-Level Architecture Design: Create diagrams showing Sequencing Environment, Reference, Workflow, and Frontend layers with interfaces.

Task 2.3 Subsystem Decomposition: Decompose into manageable subsystems. Define responsibilities and dependencies.

Task 2.4 Use Case Model Development: Create use case diagrams with all actors and interactions.

Task 2.5 Object and Class Model Design: Develop class diagrams with entities, attributes, methods, and relationships.

Task 2.6 Dynamic Models - Activity Diagrams: Create activity diagrams for authentication, analysis workflow, and database update.

Task 2.7 Dynamic Models - Sequence Diagrams: Develop sequence diagrams for FASTQ analysis, authentication, and database update.

Task 2.8 Dynamic Models - State Diagrams: Create state diagrams for Analysis, User Session, and Database objects.

Task 2.9 User Interface Design: Design UI navigation and create wireframes for all major screens with navigational paths.

Task 2.10 Hardware/Software Mapping: Document deployment architecture and hardware requirements.

Task 2.11 Data Management Strategy: Design file-based storage for FASTA, indices, results, and configuration.

Task 2.12 API and Interface Specification: Specify interfaces between components, data formats, and error codes.

Deliverables

D2.1: Design Goals Document

D2.2: High-Level Architecture Diagram

D2.3: Subsystem Decomposition Diagram

D2.4: Use Case Diagrams

D2.5: Class Diagrams

D2.6: Activity Diagrams

D2.7: Sequence Diagrams

D2.8: State Diagrams

D2.9: UI Wireframes and Navigation Structure

D2.10: Hardware/Software Mapping Documentation

D2.11: Data Management Specification

D2.12: API and Interface Specifications

WP 3: Backend Development			
Start date: October 14, 2025 **End date:** April 15 2025			
Leader:	Ege Ateş	**Members involved:**	Yiğit Ali Doğan

Objectives: Implement core computational pipeline across both semesters. CS491: develop simplified pipeline with pre-made FASTQ and dockerized Kraken2 for demo. CS492: add basecaller integration, complete preprocessing, full database management, and optimizations. Develop Snakemake workflows, integrate Kraken2, ensure robust error handling.

Tasks:

Task 3.1 Development Environment Setup: Configure Git, dependencies, Docker, and CI/CD pipeline.

Task 3.2 Snakemake Workflow Implementation: Implement modular rules for preprocessing, classification, and output processing. CS491: simplified workflow. CS492: complete workflow with basecaller.

Task 3.3 Docker Configuration (CS491): Create container with Kraken2 and pre-built database subset.

Task 3.4 Preprocessing Module (CS492): Integrate quality control tools for long reads. Handle compressed formats.

Task 3.5 Kraken2 Integration: Interface with dockerized (CS491) then local (CS492) Kraken2. Implement confidence scoring.

Task 3.6 Basecaller Integration (CS492): Integrate Icarus simulator or actual device. Convert raw signals to FASTQ.

Task 3.7 Reference Database Management: CS491: prepare pre-built subset. CS492: implement full management with import and update tools.

Task 3.8 Output Processing Module: Aggregate results, compute statistics, calculate confidence, generate JSON/CSV output.

Task 3.9 Result Storage System: Implement file-based storage for analysis history and results.

Task 3.10 Resource Management: Monitor CPU/memory, prevent overflow, support configurable limits.

Task 3.11 Error Handling and Logging: Implement comprehensive error handling and structured logging.

Task 3.12 Testing Data Preparation: CS491: curate pre-made FASTQ. CS492: configure simulator.

Deliverables

D3.1: Development environment with CI/CD

D3.2: Snakemake workflow (demo and final versions)

D3.3: Dockerized Kraken2 (CS491)

D3.4: Preprocessing module (CS492)

D3.5: Kraken2 integration

D3.6: Basecaller integration (CS492)

D3.7: Database management system (CS492)

D3.8: Output processing module

D3.9: Result storage system

D3.10: Resource management system

D3.11: Logging framework

D3.12: Test datasets (CS491) and simulator (CS492)

WP 4: Frontend Development			
Start date: October 14 2025 **End date:** April 15 2025			
Leader:	Nazlı Apaydın	**Members involved:**	Ata Uzay Kuzey, Yunus Günay

Objectives: Create intuitive Electron.js interface for non-expert users. CS491: develop basic interface showing workflow and results. CS492: refine based on feedback, add features, improve error handling. Communicate uncertainty appropriately. Follow ISO 9241-210 user-centered design principles.

Tasks:

Task 4.1 Electron.js Setup: Initialize project structure. Configure build for cross-platform deployment.

Task 4.2 Analysis Management Interface: Create file upload, analysis history list, and reopen/export features.

Task 4.3 Workflow Execution Interface: Display processing status, stage indicators, and completion notifications.

Task 4.4 Result Visualization: Create tables, charts showing species with confidence scores. Color-code confidence levels.

Task 4.5 Dataset Management Interface: Display database information. CS492: add FASTA import and rebuild features.

Task 4.6 Notification System: Implement completion, error, and status notifications.

Task 4.7 Settings Interface: Configure resources, adjust parameters, view system info.

Task 4.8 Help System: Create tooltips, contextual help, and troubleshooting guide.

Task 4.9 Error Handling (CS492): Enhance validation and user-friendly error messages.

Task 4.10 UI Polish (CS492): Refine based on feedback. Improve visual design and accessibility.

Deliverables

D4.1: Electron.js application framework

D4.2: Analysis management interface

D4.3: Workflow execution interface

D4.4: Result visualization components

D4.5: Database management interface

D4.6: Notification system

D4.7: Settings panels

D4.8: Help system

D4.9: Enhanced error handling (CS492)

D4.10: Polished UI (CS492)

WP 5: Testing & Integration

Start date: December 1 2025 **End date:** May 15 2025

Leader:	Yunus Günay	**Members involved:**	All team members

Objectives: Ensure continuous quality assurance for both demo and final versions. Conduct unit, integration, system, and performance testing. Develop 50+ test cases for Detailed Design Report. Execute tests with results for Final Report. Coordinate integration of separately developed components.

Tasks:

Task 5.1 Test Planning: Develop test plan for both CS491 and CS492 phases.

Task 5.2 Unit Testing: Create unit tests for backend modules. Achieve good coverage.

Task 5.3 Integration Testing - Demo (CS491): Test dockerized setup, pre-made file processing, frontend-backend communication.

Task 5.4 Integration Testing - Final (CS492): Test basecaller integration, local Kraken2, database operations.

Task 5.5 System Testing - Demo: End-to-end testing with curated datasets. Practice demo execution.

Task 5.6 System Testing - Final: Comprehensive testing with simulator/device. Test diverse scenarios and file sizes.

Task 5.7 Test Case Specification (CS492): Develop test cases with ID, type, objective, procedure, expected results, priority.

Task 5.8 Performance Testing: Measure processing times, memory usage, throughput. Compare demo vs final performance.

Task 5.9 Accuracy Validation: Validate classification with known compositions. Document accuracy metrics.

Task 5.10 Non-Functional Testing: Test usability, reliability, offline operation, error handling, cross-platform.

Task 5.11 Test Execution and Documentation: Execute all cases. Document results with pass/fail status and bug tracking.

Task 5.12 Demo Verification (CS491): Intensive testing of demo configuration for reliable presentation.

Task 5.13 Regression Testing (CS492): Ensure existing functionality remains working as the system evolves.

Task 5.14 Integration Coordination: Coordinate merging of backend and frontend. Resolve interface issues.

Deliverables

D5.1: Test Plan

D5.2: Unit test suite

D5.3: Demo integration tests

D5.4: Final integration tests

D5.5: Demo system test results

D5.6: Final system test results

D5.7: Test Case Specifications (50+)

D5.8: Performance reports

D5.9: Accuracy validation report

D5.10: Non-functional test results

D5.11: Test execution log for Final Report

D5.12: Demo verification report

D5.13: Regression test results

D5.14: Integration coordination log

WP 6: Demo Preparation			
Start date: November 28 2025 **End date:** December 22 2025			
Leader:	Yunus Günay	**Members involved:**	All team members

Objectives: Prepare polished CS491 demo presentation. Create dockerized setup with pre-made data. Develop presentation materials and conduct rehearsals. This separate package exists because demo implementation differs from the final system.

Tasks:

Task 6.1 Docker Configuration: Finalize container with Kraken2 and ~20-30 pathogen genomes. Test reliability.

Task 6.2 Dataset Curation: Create pre-made FASTQ files representing realistic scenarios. Document expected results.

Task 6.3 Demo Pipeline Integration: Integrate dockerized Kraken2 with simplified workflow. Optimize for demo timing.

Task 6.4 Frontend Polish: Refine UI for professional presentation.

Task 6.5 Presentation Slides: Create 2-4 slides with elevator pitch, status dashboard, architecture diagram, and roadmap.

Task 6.6 Demo Script: Write detailed script with speaking roles for all members. Assign responsibilities.

Task 6.7 Backup Plans: Prepare video, screenshots, and explanations if live demo fails.

Task 6.8 Environment Setup: Prepare laptops with a configured system. Create a setup checklist.

Task 6.9 Limitations Documentation: Document what is simplified for demo vs final version.

Deliverables

D6.1: Dockerized Kraken2 with database

D6.2: Pre-made FASTQ files with documentation

D6.3: Demo-ready integrated system

D6.4: Polished demo UI

D6.5: Presentation slides

D6.6: Demo script

D6.7: Backup materials

D6.8: Rehearsal recordings

D6.9: Setup checklist

D6.10: CS491 Demo Presentation

WP 7: Documentation & Course Deliverables

Start date: September 2 2025 **End date:** May 15 2025

Leader:	Ata Uzay Kuzey	**Members involved:**	All team members

Objectives: Produce all required CS491 and CS492 course deliverables. Maintain individual logbooks throughout the project. Develop a comprehensive user manual. Create technical documentation. This final package integrates all project work into formal reports.

Tasks:

Task 7.1 Individual Logbooks (Continuous): Each member maintains Google Docs logbook with timeline, reflections, work samples, and progress. Update weekly.

Task 7.2 Project Information Form (September): Submit form with project name, description, supervisor, and website.

Task 7.3 Project Specification Document (November): [COMPLETED ✓] Introduction, constraints, standards, requirements, feasibility.

Task 7.4 Analysis and Requirements Report (December - CS491): Comprehensive report with current system, proposed system, models (scenarios, use cases, class diagrams, activity/sequence/state diagrams, UI mockups), factors consideration, constraints, risks/B plan, project plan with Gantt chart, teamwork strategy, ethics, learning plan.

Task 7.5 Detailed Design Report (March/April - CS492): Design goals, architecture analysis, subsystem decomposition, 50+ test cases, factors consideration, teamwork details.

Task 7.6 Final Project Report (May - CS492): Requirements, architecture, implementation details, test results, maintenance plan, factors/ethics/teamwork/meeting objectives, new knowledge acquired.

Task 7.7 User Manual: Installation instructions, getting started, workflow guide, result interpretation, troubleshooting, FAQ, sample scenarios.

Task 7.8 Technical Documentation: Architecture docs, API/interface docs, code comments, workflow documentation.

Task 7.9 Project Website: Team info, project description, GitHub links, documentation downloads, demo video, presentations.

Task 7.10 Presentation Materials: CS491 and CS492 slides with demo scripts.

Task 7.11 Standards Compliance: Document IEEE 830, ISO/IEC 25010, UML 2.5.1, ISO 9241-210 compliance.

Task 7.12 Final Package Assembly (May): Compile all reports, manual, code, installers, and documentation.

Deliverables

D7.1: Individual logbooks (continuous)

D7.2: Project Information Form

D7.3: Project Specification Document

D7.4: Analysis and Requirements Report (CS491)

D7.5: Detailed Design Report (CS492)

D7.6: Final Project Report (CS492)

D7.7: User Manual

D7.8: Technical Documentation

D7.9: Project Website

D7.10: Presentation Materials

D7.11: Standards Compliance Documentation

D7.12: Final Documentation Package

4.4 Ensuring Proper Teamwork

To handle the broad set of requirements for the project the team employs:

- Regular Meetings: Weekly meetings both face to face and online, code milestones to complete before deadlines and frequent meetings with the advisor for feedback and addressing of emergency issues with code.
- Clear Role Assignments: The role assignments are clear with clear mission boundaries between the roles. Frontend engineers ensure the simplicity and usability of the application, backend engineers create an efficient workflow for the integration and running of the project while the hardware engineers will work on the GPU mapping of the classification algorithm.
- Collaboration and Deployment Tools: Github issues and issue boards for the milestones and code completion, github for version control and code sharing alongside slack for team coordination.

Task assignments of members:

- **Ege Ateş:**
  - Ege will focus on backend development, including the Snakemake workflow, implementing services and integrating the databases.
  - He will also work on the integration of signal simulators for reads and ensure the overall scalability of the project.
- **Nazlı Apaydın:**
  - Nazlı will focus on the frontend development, focusing on creating a user friendly design for the application with clear instructions and flows.
  - She will also work on creating the web depictions of the data obtained from the reads, representing them in easy to understand graphs.
- **Ata Uzay Kuzey:**
  - Ata will focus on the frontend development and the creation of web reports and graphs from obtained reads.
  - He will manage the robust integration of backend services and the local AI assistant to provide human readable results for the end user.
  - Additionally he leads the continuous documentation and course deliverables webpage.
- **Yunus Günay:**
  - Yunus will act as the lead for testing and integration, ensuring continuous quality assurance through unit, system and performance testing.
  - He is responsible for the demo preparations, creation of the dockerized setups and the environments.
  - As a cross functional member, he coordinates the merging of backend and frontend components and resolves interface issues. He contributes to the frontend work as well.
- **Yiğit Ali Doğan:**
  - Yiğit leads the system design and architecture, creating high level UML models and subsystem decompositions.
  - He will work on the backend services, creating the Snakemake workflow and integrating services.
  - Furthermore, he will specialize in the hardware/software mapping, specifically the NVIDIA CUDA acceleration for the classification algorithm and increased performance.

4.5 Ethics and Professional Responsibilities

Pathogenius must uphold high ethical standards to ensure its reliability and performance in critical clinical and field settings:

- **Privacy and Confidentiality:** Since the clinical samples may contain human host DNA, the system is constrained to process all data transiently on the local device in order to ensure that the sensitive genomic information is not exposed or uploaded to external servers.
- **Decision Support vs Diagnosis:** The system is strictly defined as a decision support tool rather than a definitive and certain diagnostic device. The interface clearly presents results as probabilistic evidence to prevent the over-interpretation by non expert users.
- **Transparency and Uncertainty:** Species level results are communicated with appropriate uncertainty markers and confidence scores. Low confidence scores findings are visually distinguished to support medical decision making.
- **Equitable Access:** To ensure accessibility in resource limited settings, the software is free of charge and does not rely on recurring subscription costs when used in offline mode.
- **Professional Conduct:** The development process adheres to recognized engineering standards including but not limited to: IEEE 830 for requirement specification and ICO/IEC 25010 for software quality models.

4.6 Planning for New Knowledge and Learning Strategies

Continuous learning is critical to navigate the evolving requirements of bioinformatics and portable diagnostics hardware:

- **Ongoing Research and Training:** Members continuously study metagenomics classification literature and validate Kraken2 or other classification algorithms' performance on modest hardware.
- **Knowledge Sharing:** Team members utilize individual logbooks and weekly meetings to share best practices in Snakemake workflow management and user centered design.
- **Technical Mastery:** Hardware specialists are acquiring NVIDIA CUDA skills to transition from CPU based tasks to GPU accelerated processing.
- **Iterative Improvement:** The phased development approach uses the CS491 milestones and additional meetings with our advisor to gather performance feedback, which informs technical refinements for the final implementation.

5 Glossary

1. **FASTQ:** A sequencing read file format that stores nucleotide sequences with per-base quality scores.
2. **FASTA:** A sequence file format typically used for storing reference genome sequences.
3. **Read:** A single nucleotide sequence produced by a sequencing device and stored in a FASTQ file.
4. **Sequencing:** The process of generating nucleotide read data from biological samples.
5. **Long-Read Sequencing:** Sequencing that produces long read fragments (used as the system's primary input style).
6. **Kraken2:** A k-mer based taxonomic classification tool used as the main classifier in the pipeline.
7. **k-mer:** A substring of length *k* extracted from reads/genomes, used for matching and classification.
8. **Index (k-mer index):** A Kraken2-compatible database structure enabling fast k-mer lookups during classification.
9. **Snakemake:** A workflow engine used to coordinate and execute the analysis pipeline reproducibly.
10. **Workflow:** An ordered set of steps (rules) executed to process inputs into final outputs.
11. **Pipeline:** The end-to-end chain of preprocessing, classification, and post-processing stages.
12. **Electron.js:** The framework used to build the application interface.
13. **Offline Execution:** System operation without requiring internet connectivity during analysis.
14. **Reference Database:** Locally stored genomes used by Kraken2 to classify reads.
15. **Metagenomic Analysis:** Sequencing-based analysis of genetic material from mixed samples to identify organisms.
16. **Taxonomic Classification:** Assigning reads to taxa such as species based on sequence similarity evidence.
17. **NVIDIA CUDA:** The GPU computing platform referenced for future acceleration support.
18. **NCBI (National Center for Biotechnology Information):** A source of curated genomic resources used for building local reference databases.
19. **GPU:** The graphics processor targeted for acceleration in later project stages.
20. **CPU:** The host processor used for analysis when GPU acceleration is unavailable or disabled.
21. **VRAM:** GPU memory that can limit accelerated processing for large workloads.
22. **Sequencing Depth:** The total number of reads representing a sample, influencing detection sensitivity.

23. **False Positive:** An incorrect classification where a read is assigned to a taxon not actually present in the sample.
24. **Confidence Score:** A numerical measure indicating the reliability of a taxonomic classification result.
25. **Low-Abundance Pathogen:** An organism present at a small proportion within a sample, making detection more difficult.
26. **Preprocessing:** Initial analysis steps applied to raw FASTQ data, such as quality filtering and cleanup, before classification.
27. **Post-Processing:** Analysis steps applied after classification to aggregate results, compute statistics, and generate reports.
28. **Batch Processing:** Executing analysis on subsets of reads sequentially to manage memory and resource constraints.
29. **Index Building:** The process of converting reference genomes into a searchable k-mer index for Kraken2.
30. **Deterministic Execution:** Pipeline behavior that produces identical outputs when given the same inputs and reference data.

6 References

[1] Y. L. Oon, Y. S. Oon, M. Ayaz, M. Deng, L. Li, and K. Song, “Waterborne pathogens detection technologies: advances, challenges, and future perspectives,” *Frontiers in Microbiology*, vol. 14, Art. no. 1286923, Nov. 2023, doi: 10.3389/fmicb.2023.1286923.

[2] Bio-Rad Laboratories, “Pathogen detection,” Bio-Rad Laboratories, Online. Available: https://www.bio-rad.com/en-tr/a/ls/pathogen-detection. Accessed: Dec. 18, 2025.

[3] Norgen Biotek Corp., “Waterborne pathogen detection,” Norgen Biotek, Online. Available: https://norgenbiotek.com/category/waterborne-pathogen-detection. Accessed: Dec. 18, 2025.

[4] K. Sandås, J. Lewerentz, E. Karlsson, L. Karlsson, D. Sundell, K. Simonyté-Sjödin, and A. Sjödin, “Nanometa Live: a user-friendly application for real-time metagenomic data analysis and pathogen identification,” *Bioinformatics*, vol. 40, no. 3, Art. no. btae108, Mar. 2024, doi: 10.1093/bioinformatics/btae108.

[5] L. E. Braley, J. B. Jewell, J. Figueroa, J. L. Humann, D. Main, G. A. Mora-Romero, N. Moroz, J. W. Woodhall, R. A. White III, and K. Tanaka, “Nanopore sequencing with GraphMap for comprehensive pathogen detection in potato field soil,” *Plant Disease*, vol. 107, no. 8, pp. 2288–2295, Aug. 2023.

[6] International Organization for Standardization, *Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—System and software quality models*, ISO/IEC 25010:2011, 2011.

[7] Object Management Group, *Unified Modeling Language (UML) Specification*, Version 2.5.1, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1

[8] International Organization for Standardization, *Systems and software engineering—Life cycle processes—Requirements engineering*, ISO/IEC/IEEE 29148:2018, 2018.