

CROPOPTIMA : AI-DRIVEN CROP SUITABILITY, HEALTH MONITORING, AND RESOURCE OPTIMIZATION SYSTEM

A Project Work Report

Submitted in Partial Fulfillment for the Award of the Degree

Of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND BUSINESS SYSTEMS

Submitted by

AYINAVILLI SIVA TEJA GOUD

21B91A5704

MAKIREDDY UDAY SAI

21B91A5730

KAVALI HEMANTHA RAYUDU

21B91A5725

PADALA KEDARNADH REDDY

21B91A5744

Under the esteemed guidance of

Dr. K. SATYANARAYANA RAJU

Assistant Professor



DEPARTMENT OF INFORMATION TECHNOLOGY
SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE
(AUTONOMOUS)

(Approved by AICTE, New Delhi, Affiliated to JNTUK, Kakinada)

CHINNA AMIRAM :: BHIMAVARAM-534204

April - 2025

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE (AUTONOMOUS)

(Approved by AICTE , New Delhi, Affiliated to JNTUK,KAKINADA)

CHINNA AMIRAM, BHIMAVARAM-534204

DEPARTMENT OF INFORMATION TECHNOLOGY



Certificate

This is to certify that the Project Work report entitled " CROPOPTIMA : AI-DRIVEN CROP SUITABILITY, HEALTH MONITORING, AND RESOURCE OPTIMIZATION SYSTEM", is Bonafide work submitted by AYINAVILLI SIVA TEJA GOUD (Regd.No: 21B91A5704), MAKIREDDY UDAY SAI (Regd.No: 21B91A5730), KAVALI HEMANTHA RAYUDU (Regd.No: 21B95A5725), PADALA KEDARNADH REDDY (Regd.No: 21B91A5744) in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology during the Academic year 2024-2025.

HOD

Dr. P RAVI KIRAN VARMA
HOD-IT Dept

GUIDE

Dr K SATYANARAYANA RAJU
Assistant Professor

CERTIFICATION OF EXAMINATION

This to certify that I have examined the concept and hereby accord my approval of it as a Project Work entitled “**CROPOPTIMA : AI-DRIVEN CROP SUITABILITY, HEALTH MONITORING, AND RESOURCE OPTIMIZATION SYSTEM**” carried out and presented by **AYINAVILLI SIVA TEJA GOUD (Regd.No: 21B91A5704), MAKIREDDY UDAY SAI (Regd.No: 21B91A5730), KAVALI HEMANTHA RAYUDU (Regd.No: 21B95A5725), PADALA KEDARNADH REDDY (Regd.No: 21B91A5744)** in a manner required for its acceptance on partial fulfillments for the award of the degree of BACHELOR OF TECHNOLOGY in Computer Science and Business Systems for which it has been submitted.

This approval does not necessarily endorse or accept every statement made opinion expressed or conclusions drawn as recorded in the Project Work report it only signifies the acceptance of the report for the purpose for which submitted.

Signature

Project Guide :

Senior Faculty :

External Examiner :

HOD :

DECLARATION

This Project Work report entitled “**CROPOPTIMA : AI-DRIVEN CROP SUITABILITY, HEALTH MONITORING, AND RESOURCE OPTIMIZATION SYSTEM**” has been carried out by us in the partial fulfillment of the requirements for the award of the degree of B.Tech (CSBS), S.R.K.R Engineering College(A). We hereby declare this project work/project report has not been submitted to any of the other university/Institute for the award of any other degree/diploma.

Name	Register No	Signature
AYINAVILLI SIVA TEJA GOUD	21B91A5704	
MAKIREDDY UDAY SAI	21B91A5730	
KAVALI HEMANTHA RAYUDU	21B91A5725	
PADALA KEDARNADH REDDY	21B91A5744	

ACKNOWLEDGEMENT

The victorious completion of this project would be incomplete without greeting those who made it possible and whose guidance and encouragement made efforts that taken to the success.

The thesis presented here is the work accomplished under the enviable and scholarly guidance of **Dr. K. SATYANARAYA RAJU**, Assistant Professor of Information Technology Department. We are grateful for the indomitable support. We express deep gratitude for her inspiring remarks and for helping with this project.

We take this opportunity to express our sincere thanks to **Dr. P RAVI KIRAN VARMA**, Head of the Department, Information Technology, for his support and encouragement.

We express our sincere thanks to **Prof. K V Murali Krishnam Raju**, PRINCIPAL, S.R.K.R Engineering College, Bhimavaram for giving us this opportunity for the successful completion of this project.

Finally, we also express our sincere thanks to other **Teaching and non-Teaching staff** for their support in accomplishing this project.

- | | | |
|----|---------------------------|------------|
| 1. | AYINAVILLI SIVA TEJA GOUD | 21B91A5704 |
| 2. | MAKIREDDY UDAY SAI | 21B91A5730 |
| 3. | KAVALI HEMANTHA RAYUDU | 21B91A5725 |
| 4. | PADALA KEDARNADH REDDY | 21B91A5744 |

INDEX

NAME OF THE CHAPTER	PAGE NO
<i>List of Figures</i>	<i>i</i>
<i>List of Tables</i>	<i>ii</i>
<i>Abstract</i>	<i>iii</i>
1 INTRODUCTION	1
2 LITERATURE REVIEW	2-3
3 EXISTING SYSTEM AND PROBLEM STATEMENT	4-5
3.1 Existing System	4
3.2 Problem Statement	5
4 PROPOSED SYSTEM	6
5 SYSTEM ARCHITECTURE	7-10
5.1 Input Layer	8
5.2 Data Preprocessing Layer	8
5.3 Feature Extraction and Processing	8
5.4 Core Processing Module	8-9
5.4.1 Crop Prediction Module	8
5.4.2 Crop Disease Detection Module	9
5.4.3 Fertilizer Recommendation Module	9
5.5 Integration Layer	9
5.6 User Interaction Layer	9
5.7 Crop Disease Detection Module	10
5.7.1 Deep Learning Model Training and Detection	10
6 SYSTEM REQUIREMENTS AND DESIGN	11-15
6.1 System Requirements	11-12
6.1.1 Hardware Requirements	11
6.1.2 Software Requirements	12
6.2 System Design	13-17
6.2.1 Admin Sequence Diagram	14
6.2.2 User Sequence Diagram	15
6.2.3 State Chart Diagram	15

6.2.4 Component Diagram	16
6.2.5 Deployment Diagram	17
7 SYSTEM IMPLEMENTATION	18-20
7.1 Data Collection and Preprocessing	18
7.1.1 Data Collection for Crop Prediction	18
7.1.2 Image Data for Crop Disease Detection	18
7.2 Crop Prediction Module	18-19
7.2.1 Feature Extraction	18
7.2.2 Model Development	19
7.2.3 Prediction Engine	19
7.2.4 Fertilizer Recommendation Module	19
7.3 Crop Disease Detection Module	20
8 RESULTS AND DISCUSSION	21-27
8.1 Results	21-24
8.1.1 Crop Prediction Module Performance	21
8.1.2 Fertilizer Recommendation Module	21
8.1.3 Crop Disease Detection Module Accuracy	22-25
8.2 Discussion	26-27
8.2.1 Efficiency of Prediction Models	26
8.2.2 Impact of Feature Selection	26
8.2.3 Strength of Image – Based Disease Detection	26
8.2.4 Challenges and Limitations	26
8.2.5 Future Prospects	27
9 CONCLUSION	28
10 REFERENCES	29-30
APPENDIX-I	31
Source Code	32-43
APPENDIX – II	44
Research Paper	45-52

LIST OF FIGURES

S.No	Figure No.	Name Of the Figure	Page No.
1	Figure 4.2	System Architecture	7
2	Figure 6.2	System Design	13
3	Figure 6.2.1	Admin Sequence Diagram	14
3	Figure 6.2.2	User Sequence Diagram	15
4	Figure 6.2.3	State Chart Diagram	16
5	Figure 6.2.4	Component Diagram	16
6	Figure 6.2.5	Deployment Diagram	17
7	Figure 8.1.1	Example Result for Crop Prediction	21
8	Figure 8.1.2	Example Result for Disease Detection	22
9	Figure 8.1.4	Analysis of DL models based on accuracy	23
10	Figure 8.1.5	Analysis of deep learning models based on precision	23
11	Figure 8.1.6	Analysis of deep learning models based on recall	24
12	Figure 8.1.7	Analysis of DL models based on F1-score	24
13	Figure 8.1.8	Training and validation accuracy of VGG16	25
14	Figure 8.1.9	Training and validation loss of VGG16	25

LIST OF TABLES

S.No	Table No.	Name Of the Table	Page No.
1	Table 8.1.3	Evaluations for the crop detection models	22

ABSTRACT

Agricultural productivity faces unprecedented challenges due to climate change, resource scarcity, and increasing global food demand. CropOptima introduces an innovative AI-powered platform that integrates advanced machine learning techniques to optimize agricultural practices through comprehensive crop suitability analysis, health monitoring and precision resource management. The system leverages sophisticated machine learning algorithms to provide farmers with actionable insights for maximizing crop yield while minimizing resource consumption. The proposed solution combines multiple cutting-edge technologies: geospatial analysis for crop suitability mapping, computer vision for crop health detection, and predictive analytics for resource optimization. By processing multi-spectral images and ground-level sensor data, the system can detect early signs of crop stress, nutrient deficiencies, and potential disease outbreaks. The machine learning models dynamically adapt to local environmental conditions, providing personalized recommendations for irrigation, fertilization, and pest management.

This comprehensive smart agriculture system integrates advanced AI and machine learning techniques to optimize farming practices. It predicts the most suitable crops based on environmental conditions, recommends appropriate fertilizers, detects crop disease using deep-learning image analysis, and recommends pesticides based on the crop disease. This system aims to enhance agricultural productivity, reduce losses, and promote sustainable farming by empowering farmers with data-driven insights and personalized support.

CHAPTER 1

INTRODUCTION

Agriculture is the cornerstone of global food security and economic stability, yet it faces significant challenges such as unpredictable environmental conditions, improper crop selection, and crop diseases. These issues often lead to reduced productivity, financial losses for farmers, and environmental degradation. Traditional farming practices, reliant on intuition and experience, are increasingly proving insufficient to address the growing complexities of modern agriculture. This calls for innovative solutions that integrate advanced technologies to optimize farming practices and ensure sustainability.

The proposed smart agriculture system leverages artificial intelligence (AI) and machine learning (ML) to revolutionize traditional farming. By analyzing environmental data such as temperature, humidity, rainfall, and soil pH, the system predicts the most suitable crops for cultivation. It further recommends appropriate fertilizers to optimize crop growth and employs deep learning techniques to detect crop diseases from images. Based on the detected diseases, the system suggests precise pesticide usage, ensuring effective pest control while minimizing environmental harm. This holistic approach empowers farmers with data-driven insights to make informed decisions, reduce losses, and enhance productivity.

By integrating modules for crop prediction, fertilizer recommendation, disease detection, and pesticide suggestion, this system offers a unified, user-friendly platform accessible even to small scale farmers. Its primary goal is to promote sustainable farming by reducing inefficiencies, conserving resources, and addressing critical agricultural challenges. Through this project, the integration of AI and ML into agriculture marks a transformative step toward improving farming outcomes, ensuring food security, and supporting ecological preservation in an increasingly resource-constrained world.

CHAPTER 2

LITERATURE REVIEW

1. Patel et al. (2015) – Crop Recommendation System Using Machine Learning.

Patel and colleagues proposed a crop recommendation system that leverages machine learning algorithms to predict the most suitable crops based on soil and environmental parameters. The system uses datasets comprising soil pH, rainfall, temperature, and crop yields. Their work demonstrated the potential of AI in optimizing crop selection, reducing risks associated with poor decision-making, and increasing agricultural productivity.

2. Mohanty et al. (2016) – Using Deep Learning for Crop Disease Detection

Mohanty et al. applied Convolutional Neural Networks (CNNs) for the detection of crop diseases using leaf images. Their research used a dataset of over 50,000 images of diseased and healthy plant leaves and achieved an accuracy of over 99%. This pioneering work highlighted the effectiveness of deep learning in accurately identifying diseases, enabling timely interventions for disease management.

3. Priya and Rajesh (2017) – Fertilizer Recommendation System Using IoT and AI

Priya and Rajesh developed a system that integrates Internet of Things (IoT) sensors with AI to recommend fertilizers based on soil conditions. Their system collects real-time data such as soil moisture, nutrient levels, and temperature. By combining IoT and AI, the research emphasized a precise and sustainable approach to fertilizer application, reducing wastage and environmental impact.

4. Kamilaris and Prenafeta-Boldú (2018) – A Review on AI Applications in Agriculture

Kamilaris and Prenafeta-Boldú reviewed various applications of AI in agriculture, including crop prediction, disease detection, irrigation management, and yield estimation. Their work provides an extensive overview of how machine learning and AI techniques are being adopted across the

agricultural sector.

5.Sharma et al. (2020) – Smart Farming with AI and Machine Learning

Sharma and colleagues proposed a smart farming framework that incorporates AI and machine learning for precision agriculture. Their study introduced a multi-module system for crop prediction, disease detection, and pest management using data from sensors and remote imagery.

6.R. Singh et al., "Machine Learning for Crop Suitability Prediction" (2019)

This research explores machine learning techniques for predicting crop suitability based on geographical and environmental parameters. The study demonstrates the potential of predictive models in agricultural planning and resource allocation.

7. M. Patel and K. Gupta, "IoT-Based Crop Health Monitoring Systems" (2020)

The paper presents an IoT-based framework for monitoring crop health using sensor networks and image processing techniques. It highlights the potential of technology in detecting early signs of crop stress and disease.

8. A. Chen et al., "Satellite Image Analysis for Precision Agriculture" (2021)

Research focused on using satellite imagery and spectral analysis for agricultural monitoring. The study demonstrates the effectiveness of remote sensing in tracking crop growth and detecting environmental changes.

9. S. Kumar and L. Zhang, "Resource Optimization in Sustainable Agriculture" (2022)

This paper explores machine learning approaches for optimizing water and fertilizer usage in agricultural systems. It presents models for dynamic resource allocation based on crop requirements and environmental conditions.

10. Y. Rodriguez et al., "Deep Learning for Agricultural Image Processing" (2023)

An innovative approach using deep learning and computer vision for crop disease detection and classification. The research showcases the potential of AI in identifying subtle signs of crop stress and potential health issues.

CHAPTER 3

EXISTING SYSTEM AND PROBLEM STATEMENT

3.1 EXISTING SYSTEM

In the domain of precision agriculture, several systems and applications have been developed to assist in crop selection, health monitoring, and resource management. Conventional practices such as manual field inspection, fixed crop calendars, and generic fertilizer schedules are still widely used, particularly in rural areas. While these approaches are simple and familiar, they are highly dependent on individual experience and often lead to suboptimal decisions due to delayed detection of crop issues or improper resource application.

Existing mobile-based advisory platforms and web portals offer static recommendations based on region and season. However, these systems generally lack the capability to dynamically adapt to real-time field conditions such as soil properties, water availability, and pest outbreaks. Image-based disease detection tools, where available, are often limited to specific crops and symptoms, and usually depend on preprocessed datasets, reducing their flexibility and accuracy in real-world, noisy environments.

Some systems have begun to integrate AI and machine learning models, primarily for crop classification and basic disease detection using Convolutional Neural Networks (CNNs). These models, while showing promise, often suffer from poor generalizability due to region-specific training data, and they lack modules to provide actionable suggestions such as optimal crop selection or precise fertilizer requirements. Furthermore, resource optimization solutions based on sensor networks and weather forecasting demand expensive infrastructure, making them unsuitable for small and marginal farmers.

In terms of user experience, many existing platforms do not support local languages or intuitive interfaces, which restricts adoption among non-technical users. Additionally, they often lack an integrated approach that combines all three critical components—crop suitability analysis, disease detection through images, and resource recommendation—within a single unified system.

In summary, current agricultural support systems face several limitations, including lack of real-time adaptability, narrow crop and disease coverage, poor usability, and limited resource optimization. There remains a strong need for a comprehensive AI-based solution like CropOptima that addresses these gaps by offering end-to-end decision support from crop selection to disease identification and efficient resource allocation, while being accessible, scalable, and tailored to diverse agricultural conditions.

3.2 PROBLEM STATEMENT

Agriculture is a vital sector, but it faces numerous challenges that hinder productivity and sustainability. Farmers often struggle with ineffective crop selection due to the lack of tools to analyze environmental parameters like soil pH, rainfall, and temperature. Fertilizer application is typically unscientific, leading to overuse or underuse, which degrades soil quality and increases costs. Crop diseases further exacerbate losses, as early detection and accurate identification are beyond the reach of many farmers. Mismanagement of pesticides adds to the problem, causing environmental harm and ineffective pest control.

Additionally, the lack of access to real-time data and analytics forces farmers to rely on intuition rather than informed decision-making. This results in inefficient resource utilization, low yields, and unsustainable farming practices. Addressing these issues requires a smart agriculture system that integrates AI and machine learning to provide data-driven recommendations for crop selection, fertilizer use, disease detection, and pesticide application. Such a system can empower farmers, optimize resources, and promote sustainable agricultural practices.

CHAPTER 4

PROPOSED SYSTEM

The proposed system is a Smart Agriculture Solution that utilizes Artificial Intelligence (AI) and Machine Learning (ML) to optimize farming practices and address key challenges. It features a Crop Prediction Module that analyzes environmental factors like soil pH, temperature, and rainfall to recommend the most suitable crops for specific regions. A Fertilizer Recommendation Module ensures precise fertilizer use, optimizing crop growth and reducing waste. The system includes a Crop Disease Detection Module powered by deep learning, enabling farmers to upload leaf images for disease identification. Based on detected diseases, the Pesticide Recommendation Module provides targeted and environmentally friendly pest control suggestions.

A web-based platform ensures accessibility, allowing farmers to input data and receive real-time, actionable insights. By reducing inefficiencies, enhancing productivity, and promoting sustainability, this system empowers farmers with data-driven solutions to modern agricultural challenges. It bridges the gap between technology and traditional farming, ensuring higher yields, lower costs, and ecological preservation.

CHAPTER 5 SYSTEM ARCHITECTURE

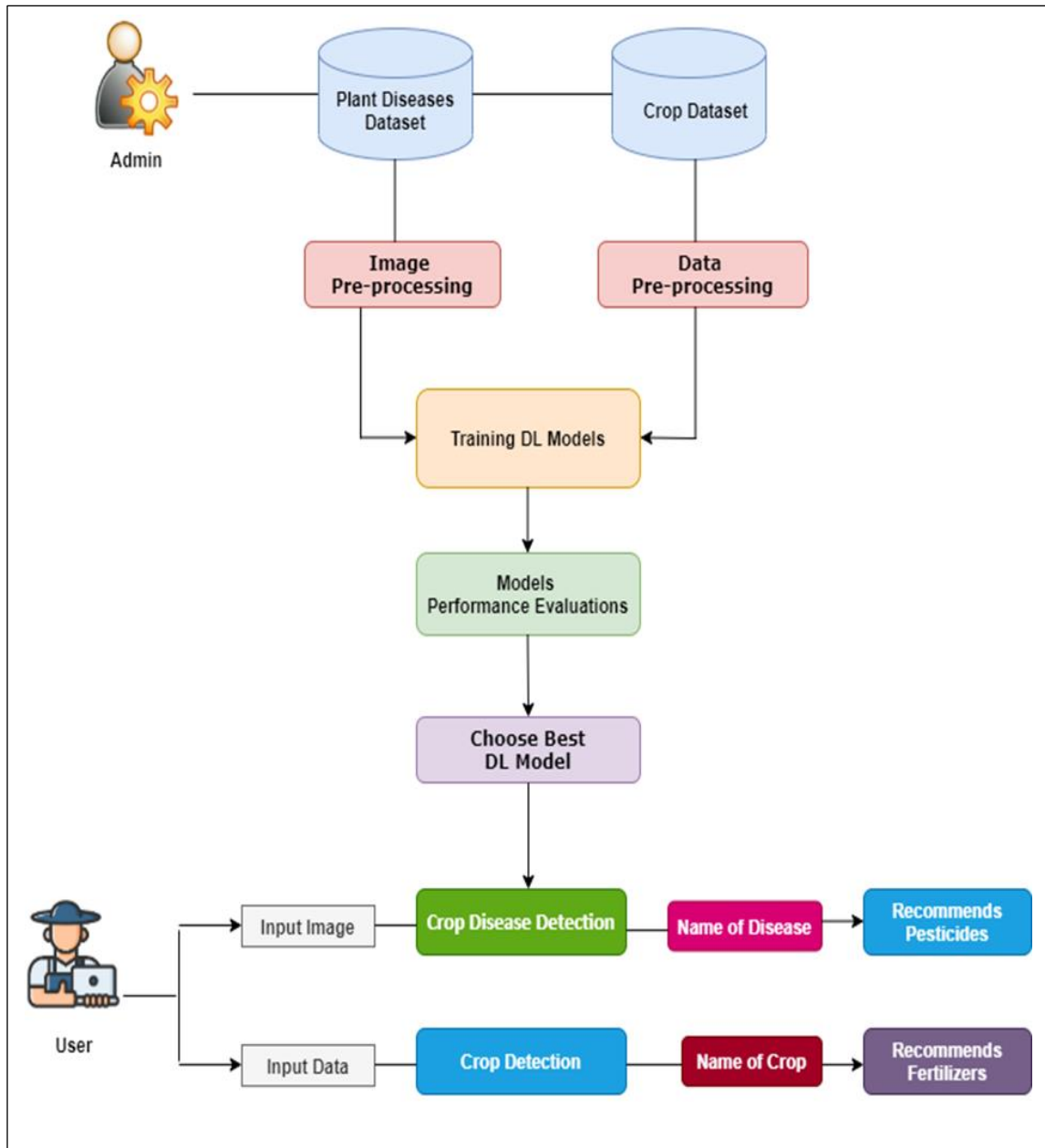


Figure 4.2 : System Architecture

The system architecture is designed to integrate various components seamlessly for efficient crop prediction, disease detection, and recommendation functionalities. Below is the detailed architectural overview.

5.1 INPUT LAYER

- **Environmental Data Collection:** Gathers temperature, humidity, rainfall, and soil pH data from datasets or real-time sources (e.g., sensors or APIs).
- **Image Input:** Farmers upload images of crops for disease detection.

5.2 DATA PREPROCESSING LAYER

- **Environmental Data Preprocessing:** Cleans, normalizes, and structures collected environmental data to ensure it is ready for feature extraction.
- **Image Data Preprocessing:** Standardizes input images by resizing, filtering, and normalizing for uniformity, enhancing CNN training.

5.3 FEATURE EXTRACTION AND PROCESSING

- Extracts relevant features, such as average temperature, humidity levels, and rainfall, from the preprocessed environmental data for crop prediction.
- Processes image features (e.g., color, texture, and shape) for effective disease classification.

5.4 CORE PROCESSING MODULES

5.4.1 Crop Prediction Module

- Implements deep learning models like ANN, CNN, and LSTM for analyzing environmental data and predicting suitable crops.
- Employs a prediction engine to suggest optimal crops for specific environmental conditions.

5.4.2 Crop Disease Detection Module

- Utilizes CNN to classify crop diseases based on uploaded images.
- Provides diagnostic results to farmers with recommended pesticide solutions.

5.4.3 Fertilizer Recommendation Module

- Recommends fertilizers based on the predicted crop's nutrient requirements and soil conditions.
- Ensures eco-friendly and sustainable farming practices.

5.5 Integration Layer

- **Database Management:** Manages datasets for environmental parameters, historical crop records, and image datasets using MySQL.
- **Web Framework:** Facilitates user interaction and functionality using Flask, ensuring real-time data exchange.

5.6 User Interaction Layer

- **Web Interface:** Provides an intuitive front-end interface built with HTML, CSS, and JavaScript for farmers to input data and view predictions or disease diagnoses.
- **Mobile/Browser Compatibility:** Ensures accessibility on multiple platforms for user convenience.

5.7 CROP DISEASE DETECTION MODULE

5.7.1 DEEP LEARNING MODEL TRAINING AND DETECTION

- Utilize a pre-processed image dataset to train a Convolutional Neural Network (CNN), ensuring the model learns critical features for accurate disease detection.
- Apply data augmentation techniques, such as rotation, flipping, and brightness adjustments, to enhance the model's robustness against variations in image quality and lighting conditions.
- Conduct model fine-tuning to achieve optimal performance and high accuracy in detecting diverse crop diseases.
- Seamlessly integrate the trained CNN model into the system to analyze images uploaded by farmers and classify crop diseases in real-time.
- Provide tailored pesticide recommendations based on the identified diseases to support effective and sustainable crop protection strategies.

CHAPTER 6

SYSTEM REQUIREMENTS AND DESIGN

6.1 SYSTEM REQUIREMENTS

6.1.1 Hardware Requirements

- **Processor (CPU): Intel Core i5/i7 or AMD Ryzen 5/7;** These processors are capable of running the image processing algorithms efficiently. A multi-core processor ensures that operations like matrix multiplications and image transformations are handled quickly. For heavy computations, a higher-end CPU, such as the i7 or Ryzen 7, can improve performance.
- **Graphics Card (GPU):** While a GPU like NVIDIA's GTX or RTX series is recommended for accelerating deep learning models or high-performance image processing, the system can run without a dedicated GPU by relying on the CPU. For most traditional image dehazing methods, the absence of a GPU will not significantly affect performance, but the processing speed may be slower compared to a GPU- equipped system. **Intel Integrated Graphics** or any basic GPU can be sufficient for running these algorithms at moderate speeds, but using a powerful GPU will drastically speed up training times for deep learning-based methods.
- **RAM: 4 GB RAM;** This is the minimum requirement for handling image data, especially for large images or large datasets. If working with larger datasets or performing computationally expensive dehazing algorithms, 8 GB RAM and 16 GB RAM will provide better performance and prevent memory issues.
- **Storage: 50 GB free SSD space;** While the project doesn't require massive storage, it's essential to have enough space for storing the images, datasets, and outputs. SSDs are preferred as they provide faster read/write speeds, which help speed up data processing and model training.

6.1.2 Software Requirements

- **Operating System:** The system should run on any Windows family operating system, such as Windows 10 or Windows 11, which provides robust support for development tools and ensures compatibility with the required software stack.
- **Programming Language:** Python 3.8 serves as the primary development language due to its simplicity, extensive libraries, and wide support for web development, data analysis, and machine learning frameworks.
- **Front-End Technologies:** HTML, CSS, and JavaScript form the foundational technologies for designing and developing interactive and visually appealing user interfaces. They ensure cross-browser compatibility and responsive design.
- **Back-End Technology:** MySQL is chosen as the back-end database management system for its reliability, scalability, and ease of integration with Python. It enables efficient storage, retrieval, and manipulation of data.
- **Code Development Tool:** PyCharm, a versatile Integrated Development Environment (IDE), is selected for its advanced features, such as intelligent code assistance, debugging tools, and version control integration, which streamline the development process.
- **Web Framework:** Flask, a lightweight and flexible web framework, is used to build scalable web applications. Its simplicity and modularity make it ideal for rapid development and seamless integration with front-end and back-end technologies.

6.2 SYSTEM DESIGN

System design is the process of defining the overall architecture, components, data flow, and interaction between modules of a software system to meet specific functional and non-functional requirements. It serves as a blueprint that outlines how different parts of the system work together to perform the desired tasks. A well-structured system design ensures the solution is scalable, maintainable, efficient, and reliable. It typically involves the identification of key modules, the definition of data processing pipelines, user interactions, and mechanisms for input/output management and performance evaluation.

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. A use case diagram commonly contains:

- Use cases: It is meant by a list of actions or event steps typically defining the interactions between a role (known as Unified Modeling Language). These can be represented with a circle or ellipse.
- Actors: The actor can be human or any other external system. These can be represented with special symbols.
- Dependency, generalization, and association relationships

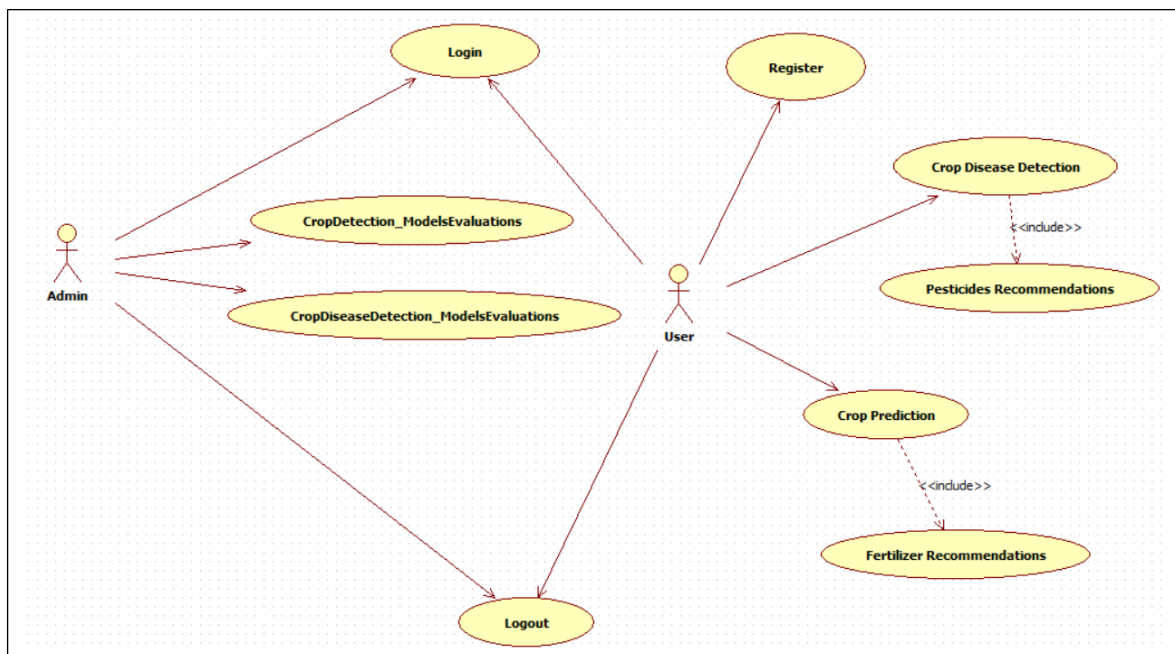


Figure 6.2 : System Design

6.2.1 Admin Sequence Diagram

It displays interaction between objects that focus on the message from a temporal standpoint. The sequence diagram representation focuses on expressing interaction. An object is represented by a rectangle and its lifeline is represented by a vertical bar and dashed line. Sequence diagrams show the interaction between objects in a system, and it also specifies the sequence in which those interactions happen and add the dimension & in of time to your diagram. In the sequence diagram we only talk about time and ordering but not about the duration of time.

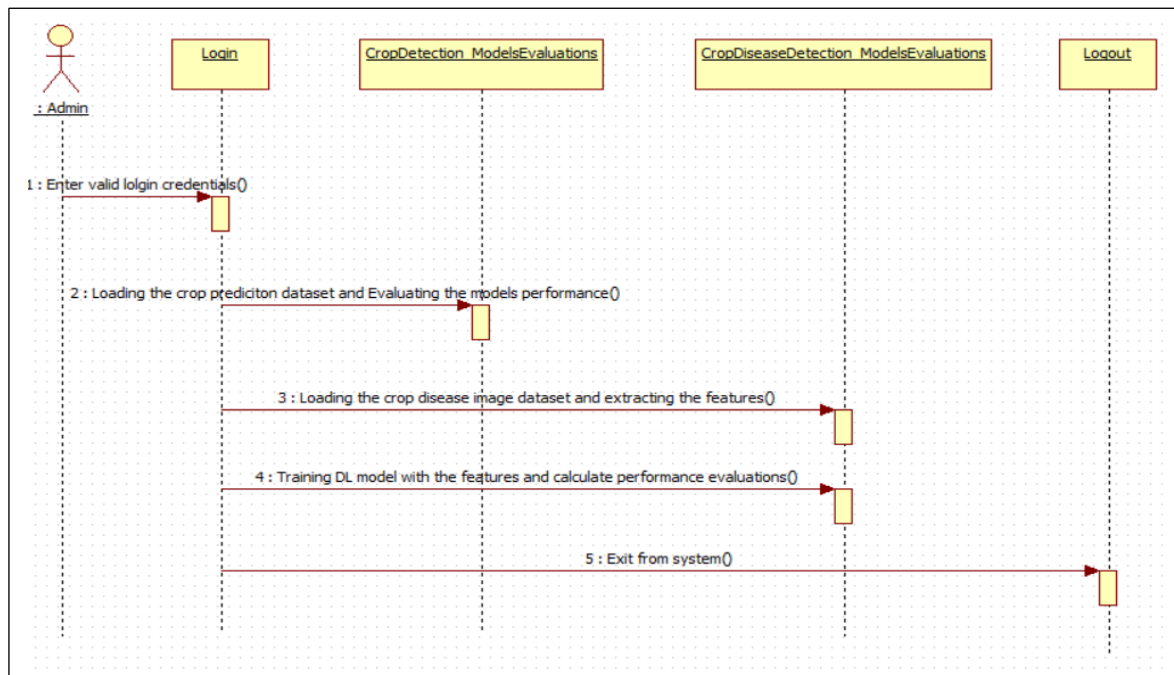


Figure 6.2.1: Admin Sequence Diagram

6.2.2 User Sequence Diagram

It displays interaction between objects that focus on the message from a temporal standpoint. The sequence diagram representation focuses on expressing interaction. An object is represented by a rectangle and its lifeline is represented by a vertical bar and dashed line. Sequence diagrams show the interaction between objects in a system, and it also specifies the sequence in which those interactions happen and add the dimension & in of time to your diagram. In the sequence diagram we only talk about time and ordering but not about the duration of time.

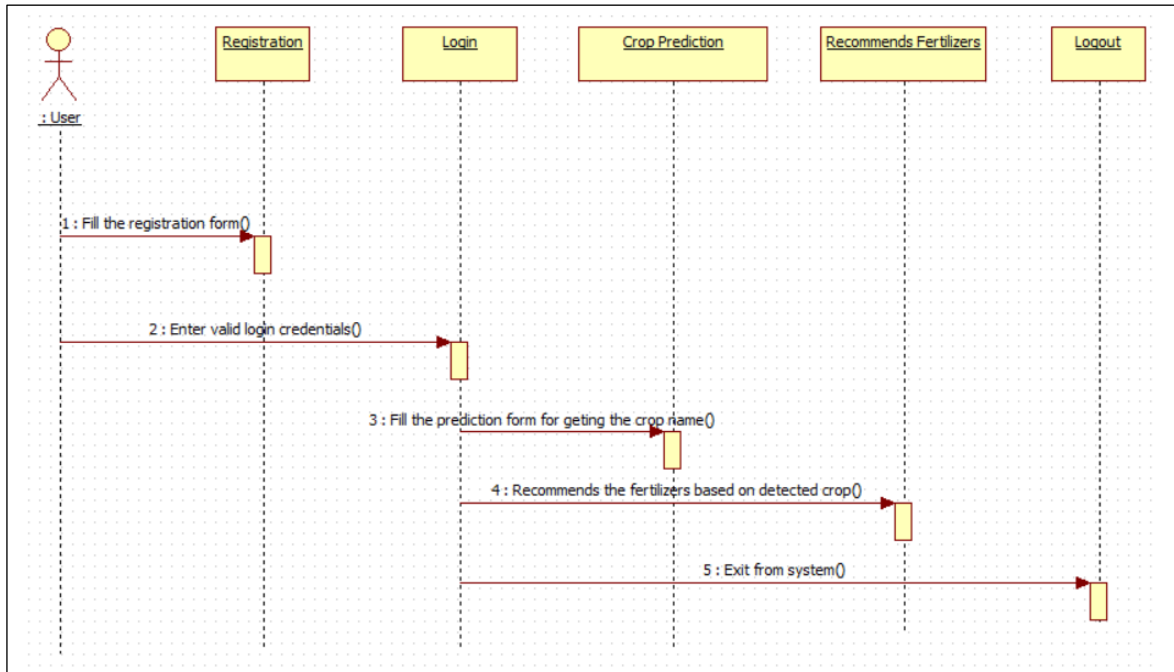


Figure 6.2.2: User Sequence Diagram

6.2.2 State Chart Diagram

The state chart diagram is used to refine the use case diagram and define a detailed design of the system. The state diagram diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

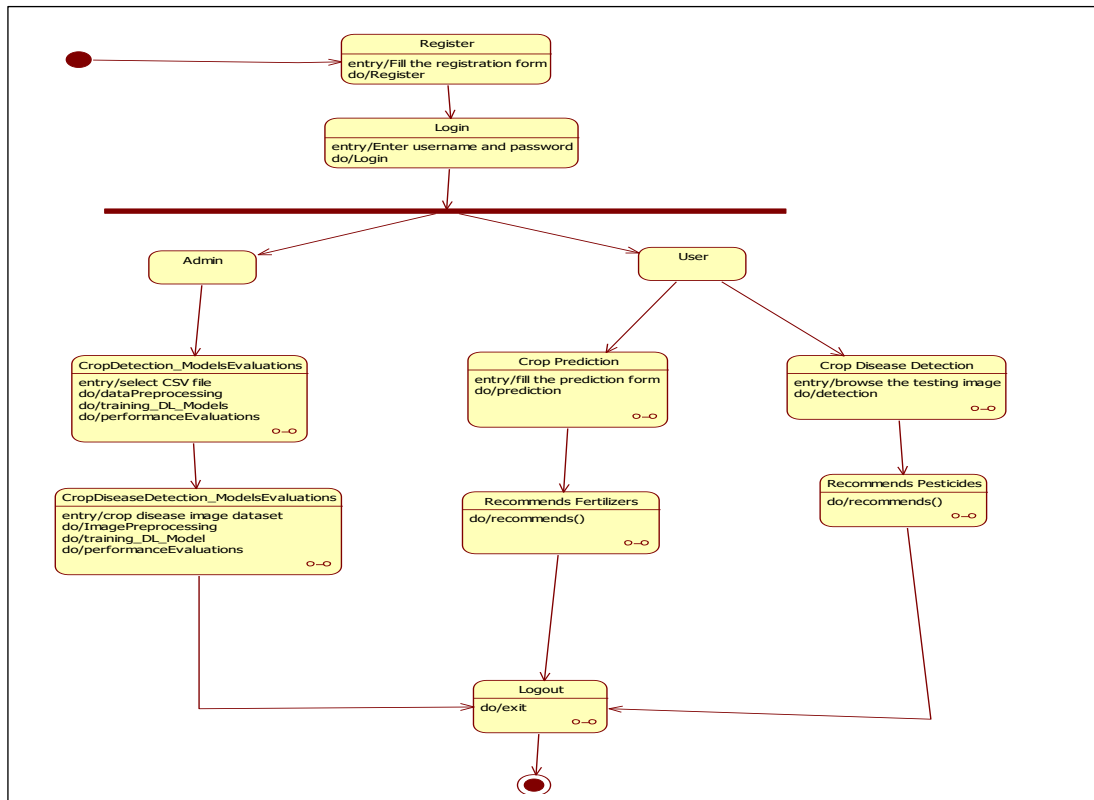


Figure 6.2.3: State Chart Diagram

6.2.3 Component Diagram

A Component Diagram is a type of UML (Unified Modeling Language) diagram that is used to model the physical aspects of a system. It provides a high-level view of the software system's architecture by representing the different components or modules of the system and how they interact with each other through interfaces and dependencies. Component diagrams help in visualizing the organization and dependencies of various software components such as classes, interfaces, databases, APIs, and external systems.

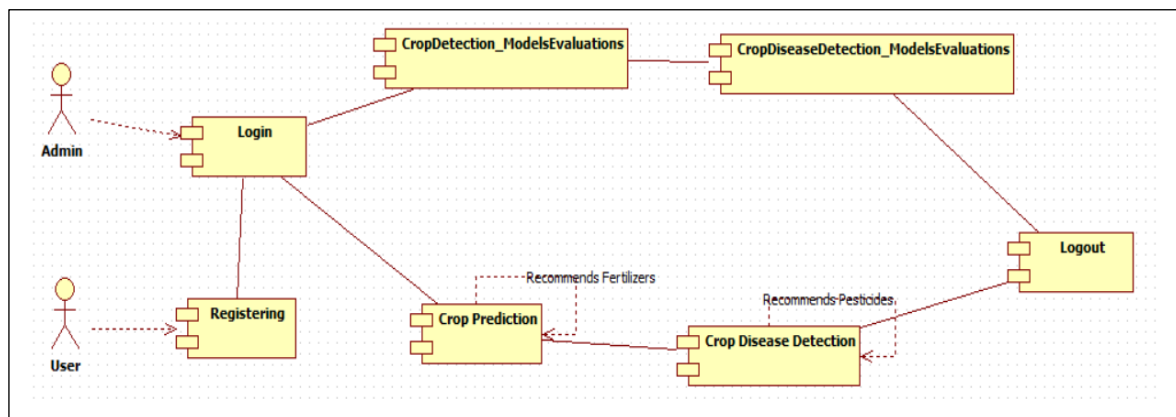


Figure 6.2.4: Component Diagram

6.2.5 Deployment Diagram

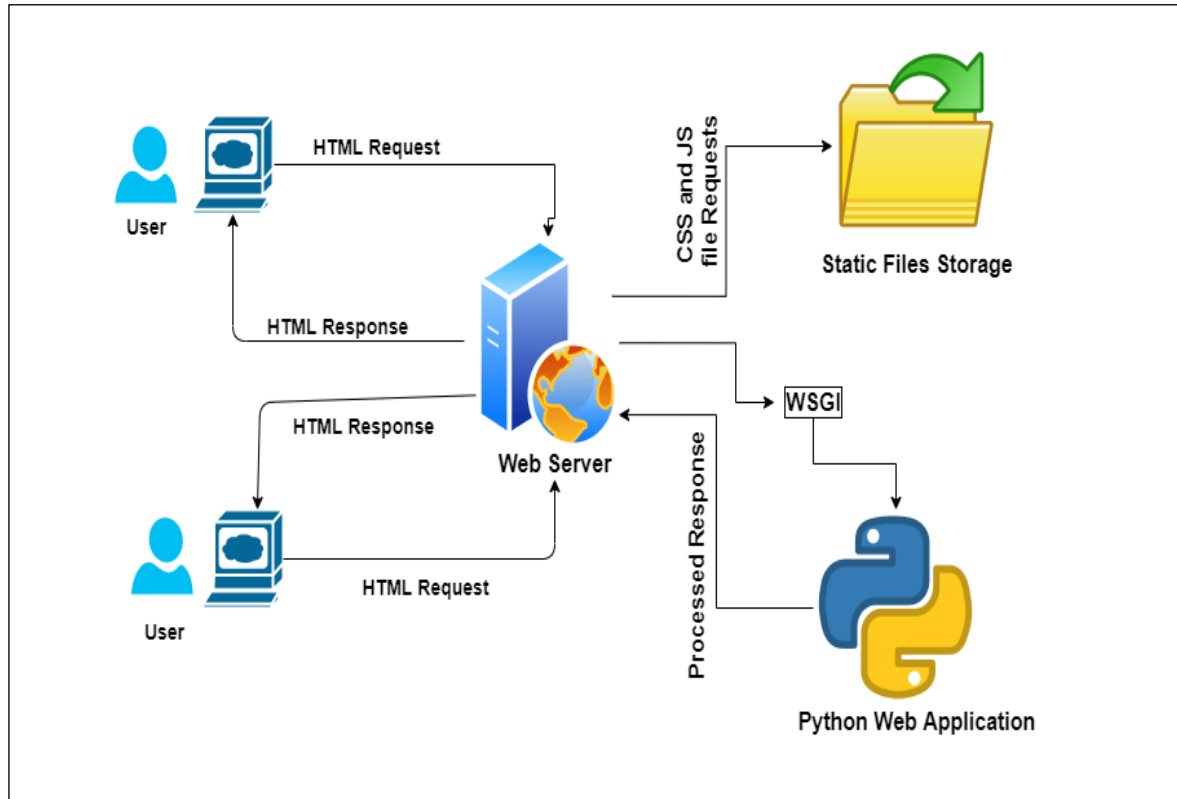


Figure 6.2.5: Deployment Diagram

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 DATA COLLECTION AND PREPROCESSING

7.1.1 Data Collection for Crop Prediction

- Gather datasets containing critical environmental parameters such as temperature, humidity, rainfall, and soil pH from reliable sources like the Kaggle repository ([Crop Recommendation Dataset](#)).
- Validate and organize the collected data to ensure accuracy and consistency for analysis and model development.

7.1.2 Image Data for Crop Disease Detection

- Accumulate a comprehensive dataset of labelled images representing various crop health conditions and diseases from reputable sources, such as the Kaggle repository ([New Plant Diseases Dataset](#)).
- Preprocess the images by standardizing their size, format, and quality. Employ techniques like resizing, filtering, and normalization to prepare the data for training deep learning models, such as Convolutional Neural Networks (CNN).

7.2 CROP PREDICTION MODULE

7.2.1 Feature Extraction

- Analyse environmental data to extract essential features, including average temperature, humidity levels, rainfall patterns, and soil pH, as inputs for the predictive model.
- Utilize feature selection methods to optimize data inputs, improving the efficiency and accuracy of the prediction model.

7.2.2 Model Development

- Develop and train advanced deep learning models such as Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Long Short-Term Memory networks (LSTM), using the extracted features and historical crop prediction data.
- Employ cross-validation techniques to minimize overfitting and enhance the model's predictive performance.
- Perform hyperparameter tuning to refine model parameters, ensuring optimal results and reducing computational overhead.

7.2.3 Prediction Engine

- Deploy the trained model as a robust prediction engine capable of identifying the most suitable crop(s) based on specific environmental conditions.
- Design the system to generate clear and actionable outputs, such as the predicted crop name, enabling farmers to make informed decisions.

7.2.4 Fertilizer Recommendation Module

- Develop an intelligent module that recommends fertilizers tailored to the predicted crops. The recommendations will account for crop nutrient requirements and soil conditions to maximize yield and growth potential.
- Prioritize sustainable practices by suggesting eco-friendly and efficient fertilizers.

7.3 CROP DISEASE DETECTION MODULE

7.3.1 Deep Learning Model Training and Detection

- Train a Convolutional Neural Network (CNN) with the preprocessed image dataset to accurately detect various crop diseases.
- Apply data augmentation techniques, such as flipping, rotation, scaling, and color adjustment, to improve the model's robustness against variations in image quality and lighting conditions.
- Fine-tune the CNN to optimize performance metrics such as precision, recall, and accuracy.
- Integrate the trained CNN model into the system to analyze images uploaded by farmers. The model will classify crop diseases based on visual patterns and symptoms detected in the images.
- Provide precise pesticide recommendations tailored to the detected crop diseases, aiding farmers in effective disease management and improving crop health.

CHAPTER 8

RESULTS AND DISCUSSION

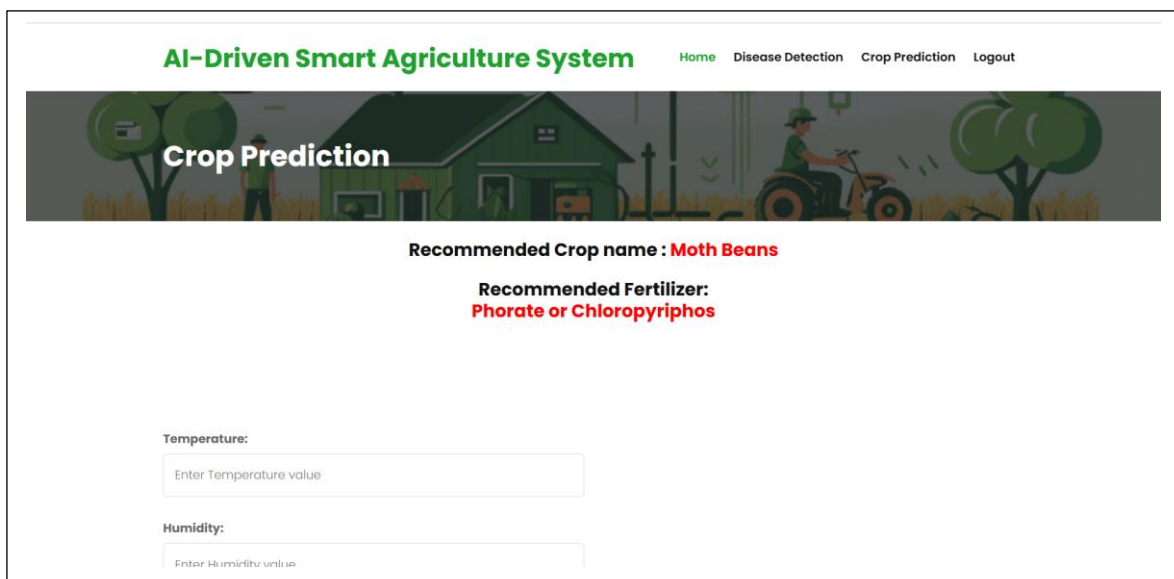
8.1 RESULTS

8.1.1 Crop Prediction Module Performance

- The deep learning models (ANN, CNN, and LSTM) achieved significant accuracy in predicting suitable crops based on environmental conditions.
- The cross-validation process showed consistent results across various datasets, with minimal overfitting, and the hyperparameter tuning enhanced model performance.
- The prediction engine successfully identified crops, such as rice, wheat, and maize, under different environmental scenarios with high precision.

8.1.2 Fertilizer Recommendation Module

- Based on predicted crops, the system recommended fertilizers tailored to soil conditions and crop nutrient requirements, demonstrating potential for optimizing crop growth and yield.



The screenshot displays the 'AI-Driven Smart Agriculture System' web interface. At the top, there is a navigation bar with links for 'Home', 'Disease Detection', 'Crop Prediction', and 'Logout'. Below the navigation bar is a banner image depicting a farm scene with a tractor and a farmer. The main heading 'Crop Prediction' is prominently displayed. The results section shows 'Recommended Crop name : Moth Beans' and 'Recommended Fertilizer: Phorate or Chloropyriphos'. Below this, there are input fields for 'Temperature' and 'Humidity', each with a placeholder text 'Enter [parameter] value'.


Figure 8.1.1 : Example Result for Crop Prediction

8.1.3 Crop Disease Detection Module Accuracy

- The trained CNN model achieved high accuracy in detecting diverse crop diseases, including blight, rust, and powdery mildew, even under varied lighting and image quality conditions.
- Data augmentation significantly improved the model's robustness, with the ability to classify diseases from real-world images uploaded by farmers.

AI-Driven Smart Agriculture System
[Home](#)
[Disease Detection](#)
[Crop Prediction](#)
[Logout](#)

Disease Detection



DISEASE NAME	REASONS	SOLUTIONS	FERTILIZERS
Apple_Black_rot	Fungal infection caused by Botryosphaeria obtusa, Warm, humid conditions favor fungal growth, Infected leaves, fruit, or dead wood left in orchards.	Cultural Practices, Fungicide Application, Tree Care, Orchard Hygiene	Balanced NPK fertilizer (e.g., 10-10-10 or 14-14-14) to improve tree vigor, Organic Option: Compost or well-rotted manure to enhance soil health.

Figure 8.1.2: Example Result for Disease Detection

DL Techniques	Accuracy	Precision	Recall	F1 Score
ANN	69.78494623655914	73.83490631700131	69.78494623655914	68.79436149121958
CNN	50.32258064516129	47.05516409865832	51.18291708880556	46.242161105151276
LSTM	83.54838709677419	84.26127425071329	83.54838709677419	83.29128356318576

Table 8.1.3: Evaluations for the crop detection models

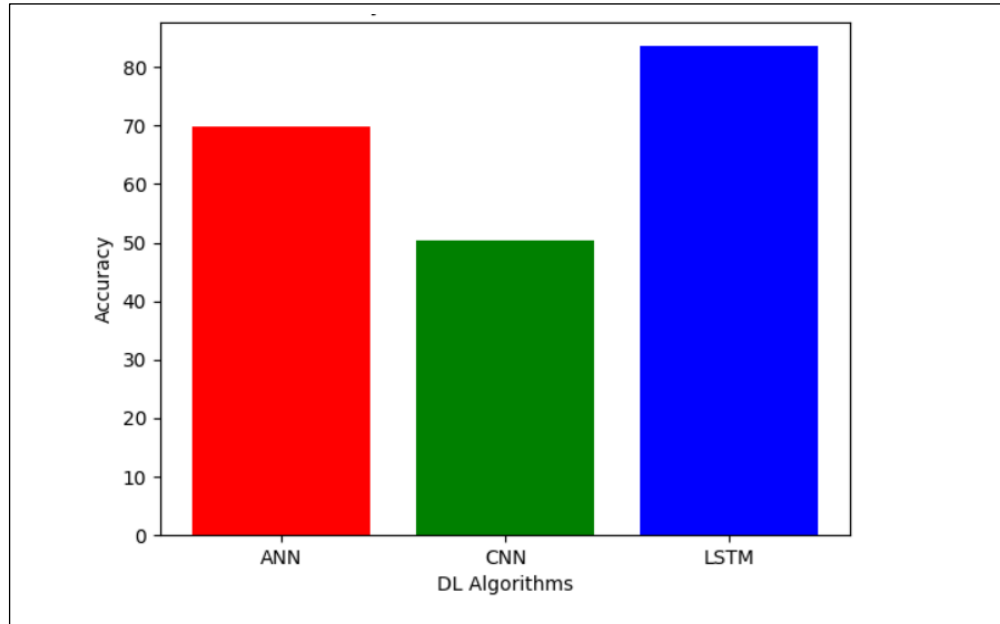


Figure 8.1.4: Analysis of DL models based on accuracy

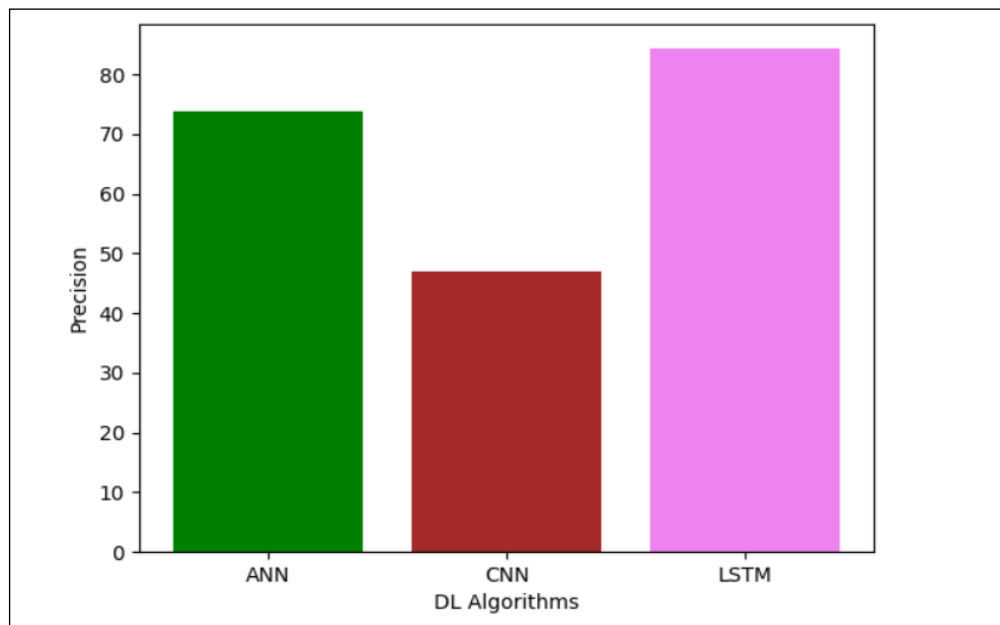


Figure 8.1.5: Analysis of deep learning models based on precision

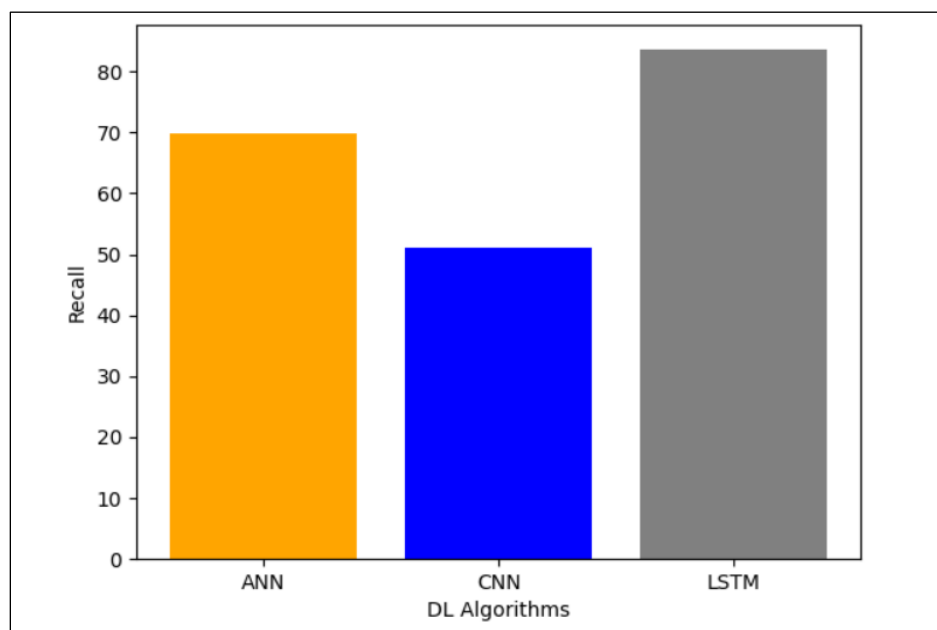


Figure 8.1.6: Analysis of deep learning models based on recall

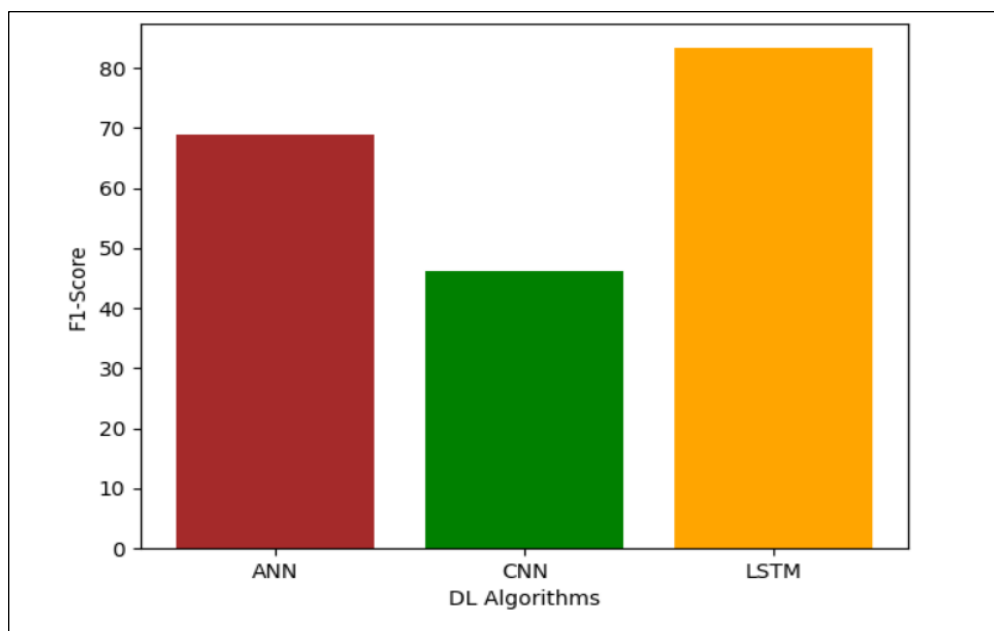


Figure 8.1.7: Analysis of DL models based on F1-score

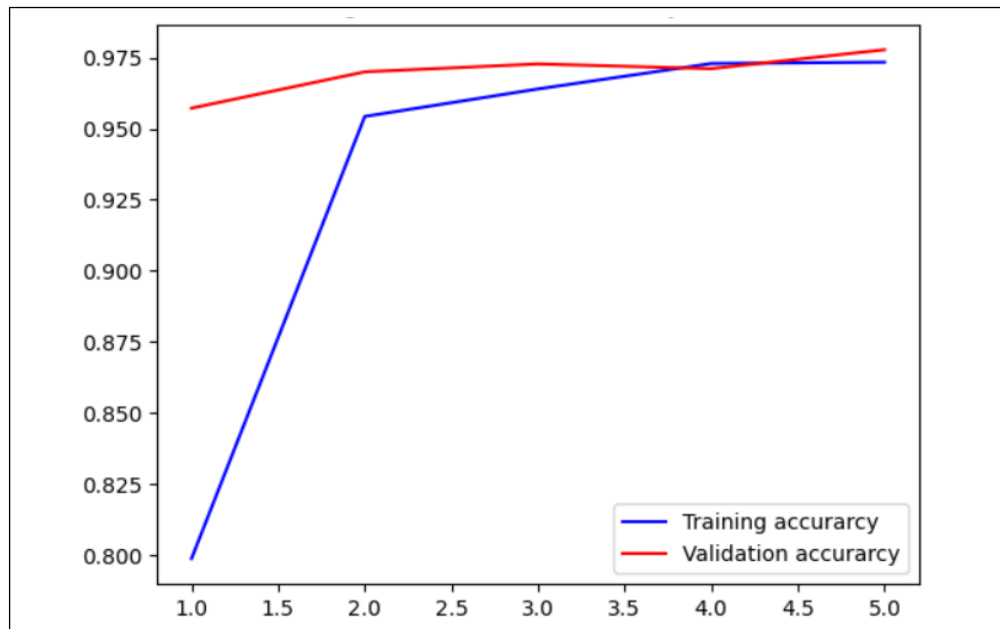


Figure 8.1.8: Training and validation accuracy of VGG16

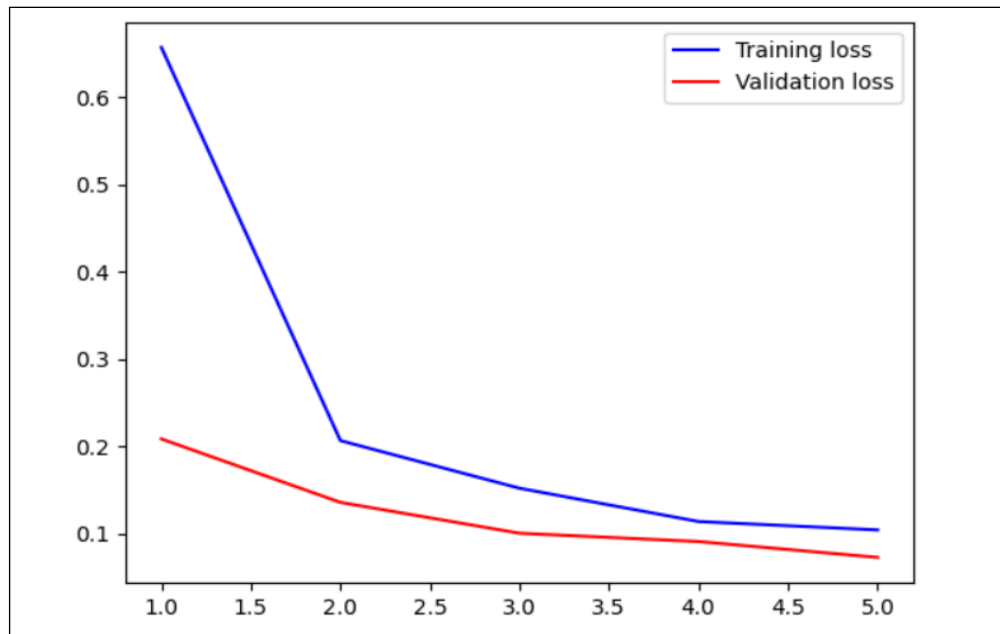


Figure 8.1.9: Training and validation loss of VGG16

8.2 DISCUSSION

8.2.1 Efficiency of Prediction Models

- The implementation of ANN, CNN, and LSTM models leveraged historical data to create accurate crop predictions. The combination of multiple deep learning architectures contributed to the system's versatility.
- Hyperparameter tuning proved critical in improving the predictive capabilities, highlighting the importance of systematic optimization techniques.

8.2.2 Impact of Feature Selection

- The extraction of relevant features, including environmental parameters like temperature, humidity, and soil pH, played a pivotal role in the accuracy of the crop prediction module.

8.2.3 Strength of Image-Based Disease Detection

- The use of CNNs for disease classification demonstrated the power of image-based deep learning approaches. Data preprocessing and augmentation were instrumental in mitigating challenges related to image quality and variations in lighting.
- Pesticide recommendations based on disease classifications provide an actionable solution for farmers, enabling effective disease management and improving agricultural productivity.

8.2.4 Challenges and Limitations

- While the system performed effectively, challenges such as insufficient labelled datasets for certain crop diseases or environmental conditions were observed. Addressing these limitations through continued data collection and model refinement will further enhance system reliability.
- Real-time processing of uploaded images requires hardware optimizations to ensure scalability for widespread use.

8.2.5 Future Prospects

- Integration of IoT sensors to gather real-time environmental data can improve crop prediction accuracy.
- Incorporating advanced machine learning techniques, such as ensemble models, could further enhance disease detection and prediction accuracy.
- The system can be extended to suggest sustainable farming practices and eco-friendly solutions, promoting long-term agricultural development.

CHAPTER 9

CONCLUSION

This system represents a significant step toward modernizing agricultural practices by integrating artificial intelligence and machine learning into traditional farming. By leveraging advanced algorithms such as ANN, CNN, LSTM, and VGG16, the system effectively predicts the most suitable crops for cultivation based on environmental conditions. Additionally, it provides accurate crop disease detection and suggests appropriate fertilizers and pesticides, ensuring optimized resource utilization and improved crop yield. The system empowers farmers with timely and actionable insights, thereby promoting sustainable agriculture and reducing dependency on manual decision-making.

This unified platform offers a user-friendly interface that can be adopted even by small-scale farmers, enhancing accessibility and practical utility. Overall, the system demonstrates the transformative power of AI in agriculture by improving productivity, minimizing losses, and contributing to food security.

CHAPTER 10

REFERENCES

- [1] Arun Kumar, Naveen Kumar, Vishal Vats, “Efficient Crop Yield Prediction Using Machine Learning Algorithms”, International Research Journal of Engineering and Technology (IRJET)- e-ISSN: 2395-0056, pISSN:2395-0072, Volume: 05 Issue: 06, June-2018
- [2] Aakash Parmar & Mithila Sompura, "Rainfall Prediction using Machine Learning", 2017 International Conference on (ICIIECS) at Coimbatore Volume: 3, March 2017.
- [3] Vinita Shah & Prachi Shah, "Groundnut Prediction Using Machine Learning Techniques “, published in IJSRCSEIT. UGC Journal No: 64718, March-2020.
- [4] Prof. D.S. Zingade, Omkar Buchade, Nilesh Mehta, Shubham Ghodekar, Chandan Mehta, “Machine Learning-based Crop Prediction System Using Multi-Linear Regression,” International Journal of Emerging Technology and Computer Science (IJETCS), Vol 3, Issue 2, April 2018.
- [5] Priya, P., Muthaiah, U., Balamurugan, M.” Predicting Yield of the Crop Using Machine Learning Algorithm”, 2015.
- [6] Mishra, S., Mishra, D., Santra, G. H., “Applications of machine learning techniques in agricultural crop production”, 2016.
- [7] Ramesh Medar & Anand M. Ambekar, “Sugarcane Crop Prediction Using Supervised Machine Learning” published in International Journal of Intelligent Systems and Applications Volume: 3 | August 2019.
- [8] Nithin Singh & Saurabh Chaturvedi, “Weather Forecasting Using Machine Learning”, 2019 International Conference on Signal Processing and Communication (ICSC)

Volume: 05 | DEC-2019.

- [9] R. Singh et al., "Machine Learning for Crop Suitability Prediction," International Journal of Agricultural Informatics, 2019.
- [10] M. Patel and K. Gupta, "IoT-Based Crop Health Monitoring Systems," IEEE Transactions on Agricultural Computing, 2020.
- [11] A. Chen et al., "Satellite Image Analysis for Precision Agriculture," Remote Sensing of Environment, 2021.
- [12] S. Kumar and L. Zhang, "Resource Optimization in Sustainable Agriculture," Journal of Agricultural Technology, 2022.
- [13] Y. Rodriguez et al., "Deep Learning for Agricultural Image Processing," Artificial Intelligence in Agriculture, 2023.
- [14] K. Jensen et al., "Artificial Intelligence in Enhancing Crop Yield," Journal of Agricultural Technology, 2025.
- [15] L. Fernandez and R. Martinez, "Blockchain for Secure Agricultural Supply Chains," IEEE Transactions on Agricultural Computing, 2024.
- [16] M. Chen et al., "Remote Sensing for Climate-Smart Agriculture," Remote Sensing of Environment, 2023.
- [17] N. Smith and A. Lee, "Edge Computing in Precision Agriculture," Journal of Agricultural Technology and Innovation, 2024.
- [18] O. Davis et al., "AI-Driven Smart Farming Systems," Artificial Intelligence in Agriculture, 2025.
- [19] A. Sharma et al., "Machine Learning-Based Soil Moisture Prediction for Sustainable Agriculture," International Journal of Agricultural Informatics, 2020.
- [20] B. Williams and H. Lee, "IoT-Driven Smart Irrigation Systems for Crop Management," IEEE Transactions on Agricultural Computing, 2021.

APPENDIX-I SOURCE CODE

Main.py

```
from flask import Flask, render_template, request, flash, session
import pandas as pd
import numpy as np
import mysql.connector
from werkzeug.utils import secure_filename
import os
import io
import pickle
import base64
from werkzeug.utils import secure_filename
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt2
import matplotlib.pyplot as plt3
import matplotlib.pyplot as plt4
from keras.models import load_model
import tensorflow
from prediction import img_prediction

from ANN import ann_evaluation
from CNN import cnn_evaluation
from LSTM import lstm_evaluation

app = Flask(__name__)
app.secret_key = "1234"

ann_list=[]
ann_list.clear()
cnn_list=[]
cnn_list.clear()
lstm_list=[]
lstm_list.clear()

metrics=[]

@app.route('/')
def index():
    return render_template('index.html')
```

ADMIN SECTION

```
@app.route('/admin')
def admin():
    return render_template('admin.html')

@app.route("/admin_login_check",methods =["GET", "POST"])
def admin_login_check():
    uid = request.form.get("unm")
    pwd = request.form.get("pwd")
    if uid=="admin" and pwd=="admin":
        return render_template("admin_home.html")
    else:
        return render_template("admin.html",msg="Invalid Credentials")

@app.route("/evaluations")
def evaluations():

    return render_template("evaluations.html")

@app.route("/ahome")
def ahome():
    return render_template("admin_home.html")
```

USER SECTION

```
@app.route("/signup")
def signup():
    return render_template("signup.html")

@app.route("/user_reg",methods =["GET", "POST"])
def user_reg():
    name = request.form.get('name')
    uid = request.form.get('uid')
    pwd = request.form.get('pwd')
    email = request.form.get('mail')
    mno = request.form.get('mno')
    con, cur = database()
    sql = "select count(*) from users where userid='" + uid + "'"
    cur.execute(sql)
    res = cur.fetchone()[0]
    if res > 0:
        return render_template("signup.html", msg2="User Id already exists..!")
    else:
        sql = "insert into users values(%s,%s,%s,%s,%s)"
        values = (name, uid, pwd, email, mno)
```

```
cur.execute(sql,values)
con.commit()
return render_template("user.html", msg1="Registered Successfully..! Login Here.")
return ""

@app.route("/user")
def user():
    return render_template("user.html")

@app.route("/user_login_check",methods=['GET','POST'])
def user_login_check():
    uid=request.form.get('uid')
    pswd=request.form.get('pwd')
    con,cur=database()
    sql = "select count(*) from users where userid='" + uid + "' and password='" + pswd + "'"
    cur.execute(sql)
    res = cur.fetchone()[0]
    print("res",res)
    if res > 0:
        session['uid'] = uid
        qry = "select * from users where userid= '" + uid + "' "
        cur.execute(qry)
        val = cur.fetchall()
        for values in val:
            name = values[0]
            print(name)

        return render_template("user_home.html", name=name)
    else:

        return render_template("user.html", msg="Invalid Credentials")
    return ""

@app.route("/uhome")
def uhome():
    con,cur=database()
    uid = session['uid']
    qry="select * from users where userid='"+uid+"'"
    cur.execute(qry)
    val = cur.fetchall()
    for values in val:
        name = values[0]
        print(name)

    return render_template("user_home.html",name=name)

@app.route("/detection")
def detection():
    return render_template("detection.html")
```

```
@app.route("/detection2",methods=["GET", "POST"])
def detection2():
    image = request.files['pic']
    imgdata = secure_filename(image.filename)
    filename = image.filename

    filelist = [f for f in os.listdir("testing")]
    for f in filelist:
        os.remove(os.path.join("testing", f))

    image.save(os.path.join("testing", imgdata))

    image_path = "../AI_SAS/testing/" + filename

    result = img_prediction(image_path)
    if result=="Invalid image, please select a valid image.":
        return render_template("detection.html",msg="invalid")
    print(result)

    con,cur=database()
    qry="select * from remedies where disease    " \
        "="+result+"""
    cur.execute(qry)
    res=cur.fetchall()
    print(res)

    return render_template("results.html",detailes=res)

@app.route("/fertilizers/<result>")
def fertilizers(result):
    con,cur=database()
    qry="select * from fertilizers where pest_name="+result+"""
    cur.execute(qry)
    res=cur.fetchall()
    print(res)

    return render_template("fertilizers.html",detailes=res)

@app.route("/crop_evaluations")
def crop_evaluations():
    accuracy_ann, precision_ann, recall_ann, fscore_ann = ann_evaluation()

    ann_list.append("ANN")
    ann_list.append(accuracy_ann)
    ann_list.append(precision_ann)
    ann_list.append(recall_ann)
    ann_list.append(fscore_ann)

    accuracy_cnn, precision_cnn, recall_cnn, fscore_cnn = cnn_evaluation()
```

```
cnn_list.append("CNN")
cnn_list.append(accuracy_cnn)
cnn_list.append(precision_cnn)
cnn_list.append(recall_cnn)
cnn_list.append(fscore_cnn)

accuracy_lstm, precision_lstm, recall_lstm, fscore_lstm = lstm_evaluation()

lstm_list.append("LSTM")
lstm_list.append(accuracy_lstm)
lstm_list.append(precision_lstm)
lstm_list.append(recall_lstm)
lstm_list.append(fscore_lstm)
metrics.clear()
metrics.append(ann_list)
metrics.append(cnn_list)
metrics.append(lstm_list)
accuracy_list=[accuracy_ann,accuracy_cnn,accuracy_lstm]

bars = ('ANN', 'CNN', 'LSTM')
y_pos = np.arange(len(bars))
plt.bar(y_pos, accuracy_list, color=['red', 'green', 'blue'])
plt.xticks(y_pos, bars)
plt.xlabel('DL Algorithms')
plt.ylabel('Accuracy')
plt.title('Analysis between DL Accuracies')
plt.savefig('static/accuracy.png')
plt.clf()

precision_list=[precision_ann,precision_cnn,precision_lstm]
bars = ('ANN', 'CNN', 'LSTM')
y_pos = np.arange(len(bars))
plt2.bar(y_pos, precision_list, color=['green', 'brown', 'violet'])
plt2.xticks(y_pos, bars)
plt2.xlabel('DL Algorithms')
plt2.ylabel('Precision')
plt2.title('Analysis between DL Precisions')
plt2.savefig('static/precision.png')
plt2.clf()

recall_list=[recall_ann,recall_cnn,recall_lstm]

bars = ('ANN', 'CNN', 'LSTM')
y_pos = np.arange(len(bars))
plt3.bar(y_pos, recall_list, color=['orange', 'blue', 'gray'])
plt3.xticks(y_pos, bars)
plt3.xlabel('DL Algorithms')
plt3.ylabel('Recall')
plt3.title('Analysis between DL Recall')
```

```
plt3.savefig('static/recall.png')
plt3.clf()

f1score_list=[fscore_ann,fscore_cnn,fscore_lstm]
bars = ('ANN', 'CNN', 'LSTM')
y_pos = np.arange(len(bars))
plt.bar(y_pos, f1score_list, color=['brown', 'green', 'orange'])
plt.xticks(y_pos, bars)
plt.xlabel('DL Algorithms')
plt.ylabel('F1-Score')
plt.title('Analysis between DL F1-Score')
plt4.savefig('static/f1score.png')
plt4.clf()
return render_template("crop_evaluations.html", evaluations=metrics)
```

```
@app.route("/crop_detection")
def crop_detection():
    return render_template("crop_detection.html")
```

```
@app.route("/crop_prediction",methods=["GET", "POST"])
def crop_prediction():
```

```
    temperature = request.form.get("temp")
    humidity = request.form.get("humd")
    ph = request.form.get("ph")
    rainfall = request.form.get("rainfall")
```

```
    X_test=[[float(temperature),float(humidity),float(ph),float(rainfall)]]
    print(X_test)
    model_path = 'lstm_model.h5'
    model = tensorflow.keras.models.load_model(model_path)
    prediction = model.predict(np.array(X_test, ndmin=0))
    #prediction = model.predict(X_test)
    lb = pickle.load(open('label_transform.pkl', 'rb'))
    prediction_result = lb.inverse_transform(prediction)[0]
    print(prediction_result)
```

```
    con, cur = database()
    qry = "select * from crop_fertilizers where crop_name='" + prediction_result + "' "
    cur.execute(qry)
    fertilizer = cur.fetchone()[1]
    print("fertilizer=", fertilizer)
    return render_template("crop_detection.html",result=prediction_result,fertilizer=fertilizer)
```

DATABASE CONNECTION

```
def database():
    con = mysql.connector.connect(host="127.0.0.1", user='root', password="root",
    database="ai_sas")
    cur = con.cursor()
    return con, cur
if __name__ == '__main__':
    app.run(debug=True,port="2024")
```

PREDICTION MODULE

```
import pickle
import os
import sys
import numpy as np
import operator
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
#from tensorflow.keras.utils import img_to_array,load_img
from keras.models import Sequential, load_model
from keras.preprocessing import image
from keras.preprocessing import image as image_utils
from keras.preprocessing import image
from keras import backend as K
import cv2

def img_prediction(test_image):
    K.clear_session()
    data = []
    img_path = test_image
    testing_img = cv2.imread(img_path)
    cv2.imwrite("../AI_SAS/static/detection.jpg", testing_img)
    model_path = 'vgg16_model.h5'
    model = load_model(model_path)
    test_image = load_img(img_path, target_size=(128, 128))
    test_image = img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis=0)
    test_image /= 255
    prediction = model.predict(test_image)
    max_confidence = np.max(prediction)
    print("max_confidence",max_confidence)

    # Check confidence threshold
    if max_confidence < 0.7: # Adjust threshold as needed
        return "Invalid image, please select a valid image."
    else:
        lb = pickle.load(open('label_transform.pkl_vgg16', 'rb'))
        #print("lb",lb)
```



```
prediction= lb.inverse_transform(prediction)[0]  
print("prediction", prediction)
```

```
K.clear_session()  
return prediction
```

```
#img_prediction("apple1.JPG")
```

LSTM.py

```
import pandas as pd  
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score  
import numpy as np  
from tensorflow.keras.layers import Input, Dense, LSTM  
from tensorflow.keras.models import Sequential  
from sklearn.model_selection import train_test_split  
from sklearn.neighbors import KNeighborsClassifier  
from tensorflow.keras.layers import Embedding  
import os  
from sklearn.preprocessing import LabelBinarizer  
import pickle  
from keras.models import load_model  
import tensorflow  
  
def lstm_evaluation():  
    training_df = pd.read_csv("../AI_SAS/cropdata/crop_dataset.csv", sep=',')  
  
    x_train = training_df.iloc[:, :-1]  
    y_train = training_df.iloc[:, -1]  
  
    label_binarizer = LabelBinarizer()  
    labels = label_binarizer.fit_transform(y_train)  
  
    pickle.dump(label_binarizer, open('label_transform.pkl', 'wb'))  
    n_classes = len(label_binarizer.classes_)  
  
    # split the training and test data  
    train_features, test_features, train_targets, test_targets = train_test_split(x_train, labels,  
                                                                                  train_size=0.7, test_size=0.3, random_state=23, stratify=labels)  
  
    if os.path.exists("../AI_SAS/lstm_model.h5"):  
        model_path = 'lstm_model.h5'  
        lstm_model = tensorflow.keras.models.load_model(model_path)  
  
        ytest = np.argmax(test_targets, axis=1)  
  
        y_pred = np.argmax(lstm_model.predict(test_features), axis=1)
```

```
acc = accuracy_score(ytest, y_pred) * 100

precsn = precision_score(ytest, y_pred, average="macro") * 100

recall = recall_score(ytest, y_pred, average="macro") * 100

f1score = f1_score(ytest, y_pred, average="macro") * 100

else:
    model = Sequential()
    model.add(Embedding(5000, 32, input_length=4))
    model.add(LSTM(150))
    model.add(Dense(n_classes, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# print(model.summary())
model.fit(train_features, train_targets, epochs=30, batch_size=4)

model.save("lstm_model.h5")

ytest = np.argmax(test_targets, axis=1)

y_pred = np.argmax(model.predict(test_features), axis=1)

acc = accuracy_score(ytest, y_pred) * 100

precsn = precision_score(ytest, y_pred, average="macro") * 100

recall = recall_score(ytest, y_pred, average="macro") * 100

f1score = f1_score(ytest, y_pred, average="macro") * 100


print(acc)
print(precsn)
print(recall)
print(f1score)

return acc,precsn,recall,f1score
```

VGG16.py

```
import os
import numpy as np
import pickle
import cv2
from keras.applications.vgg16 import VGG16
import keras
from os import listdir
```

```
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array

from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.preprocessing import image
import sys
def build_vgg16():
    try:
        MODEL_PATH = 'vgg16_model.h5'
        EPOCHS = 5
        INIT_LR = 0.0001
        BS = 64
        default_image_size = tuple((128, 128))
        image_size = 0
        width = 128
        height = 128
        depth = 3
        print("[INFO] Loading Training dataset images...")
        DIRECTORY = "..\\leaf_disease\\Dataset"
        CATEGORIES =
['Apple_Black_rot','Peach_Bacterial_spot','Pepper_bell_Bacterial_spot','Potato_Early_blight','Strawb
erry_Leaf_scorch','Tomato_Target_Spot']
        data = []
        clas = []

        for category in CATEGORIES:
            #print(category)
            path = os.path.join(DIRECTORY, category)
            #print(path)
            for img in os.listdir(path)[:1000]:
                #print("img1", img)
                img_path = os.path.join(path, img)
                img = image.load_img(img_path, target_size=(128,128))
                img = img_to_array(img)
                data.append(img)
                clas.append(category)

        label_binarizer = LabelBinarizer()
        image_labels = label_binarizer.fit_transform(clas)
```

```
print("image_labels",image_labels)
pickle.dump(label_binarizer, open('label_transform.pkl_vgg16', 'wb'))
n_classes = len(label_binarizer.classes_)
print(n_classes)
np_image_list = np.array(data, dtype=np.float16) / 225.0
print("np_image_list",np_image_list)

x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.3)

if os.path.exists(MODEL_PATH):
    print(f'[INFO] Model already exists at {MODEL_PATH}. Loading model...')
    classifier = load_model(MODEL_PATH)

else:

    base_model=VGG16(include_top=False,input_shape=(128,128,3))
    base_model.trainable=False
    classifier=keras.models.Sequential()
    classifier.add(base_model)
    classifier.add(Flatten())
    classifier.add(Dense(6,activation='softmax'))
    classifier.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

    print("[INFO] training network...")
    aug = ImageDataGenerator(
        rotation_range=25, width_shift_range=0.1,
        height_shift_range=0.1, shear_range=0.2,
        zoom_range=0.2, horizontal_flip=True,
        fill_mode="nearest")

    history = classifier.fit_generator(
        aug.flow(x_train, y_train, batch_size=BS),
        validation_data=(x_test, y_test),
        steps_per_epoch=len(x_train) // BS,
        epochs=EPOCHS, verbose=1
    )

    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(acc) + 1)
    # Train and validation accuracy
    plt.plot(epochs, acc, 'b', label='Training accuracy')
    plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
    plt.title("Training and Validation accuracy Of VGG16")
    plt.legend()
    plt.savefig('vgg16_accuracy.png')

    plt.figure()
```

```
# Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss Of VGG16')
plt.legend()
plt.savefig("vgg16_loss.png")
plt.show()

print("[INFO] Calculating VGG16 model accuracy")
scores = classifier.evaluate(x_test, y_test)
print("Test Accuracy:", {scores[1]*100})
vgg16_accuracy=scores[1]*100

print(vgg16_accuracy)
print("Training Completed..!")
# save the model to disk
print("[INFO] Saving model...")
classifier.save('vgg16_model.h5')

except Exception as e:
    print("Error=" , e)
    tb = sys.exc_info()[2]
    print(tb.tb_lineno)

build_vgg16()
```

APPENDIX-II
RESEARCH PAPER