

CROP PREDICTION ON ENVIRONMENTAL FACTORS USING MACHINE LEARNING

Major Project Report Submitted to

SRI PADMAVATI MAHILA VISVAVIDYALAYAM

In Partial fulfillment of the requirement for the

MASTER OF COMPUTER APPLICATIONS

IV SEMESTER

By

MERNEEDI LAKSHMI (2023MCA16061)

Under the guidance of

Dr. P. BHARGAVI

Assistant Professor



Accredited by NAAC with A⁺ Grade

ISO 21001 : 2018 Certified

DEPARTMENT OF COMPUTER SCIENCE

**SRI PADMAVATI MAHILA VISVAVIDYALAYAM
(Women's University)**

Tirupati-517502(A.P), Andhra Pradesh

JUNE, 2025

DEPARTMENT OF COMPUTER SCIENCE
SRI PADMAVATI MAHILA VISVAVIDYALAYAM
(Women's University)

Tirupati-517502, Andhra Pradesh, India

Accredited with **A+** Grade by NAAC



CERTIFICATE

This is to certify that the project work entitled "**CROP PERDICTION ON ENVIRONMENTAL FACTORS USING MACHINE LEARNING**" is a bonafide record of work carried out by **MERNEEDI LAKSHMI (2023MCA16061)** in the **Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam, Tirupati** in partial fulfillment of the requirements of IV Semester of **MASTER OF COMPUTER APPLICATIONS**. The content of the Project Report has not been submitted to any other University/Institute for the award of any degree.

Guide

Head of the Department

Date: 20/06/2025


PROJECT COMPLETION CERTIFICATE

This is to certify that **Ms. Merneedi Lakshmi** bearing the **Roll No: 2023MCA16061** pursuing **Master of Computer Applications** at **Sri Padmavati Mahila Visvavidyalayam, Tirupati**. She is successfully completed the project based Internship entitled "**Crop Prediction On Environmental Factors Using Machine Learning**" based on "**Machine Learning, Python, HTML, CSS, JavaScript**" domain during the period from **04th April, 2025 to 14th June, 2025** at our organization.

During this period we found that she is sincere, hardworking, and technically sound and result oriented. She worked well during her tenure.

We take this opportunity to wish her all the best for her future.

For Green Tree Softtech Solutions Private Limited



M. Naveen Kumar Reddy
HR



DECLARATION

We hereby declare that MCA IV Semester Major Project entitled “**CROP PREDICTION ON ENVIRONMENTAL FACTORS USING MACHINE LEARNING**” was done at the **Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam, Tirupati**, in the year 2024-2025 under the guidance of **Dr P. BHARGAVI, Assistant Professor** in partial fulfillment of requirements of MCA IV Semester.

We also declare that this project is our original contribution of the best of our knowledge and belief. We further declare that this work has not been submitted for the award of any other degree of this or any other university/Institution.

Signature of the Student(s)

ACKNOWLEDGEMENT

We are greatly indebted to our guide **Dr P. BHARGAVI, Assistant Professor** for taking keen interest on our project work and providing valuable suggestions in all the possible areas of improvement.

We express our sincere thanks to the teaching staff of the Department of Computer Science for extending support and encouragement to us in all the stages of the project work.

We gratefully acknowledge and express our gratitude to the non-teaching staff of the Computer Science Department who supported us in preparing the project report.

Signature of the Student(s)

INDEX

SL. No.	Content	Page No.
	ABSTRACT	i
1.	INTRODUCTION	1
1.1.	University profile	
1.2.	Organization profile	
2.	PROBLEM DEFINITION	2-4
2.1.	Aim	
2.2.	Problem Definition	
2.2.1.	Existing System	
2.2.2.	Proposed System	
2.3.	Objectives	
2.4.	Key Features	
2.5.	Benefits	
2.6.	Data Set	
3.	SYSTEM ANALYSIS	5-12
3.1.	Software Requirement Specifications	
3.2.	System Requirement	
3.2.1.	Hardware Requirements	
3.2.2.	Software Requirements	
3.3.	Feasibility Study	
3.3.1.	Operational Feasibility	
3.3.2.	Technical Feasibility	
3.3.3.	Economic Feasibility	
3.4.	Modelling Approaches	
	(Based on the project)	
3.4.1.	UML diagrams	
3.4.1.1.	The Use Case Diagram	
3.4.1.2.	Active Diagram	
3.4.1.3.	Sequence Diagram	
3.4.1.4.	Class Diagram	

SL. No.	Content	Page No.
4. SYSTEM DESIGN		13
4.1. Design principle		
5. SYSTEM TESTING		14-17
5.1. Testing Schemes		
5.1.1	Unit Testing	
5.1.2	Integration Testing	
5.1.3	System Testing	
5.1.4	Path Testing	
5.2. Test Cases		
6. IMPLEMENTATION		18-20
7. CONCLUSION		21-22
7.1.	Performance of Proposed System	
7.2.	Limitation	
7.3.	Future Enhancements	
BIBILOGRAPHY		23
APPENDICES		24-53
APPENDIX A: Screens		
•	Screen Shots	
APPENDIX B: User Manual		
APPENDIX C: Source code		

ABSTRACT

The Crop Prediction System is a web-based platform designed to assist farmers in selecting the most suitable crops based on real-time soil and environmental conditions. This project applies machine learning techniques to analyze agricultural data, aiming to improve crop selection decisions, optimize resource usage, and enhance agricultural productivity through data-driven recommendations.

The system provides an intuitive interface for users to register, log in, and input key agricultural parameters such as nitrogen, phosphorus, potassium, pH level, temperature, humidity, and rainfall. Based on the submitted data, the system utilizes advanced prediction algorithms to generate accurate crop recommendations. Key features include user registration and authentication, secure login, real-time form validation, dynamic crop prediction, and historical data management to maintain user records and prediction history.

Technologies used for development include **HTML, CSS, and JavaScript** for the front-end interface, ensuring a user-friendly and responsive design. The back-end is developed using **Python with the Flask framework**, while machine learning functionality is implemented using the **Random Forest Classifier algorithm** for precise and reliable predictions. The system integrates **data preprocessing, model training, and evaluation** to ensure optimal prediction accuracy.

By integrating modern technologies into agriculture, the project empowers farmers with smart decision-making tools, minimizes risks associated with conventional farming practices, and promotes sustainable and profitable farming.

1. INTRODUCTION

1.1 UNIVERSITY PROFILE

Sri Padmavati Mahila Visvavidyalayam (university for women) was founded in the year 1983 by N.T. Rama Rao, the Chief Minister of Andhra Pradesh, with the fervent desire to train women students as better builders of nation and to include skills of leadership in all aspects of life. The University was established under the Sri Padmavati Mahila Visvavidyalayam Act of 1983, which has come in to force on 14th of April 1983, it was started with ten faculties and 300 students and 20 staff members. In pursuance of objectives of university is awarded “A+ Grade” by NAAC.

The campus of Sri Padmavati Mahila Visvavidyalayam is spread out in lush green area of 138.43 acres. The university is situated as a distance of 3 kilometers from railway and bus stations of Tirupati. The campus has the necessary buildings to run its academic programs and administrative machinery. There are separate Buildings for humanities and sciences, university’s Administration, Central Library, University Auditorium, Sericulture complex and school of Pharmaceutical Sciences and also an independent building for Computer Science, Computer Centre and examination hall.

1.2 ORGANIZATION PROFILE

Founded in 2014 and based in Chennai, **Green Tree Softtech Solutions Pvt Ltd** is a leading provider of software development, embedded systems, power electronics, robotics, and IT solutions. With a team of highly skilled professionals and over 40,000 hours of real-time project expertise, the company offers services including ERP solutions, mobile app development, billing software, IoT lab materials, PCB design, and digital marketing. Focused on innovation, performance, and client satisfaction, GTSS is committed to building self-sustainable startups in machinery and electric drive systems. Guided by values like accountability, integrity, and teamwork, the company aims to turn “Impossible into Possible” through cutting-edge technology and strong customer relationships.

2.PROBLEM DEFINITION

2.1. AIM:

The goal of this project is to develop a platform that helps farmers select the most suitable crops based on current soil and environmental conditions using machine learning. The system is designed to be user-friendly, allowing farmers to input key agricultural data and receive accurate crop recommendations. Whether a farmer is planning for a new season or optimizing resource use, this platform aims to make the decision-making process smarter, more efficient, and more reliable.

2.2. PROBLEM DEFINITION:

A crop prediction system for agriculture addresses several challenges that farmers face while deciding which crops to cultivate based on changing environmental conditions. These issues include:

- **Lack of Accurate Crop Selection:** Farmers often rely on traditional experience-based methods for crop selection, which may not always align with current soil health and climatic conditions, leading to poor yield.
- **Limited Access to Expert Knowledge:** Many farmers do not have easy access to agricultural experts or scientific advice that can guide them in selecting the right crops for their land.
- **Unpredictable Weather Conditions:** Fluctuations in temperature, humidity, and rainfall can heavily impact crop productivity. Without proper forecasting and data analysis, farmers are exposed to high risk.
- **Inefficient Use of Resources:** Improper crop selection may result in the wastage of important resources like fertilizers, water, and labor, increasing the cost of farming without guaranteed returns.
- **Complex Data Analysis:** Understanding the relationship between soil nutrients, pH levels, and environmental factors requires technical knowledge, which may not be accessible to many farmers.
- **Lack of Decision Support Systems:** There is a need for a simple, automated system that can analyze multiple factors and provide clear, data-driven crop recommendations to support better decision-making.

2.3. OBJECTIVE:

This project aims to simplify and improve the crop selection process for farmers by developing a user-friendly crop prediction system using machine learning. The main objectives of the system are:

- **Provide Accurate Crop Recommendations:** Help farmers select the most suitable crops based on current soil composition and environmental conditions, increasing the chances of better yield.
- **Promote Accessibility:** Allow farmers to access expert-level predictions and recommendations from any location without the need for complex technical knowledge or physical consultations.
- **Ensure Data-Driven Decision Making:** Analyze soil parameters (such as nitrogen, phosphorus, potassium, pH) and environmental factors (temperature, humidity, rainfall) to generate reliable and scientifically-backed suggestions.
- **Enhance Resource Efficiency:** Support better utilization of fertilizers, water, and land resources by recommending crops that match the field's current condition, thereby minimizing waste and cost.
- **Provide Real-Time Feedback:** Enable farmers to instantly receive crop suggestions and access their prediction history, allowing better planning for future harvests.
- **Simplify System Usage:** Deliver a clean, intuitive interface where farmers can easily input data and receive recommendations without unnecessary complexity.

2.4. KEY FEATURES:

- **Crop Prediction Form:** Farmers can input essential soil parameters (Nitrogen, Phosphorus, Potassium, pH) and environmental factors (Temperature, Humidity, Rainfall) to receive crop recommendations.
- **Machine Learning-Based Predictions:** The system uses trained machine learning models to analyze the input data and suggest the most suitable crops for cultivation.
- **User Registration and Authentication:** Farmers can securely register and log in to the platform to access prediction features and maintain their personal data.
- **Real-Time Validation:** Input fields provide real-time validation to ensure data accuracy while filling out registration and prediction forms.
- **Prediction History Tracking:** Users can view and track their previous crop predictions and registration details for better planning and monitoring.

- **Secure Data Management:** All user data and prediction history are securely stored and handled to maintain confidentiality and integrity.
- **Simple and User-Friendly Interface:** The platform provides an intuitive, easy-to-use interface that allows farmers to use the system without any technical expertise.
- **Google Account Integration:** The system offers an option to register and login using Google accounts for easier accessibility.

2.5.BENEFITS:

- **For Farmers:** Provides accurate and scientific crop recommendations based on real-time soil and environmental data, reducing the risk of poor crop selection and improving yield and profitability.
- **For Agricultural Planning:** Helps optimize the use of resources such as fertilizers, water, and land, ensuring efficient crop planning and reducing unnecessary costs.
- **For Both Farmers and the Agriculture Sector:** The system promotes data-driven decision-making, increases farming efficiency, encourages sustainable agricultural practices, and supports long-term productivity growth.

2.6.DATA SET:

The dataset used for this project contains agricultural data with various environmental and soil parameters. The input features include Nitrogen (N), Phosphorus (P), Potassium (K), pH, Temperature, Humidity, and Rainfall. Each data instance is labeled with the corresponding crop that grows best under those specific conditions.

This dataset was used to train and test the machine learning model, enabling it to predict the most suitable crop based on the values provided by the user. The data was preprocessed to handle any inconsistencies, missing values, and to ensure proper formatting before being fed into the model.

The dataset plays a crucial role in the accuracy and performance of the crop prediction system, as it directly influences the learning capability of the model.

3.SYSTEM ANALYSIS

3.1.SOFTWARE REQUIREMENT ANALYSIS:

The model that is basically being followed is the “Software Development life Cycle Model”, which state the feasibility study is done once the part is over the requirement analysis and project planning is beings. If system exists once the modification and addition of new module is need analysis of present system can we used as basic model.

The design starts after the requirement analysis complete and the coding begins after the design is completed once the programming completed the testing is done. In this model the sequence of activity performed in a software development project are:

- Planning
- Requirement
- Design
- Software development
- Testing
- Deployment
- Operation and maintenance

3.2.SYSTEM REQUIREMENT:

Requirement specifications plays an important role to create quality software solution requirements are refined and analysis to assess the clarity.

Requirements are represented in a manner that ultimately leads to successful software implementation. Each requirement must be consistent with overall objective.

The development of this project deals with the following requirements.

- Software requirements
- Hardware requirements

3.2.1. SOFTWARE REQUIREMENTS:

The software requirements specifications produce at the culmination of the analysis tasks.

One of the most difficult tasks is that the selection this software, once system requirement is known by determinant whether a particular software package fits the requirements.

- Technologies : HTML, CSS, JAVA SCRIPT
- Operating System : Windows OS

3.2.2. HARDWARE REQUIREMENTS:

The selection of hardware is very important in the existence and the proper working of any software. In this selection of hardware, the size and capacity requirements are also important.

- Processor : Intel i5
- Run : 8GB
- Keyboard : 84 Keys
- Monitor : CRT or LCD

3.3. FEASIBILITY STUDY:

Feasibility Study in Software Engineering is a study to evaluate feasibility of proposed project or system. Feasibility study is one of stage among important four stages of Software Project Management Process. As name suggests feasibility study is the feasibility analysis or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view. Feasibility study is carried out based on many purposes to analyze whether software product will be right in terms of development, implantation, contribution of project to the organization etc.

The system has been tested for feasibility in the following points.

- Operational Feasibility
- Technical Feasibility
- Economic Feasibility

3.3.1. OPERATIONAL FEASIBILITY:

- **Workflow Analysis:** Farmers register and log in to the system, enter soil parameters (N, P, K, pH) and environmental data (Temperature, Humidity, Rainfall), submit the form, and receive real-time crop recommendations.
- **User Training Needs:** Minimal training required due to the simple and intuitive user interface.
- **Integration:** The system can be further integrated in future with IoT sensors or external APIs for automated real-time data collection.

3.3.2. TECHNICAL FEASIBILITY:

- **System Development:** Developed using Python with Flask framework for back-end logic and machine learning, HTML/CSS/JavaScript for front-end, and Scikit-learn for ML model training and predictions.
- **Infrastructure:** Requires a web server for hosting, Python environment for execution, and secure storage for user data.
- **Scalability:** The system is designed to accommodate increasing user data, additional features, and expansion to mobile platforms in the future.

3.3.3. ECONOMIC FEASIBILITY:

- **Cost Analysis:** Includes expenses for web hosting, domain registration, cloud storage, and potential use of third-party APIs or services in future upgrades.
- **Revenue Projection:** The system can be expanded for commercial use by offering premium subscriptions or integration with government or private agricultural advisory services.
- **Return on Investment (ROI):** Significant long-term benefits for farmers in terms of increased yield, resource optimization, and reduced decision-making errors, making it highly cost-effective.

3.4. MODELLING APPROACHES:

3.4.1. UML DIAGRAMS:

UML stands for unified modelling language, is a standardized general-purpose visual modelling language in the field of Software Engineering. It is used for constructing and documenting a system or a project. This is widely used by people such as engineers to make module structure of what they want to build.

The goal of the UML is to become a common language for creating model of object oriented computer software. UML is standard language for specifying, visualizing, constructing, and documenting the primary artifacts of the software system, as well as for business modelling and other non-software systems.

The Unified Modelling Language is a standard language for specifying, visualization, Constructing and documenting the artifacts of software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

GOALS:

The primary goals in the design of the UML are as follows:

- ◆ Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- ◆ Provide extensibility and specialization mechanisms to extend the core concepts.
- ◆ Be independent of particular programming languages and development process.
- ◆ Provide a formal basis for understanding the modelling language.
- ◆ Encourage the growth of OO tools market.
- ◆ Support higher level development concepts such as collaborations, frameworks, patterns and components.
- ◆ Integrate best practices.

3.4.1.1. THE USE CASE DIAGRAM:

A The use case diagram for Crop Prediction on Environmental Factors Using Machine Learning shows the interaction between the farmer and the system. The farmer registers, logs in, enters soil and environmental data, submits the information, and receives the crop recommendation. The system processes the data and displays the result, while also allowing the user to view prediction history and logout. Optionally, an admin can manage user accounts and monitor system activities. This diagram helps to visualize the system's core functionalities and how users interact with them.

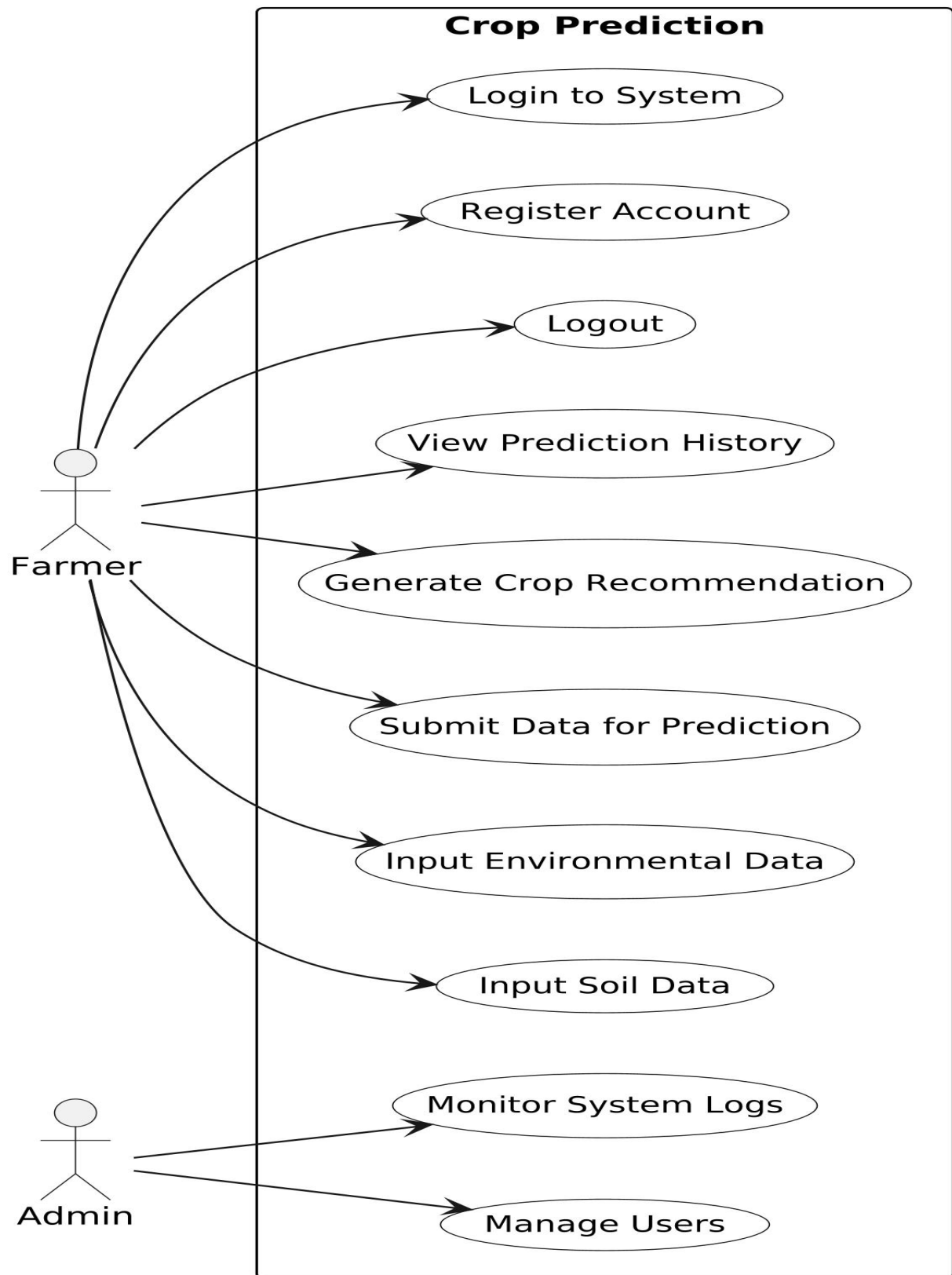


Figure 3.1 Use Case Diagram

3.4.1.2. ACTIVITY DIAGRAM:

The activity diagram for Crop Prediction on Environmental Factors Using Machine Learning represents the flow of actions from user registration to crop recommendation. The user first registers and logs into the system, enters soil and environmental parameters, and submits the data for processing. The system applies a machine learning model to analyze the input and generate a crop recommendation. The result is displayed to the user, and they can also view their prediction history before logging out.

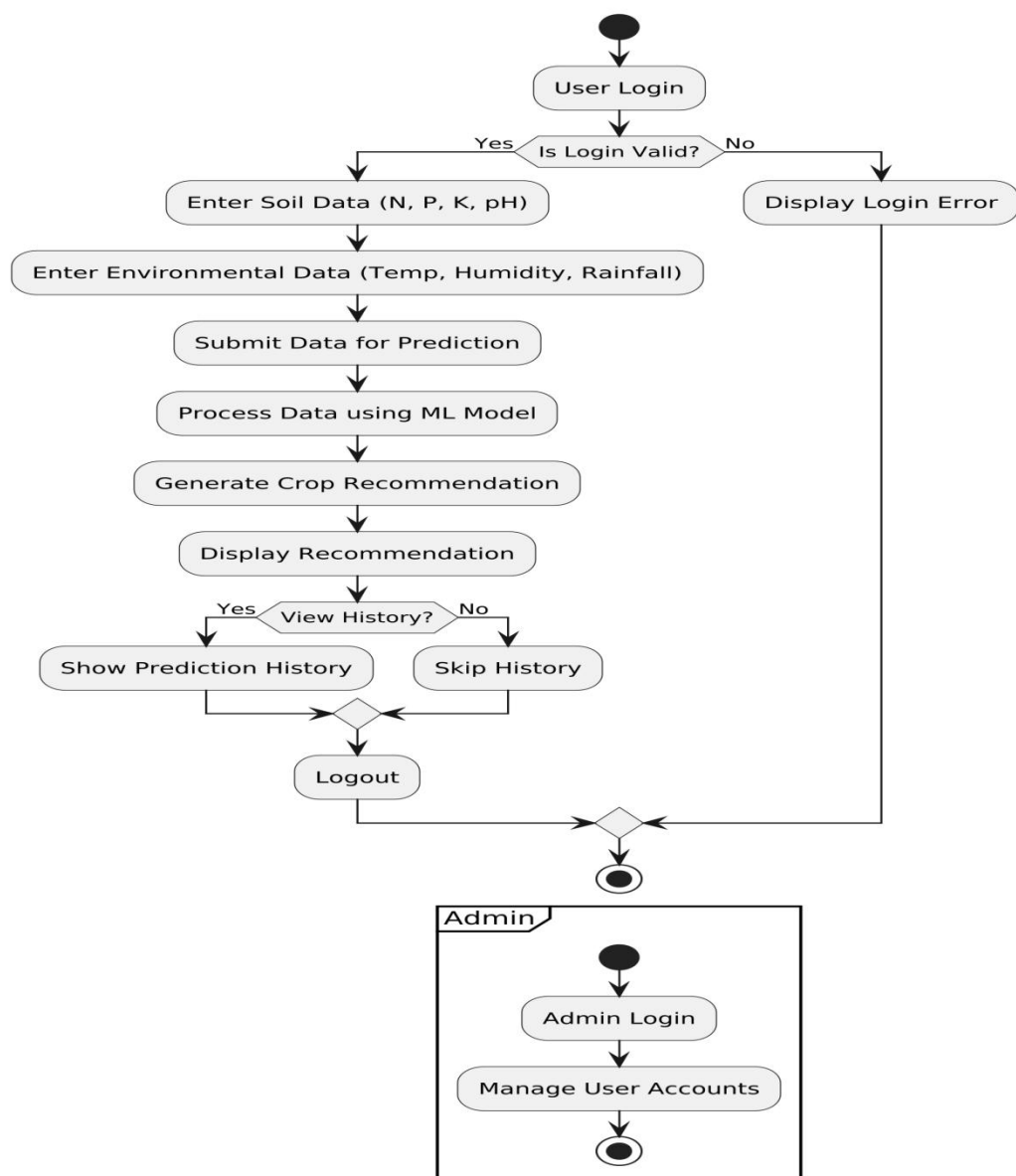


Figure 3.2 Activity Diagram

3.4.1.3. SEQUENCE DIAGRAM:

A sequence diagram for Crop Prediction on Environmental Factors Using Machine Learning illustrates the interaction between the farmer, system interface, machine learning model, and the database. The process begins with the farmer logging into the system and submitting soil and environmental parameters. The system processes the input, passes it to the machine learning model, and generates a crop recommendation. The recommendation is then displayed to the user, and the data is stored for history tracking. Admins can also interact with the system to manage user accounts and monitor system activities.

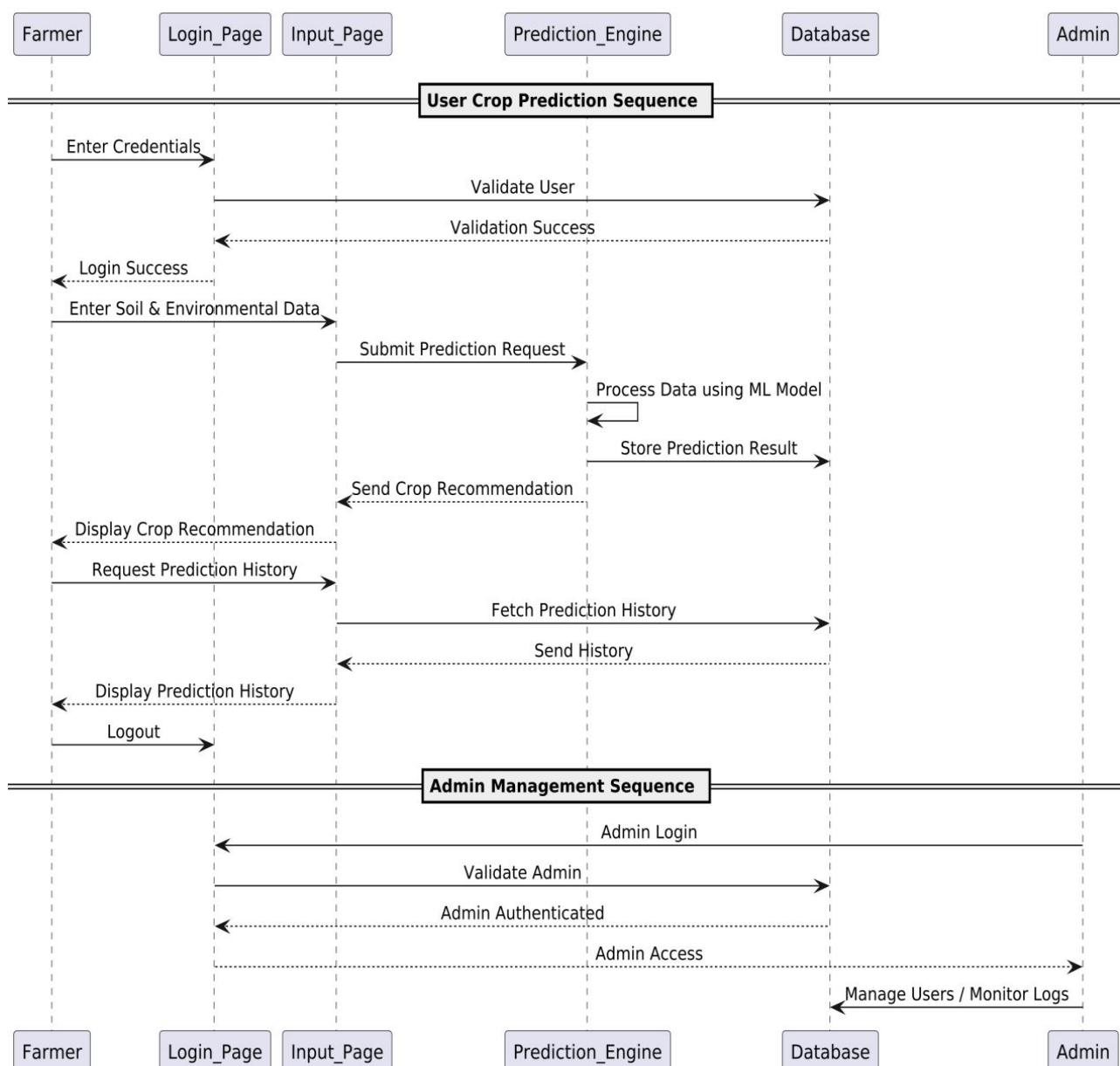


Figure 3.3 Sequence Diagram

3.4.1.4. CLASS DIAGRAM:

The class diagram represents the Crop Prediction on Environmental Factors Using Machine Learning system with key entities and their relationships. The User class manages registration, login, and stores personal details. The CropPrediction class handles input data such as soil and environmental parameters, and links to the MLModel class which processes the input and generates crop recommendations. The PredictionHistory class stores past predictions linked to each user for future reference. The Admin class manages user accounts and monitors system activity. Users submit data, the system processes predictions, and admins oversee account management.

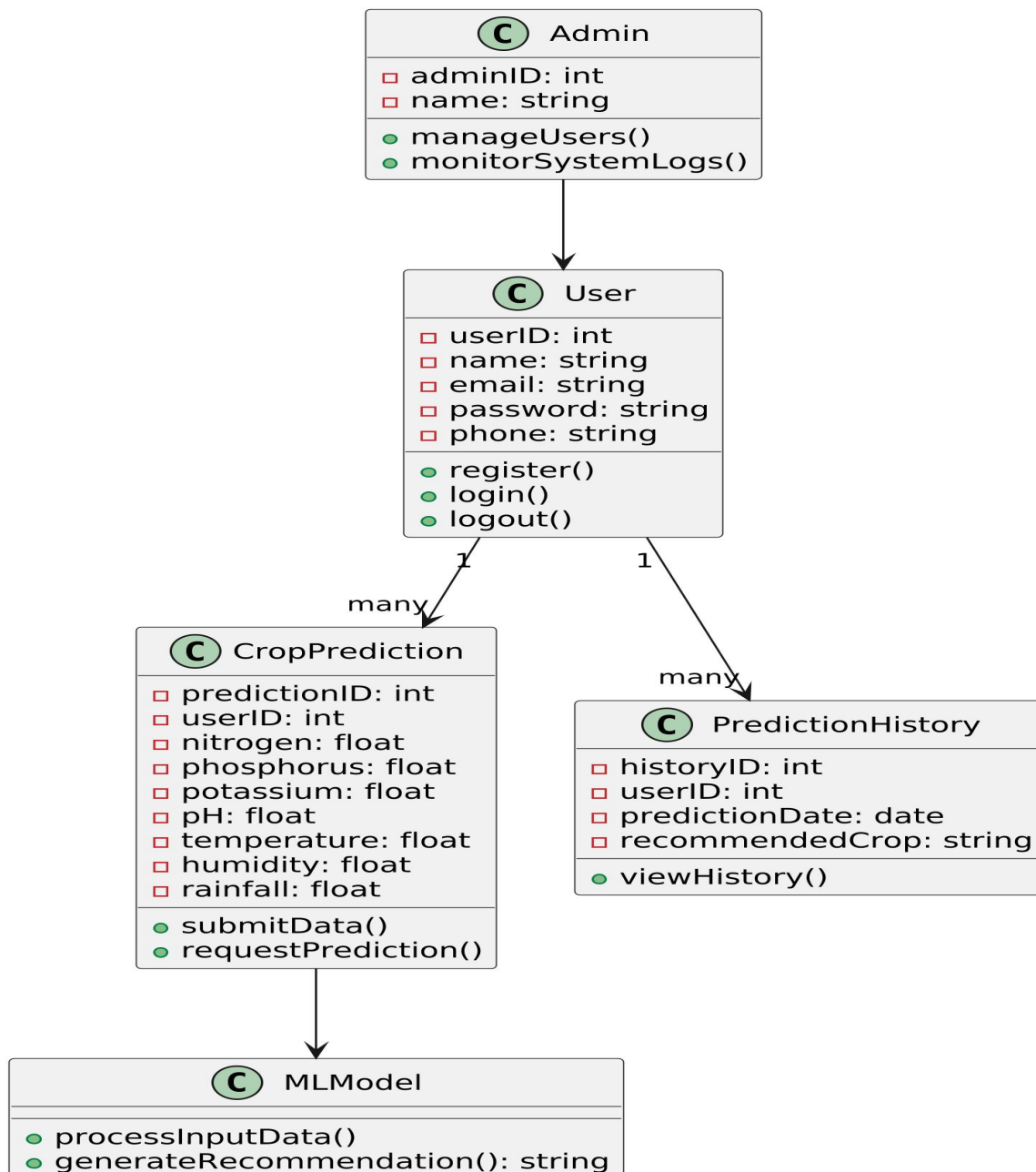


Figure 3.4 Class Diagram

4. SYSTEM DESIGN

The design phase of software system to build an answer for the issue as characterized in the requirement elicitation process. Design stage sets up a general architecture by dividing the software are parts, the connections and condition between such segments is then settled

4.1. DESIGN PRINCIPLE:

Software Design is also a process to plan or convert the software requirements into step that are needed to be carried out to develop a software system There are several principles that are used to organize and arrange. the structural components of Software design Software to Designs in which these principles are applied affect the content and the working process of the software from the beginning.

Scalability: A system is scalable if it is designed so that it can handle additional load and will still operate efficiently.

Reliability: A system is reliable if it can perform the function as expected, it can tolerate user mistakes, is good enough for the required use case, and it also prevents unauthorized access or abuse.

Availability: A system is available if it is able to perform its functionality (uptime/total time). Note reliability and availability are related but not the same. Reliability implies availability but availability does not imply reliability.

Efficiency: A system is efficient if it is able to perform its functionality quickly Latency, response time and bandwidth are all relevant metrics to measuring system efficiency.

Maintainability: A system is maintainable if it easy to make operate smoothly, simple for new engineers to understand, and easy to modify for unanticipated use cases.

Basic design principles that enable the software engineer to navigate the design process are

- The design should be tracked to the analysis model.
- The design should not reinvent the wheel.
- The design should exhibit uniformity and integrity.
- The design should be structure to accommodate changes.
- The design is not coding the coding is not a design.

The Golden rules for system design:

- Strive for consistency
- Enable Short-cuts for frequent users
- Informative feedback
- Design dialogues to yield closer
- Offer simple error Handling
- Permit easy reversal of actions
- Support internal locus of control

5. SYSTEM TESTING

System testing is the process of testing the entire system as a whole to ensure it meets the required specifications. It checks the interaction between all components, including functionality, performance, security, and usability. The goal is to identify defects and ensure the system works correctly and reliably before deployment.

5.1. Testing Schemes:

5.1.1. Unit Testing:

In unit testing, individual components of the system were tested in isolation. For example, the Crop Prediction Form was tested to verify field-level validations. Each field was tested for correct data types (numerical input), required field completion, and range checks (such as pH value between 0 and 14). The form correctly displayed error messages for invalid data and allowed submission only after all validations passed.

5.1.2. Integration Testing:

Integration testing focused on the interaction between the frontend and backend. After submitting valid data through the form, the system was tested to ensure that input data is properly processed, passed to the machine learning model, and that accurate crop predictions are returned. The integration also verified that prediction results are stored in the user's history for future reference.

5.1.3. System Testing:

System testing was conducted to validate complete user flows. This includes registration, login, input submission, prediction generation, history retrieval, and logout. The system was tested with multiple users to ensure proper session management, data security, and consistent prediction accuracy across all user operations.

5.1.4. Path Testing:

Basic Path Testing is a white-box testing approach used to verify that all independent logical paths within a module execute correctly at least once. It helps identify logical errors, missing validations, and unhandled conditions.

In this project, path testing was applied to the Crop Prediction Module, which collects environmental and soil data, validates input, and predicts the suitable crop using a

machine learning model. The focus was to ensure that the form handles valid inputs, detects invalid entries, and gracefully manages system errors.

Control Flow Steps:

- User opens the crop prediction form.
- System checks for required field completion and input types.
- If valid, data is sent to the Random Forest model.
- The prediction is displayed and stored
- If invalid or an exception occurs, an appropriate error message is shown.

Cyclomatic Complexity:

Using the formula $V(G) = E - N + 2P$:

- Edges (E) = 10
- Nodes (N) = 9
- Components (P) = 1

Result: $V(G) = 10 - 9 + 2(1) = 3$

Thus, three independent execution paths were tested.

Independent Paths:

- P1: Valid inputs → Submit → Predict crop → Show result → Save to history
- P2: Invalid inputs → Show validation error
- P3: Valid inputs → Model error → Show exception message

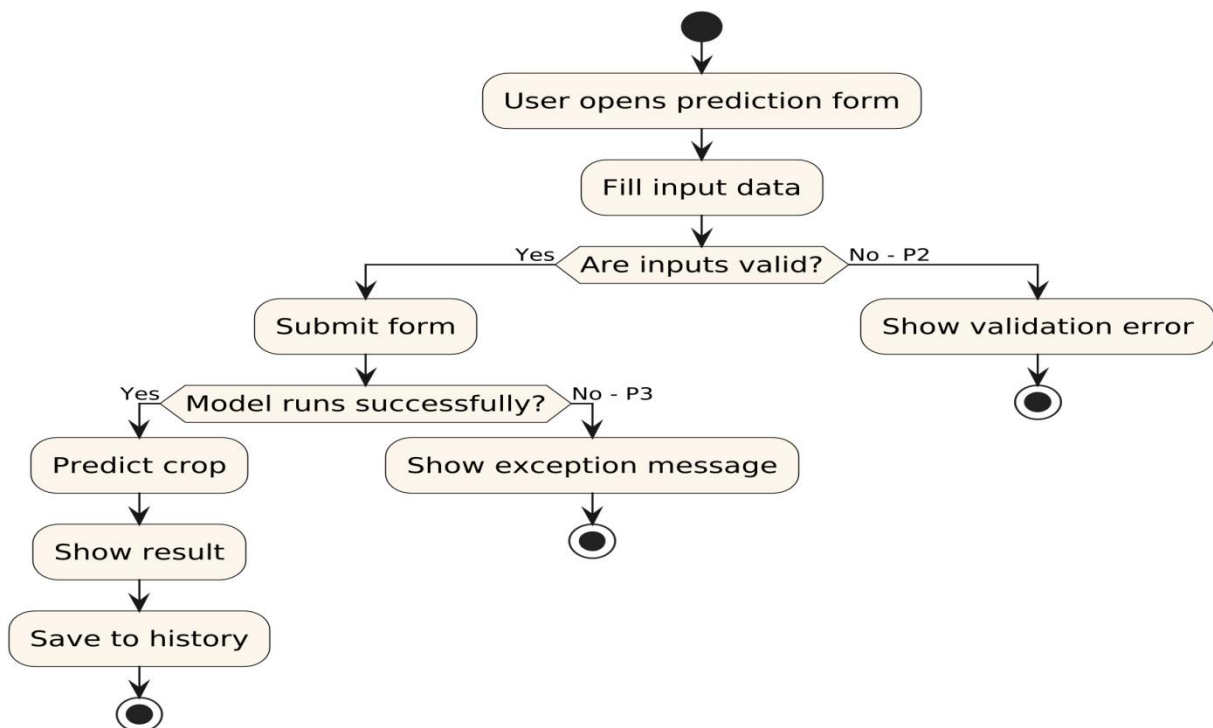


Figure 5.1 Path Testing Diagram

5.2. Test Cases:

1. User Login

Test Case ID	Scenario	Input	Expected Output	Result	Status
TC01	Valid login	Email: farmer@gmail.com, Password: Farm@123	Redirect to user dashboard	As expected	Pass
TC02	Invalid password	Email: farmer@gmail.com, Password: wrongpass	Show error: "Invalid credentials"	As expected	Pass
TC03	Empty fields	Email: (empty), Password: (empty)	Show warning: "Fields cannot be empty"	As expected	Pass

2. Registration Form

Test Case ID	Scenario	Input	Expected Output	Result	Status
TC04	New valid user	Name, email, password, acres, etc.	Show: "Registration successful"	As expected	Pass
TC05	Duplicate registration	Email already registered	Show: "Email already registered"	As expected	Pass
TC06	Invalid input format	Name with numbers, invalid phone/email	Show field-specific validation errors	As expected	Pass

3. Crop Prediction

Test Case ID	Scenario	Input	Expected Output	Result	Status
TC07	Valid prediction	Correct numeric values for all inputs	Recommended crop displayed	As expected	Pass
TC08	Invalid input	Letters instead of numbers	Show: "Invalid input format"	As expected	Pass
TC09	Missing values	Leave one or more fields empty	Show: "All fields are required"	As expected	Pass

4. Prediction History

Test Case ID	Scenario	Input	Expected Output	Result	Status
TC11	View history	Logged-in user	Display past predictions with details	As expected	Pass
TC12	Update history entry	Edit previous inputs	Update crop prediction & timestamp	As expected	Pass
TC13	Delete history entry	Click delete on record	Remove that entry from history	As expected	Pass

5. Google Login Integration

Test Case ID	Scenario	Input	Expected Output	Result	Status
TC14	Valid Google account	Google account login	Redirect to dashboard, auto-register	As expected	Pass
TC15	Unlinked Google email	Google login (new email)	Create profile, prompt additional info	As expected	Pass

6. IMPLEMENTATION

This project implements the Crop Prediction on Environmental Factors Using Machine Learning using HTML, CSS, JavaScript, Python (Flask), and Machine Learning with Scikit-learn (Random Forest Classifier). The system is designed to offer farmers a user-friendly web platform for accurate crop prediction based on real-time soil and environmental parameters.

i. Registration Page:

HTML

- Registration form with fields for name, email, phone, password, confirm password, acres, location, last crop, and experience.

CSS

- Responsive form design with proper spacing, fonts, and modern UI design.
- Styled input fields and buttons for clear user experience.

JAVASCRIPT:

- Real-time form validation for name, email, phone, and password strength.
- Password validation ensuring minimum length, capital letters, special characters, and numbers.
- Alert messages for validation errors.
- Integration of Google account login redirection.

ii. Login Page:

HTML

- Login form with fields for email and password.
- Link for Google Login.

CSS

- Simple and professional design with smooth transitions.
- Consistent form styling with registration page.

JAVASCRIPT

- Input validation for login fields.
- Google account login redirection functionality.
- Display appropriate login success or error messages.

iii. Dashboard Page:

HTML

- Displays a welcome section for farmers after successful login.
- Provides quick links to prediction form, history, blogs, and logout.
- Displays key statistics like number of predictions made, recommended crops, and recent activity.

- Includes an animated carousel for featured crops and farming tips.
- Embedded YouTube farming tutorial videos for farmer education.

CSS

- Fully responsive layout using grid and flexbox.
- Modern design with animations and hover effects for better user interaction.
- Professional color schemes matching agriculture and nature themes.
- Smooth transitions and clean typography for readability.

JAVASCRIPT

- Input validation for login fields.
- Google account login redirection functionality.
- Display appropriate login success or error messages.

PYTHON (FLASK)

- Fetches and renders user-specific data for dashboard statistics.
- Manages session control to ensure only logged-in users can access dashboard.

iV. Prediction Page (Crop Input Form)

HTML

- Form to input soil parameters (Nitrogen, Phosphorus, Potassium, pH) and environmental parameters (Temperature, Humidity, Rainfall).
- Submit button for requesting prediction.

CSS

- Clean and minimal form layout using grid and flexbox.
- Styled submit button and input fields for consistency.
- Fully responsive across devices.

JAVASCRIPT

- Real-time validation for numeric input fields.
- Prevent submission if fields are empty or invalid.

PYTHON(FLASK)

- Handles form submission, validates data, and calls ML model for prediction.
- Passes prediction result to result page for display.

MACHINE LEARNING

- Random Forest Classifier trained on historical crop data.
- Predicts the most suitable crop based on the provided input.

V. Result Page

HTML

- Displays the predicted crop recommendation.
- Option to navigate back to home or prediction form.

CSS

- Centered and visually appealing result display.
- Highlighted recommended crop with professional formatting.

PYTHON(FLASK)

- Receives prediction from ML model and renders the result template dynamically.

Vi. History Page

HTML

- Table to display previously submitted predictions by the user.
- Shows input data and predicted crop.

CSS

- Table formatting with alternating row colors for readability.
- Responsive layout for mobile and desktop views.

PYTHON(FLASK)

- Reads and writes prediction history from a JSON data file
- Displays user-specific history records.

Vii. Logout Functionality

JAVASCRIPT & FLASK

- Clears session storage or cookies.
- Safely redirects user back to login page after logout.

7. CONCLUSION

The Crop Prediction on Environmental Factors Using Machine Learning system provides an efficient, user-friendly, and intelligent platform for farmers to make accurate decisions on crop selection. By utilizing machine learning algorithms, specifically the Random Forest Classifier, the system analyzes real-time soil parameters and environmental conditions to recommend the most suitable crops for cultivation. This approach helps farmers to maximize yield, optimize resource usage, and reduce the risks associated with traditional farming methods.

The system integrates secure user registration, real-time form validation, interactive dashboards, and history tracking to offer a complete and modern agricultural decision-support tool. The combination of a responsive frontend, reliable backend processing, and machine learning prediction ensures the platform delivers both accuracy and accessibility to end-users.

7.1. Performance of Proposed System

The Crop Prediction System demonstrates strong performance across multiple areas:

- **Accurate Crop Prediction:** Generates reliable crop recommendations based on soil nutrients and environmental factors.
- **Secure User Authentication:** Ensures that only registered users can access prediction functionalities.
- **User-Friendly Interface:** Provides simple, intuitive navigation for farmers with real-time form validations.
- **Efficient Data Processing:** Processes user inputs and generates predictions quickly using optimized machine learning models.
- **Prediction History Tracking:** Maintains historical prediction data for user reference and long-term planning.
- **Real-Time System Responses:** Ensures users receive instant feedback on their predictions and system interactions.

7.2. Limitations

- The machine learning model is limited to the quality and scope of the dataset used for training.
- Prediction accuracy may vary if environmental changes occur outside the trained data range.
- The system currently requires manual input of data instead of automated data collection from IoT sensors.
- Limited external data integration such as real-time weather APIs.
- Currently, no mobile application is available for on-the-go access.

7.3. Future Enhancements

- Integrate live data collection through IoT-based soil sensors and weather APIs.
- Improve the machine learning model by including larger and more diverse agricultural datasets.
- Develop a mobile application for farmers to access predictions conveniently.
- Add multi-language support to make the system accessible to a wider range of users.
- Implement AI-driven advisory features for fertilizer optimization and pest control.
- Enhance data security using encryption and multi-factor authentication.
- Include real-time notifications and updates for users on environmental changes.
- Allow admin-based monitoring and control features for agricultural organizations.

8.BIBILOGRAPHY

For a project like **Crop Prediction on Environmental Factors Using Machine Learning**, the following resources were referred:

Books:

Jennifer Robbins, "Learning Web Design", O'Reilly Media, 2018.

Tom M. Mitchell, "Machine Learning", McGraw-Hill Education, 1997.

Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn", O'Reilly Media, 2017.

Online Tutorials and Documentation:

W3Schools, (<https://www.w3schools.com/>)

Flask Official Documentation. (<https://flask.palletsprojects.com/>)

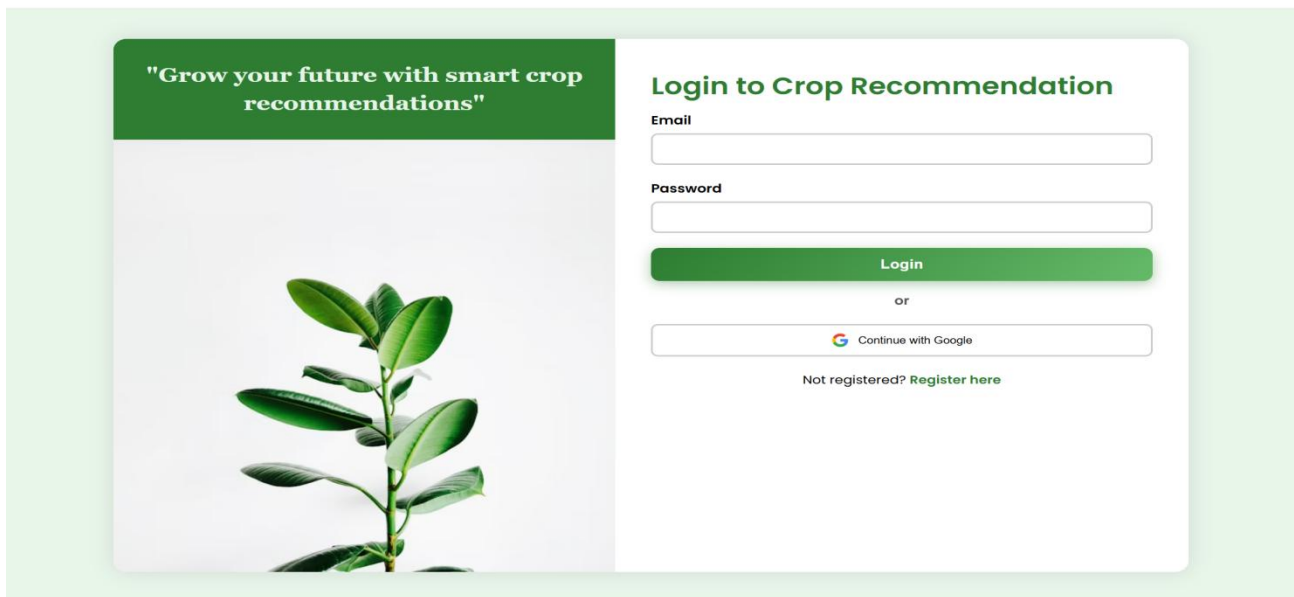
Python Official Documentation. (<https://docs.python.org/3/>)

Scikit-learn Official Documentation.

(<https://scikit-learn.org/stable/documentation.html>)

APPENDICES1

APPENDIX A: Screenshots



The screenshot shows a login page for a crop recommendation system. On the left, there is a green header with the text "Grow your future with smart crop recommendations" and a large image of a green plant. On the right, the title "Login to Crop Recommendation" is displayed. Below the title, there are input fields for "Email" and "Password". A green "Login" button is positioned below the password field. Below the button, the word "or" is centered. A "Continue with Google" button, featuring the Google logo, is located below "or". At the bottom, a link "Not registered? Register here" is provided.

"Grow your future with smart crop recommendations"


Login to Crop Recommendation

Email

Password

Login

or

 Continue with Google

Not registered? [Register here](#)

Fig-A1 Login Page



The screenshot shows a registration page for the same crop recommendation system. On the left, there is a green header with the text "Grow your future with smart crop recommendations" and a large image of a green plant. On the right, the title "Register for Crop Recommendation" is displayed. Below the title, there are input fields for "First Name", "Last Name", "Phone Number", "Age", "Experience (in years)", "Acres of Land", "Last Crop Grown", "Location (optional)" (with a sub-label "City or Region"), "Email", "Password", and "Confirm Password". A green "Register" button is positioned below the "Confirm Password" field. Below the button, the word "or" is centered. A "Continue with Google" button, featuring the Google logo, is located below "or". At the bottom, a link "Already registered? Login here" is provided.

"Grow your future with smart crop recommendations"

Register for Crop Recommendation

First Name

Last Name

Phone Number

Age

Experience (in years)

Acres of Land

Last Crop Grown

Location (optional)
City or Region

Email

Password

Confirm Password

Register

or

 Continue with Google

Already registered? [Login here](#)

Fig-A2 Register page

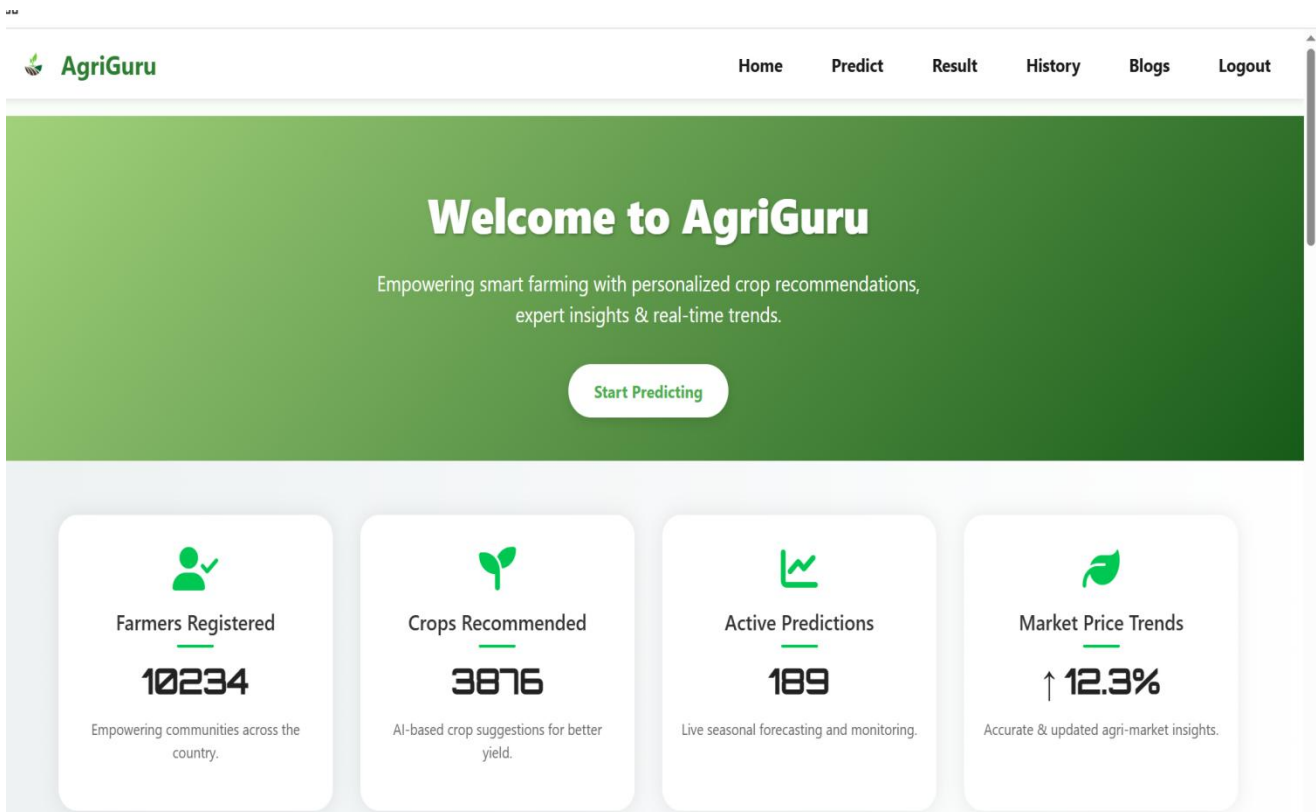


Fig-A3 Dashboard Page

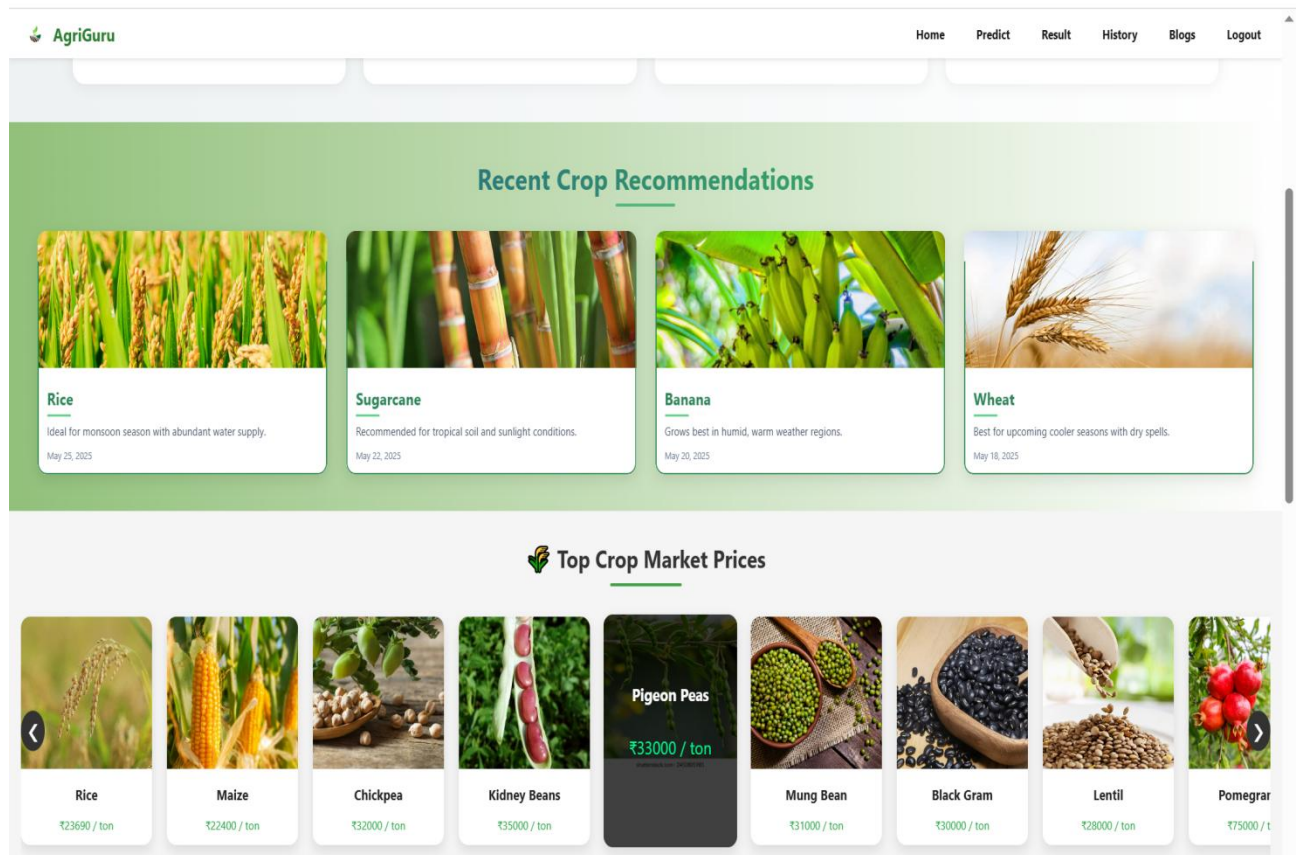


Fig-A4 Dashboard Page

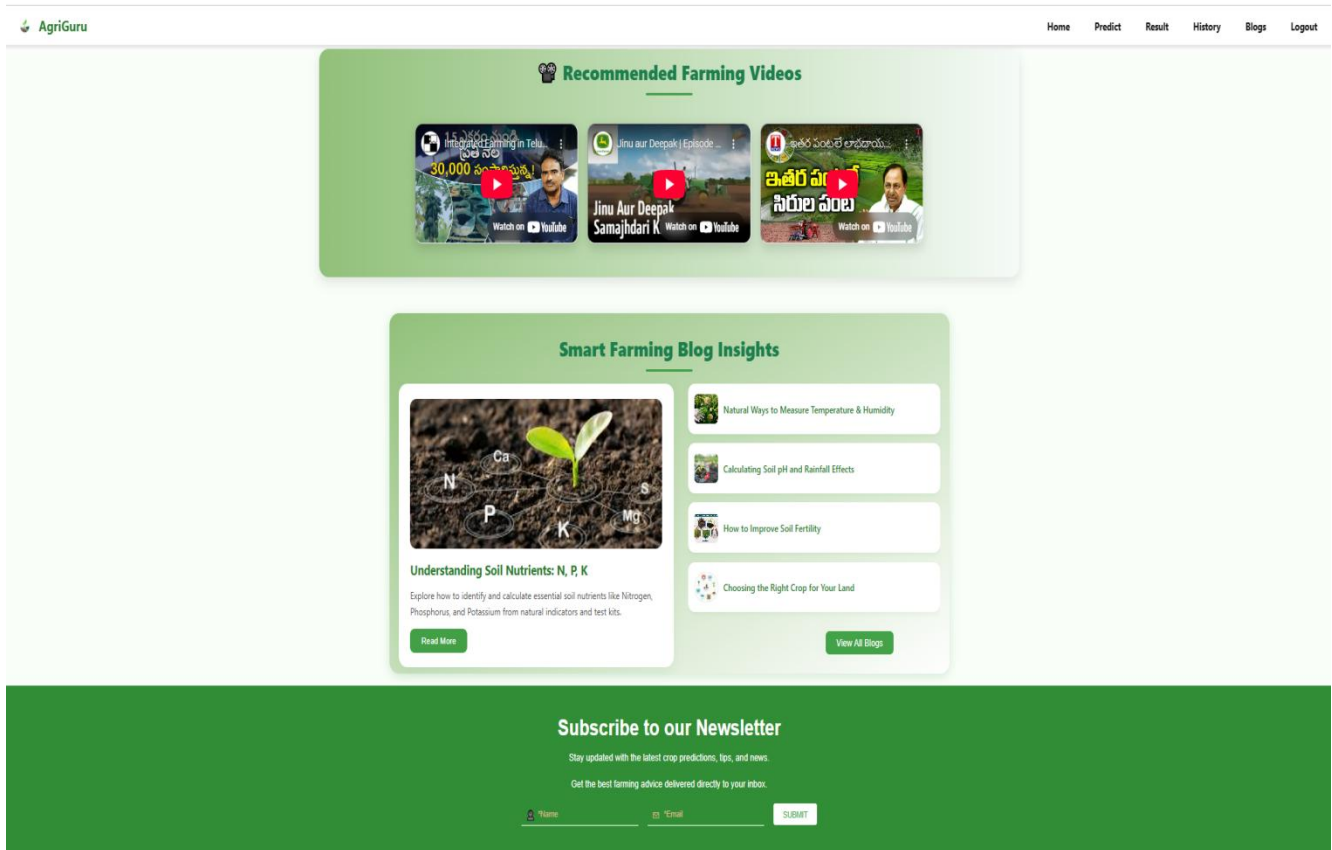


Fig-A5 Dashboard Page

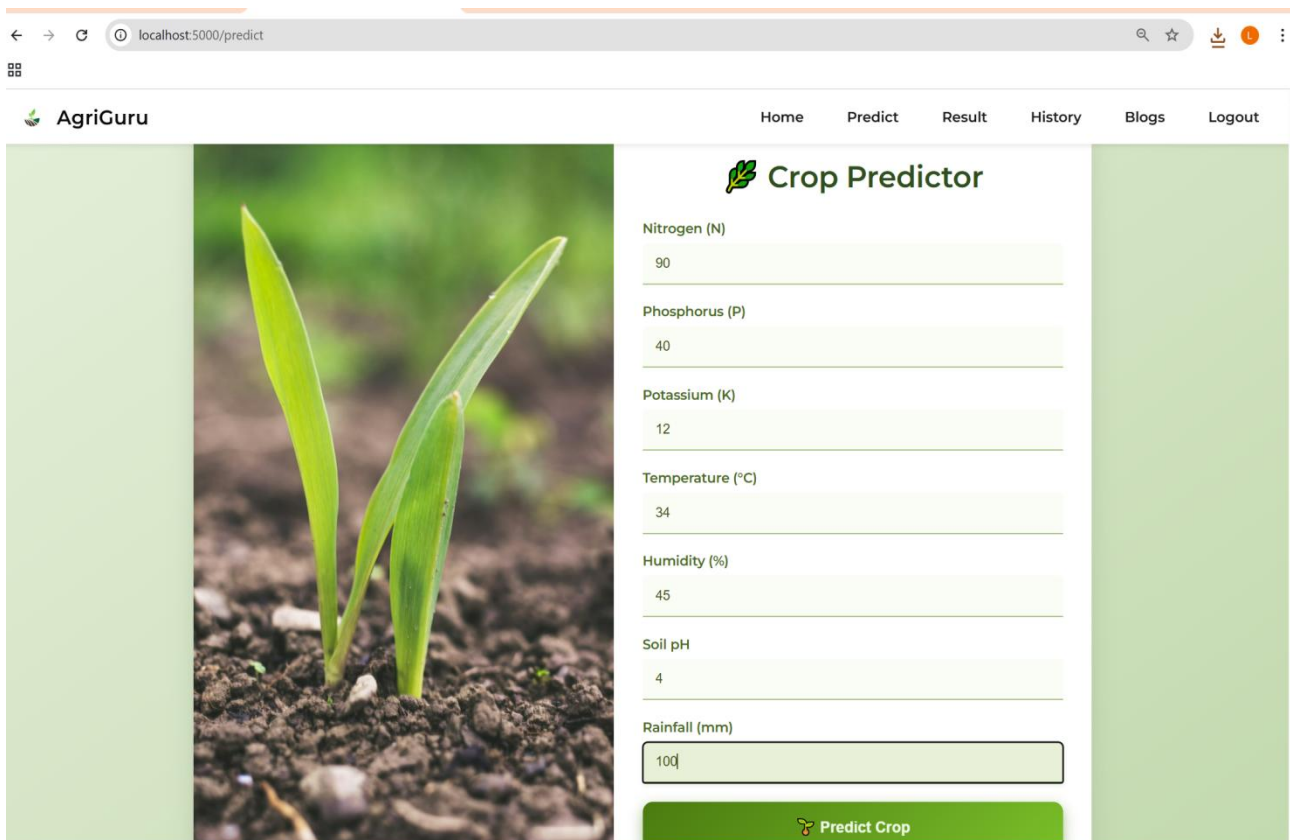


Fig-A6 Prediction Page



CROP PREDICTION RESULT

Maize



[← Back to Home](#)

Fig-A7 Result Page



Farmer & Crop Prediction History

Farmer Details

Name: SatyaNarayana Merneedi

Email: veerababu@gmail.com

Phone: 9987654323

Age: 45

Experience: 20 years

Acres: 4.0

Last Crop: rice

#	N	P	K	Temp (°C)	Humidity (%)	pH	Rainfall (mm)	Predicted Crop	Timestamp	Actions
1	90.0	42.0	43.0	20.0	82.0	6.0	200.0	rice	2025-06-16 15:17:06 IST+0530	Update Remove

Fig-A8 History Page

Appendix B: User Manual

This user manual provides step-by-step instructions for using the **Crop Prediction on Environmental Factors Using Machine Learning** system to ensure a smooth and efficient experience.

Getting Started

1. Accessing the Platform

- Open your preferred web browser and navigate to the Crop Prediction System website.
- Create an account by completing the registration form with your personal details such as name, email, phone number, location, land area, farming experience, and password.

2. Logging in

- Enter your registered email and password on the login page.
- Upon successful login, you will be redirected to the main dashboard.

Dashboard Overview

Upon logging in, the dashboard provides:

- Quick access to crop prediction, prediction history, blogs, and other features.
- Display of recent predictions made, including recommended crops.
- Educational content including blogs, farming tips, and embedded YouTube videos.
- Navigation links for prediction, history, blogs, and logout options.
- Newsletter subscription for updates on agricultural trends.

Crop Prediction Module

1. Submitting Prediction Data

- Navigate to the prediction page from the dashboard.
- Enter values for:
 - Nitrogen (N)
 - Phosphorus (P)
 - Potassium (K)
 - pH value
 - Temperature

- Humidity
- Rainfall
- Click the Submit button to receive crop recommendation.

2. Viewing Prediction Result

- The system will process the input through the trained machine learning model and display the most suitable crop recommendation.
- Prediction results are saved automatically to your prediction history.

Prediction History

- Access the history page to view all your past predictions.
- Each entry includes your submitted data and the recommended crop.

User Authentication

- Only registered users can access prediction features.
- The logout feature securely ends the session and redirects the user to the login page.

Blog and Educational Content

- Access the blog section to read articles on smart farming, modern agricultural practices, and crop management.
- Blogs are regularly updated to keep users informed.

Newsletter Subscription

- Subscribe to the newsletter to receive updates and tips related to agriculture and crop management.
- A confirmation message will appear after successful subscription.

Key Features

- **User-Friendly Interface** – Simple and intuitive design with easy navigation for farmers.
- **Real-Time Validation** – Immediate input validation during data entry to avoid incorrect submissions.
- **Machine Learning Integration** – Accurate crop recommendations using Random Forest Classifier.
- **Prediction History** – Tracks past predictions for reference and analysis.
- **Secure Access Control** – Login/logout system for secure user management.

Appendix C

Source Code:

```
from flask import Flask, render_template, redirect, url_for, session, request, flash
from flask_dance.contrib.google import make_google_blueprint, google
from flask_dance.consumer import oauth_authorized
import pickle
import os
import json
import pytz
from datetime import datetime
from dateutil import parser # for parsing ISO datetime strings

app = Flask(__name__)
app.secret_key = 'farm_secret'

# Allow insecure HTTP for local testing (remove in production)
os.environ['OAUTHLIB_INSECURE_TRANSPORT'] = '1'

# Files to store data persistently
FARMERS_FILE = 'farmers.json'
PREDICTIONS_FILE = 'predictions.json'

def load_farmers():
    if os.path.exists(FARMERS_FILE):
        with open(FARMERS_FILE, 'r') as f:
            return json.load(f)
    return []

def save_farmers():
    with open(FARMERS_FILE, 'w') as f:
        json.dump(farmers, f)

def load_predictions():
    if os.path.exists(PREDICTIONS_FILE):
        with open(PREDICTIONS_FILE, 'r') as f:
            preds = json.load(f)
            # Convert 'created_at' string back to datetime object
            for p in preds:
                if 'created_at' in p:
                    p['created_at'] = parser.isoparse(p['created_at'])
            return preds
    return []

def save_predictions():
    # Before saving, convert datetime objects to ISO strings
    preds_copy = []
    for p in predictions:
        p_copy = p.copy()
        if 'created_at' in p_copy and isinstance(p_copy['created_at'], datetime):
            p_copy['created_at'] = p_copy['created_at'].isoformat()
        preds_copy.append(p_copy)
```

```

with open(PREDICTIONS_FILE, 'w') as f:
    json.dump(preds_copy, f)

# Load ML model
model = pickle.load(open('rfmodel.pkl', 'rb'))

# Google OAuth blueprint setup
google_bp = make_google_blueprint(
    client_id="695075020518-
jc0r3ds0jqu6b4vt1lrmmmp5iv73b5onn.apps.googleusercontent.com",
    client_secret="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",
    scope=[
        "openid",
        "https://www.googleapis.com/auth/userinfo.email",
        "https://www.googleapis.com/auth/userinfo.profile"
    ],
    redirect_to="dashboard"
)

app.register_blueprint(google_bp, url_prefix="/login")

@oauth_authorized.connect_via(google_bp)
def log_redirect_uri(blueprint, token):
    print("Redirect URI used:", blueprint.redirect_url)

farmers = load_farmers()
predictions = load_predictions()

@app.route('/')
def home():
    return redirect(url_for('login'))

@app.route('/register', methods=['GET', 'POST'])
def register():
    if 'user' in session:
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        email = request.form['email']
        if any(f['email'] == email for f in farmers):
            return "Email already registered. Please login.", 400

    data = {
        'name': request.form['fname'],
        'surname': request.form['surname'],
        'phone': request.form['phone'],
        'age': int(request.form['age']),
        'experience': int(request.form['exp']),
        'acres': float(request.form['acres']),
        'last_crop': request.form['crop'],
        'email': email,
        'password': request.form['password']
    }

```

```

        farmers.append(data)
        save_farmers()
        flash("Registration successful! Please login.")
        return redirect(url_for('login'))

    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        user = next((f for f in farmers if f['email'] == email and f['password'] == password), None)
        if user:
            session['user'] = {
                'name': user['name'],
                'surname': user['surname'],
                'email': user['email']
            }
            flash('Login successful!')
            return render_template('login.html', success=True)
        else:
            flash('Invalid email or password.')
            return redirect(url_for('login'))

    return render_template('login.html')

@app.route('/dashboard')
def dashboard():
    if not google.authorized and 'user' not in session:
        return redirect(url_for('login'))

    if google.authorized and 'user' not in session:
        resp = google.get("/oauth2/v2/userinfo")
        if not resp.ok:
            return "Failed to fetch user info from Google.", 400
        info = resp.json()
        email = info.get('email', "")
        name = info.get('given_name', "")
        surname = info.get('family_name', "")

        if not any(f['email'] == email for f in farmers):
            farmers.append({
                'name': name,
                'surname': surname,
                'phone': "",
                'age': 0,
                'experience': 0,
                'acres': 0.0,
                'last_crop': "",
                'email': email,
                'password': ""
            })

```



```

    })
    save_farmers()

    session['user'] = {
        'name': name,
        'surname': surname,
        'email': email
    }
    flash("Login successful!")

    user_data = session.get('user', {})
    return render_template('dashboard.html', user=user_data)

@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if not google.authorized and 'user' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        try:
            data = [
                float(request.form['N']),
                float(request.form['P']),
                float(request.form['K']),
                float(request.form['temperature']),
                float(request.form['humidity']),
                float(request.form['ph']),
                float(request.form['rainfall'])
            ]
            pred = model.predict([data])[0]

            # Add current timestamp (UTC)
            now_utc = datetime.now(pytz.utc)

            predictions.append({
                'farmer': session['user']['email'],
                'inputs': data,
                'crop': pred,
                'created_at': now_utc # store as datetime object here; will convert on save
            })
            save_predictions()
            return render_template('result.html', predicted_crop=pred)
        except Exception as e:
            return f"Error: {str(e)}", 400

    return render_template('predict.html')

@app.route('/history')
def history():
    if 'user' not in session:
        return redirect(url_for('login'))

```

```

user_email = session['user']['email']
farmer = next((f for f in farmers if f['email'] == user_email), None)
user_predictions = [p for p in predictions if p['farmer'] == user_email]

# Timezones
local_tz = pytz.timezone('Asia/Kolkata')

for p in user_predictions:
    if 'created_at' in p:
        dt = p['created_at'] # datetime object
        dt_local = dt.astimezone(local_tz)
        p['timestamp'] = dt_local.strftime("%Y-%m-%d %H:%M:%S %Z%z")
    else:
        p['timestamp'] = "N/A"

return render_template('history.html', farmer=farmer, predictions=user_predictions)

@app.route('/remove/<int:index>', methods=['POST'])
def remove_prediction(index):
    if 'user' not in session:
        return redirect(url_for('login'))

    user_email = session['user']['email']
    user_predictions = [p for p in predictions if p['farmer'] == user_email]

    if 0 <= index < len(user_predictions):
        predictions.remove(user_predictions[index])
        save_predictions()
    return redirect(url_for('history'))

@app.route('/update/<int:index>', methods=['GET', 'POST'])
def update_prediction(index):
    if 'user' not in session:
        return redirect(url_for('login'))

    user_email = session['user']['email']
    user_predictions = [p for p in predictions if p['farmer'] == user_email]

    if 0 <= index < len(user_predictions):
        if request.method == 'POST':
            try:
                new_data = [
                    float(request.form['N']),
                    float(request.form['P']),
                    float(request.form['K']),
                    float(request.form['temperature']),
                    float(request.form['humidity']),
                    float(request.form['ph']),
                    float(request.form['rainfall'])
                ]
                new_crop = model.predict([new_data])[0]

```

```

        user_predictions[index]['inputs'] = new_data
        user_predictions[index]['crop'] = new_crop

        # Update timestamp on update
        user_predictions[index]['created_at'] = datetime.now(pytz.utc)

        save_predictions()
        return redirect(url_for('history'))
    except Exception as e:
        return f"Update Error: {e}", 400

    prediction = user_predictions[index]
    return redirect(url_for('predict'))

    return redirect(url_for('history'))
@app.route("/")
def index():
    google_info = None
    current_user_email = ""
    if google.authorized:
        resp = google.get("/oauth2/v2/userinfo")
        google_info = resp.json()
        current_user_email = google_info.get("email", "")
    return render_template("index.html", google_info=google_info,
current_user_email=current_user_email)

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('login'))

@app.route('/blogs')
def blog():
    return render_template('blogs.html')

if __name__ == "__main__":
    app.run(host='localhost', port=5000, debug=True)

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Crop Predictor</title>
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;600&display=swap"
rel="stylesheet" />
    <style>
        * {
            margin: 0; padding: 0; box-sizing: border-box;
        }

```

```

body {
  font-family: 'Montserrat', sans-serif;
  background: linear-gradient(145deg, #e6f0db, #c1d8ac);
  min-height: 100vh;
  padding-top: 5px;
  padding-bottom: 50px;
}

/* Navigation */
nav {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  background: white;
  color: rgb(8, 8, 8);
  padding: 12px 24px;
  display: flex;
  justify-content: space-between;
  align-items: center;
  font-weight: 700;
  z-index: 1000;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1); /* optional for depth effect */
}

nav .logo {
  font-size: 1.5rem;
}

nav ul {
  list-style: none;
  display: flex;
  gap: 29px;
  margin: 0;
  padding: 0;
}

nav ul li a {
  color: rgb(26, 25, 25);
  font-weight: 700;
  font-size: 17px;
  text-decoration: none;
  padding: 8px 12px;
  border-radius: 6px;
  transition: all 0.3s ease;
  position: relative;
  z-index: 1;
  overflow: hidden;
}

nav ul li a::before {
  content: "";
  position: absolute;
  top: 0; left: 0;

```

```

width: 100%; height: 100%;
background-color: #e6f4dd;
z-index: -1;
transition: transform 0.3s ease;
transform: scale(0.8);
opacity: 0;
border-radius: 6px;
}

nav ul li a:hover {
  color: #3f6e28;
}

nav ul li a:hover::before {
  transform: scale(1);
  opacity: 1;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
}

.wrapper {
  display: flex;
  flex-wrap: wrap;
  width: 95%;
  max-width: 1100px;
  background: white;
  border-radius: 20px;
  overflow: hidden;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
  margin: 100px auto 0 auto; /* Push down below navbar */
}

.image-side {
  flex: 1;
  min-width: 300px;
  background: url('static/pre.jpeg') no-repeat center center/cover;
  min-height: 100%;
}

.form-side {
  flex: 1;
  min-width: 300px;
  padding: 40px 35px;
  background: rgba(255, 255, 255, 0.95);
  display: flex;
  flex-direction: column;
  justify-content: center;
}

.form-side h1 {
  color: #2f4f1f;
  font-weight: 600;
  font-size: 2.2rem;
  text-align: center;
}

```

```

    margin-bottom: 30px;
}

form {
    display: flex;
    flex-direction: column;
}

.input-group {
    margin-bottom: 22px;
}

label {
    display: block;
    margin-bottom: 8px;
    color: #4a6a25;
    font-weight: 600;
}

input[type="text"] {
    width: 100%;
    padding: 14px 16px;
    font-size: 1rem;
    border: none;
    border-bottom: 2px solid #a9c284;
    background-color: #f9fdf5;
    border-radius: 6px 6px 0 0;
    transition: border-color 0.3s ease;
    font-weight: 500;
    color: #3d521b;
}

input[type="text"]::placeholder {
    color: #b8c3a6;
    font-style: italic;
}

input[type="text"]:focus {
    border-bottom-color: #3e7d12;
    background-color: #e8f0d7;
    box-shadow: 0 2px 5px rgba(62, 125, 18, 0.2);
}

button.submit-btn {
    padding: 16px 0;
    background: linear-gradient(135deg, #4b7b10, #7abf2a);
    border: none;
    border-radius: 12px;
    font-weight: 700;
    font-size: 1.15rem;
    color: #f0f9e8;
    cursor: pointer;
    box-shadow: 0 6px 15px rgba(75, 123, 16, 0.5);
}

```

```

        transition: background 0.35s ease, transform 0.2s ease;
    }

    button.submit-btn:hover {
        background: linear-gradient(135deg, #3e670d, #699b21);
        transform: scale(1.05);
    }

    .footer-note {
        text-align: center;
        margin-top: 18px;
        font-size: 0.9rem;
        color: #6e8556;
    }

    @media (max-width: 768px) {
        .wrapper {
            flex-direction: column;
        }

        .image-side {
            height: 220px;
        }

        .form-side {
            padding: 30px 25px;
        }
    }
</style>
</head>
<body>
<!-- Navbar -->
<header>
<nav>
<div class="logo" style="display: flex; align-items: center; gap: 10px;">
    
    <span>AgriGuru</span>
</div>

<ul>
<li><a href="/dashboard">Home</a></li>
<li><a href="/predict">Predict</a></li>
<li><a href="/result">Result</a></li>
<li><a href="/history">History</a></li>
<li><a href="/blogs">Blogs</a></li>
<li><a href="/logout">Logout</a></li>
</ul>
</nav>
</header>

```

```

<div class="wrapper">
  <div class="image-side"></div>

  <div class="form-side">
    <h1> Crop Predictor</h1>
    <form action="/predict" method="POST">
      <div class="input-group">
        <label for="N">Nitrogen (N)</label>
        <input type="text" id="N" name="N" placeholder="e.g., 90" required />
      </div>

      <div class="input-group">
        <label for="P">Phosphorus (P)</label>
        <input type="text" id="P" name="P" placeholder="e.g., 40" required />
      </div>

      <div class="input-group">
        <label for="K">Potassium (K)</label>
        <input type="text" id="K" name="K" placeholder="e.g., 40" required />
      </div>

      <div class="input-group">
        <label for="temperature">Temperature (°C)</label>
        <input type="text" id="temperature" name="temperature" placeholder="e.g., 25"
required />
      </div>

      <div class="input-group">
        <label for="humidity">Humidity (%)</label>
        <input type="text" id="humidity" name="humidity" placeholder="e.g., 80" required />
      </div>

      <div class="input-group">
        <label for="ph">Soil pH</label>
        <input type="text" id="ph" name="ph" placeholder="e.g., 6.5" required />
      </div>

      <div class="input-group">
        <label for="rainfall">Rainfall (mm)</label>
        <input type="text" id="rainfall" name="rainfall" placeholder="e.g., 200" required />
      </div>

      <button type="submit" class="submit-btn"> Predict Crop</button>
    </form>
    <p class="footer-note">Powered by smart agri-tech solutions</p>
  </div>
</div>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />

```



```

<title>Crop Prediction Result</title>
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet" />
<style>
  * {
    margin: 0; padding: 0; box-sizing: border-box;
  }

  body {
    font-family: 'Montserrat', sans-serif;
    background: linear-gradient(145deg, #e6f0db, #c1d8ac);
    min-height: 100vh;
    padding-top: 150px; /* space for fixed navbar */
    padding-bottom: 50px;
    color: rgb(8, 8, 8);
  }

  /* Navigation */
  nav {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    background: white;
    color: rgb(8, 8, 8);
    padding: 12px 24px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    font-weight: 700;
    z-index: 1000;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
  }

  nav .logo {
    display: flex;
    align-items: center;
    gap: 10px;
    font-size: 1.5rem;
  }

  nav .logo img {
    width: 40px;
    height: 40px;
    object-fit: contain;
  }

  nav ul {
    list-style: none;
    display: flex;
    gap: 29px;
    margin: 0;

```

```

padding: 0;
}

nav ul li a {
color: rgb(26, 25, 25);
font-weight: 700;
font-size: 17px;
text-decoration: none;
padding: 8px 12px;
border-radius: 6px;
transition: all 0.3s ease;
position: relative;
z-index: 1;
overflow: hidden;
}

nav ul li a::before {
content: "";
position: absolute;
top: 0; left: 0;
width: 100%; height: 100%;
background-color: #e6f4dd;
z-index: -1;
transition: transform 0.3s ease;
transform: scale(0.8);
opacity: 0;
border-radius: 6px;
}

nav ul li a:hover {
color: #3f6e28;
}

nav ul li a:hover::before {
transform: scale(1);
opacity: 1;
box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
}

/* Container and page styles */
.container {
background: white;
border-radius: 12px;
padding: 30px 40px;
max-width: 600px;
width: 100%;
box-shadow: 0 3px 8px rgba(46, 125, 50, 0.3);
text-align: center;
font-family: Arial, sans-serif;
color: #2e7d32;
margin: auto;
}

```

```

.message-box {
  background-color: #a5d6a7;
  padding: 20px;
  border-radius: 8px;
  font-weight: bold;
  color: white;
  margin-bottom: 25px;
}
h1 {
  font-family: 'Poppins', sans-serif;
  font-weight: 600;
  font-size: 1.8rem;
  margin-bottom: 20px;
  color: #27632a;
  animation: colorFade 5s ease-in-out infinite;
  text-transform: uppercase;
  letter-spacing: 1px;
  position: relative;
  padding-bottom: 8px;
}

h1::after {
  content: "";
  position: absolute;
  left: 50%;
  bottom: 0;
  transform: translateX(-50%);
  width: 60px;
  height: 3px;
  background-color: #3f7d2a;
  border-radius: 2px;
  animation: underlineFade 5s ease-in-out infinite;
}

@keyframes underlineFade {
  0%, 100% {
    background-color: #3f7d2a;
  }
  50% {
    background-color: #a5d6a7;
  }
}

@keyframes colorFade {
  0%, 100% {
    color: #27632a;
  }
  50% {
    color: #a5d6a7;
  }
}

```

```

.crop-name {
  font-family: 'Poppins', sans-serif;
  font-weight: 700;
  font-size: 2.2rem; /* smaller size */
  margin-bottom: 25px;
  color: #111111;

  letter-spacing: 0.8px;
}

img.crop-image {
  width: 300px;
  height: 200px;
  object-fit: cover;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0,0,0,0.15);
  margin-bottom: 30px;
}

a.back-button {
  display: inline-block;
  text-decoration: none;
  padding: 12px 30px;
  border: 2px solid #2e7d32;
  border-radius: 30px;
  color: #2e7d32;
  font-weight: 600;
  transition: background-color 0.3s ease, color 0.3s ease;
}

a.back-button:hover {
  background-color: #2e7d32;
  color: white;
}
</style>
</head>
<body>
<header>
<nav>
  <div class="logo">
    
    <span>AgriGuru</span>
  </div>
  <ul>
    <li><a href="/dashboard">Home</a></li>
    <li><a href="/predict">Predict</a></li>
    <li><a href="/result">Result</a></li>
    <li><a href="/history">History</a></li>
    <li><a href="/blogs">Blogs</a></li>
    <li><a href="/logout">Logout</a></li>
  </ul>

```

```

</nav>
</header>

<div class="container">
  <h1>Crop Prediction Result</h1>

  {% if predicted_crop == "rice" %}
    <h2 class="crop-name">Rice</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "maize" %}
    <h2 class="crop-name">Maize</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "chickpea" %}
    <h2 class="crop-name">Chickpea</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "kidneybeans" %}
    <h2 class="crop-name">Kidney Beans</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "pigeonpeas" %}
    <h2 class="crop-name">Pigeon Peas</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "mothbeans" %}
    <h2 class="crop-name">Moth Beans</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "mungbean" %}
    <h2 class="crop-name">Mung Bean</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "blackgram" %}
    <h2 class="crop-name">Black Gram</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "lentil" %}
    <h2 class="crop-name">Lentil</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "pomegranate" %}
    <h2 class="crop-name">Pomegranate</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "banana" %}
    <h2 class="crop-name">Banana</h2>
    <div class="image-container"></div>
  {% elif predicted_crop == "mango" %}
    <h2 class="crop-name">Mango</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "grapes" %}
    <h2 class="crop-name">Grapes</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "watermelon" %}
    <h2 class="crop-name">Watermelon</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "muskmelon" %}
    <h2 class="crop-name">Muskmelon</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "apple" %}
    <h2 class="crop-name">Apple</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "orange" %}
    <h2 class="crop-name">Orange</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "papaya" %}
    <h2 class="crop-name">Papaya</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "coconut" %}
    <h2 class="crop-name">Coconut</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "cotton" %}
    <h2 class="crop-name">Cotton</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "jute" %}
    <h2 class="crop-name">Jute</h2>
    <div class="image-container"></div>
    {% elif predicted_crop == "coffee" %}
    <h2 class="crop-name">Coffee</h2>
    <div class="image-container"></div>
    {% else %}
    <h2 class="crop-name">No prediction available</h2>
    {% endif %}

    <a href="/dashboard" class="back-button">←Back to Home</a>
</div>
</body>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

```

```

<title>Crop Recommendation - Register</title>
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap"
rel="stylesheet">
<style>
/* Your existing styles unchanged */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Poppins', sans-serif;
  height: 100vh;
  background: #e8f5e9;
  display: flex;
  justify-content: center;
  align-items: center;
  overflow: hidden;
}

.container {
  width: 90%;
  max-width: 1200px;
  height: 90vh;
  display: flex;
  border-radius: 15px;
  overflow: hidden;
  box-shadow: 0 0 25px rgba(0,0,0,0.1);
}

.left {
  flex: 1;
  display: flex;
  flex-direction: column;
  background: linear-gradient(to bottom, #2e7d32 50%, #fff 50%);
}

.quote-top {
  padding: 30px;
  font-size: 1.7rem;
  font-weight: bold;
  color: #e8f5e9;
  text-align: center;
  font-family: 'Georgia', serif;
  line-height: 1.4;
  background-color: #2e7d32;
  animation: fadeInDown 1s ease-in-out;
}

it"]:active {
  transform: scale(0.97);
}

```

```

button[disabled] {
  opacity: 0.6;
  cursor: not-allowed;
}

@keyframes fadeInUp {
  from { transform: translateY(20px); opacity: 0; }
  to { transform: translateY(0); opacity: 1; }
}

@keyframes fadeInDown {
  from { transform: translateY(-20px); opacity: 0; }
  to { transform: translateY(0); opacity: 1; }
}
</style>
</head>

<body>
{% with messages = get_flashed_messages() %}
{% if messages %}
  <div style="position: fixed; top: 10px; left: 50%; transform: translateX(-50%); background-
color: #4caf50; color: white; padding: 10px 20px; border-radius: 5px; z-index: 9999;">
    {{ messages[0] }}
  </div>
<script>
  setTimeout(() => {
    document.querySelector("[style*='position: fixed']").style.display = 'none';
  }, 3000);
</script>
{% endif %}
{% endwith %}

<div class="container">
  <div class="left">
    <div class="quote-top">
      "Grow your future with smart crop recommendations"
    </div>
    <div class="image-area"></div>
  </div>
  <div class="right">
    <h2>Register for Crop Recommendation</h2>
    <form id="registerForm" method="POST" action="/register" novalidate>
      <div class="form-group">
        <label>First Name</label>
        <input type="text" name="fname" id="fname" required oninput="validateName(this)">
        <div class="form-message" id="fnameMsg">Only letters allowed and cannot be
empty.</div>
      </div>
      <div class="form-group">
        <label>Last Name</label>
        <input type="text" name="surname" id="surname" required
oninput="validateName(this)">
        <div class="form-message" id="surnameMsg">Only letters allowed and cannot be

```



```

empty.</div>
</div>
<div class="form-group">
  <label>Phone Number</label>
  <input type="text" name="phone" id="phone" required oninput="validatePhone(this)">
  <div class="form-message" id="phoneMsg">Only digits allowed and cannot be
empty.</div>
</div>
<div class="form-group">
  <label>Age</label>
  <input type="number" name="age" id="age" min="18" max="100" required
oninput="validateAge(this)">
  <div class="form-message" id="ageMsg">Age must be between 18 and 100.</div>
</div>
<div class="form-group">
  <label>Experience (in years)</label>
  <input type="number" name="exp" id="exp" min="0" required
oninput="validateExp(this)">
  <div class="form-message" id="expMsg">Experience cannot be negative or
empty.</div>
</div>
<div class="form-group">
  <label>Acres of Land</label>
  <input type="number" name="acres" id="acres" min="0.01" step="0.01" required
oninput="validateAcres(this)">
  <div class="form-message" id="acresMsg">Acres must be greater than 0.</div>
</div>
<div class="form-group">
  <label>Last Crop Grown</label>
  <input type="text" name="crop" id="crop" required oninput="validateCrop(this)">
  <div class="form-message" id="cropMsg">This field cannot be empty.</div>
</div>
<div class="form-group">
  <label>Location (optional)</label>
  <input type="text" name="location" id="location" placeholder="City or Region">
</div>
<div class="form-group">
  <label>Email</label>
  <input type="email" name="email" id="email" required oninput="validateEmail(this)">
  <div class="form-message" id="emailMsg">Please enter a valid email address.</div>
</div>
<div class="form-group">
  <label>Password</label>
  <input type="password" name="password" id="password" required
oninput="checkPassword(this); validateForm()">
  <div class="password-requirements" id="passReq">
    <span id="len" class="req">✗ At least 8 characters</span>
    <span id="cap" class="req">✗ One uppercase letter</span>
    <span id="num" class="req">✗ One number</span>
    <span id="spec" class="req">✗ One special character</span>
  </div>
</div>
<div class="form-group">

```

```

        <label>Confirm Password</label>
        <input type="password" id="cpassword" required oninput="matchPasswords();
validateForm()">
        <div class="form-message" id="confirmMsg"></div>
    </div>
    <button type="submit" id="submitBtn" disabled>Register</button>
    <div style="text-align: center; margin: 15px 0; font-weight: 600; color: #555;">or</div>
    <!-- Google OAuth button -->
    <button type="button" class="google-btn"
onclick="window.location.href='/login/google'">
        
        Continue with Google
    </button>

    <div class="already">Already registered? <a href="/login">Login here</a></div>
</form>
</div>
</div>

<script>
// Show error on input; hide on blur regardless of validity
function addValidation(input, validateFn, msgId) {
    const msgEl = document.getElementById(msgId);

    input.addEventListener('input', () => {
        if (!validateFn(input)) {
            msgEl.style.display = 'block';
            input.style.borderColor = 'red';
        } else {
            msgEl.style.display = 'none';
            input.style.borderColor = '#ccc';
        }
    });

    input.addEventListener('blur', () => {
        // On blur hide error and reset border regardless of validity
        msgEl.style.display = 'none';
        input.style.borderColor = '#ccc';
    });
}

function validateName(input) {
    const val = input.value.trim();
    const regex = /^[A-Za-z]+$/;
    return val && regex.test(val);
}

function validatePhone(input) {
    const val = input.value.trim();
    const regex = /^[0-9]+$/;
    return val && regex.test(val);
}

```

```

function validateAge(input) {
  const val = Number(input.value);
  return val && val >= 18 && val <= 100;
}

function validateExp(input) {
  const val = Number(input.value);
  return val !== "" && !isNaN(val) && val >= 0;
}

function validateAcres(input) {
  const val = Number(input.value);
  return val > 0 && !isNaN(val);
}

function validateCrop(input) {
  const val = input.value.trim();
  return val.length > 0;
}

function validateEmail(input) {
  const val = input.value.trim();
  const regex = /^[^s@]+@[^s@]+\.[^s@]+$/;
  return regex.test(val);
}

// Password validation is more complex, so handle separately
function checkPassword(input) {
  const pass = input.value;
  const reqs = {
    len: document.getElementById('len'),
    cap: document.getElementById('cap'),
    num: document.getElementById('num'),
    spec: document.getElementById('spec')
  };
  const passReq = document.getElementById('passReq');

  passReq.style.display = pass.length > 0 ? 'block' : 'none';

  // Validate length
  reqs.len.textContent = pass.length >= 8 ? "✓ At least 8 characters" : "✗ At least 8 characters";
  reqs.len.style.color = pass.length >= 8 ? "green" : "red";

  // Uppercase letter
  const capOk = /[A-Z]/.test(pass);
  reqs.cap.textContent = capOk ? "✓ One uppercase letter" : "✗ One uppercase letter";
  reqs.cap.style.color = capOk ? "green" : "red";

  // Number
  const numOk = /\d/.test(pass);
  reqs.num.textContent = numOk ? "✓ One number" : "✗ One number";

```

```

reqs.num.style.color = numOk ? "green" : "red";

// Special character
const specOk = /^[^A-Za-z0-9]/.test(pass);
reqs.spec.textContent = specOk ? "✔ One special character" : "✗ One special character";
reqs.spec.style.color = specOk ? "green" : "red";
}

function passwordIsValid() {
  const pass = document.getElementById('password').value;
  return (
    pass.length >= 8 &&
    /[A-Z]/.test(pass) &&
    /\d/.test(pass) &&
    /^[^A-Za-z0-9]/.test(pass)
  );
}

function matchPasswords() {
  const pass = document.getElementById('password').value;
  const cpass = document.getElementById('cpassword').value;
  const msg = document.getElementById('confirmMsg');
  if (cpass.length === 0) {
    msg.style.display = 'none';
    return false;
  }
  if (pass === cpass) {
    msg.textContent = "Passwords match";
    msg.style.color = "green";
    msg.style.display = "block";
    return true;
  } else {
    msg.textContent = "Passwords do not match";
    msg.style.color = "red";
    msg.style.display = "block";
    return false;
  }
}

// Validate form for submit button enabling
function validateForm() {
  const valid =
    validateName(document.getElementById('fname')) &&
    validateName(document.getElementById('surname')) &&
    validatePhone(document.getElementById('phone')) &&
    validateAge(document.getElementById('age')) &&
    validateExp(document.getElementById('exp')) &&
    validateAcres(document.getElementById('acres')) &&
    validateCrop(document.getElementById('crop')) &&
    validateEmail(document.getElementById('email')) &&
    passwordIsValid() &&
    matchPasswords();

```

```

    document.getElementById('submitBtn').disabled = !valid;
  }

window.onload = function() {
  addValidation(document.getElementById('fname'), validateName, 'fnameMsg');
  addValidation(document.getElementById('surname'), validateName, 'surnameMsg');
  addValidation(document.getElementById('phone'), validatePhone, 'phoneMsg');
  addValidation(document.getElementById('age'), validateAge, 'ageMsg');
  addValidation(document.getElementById('exp'), validateExp, 'expMsg');
  addValidation(document.getElementById('acres'), validateAcres, 'acresMsg');
  addValidation(document.getElementById('crop'), validateCrop, 'cropMsg');
  addValidation(document.getElementById('email'), validateEmail, 'emailMsg');

  // Password input
  const passInput = document.getElementById('password');
  passInput.addEventListener('input', () => {
    checkPassword(passInput);
    validateForm();
  });
  passInput.addEventListener('blur', () => {
    document.getElementById('passReq').style.display = 'none';
    passInput.style.borderColor = '#ccc';
  });

  // Confirm password input
  const cpassInput = document.getElementById('cpassword');
  cpassInput.addEventListener('input', () => {
    matchPasswords();
    validateForm();
  });
  cpassInput.addEventListener('blur', () => {
    document.getElementById('confirmMsg').style.display = 'none';
    cpassInput.style.borderColor = '#ccc';
  });

  // Validate form initially and on any input
  const inputs = document.querySelectorAll('#registerForm input');
  inputs.forEach(input => {
    input.addEventListener('input', validateForm);
  });
};

// Prevent submit if invalid
document.getElementById('registerForm').addEventListener('submit', function(e){
  if (document.getElementById('submitBtn').disabled) {
    e.preventDefault();
    validateForm();
  }
});
</script>

</body>
</html>

```