Team & Project Vision and Plan Presentation (For Meeting Logs, Sprint Hours and Project Overview)

Requirement Analysis

Functional Requirements

Based on the customer meeting held on October 18 at 15:00 (Danko [Product Owner]), the system must service to two user roles:

Regular Users (Customer) and **Admin Users**. The following functionalities are essential:

For Regular Users:

1. **User Authentication:**
   - Users can **sign in** to the platform using email and password.
   - **Invite required** for account creation.
2. **Model Selection & Usage:**
   - Users can **select from different available models** for processing.
   - Users can communicate with models through a standalone API not tied to the FE of the application
3. **File Upload & Model Input:**
   - Users can **upload files** to be used as input for the selected models.
4. **Past Optimization Tracking:**
   - Users can view their **past optimizations**, including both input files and the corresponding output.
5. **Result Comparison:**
   - Users can **compare the result files** from previous optimizations.
6. **Download Functionality:**
   - Users can **download model outputs**.
   - Users can also **download graphs or images** generated during the processing, especially visual results like PLT graphs.

For Admin Users:

1. **User Management:**
   - Admins can **invite users** to the platform and **revoke access** if needed.
2. **Model Management:**

- Admins can **add or remove models from the system**, allowing them to:
    - Connect new models to the platform's frontend and backend services
    - Configure API endpoints for model integration
    - Disable or deactivate models that are no longer needed
- Admins can **assign specific models to particular customers** for custom access.

3. **Full Regular User Access:**
    - Admins can perform all the operations available to regular users.

4. **User & Model Statistics:**
    - Admins can view **statistics for each user** (usage frequency, model selection) and **statistics per model** (performance, usage).

Non-Functional Requirements

1. **Scalability:**
    - The system should be **horizontally scalable** meaning that each service/module (e.g., model processing, user management, frontend, backend) can scale independently to handle increased load.
    - Platform components should be **independent**
2. **Communication Protocols**
    - All communications between services and clients should use **secure protocols** such as **HTTPS** to ensure data integrity and security
3. **Data formats:**
    - Input data will be provided in an Excel table
    - Output of model is a PLT graph
4. **Movable:**
    - Platform components should be movable between different hosting options
5. **Security:**
    - The database should store **customer information securely**, implying the need for data encryption, secure user authentication, and authorization protocols.

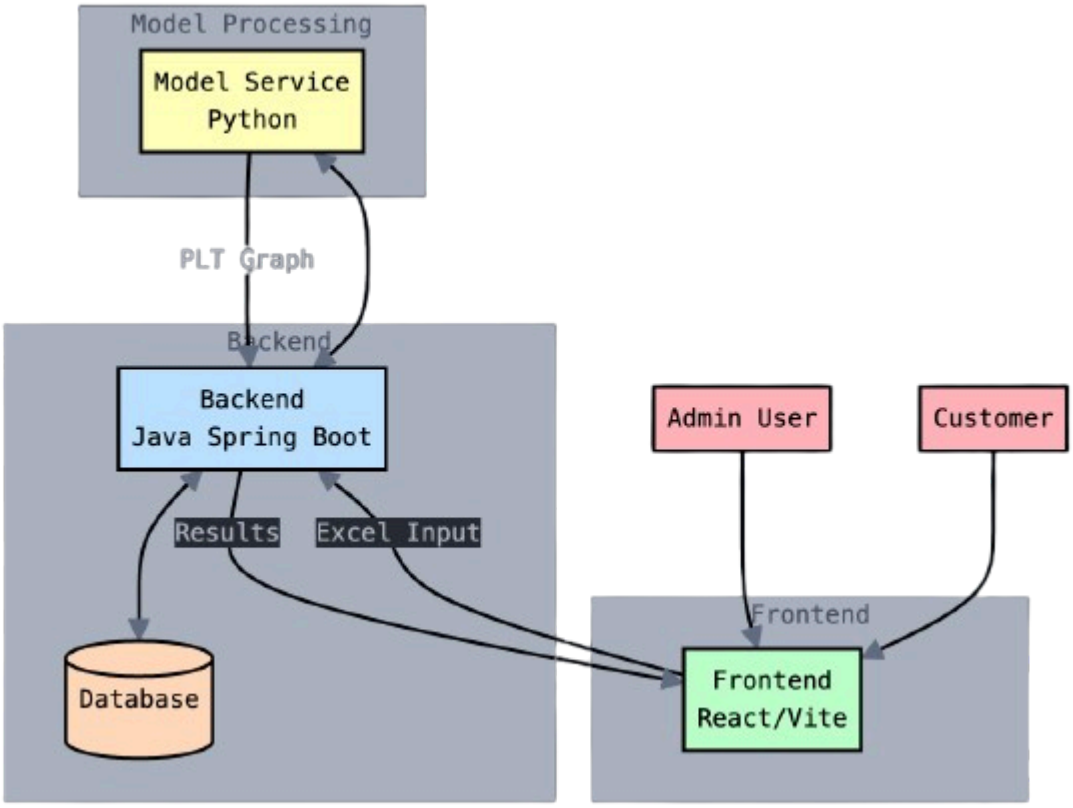
Additional Requirements:

The system should **collect data about model performance** for future analysis, including:

- **Core Performance Metrics**
  - Model execution time
  - Prediction accuracy rates
  - Resource utilization (CPU, memory, GPU)
- **User Interaction Data**
  - Usage patterns and frequency
  - Query types and volumes
  - Response times and success rates

High-Level System Design Analysis

Architecture Overview:

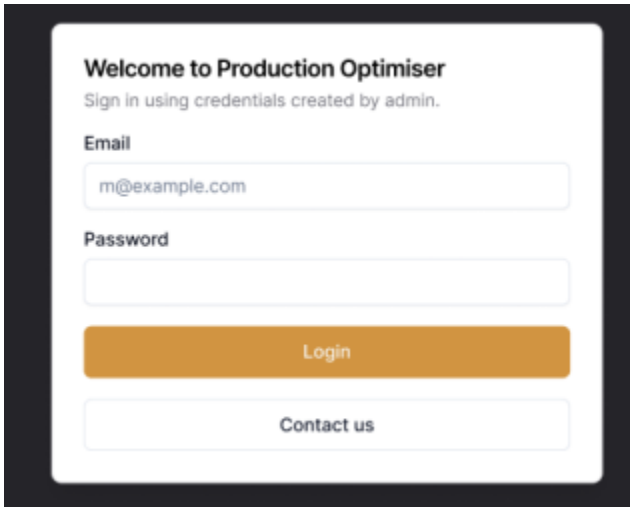


The architecture consists of three primary components:

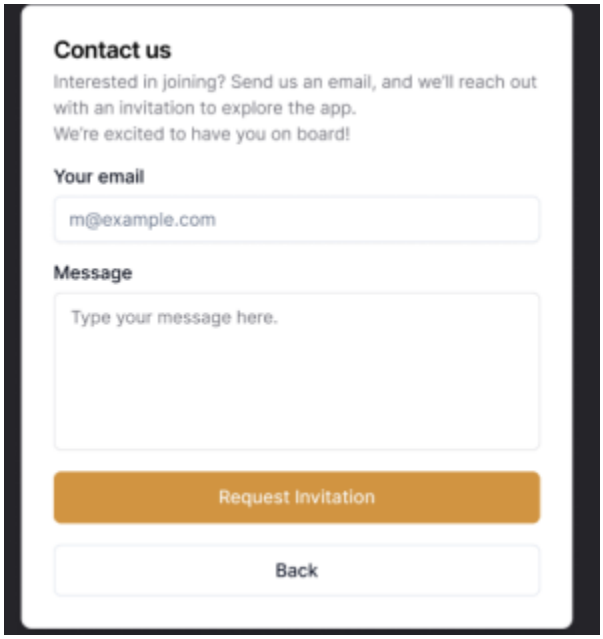
1. **Frontend (React/Vite):**
    - **Core Interface Components**
        - Handles user interfaces for both Admin and Regular users
        - Ensures responsive and intuitive design across devices
        - Communicates with backend for all user interactions

Figma Mockups:

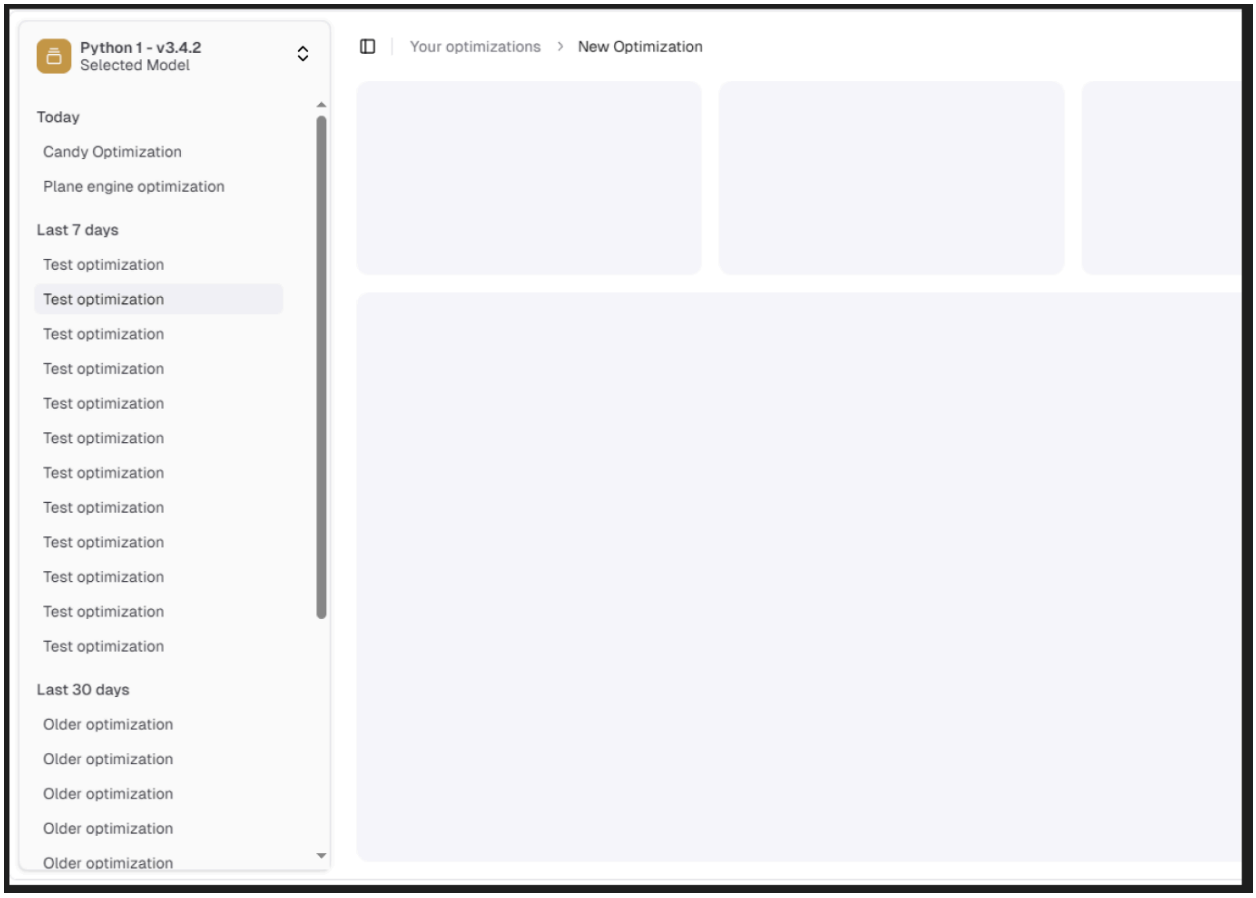
1) **Login Page:**



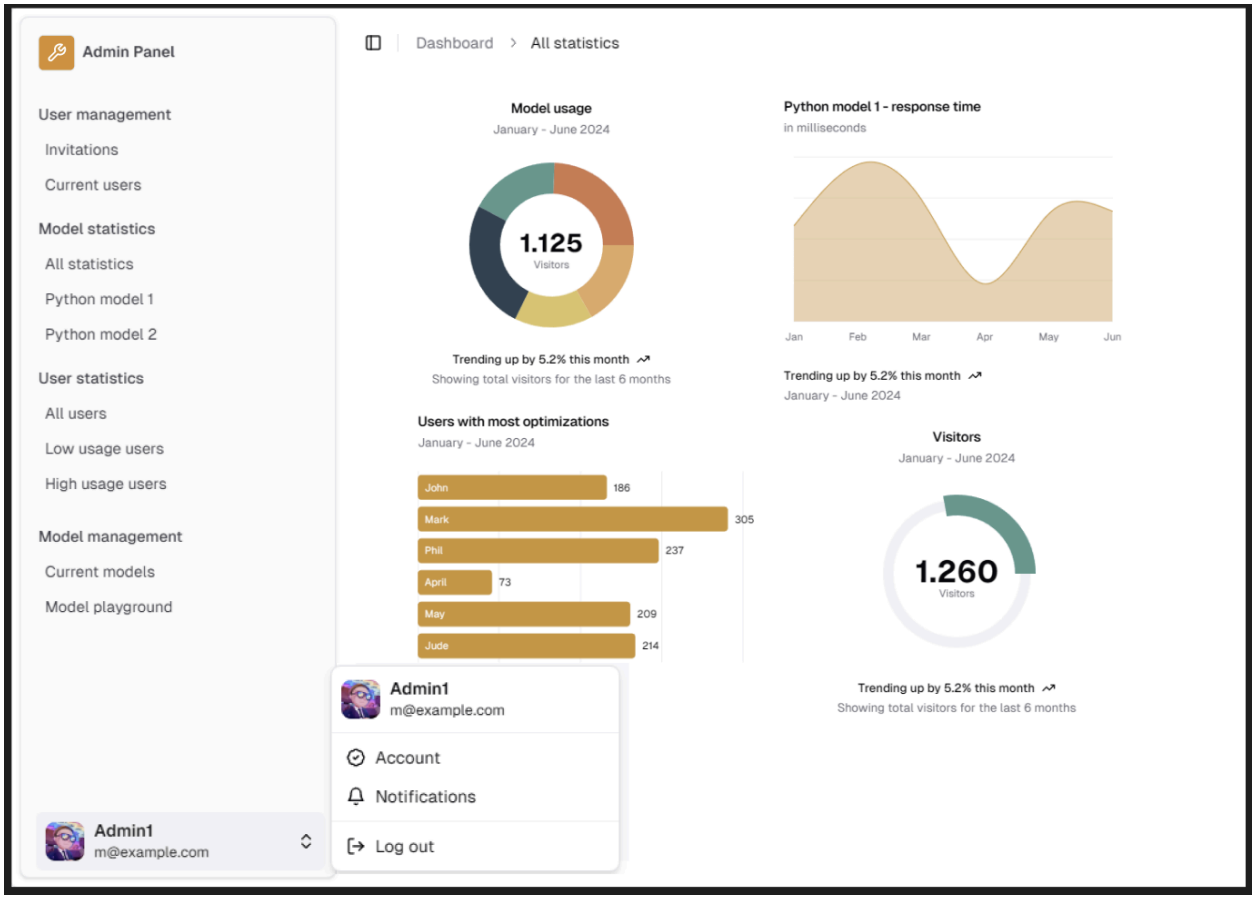
2) **Registration/permission inquiry form**



3) Optimisations and history view



4) Model metrics and statistics



- **Regular User Features**
  - Model selection and configuration interface
  - File upload management
  - Results visualization
  - Download capabilities:
    - Model outputs
    - Generated graphs and images
    - Analysis reports
- **Admin Dashboard**
  - User management interface

        - Model configuration and deployment controls
        - Analytics display:
            - Per-user statistics
            - Per-model performance metrics
            - System usage insights
    - **Data Communication**
        - Handles API interactions with backend
        - Manages data transfer for file uploads
        - Real-time status updates and notifications
2. **Backend (Java Spring Boot):**
    - Serves as the central service, linking the frontend to both the database and the model processing system.
    - Handles user authentication, model selection, and the management of user-uploaded files along with optimization results.
    - Fetches and delivers processed outputs to the frontend, including any graphs or images produced by the model.
    - Saves optimization results and historical data in the database for future reference.
    - Oversees user session management and enforces access control measures for secure interactions.
    - Validates and processes uploaded files, ensuring data integrity and compatibility with model requirements.
    - Implements robust error handling mechanisms and provides real-time notifications to enhance user experience and system reliability.

Model Processing Service (This is where customers algorithm in Python is):

- Dedicated service that handles **model execution** and processing of **input files**.
- Once processing is complete, it generates results and **PLT graphs**, which are forwarded to the backend.
- It also gathers performance metrics to track the efficiency of various models.
- REST API that handles requests to models

Database:

- Stores user details, models, input files, results, and statistics.
- It must handle **secure storage** of user data and ensure **fast access** to past optimizations and results.

System Behavior:

1. **User Workflow (Regular User):**
    - **Authentication & Session**
        - User signs in via **email + password**
        - Session remains active during use
        - Auto-logout triggers after predefined inactivity period
    - **Model Interaction**
        - Selects a model from the available list
        - Uploads input files for processing
        - System validates uploads and provides guidance if errors occur
            - Example: Prompts user to upload .csv file if wrong format is detected
            - Clear error messages with resolution steps
    - **Results Management**
        - Views results from the model execution
        - Downloads result files or graphs
        - Compares results from past optimizations
        - Accesses and downloads historical outputs
2. **Admin Workflow:**
    - Admin signs in using **email + password**.
    - Manages users by inviting or revoking access.
    - Links or unlinks models, assigns specific models to users.
    - Views **statistics** on users and models to monitor usage and performance.
    - Can perform all actions available to regular users.
