SoilModeler Phase 1 Report

Project: The Spectral Soil Modeler

Course: Software Systems Development

International Institute of Information Technology, Hyderabad

Contributors:

1. Anupam Dwivedi - 2025201032
2. Devansh Singh - 2025204007
3. Shobhan Parida - 2025201043
4. Krishna Pokuri - 2025202024
5. Yashwanth B K - 2025201028

Date: Tuesday 30th September, 2025

Contents

1 Understanding of the Project

The Spectral Soil Modeler is an automated machine learning (ML) workflow designed as an interactive web application.

Key Objectives:

- Automate the creation and evaluation of all preprocessing and model pipelines.
- Provide robust validation using cross-validation and metrics such as R^2, RMSE, and RPD.
- Visualize results through an intuitive Streamlit dashboard (leaderboard, prediction plots, diagnostics).
- Enable model diagnosis via feature importance plots to highlight predictive spectral bands.
- Export best models for reuse on new, unlabeled data.

2 Stakeholder Analysis: Personas, Roles, and System Actors

User Personas

- **Primary Persona (Soil Scientist):** Domain expert requiring an intuitive UI for modeling experiments without coding.
- **Secondary Persona (Project Lead/PI):** Senior researcher focused on scientific rigor, reproducibility, and output analysis.

Builder Roles

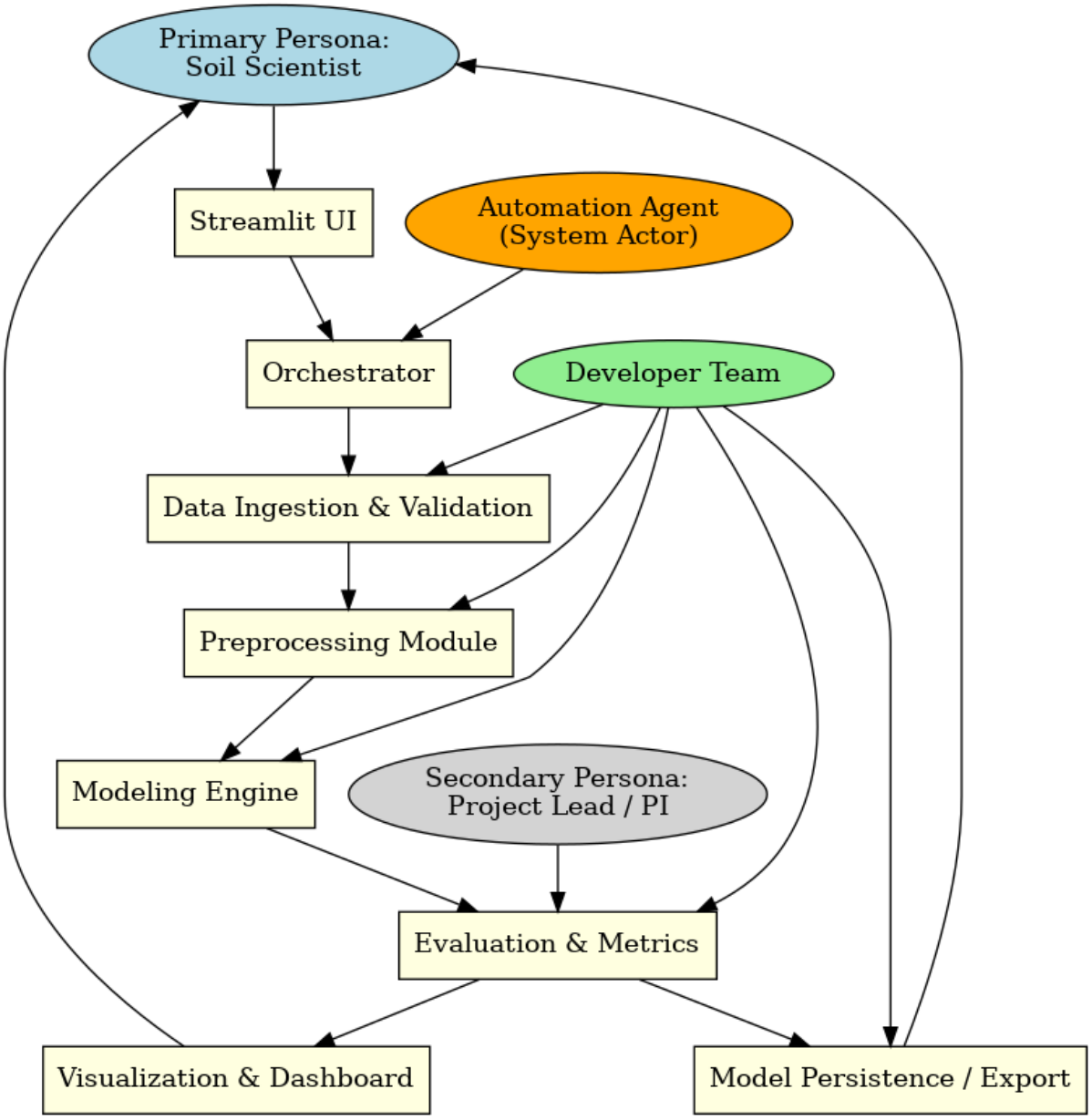
- **Developer Team:** Responsible for building, maintaining, and extending the application.

System as an Actor

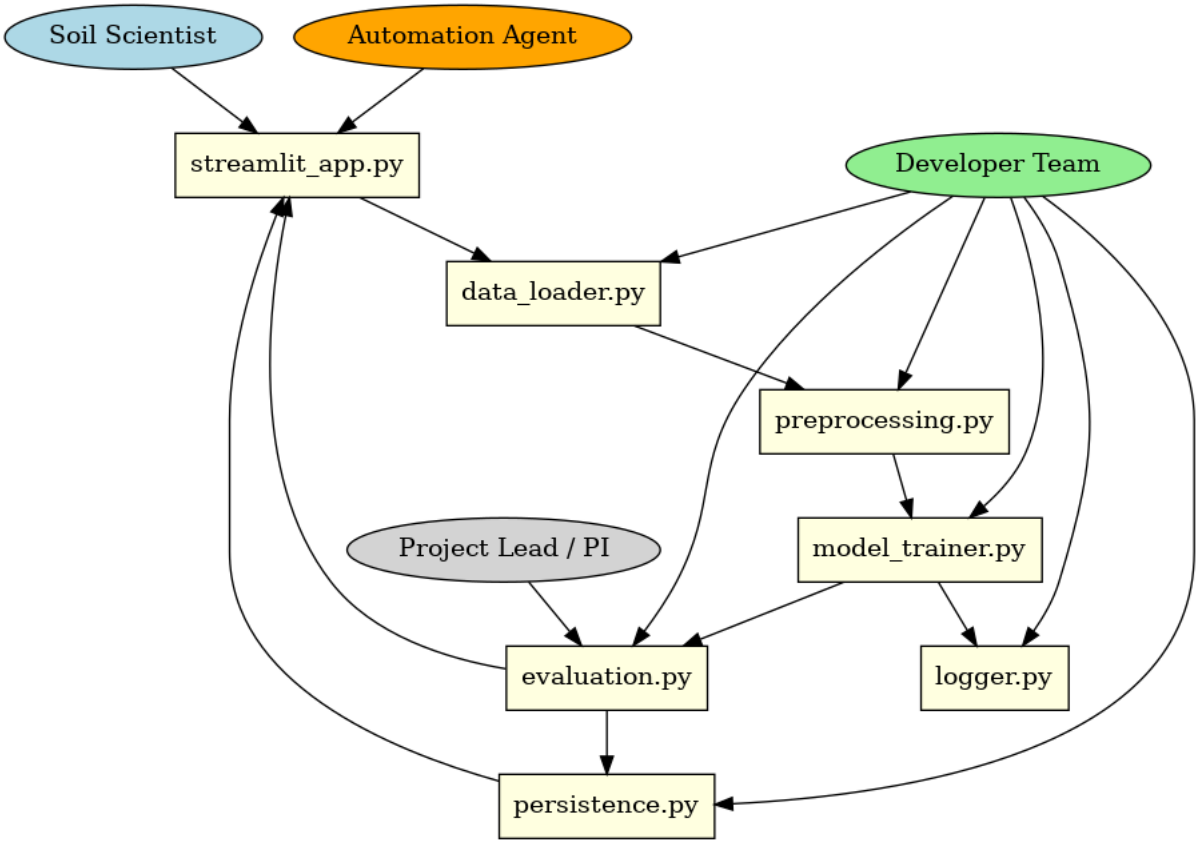
- **Automation Agent:** Executes commands, runs pipelines, calculates metrics, handles logging and errors.

3 Solution Diagrams

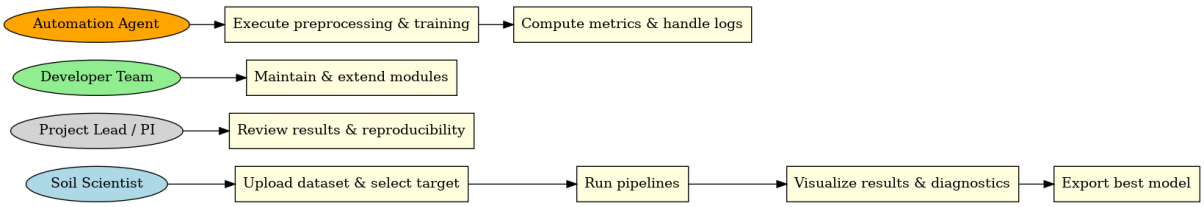
High-Level Design (HLD)



Low-Level Design (LLD)



Use Case Diagram



4 Proposed Solution Description

Through a Streamlit interface, soil scientists upload datasets, select target properties, and run automated experiments. The Automation Agent orchestrates data validation, preprocessing (reflectance, absorbance, continuum removal), model training (PLSR, Cubist, GBRT, KRR, SVR), and evaluation (R^2, RMSE, RPD). Results are displayed in interactive dashboards and best models can be exported.

The system's Low-Level Design divides functionality into modular Python components:

- `data_loader` (ingestion and validation),
- `preprocessing` (spectral transforms),
- `model_trainer` (training and CV),

- `evaluation` (metrics and importance),
- `persistence` (save/export models),
- `logger` (run tracking).

5 Technology Stack

- **Backend & ML:** Python, Pandas, NumPy, Scikit-learn, Joblib
- **Frontend/Visualization:** Streamlit, Matplotlib, Seaborn
- **Auxiliary Tools:** Git (version control), LLMs (Gemini, Claude, ChatGPT) for boilerplate code

6 Proposed Timeline

October 2025 — Core Backend Development

- Project kickoff and environment setup
- Implement `data_loader.py` (ingestion, validation, splitting)
- Implement `preprocessing.py` (reflectance, absorbance, continuum removal, filters)
- Implement core modeling engine and cross-validation logic (`model_trainer.py`)
- Unit tests for backend components

November 2025 — UI, Integration & Refinement

- Build Streamlit UI scaffold and pages
- Integrate frontend with backend workflows
- Implement visualizations (leaderboard, prediction plots, feature importance)
- Add model persistence/export (joblib, optional MLflow)
- End-to-end testing and user acceptance testing (UAT)

December 2025 — Finalization & Submission

- Polish UI and performance optimizations
- Prepare developer and user documentation (including LLM usage notes)
- Containerize app (Docker) and prepare submission package(optional)
- Final stakeholder demo and submission by Dec 2, 2025