RASD

Requirements Analysis and Specification Document for "Production Optimizer"

Authors: Fran Hruza, Danko Čurlin, Georgs Lukass Rozins, Paula Šalković, Shreesh Kumar Jha, Giovanni Orciuolo, Samarth Bhatia

Version: 3

Date: 10.01.2025.

Supervisors: Elisabetta Di Nitto (POLIMI), Igor Čavrak (FER)

Table of Contents

Chapter 1: Project Introduction

1.1 Purpose

The purpose of this project is to develop a web-based platform designed to enhance production efficiency through the use of service tools. In modern industries, production processes often face challenges such as bottlenecks, resource overuse, and suboptimal scheduling. This platform addresses these issues by offering users the ability to analyze production data and execute tailored service tools, enabling informed decision-making.

1.1.1 The importance of optimization

Production service tools are mathematical tools used to identify the best outcomes within defined constraints, such as time, cost, or resource availability. These service tools enable users to uncover strategies that improve productivity, reduce costs, and minimize waste. For example, they can determine the most efficient way to allocate resources across a factory floor, plan production schedules, or manage inventory levels. In short, they transform raw data into actionable insights that improve operational efficiency.

1.1.2 Features of the Platform

The platform is designed to be secure, scalable, and easy to use, while also providing powerful tools for optimization. Key functionalities include:

1. Data Upload: Users can upload production data in common formats, such as Excel files, ensuring accessibility for diverse user groups.
2. Optimization Execution: Users can run domain-specific service tools to address challenges like resource allocation, production scheduling, and cost minimization.
3. Visualization of Results: Outputs are presented through PLT graphs—a set of visualizations generated using Matplotlib, a Python-based library. These graphs help users analyze results clearly and effectively.
4. Historical Optimization Data: The platform stores results from past optimization runs, allowing users to review and compare historical performance trends.
5. Scenario Comparison: Users can analyze and compare multiple optimization outcomes to evaluate the impact of different input parameters or cons.

1.1.3 Key Concepts

- *Production Service Tools:*
    - These service tools are specifically designed to improve production-related processes, focusing on factors such as resource utilization, machine scheduling, and workflow efficiency. Unlike generic optimization tools, production service tools account for the unique constraints and requirements of industrial operations.

- *Execution of Service Tools:*
    - Execution refers to running a service tool using user-provided data to generate actionable outputs. These outputs help users identify the most effective solutions to their production challenges.

- *PLT Graphs:*
    - PLT graphs are visual representations of optimization results created using Python's Matplotlib library. Examples include line charts, bar graphs, and scatter plots. These graphs simplify complex data, making results easy to interpret and compare.

- *Historical Optimizations:*
    - This feature allows users to access and analyze results from previous optimization runs. By reviewing historical data, users can track progress, identify trends, and refine their strategies over time.

- *Significance of the Platform:*
    - This platform bridges the gap between complex optimization techniques and practical applications in production environments. By providing a user-friendly interface, domain-specific tools, and powerful visualization capabilities, it enables users to make data-driven decisions that enhance production efficiency and overall operational performance.

1.2 Goals

The primary goal of this project is to create a robust platform that addresses the practical challenges of optimizing production processes through user-friendly tools, efficient data management, and actionable insights. These goals focus on providing practical solutions that cater to the needs of production managers, engineers, and analysts who rely on data-driven decision-making to improve efficiency and reduce costs.

1.2.1 Key Goals and Objectives:

[G1] Create a Secure and Scalable Platform

The platform must ensure security and reliability while scaling to meet user demands.

- Implement robust authentication and authorization to protect sensitive production data.
- Guarantee data privacy and security, safeguarding both user inputs and optimization results.
- Support concurrent execution of service tools, ensuring smooth performance even with multiple users.
- Enable horizontal scaling to accommodate growing user and service tool demand.

[G2] Enable Efficient Service Tool Management

Managing service tools is central to the platform's purpose, enabling users to interact with production-specific optimization tools.

- Support multiple service tools for different production challenges.
- Allow users to link service tools to workflows and edit existing configurations.
- Provide service tool-specific access control to ensure secure and flexible usage.
- Support standalone API access for seamless integration with external systems.

[G3] Optimize Data Handling

Handling production data efficiently is critical for enabling meaningful optimization.

- Allow users to upload production data through Excel files, focusing on seamless and error-free handling of production-specific formats.
- Visualize results clearly by displaying service tool response graphs, including PLT graphs for trend analysis and scenario comparison.
- Enable users to store and manage historical optimization data for long-term insights.
- Provide tools to compare optimization results, helping users evaluate scenarios and choose the most effective solutions.

[G4] Implement Role-based Access

Introduce a role-based access system to ensure secure and controlled platform usage.

- Define and support distinct roles, including admin and regular users.
- Facilitate granular permission management to tailor access to specific resources.
- Enable service tool-specific access control to ensure secure and precise usage.
- Incorporate an invitation-based registration system to manage user onboarding effectively.

[G5] Provide Comprehensive Analytics

Offer analytics tools to track performance and generate insights for effective decision-making.

- Develop tools to track service tool usage and evaluate performance metrics.
- Monitor user activity for insights into platform engagement and behavior.
- Generate detailed usage reports to support administrative oversight and decision-making.
- Collect and analyze performance metrics to identify optimization opportunities and maintain operational efficiency.

[G6] Ensure System Reliability

Ensure the platform remains dependable and operational under all conditions.

- Implement robust error-handling mechanisms to minimize disruptions and ensure smooth operation.
- Provide comprehensive system health monitoring to detect and address issues proactively.
- Enable automated backups to safeguard data and ensure quick recovery in case of failures.
- Support high availability ensures the platform remains accessible and operational under various conditions.

1.2.2 Addressing Practical Needs:

The goals above focus on solving real-world challenges faced in production optimization. They directly relate to the core functions identified in the purpose section:

- Upload Production Data:
  - The platform will support uploading production-specific data (specified in the optimizer's documentation) through Excel files, ensuring compatibility with user needs.
- Visualize Results through PLT Graphs:
  - Detailed response graphs will visualize outcomes, enabling users to interpret data easily.
- Compare Optimization Results:
  - Users will be able to analyze different scenarios side by side, empowering informed decision-making.

By aligning these goals with user-centric needs, the platform will deliver practical solutions that address the specific challenges of improving production efficiency through optimization.

1.3 Product Scope

Core Capabilities:

1. User Interaction and Management

The platform will enable users to interact with the system seamlessly and securely.

- Authentication and User Management: Provide secure login, registration, and role-based access control.
- Responsive Interface: Design intuitive and accessible navigation across devices.
- Error and Progress Feedback: Display clear error messages, loading indicators, and progress tracking for user actions.

2. Data Handling

The system will facilitate seamless management and processing of user data.

- Data Upload: Support uploading of production-specific Excel files.
- Data Storage and Management: Ensure secure and efficient storage of user inputs, service tool configurations, and results.

- Historical Data Access: Allow users to review and analyze optimization results from past runs.

3. Service Tool Execution and Management

The platform will support executing and managing multiple service tools tailored to production challenges.

- Service Tool Selection and Configuration: Enable users to choose and configure service tools to suit their specific needs.
- Service Tool Execution: Process user inputs through selected service tools and generate actionable outputs.
- Scenario Comparison: Allow users to compare optimization results to evaluate the impact of varying inputs or constraints.

4. Results Visualization and Reporting

Clear visualization of results is essential for actionable insights.

- Results Display: Use graphical representations (e.g., PLT graphs) to visualize service tool outputs effectively.
- Result Storage and Download: Enable users to save and download optimization results for further analysis.
- Analytics Dashboard: Provide tools to track service tool usage, user activity, and system performance metrics.

5. System Reliability and Security

The platform will prioritize reliability and security to ensure consistent and trustworthy operations.

- Error Handling and Recovery: Implement mechanisms to detect and address errors proactively.
- Data Security: Use encryption and secure file handling to protect sensitive production data.

High Availability: Ensure minimal downtime and smooth user experiences through automated backups and performance monitoring.

1.4 Terminology and Definitions

Term	Full Name	Description
ANON	Anonymous User	Unauthenticated user with access only to login/registration request
REG	Regular User	An authenticated user with service tool access rights
ADMIN	Administrator	User with full system administrative privileges
PLT	Plot	Graph output from service tools
OPT	Optimization	Process of running production data through service tools
TOOL	Service Tool	Python-based algorithm for production optimization
FE	Frontend	Web-based user interface
BE	Backend	Server-side application logic
API	Application Programming Interface	Interface for programmatic access to service tools
UI/UX	User Interface/Experience	Application interface and interaction design

1.5 Revision History

Version	**Date**	**Description**	**Author**
1.0	Oct, 2024	Initial Draft	Shreesh Kumar Jha, Samarth Bhatia, Danko, Franz
2.0	Oct, 2024	Changes Made After DIscussion with Customers	Shreesh Kumar Jha, Samarth Bhatia, Danko, Franz
2.1	21 Nov 2024	Major Changes	Shreesh Kumar Jha, Samarth Bhatia, Danko
2.2	4 Dec 2024	Major revision addressing feedback	Shreesh Kumar Jha, Samarth Bhatia, Georgs Lukass Rozins
3	10 Jan 2025	Revision, formatting, checking compliance with guidelines	Georgs Lukass Rozins

Chapter 2: Requirements Analysis

2.1 Functional Requirements

2.1.1 User Authentication and Management

Secure and reliable user authentication and management are critical for the platform. The requirements include:

[FR1] User Authentication

- Support for email and password-based login.
- Implementation of session management to track user activity securely.
- Password reset functionality for user convenience.
- Account lockout after multiple failed login attempts to enhance security.

[FR2] Account Management

- Invitation-based registration for controlled access.
- Support for email changes by users.
- Allow users to change their passwords securely.
- Enable account deactivation for user control over their profiles.

[FR3] Admin Controls

- The ability for admins to invite new users to the platform.
- Manage access to service tools based on user roles.
- Assign and update roles for users (e.g., admin, regular user).
- Monitor user activity and service tool activity for operational oversight.

[FR4] Session Handling

- Automatic session timeout to log out inactive users.
- Support for concurrent session management, allowing/disabling multiple sessions per user.
- Remember me functionality for easier access without compromising security.
- Secure logout to prevent unauthorized access after user sign-out.

2.1.2 Service Tool Management

This section focuses on handling service tools, from configuration to execution.

[FR5] Service Tool Support

- Support for multiple service tools to address varied production challenges.
- Allow users to configure service tool parameters for flexibility.
- Provide options for service tool input and output configuration.
- Ensure input validation to prevent errors and enhance data integrity.

[FR6] Admin Service Tool Controls

- Allow admins to link service tools to workflows.
- Implement access control to ensure only authorized users can interact with specific service tools.
- Enable admins to modify existing service tools when updates are required.

[FR7] Service Tool Access

- Implement role-based access to grant permissions based on user roles.
- Support user-based access for custom permissions where needed.

[FR8] API Integration

- Provide RESTful API endpoints for seamless integration with external systems.
- Support authentication tokens for secure API access.
- Implement robust error handling for reliable interactions with the API.

Note: For a detailed view of how these service tool management requirements are implemented in the system flow, refer to the Optimization Process Sequence Diagram.

2.1.3 File Operations

Handling production data and results efficiently is a key aspect of the platform.

[FR9] File Upload

- Support for Excel file uploads, focusing on production-specific data formats.
- Provide upload progress tracking for user feedback.

[FR10] Result Downloads

- Enable downloads in multiple formats to suit diverse needs.
- Support for downloading the entire service tool response or specific outputs.
- Implement error handling to ensure smooth operation.

[FR11] Service Tool Output Graph Transfer

- Allow users to save and retrieve output graphs for review.
- Provide the ability to generate graphs based on raw values.
- Support download options for graphical results.

[FR12] Historical Data

- Store both input and output data for past optimization runs.
- Include search functionality for easy access to historical records.
- Enable users to compare different historical results to identify trends or improvements.

2.1.4 Analytics and Monitoring

Tracking and analyzing platform usage ensures ongoing improvement and reliability.

[FR13] Usage Statistics

- Track metrics such as service tool execution counts, login activity, and success/failure rates.
- Measure response times for system efficiency.

[FR14] Performance Metrics

- Monitor system health to ensure consistent and reliable operation.

[FR15] Justification of System

The system plays a key role in facilitating the registration process, ensuring that requests are handled efficiently and securely:

- Validates inputs to prevent errors such as invalid email formats.
- Checks for duplicates to avoid redundant processing.
- Ends notifications to both admins and users, streamlining communication.
- Maintains logs of all registration requests for future reference and auditing.

2.2 Actors and Stakeholders

2.2.1 Primary Actors

1. **Anonymous User**
    - Can request access to the platform
    - Can view basic platform information
2. **Regular User**
    - Can access assigned service tools
    - Can upload input files
    - Can view and download the results
    - Can track optimization history
3. **Administrator**
    - Can manage user access
    - Can configure service tools
    - Can view system analytics
    - Has all regular user capabilities

2.2.2 Supporting Actors

1. **Service Tool Execution**
    - Executes optimization calculations
    - Generates PLT graphs
    - Provides performance metrics
2. **Authentication Service**

    - Manages user authentication
    - Handles session management
    - Controls access permissions

2.3 Use Cases

UC1: User Registration

Actor: Anonymous User

Purpose: Allow new users to request access to the platform, ensuring that only authorized individuals are granted access.

Entry Condition: The user does not have access to the platform and needs to request registration.

Main Event Flow:

1. The user visits the registration request page.
2. The user provides their email and company details in the registration form.
3. The user submits the registration request.
4. The system validates the provided email format and checks for duplicate email entries.
5. If the email is valid and unique, the system notifies the admin of the new registration request.
6. The admin reviews the request, including the provided business justification.
7. Based on the review, the admin either approves or rejects the request:
8. If approved, the system sends a registration invitation to the user's email.
9. If rejected, the system sends a rejection notification to the user, including the reason for denial.

Exit Condition:

The user receives either a registration invitation or a rejection notification.

Exception Handling:

1. Invalid Email Format:
    a. The system immediately notifies the user of the error when the form is submitted.
    b. The user is prompted to provide a valid email address before re-submitting the request.
2. Duplicate Email Request:

   a. The system detects duplicate requests and displays an error message to the user, indicating that a request with the same email is already under review or has been processed.
   b. The user is advised to contact support if they believe this is an error.

UC2: Service Tool Execution

Actor: Regular User

Purpose: Allow authenticated users to execute service tools, ensuring proper data validation and results generation.

Entry Condition: The user is authenticated and has access to the required service tool.

Main Event Flow:

1. The user selects the service tool.
2. The user uploads an Excel input file.
3. The system validates the file format and checks for errors.
4. Upon successful validation, the service tool processes the data.
5. The system generates PLT graphs based on the processed data.
6. Results are made available for download.

Exit Condition: The optimization results are successfully generated and available for the user to download.

Exception Flow:

- If the file format is invalid, the system notifies the user and prompts for a valid file upload.
- If a service tool execution error occurs, the system notifies the user and logs the error for further analysis.

Special Requirements:

- The uploaded Excel file size must not exceed 10MB.
- Service tool processing must be completed within a 5-minute timeout period.

UC3: Result Management

Actor: Regular User

Purpose: Enable users to manage and review optimization results effectively, including viewing, downloading, and comparing runs.

Entry Condition: The user has successfully completed one or more optimization runs.

Main Event Flow:

1. The user accesses their optimization history.
2. The user selects a specific optimization run.
3. The system displays the associated PLT graphs.
4. The user downloads the results for local review.
5. The user optionally compares the selected run with up to two other optimization runs.

Exit Condition: The user successfully obtains the required results and/or completes the desired comparisons.

Exception Flow:

- If the requested results are not found, the system notifies the user and provides troubleshooting options.
- If a download error occurs, the system notifies the user and prompts to retry the download.

Special Requirements:

- Optimization results are retained in the system for a maximum of 30 days.
- Comparison functionality is restricted to a maximum of three optimization runs at a time.

UC4: User Management

Actor: Administrator

Purpose: Allow administrators to manage user accounts, configure access permissions, and assign service tools efficiently while maintaining system oversight.

Entry Condition: The administrator is authenticated and authorized to perform user management tasks.

Main Event Flow:

1. The administrator accesses the user management section of the system.
2. The administrator reviews pending user access requests.
3. Based on the review, the administrator approves or denies access requests.
4. The administrator manages user permissions by adding, modifying, or revoking specific permissions.

5. The administrator assigns appropriate service tools to users as needed.
6. The administrator reviews user activity logs to ensure compliance with system usage policies.

Exit Condition: User access permissions and assignments are successfully configured and updated.

Exception Flow:

- If the system quota for users or service tools is reached, the system notifies the administrator and prevents further additions.
- If an invalid combination of permissions is attempted, the system provides an error message and suggests corrective actions.

Special Requirements:

- All user management actions must be recorded in audit logs for accountability.
- Email notifications must be sent to users and administrators for any access or permission changes.

UC5: Service Tool Configuration

Actor: Administrator

Purpose: Enable administrators to configure and manage service tools, ensuring proper functionality and access control.

Entry Condition: The administrator is authenticated and authorized to perform service tool configuration tasks.

Main Event Flow:

1. The administrator accesses the service tool management section of the system.
2. The administrator configures the necessary service tool parameters.
3. The administrator sets access permissions to define which users or groups can use the service tool.
4. The administrator tests the service tool execution to validate the configuration and ensure it operates correctly.
5. Upon successful testing, the administrator publishes the changes to make the service tool available for use.

Exit Condition: The service tool is successfully configured, tested, and ready for use by authorized users.

Exception Flow:

- If a configuration error occurs, the system notifies the administrator and provides details to assist in correcting the issue.
- If the test execution fails, the system notifies the administrator and logs the error for troubleshooting.

Special Requirements:

- Version control must be implemented to track and manage changes to the service tool configuration.
- Updates to the service tool configuration must include corresponding updates to the system documentation.

2.4 User Stories

US1: User Registration

As an anonymous user, I want to request access to the platform so that I can utilize its features.

Acceptance Criteria:

- The system provides a registration page for new users.
- Users can submit their email and company details.
- The system validates the email format and checks for duplicates.
- Upon approval, the system sends a registration invitation; if rejected, a notification with the reason is sent.

Relation to UC1 (User Registration):
This user story directly complements UC1 by addressing the system's functionality to handle user registration requests. Together, they provide a detailed flow for requesting, validating, and granting or denying access to the platform.

US2: Service Tool Access

As an authenticated user with service tool access, I want to execute service tools *so that I can process data and obtain results.*

Acceptance Criteria:

- Users can select from available service tools.
- Users can upload Excel input files for processing.

- The system validates the file format and size (up to 10MB).
- The system processes the data within a 5-minute timeout.
- Generated results, including PLT graphs, are available for download.

Relation to Service Tool Execution Use Case (implied but not explicitly detailed):
This user story aligns with the use case for service tool execution by elaborating on the steps for selecting a service tool, uploading data, processing results, and accessing outputs.

US3: Result Analysis

As a user who has completed optimizations, I want to analyze and compare my results so that I can evaluate performance across different runs.

Acceptance Criteria:

- Users can access their optimization history.
- Users can view PLT graphs for specific optimization runs.
- Users can download results for offline analysis.
- Users can compare up to three optimization runs simultaneously.
- Results are stored and accessible for 30 days.

Relation to Historical Data Use Case (implied but not explicitly detailed):
This story complements the system's ability to provide historical data access and result comparison. Together, they outline how users retrieve and analyze historical optimization data.

US4: User Management

As an administrator, I want to manage user access and permissions so that I can control platform usage and maintain security.

Acceptance Criteria:

- Administrators can access the user management section.
- Administrators can review and approve or deny access requests.
- Administrators can assign or modify user permissions and service tool access.
- User management actions are recorded in audit logs.
- Email notifications are sent to users upon changes to their access or permissions.

Relation to Admin Controls in UC1 (Admin Approves Access) and other user management use cases:
This story aligns with UC1 by emphasizing admin responsibilities for approving access requests

and complements other use cases that involve managing roles, permissions, and activity monitoring.

<u>US5: Service Tool Management</u>

As an administrator, I want to configure and manage service tools so that users can execute them effectively.

Acceptance Criteria:

- Administrators can access the service tool management section.
- Administrators can configure service tool parameters and set access permissions.
- Administrators can test service tool execution to ensure proper functionality.
- Successful configurations can be published for user access.
- Version control is implemented to track changes.
- Documentation is updated to reflect configuration changes.

Relation to Service Tool Configuration Use Case (implied but not explicitly detailed):
This story complements the use case for service tool configuration by detailing the administrator's role in setting parameters, testing service tools, and ensuring proper documentation. Together, they provide a complete process for effective service tool management.

2.5 System Boundaries and Constraints

1. **Technical Boundaries**:
    - Web-based platform only
    - Excel file input only
    - PLT graph output format
    - REST API communication
2. **Functional Boundaries**:
    - Limited to configured service tools
    - Maximum file size restrictions
    - Processing time limits
    - Result storage duration
3. **Security Constraints**:
    - HTTPS required
    - JWT authentication
    - Role-based access
    - Data encryption
4. **Performance Constraints**:

- Maximum concurrent users
- API rate limits
- Response time SLAs
- Storage quotas

2.6 Non-functional Requirements

2.6.1 Security Requirements

[NFR1] Authentication Security

Requirement:
The system must ensure secure authentication processes to protect user accounts.

Performance Specifications:

1. **Password Encryption:** All passwords must be securely encrypted.
2. **Token Management:** Tokens must be securely generated and managed.
3. **Session Security:** Active sessions must be protected against hijacking.
4. **Access Control:** Only authorized users must access specific resources.

[NFR2] Data Protection

Requirement:
The system must protect data to ensure security and compliance.

Performance Specifications:

1. **Secure Transmission:** All data must be transmitted securely using encryption.
2. **Access Logging:** All access to data must be logged for auditing.
3. **Privacy Compliance:** The system must adhere to privacy regulations.

[NFR3] Communication Security

Requirement:
The system must secure all communications to prevent unauthorized access or tampering.

Performance Specifications:

1. **SSL/TLS Implementation:** All communications must use secure protocols like SSL/TLS.
2. **API Security:** APIs must be protected against unauthorized access and attacks.
3. **Input Validation:** Inputs must be validated to prevent injection attacks.
4. **Output Sanitization:** Outputs must be sanitized to prevent data leaks.

[NFR4] Security Monitoring

Requirement:
The system must actively monitor for security threats and vulnerabilities.

Performance Specifications:

1. **Audit Logging:** All system activities must be logged for auditing purposes.
2. **Vulnerability Scanning:** Regular scans must identify and address vulnerabilities.
3. **Security Updates:** The system must be updated regularly with security patches.

2.6.2 Performance Requirements

[NFR5] Concurrency

Requirement:
The system must handle multiple users and tasks simultaneously for seamless performance.

Performance Specifications:

1. **Multiple User Support:** Support at least 100 concurrent users without performance degradation. *(Subject to validation.)*
2. **Parallel Service Tool Execution:** Enable at least 10 simultaneous service tool executions with minimal resource contention. *(Estimate, pending testing.)*
3. **Load Balancing:** Dynamically distribute tasks across servers to optimize resource utilization and avoid bottlenecks.

Testing Criteria:

- Simulate 100 concurrent users and validate performance stability.
- Test at least 10 simultaneous service tool executions with acceptable latency.
- Ensure load balancing dynamically manages server resources.

[NFR6] Response Time

Requirement:
The system must meet specific response time objectives to provide a responsive and efficient user experience.

Performance Specifications:

1. **API Response SLAs:** APIs should respond within 1 second for 95% of requests under normal load. *(Assumption: Based on typical high-performance systems.)*
2. **Page Load Times:** Pages should load within 3 seconds for 90% of users on a standard broadband connection. *(Assumption: For systems with a moderate load, this is acceptable; can be adjusted based on product requirements.)*
3. **Service Tool Execution Time:** Service tool execution must complete:
    - Within 30 seconds for small datasets (less than 1MB).
    - Within 2 minutes for large datasets (up to 10MB) under normal system load. *(Estimate, requires validation through testing.)*

Testing Criteria:

- Measure API responses under normal load to confirm that 95% meet the 1-second threshold.
- Monitor page load times for 90% of users under standard broadband conditions.
- Benchmark service tool execution times with varying dataset sizes to ensure compliance under normal load conditions.

[NFR7] File Handling

Requirement:
The system must support efficient file handling to optimize storage and retrieval processes.

Performance Specifications:

1. **Efficient Storage:** Files must be stored in a way that minimizes disk space usage.
2. **Quick Retrieval:** Files must be retrievable within a reasonable time under normal load.
3. **Compression:** Files should be automatically compressed to save storage space.

[NFR8] Scalability

Requirement:
The system must scale effectively to handle increasing loads and distribute tasks efficiently.

Performance Specifications:

1. **Horizontal Scaling:** The system must support adding servers or resources to handle higher loads.
2. **Load Distribution:** Tasks must be dynamically distributed across resources to maintain performance.

2.6.3 Optimization Environment

(The optimization environment refers to the system configuration and infrastructure that supports the efficient, secure, and reliable execution of optimization algorithms. It ensures fast execution through optimized resource use, scales seamlessly to handle varying workloads, and maintains robust security with data protection and process isolation. Designed for high availability and fault tolerance, it adapts to diverse service tools and data requirements, serving as the backbone for executing service tools and processing production data.)

2.6.4 Optimization Algorithm Execution Environment

Runtime Environment:

Requirement:
The system must provide a dedicated and controlled environment for executing optimization algorithms to ensure isolation, compatibility, and efficient resource utilization.

Specifications:

1. **Dedicated Execution:** Each algorithm instance runs in a dedicated Docker container to ensure isolation and consistent execution.
2. **Python 3.9+ Runtime:** All algorithms execute in a Python environment with pre-installed dependencies to ensure compatibility and reliability.
3. **Resource Limits:** Containers are allocated specific CPU and memory resources to prevent overuse and maintain balanced system performance.

Resource management:

Requirement:
The system must efficiently allocate, monitor, and scale computational resources to support reliable and high-performance optimization tasks.

Specifications:

1. **Dynamic Resource Allocation:** Resources are dynamically assigned based on task complexity and current system load.
2. **Concurrent Execution:** Multiple optimization algorithms execute simultaneously, enabling parallel processing without performance degradation.
3. **Resource Monitoring and Scaling:** Real-time resource tracking allows the system to scale horizontally as needed to meet demand.

Integration Points:

Requirement:
The system must facilitate seamless interaction between the optimization environment and other system components to ensure usability and reliability.

Specifications:

1. **REST API Interface:** Provides a standard API for programmatically invoking optimization algorithms.
2. **File System Integration:** Supports secure reading of input data (e.g., Excel files) and storage of output results.
3. **Result Retrieval:** Ensures users can easily access and download algorithm outputs.
4. **Error Handling and Recovery:** Detects and recovers from execution errors to maintain system stability.

Security Measures:

Requirement:
The system must ensure data protection and maintain integrity during the execution of optimization algorithms.

Specifications:

1. **Access Control:** Restricts algorithm execution to authorized users and roles.
2. **Data Encryption:** Encrypts data during transmission to safeguard sensitive information.
3. **Secure File Handling:** Ensures input and output files are securely managed to prevent unauthorized access or tampering.

Monitoring:

Requirement:
The system must track and log performance and execution metrics to optimize operations and support issue identification.

Specifications:

1. **Real-time Metrics:** Provides live data on execution times, resource usage, and success rates.
2. **Performance Profiling:** Identifies bottlenecks or inefficiencies in algorithm execution.
3. **Error Logging:** Records errors to facilitate troubleshooting and continuous improvement.

2.7 Technical Stack

2.7.1 Frontend Technologies

- React 18
- Vite
- Tailwind CSS
- Chart.js
- Axios

2.7.2 Backend Technologies

- Spring Boot 3
- Spring Security
- JWT
- PostgreSQL
- Test Containers

2.7.3 Service Tool Service Technologies

- Python 3.9+
- FastAPI
- Matplotlib
- NumPy/Pandas

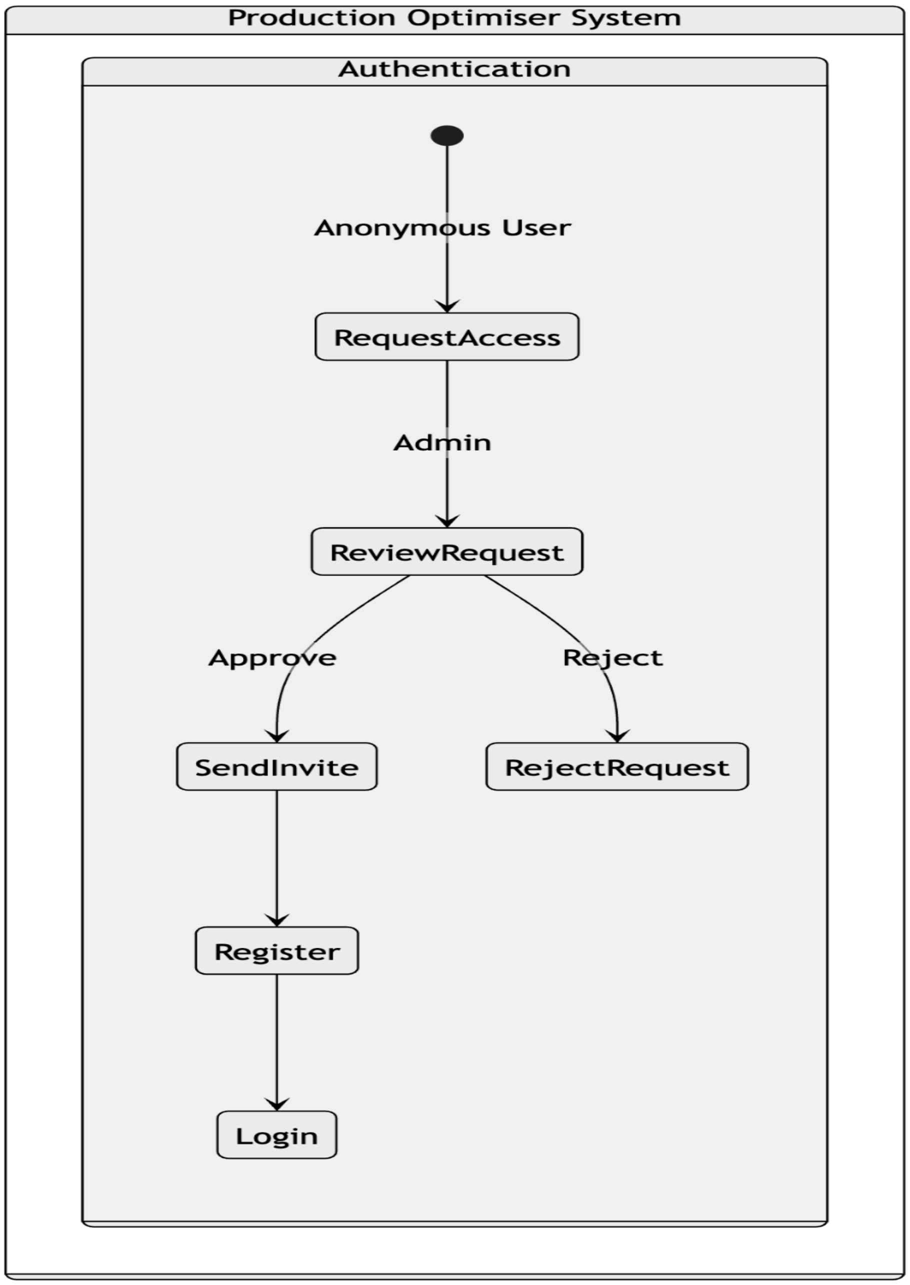
2.7.4 Infrastructure

- Docker

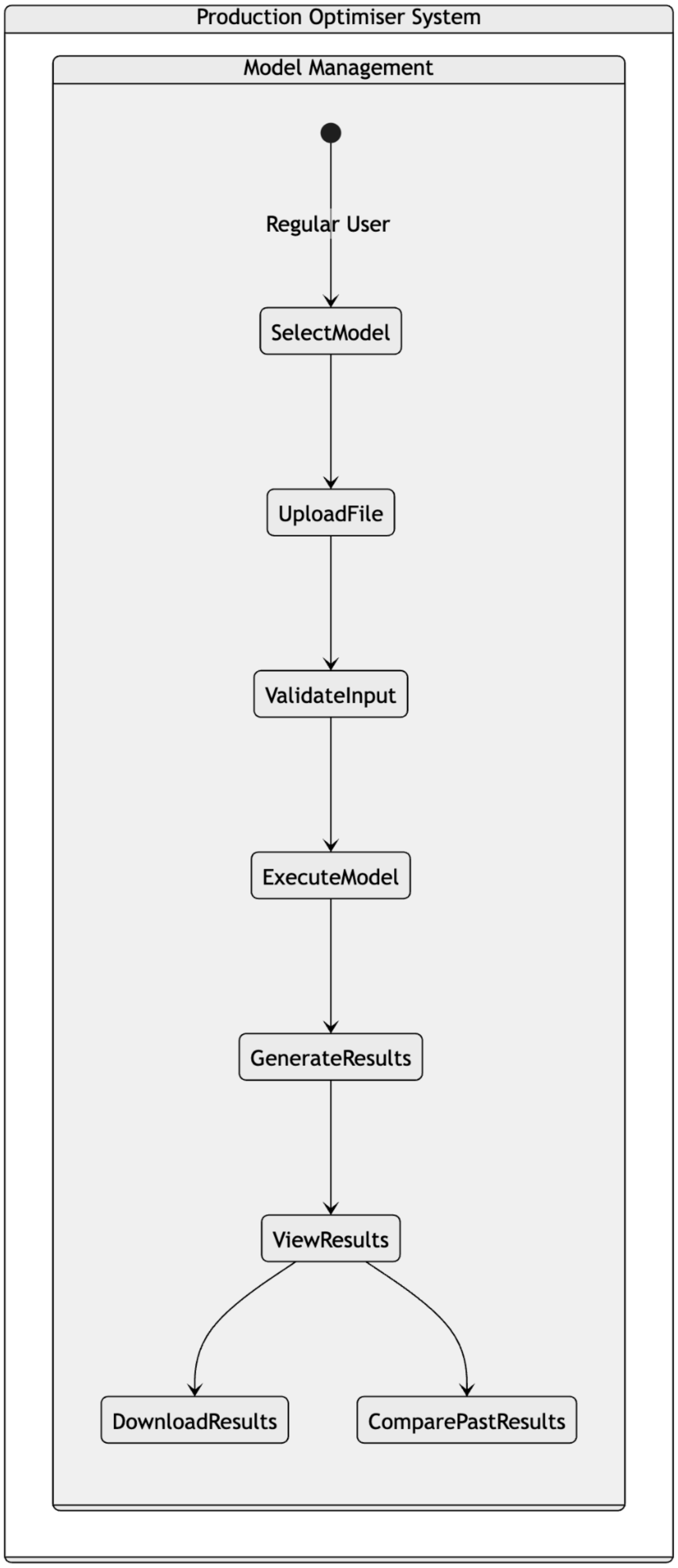
- Github
- GitHub Actions

2.8 Use Case Diagrams

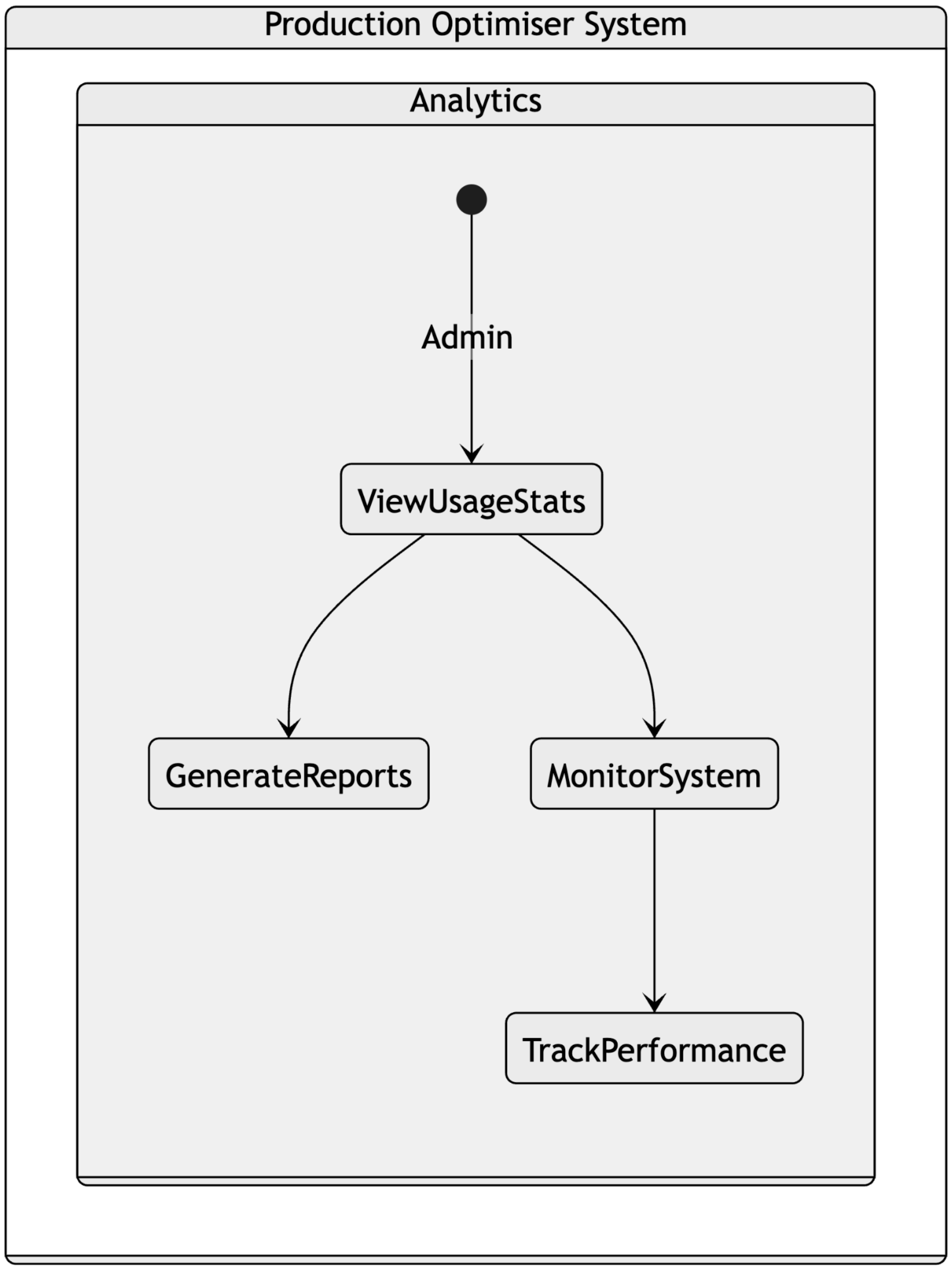
2.8.1 User Authentication Diagram



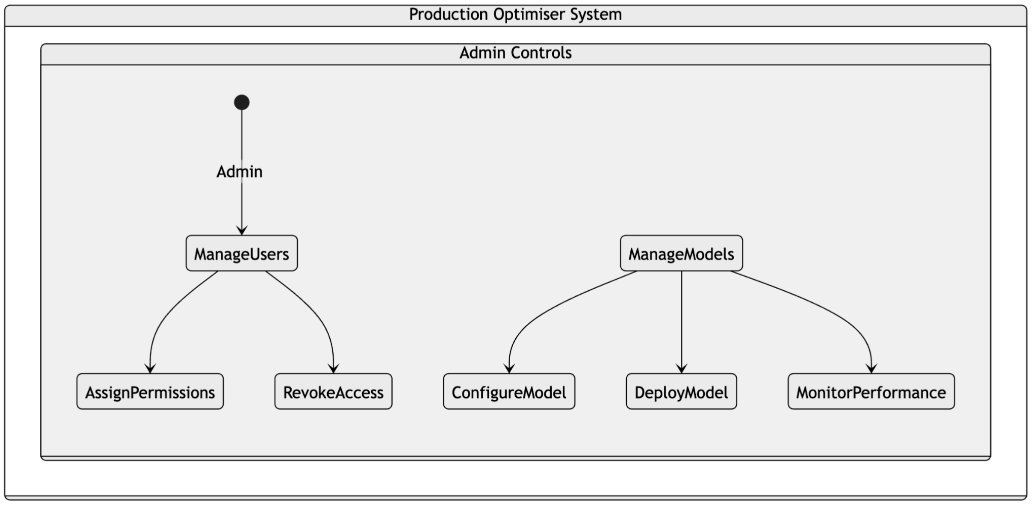
2.8.2 Service Tool Management Diagram



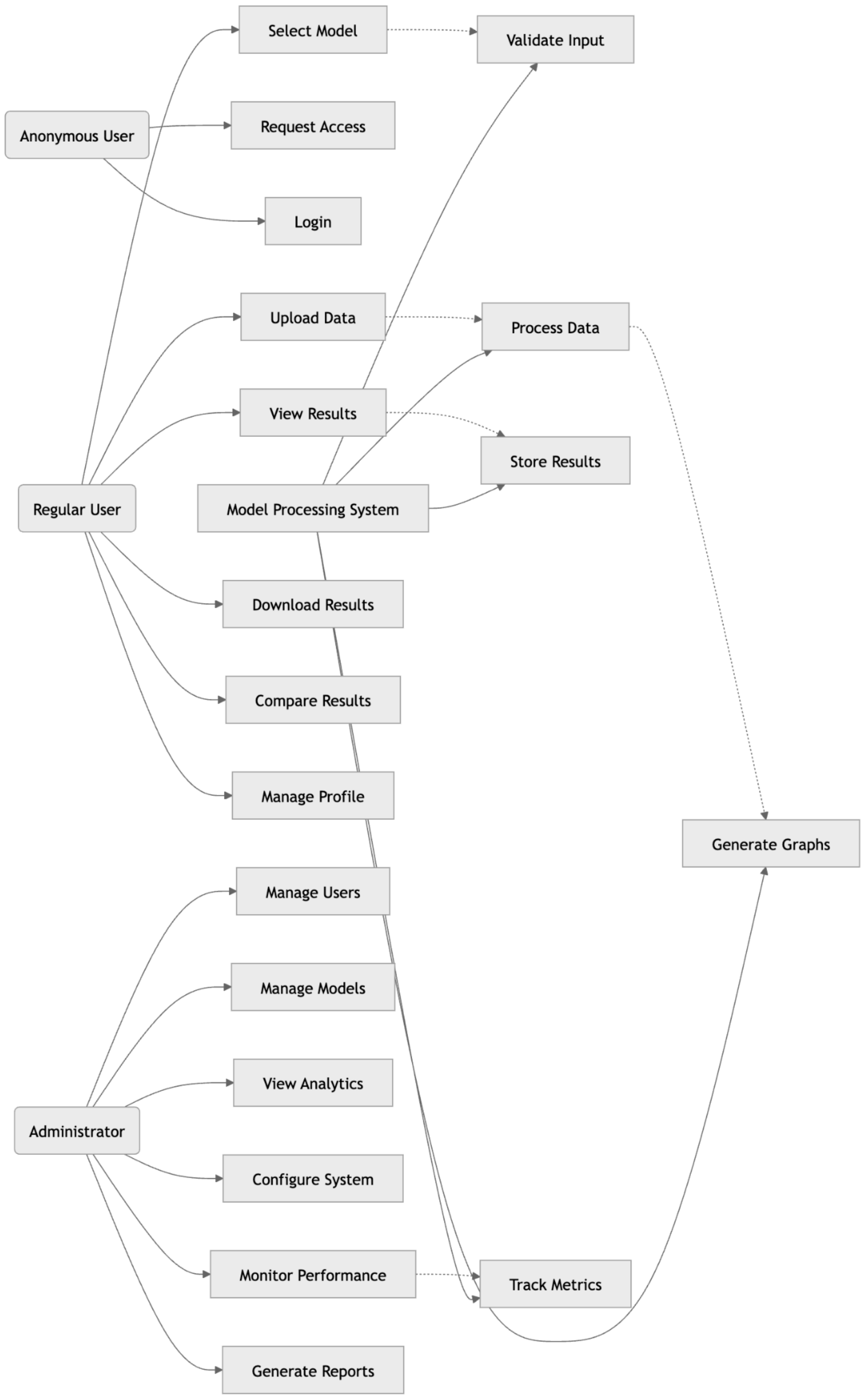
2.8.3 Analytics Diagram



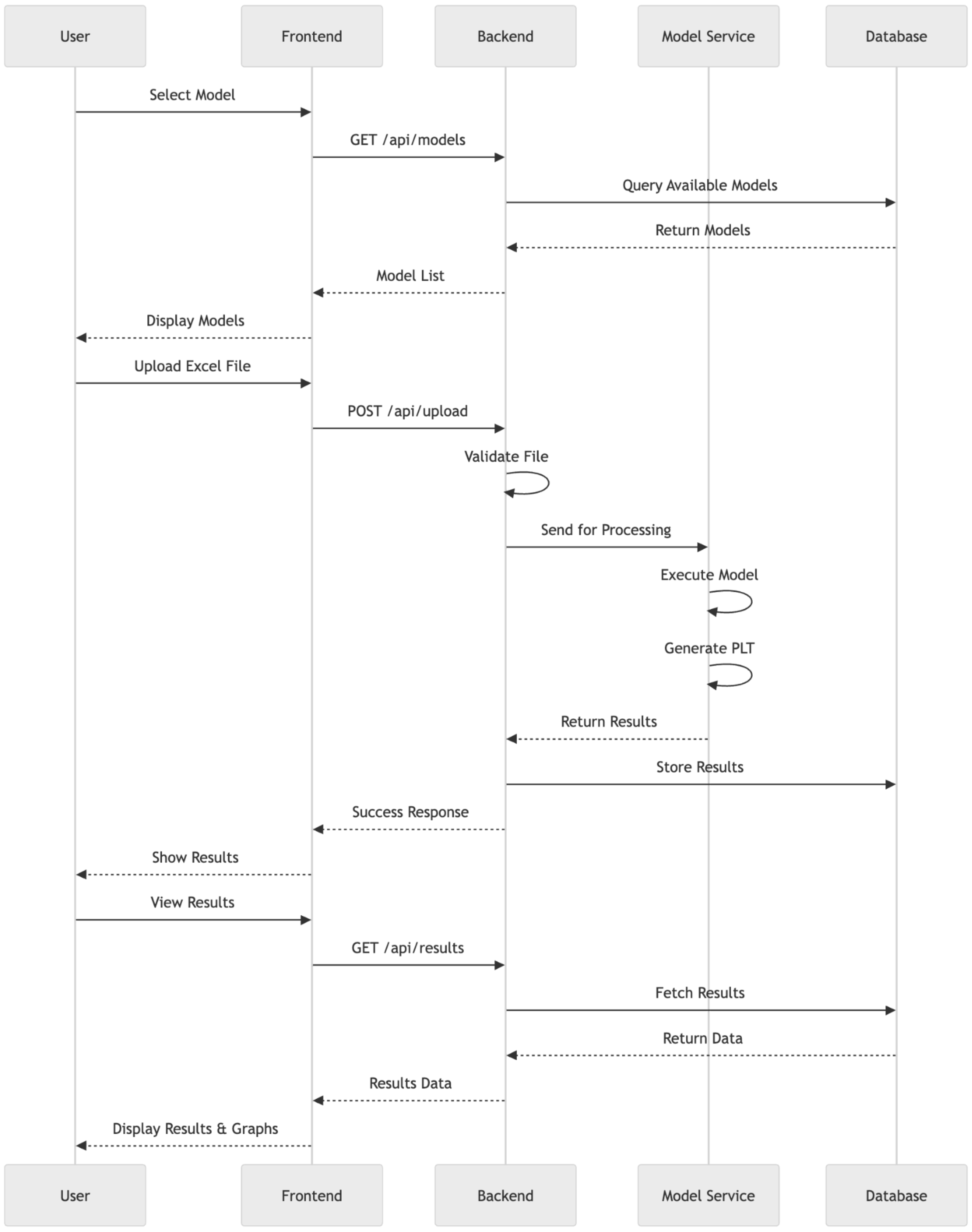
2.8.4 Admin Controls Diagram



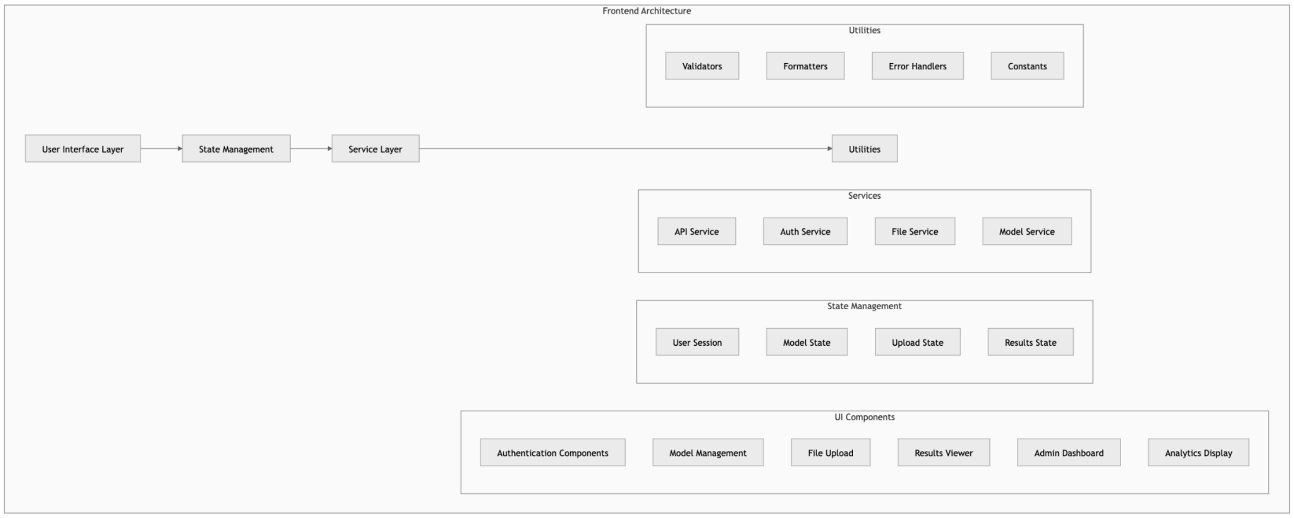
2.8.5 Actor-Relationship Use Case Diagram



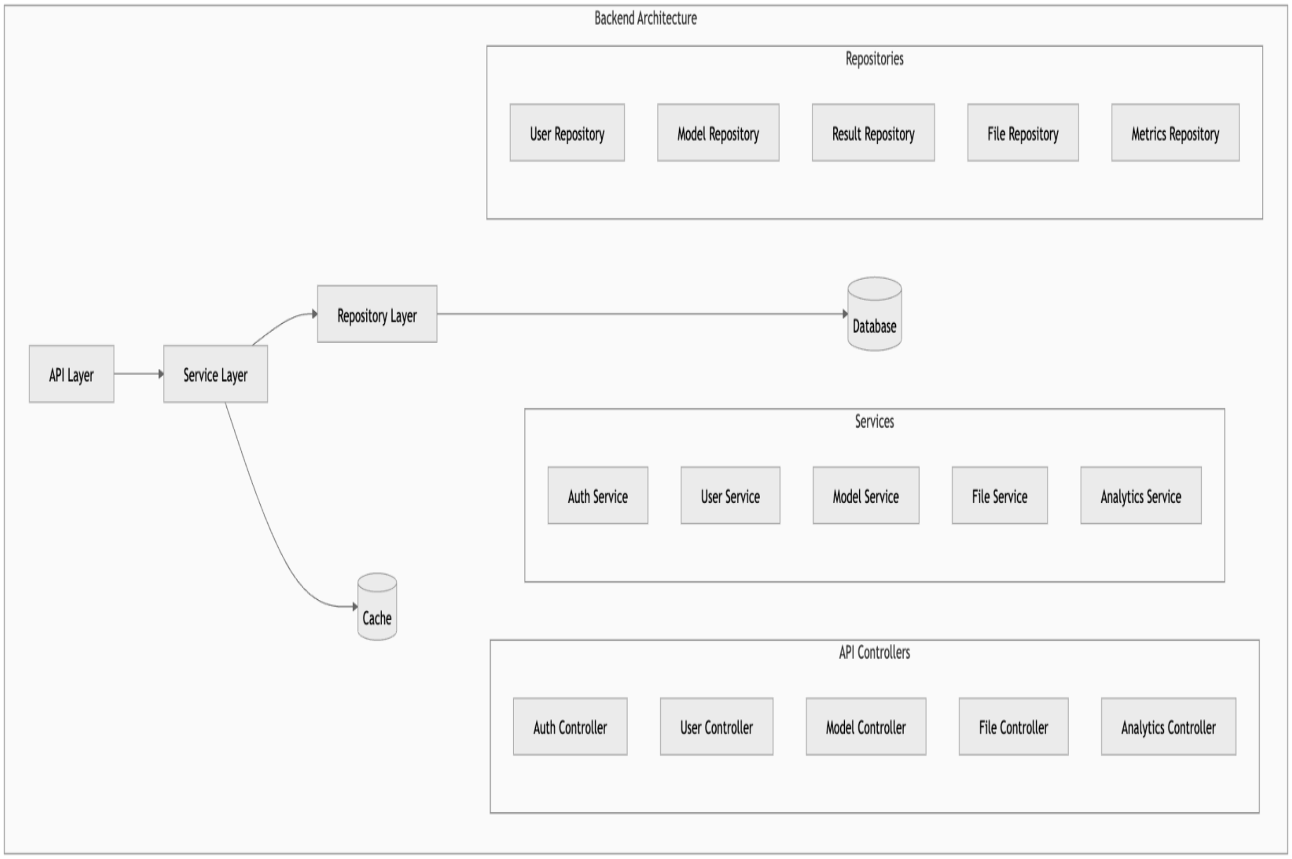
2.8.6 Service Tool Execution - Sequence Diagram



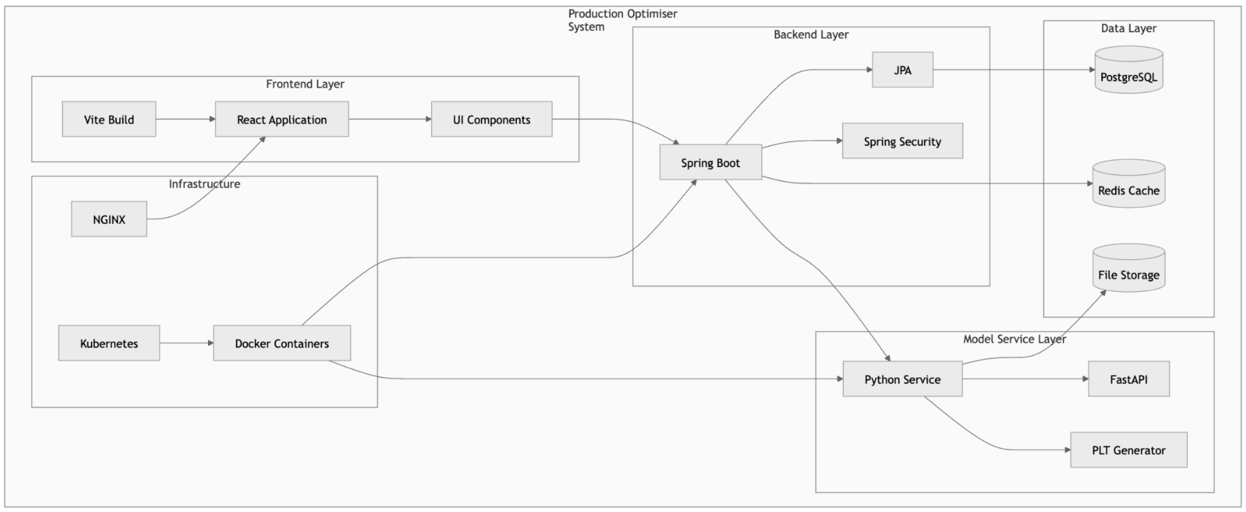
2.8.7 Frontend - Architecture



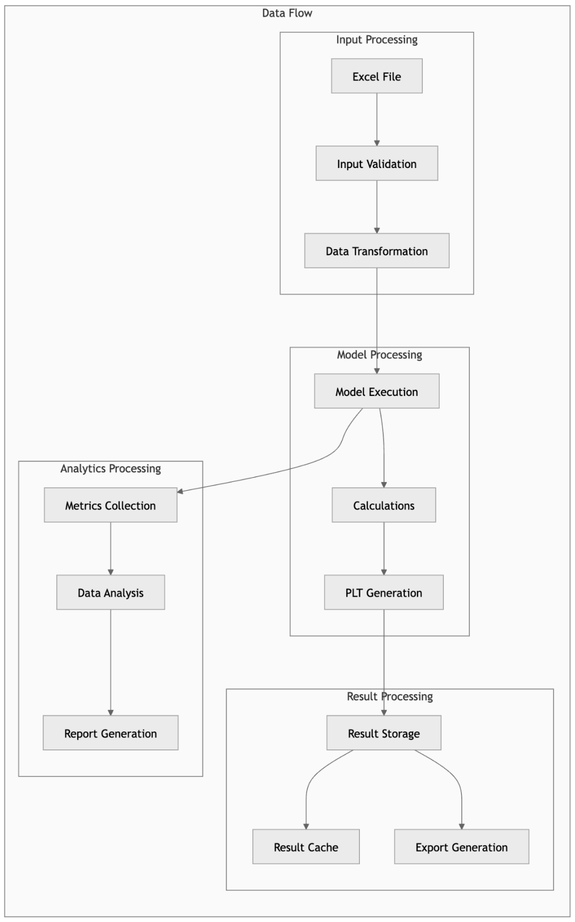
2.8.8 Backend - Architecture



2.8.9 System Integration Architecture



2.8.10 Process Flow



Process Flow Explanation

1. Input Processing

The initial phase, where user-provided data (typically in Excel file format) is prepared for optimization.

- **Excel File Upload:**
  Users upload their production data in Excel format, serving as the input for service tools.
- **Input Validation:**
  The system verifies the correctness of the uploaded file, checking the format, required fields, and data consistency. If errors are detected, users are prompted to correct them.
- **Data Transformation:**
  Validated data is converted into a format suitable for service tools, such as transforming Excel rows into structured data objects.

2. Service Tool Processing

This phase involves running service tools on the prepared data.

- **Service Tool Execution:**
  The system executes the selected service tool using the transformed data as input.
- **Calculations:**
  The service tool performs computations to generate optimal results based on user-defined parameters and constraints.
- **PLT Generation:**
  Results are visualized using PLT graphs (via Matplotlib), providing users with clear and interpretable visual representations of the outcomes.

3. Analytics Processing

This phase focuses on monitoring and analyzing the performance of the optimization process.

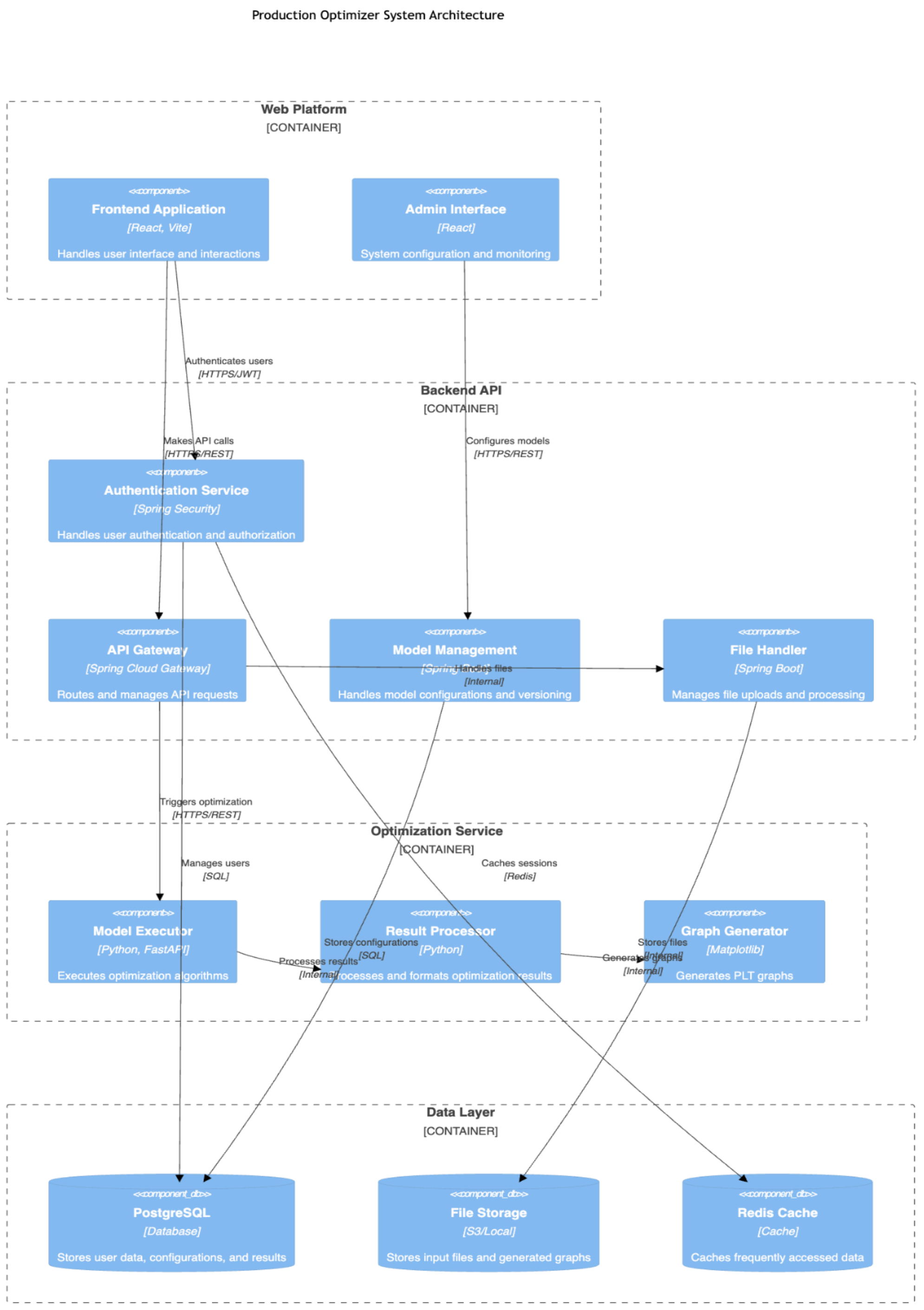
- **Metrics Collection:**
  The platform tracks performance metrics, such as execution time, success rates, and resource usage.
- **Data Analysis:**
  Metrics are analyzed to provide insights into system and service tool performance.
- **Report Generation:**
  The system generates comprehensive reports for users and administrators, highlighting performance trends and key outcomes.

4. Result Processing

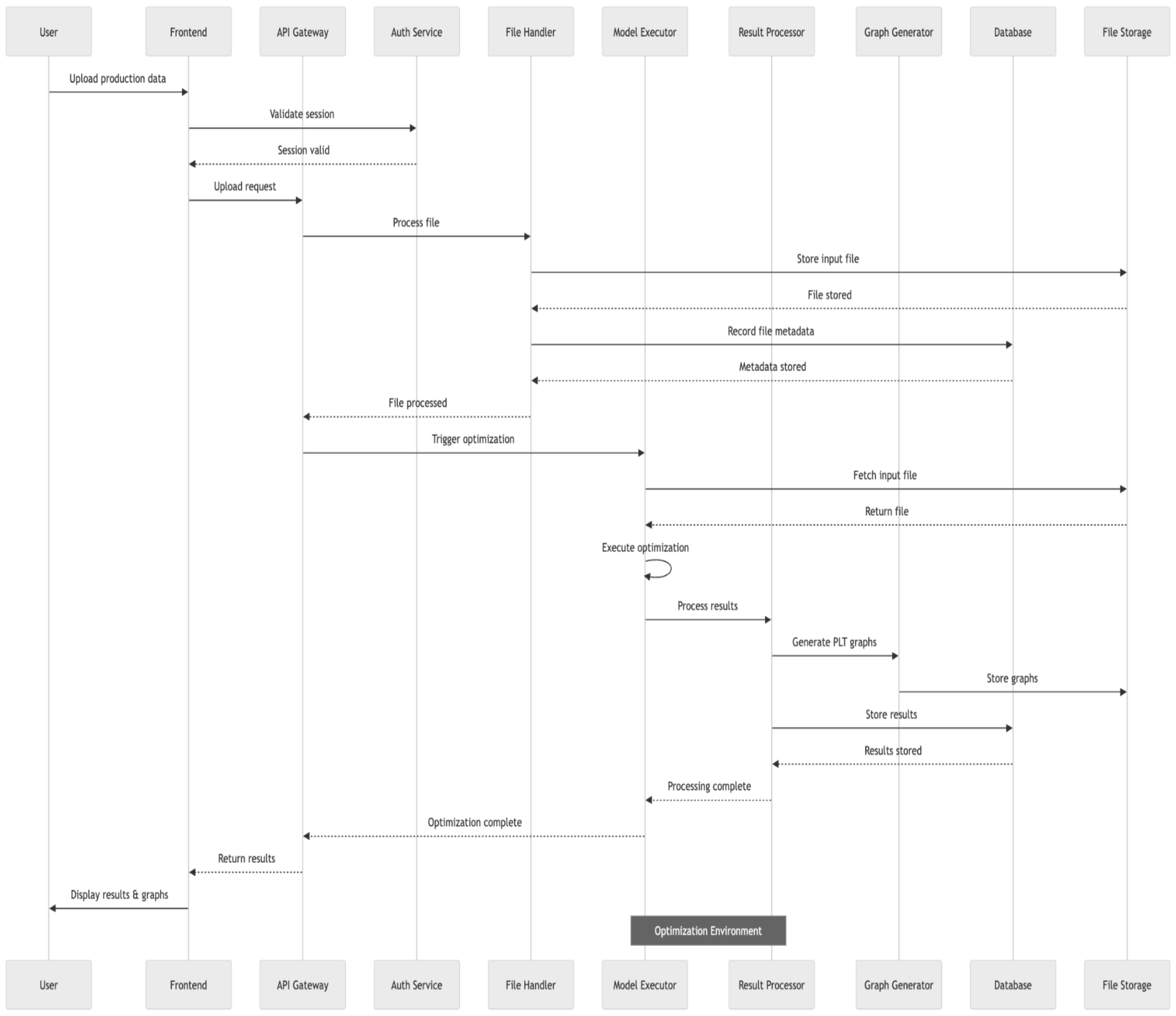
The final phase involves storing and presenting the results to users.

- **Result Storage:**
  Results, including raw outputs, graphs, and metadata, are saved for future reference.
- **Export Generation:**
  Users can export results in various formats (e.g., PDF, Excel, CSV) for offline analysis.

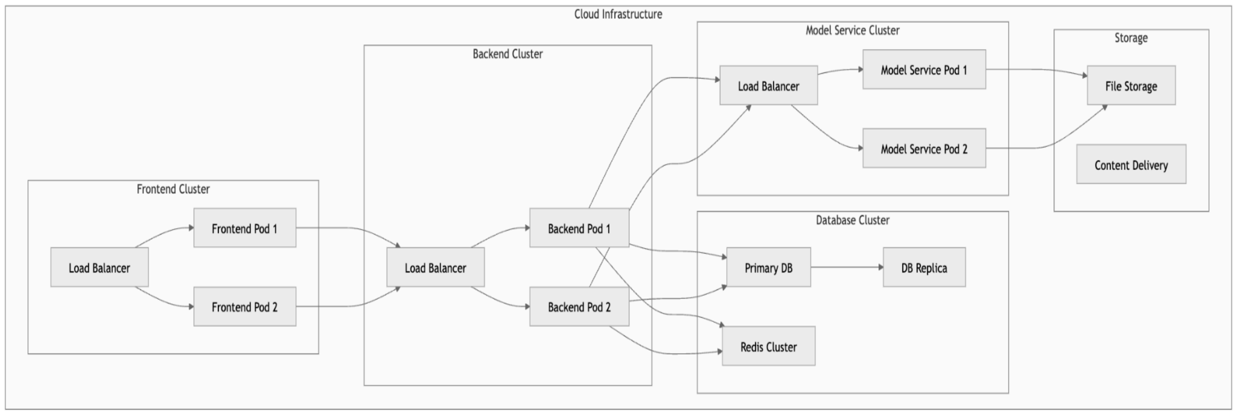
2.8.11 Systems Architecture Views: C4 Component Diagram



2.8.12 Optimization Process Sequence Diagram



2.8.13 Example Deployment Architecture



Chapter 3: Data Requirements

3.1.1 Data Service Tool

- Authentication Data Schema
  - Username/email (unique identifier)
  - BCrypt hashed password storage
  - JWT token management
  - Password reset tokens
  - Login attempt tracking
  - Session information
- User Data Schema
  - User profile information
  - Authentication credentials
  - Access permissions
  - Activity history
- Service Tool Data Schema
  - Service Tool configurations
  - Input parameters
  - Output specifications
  - Version history
- Results Data Schema
  - Optimization results
  - PLT graph data
  - Performance metrics
  - Historical comparisons

3.1.2 Data Validation Rules

- Input File Validation
  - File format specifications
  - Required fields
  - Data type constraints
  - Value range restrictions
- User Input Validation
  - Form field requirements
  - Input sanitization rules
  - Cross-field validations
  - Business rule validations

3.1.3 Data Retention Policies

- User Data Retention
  - Active account data

    - Deactivated account handling
    - Audit log retention
    - Backup retention periods
- Results Data Retention
    - Optimization results storage duration
    - Archive policies
    - Data cleanup procedures
    - Backup requirements

3.2 Integration Requirements

3.2.1 External System Interfaces

- Authentication Systems
    - JWT token management and refresh strategies
    - BCrypt password hashing implementation
    - Session management and timeout policies
    - Token blacklisting mechanisms
- Storage Systems
    - File storage integration
    - Database connections
    - Backup systems
    - Archive management

3.2.2 API Specifications

- REST API Documentation
    - Endpoint definitions
    - Request/response formats
    - Authentication requirements
    - Rate limiting policies
- Internal API Requirements
    - Service communication protocols
    - Error handling
    - Retry policies
    - Circuit breaker patterns

3.3 Compliance Requirements

3.3.1 Security Standards

- Authentication Security
  - JWT token configuration
    - Token expiration time: 15 minutes
    - Refresh token expiration: 7 days
    - Signature algorithm: HS256
    - Token payload structure
    - Token storage guidelines
  - Password Security
    - BCrypt work factor: 12+
    - Password complexity requirements
    - Password change policies
    - Password reset procedures
    - Maximum failed attempt limits
  - Session Security
    - Token validation requirements
    - Token refresh mechanisms
    - Concurrent session handling
    - Session termination procedures
    - Token revocation mechanisms
- Data Protection
  - Encryption requirements
  - Key management
  - Access control policies
  - Security audit procedures
- Compliance Frameworks
  - Industry-specific regulations
  - Data privacy requirements
  - Security certifications
  - Audit requirements

3.3.2 Performance Standards

- Response Time Requirements
  - Page load time limits
  - API response times
  - Service tool execution timeouts
  - Background job limits
- Availability Requirements
  - Uptime guarantees

    - Maintenance windows
    - Failover requirements
    - Disaster recovery

3.4 Testing Requirements

3.4.1 Test Types

- Unit Testing
    - Component test coverage
    - Integration test scope
    - Performance test criteria
    - Security test requirements
- User Acceptance Testing
    - Test scenarios
    - Acceptance criteria
    - Test data requirements
    - Sign-off procedures

3.4.2 Quality Metrics

- Code Quality
    - Code coverage requirements
    - Static analysis rules
    - Review procedures
    - Documentation standards
- Performance Metrics
    - Load testing criteria
    - Stress testing requirements
    - Scalability benchmarks
    - Monitoring requirements

Chapter 4: Implementation Guidelines

4.1 Development Standards

4.1.1 Coding Standards

Frontend Development

- **React Component Guidelines:**
  Components must be reusable and modular, with clearly defined props and states. Adhere to established React best practices for code readability and maintainability.
- **State Management Patterns:**
  Use a centralized state management library (e.g., Redux or Context API) to handle complex application states. Avoid unnecessary state duplication to minimize errors and improve maintainability.
- **UI/UX Consistency Rules:**
  Ensure a uniform design by adhering to the platform's design system or style guide. This includes maintaining consistent use of colors, fonts, button styles, spacing, and responsiveness across all components.
- **Performance Optimization:**
  Optimize component rendering to prevent unnecessary re-renders. Utilize lazy loading for heavy assets and prioritize quick page load times to enhance performance.

Backend Development

- **API Design Patterns:**
  Follow RESTful principles to design clean, predictable, and maintainable APIs. Ensure proper versioning and use clear endpoint naming conventions.
- **Database Access Patterns:**
  Implement secure and efficient database queries using Object-Relational Mapping (ORM) tools like SQLAlchemy or Sequelize. Avoid redundant queries and ensure proper indexing to optimize performance.
- **Security Implementation:**
  Apply secure development practices, including input validation, data encryption, and token-based authentication (e.g., JWT). Sanitize all data to prevent injection attacks.
- **Error Handling Standards:**
  Handle exceptions systematically to provide informative and user-friendly error responses while logging detailed errors for developers to facilitate debugging and improvement.

4.1.2 Documentation Requirements

Technical Documentation

- **API Documentation:**
  Detailed information about each API endpoint, including input parameters, expected responses, and error codes.
- **Architecture Documentation:**
  Diagrams and descriptions of the system architecture, data flow, and integration points to ensure a clear understanding of the platform's structure.
- **Development Setup Guides:**
  Step-by-step instructions for setting up the development environment, including required tools, dependencies, and configurations.
- **Deployment Procedures:**
  Comprehensive documentation of deployment steps for staging and production environments, ensuring smooth and consistent releases.

User Documentation

- **User Manuals:**
  Clear, user-friendly instructions for end-users on performing key tasks such as uploading data, running service tools, and downloading results.
- **Administrator Guides:**
  Detailed instructions for managing users, monitoring system activity, and performing maintenance tasks.
- **API Integration Guides:**
  Guidance for third-party developers on integrating their applications with the platform via APIs, with examples and best practices.
- **Troubleshooting Guides:**
  Solutions for common issues and errors encountered by users or administrators, enabling quick resolution.

4.2 Deployment Requirements

4.2.1 Environment Specifications

Development Environment

- **Setup Requirements:**
  Install required tools, including Node.js, Python, and Docker, and configure local environments for consistent development workflows.
- **Tool Configurations:**
  Specify and enforce the use of IDE plugins, linters, formatters, and other tools to maintain code quality and consistency.
- **Testing Frameworks:**
  Use automated testing frameworks like Jest for frontend testing and Pytest for backend testing, covering unit, integration, and regression tests.
- **CI/CD Integration:**
  Implement pipelines for continuous testing, building, and deployment to ensure efficient and reliable release processes.

Production Environment

- **Infrastructure Requirements:**
  Utilize cloud infrastructure, including virtual machines, containers, and storage solutions, to host the system.
- **Scaling Configurations:**
  Enable horizontal scaling with load balancers to distribute requests and handle increased demand.
- **Monitoring Setup:**
  Deploy monitoring tools like Prometheus and Grafana to track system health and performance in real time.
- **Backup Procedures:**
  Implement automated daily backups for critical data and configurations to ensure data integrity and disaster recovery.

4.2.2 Release Management

Version Control

- **Branching Strategy:**
  Follow GitFlow or a similar strategy to manage feature development, bug fixes, and releases.
- **Release Tagging:**
  Adopt semantic versioning (e.g., v1.0.0) for clear and consistent version identification.
- **Change Management:**
  Maintain a changelog to document all updates, with mandatory reviews and approvals for changes before release.
- **Rollback Procedures:**
  Prepare rollback scripts or processes to revert to a previous stable version in case of deployment failures.

Deployment Procedures

- **Deployment Checklist:**
  Ensure all pre-deployment tasks are completed, including passing tests, code reviews, and configuring environments.
- **Validation Steps:**
  Conduct smoke tests and sanity checks post-deployment to confirm expected system functionality.
- **Rollback Procedures:**
  Document detailed steps for reverting changes in case of critical issues during or after deployment.
- **Post-Deployment Verification:**
  Monitor system performance after deployment, promptly addressing any issues reported by users.