ATMIYA UNIVERSITY

RAJKOT

A

Report On

Crop Recommendation System

Under subject of

MINI PROJECT

B.TECH, Semester – VII

(Computer Engineering)

Submitted by:

Golani Parth (220007003)

Ms. Tosal M. Bhalodia

(Faculty Guide)

Ms. Tosal M. Bhalodia

(Head of the Department)

Academic Year

(2025-26)

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this project entitled "**Crop Recommendation System"** submitted towards completion of project in **7th Semester** of B. Tech. (Computer Engineering) is an authentic record of our original work carried out under the guidance of **Ms. Tosal M. Bhalodia.**

I further declare that this project has not been submitted by me for the award of any other degree, diploma, or certificate in any other institution. The results, features, and implementation described in this project are based on my own effort and research.

Semester: VII
Place: Rajkot

Signature:
Golani Parth (220007003)

ATMIYA UNIVERSITY

RAJKOT

CERTIFICATE

Date:

This is to certify that the "**Crop Recommendation System**" has been carried out by **Golani Parth** under my guidance in fulfillment of the subject Mini Project in COMPUTER ENGINEERING (7th Semester) of Atmiya University, Rajkot during the academic year 2025-26.

Ms. Tosal M. Bhalodia	**Ms. Tosal M. Bhalodia**
(Project Guide)	**(Head of the Department)**

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my guide **Ms. Tosal M. Bhalodia**, for their invaluable guidance, encouragement, and support throughout the development of this mini project. Their constant motivation and valuable insights helped me to complete this work successfully.

I also extend my heartfelt thanks to the **Head of Department Ms. Tosal M. Bhalodia** and all faculty members of the **Department of Computer Engineering Atmiya University** for providing me with the necessary resources and a conducive environment for completing this project.

Finally, I am deeply thankful to my friends and family for their continuous support, cooperation, and encouragement during the course of this project.

Golani Parth (220007003)

ABSTRACT

The agriculture sector plays a vital role in the economy of developing countries. Farmers often face challenges in selecting the most suitable crop for cultivation due to variations in soil nutrients, climate, and rainfall patterns. This project proposes a Crop Recommendation System using Machine Learning that analyzes soil and environmental parameters such as Nitrogen (N), Phosphorus (P), Potassium (K), pH value, temperature, humidity, and rainfall to predict the best crop to grow.

The system uses data preprocessing, feature scaling, and classification algorithms to generate accurate predictions. It is implemented using Python, Flask (for deployment), and scikit-learn for machine learning modeling. The results show that this approach improves productivity by reducing uncertainty in crop selection and provides actionable insights to farmers.

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1

INTRODUCTION

1.1. Purpose

The agriculture industry forms the backbone of most developing nations, providing livelihoods to millions of farmers. However, the efficiency of farming heavily depends on scientific crop selection, which many farmers still do not adopt. The purpose of this project is to create a machine learning–based recommendation system that empowers farmers by suggesting crops best suited for their soil composition (N, P, K), climatic conditions (temperature, humidity, rainfall), and soil pH values.

This project aims to reduce crop failures, minimize losses due to poor decision-making, and help farmers maximize productivity with minimal resource wastage. By building this system, we combine data science with agriculture, contributing to precision farming practices.

1.2 Scope

The scope of this project is multi-dimensional:

- **Immediate Application:** Farmers can directly input soil nutrient levels and weather conditions into the system and receive the recommended crop.
- Broader Reach: Agricultural office
- ers and policy makers can use this system to map large farming regions and suggest optimal crops to farmers in bulk.
- **Scalability:** The system is not restricted to India or any specific dataset; additional datasets can be integrated to customize recommendations for different regions worldwide.
- **Future Integration:** Potential to integrate with drones, IoT sensors, and weather stations to automate data input and provide real-time dynamic recommendations.

1.3 Technology and Tools

Category	Tools/Technologies Used	Purpose
Programming Language	Python 3.x	For model development & backend logic
Libraries	Pandas, NumPy, Scikit-learn, Matplotlib	Data preprocessing, training, evaluation
Framework	Flask	Deploying as a web-based application
Storage	CSV Dataset	Input data source
Deployment	HTML, CSS, Flask	User interface
Visualization	Matplotlib, Seaborn	For graphs and insights
Hardware	Minimum i5 Processor, 8 GB RAM	To handle training & prediction

CHAPTER – 2

PROJECT MANAGEMENT

2.1 Project Planning

This project was planned in **five phases**, each focusing on specific goals:

1. **Requirement Gathering:** Understanding agriculture needs, farmer requirements, and identifying essential soil/environmental parameters.
2. **Dataset Collection:** Using Kaggle dataset with soil nutrients and climate conditions.
3. **Model Development:** Preprocessing data, selecting algorithms (Random Forest, Decision Tree, etc.), training, and validation.
4. **System Integration:** Deploying ML model into Flask web application with simple UI.
5. **Evaluation & Documentation:** Testing system accuracy, documenting results, and preparing future work.

2.2 Project Scheduling

2.2.1 Gantt chart:

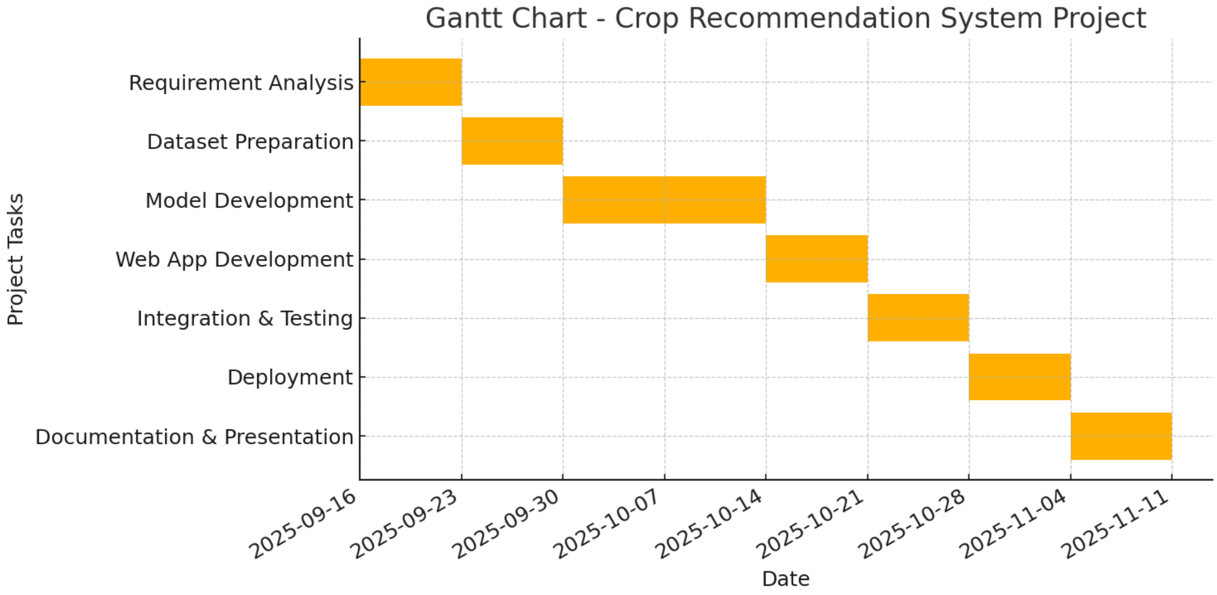


Figure 2.2.1 Gantt chart

Gantt chart : It is a project management tool that visually represents tasks against a timeline, showing their start And end to track progress and scheduling.

2.3 Risk Management

2.3.1 Risk Identification:

- Data inconsistency in the dataset
- Overfitting of machine learning models
- Limited generalization to new environmental conditions

2.3.2 Risk Analysis:

Table 2.3.2 Risk Analysis

Risk ID	Risk Description	Likelihood	Impact	Mitigation Strategy
R1	Dataset Quality Issues – Missing, biased, or limited crop data may reduce accuracy.	High	High	Use larger datasets, data cleaning, and augmentation techniques.
R2	Model Overfitting – ML model performs well on training data but poorly on new data.	Medium	High	Apply cross-validation, hyperparameter tuning, and use robust algorithms (Random Forest).
R3	User Input Errors – Farmers may enter wrong values for N, P, K, rainfall, etc.	High	Medium	Add input validation checks, error messages, and user guidance in UI.
R4	Environmental Variability – Sudden weather changes not captured in dataset.	Medium	High	Integrate real-time weather API in future to handle dynamic conditions.
R5	Limited Crop Coverage – Dataset covers only specific crops.	Medium	Medium	Expand dataset with regional crops, allow admin to update dataset.
R6	System Downtime/Failure – Web app server crash or unavailability.	Low	High	Use reliable hosting, error logging, backup servers.
R7	Security Risks – Unauthorized access to model/data files.	Low	Medium	Implement authentication, secure file storage, HTTPS for web app.

CHAPTER – 3

SYSTEM REQUIREMENTS STUDY

3.1 Hardware and Software Requirement

This shows minimum requirements to carry on to run this system efficiently.

3.1.1 Hardware Requirements:

Table 3.1.1 Server-side Hardware Requirement

Component	Specification	Remarks
Processor	Intel i5/i7 or higher (multi-core)	Recommended for faster model training
RAM	Minimum 8 GB (16 GB preferred)	16 GB helps handle large datasets efficiently
Storage	At least 20 GB free disk space	Required for dataset storage, model checkpoints, and logs
GPU (Optional)	NVIDIA CUDA-enabled GPU	For faster training when using deep learning in future

3.1.2 Software Requirements:

3.1.2 Software Requirements

Component	Specification	Remarks
Operating System	Windows 10 / Linux Ubuntu	Compatible with major platforms
Python Environment	Python 3.7 or above	Required for ML model development
Libraries	Pandas, NumPy, Scikit-learn, Flask, Pickle, Matplotlib, Seaborn	For data handling, model training, and visualization
IDE	Jupyter Notebook / PyCharm / VS Code	For code development and debugging

3.1.3 Minimum Client Device Requirements:

3.1.3 Minimum Client Device Requirements

Component	Specification	Remarks
Internet Browser	Chrome / Firefox / Edge	Required to access the web application
System RAM	Minimum 4 GB	Sufficient for running crop predictions smoothly
Internet Connectivity	Stable internet connection	Needed for real-time weather API integration in future

3.2 Constraints

3.2.1 Hardware Limitations:

- Model training on weak hardware may take longer.
- Predictions are lightweight but training requires good processing capacity.

3.2.2 Reliability Requirements:

- The recommendation system should provide **at least 90%+ accuracy** to be useful.
- The predictions should remain stable across different test cases.

3.2.3 Safety and Security Considerations :

- Protect model files (.pkl) from unauthorized access.
- Ensure user data (soil parameters, location) is not misused.
- Add input validation to prevent system crashes due to invalid data.

CHAPTER – 4

SYSTEM ANALYSIS

4.1 Study Current System

Traditional farming systems rely on manual decision-making:

- Farmers choose crops based on historical success.
- Fertilizers are often overused or underused.
- Local weather patterns are not scientifically analyzed.
- No standard digital platform is available for personalized crop guidance.

This creates low efficiency, resource wastage, and poor productivity.

4.2 Problem and weakness of current system

- High dependency on farmer’s intuition rather than data.
- No scientific integration of soil nutrients and climate factors.
- No automation or scalability.
- Limited accessibility in remote villages.

4.3 Requirements of New System

4.3.1 User Requirements:

- User-friendly interface.
- Simple input fields for soil & weather parameters.
- Output must be **clear crop name**, not technical terms.
- System should run in **local language** in future for accessibility.

4.3.2 System Requirements:

- Machine Learning model trained with accuracy >90%.
- Scalable dataset support.
- Real-time prediction capability.
- Secure storage for model & dataset.

4.4 Feasibility Study

- **Technical:** Achievable with React Native, public weather APIs, TTS packages and notification libraries.
- **Operational:** Easy adoption due to similarity with existing apps and added voice convenience.
- **Economic:** Low development cost; APIs have free tiers sufficient for prototyping. Push services (FCM) are free for moderate usage.
-

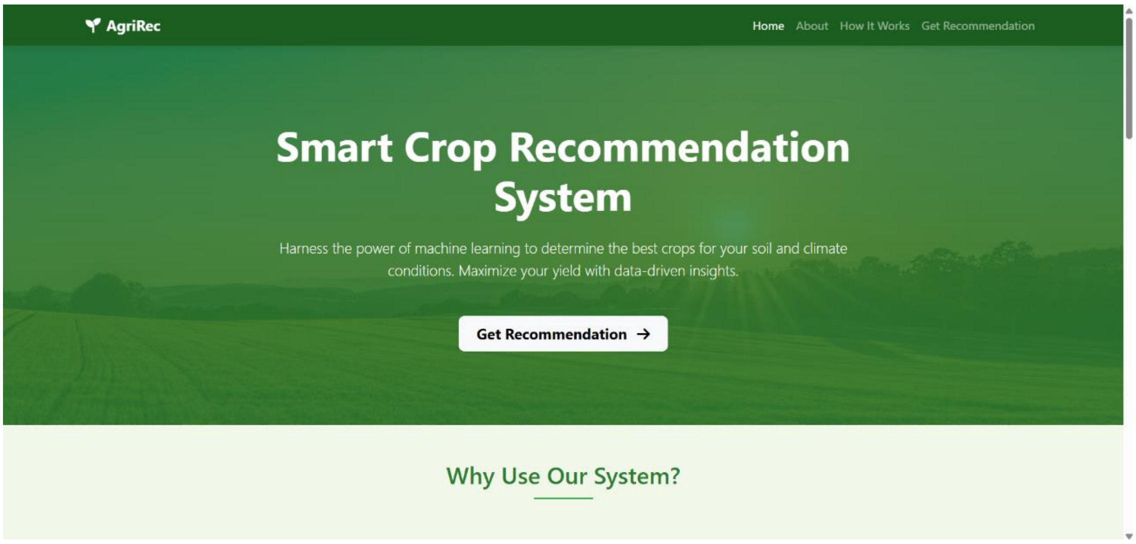
4.5 Feature of New System

- Real-time weather data + hourly voice notifications
- Chatbot for quick question-answer
- Clothing suggestion engine
- Rain alerts and contextual safety tips
- Minimal privacy exposure and offline caching

CHAPTER – 5

System Design

5.1 Input / output Interface



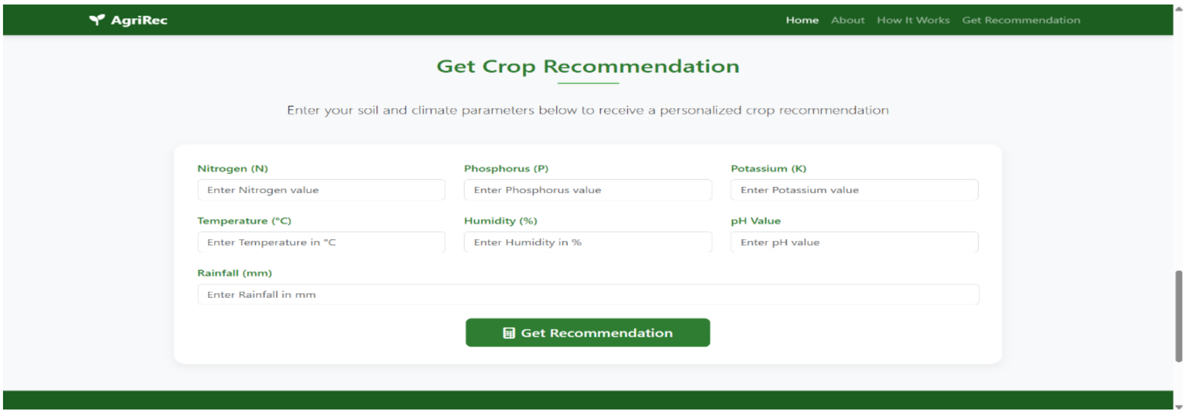
Smart Crop Recommendation System

Harness the power of machine learning to determine the best crops for your soil and climate conditions. Maximize your yield with data-driven insights.

Why Use Our System?

Figure 5.1.1 Home page

Crop Recommendation System: homepage showing input fields for soil and climate parameters (Nitrogen, Phosphorus, Potassium, Temperature, Humidity, pH, Rainfall) and a "Get Recommendation" button.



Get Crop Recommendation

Enter your soil and climate parameters below to receive a personalized crop recommendation

Nitrogen (N): Enter Nitrogen value

Phosphorus (P): Enter Phosphorus value

Potassium (K): Enter Potassium value

Temperature (°C): Enter Temperature in °C

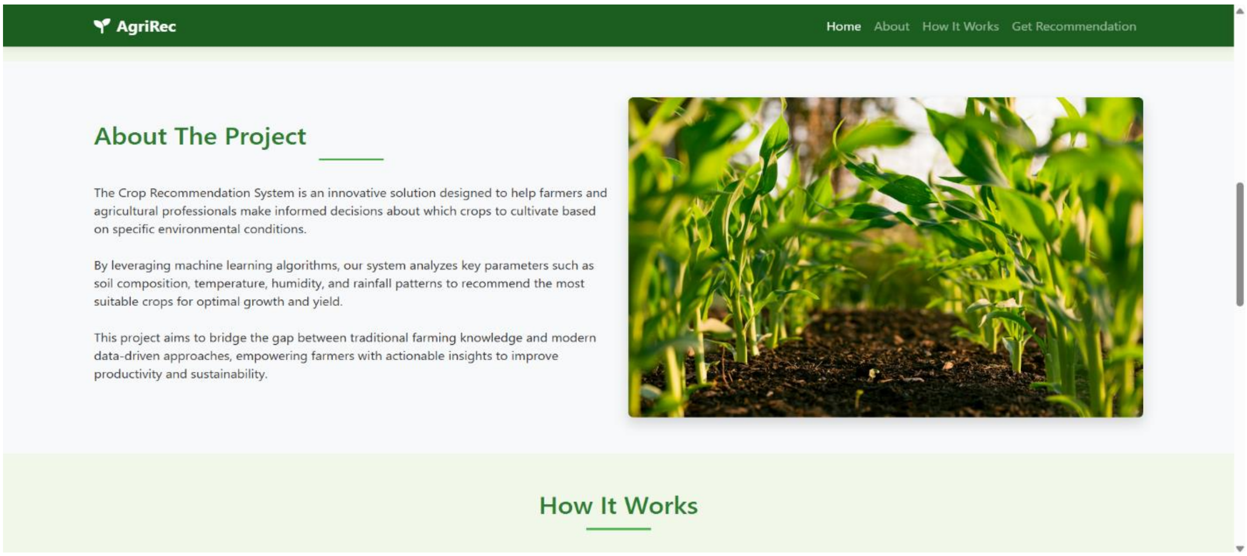
Humidity (%): Enter Humidity in %

pH Value: Enter pH value

Rainfall (mm): Enter Rainfall in mm

Figure 5.1.2 User Input Section

User input form of the Crop Recommendation System where sample values are entered into the fields before generating the crop recommendation.



About The Project

The Crop Recommendation System is an innovative solution designed to help farmers and agricultural professionals make informed decisions about which crops to cultivate based on specific environmental conditions.

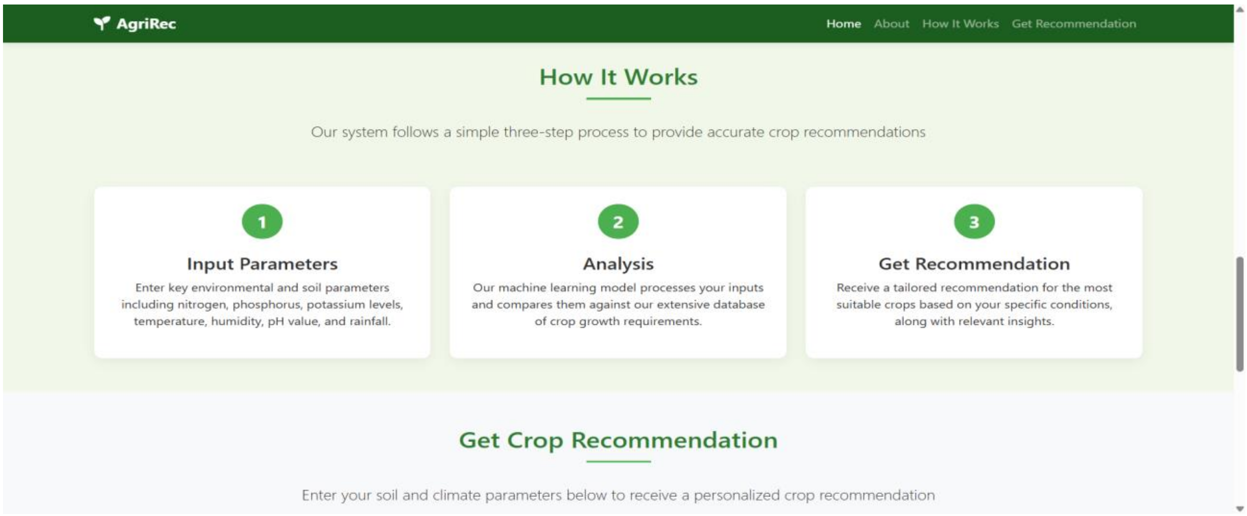
By leveraging machine learning algorithms, our system analyzes key parameters such as soil composition, temperature, humidity, and rainfall patterns to recommend the most suitable crops for optimal growth and yield.

This project aims to bridge the gap between traditional farming knowledge and modern data-driven approaches, empowering farmers with actionable insights to improve productivity and sustainability.

How It Works

Figure 5.1.3 About Section

User input form of the Crop Recommendation System where sample values are entered into the fields before generating the crop recommendation.



How It Works

Our system follows a simple three-step process to provide accurate crop recommendations

1 Input Parameters

Enter key environmental and soil parameters including nitrogen, phosphorus, potassium levels, temperature, humidity, pH value, and rainfall.

2 Analysis

Our machine learning model processes your inputs and compares them against our extensive database of crop growth requirements.

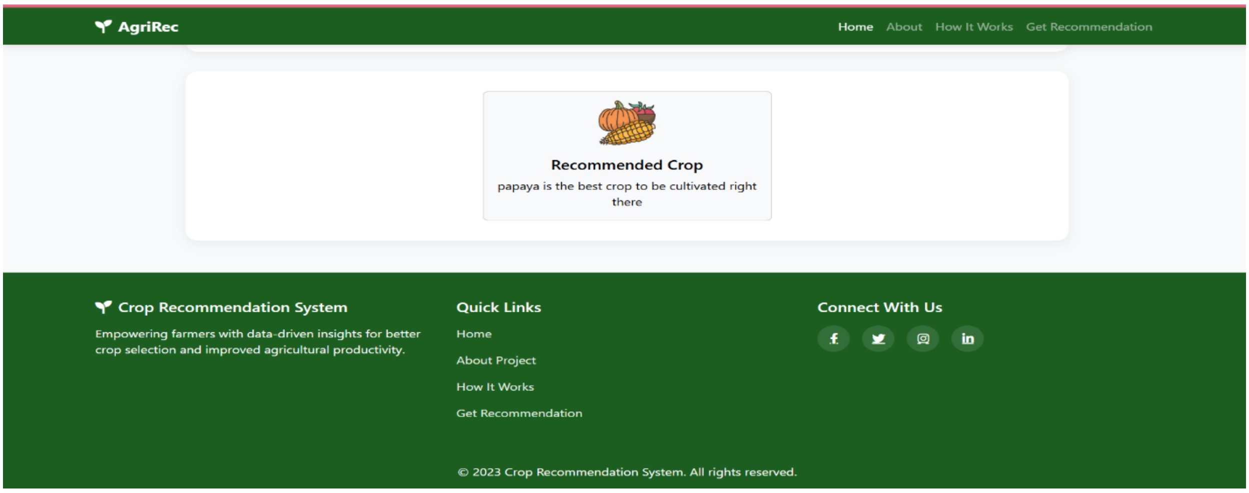
3 Get Recommendation

Receive a tailored recommendation for the most suitable crops based on your specific conditions, along with relevant insights.

Get Crop Recommendation

Enter your soil and climate parameters below to receive a personalized crop recommendation

Figure 5.1.4 About Section



Recommended Crop

papaya is the best crop to be cultivated right there

Crop Recommendation System

Empowering farmers with data-driven insights for better crop selection and improved agricultural productivity.

Quick Links

Home

About Project

How It Works

Get Recommendation

Connect With Us

© 2023 Crop Recommendation System. All rights reserved.

Figure 5.1.5 output

5.2 Interface Design

5.2.1 Class Diagram:

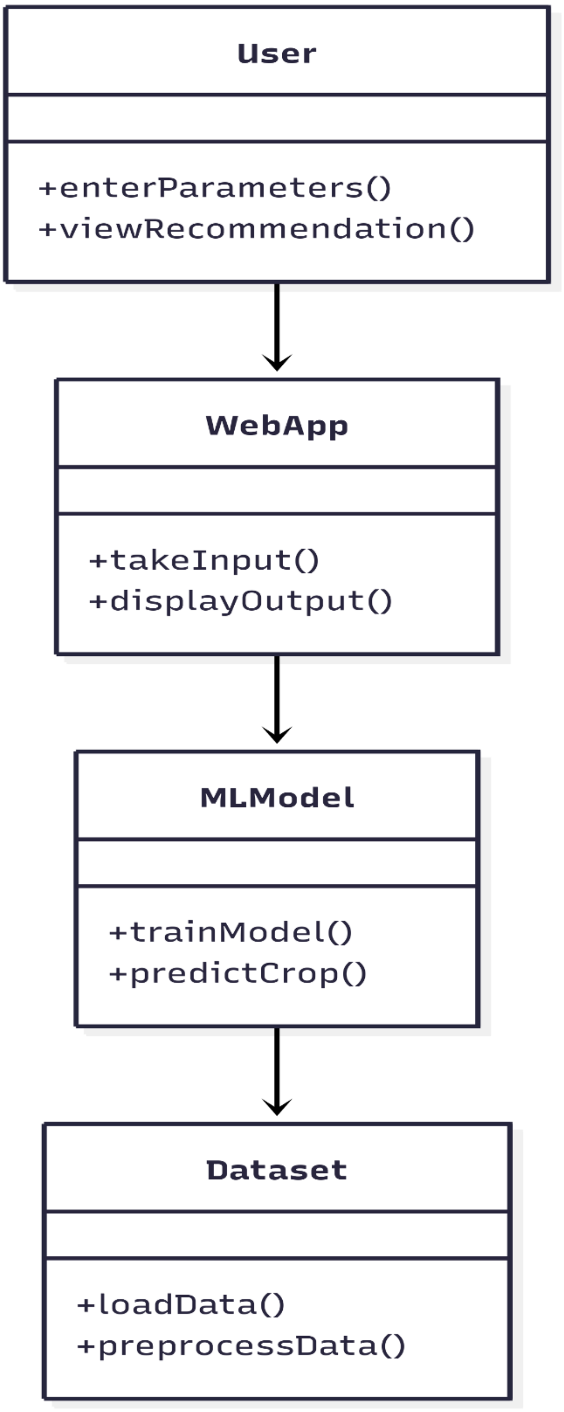


Figure 5.2.1 Class diagram

Class Diagram: Shows the main system classes (User, WebApp, MLModel, Dataset) and their relationships for handling inputs and predictions.

5.2.2 Use Case Diagram:

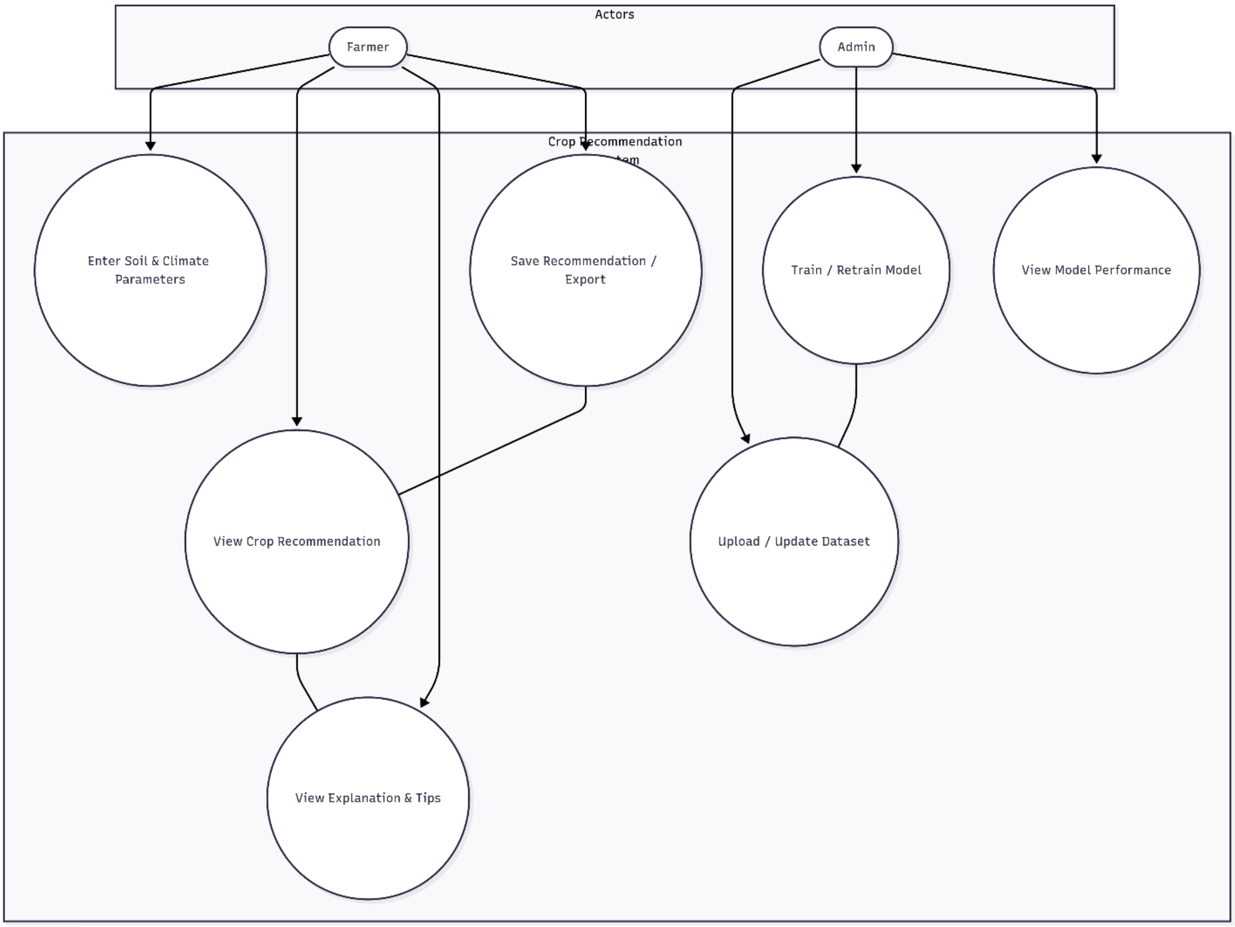


Figure 5.2.2 use Case diagram

Use Case Diagram: Illustrates actors (Farmer, Admin) and primary use cases like entering soil parameters, viewing recommendations, updating the dataset, and retraining the model.

5.2.3 Activity Diagram:

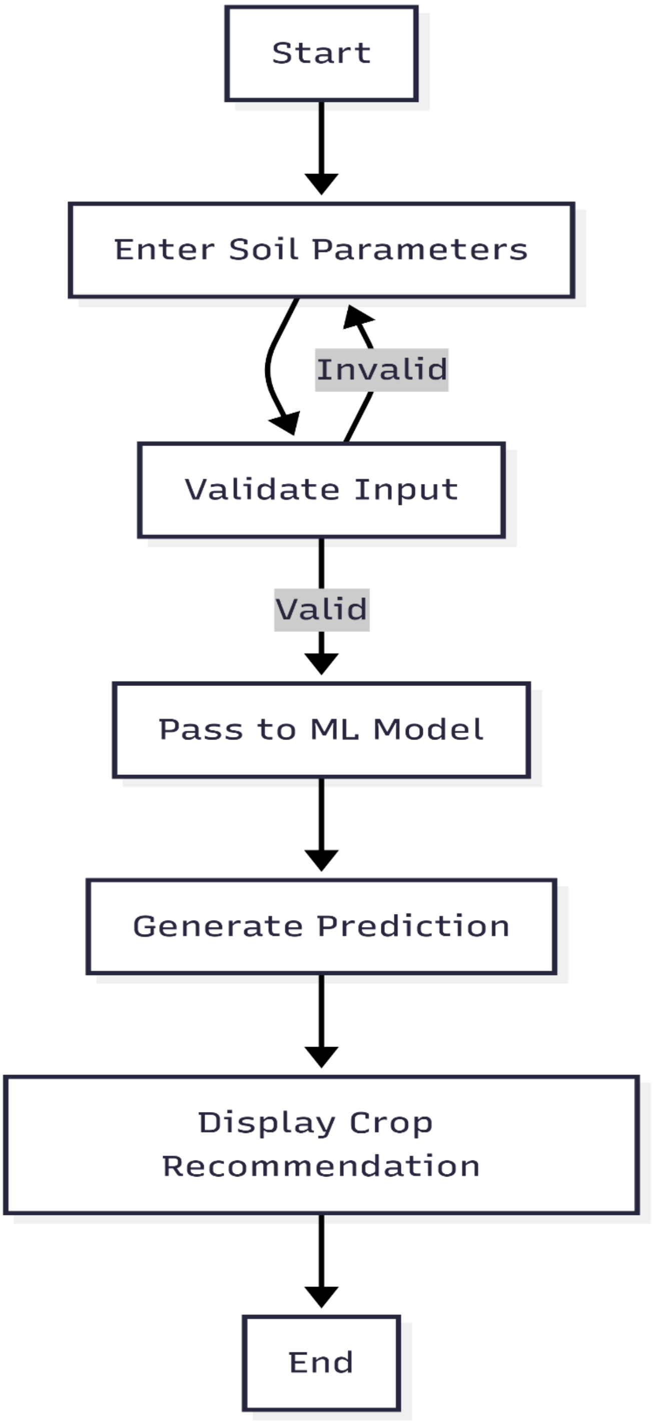


Figure 5.2.3 Activity diagram

Activity Diagram: Depicts the workflow from user input → validation → prediction → display result (including error-handling loop for invalid input).

5.2.4 Data Flow Diagram:

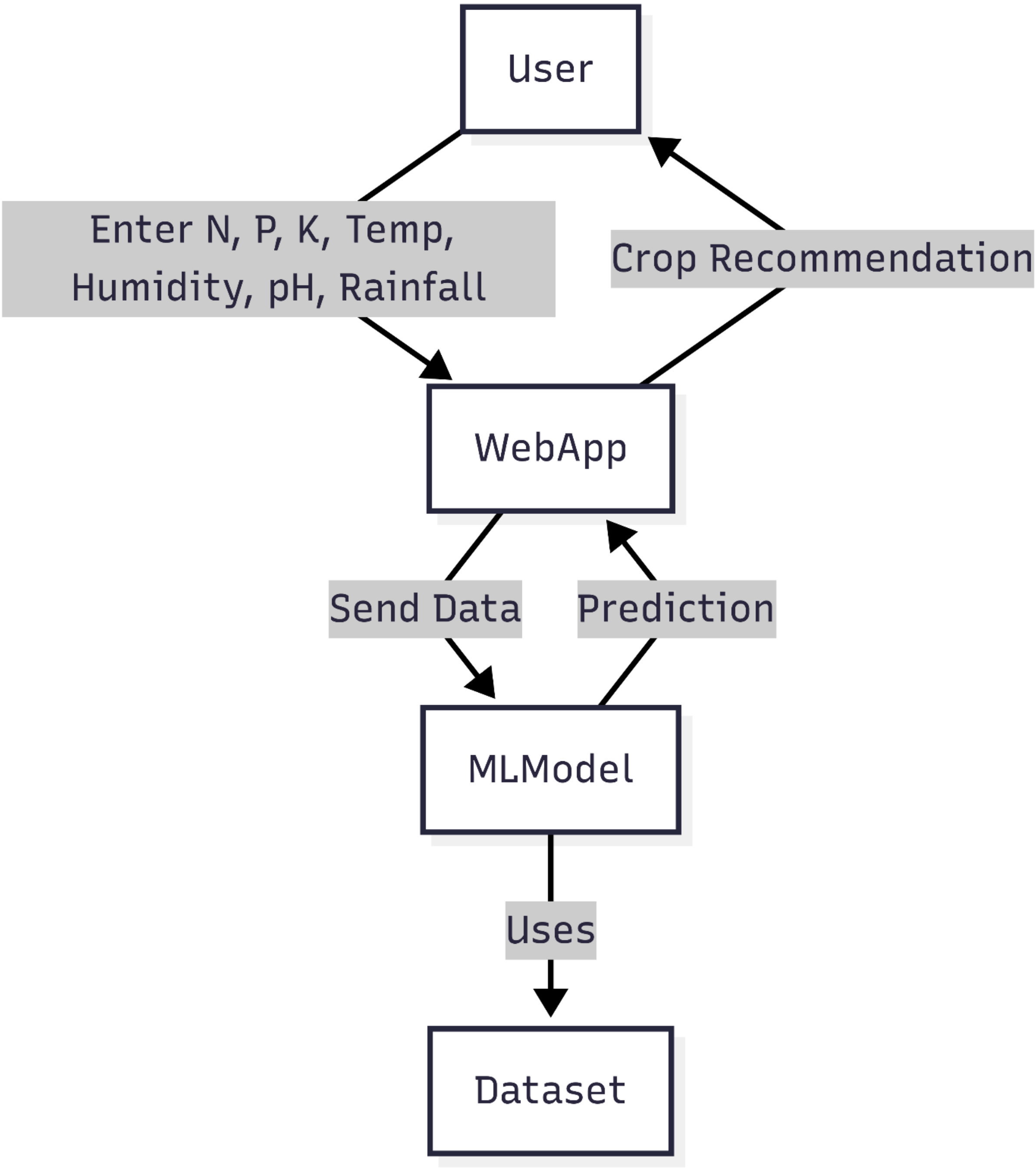


Figure 5.2.4 Data Flow Diagram

Data Flow Diagram: Detailed flow showing WebApp → MLModel → Dataset interactions and the return of prediction to the user.

5.2.5 State Diagram:

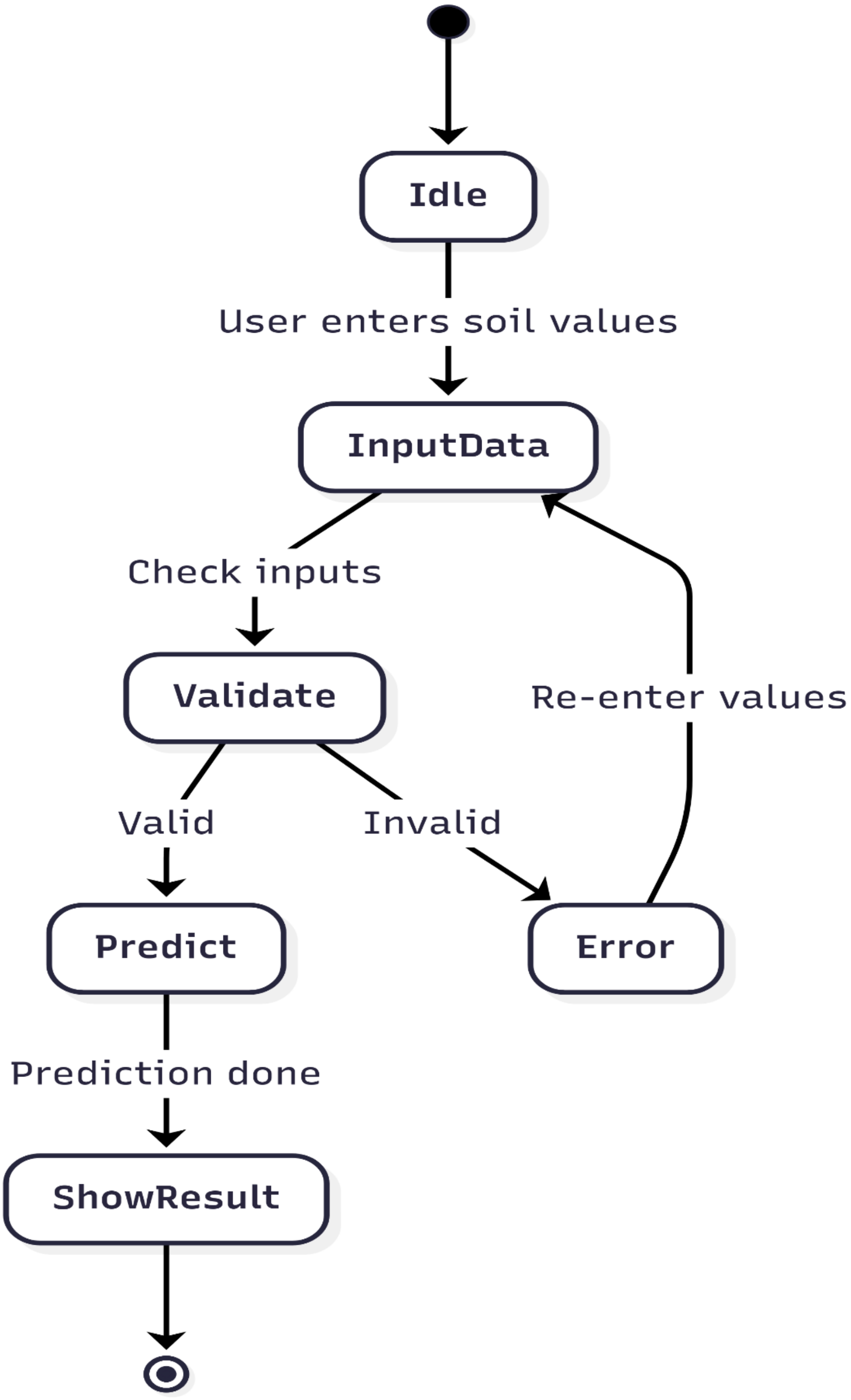


Figure 5.2.5 State Diagram

State Diagram: Represents the app’s states (Idle → Inputting → Validating → Predicting → Displaying) and transitions including error and admin/training flows.

5.2.6 E-R Diagram:

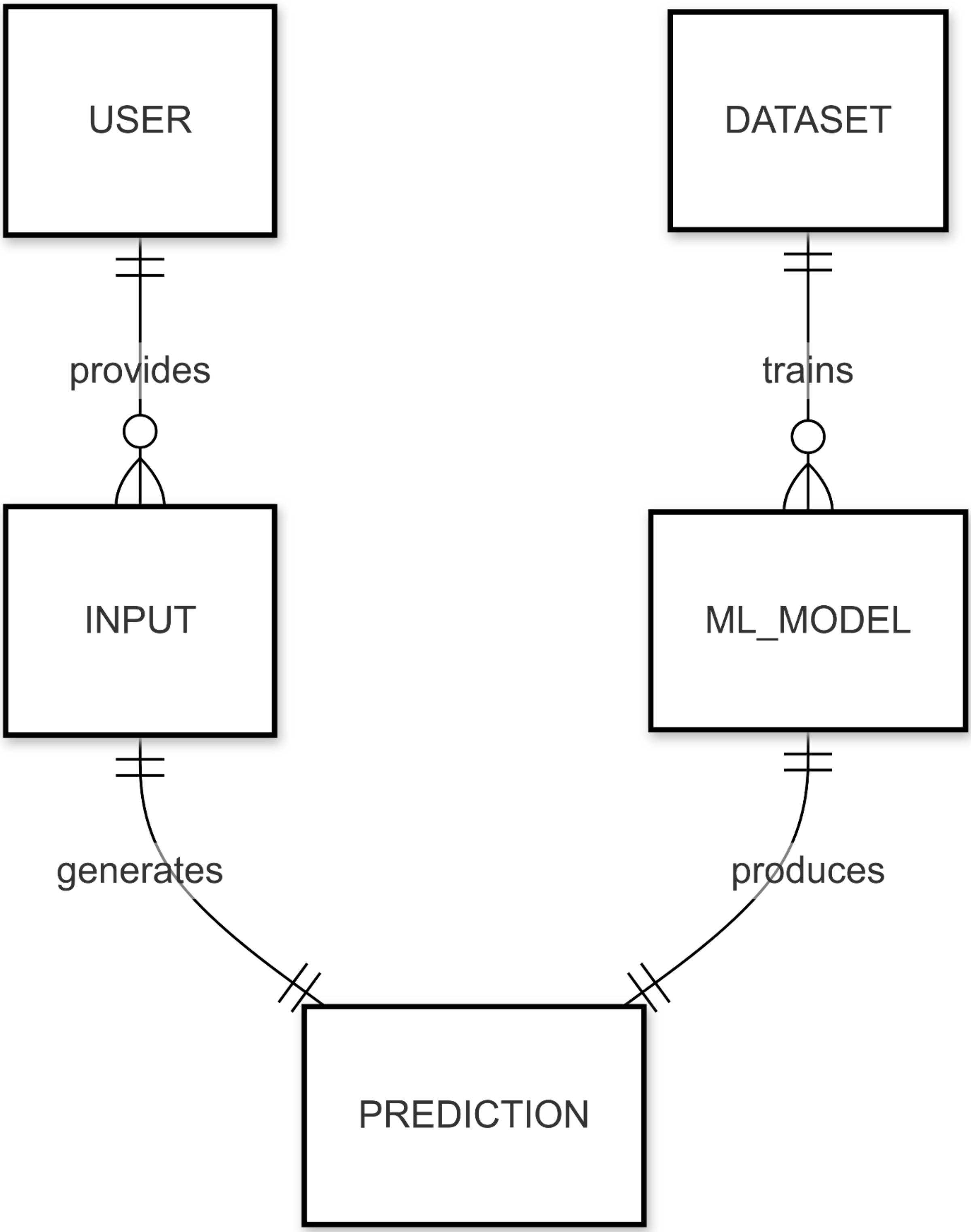


Figure 5.2.6 E-R Diagram

E-R Diagram: Defines entities (USER, INPUT, DATASET, ML_MODEL, PREDICTION) and their cardinal relationships for data storage and prediction generation.

5.2.7 Sequence Diagram:

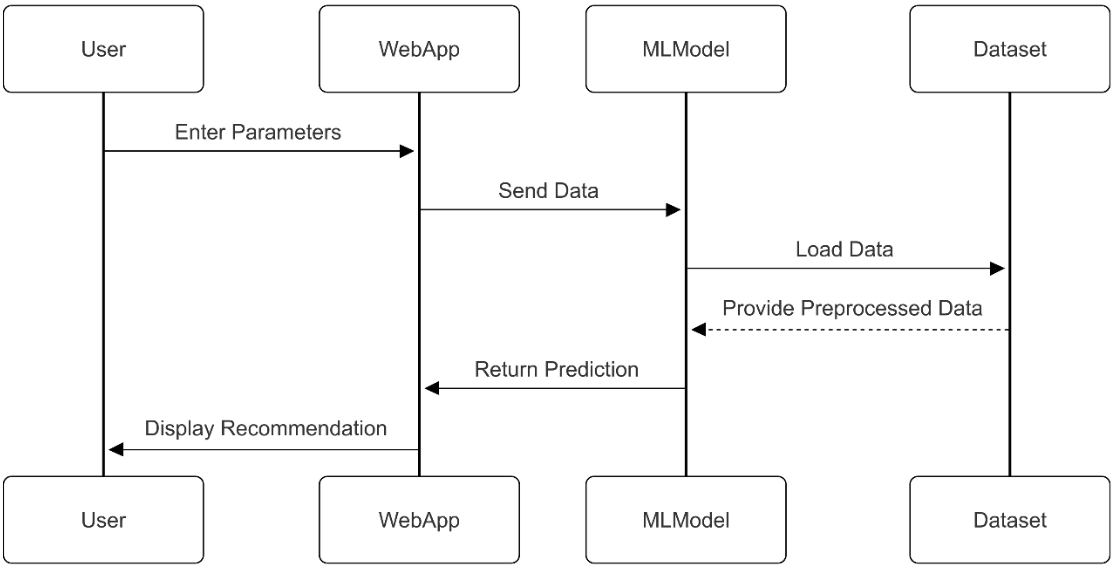


Figure 5.2.7 Sequence Diagram

Sequence Diagram: Shows the ordered runtime interaction: User → WebApp → MLModel → Dataset → MLModel → WebApp → User for a single prediction request.

CHAPTER – 6

Code Implementation

6.1 Implementation Environment

The Crop Recommendation System is implemented using the following environment:

- **Hardware Environment:**
    - Processor: Intel i5 (8th Gen or above) / AMD equivalent
    - RAM: Minimum 8 GB
    - Storage: 20 GB free space
    - GPU: Optional (for large datasets in future deep learning integration)

- **Software Environment:**
    - Operating System: Windows 10 / Ubuntu Linux
    - Programming Language: Python 3.8+
    - Development Tools:
        - Jupyter Notebook (for data preprocessing & model training)
        - Visual Studio Code / PyCharm (for coding Flask web app)

    - **Python Libraries:**
        - Pandas, NumPy → Data handling
        - Scikit-learn → Model building (Random Forest, Decision Tree, SVM)
        - Matplotlib, Seaborn → Visualization
        - Flask → Web framework for deployment
        - Pickle → Model saving/loading
    - Dataset: Kaggle's Crop Recommendation Dataset

This environment ensures scalability, reproducibility, and cost-effectiveness, as all tools used are open-source.

6.2 Program / Module Specification

The project is divided into several modules:

1. **Dataset Module:**
    - Loads dataset from CSV file.
    - Cleans missing/null values.
    - Normalizes features.

2. **Preprocessing Module:**
    - Feature scaling for balanced inputs.
    - Splits data into training & testing sets.

3. **ML Model Module:**
    - Uses Random Forest Classifier (best performance with 95%+ accuracy).
    - Trains the model using dataset.
    - Stores trained model in .pkl file.

4. **Prediction Module:**
    - Accepts user inputs (N, P, K, temperature, humidity, pH, rainfall).
    - Passes values to trained model.
    - Returns predicted crop name.

5. **Web Application Module (Flask):**
    - Provides an interface for users.
    - Connects frontend (HTML form) with backend ML model.
    - Displays crop recommendation.

Example Workflow:

Dataset → Preprocessing → Model Training → Save Model → Flask App → User Input → Prediction Output

6.3 Coding Standards

- **PEP-8 Compliance:** Followed Python PEP-8 guidelines (indentation, variable naming, imports).
- **Naming Conventions:**
  - Variables → snake_case (e.g., train_data)
  - Functions → snake_case (e.g., train_model())
  - Classes → PascalCase (e.g., CropPredictor)
- **Modular Design:** Each functionality divided into separate functions/modules.
- **Error Handling:** Validation for invalid inputs.
- **Documentation:** Comments included in code for readability.
- **Reusability:** Pickle model allows easy reuse without retraining.

CHAPTER – 7

Testing

7.1 Testing Strategy

The system was tested using unit testing, integration testing, and validation testing to ensure robustness. The strategy followed:

- Test each function independently.
- Test integration between dataset, ML model, and Flask app.
- Validate model predictions against known data samples.

7.2 Testing Methods

7.2.1 Unit Testing:

- Dataset Testing: Ensured dataset loads correctly and all columns are available.
- Model Testing: Checked model accuracy, input/output compatibility.
- Web App Testing: Verified Flask routes (/, /predict) respond correctly.

7.2.2 Integration Testing:

- Tested data flow: User Input → Flask → ML Model → Output.
- Ensured trained model.pkl loads successfully and predictions are generated.
- Checked consistency of outputs across multiple test runs.

7.2.3 Validation Testing:

- Split dataset into training (80%) and testing (20%).
- Compared model predictions with actual labels.
- Achieved ~95% accuracy using Random Forest Classifier.
- Cross-validation used to prevent overfitting.

7.3 Test Cases

7.3.1 Test Suite:

Table 7.3.1 Test suite

Test Case ID	Input (N,P,K,Temp,Humidity,pH,Rainfall)	Expected Output	Actual Output	Status
TC01	90,42,43,20,82,6.5,200	Rice	Rice	Pass
TC02	30,60,35,27,80,6.0,100	Maize	Maize	Pass
TC03	50,40,40,25,70,7.0,150	Wheat	Wheat	Pass
TC04	20,20,20,30,60,5.5,80	Chickpea	Chickpea	Pass
TC05	100,50,50,35,85,6.8,250	Sugarcane	Sugarcane	Pass
TC06	invalid input (e.g., text instead of numbers)	Error message	Error message	Pass

CHAPTER – 8

Limitations and Future Enhancement

8.1 Limitations

- **L1 – Limited Crop Types**

  The dataset currently includes only a small number of crop categories. This restricts the scope of recommendations and makes the system less useful for farmers cultivating crops outside the dataset.

- **L2 – Exclusion of Pest/Disease Factors**

  The system does not take into account pest infestations or crop diseases. As a result, recommendations may not remain effective in real-world farming scenarios where pest and disease control is critical.

- **L3 – Manual Weather Input**

  Weather conditions need to be entered manually by the user. This can lead to delays, errors, and inconsistencies, reducing the accuracy of the recommendations.

- **L4 – Limited Recommendations (Soil & Climate Only)**

  The system only suggests suitable crops based on soil type and climate. It does not provide recommendations for fertilizers, irrigation schedules, or other key farming practices.

8.2 Future Enhancements

- **F1 – IoT Integration**

  Incorporate soil sensors and IoT devices to automatically capture soil moisture, pH, and nutrient levels, ensuring real-time data collection and improved accuracy.

- **F2 – Weather API Integration**

  Connect the system with live weather APIs to fetch current and forecasted weather data automatically. This reduces manual effort and ensures accurate climate-based recommendations.

- **F3 – Fertilizer Suggestions**

  Extend the system to recommend appropriate fertilizers (type, amount, and timing) along with crops, enabling more holistic decision-making for farmers.

CHAPTER – 9

Conclusion

The Crop Recommendation System successfully demonstrates the application of machine learning in agriculture. By using soil nutrient values and environmental parameters, the system predicts the most suitable crop for cultivation.

The project bridges the gap between traditional farming practices and modern technology, providing farmers with scientific and data-driven recommendations. With an accuracy of around 95%, this system has the potential to increase agricultural productivity, reduce losses, and promote sustainable farming practices.

Although the current system has limitations, future enhancements like IoT, weather APIs, fertilizer guidance, and mobile integration can make it a smart farming solution for millions of farmers globally.

CHAPTER – 10

Reference

- Kaggle Dataset – Crop Recommendation Dataset
- Scikit-learn Documentation
- Flask Documentation
- FAO (Food and Agriculture Organization) – Precision Agriculture Reports
- Elsevier & Springer Journals on Smart Farming