SMART CROP RECOMMENDATION SYSTEM

SOFTWARE REQUIREMENTS SPECIFICATION

VERSION 1

5 MAY,2025

PREPARED BY

KHUSHI THAKUR (05319051623)

KRITI RASTOGI (20119051623)

UNIVERSITY SCHOOL OF AUTOMATION AND ROBOTICS, GGSIPU EDC

1. Introduction

1.1 Purpose

The objective of this document is to analyse the capability of ML methods for crop prediction using a collected dataset obtained with crop simulation models and to specify the requirements of a web-based Crop Yield Prediction application. The system will predict the most suitable crop to cultivate based on soil and environmental parameters like pH, rainfall, temperature, nitrogen (N), phosphorus (P), and potassium (K) levels using trained ML models.

1.2 Scope

This application will help farmers and agricultural planners make informed decisions. It provides a user-friendly interface for inputting environmental data and utilizes machine learning to recommend optimal crops.

1.3 Definitions, Acronyms, and Abbreviations

- ML: Machine Learning
- UI: User Interface
- DBMS: Database Management System
- N, P, K: Nitrogen, Phosphorus, Potassium
- SVM: Support Vector Machine

1.4 References

1. Ansarifar, J., Wang, L., & Archontoulis, S. V. (2023). An interaction regression model for crop yield prediction. Scientific Reports. Retrieved from https://doi.org

2. Crop Recommendation Dataset. Retrieved from Kaggle. https://www.kaggle.com/code/theeyeschic o/crop-analysis-and-prediction/notebook

3. K-Nearest Neighbours. GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/k-nearest neighbours/

4. Support Vector Machine Algorithm. GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/support vector-machine-algorithm/

5. Creating Linear Kernel SVM in Python. GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/creating linear-kernel-svm-in-python/

6. Machine Learning: Random Forest Algorithm. Javatpoint. Retrieved from https://www.javatpoint.com/machine learning-random-forest-algorithm

7. Technical Advances in Plant Science. (2023). Frontiers in Plant Science, 14. Retrieved from https://doi.org/10.3389/fpls.2023.1128388

2. Overall Description

2.1 Product Perspective

- The system shall be a standalone web application.
- It shall use a Streamlit-based frontend.
- It shall connect to machine learning models for crop prediction.
- It shall not rely on external software beyond standard libraries and hosting services.

2.2 Product Functions

- The system shall accept soil and environmental input parameters.
- It shall process the input data using trained ML models.
- It shall predict the most suitable crop for given conditions.
- It shall display the predicted crop and confidence level.

2.3 User Classes and Characteristics

- **Farmers**:
    - Shall have basic technical skills.
    - Shall input data to receive crop suggestions.
- **Researchers**:
    - Shall have advanced technical understanding.
    - Shall analyze prediction logic and performance.

2.4 Operating Environment

- The system shall be web-based.
- It shall be developed using Streamlit.
- It shall be hosted on Streamlit Cloud (or similar).
- Backend shall be in Python using scikit-learn, pandas, and numpy.
- It shall be accessible via modern web browsers.
- It shall require no installation by the user.

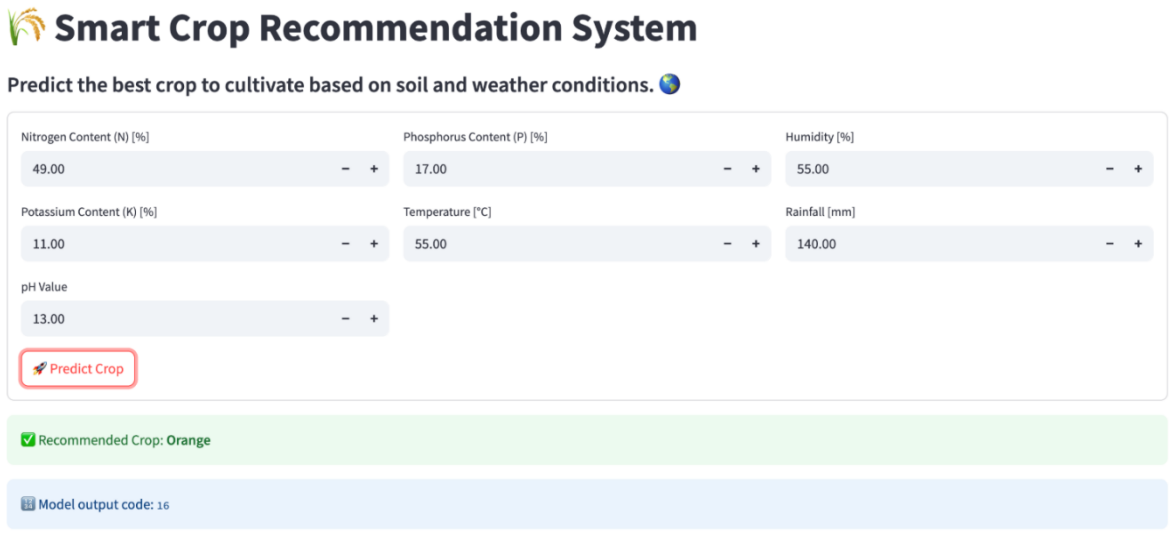
2.5 Design and Implementation Constraints

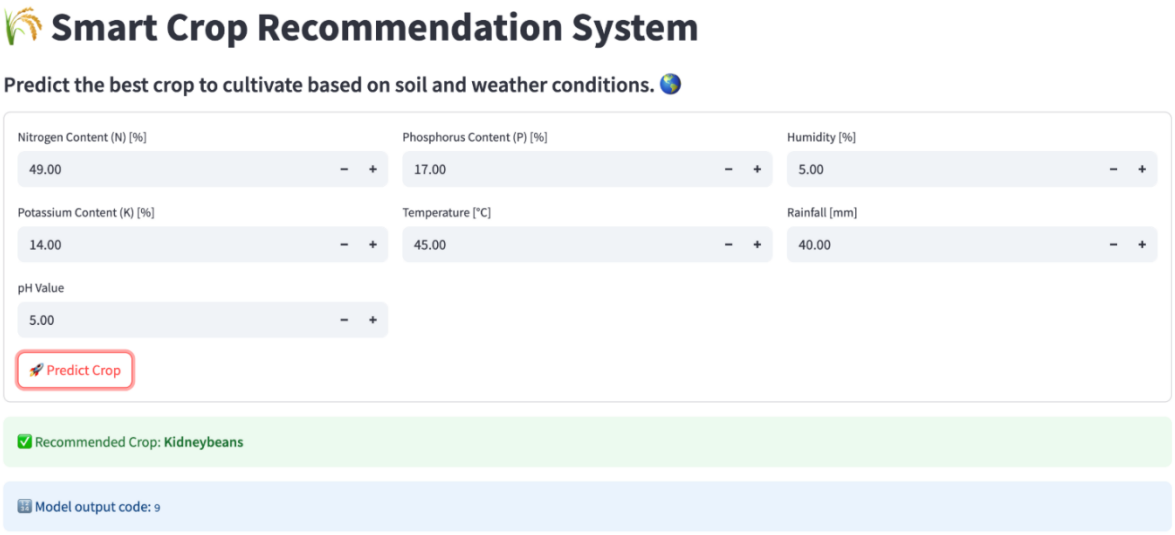
- The system shall be limited to the available training dataset.
- It shall require a stable internet connection.

3. Specific Requirements

3.1 User Interface

The system has a simple and responsive web interface, built using Streamlit. It includes input forms, dropdowns for soil/weather parameters, and a clean display of the predicted crop along with accuracy.





3.2 Authentication and Authorization

Although basic login/signup is not mandatory for the current use case, the system can be further designed to support user authentication with email verification in future updates

3.3 User Management
Users will able to submit data without registration and receive predictions instantly. Future versions may include user history and profile-based recommendations.

3.4 Performance Requirements
The system must return predictions in under 3 seconds, even during peak load. The application should remain responsive and fluid regardless of input size.

3.5 Security Requirements
User data will be handled securely. Secure login, HTTPS access, and API protection will be ensured. Sensitive data like user input logs or admin panels will be accessible only to authorized personnel.

3.6 Database Requirements
A relational database will store user submissions, crop prediction logs, and any associated metadata. The database will be optimized for read-heavy operations.

3.7 Maintenance and Support
The system will support regular updates for both the machine learning model and the UI/UX. A bug tracking mechanism will be in place, and logs will be maintained to monitor performance and errors.

4. External Interface Requirements

4.1 User Interfaces
Web interface using HTML, CSS, JavaScript.

4.2 Hardware Interfaces
No specific hardware; works on standard devices**.**

4.3 Software Interfaces

- **Streamlit**: Used to build and host the web application UI
- **Python**: Core language for logic, processing, and ML model integration
- **scikit-learn**: For crop prediction using trained machine learning models
- **pandas**: For handling user inputs and tabular data processing
- **numpy**: For numerical computations and data transformation
- **Matplotlib**: For data visualization if graphs are included
- **Browser Interface**: Google Chrome, Firefox, or any modern web browser for accessing the app

4.4 Communication Interfaces

- **HTTP/HTTPS Protocols**: Used for client-server communication via web browser
- **Streamlit Internal API**: Handles interaction between frontend (UI) and backend (Python logic) within the same app

- **Form-based data submission**: Inputs are sent to the backend via Streamlit forms and processed instantly
- **Bi-directional interaction**: User inputs lead to dynamic outputs (crop recommendation + accuracy shown in real-time)

5. Non-Functional Requirements

5.1 Usability

The system provides an intuitive and user-friendly interface with easy navigation. It can be adapted to include localized content in regional languages if needed.

5.2 Reliability

The system is expected to be operational 99% of the time, ensuring consistent availability and minimal downtime.

5.3 Scalability

Designed to efficiently handle an increasing number of users and larger datasets without compromising performance or accuracy.

5.4 Availability

Accessible 24/7 via web browsers, with minimal interruptions. Scheduled maintenance will be planned during off-peak hours.

5.5 Response Time

Crop predictions are expected to be delivered within 2–3 seconds of submitting the input data.

5.6 Data Backup and Recovery

Daily backups of the crop data and user submissions will be taken. The system can recover fully from any failure within 24 hours.

5.7 Accessibility

The interface will be keyboard-navigable and compatible with screen readers, ensuring inclusivity for users with visual or motor impairments.

5.8 Compatibility

Supports all modern browsers and is fully functional on both desktop and mobile devices for maximum reach.

6. System Models

6.1 Use Case Diagram

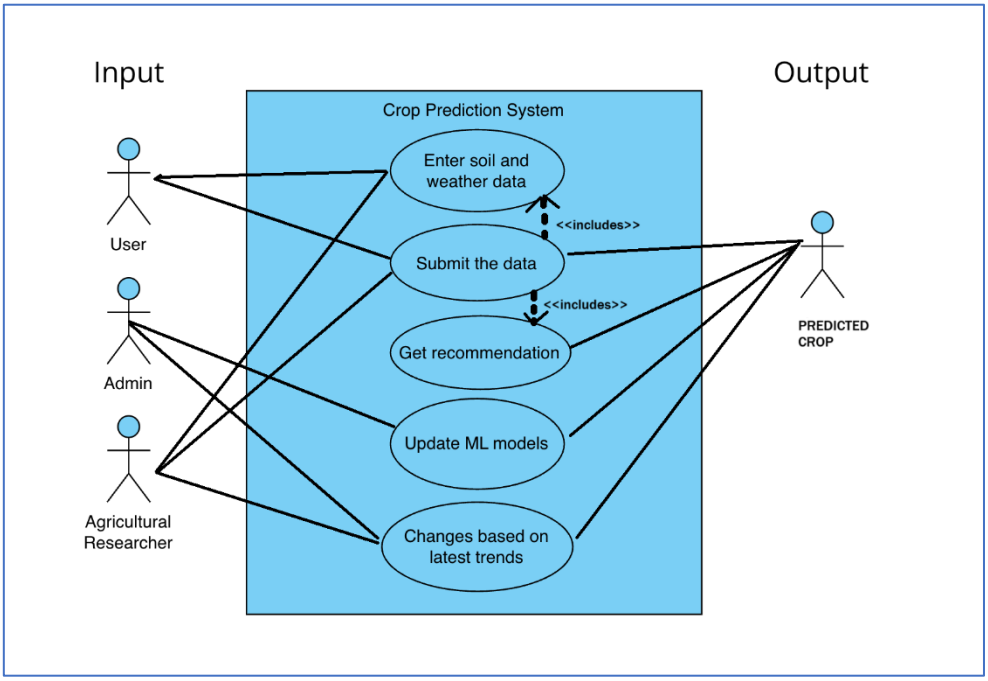


Fig: use case diagram for smart crop recommendation system

PRE REQUISITES

A. Introduction

- The system enables users to input environmental data and receive crop recommendations.
- Admins and researchers can update the model based on recent agricultural trends.

B. Actors

- **User**: Provides input and receives crop prediction.
- **Admin**: Oversees model updates and manages submissions.
- **Agricultural Researcher**: Updates models based on trends and contributes expert data.
- **Predicted Crop (System Output Actor)**: Represents the output interface.

C. Pre-Conditions

- User is authenticated.
- Input values are ready.
- ML model is trained and deployed.

D. post-conditions

- Suitable crop is predicted and displayed.
- User's input and prediction are stored.

- Model is updated.

E. Flow of Events

E.1 Basic Flow

- User enters soil and weather data.
- User submits data.
- System processes and recommends crop.

E.2 Alternative Flows

- Admin updates ML model.
- Researcher modifies model using trend data.
- If data is invalid, error message shown.
- If model unavailable, fallback or delay.

F. Special Requirements

- Only Admins and Researchers can access update functionality.
- Submission must include all required data fields.
- Recommendations are to be generated within 2–3 seconds.
- Changes in trends must trigger validation before model updates.

G. Use Case Relationships

- "Submit the Data" includes "Enter Soil and Weather Data"
- "Get Recommendation" includes "Submit the Data"

6.2 Class Diagram

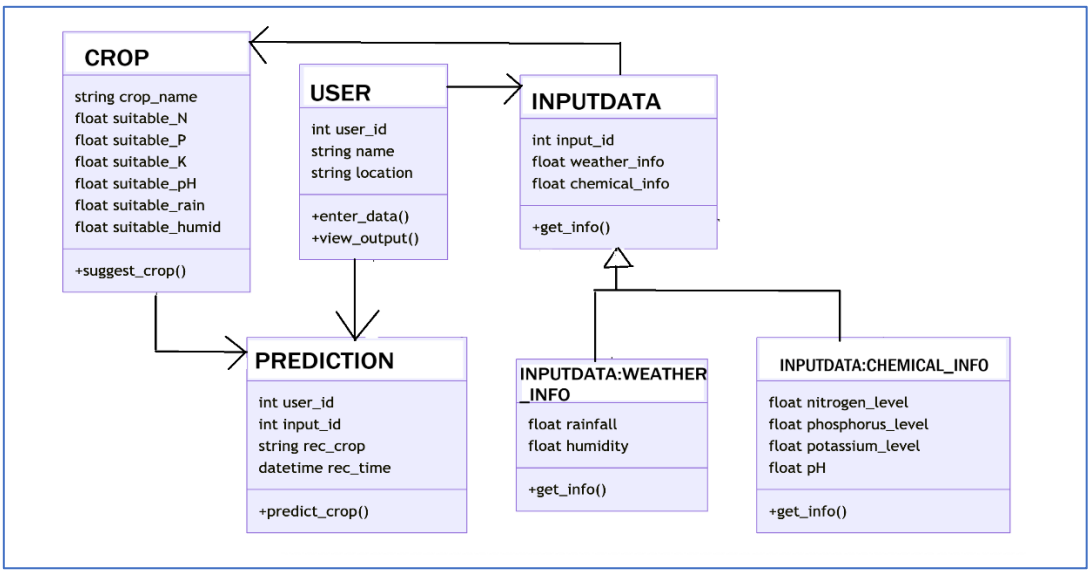


Fig: Class diagram for smart crop recommendation system

PRE REQUISITES

A. Introduction

- Represents the static structure of the system, showing classes like User, InputData, Crop, and Prediction, along with their relationships.

B. Pre-Conditions

- Valid user input available which is further specified (chemical + weather data)
- Crop database must be defined
- ML model must be integrated with prediction logic

C. Post-Conditions

- Predicted crop is returned
- Prediction and input are stored in system

D. Flow of Events (Implied by Classes)

- User calls enter_data()
- InputData and its subclasses collect values
- Prediction uses input to call predict_crop()
- Crop class returns matching crop using suggest_crop()

E. Special Requirements

- Input must match expected data types (float, int)
- Unique IDs for each user and input set

6.3 Entity-Relationship Diagram (ERD)

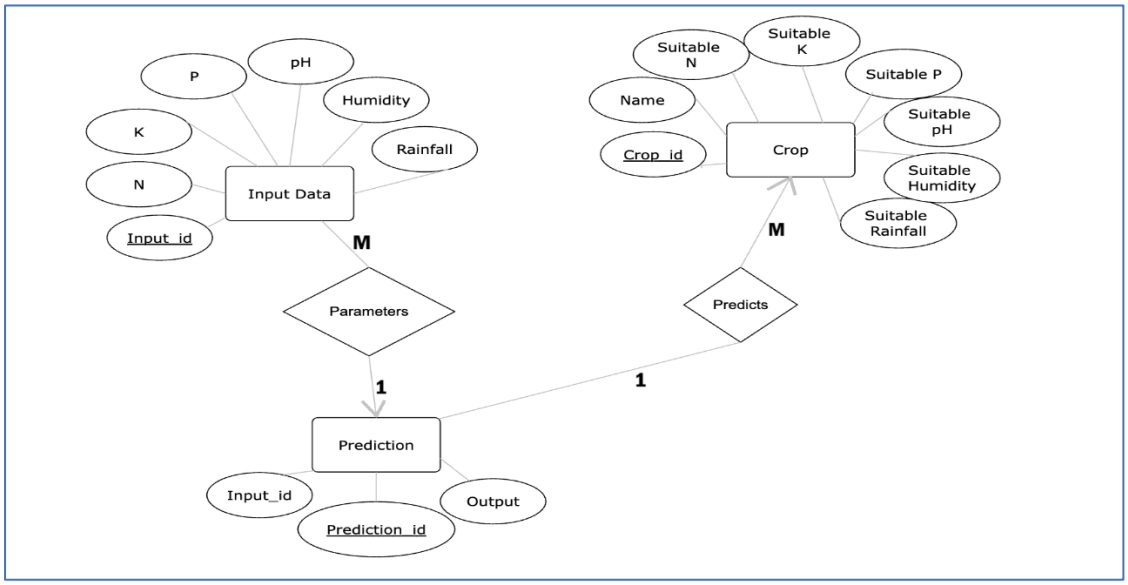


Fig: ER Diagram for smart crop recommendation system

PRE REQUISITES

- Input data must include values for Nitrogen (N), Phosphorus (P), Potassium (K), pH, Humidity, and Rainfall.
- Each input entry must have a unique Input_id.
- Crop database must already contain various crops with their corresponding ideal/suitable values for N, P, K, pH, Humidity, and Rainfall.
- Each crop entry must have a unique Crop_id.
- Prediction logic should be capable of matching input parameters against the crop database.
- The system must map many input data entries to one prediction (M:1 relationship).
- Predictions must be stored with a unique Prediction_id, linking back to Input_id and generating the final Output.
- The ML model (or logic) must already be trained or set up to compare and predict accurately.
- Input and Crop data should be validated before making a prediction.
- Data must be normalized/formatted correctly to allow parameter matching.

7. Appendices

7.1 Glossary

- Crop: The amount of same plant produced per unit area.
- Prediction: ML-generated suggestion of best crop.

7.2 Document Revision History

- Version 1.0 – Initial Draft – 05-May-2025