B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

OOMD Mini Project Report

Crop Recommendation System

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering in
Computer Science and Engineering

Submitted by:

Chirag S (1BM23CS079)
D A Chethan (1BM23CS083)
Hiran B(1BM23CS113)
Hitish Rao P (1BM23CS116)

Department of Computer Science and Engineering
B.M.S.College of Engineering Bull Temple Road,
Basavanagudi, Bangalore 560 019
2025-26

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **Chirag S (1BM23CS079), D A Chethan (1BM23CS083), Hiran B (1BM23CS113), Hitish Rao P (1BM23CS116)** students of 5th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this OOMD Mini Project entitled "**Crop Recommendation System**" has been carried out in Department of CSE, B.M.S. College of Engineering, Bangalore during the academic semester August 2025- December 2025. I also declare that to the best of our knowledge and belief, the OOMD mini Project report is not from part of any other report by any other students.

Signature of the Candidate

Chirag S (1BM23CS079)
D A Chethan (1BM23CS083)
Hiran B (1BM23CS113)
Hitish Rao P(1BM23CS116)

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOMD Mini Project titled "**Crop Recommendation System"** has been carried out by **Chirag S(1BM23CS079), D A Chethan (1BM23CS083), Hiran B (1BM23CS113), Hitish Rao P (1BM23CS116)** during the academic year 2025-2026.

Signature of the Faculty in Charge

Table of Contents

Chapter 1: Problem Statement

Agricultural productivity in many regions is significantly affected by farmers' limited access to accurate, datadriven recommendations regarding suitable crops for their land. Most small-scale and traditional farmers rely on experience-based or generic advice, which often fails to consider critical factors such as soil type, nutrient levels, local climate conditions, rainfall patterns, and historical crop performance. This leads to reduced yields, soil depletion, inefficient resource usage, and financial losses.

To address this gap, there is a need for an intelligent system that can analyze multiple environmental and soil parameters and provide personalized, scientifically backed crop recommendations. An AI-based crop recommendation system can assist farmers by processing real-time and historical data, identifying optimal crop options, and supporting informed agricultural decision-making.

This project aims to design and model an AI-Based Crop Recommendation System using Object-Oriented Modelling (OOM) principles. The system will intake soil attributes (such as pH, nitrogen, phosphorus, and potassium levels), climatic factors (such as temperature and rainfall), and location-specific data to recommend the most suitable crops for cultivation. The solution intends to enhance agricultural efficiency, optimize resource utilization, support sustainable farming, and ultimately improve farmers' income and productivity.

Chapter 2: Software Requirement Specification

AI-Based Crop Recommendation System

Problem Statement

Design and develop an **AI-Based Crop Recommendation System (ACRS)** that assists farmers in choosing the most suitable crop based on their **soil parameters**, **location**, and **real-time weather conditions**.
The system should allow farmers to input soil data, fetch weather dynamically, and receive accurate crop recommendations generated by a **Generative AI model**. Additional features include maintaining prediction history, profile management, and feedback submission to improve system usability.

1. Introduction

1.1 Purpose of this Document

The purpose of this SRS document is to clearly define all requirements for the **AI-Based Crop Recommendation System**, including functional, non-functional, performance, and interface specifications.
It serves as a reference for developers, testers, and stakeholders to ensure a consistent understanding of system goals and behavior.

1.2 Scope of this Document

The ACRS aims to modernize and simplify crop selection decisions for farmers by providing:

- AI-based crop recommendations
- Soil parameter and nutrient analysis
- Real-time weather fetching based on farmer's location
- Prediction history storage
- Farmer profile management
- Feedback submission module
- Simple, visual, and mobile-friendly UI

The SRS also outlines development tools, constraints, and estimated effort required for the project.

1.3 Overview

The system collects soil data from farmers, automatically fetches weather data, processes all inputs, and generates crop predictions using **Google Gemini AI**.
The system improves decision-making, reduces wrong crop choices, and enhances productivity by offering personalized agricultural guidance.

2. General Description

The ACRS is designed for:

- **Farmers**

Enter soil values, fetch weather, get crop recommendations, view history, and submit feedback. •

System (AI Model)

Processes input, generates predictions, and provides explanations.

• Administrators (Optional Future Scope)

Monitor system performance and accuracy trends.

3. Functional Requirements

3.1 Farmer Services

- Create basic profile (name, location, language).
- Enter soil parameters (pH, N, P, K, moisture).
- Automatically fetch weather using location.
- Get AI-generated crop recommendations and explanations.
- View confidence levels for each recommendation.
- Check prediction history.
- Submit feedback on system accuracy.
- Update profile details.

3.2 AI Recommendation Engine

- Combine soil data and weather metrics into a structured prompt.
- Invoke Gemini AI model to generate recommendations.
- Produce ranked list of crops with reasoning.
- Handle error cases (empty response, invalid input).
- Log predictions for future improvement.

3.3 Weather Fetching Module

- Accept location name from farmer.
- Validate location and fetch data using Open-Meteo API (or equivalent).
- Extract temperature, humidity, rainfall, wind speed.
- Display errors for invalid locations.

3.4 Prediction History

- Store details of previous predictions locally.
- Display past crops, confidence score, date, and explanation.
- Allow farmers to view previous inputs.

3.5 Feedback Module

- Allow users to give 1–5 star ratings.
- Accept textual comments.
- Display success message after submission.

3.6 User Interface Requirements

- Clean, modern UI built using **React + Tailwind + shadcn/UI**.
- Responsive design for mobile and laptop.
- Simple navigation bar with links (Home, Get Started, History, Profile, Feedback).

4. Interface Requirements

4.1 User Interface

- Home page with hero banner and feature cards.
- Soil & Weather data form.
- Prediction result page with explanations.
- Farmer profile dashboard.
- Feedback page.
- All forms must provide real-time validation and error messages.

4.2 Integration Interfaces

- Google Gemini API for prediction generation.
- Open-Meteo API (or similar) for real-time weather data.
- Browser local storage for history management.
- Environment variables to store API keys securely.

5. Performance Requirements

5.1 Response Time

- Weather API response: < 2 seconds
- AI prediction response: 3–5 seconds • UI transitions must be instant and fluid.

5.2 Scalability

- UI should support high traffic if hosted online.
- Modular architecture for future expansion (adding disease detection, more crops).

5.3 Data Integrity

- Ensure correct mapping of soil + weather data before sending to AI.
- Prevent duplicate history entries.

- Maintain clean formatting of AI responses.

6. Design Constraints

6.1 Hardware Limitations

- Runs on mobile phones, tablets, and standard PCs.
- Requires basic internet connectivity for API requests

6.2 Software Dependencies

- React
- Tailwind CSS
- Shadcn/ui components
- Open-Meteo API
- Gemini API
- Node.js + npm
- Modern browsers (Chrome, Edge, Firefox)

7. Non-Functional Attributes

7.1 Security

- API keys stored in `.env`.
- HTTPS required if deployed online.
- Input validation for all forms.

7.2 Reliability

- Weather and AI API failures handled gracefully with fallback messages.
- Local history always accessible offline.

7.3 Scalability

- Component-based architecture allows easy feature expansion.

7.4 Portability

- Works on all major browsers and mobile devices.

7.5 Usability

- Guided flow for farmers.
- Clean icons, large button sizes, simple form layouts.

7.6 Reusability

- AI wrapper functions and weather fetch utilities are modular.

7.7 Compatibility

- Compatible with latest React versions and web standards.

7.8 Data Integrity

- Prevents blank or inconsistent user inputs.
- Ensures well-structured prompts sent to AI for accurate results.

8. Preliminary Schedule and Budget

Development Time: 4–6 weeks

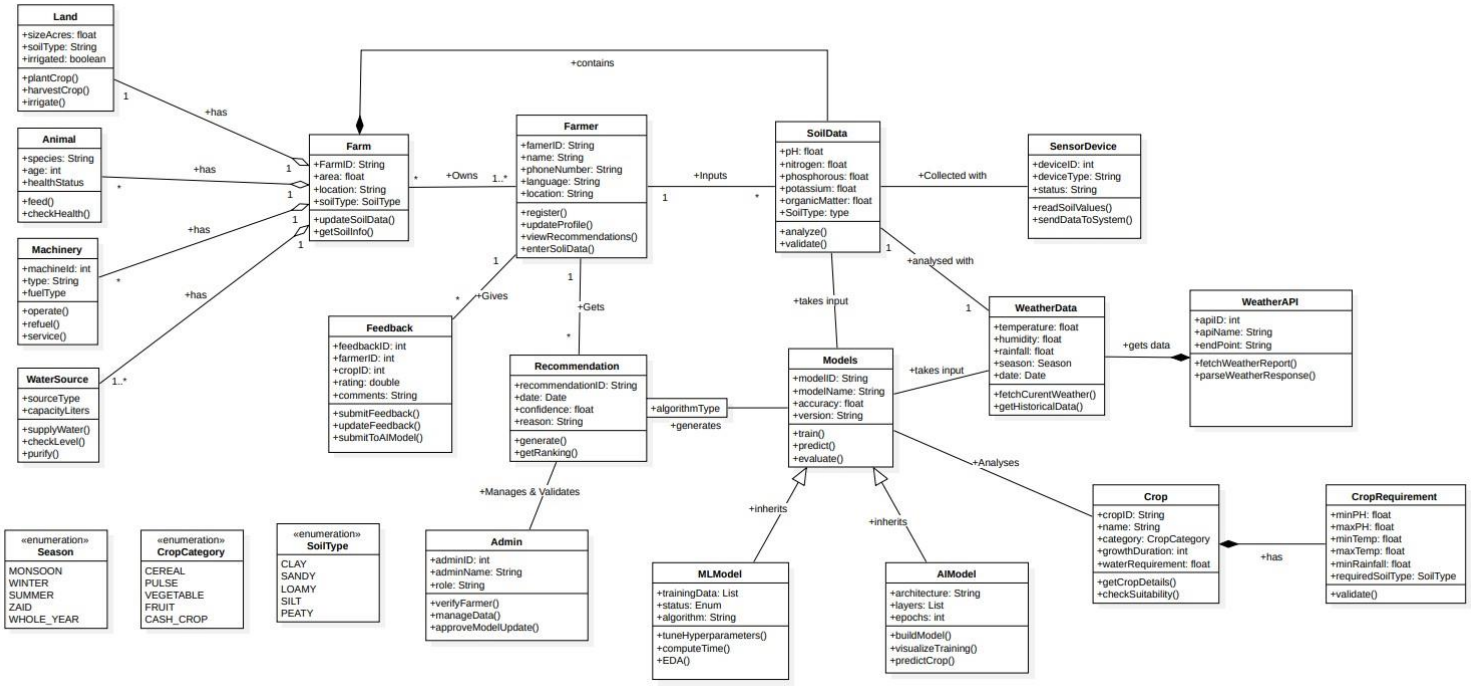
(Including UI design, integration, testing, and documentation)

Budget: Minimal (only UI + AI API usage)

- Free open-source stack
- Gemini API usage based on tokens
- Optional hosting costs

Chapter 3: Class Modelling

1. Class Diagram



1.1. Farm

Relevance:
The Farm class represents the central agricultural unit where soil is analyzed, crops are grown, and farming operations occur. It contains primary attributes such as farm ID, area, location, and soil type.
This class is essential for associating all entities like animals, machinery, land, and water sources with a specific farm. It also stores soil data updates and serves as the main container for farm-related activities. 1.2. Farmer

Relevance:
The Farmer class represents the primary user interacting with the system. It stores farmer details such as name, phone number, language, and associated farm information.
This class allows farmers to register, update profiles, enter soil data, and view recommendations. It is central to enabling personalized AI-based agricultural guidance.

1.3. SoilData

Relevance:
The SoilData class captures detailed soil parameters including pH, nitrogen, phosphorus, potassium, and organic matter.
It plays a critical role in determining crop suitability. The system uses this class to validate, analyze, and send accurate data to the AI model for recommendation generation.

1.4. WeatherData

Relevance:
The WeatherData class represents real-time atmospheric conditions such as temperature, humidity, rainfall, and season.
This class ensures that recommendations consider the current environment. WeatherData integrates with WeatherAPI and supports historical lookups for improved predictions.

1.5. WeatherAPI

Relevance:
The WeatherAPI class interacts with external meteorological services. It fetches weather reports and parses them for use within the system.
This class ensures that farmers receive accurate and location-specific weather data, which is essential for AI recommendation accuracy.

1.6. Models (Base AI Model Class)

Relevance:
The Models class acts as an abstract foundation for all machine-learning components. It stores model metadata such as ID, name, accuracy, and version.
It provides generic functions like training, prediction, and evaluation. This enables scalable integration of different AI models within the system.

1.7. MLModel (Derived from Models)

Relevance:
The MLModel class represents traditional machine learning models (like Random Forest, Decision Trees). It handles tasks such as training on structured datasets, computing accuracy, tuning hyperparameters, and exploratory data analysis (EDA).
This class offers a simpler alternative to deep learning models and is useful for structured crop recommendation tasks.

1.8. AIModel (Derived from Models)

Relevance:
The AIModel class represents advanced deep-learning or generative AI architectures. It includes parameters like layers, epochs, and internal training logic.
It generates more contextual crop predictions and visualizes training metrics. This class enhances system intelligence and adaptability.

1.9. Recommendation

Relevance:
The Recommendation class stores the AI-generated crop suggestions. It includes recommendation ID, date, confidence score, explanation, and algorithm type used.
This class helps farmers understand the reasoning behind suggested crops and maintains accountability in the prediction process.

1.10. Crop

Relevance:
The Crop class contains all crop-related information such as crop name, category, growth duration, and water requirement.
It helps both AI models and farmers understand the agronomic details needed to check suitability and perform crop ranking.

1.11. CropRequirement

Relevance:
The CropRequirement class defines the ideal environmental and soil conditions required for each crop, including pH, temperature, rainfall, and soil type.
It ensures the system correctly validates whether a crop is suitable based on farmer-provided soil and weather data.

1.12. Feedback

Relevance:
The Feedback class captures farmer responses about the accuracy and usefulness of predictions. It stores rating, comments, crop ID, and farmer ID.
This class helps the system improve its future recommendations and provides a channel for user engagement and system refinement.

1.13. Admin

Relevance:
The Admin class represents system administrators responsible for verifying farmers, managing data, and validating AI models.
It ensures system integrity, monitors operations, and maintains the overall functionality of the recommendation platform.

1.14. SensorDevice

Relevance:
The SensorDevice class represents IoT soil sensors used in farms. These sensors capture real-time soil values and send them automatically to the system.
This class supports automated data collection and reduces manual input errors.

1.15. Land

Relevance:
The Land class defines the physical characteristics of farmland such as size, soil type, and irrigation status. It supports essential operations like planting, harvesting, and irrigation, providing the base environment for crop growth.

1.16. Animal

Relevance:
The Animal class represents livestock present on the farm. It includes attributes like species, age, and health status. This class tracks animal activities and integrates with farm management but does not directly influence crop prediction.

1.17. Machinery

Relevance:
The Machinery class represents tractors, harvesters, and other farming equipment.
It helps track operational cycles, refueling, and service schedules, contributing to broader farm management. 1.18.

WaterSource

Relevance:
The WaterSource class defines the water supply for a farm, including capacity, source type, and purification properties.
It is essential for irrigation planning and overall sustainability of farming operations.

1.19. Enumerations (Season, CropCategory, SoilType)

Relevance:
Enumerations provide controlled vocabulary for the system:

- Season: Different crop seasons
- CropCategory: Cereal, fruit, vegetable, etc.
- SoilType: Clay, sandy, loamy, etc.

Chapter 4: State Modeling

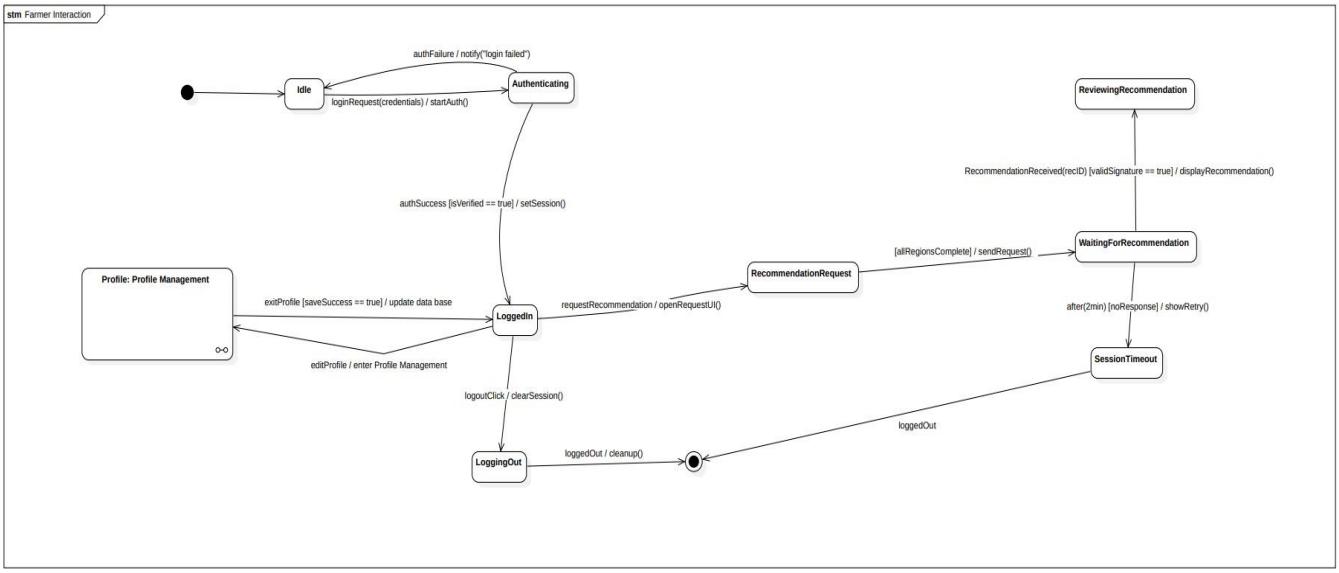


Fig 4.1

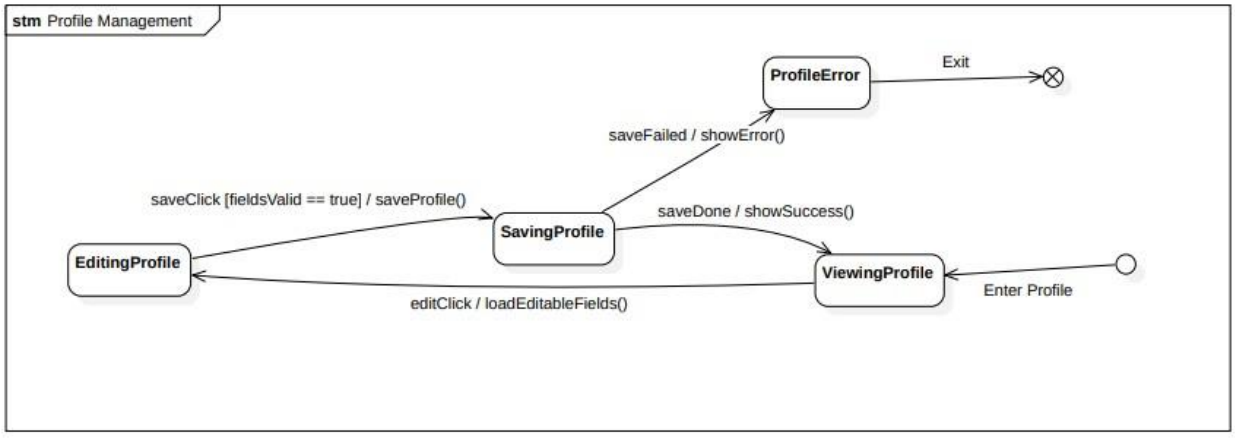


Fig 4.2

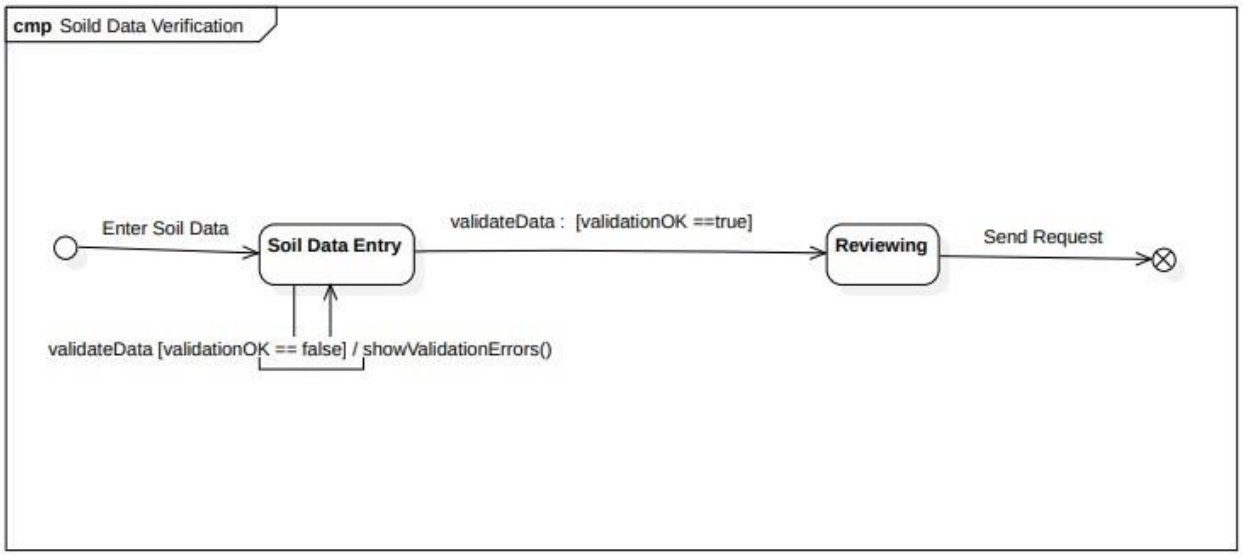

cmp Soild Data Verification
Enter Soil Data
Soil Data Entry
validateData : [validationOK ==true]
Reviewing
Send Request
validateData [validationOK == false] / showValidationErrors()

Fig 4.3

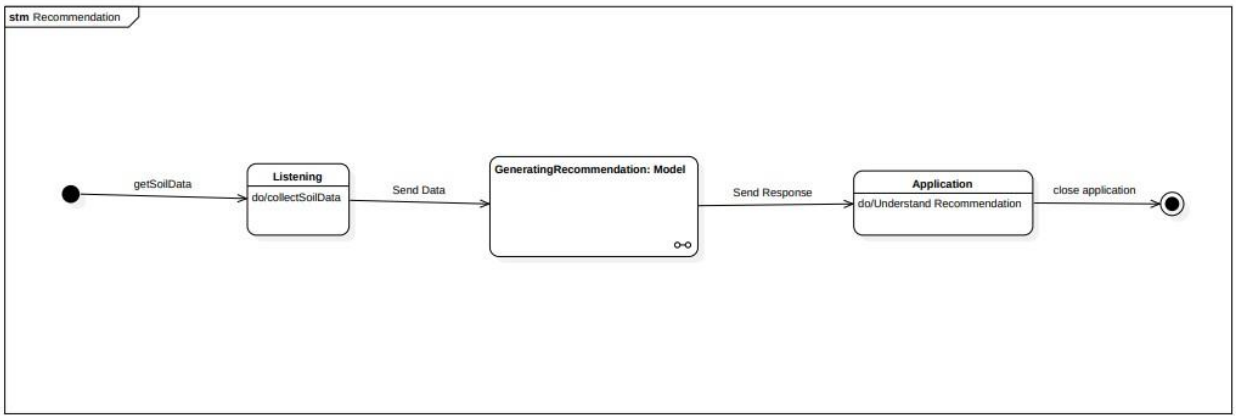

stm Recommendation
getSoilData
Listening
do/collectSoilData
Send Data
GeneratingRecommendation: Model
Send Response
Application
do/Understand Recommendation
close application

Fig 4.4

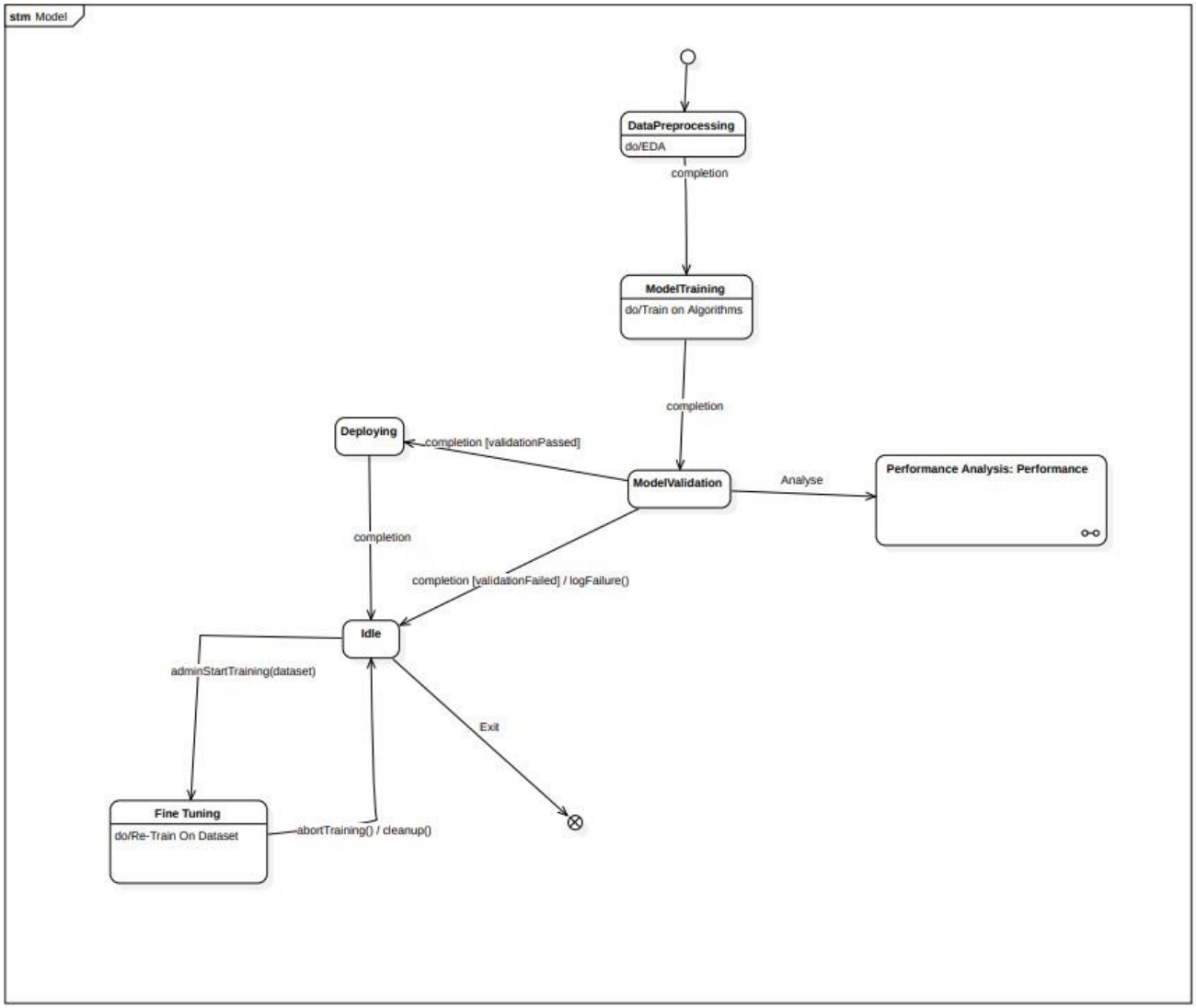

stm Model
DataPreprocessing
do/EDA
completion
ModelTraining
do/Train on Algorithms
completion
Deploying
completion [validationPassed]
ModelValidation
Analyse
Performance Analysis: Performance
completion
completion [validationFailed] / logFailure()
Idle
adminStartTraining(dataset)
Exit
Fine Tuning
do/Re-Train On Dataset
abortTraining() / cleanup()

Fig 4.5

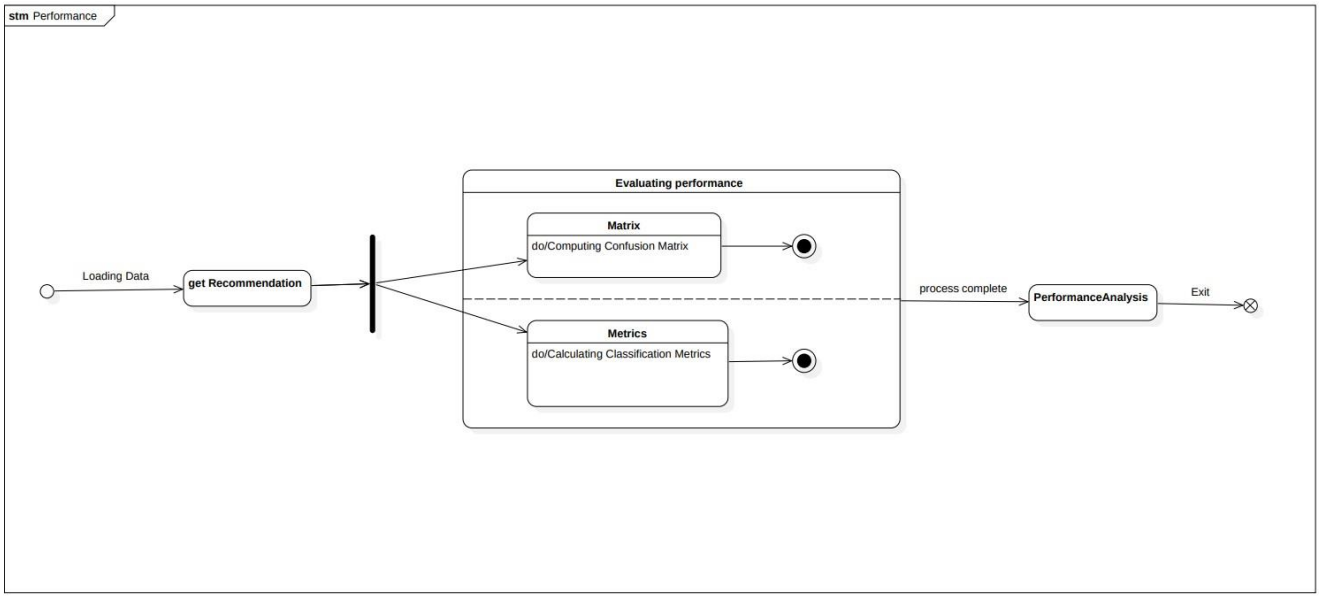

stm Performance
Loading Data
get Recommendation
Evaluating performance
Matrix
do/Computing Confusion Matrix
Metrics
do/Calculating Classification Metrics
process complete
PerformanceAnalysis
Exit

Fig 4.6

2.1 Explanation of Each State

(For all 3 state machines)

2.1.1 Farmer Interaction State Machine

State: Idle

- This is the **initial state** where the farmer has not interacted with the system yet.
- The farmer may choose to log in, view the home page, or access basic information.
- No session or authentication process has started.
- The system remains inactive until the farmer initiates login.

State: Authenticating

- Triggered when the farmer submits login credentials.
- The system validates the credentials with stored records.
- If validation succeeds, the session is created; otherwise, an error is shown.
- This state ensures secure access to personalized features.

State: LoggedIn

- Represents a successful login, where the farmer now has an active session.
- The farmer can navigate to recommendations, profiles, or soil-entry forms.
- This is the main interactive state where all major functionality becomes accessible.
- The system maintains session data while the user performs operations.

State: RecommendationRequest

- Entered when the farmer requests a crop recommendation.
- The system opens the input UI for soil & weather details.
- Required parameters are checked for completeness before sending to the AI model.
- Prepares the system for generating the recommendation request.

State: WaitingForRecommendation

- The system has sent all collected data to the AI model for analysis.
- It waits for a response from the AI engine or connected APIs.
- This state handles asynchronous operations and includes timeout conditions.
- The UI shows progress/loading feedback to the farmer.

State: ReviewingRecommendation

- Triggered when the AI returns valid crop recommendations.

- The farmer can read the recommended crops, confidence score, and explanation.
- This state enhances decision-making by presenting clear and interpretable results.
- From here, the farmer can submit feedback or start a new query.

State: SessionTimeout

- Entered when the system does not receive a response within the allowed time.
- Farmers are notified that no response was received.
- The UI may allow retrying the request.
- Prevents system hang-ups or indefinite waiting.

State: LoggingOut

- Activated when the farmer chooses to log out.
- The system clears the active session, temporary data, and UI state.
- Ensures secure exit and protection of user information.
- After cleanup, returns to the Idle state.

2.1.2 Profile Management State Machine

State: ViewingProfile

- The farmer is viewing their profile details such as name, location, and language.
- This is the default state when entering the profile management UI.
- No modifications are being made at this point.
- Acts as the base state for profile editing transitions.

State: EditingProfile

- Triggered when the farmer chooses to edit their profile details.
- The system loads current data into editable input fields.
- User can modify their information such as phone number or location.
- Validation checks may start before saving.

State: SavingProfile

- Activated when the farmer clicks the save button.
- The system validates fields and attempts to update the database.
- Shows saving animations or progress indicators.
- This state ensures that only correct and validated data is saved.

State: ProfileError

- Entered when validation fails or a database update error occurs.
- UI informs the farmer of invalid fields or technical problems.
- Farmer may retry editing or exit the profile module.
- Ensures robust handling of incorrect profile updates.

2.1.3 Soil Data Verification State Machine

State: Soil Data Entry

- Represents the initial state where the farmer enters soil parameters (pH, N, P, K, moisture).
- User input is collected but not yet validated.
- All fields must be completed before moving forward.
- The UI guides the farmer through required inputs.

State: Reviewing

- Triggered when soil data passes validation checks.
- The system formats the data and prepares it for sending to the AI recommendation model.
- Farmers may review their inputs before final submission.
- Ensures correctness and prevents incorrect data propagation.

State: Soil Data Error

(Implicit from transition)

- Activated when the entered soil data fails validation.
- Alerts the farmer about missing or invalid values.
- Prevents system from generating inaccurate recommendations.
- User is sent back to the soil entry form to correct errors.

2.2 Explanation of Each Event / Transition

Event: loginRequest(credentials)

- Farmer enters login details and submits.
- System transitions from Idle → Authenticating.

Event: authFailure / notify("Login failed")

- Invalid credentials detected.
- System stays in Idle, showing error.

Event: authSuccess [isVerified == true]

- Credentials verified successfully.
- Moves Authenticating → LoggedIn.

Event: editProfile / enterProfileManagement

- Farmer selects the profile option.
- Moves LoggedIn → ViewingProfile.

Event: saveClick [fieldsValid == true]

- Farmer submits updated profile data.
- Moves EditingProfile → SavingProfile.

Event: saveDone / showSuccess()

- Successful profile update.
- Moves SavingProfile → ViewingProfile.

Event: validateData [validationOK == true]

- Soil data is correct.
- Moves Soil Data Entry → Reviewing.

Event: validateData [validationOK == false]

- Soil data contains errors.
- Stays in Soil Data Entry (show errors).

Event: requestRecommendation / openRequestUI

- Farmer initiates crop prediction request.
- Moves LoggedIn → RecommendationRequest.

Event: sendRequest

- System sends soil + weather details to AI.
- Moves RecommendationRequest → WaitingForRecommendation.

Event: RecommendationReceived

- AI generates prediction successfully.
- Moves WaitingForRecommendation → ReviewingRecommendation.

Event: noResponse / showRetry()

- AI fails to respond.
- Moves WaitingForRecommendation → SessionTimeout.

Event: logoutClick / clearSession()

- Farmer logs out manually.
- Moves LoggedIn → LoggingOut.

Event: loggedOut / cleanup()

- System clears data and ends session.
- Moves LoggingOut → Idle.

Chapter 5: Interaction Modeling

5.1 Use Case Diagram

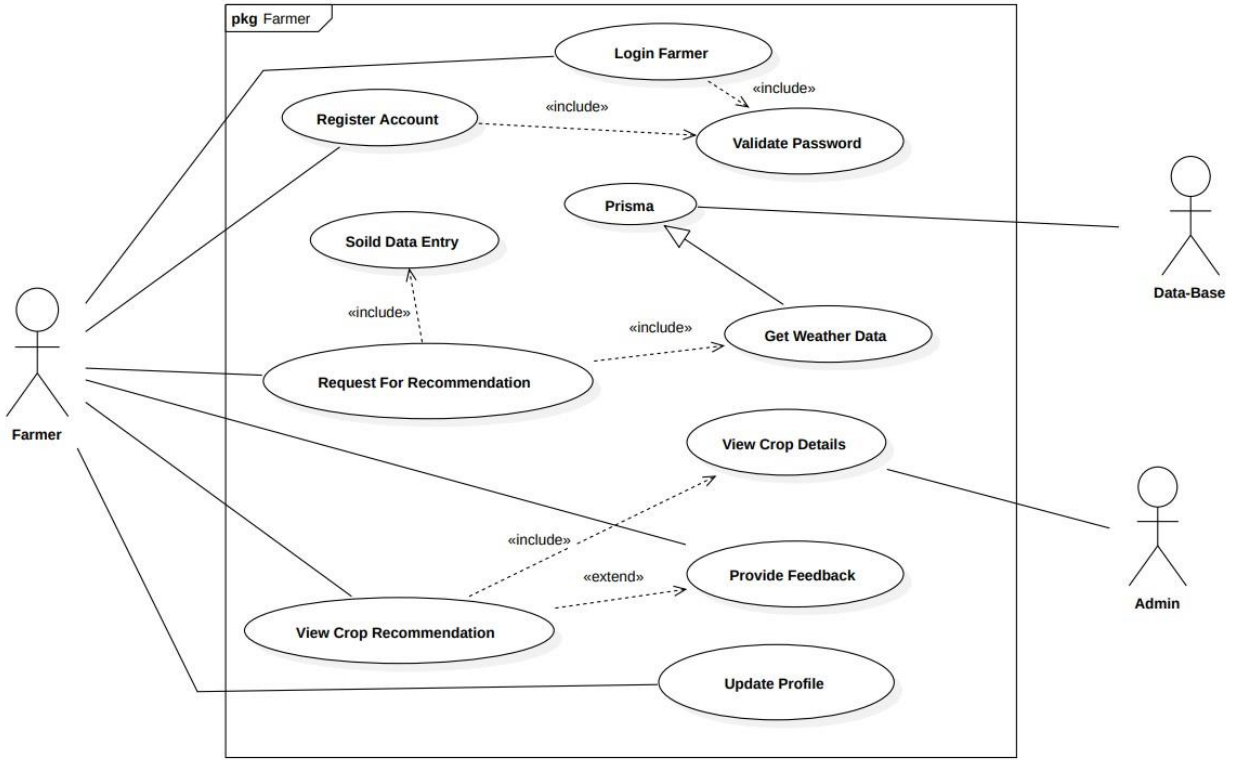


Fig 5.1.1

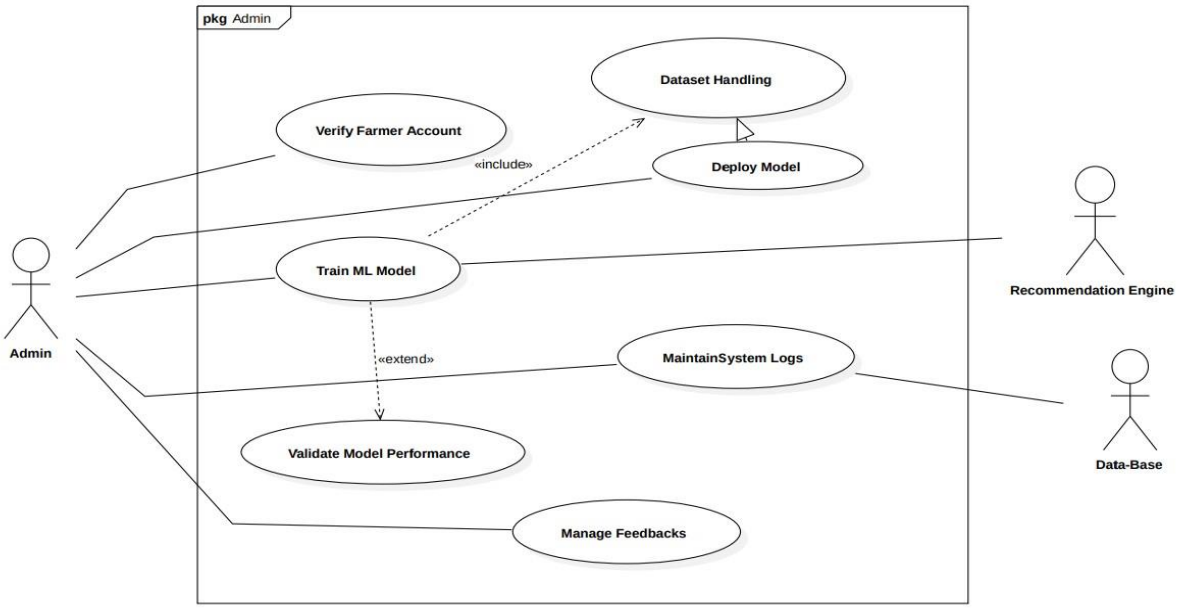


Fig 5.1.2

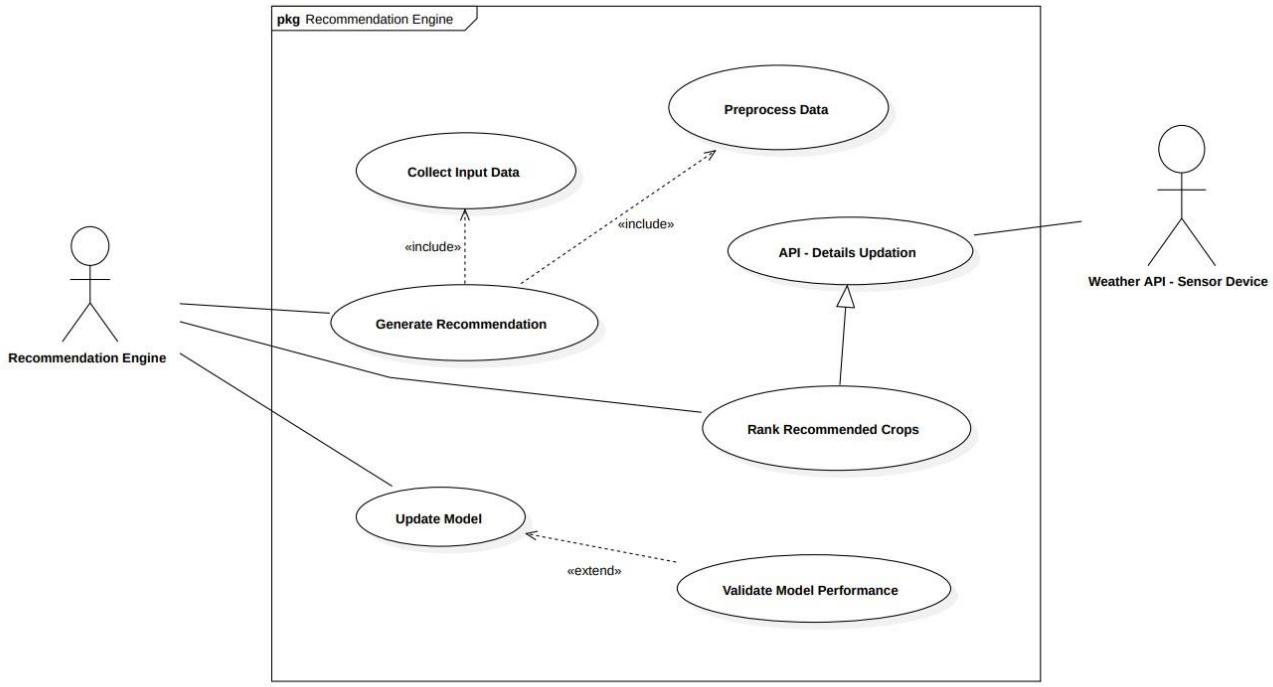


Fig 5.1.3

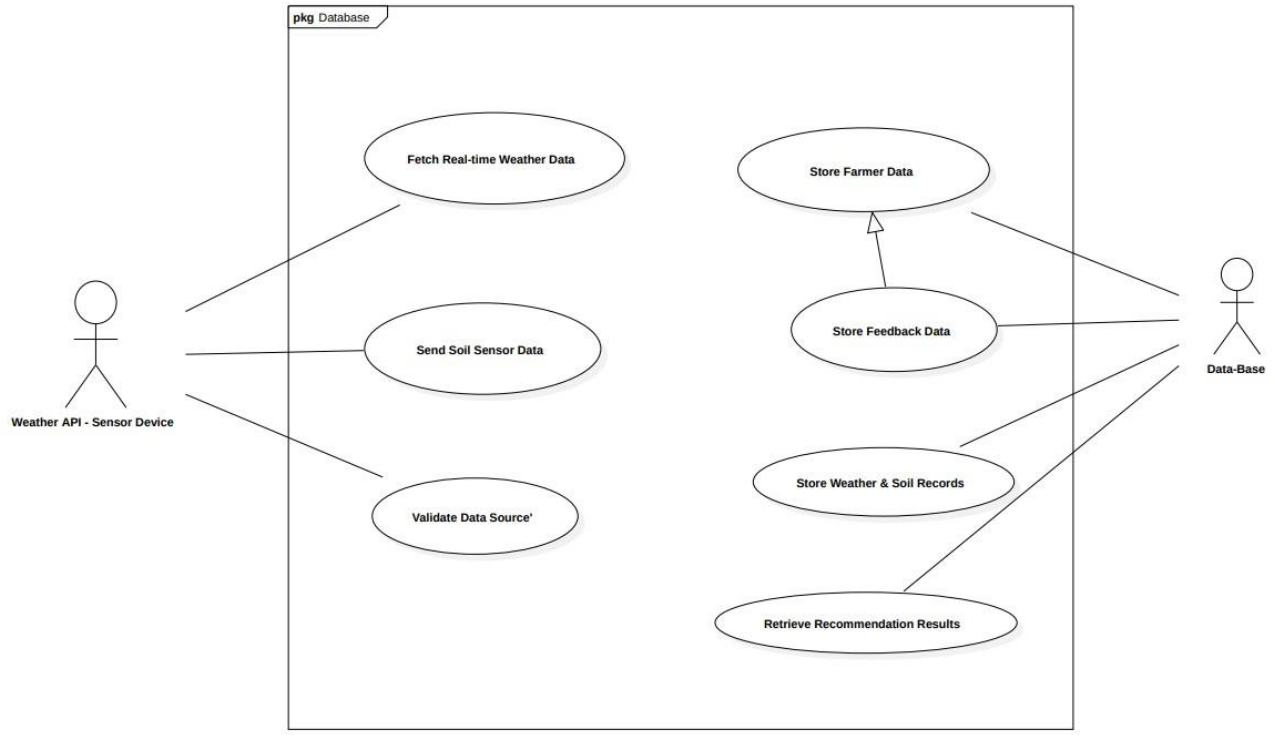


Fig 5.1.4

Relevance of Each Actor

1. Farmer

The Farmer is the primary user of the FarmWise system and interacts with almost every major feature.
Their inputs—soil data, location, weather retrieval requests, and feedback—form the core foundation of all AI computations.
Farmers depend on the system for actionable insights, crop recommendations, and profile management, making their role essential for system utility and real-world impact.
Their continuous use also helps the model learn and improve through feedback loops.

2. Admin

Admins handle system-level tasks such as verifying farmers, managing datasets, monitoring the AI model's performance, and updating system configuration.
Their role ensures that the platform remains secure, accurate, and aligned with agricultural standards.
Admins also manage feedback records and oversee deployment and version control of ML and AI models. They act as the supervisory authority responsible for maintaining system integrity and reliability.

3. Recommendation Engine (AI Model)

This AI actor represents the machine learning system that processes farmer inputs, analyzes soil & weather data, and generates the final ranked list of recommended crops.
It performs essential tasks such as preprocessing input data, updating the model, evaluating performance, and ranking outputs.
This actor transforms user-provided raw data into meaningful agricultural insights.
Its efficiency and accuracy directly influence the effectiveness of the entire platform.

4. Weather API / Sensor Device

The Weather API provides real-time climatic conditions such as temperature, humidity, and rainfall.
Sensor devices may provide soil moisture or real-time soil readings, enriching accuracy and reducing manual input.
These external actors ensure that predictions are based on **current environmental conditions**, making the recommendations more relevant and location-specific.
They form the backbone of dynamic and automated data integration.

5. Database

The Database actor securely stores all persistent system data, including farmer profiles, soil records, weather logs, recommendation histories, and feedback.
It ensures that the system retrieves and updates information consistently for all modules.
The database enables transparency, reusability, and historical tracking for both farmers and admins.
It is a crucial backbone for ensuring that data storage, retrieval, and validation flow smoothly across the entire architecture.

Relevance of Each Use Case

1. Register Account / Login Farmer

This use case enables farmers to securely authenticate and gain access to personalized recommendations.
It includes password validation and session creation to ensure protected access.
User identity is crucial for storing personalized data and maintaining accurate history records. This is foundational for all further interactions.

2. Soil Data Entry

Allows farmers to input key soil attributes like pH, nitrogen, phosphorus, potassium, and moisture.
These inputs form the primary data needed for accurately predicting suitable crops.
Validations ensure that incorrect or incomplete soil details are not forwarded to the AI model. This use case plays a crucial role in ensuring clean and reliable data.

3. Get Weather Data

Fetches real-time weather based on the farmer's location using the Weather API.
Includes dynamic retrieval of temperature, humidity, and rainfall.
The integration ensures that recommendations remain climate-aware. It minimizes manual steps and improves prediction accuracy.

4. Request for Recommendation

Once soil and weather data are ready, the farmer initiates the AI prediction process.
This use case sends the collected data to the Recommendation Engine.
It handles asynchronous communication and begins the recommendation workflow. This is a central use case that triggers the core feature of the system.

5. View Crop Recommendation

Displays the list of crops recommended by the AI model along with confidence scores, explanations, and timestamp.
Helps farmers make informed decisions about optimal cultivation strategies.
This is the primary output that the system is designed to deliver.
Enhances usability by presenting insights in a simple and understandable manner.

6. Provide Feedback

Allows farmers to share ratings and comments about received recommendations.
Feedback helps improve future model accuracy by serving as training and validation data.
Admins use this information to monitor recommendation effectiveness.
This use case forms the feedback loop essential for continuous AI improvement.

7. Update Profile

Enables farmers to modify personal details such as name, phone number, language, or farm location.
Ensures that the recommendation engine always receives updated information.
Improves personalization and enhances user experience. Important
for maintaining accurate farmer records.

8. Verify Farmer Account (Admin)

Admins validate and approve newly registered farmers to maintain system credibility.
Ensures misuse prevention and prevents unauthorized access.
This use case maintains data integrity and system trustworthiness.
Admin approval is required before a farmer can access advanced features.

9. Dataset Handling (Admin)

Admins upload and manage datasets used for model training.
Ensures that the recommendation engine is trained on up-to-date agricultural data.
This use case is critical for AI model accuracy.
Includes cleaning, updating, and validating dataset quality.

10. Train ML Model (Admin / Recommendation Engine)

Triggers the ML model training process using available datasets.
Improves the AI system's capability to provide precise recommendations.
Admins monitor the training workflow, while the engine performs the computation. Ensures the continuous evolution of the system.

11. Validate Model Performance

Checks prediction accuracy, consistency, and reliability of the model.
Ensures that the deployed model meets agricultural standards and internal benchmarks.
Admins oversee performance summary and decide deployment readiness. This guarantees high-quality outputs for farmers.

12. Manage Feedback (Admin)

Admins review user feedback and identify issues affecting prediction quality.
Helps improve system trust and farmer satisfaction.
Feedback insights guide future system enhancements.
Contributes to transparent system governance.

13. Generate Recommendation (Engine)

The engine processes soil data, weather data, and crop requirements to compute the most suitable crops.
Executes the algorithm pipeline: preprocessing → prediction → ranking.
The backbone of the crop recommendation workflow. Ensures data-driven output for end-users.

14. Rank Recommended Crops

Sorts predicted crops based on suitability and confidence scores.
Ensures the most optimal crop is listed first.
Enhances farmer decision-making.
Makes results easy to interpret and act upon.

15. Fetch Real-Time Weather Data (Database/API)

Retrieves live weather logs and stores them for reference.
Supports accurate and timely predictions.

Electronic weather logs allow comparison and long-term seasonal analysis. Essential for adaptive agricultural planning.

5.2 Sequence Diagram

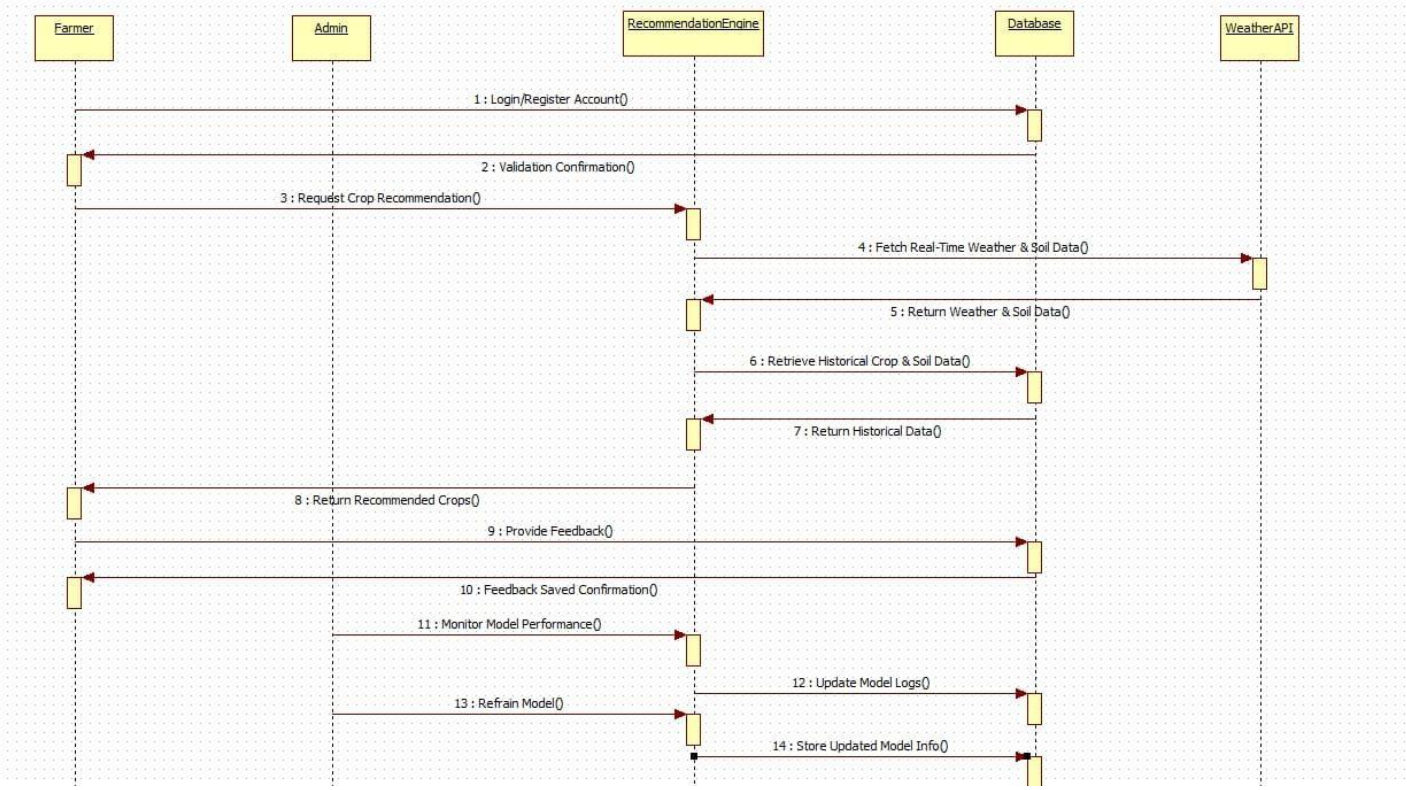


Fig 5.2.1

Sequence Diagram Explanation

The sequence diagram illustrates the complete workflow of how the FarmWise AI Crop Recommendation System processes a farmer's request—from registration to receiving recommendations, and finally to model retraining. The interaction begins when the **Farmer** registers or logs into the system. The request is forwarded to the **Admin**, who validates the farmer's account and sends back a confirmation, allowing the farmer to proceed.

Once authenticated, the Farmer initiates a **Request Crop Recommendation** action. This request is sent to the **Recommendation Engine**, which begins gathering necessary data for analysis. The engine contacts the **Database** to fetch previously stored soil and crop records and simultaneously interacts with the **Weather API** to retrieve real-time weather conditions such as temperature, humidity, and rainfall. After collecting weather details, the API returns the information to the engine, while the database sends historical soil and crop data back for deeper analysis.

With all required inputs available, the Recommendation Engine processes the data and generates a list of suitable crops tailored to the farmer's soil conditions and climate. It then sends the **Recommended Crops** back to the Farmer. After reviewing the results, the farmer submits feedback regarding the recommendation quality. This feedback travels to the Recommendation Engine, which validates and stores it in the Database.

The final part of the diagram focuses on system improvement through AI retraining. Once feedback is saved, the **Admin** monitors model performance based on aggregated feedback, prediction accuracy, and system logs. The admin may then trigger model retraining, prompting the Recommendation Engine to update its model parameters.

5.3 Activity Diagram

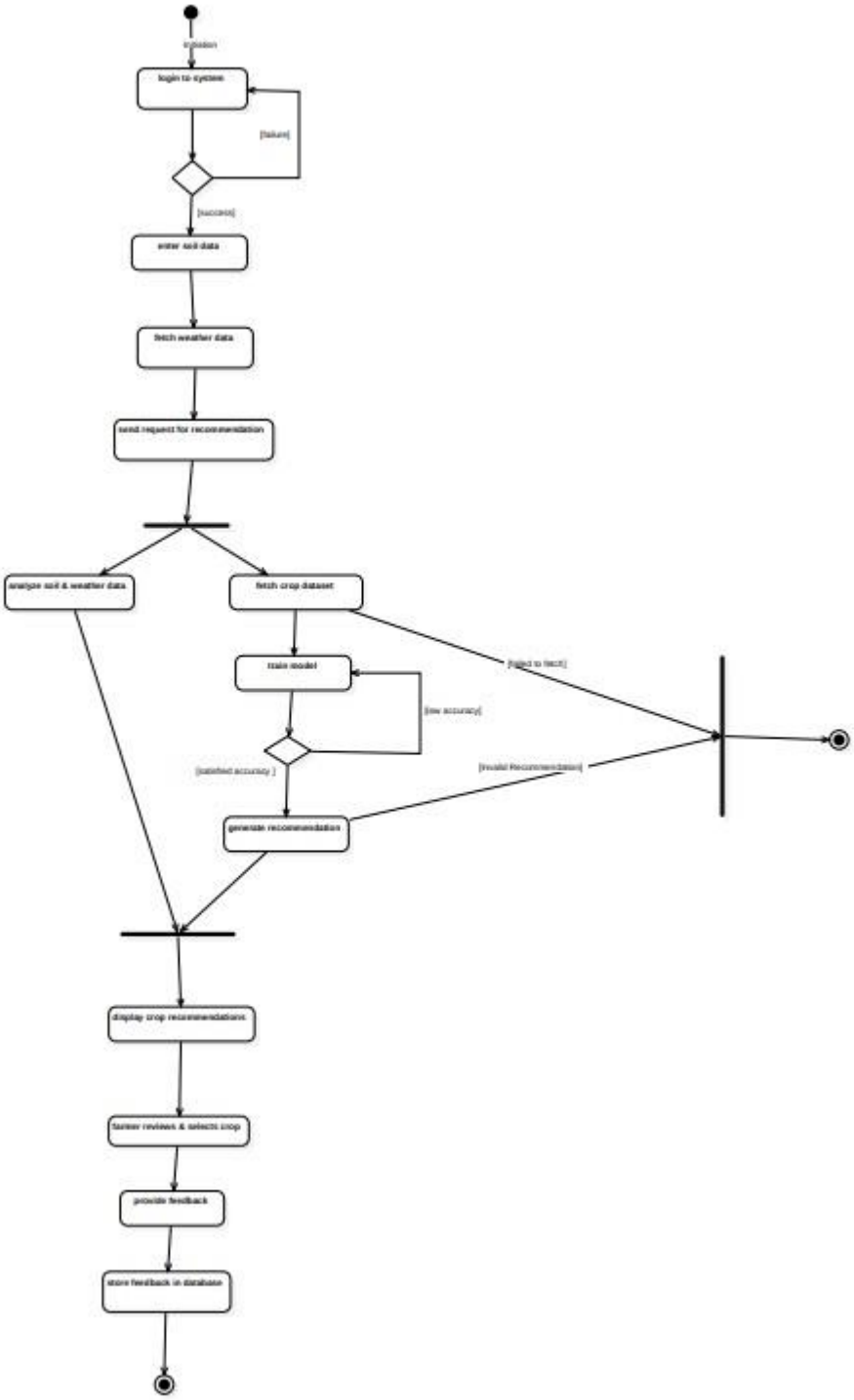


Fig 5.3.1

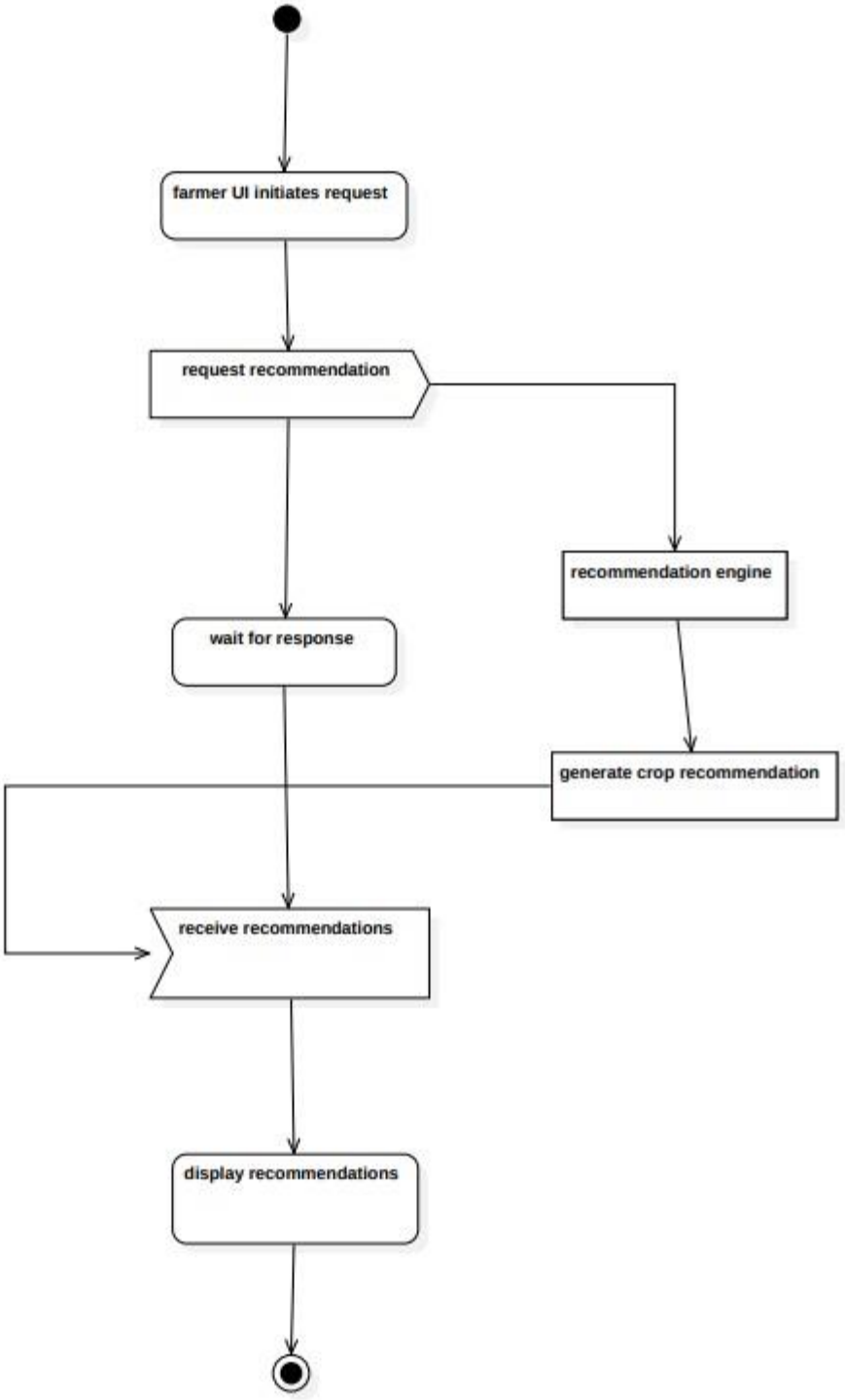

farmer UI initiates request
request recommendation
recommendation engine
wait for response
generate crop recommendation
receive recommendations
display recommendations

Fig 5.3.2

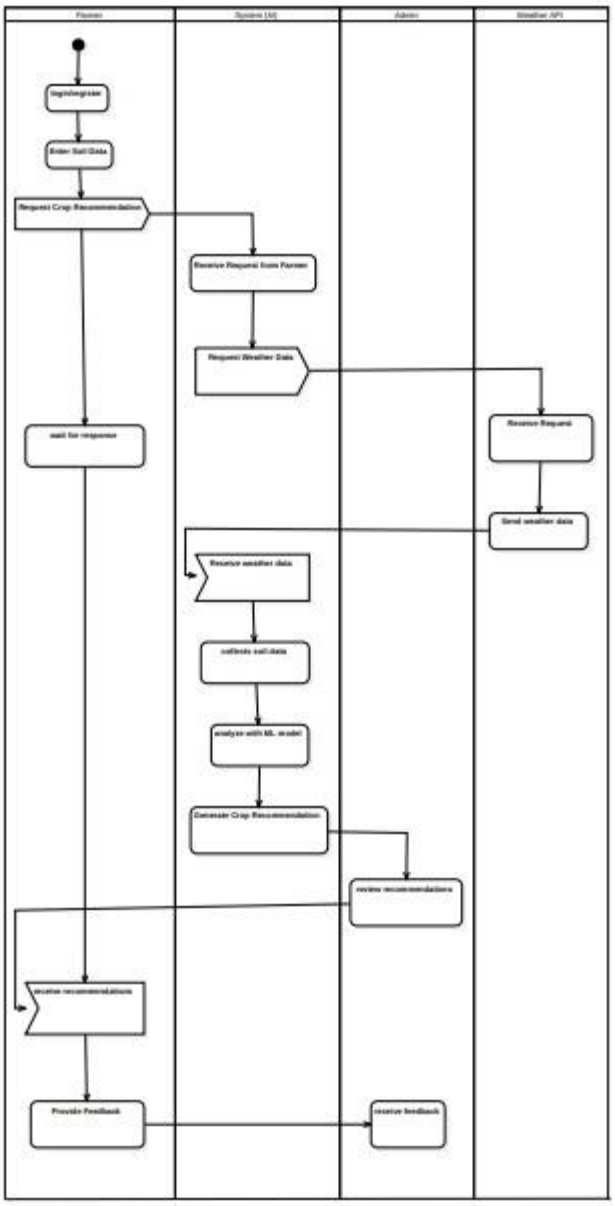

Farmer
System (AI)
Admin
Weather API
Login/register
Enter Soil Data
Request Crop Recommendation
Receive Request from Farmer
Request Weather Data
Receive Request
wait for response
Send weather data
Receive weather data
collects soil data
analyse with ML model
Generate Crop Recommendation
review recommendations
receive recommendations
Provide Feedback
receive feedback

Fig 5.3.2

Explanation:

The activity diagrams illustrate three main aspects of the system:

1. Complete System Workflow (Fig 5.3.1):

This swimlane diagram shows the interaction between four key actors: Farmer, System (AI), Admin, and Weather API. The process begins when a farmer logs in and sends soil data. The system then requests and receives crop recommendations, which involve requesting weather data from an external API. The workflow includes activities such as receiving weather data, collecting and analyzing data with an ML model, and generating crop recommendations. The admin component handles reviewing recommendations, while the system provides feedback to complete the cycle.

2. Recommendation Request Process (Fig 5.3.2):

This simplified diagram focuses on the farmer's interaction with the recommendation engine. It shows the sequential flow from when the farmer initiates a request through the UI, the system processes the request and waits for a response from the recommendation engine, receives the AI-generated recommendations, and finally displays them to the farmer.

3. Detailed Processing Logic (Fig 5.3.3):

This diagram illustrates the internal decision-making process of the system. It includes login authentication, input validation, weather data retrieval, decision nodes for handling various scenarios (such as valid/invalid accounts and data processing), and the final steps of displaying recommendations and providing feedback to farmers.

These activity diagrams collectively demonstrate how the AI-based system streamlines the crop recommendation process, integrating real-time weather data, machine learning algorithms, and user-friendly interfaces to support farmers in optimizing their agricultural practices.

Chapter 6: UI Design with Screenshots

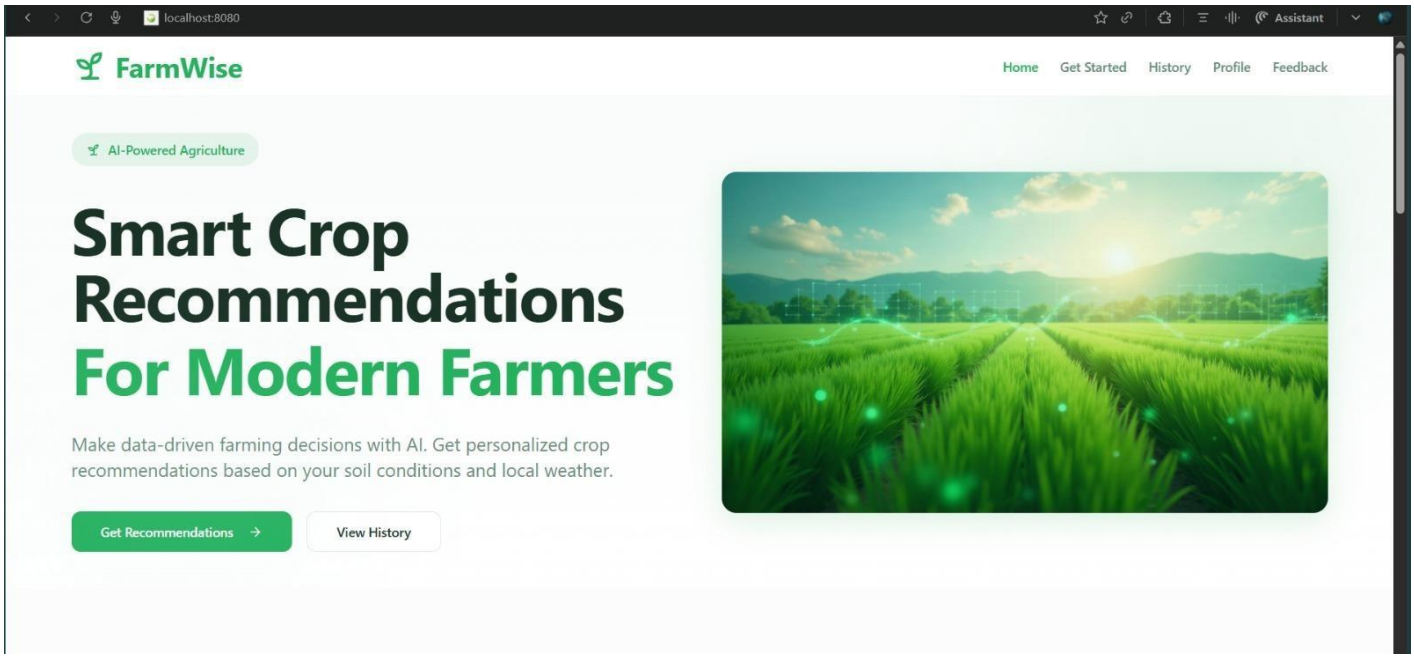


Fig 6.1

This figure shows the **home page** of the application, designed with a clean and aesthetic green theme to reflect modern agriculture. The hero section introduces the system's purpose — providing smart AI-based crop recommendations for farmers. It includes prominent action buttons for quick navigation such as "Get Recommendations" and "View History." A visually appealing hero image reinforces the concept of healthy, green farmlands supported by intelligent decision-making.

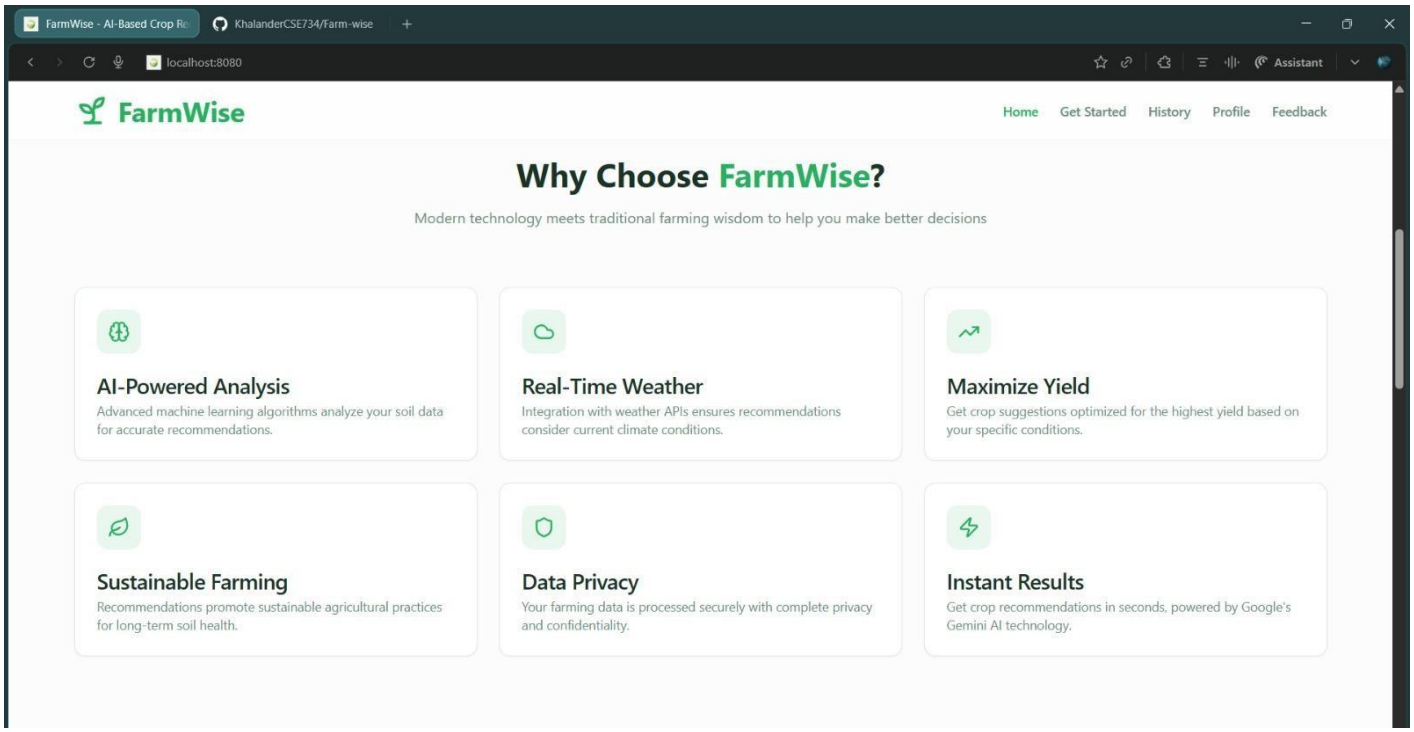


Fig 6.2

Figure 6.2 highlights the key features offered by the system in a structured card layout. Each feature card explains an important capability such as AI-powered analysis, real-time weather, sustainable farming, and maximizing crop yield. This section helps users quickly understand the benefits and scope of the platform. The minimal and modern UI improves readability and enhances user experience.

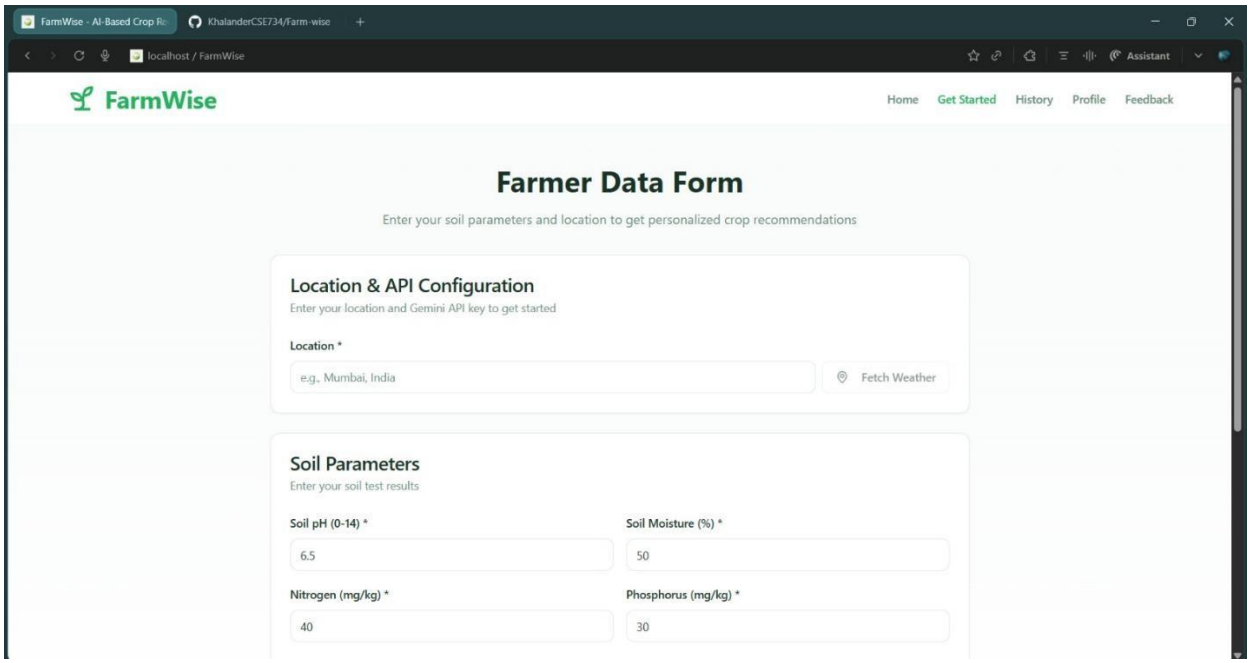


Fig 6.3

This figure displays the **Farmer Data Form**, where users can enter their location and soil parameters for analysis. The form includes fields such as soil pH, moisture, nitrogen, phosphorus, and potassium values. A dedicated button allows users to fetch real-time weather data based on the provided location. The layout is kept simple and intuitive for easy use by farmers and general users.

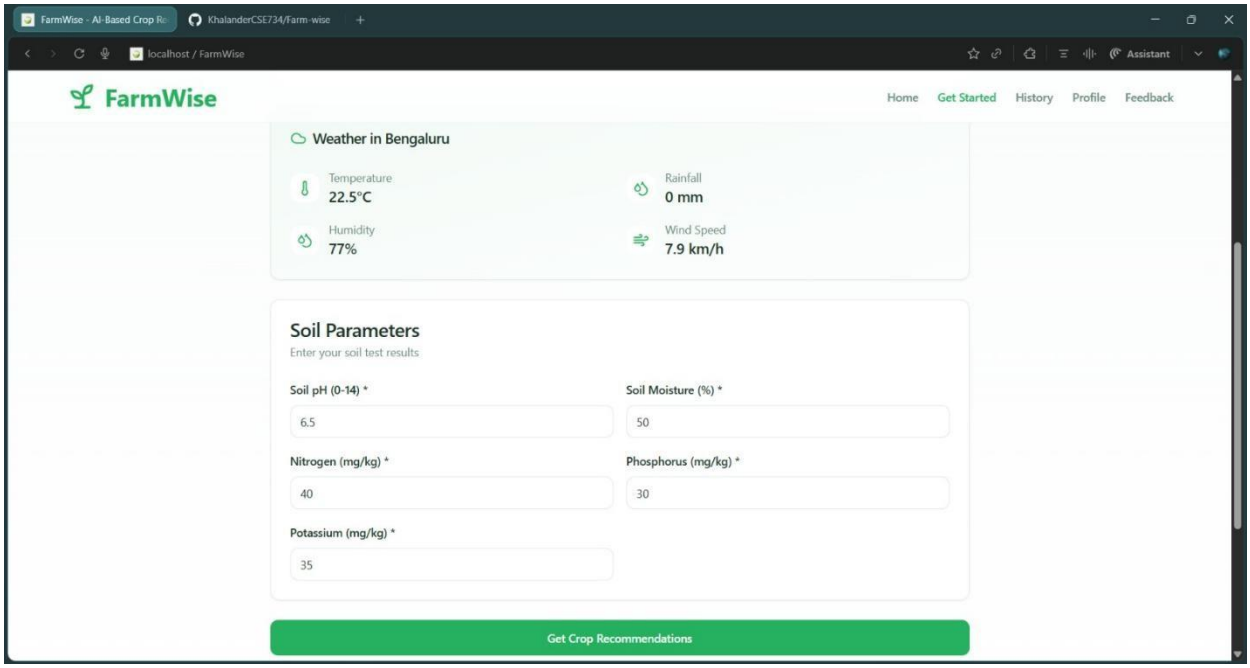


Fig 6.4

Figure 6.4 shows the same form after successfully fetching the real-time weather information. Weather details such as temperature, humidity, rainfall, and wind speed are automatically displayed. This enhances accuracy as farmers do not have to input weather conditions manually. The form now becomes fully ready for soil + weatherbased crop prediction using AI.

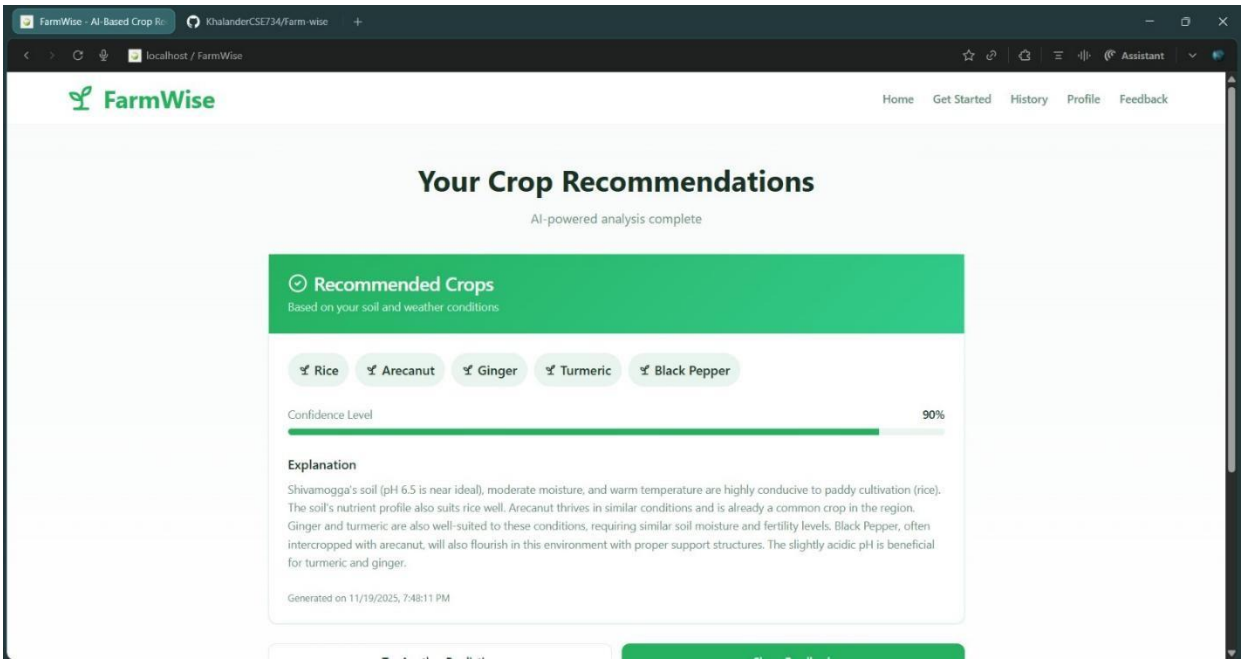


Fig 6.5

This figure presents the **AI-generated crop recommendations** returned by the Gemini model. The system displays a list of suitable crops along with a confidence score and explanation of the prediction. The result card maintains a clean green highlight to emphasize successful analysis. The explanation helps farmers understand why certain crops are more suitable for their soil and climatic conditions.

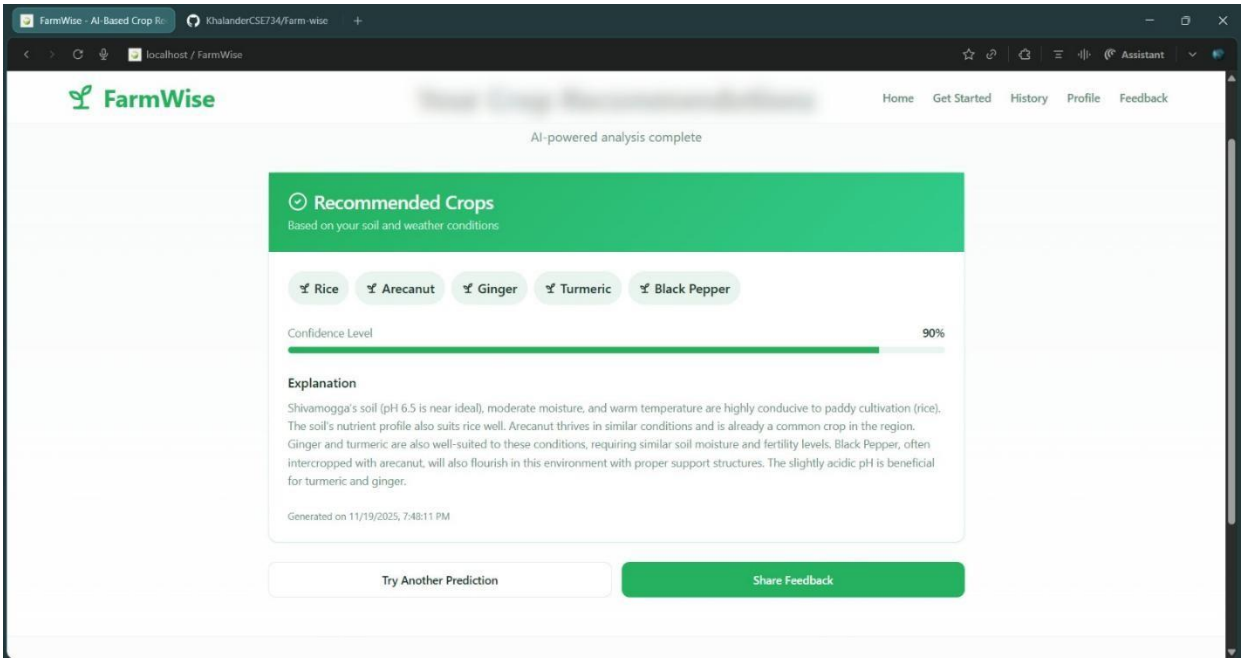


Fig 6.6

Figure 6.6 shows another version of the crop recommendation output, with an additional set of user actions such as “Try Another Prediction” and “Share Feedback.” The UI ensures smooth navigation for farmers to continue

using the system. The result layout and detailed explanation remain consistent, maintaining clarity and trust in the AI output. This section helps farmers make informed agricultural decisions.

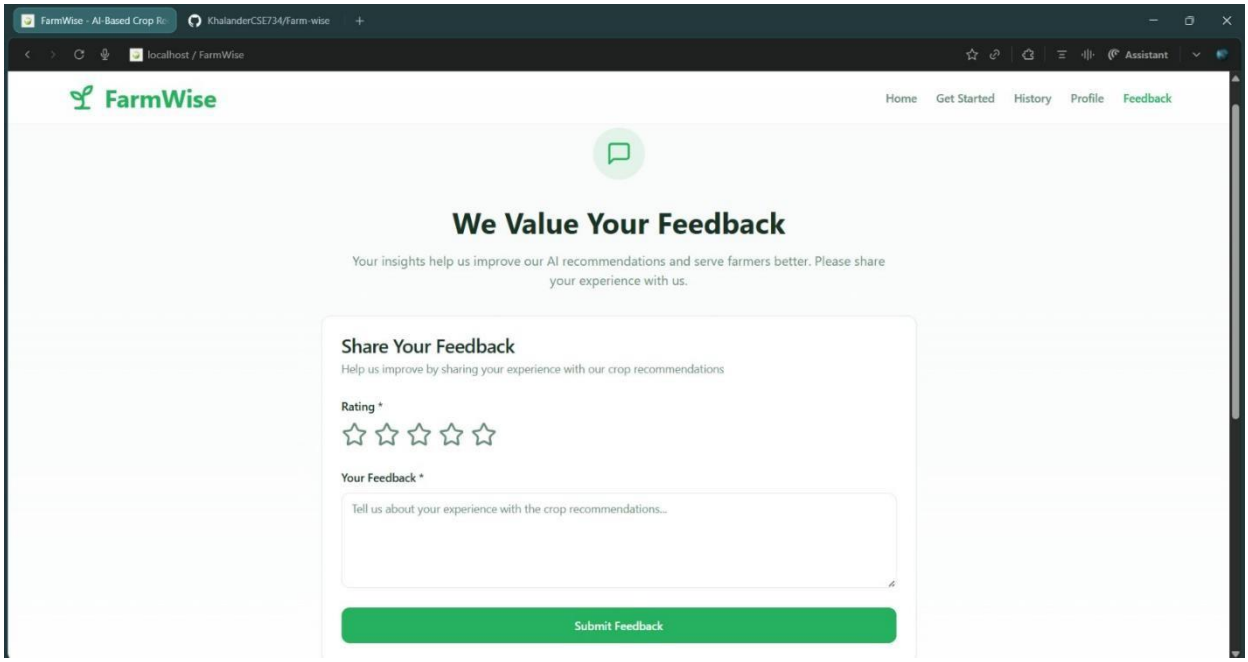


Fig 6.7

This figure displays the **Feedback Page**, where farmers can rate their experience and provide written comments. The interface includes a star-based rating component and a text box for detailed feedback. A clean and minimal card layout ensures simplicity and encourages users to share input. This feedback helps improve the system's predictions and overall user satisfaction.