Abstract

The integration of Machine learning (ML) into agricultural decision-making systems holds immense potential to address critical challenges such as crop yield optimization, resource management, and pest/disease control. This project presents Agri Help, a unified web-based platform incorporating four distinct 2 ML models and 2 DL models for crop recommendation, fertilizer recommendation, plant disease classification, and insect pest detection. The system processes both tabular data (soil properties, weather conditions) and image inputs (leaf/pest images) to deliver actionable insights for farmers. For crop and fertilizer recommendations the Random Forest classifier achieves 99.6% and 99.94% accuracy respectively, by analyzing soil nutrients, pH, and climatic variables. Disease and pest detection models utilize transfer learning of ResNet50v2 and transfer learning of MobileNetv2, achieving 94.53% and 96.03% respectively. The scalable backend architecture employs APIs and cloud-hosted models, ensuring accessibility in resource-constrained regions. By unifying fragmented agricultural decision tools into a single platform, Agri Help addresses the limitations of isolated systems and empowers farmers with data-driven precision agriculture tools. Experimental validation demonstrates robustness across diverse agro climatic conditions, highlighting its potential to transform modern agriculture.

List of Figures

List of Tables

List of Abbreviations

ABBREVIATION	ILLUSTRATION
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
RF	Random Forest
OTP	One-Time Password
API	Application Programming Interface
UI	User Interface
UX	User Experience
HTML	HyperText Markup Language
CSS	Cascading Sytle Sheets
JS	JavaScript
MLP	Multi-Layer Perceptron
DB	Database
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
PIL	Python Image Library
NLP	Natural Language Processing
IoT	Internet of Things

Contents

Chapter 1

Introduction

1.1 Background

Agriculture supports most economies, but it is challenged by uncertain weather conditions, infestations, and low resource efficiency. ML and DL technologies have brought about novel solutions for challenges in agriculture through data-based decision-making. This project presents an integrated system that uses two ML and two DL models to solve some of the most important problems in agriculture: Crop Recommendation, Fertilizer Recommendation, Plant Disease Classification, and Insect Classification. The Crop Recommendation model helps farmers choose the best crops according to soil characteristics, weather, and past data. The Fertilizer Recommendation model offers customized fertilizer recommendations to maximize nutrient use, minimizing cost and environmental degradation. The Plant Disease Classification and Insect Classification models use image-based DL methods to detect diseases and insects, allowing for early treatment and reducing crop loss. The suggested system is built as a web application with a frontend to handle user interactions, a backend to process the requests, and a database to save user information. The ML models are run on a scalable cloud infrastructure so that they can be accessed easily and reliably. With these models consolidated into a single platform, the system gives a comprehensive solution for farmers with improved productivity, sustainability and profitability. In recent years, DL has appeared as a influential tool in various domains, including agricultural, security, finance, and healthcare. However, the challenge of training deep neural

networks from scratch remains a significant hurdle, primarily due to the extensive computational resources required and the need for large-scale labelled datasets. This is where Transfer Learning plays a crucial role. Transfer learning is a ML approach that utilizes a model previously trained on one task and applies its learned knowledge to a different but related task, developed on a massive dataset, is fine-tuned to perform a different but related task with minimal training data. It allows researchers to leverage existing knowledge from a well-trained model, thereby improving efficiency and accuracy while significantly reducing the need for expensive computational power. For instance, models like ResNet50 and MobileNetV2 are trained on vast image datasets such as ImageNet. Instead of training a new model from scratch for a specific application, such as plant disease detection, medical imaging, or facial recognition, transfer learning allows the reuse of these models by fine-tuning them for specific datasets. This method improves accuracy and reduces the risk of overfitting, accomplishing it an supreme solution in scenarios where labelled data is scarce. Our research focuses on the application of transfer learning for detection of diseases in crops, where we fine-tune pre-trained CNNs to identify plant diseases with high precision. By integrating these models into an AI-driven decision-support system, farmers and agricultural experts can make informed decisions to prevent crop losses. The study explores the comparative performance of multiple CNN architectures and evaluates their effectiveness based on recall, F1-score, accuracy, and precision. The rapid advancements in web technologies have transformed how businesses, organizations, and individuals communicates digital systems. With the increasing demand for user-friendly, scalable, and efficient web applications, modern development frameworks like React.js, Angular, and Vue.js have gained widespread adoption. Simultaneously, backend technologies such as Node.js, Django, and Flask enable seamless communication between web interfaces and databases, ensuring live data processing and enhanced user experiences. The importance of web platforms is particularly evident in domains like e-commerce, education, healthcare, and finance, where digital platforms serve as primary touchpoints for users. With the growing need for personalized services, AI-driven web applications are becoming increasingly prominent, offering recommendation systems, chatbots, real-time analytics, and predictive insights. Our project focuses on developing a robust and scalable web application that integrates capabilities. The application provides real-time predictions based on

user input, intuitive UI/UX design, and seamless database interactions. By leveraging technologies such as React.js for the frontend, Node.js and Express.js and Flask for the backend, and MongoDB for the database, we ensure a responsive and dynamic web application. The project highlights the architectural design, database schema, API integrations, and front-end development strategies used to build a high-performance web application. We also present a comparative analysis of different frameworks, tools, and best practices adopted to optimize speed, security, and scalability.

1.2 Problem Statement

Agriculture faces challenges such as unpredictable weather, soil degradation, inefficient resource use, and pest infestations, leading to reduced crop yield and farmer income. To develop an AI-driven web application, AgriHelp, that integrates ML models for precision agriculture by providing real-time crop recommendations, soil health analysis, and pest detection. The platform will assist farmers in making data-driven decisions to encourage eco-friendly agricultural methods, optimize yield and reduce resource wastage. By leveraging transfer learning-based models and intuitive UI/UX design, the system will ensure accessibility for farmers while delivering accurate insights, thereby enhancing agricultural productivity and efficiency.

1.3 Motivation

- Agriculture faces challenges such as unpredictable weather, soil degradation, inefficient resource use, and pest infestations, leading to reduced crop yield and farmer income.
- Traditional farming methods rely on intuition rather than data-driven decision-making, resulting in inefficiencies and increased resource wastage.
- AI and ML advancements offer the potential to enhance crop monitoring, optimize resource management, and provide precise, real-time recommendations.
- Existing solutions are often fragmented, inaccessible to small-scale farmers, or require

high technical expertise.

- There is a need for a unified, user-friendly platform that integrates ML powered findings to help farmers choose wisely based on insights.
- AgriHelp aims to empower farmers with real-time, AI-based solutions to improve productivity, reduce costs, and promote sustainable farming practices.
- The platform leverages transfer learning, DL, and an intuitive web interface to connect the advanced research and realistic implementation.
- By providing accessible precision agriculture tools, AgriHelp contributes to food security, environmental sustainability, and efficient agricultural resource management.

1.4 Scope and Objectives

1.4.1 Scope

The scope of this project focuses on the development of AgriHelp, an AI-driven web application designed to assist farmers in making informed agricultural decisions. The platform integrates ML models and data analytics to provide actionable insights for precision farming, helping farmers optimize crop selection, predict yields, and manage resources efficiently. AgriHelp focuses to merge the traditional farming techniques and modern, data-driven agricultural techniques by offering a user-friendly web interface accessible to farmers, agronomists, and agricultural researchers. The system utilizes historical and real-time agricultural data to generate recommendations for crop health monitoring, disease prediction, and optimal farming practices. It leverages publicly available agricultural datasets and user-input data to generate AI-powered insights. Future enhancements could include integration with government agricultural databases, weather APIs, and financial advisory tools to further support farmers.

1.4.2 Objectives

- **Crop Recommendation System:** Develop a ML based model that suggests the most suitable crops based on climate conditions, past yield data soil type.
- **Pest and Disease Prediction:** Implement an AI-powered system to predict possible pest infestations and plant diseases based on user-uploaded images and agricultural data.
- **User Friendly Web Interface:** Design an intuitive and accessible web platform that allows farmers to input their data and receive actionable recommendations without requiring advanced technical knowledge.
- **Scalability and Future Integration:** Enable potential expansion of the system by integrating APIs for government schemes, loan assistance, and agronomic support.

1.5 Sustainable Development Goals (SDG) Addressed

Our project aligns with the following Sustainable Development Goals (SDGs):

- **SDG 2: Zero Hunger:** By improving crop yield and disease prevention, AI-powered solutions help ensure food security.
- **SDG 12: Responsible Consumption and Production:** Efficient resource management and precision farming techniques reduce wastage.
- **SDG 13: Climate Action:** Sustainable agricultural practices minimize the negative impact on the environment, reducing carbon footprints.

1.6 Organization of Project Report

The organization of this project report encompasses a structured breakdown of all the essential aspects involved in developing the AgriHelp platform. The report begins with an Introduction, which outlines the need for a unified AI-driven platform for precision agriculture and sustainable

farming, emphasizing the challenges faced by farmers in accessing accurate crop recommendations, disease detection, and market trends. It also highlights the role of artificial intelligence in addressing these issues and the potential impact of the proposed solution on the agricultural sector. The Literature Survey follows, summarizing related research and existing approaches in precision agriculture, AI-based crop recommendation systems, and sustainable farming techniques. The Requirements section outlines the essential functional and non-functional specifications of the system, along with the tools, technologies, databases, and services used to develop and deploy the AgriHelp platform. The Project Plan details the different phases of development and the timeline for execution. The Methodology section describes the approach taken for preprocessing, model selection, data collection, training, and overall system architecture. The Design section presents visual representations of the platform's workflow, including user interactions, AI-based predictions, and recommendation mechanisms. The Implementation section elaborates on the core functionalities of AgriHelp, such as crop recommendation, disease detection, yield prediction, and market analysis, along with its integration into a fully functional web application. The Testing section evaluates the system's accuracy, efficiency, and overall performance based on real-world agricultural data. The Results section analyses the outcomes of the developed platform, highlighting its effectiveness in providing valuable insights for farmers and stakeholders. Overall, the report gives a detailed overview of the conceptualization, design, implementation, testing, and evaluation of the AgriHelp platform as an AI-driven solution for modern agricultural challenges.

Chapter 2

Literature Survey

This research [1] examined a DL technique for the automated detection of plant diseases and pest outbreaks in farm settings. A CNN, particularly the Inception-ResNet-v2 model, was trained on a dataset of more than 47,000 images that illustrated 27 diseases across ten types of crops. The technology showed potential for effective diagnosis with an identification rate of 86.1%. Additionally, a refined and proven mobile app could identify farming issues and provide recommendations. This article offers an overview of farming research projects using image processing methodology, i.e., crop disease and pest diagnosis [2]. It sees that technology used determines the correctness of diagnosis and it advises that combining two or more varieties of data will yield better results. The research points to the possible use of GANs, DL, IoT, and ML in this field. The use of CNN models on mobile platforms may improves access to this technology by users. This [3] research puts forward a DL based autonomous processing and classification idea for leaf data with an emphasis on salient features. It explains how visual attention processes may be employed to modify computer algorithms for classification with consideration for the variety present in natural objects such as plant varieties. The research also tackles the issue of insufficiency of data by employing data augmentation methods to extend the quantity of information to use for image classification tasks. The proposed technique of detection and classification of pests was made more effective through experiments conducted, and the outcomes were promising.

Reviewing the current body of knowledge regarding the impacts of climate change on plants,

this research [4] sets out important research questions and potential avenues for this critical field of research. It also points out the significance of early diagnosis by showing how vulnerable plants are too many pests and diseases that can severely cut down their production and quality. Many studies that utilize DL and ML methods to detect pests and plant diseases in various crops, including bananas, rice, mangoes, tomatoes, and grapes, are tabulated in the literature review section. The studies reviewed as a whole show how these technologies can be used to enhance crop management and yield. As per this paper, research on crop-harming pest and disease confirmation with spectral brightness coefficients and formal representation analysis was conducted between 2021 and 2023 [5]. The research examines spectral data of satellite images and classifies the presence of farm problems based on ML algorithms such as LR, XGBoost, and CNN. The overall objective is to detect diseases and pests with accuracy and in a timely manner in order to boost agricultural production. The findings would be applied in creating new methods of farming that enhance yields while reducing control costs. This research examines the benefits of a number of DL models in disease and pest detection on leaves [6]. It explains in depth how these DL models are utilized for this very purpose in agriculture. The review compiles methods of detecting pests along with their performance measures and discusses the efficacy of various models. DL capability to enhance the effectiveness of identification of agricultural problems is emphasized at the conclusion of the paper.

In this research, a [7] technique for monitoring and forecasting pests and illnesses in agricultural crops using artificial intelligence is proposed. The strategy entails establishing system goals and making use of a broad dataset that includes satellite images, real-time data from IoT devices in the field, and historical outbreak records. The crucial steps include data preprocessing (cleaning, normalization, augmentation) and identifying significant features related to disease and pest outbreaks. Dimensionality reduction techniques like PCA may also be employed. This literature review [8] examines the application of ML and DL methodologies for identifying cotton pests and diseases to enhance crop yield for agronomists. It discusses various research articles that used different data sources, data features, and processing techniques to achieve better cotton production. The review covers techniques like K-means clustering, neural networks, and IoT-based detection systems used for identifying and managing cotton ailments. The paper

also looks for weak fields in present research, such as the lack of integration with mobile applications and the need for high-quality, diverse image datasets. This research investigates plant leaf pests [9] and diseases from an unsupervised learning perspective to address the limitations of existing datasets. To identify and find anomalous regions on plant leaves without the need for labelled data, it presents a DL correlation model that draws inspiration from image restoration. The experimental findings show that this approach produces good anomaly localization and identification at the pixel and image levels.

This research explored the use of artificial intelligence in the form of computer vision to quickly and accurately detect agricultural pests [10]. Various iterations of the YOLO object detection algorithm were explored with both healthy and ill plant datasets. The research indicated that the most recent YOLO versions (v7 and v8) were able to detect more quickly and accurately than earlier approaches. The results show that tremendous potential exists for these cutting-edge AI methods to enhance the detection of pests and plant diseases in agricultural environments. This paper provides a convolutional neural network model [11] for photo-diagnosing plant leaf diseases. The CNN was trained on images of healthy and unhealthy leaves to differentiate between the two and classify diseases according to patterns of deficiency. The article suggests a CNN-based model for agricultural enhancement, although it recognizes the requirement of high processing power and better plant data. As per the report, obtaining profitable outcomes for all parties in agriculture is reliant on diagnosing plant diseases properly. The aim of this study [12] was to apply DL to develop a system for Xinjiang-region crop disease and insect pest detection. The ResNet-50 architecture, which is the main model employed, proved to have an impressive accuracy of 95.2% in recognizing a wide range of agricultural issues. The performance of the system was also proved to be balanced with regard to accuracy, complexity, and speed when contrasted with other DL systems. The identification method developed has great potential for real-time observation and early warning in agriculture, allowing farmers to respond swiftly.

A novel DL model [13] to detect plant leaf diseases and farming pests was developed in this work. This integrated hybrid model, developed based on the EfficientNetB0 architecture, was designed to be deployable in low-resource and sparsely trained environments. The model's early detection of plant diseases and farm pests has great potential for making timely recom-

mendations and solutions. The experimental research on several public datasets illustrated the high precision of the model. This paper [14] utilized spectral data analysis and remote sensing to investigate agricultural challenges better, especially in Kazakhstan's steppe areas. With the application of spectral brightness coefficients from ground and satellite measurements, environmental stressors and vegetation health can be evaluated without causing any damage. The analysis of organized information facilitated the detection of patterns and trends in the prevalence of disease and pests across a broad spectrum of time and space.

To comprehend the processes of disease and pest spread and formulate effective management measures, the research underscores the need to marry general knowledge and abstract concepts in data. By providing AI-based solutions for managing diseases and pests as well as improving farming methods, this research [15] solved the problems that Sri Lankan banana farmers encounter. Based on DL and image processing, a smartphone application named Banana Buddy was developed for accurately diagnosing banana diseases and detecting pseudo stem weevil infestations. By employing data augmentation and careful preprocessing, the system was able to achieve high accuracy levels (99.1% for diseases and 100% for pests). The objective of future studies is to expand the system's ability to identify a wider range of banana plant issues.

This study tested the use of transfer learning in the early diagnosis of banana fruit leaf diseases even though high detection accuracy remains hard to attain [16]. To address this, the research compared some models in a creative modular strategy. The research illustrates the ways in which DL and ML can greatly enhance plant lesion identification's accuracy and performance.

It is hoped that by creating diagnostic systems based on useful techniques, it will be able to minimize the loss of fruit. This article is a concise list of references [17] on pest detection on plants based on image processing and DL. It comprises citations of 2016 and 2020 conference proceedings on this subject. This research aimed to create a [18] method for automatically identifying potato diseases, especially when limited images of new leaf problems are available. The study used transfer learning with EfficientNet and ConvNeXt architectures to overcome the challenges of small datasets and data imbalance, employing data augmentation techniques. The EfficientNet model was found to be superior to the ConvNeXt model and previous research in detecting diseases on potato plant leaves, achieving a high accuracy. The research highlights the

financial support received for this work.

Chapter 3

Requirements

3.1 Functional Requirements

3.1.1 User Registration and Authentication

- New users have the option to watch a demo video on how to use the website.
- Users can sign up with personal details including name, age, email, phone number, and location.
- Existing users can log in securely using password authentication and email.

3.1.2 Farmer and Farm Information Collection

- The platform collects detailed farmer information, including:
  - Name
  - Phone Number
  - Location
- The platform gathers farm-specific information, such as:
  - Total land area in acres

  - Farming Type (Traditional, Organic, Greenhouse, Others, Mixed)
  - Current Crops (Corn, Soybean, Cotton, Tomatoes, Carrot, Wheat, Rice, Potatoes, Others)
  - Planned Crops (Corn, Soybean, Cotton, Tomatoes, Carrot, Wheat, Rice, Potatoes, Others)
  - Growing Season Used (Spring, Summer, Fall, Winter, Year Around)

- The platform gathers Challenges and Goals, such as:
  - Challenges: Pest management, soil health, yield optimization, market access, disease control, water management, etc.
  - Goals: Increase yield, reduce cost, improve soil health, better pest management, access market intelligence, sustainable farming practices, water conservation, etc.

- The platform gathers Current Technology Usage, such as:
  - Irrigation Systems
  - Soil Testing
  - Weather Monitoring
  - Farm Management Software
  - None

3.1.3 AI-Driven Crop Recommendation System

- The platform suggests optimal crops based on:
  - Fertilizer recommendation based on soil chemical properties and crop type (rainfall, nitrogen, phosphorus, potassium, pH level, soil color, temperature)
  - Historical farming data
  - Crop recommendation based on soil chemical properties (rainfall, nitrogen, phosphorus, potassium, pH level, soil color, temperature)
  - Sustainability factors, promoting eco-friendly and high-yield crops

3.1.4 Plant Disease and Pest Identification System

- Farmers can upload images (up to 10 MB) of diseased plants or pests.
- The platform uses Transfer Learning-based deep learning models to classify:
  - Type of disease affecting the plant
  - Type of pest detected

3.1.5 Prediction History

- Save all user predictions (crop, fertilizer, disease, pest).
- Retrieve and display past predictions.

3.1.6 Contact and Feedback

- OTP sent via SMS or email verifications.
- Store messages in the database with timestamp.

3.2 Non-Functional Requirements

- **Performance:** The platform should process farmer data and generate crop recommendations with minimal latency, ensuring near real-time responses. The disease identification model must achieve high accuracy (above 90%) for precise classification.
- **Scalability:** The system should support multiple farmers concurrently and be capable of integrating additional features such as livestock management and climate prediction.
- **Usability:** The platform should have an intuitive and user-friendly UI, providing clear instructions and tooltips. It must support regional languages to improve accessibility for farmers. The website should be mobile-responsive for seamless access via smartphones.

- **Reliability and Availability:** The system must be available 24/7 with minimal downtime. AI models should be updated periodically to enhance accuracy based on new agricultural data.
- **Maintainability:** The system should be designed with a modular architecture to facilitate easy updates and improvements. Regular bug fixes and enhancements should be implemented to ensure smooth functioning.
- **Security:** All farmer data, including personal and farm-related details, must be securely stored using encryption and password hashing. Authentication mechanisms such as OAuth and two-factor authentication should be implemented to prevent unauthorized access. Images uploaded for disease or pest classification should be stored securely in cloud storage.

3.3 Tools and Technologies Used

Table 3.1: Tools and Technologies Used

Category	**Tools / Technologies / Frameworks**
Frontend Development	React.js, HTML, Tailwind CSS, JavaScript, Vite (build optimization), FontAwesome for icons
Backend Development	Flask, NodeJS, ExpressJS, Mongoose ODM for schema modeling, Multer for file handling
Communications and Notifications	Nodemailer for email communications, Twilio for SMS notifications
Machine Learning Frameworks	TensorFlow, PyTorch, Scikit-learn, NumPy, Pandas
Deep Learning for Transfer Learning	ResNet50, MobileNet V2
Cloud Services and Deployment	Render.com for web service hosting, Cloudinary for image storage, MongoDB Atlas for database hosting, GitHub for version control repository
APIs Testing	Postman
Security	HTTPS for secure data transmission, JWT Authentication
Development Tools	Git for version control, Nodemon for development server, Postman for API testing, Kaggle notebooks for model development
Security and Data Management	Bcrypt for password hashing, CORS for cross-origin requests
Data Visualization	Matplotlib, Seaborn

Chapter 4

Project Plan

4.1 Project Phases and Timeline

This proposed timeline outlines a 15-week research project for developing an intelligent prediction and recommendation systems to help the farmers. Each week focuses on a specific task to ensure a well-structured and efficient workflow.

Week 1: Topic Finalization

- Identify major agricultural challenges (crop disease, soil health, market trends).
- Define core objectives, including crop health monitoring and yield prediction.
- Determine end-users (farmers, agricultural experts, market analysts).

Week 2: Literature Survey

- Study recent research papers on crop disease detection, yield forecasting, and smart farming solutions.
- Analyze existing DL and ML methods applied in agriculture.
- Recognize performance metrics.

Week 3: Study of Existing Methods

- Review CNN, YOLO, and other AI-based models for crop disease detection.
- Analyze traditional farming methods and limitations in real-time monitoring.
- Compare different ML models for yield prediction and market trend forecasting.

Week 4: Review Paper Drafting

- Draft initial review paper summarizing research gaps and solutions.
- Structure sections into introduction, literature review, comparative analysis, and conclusion.

Week 5: Synopsis and Drafting

- Create a detailed synopsis including project objectives, methodology, and expected outcomes.
- Outline the implementation plan, milestones, and evaluation strategies.
- Submit synopsis for internal guide review and approval.

Week 6: Review Paper Finalization and Publication

- Finalize review paper with edits and improvements based on feedback.
- Choose a suitable conference for submission.

Week 7–9: Data Collection and Preprocessing

- Collect and curate image datasets for crop diseases and soil health.
- Gather historical crop yield and market data.
- Perform data cleaning, augmentation, and annotation for improved model robustness.

Week 10–11: Model Training and Fine-Tuning

- Train DL models for crop disease detection.
- Fine-tune DL models for yield prediction and market analysis.
- Check model's response using acuracy, confusion-matrix, and precision-recall curves.

Week 12: Model Testing and Validation

- Validate models on real-world test datasets.
- Assess real-time performance using field images and market price trends.

Week 13–14: Web App Implementation

- Integrated ML models with MERN stack frontend and backend.
- Developed UI for predictions, input forms, and result display using React.
- Set up Express.js backend to handle API routes and connect with Flask model APIs.
- Implemented JWT-based authentication and MongoDB integration for storing user data.
- Deployed frontend (Vercel) and backend (Render), and tested end-to-end functionality.

Week 15: Research Paper Finalization and Publication

- Document complete methodology, results, architecture, and findings.
- Structure main research paper for final submission including graphs and metrics.
- Submit to target journal/conference.

Chapter 5

Methodology

5.1 Data Collection

5.1.1 Crop and Fertilizer Recommendations Dataset

We began with the "Crop and Fertilizer Dataset for Maharashtra" which contains 5,073 records. Each record captures key agronomic parameters such as soil color, nitrogen, phosphorus, potassium, pH, rainfall, and temperature along with the corresponding crop and its recommended fertilizer. The dataset spans a diverse range of crops (e.g., Sugarcane, Jowar, Cotton, Rice, Wheat, Groundnut, Maize, Tur, Urad, Moong, Gram, Masoor, Soybean, Ginger, Turmeric, Grapes, Tomato, and Potato). For improved analysis, we restructured the data into the "Maharashtra Fertilizer New Data Set" by grouping records by crop and fertilizer, ensuring a balanced representation (for example, Sugarcane: 1,010 records; Wheat: 859; Cotton: 650; and fertilizers such as Urea: 1,294 records, DAP: 594 records, among others). This refined dataset provides a robust foundation for generating region-specific crop and fertilizer recommendations.

5.1.2 Plant Leaf Disease Dataset

To aid plant disease diagnosis, we combined image data from various public sources (such as Kaggle and Mendeley) for crops like Sugarcane, Jowar, Cotton, Rice, Wheat, Groundnut, Maize, Urad, Soybean, Ginger, Turmeric, Grapes, Tomato, and Potato. The resulting "Maharashtra

Plant Leaf Disease Data Set" contains 95,563 images that are carefully classified by crop and disease type; each class has images count in the range of 1700 - 2000. For example, the Wheat category has images of Brown Rust, Yellow Rust, and Healthy leaves (1,700 images each), whereas the Tomato category has six conditions (with 1,700 images each). This collected dataset supports faithful training and testing of deep models for true disease diagnosis.

5.1.3 Maharashtra Pest Dataset

Various Out of the complete IP102-Dataset (initially 75,222 images over 102 classes of pests), we chose 30 pest classes most common in Maharashtra from surveys and local accounts. To address class imbalance, we used image augmentation methods, and thus the "Pest Data Set Agriculture" comprises 30,000 images—each of the 30 classes of pests (e.g., Brevipoalpus lewisi McGregor, Colomerus vitis, Alfalfa seed chalcid, etc.) is represented equally by 1,000 images. This balanced dataset represents the regional pest profile correctly and enables strong model development for pest identification.

5.1.4 Methodology

- **Crop and Fertilizer Recommendation Model**

  The crop suggestion model utilizes a RF Classifier to predict the most suitable crop based on soil nutrients pH, rainfall, temperature, and (N, P, K). The dataset is preprocessed by handling missing values, encoding categorical features using OneHotEncoder, and dividing it into 80% training and 20% testing. The model is trained to provide high-accuracy crop recommendations, helping farmers optimize their yield. The fertilizer suggestion system also utilizes a RF Classifier to provide the best fertilizer recommendation depending on the chosen crop and soil characteristics. It provides accurate fertilizer suggestions, enhancing productivity and soil health. The system offers multiple fertilizer suggestions when available, improving agricultural decision-making in the real world.

- **Plant Leaf Disease Model**

  For leaf disease detection in plants, we took a transfer learning solution with a ResNet50

backbone. In this approach, the strength of pre-trained models is utilized to adapt rapidly to our task even when our dataset is quite small as presented in figure given below Using a model that is pre-trained on ImageNet, we benefit from rich feature representations developed over a big dataset, enhancing our detection accuracy. We employ ResNet50 as our underlying feature extractor, stripping it of its fully connected top layer. The pre-

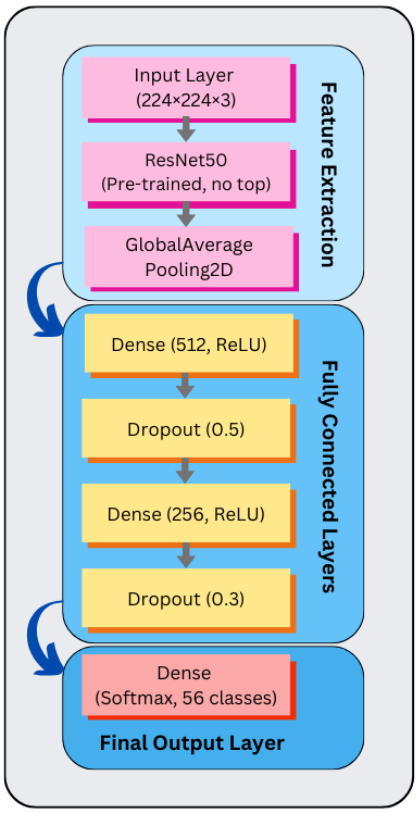

Figure 5.1: Proposed Plant Leaf Disease Classification Model

trained weights are first frozen to retain the general features learned through large-scale training. This freezing process is an important part of transfer learning, enabling the network to be used as a strong feature extractor without the need for a lot of labeled data. Through the use of these pre- trained features, we guarantee that our model will be able to generalize well to other plant diseases. The ResNet50 base output is fed into a Global Average Pooling layer that reduces spatial features to a dense representation. This is then along with a Dropout layer that has a dropout rate of 50% to avoid overfitting and topped

by a thick layer of 512 units using ReLU activation. One more dense layer of 256 units with ReLU activation continues, followed by a Dropout layer with a dropout rate of 30%. Finally, the network ends with a dense softmax layer with the number of neurons being the total number of disease classes (56 classes) to enable multi-class classification. The training is done in two stages: initial training and fine-tuning. During the initial stage, when the base is frozen with ResNet50, we train the custom classification head using the Adam optimizer and the initial learning rate of 1e-3 and categorical cross-entropy loss. The initial phase validates that the model can effectively translate the pre-trained features to our plant disease database. After that, we train the head of classification well before proceeding further with fine-tuning by keeping the top layers of the ResNet50 model non-restricted but freezing the bottom 90%. The model is then recompiled with a lower learning rate (1e-5) to refine the feature extractor's ability to recognize the subtle variations in plant disease symptoms. This controlled fine-tuning further enhances the detection accuracy while preventing overfitting.

- **Pest Classification Model**

  For pest recognition, as shown in the figure 5.2. We also employ a transfer learning strategy—this time utilizing a MobileNetV2 backbone—to efficiently adapt a pre-trained model to our specific task. By leveraging MobileNetV2, which has been pre-trained on ImageNet, we can extract meaningful features from pest images while significantly reducing the need for extensive labelled datasets. As our base feature extractor, we use MobileNetV2 with its top classifier removed. Initially, the model's weights are frozen, allowing it to function as a fixed feature extractor that benefits from transfer learning. This approach enables us to utilize the robust, general image features learned from large-scale datasets, ensuring an effective foundation for pest classification.

  Extracted features go through a Global Average Pooling layer that reduces the spatial dimensions by taking the mean of each of the feature maps. The minimal feature representation passes through a 512 neuron, ReLU activation dense layer followed by a 50% Dropout layer for protection against overfitting. A second dense layer of 256 neurons with ReLU activation and another Dropout at a rate of 30% is added, and then there is a

softmax output layer that has the same number of neurons as the 30 pest classes in our dataset. The training process has two stages: initial training and fine-tuning. In initial training, we build the model with the Adam optimizer having a learning rate of 1e-3 and categorical cross-entropy loss. In this phase, the custom classification head is trained while the MobileNetV2 base is frozen. This allows the new layers to adapt rapidly to the pest dataset while maintaining the pre-trained feature representations. After the classifier head has converged, we now fine-tune by unfreezing a portion of the uppermost layers of MobileNetV2 and leave the lower 75% frozen. The model is then retrained using a significantly reduced learning rate of 1e-5 so that the pre-trained features can accommodate domain-specific pest patterns. This precise tuning further optimizes classification accuracy to produce a very efficient pest identification model that runs well on our test set.

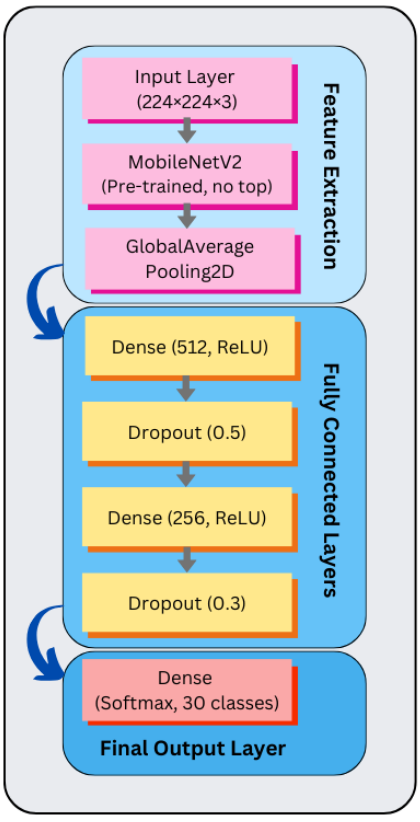

Figure 5.2: Proposed Pest Classification Model

5.2 System Architecture

5.2.1 Frontend and Backend Data flow

The system follows a well-organized, modular architecture ensuring scalability, maintainability, and clarity. The frontend layer, developed using React.js, serves as the primary user interface where farmers can interact with the system. It allows them to input numerical data such as Nitrogen (N), Phosphorous (P), and Potassium (K) levels, along with environmental parameters, and also upload images for analysis. The interface dynamically renders content, ensuring a smooth user experience with form validation to ensure accurate input before submission. It communicates with the backend through API requests, managing responses to display predictions and recommendations. The backend layer consists of a Node.js server that acts as an intermediary between the frontend, machine learning models, and the database. It manages API requests, processes data, interacts with MongoDB for data storage and retrieval, and communicates with the Flask API Gateway. The Flask API Gateway handles requests from the Node.js server and routes them to the appropriate machine learning models, ensuring smooth orchestration. It aggregates responses and sends them back to the Node.js server.

The system employs two ML and two DL models respectively: a Crop Recommendation Model and a Fertilizer Recommendation Model, both using the RF algorithm to analyse soil properties and recommend crops or fertilizers; an Insect Identification Model leveraging ResNet50 for pest detection in uploaded images; and a Plant Disease Classification Model utilizing MobileNetV2 to identify crop diseases from leaf images and suggest treatments. MongoDB serves as the database layer, efficiently handling queries and storing user inputs, historical predictions, and analysis results while ensuring scalability and flexibility. The data flow begins with farmers entering data or uploading images via the React.js interface, which sends API requests to the Node.js server. The server routes these to the Flask API Gateway, which processes them through the relevant machine learning models. Once predictions or recommendations are generated, the Flask Gateway consolidates the responses and sends them back to the Node.js server, which stores the results in MongoDB and forwards them to the frontend. Finally, the farmers receive their required analyses, predictions, or recommendations on their interface, ensuring a

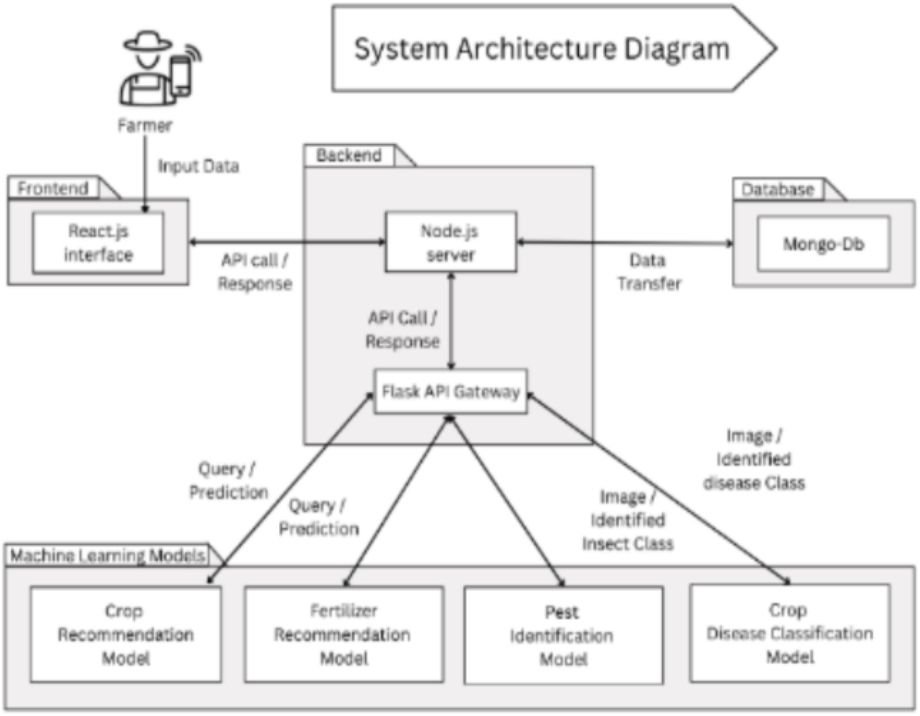

Figure 5.3: System Architecture Diagram

seamless and effective user experience.

5.3 Algorithms Used

5.3.1 Random Forest

RF is a powerful ensemble-based algorithm utilized for both classification and regression tasks. It works by creating a large number of decision trees during training and combining their results to make a final prediction. Each model is trained on a random sample from the dataset using a technique called bootstrap sampling, and at each node split, it considers a random subset of features. This randomness ensures that the individual trees are diverse, reducing the risk of overfitting and improving the overall model performance. During prediction, the RF aggregates the outputs of all trees—either by taking the majority vote (in classification) or the average prediction (in regression). Because it uses multiple trees and randomization, RF is highly robust to noise, can handle missing values, and performs well on large datasets with high dimension-

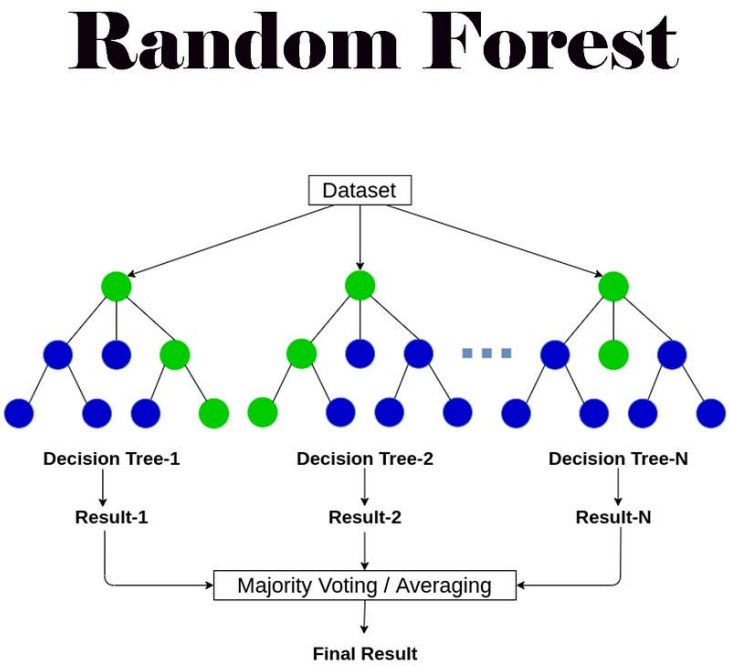

Figure 5.4: Random Forest Algorithm

ality. It also provides useful insights like feature importance, helping users understand which factors are most influential in making decisions. Another strength of the RF algorithm is its ability to handle non-linear relationships and interactions between features. Unlike linear models that assume a straight-line relationship between input and output, RF can model complex patterns because each decision tree splits the data based on different combinations of features and thresholds. As a result, it performs well across a wide range of datasets and domains, including agriculture, finance, healthcare, and more. Additionally, RF is known for its robustness against overfitting, especially when compared to single decision trees. This is due to the averaging of predictions across many trees, which smooths out noise and reduces the variance. The model also supports techniques such as feature importance ranking, which can help researchers identify the most influential variables in their prediction task. For example, in your agri-help webapp, it can reveal which soil nutrient (N, P, K) or environmental parameter has the most impact on crop or fertilizer recommendations.

5.3.2 MobileNetV2 (Deep Learning – Lightweight CNN)

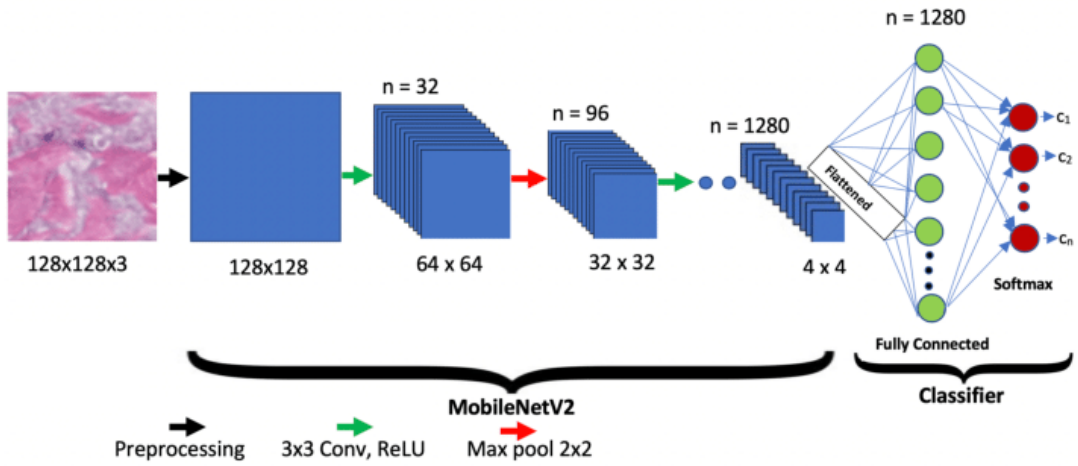

Figure 5.5: MobileNetV2 Algorithm

MobileNetV2 is a lightweight and efficient deep learning model designed for mobile and embedded vision applications. It builds upon the success of the original MobileNet by introducing two key innovations: depthwise separable convolutions and inverted residuals with linear bottlenecks. Depthwise separable convolutions split the standard convolution into two separate layers—depthwise (for spatial filtering) and pointwise (for combining channels)—greatly reducing the number of parameters and computational cost without compromising much on accuracy. The inverted residual blocks further improve efficiency by using a narrow input and output layer with an expanded intermediate layer. This structure helps in retaining important features while reducing memory usage. MobileNetV2 is especially effective in real-time applications such as image classification, object detection, or in your case, plant disease identification, where speed and resource efficiency are essential. Its compact size and high accuracy make it ideal for running directly on devices like smartphones or edge systems in agriculture support tools. Another important aspect of MobileNetV2 is its use of linear bottlenecks in the residual blocks, which help preserve feature representations during dimensionality reduction. Unlike traditional deep networks that apply non-linear activation functions like ReLU at every stage, MobileNetV2 uses a linear layer at the end of each residual block. This avoids loss of information that can occur due to non-linear transformations, especially when reducing feature dimensions. This design makes

it particularly effective for fine-grained visual tasks like detecting subtle patterns in plant leaves for disease classification. Moreover, MobileNetV2 is highly suitable for deployment on low-power devices such as smartphones, Raspberry Pi, or IoT-based field sensors. Its lightweight architecture ensures fast inference times, minimal latency, and lower memory consumption, all of which are crucial for real-time agricultural applications. In your agri-help webapp, using MobileNetV2 allows farmers to upload crop images and get instant disease diagnosis, even in areas with limited connectivity or processing power, making AI assistance more accessible and practical in rural settings.

5.3.3 ResNet-50 (Deep Learning – CNN with Residual Connections)

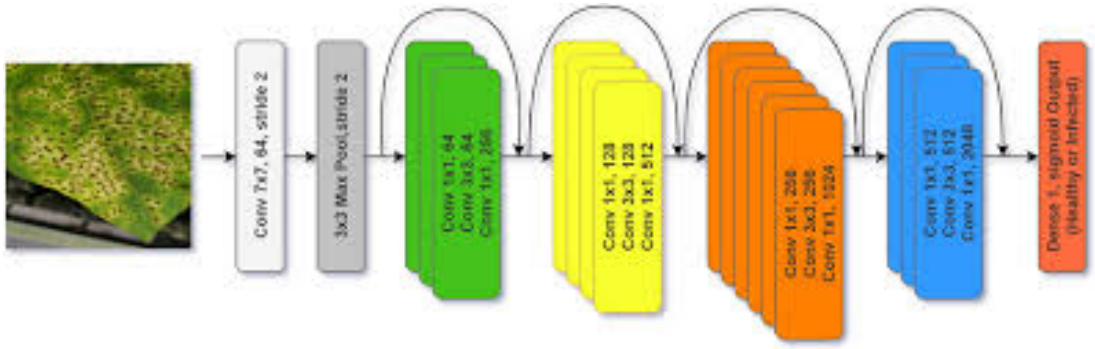

Figure 5.6: ResNet-50 Algorithm

ResNet50 is a deep convolutional neural network with 50 layers, designed to solve the problem of vanishing gradients in very deep networks. It introduces the concept of residual learning, where shortcut connections (also called skip connections) allow the model to learn the difference (residual) between the input and the output of a few stacked layers. This helps the network train deeper architectures without degradation in performance, as it enables gradients to flow directly through the network during backpropagation. The architecture of ResNet50 includes multiple residual blocks, each consisting of convolutional layers and a shortcut that bypasses one or more layers. This design allows the model to focus on learning residual mappings rather than trying to fit the entire transformation from input to output. In practical applications like insect identification or object recognition in agricultural systems, ResNet50 performs exceptionally well, offering high accuracy while being optimized enough to be integrated into real-world deep learn-

ing pipelines. Another key innovation in ResNet50 is the concept of identity shortcuts, which are used to skip one or more layers in the neural network. These shortcuts enable the flow of gradients directly through the network during backpropagation, effectively tackling the vanishing gradient problem seen in deep networks. This allows the model to be much deeper (50 layers in ResNet50) while still being trainable, enabling it to learn highly abstract and complex visual features from images. This makes ResNet50 an ideal choice for tasks such as insect identification in your agri-help webapp, where distinguishing between visually similar pests is critical. Additionally, ResNet50 benefits from transfer learning, allowing you to use pre-trained weights from large datasets like ImageNet and fine-tune them on your custom agricultural dataset. This significantly reduces training time and increases accuracy, even with relatively smaller datasets. In your project, applying ResNet50 with transfer learning ensures high-performance image classification for pest detection, empowering farmers with accurate and timely information to prevent crop damage and improve yield outcomes.

Table 5.1: Comparison Between Different Algorithms

Algorithm	**Domain**	**Pros**	**Ideal For**
Random Forest	Traditional ML	Interpretable, robust, flexible	Tabular data
MobileNetV2	Lightweight DL (CNN)	Fast, mobile-friendly	Real-time mobile vision
ResNet-50	Deep DL (CNN)	Powerful, accurate	Complex image tasks

Chapter 6

Design

6.1 Use Case Diagram

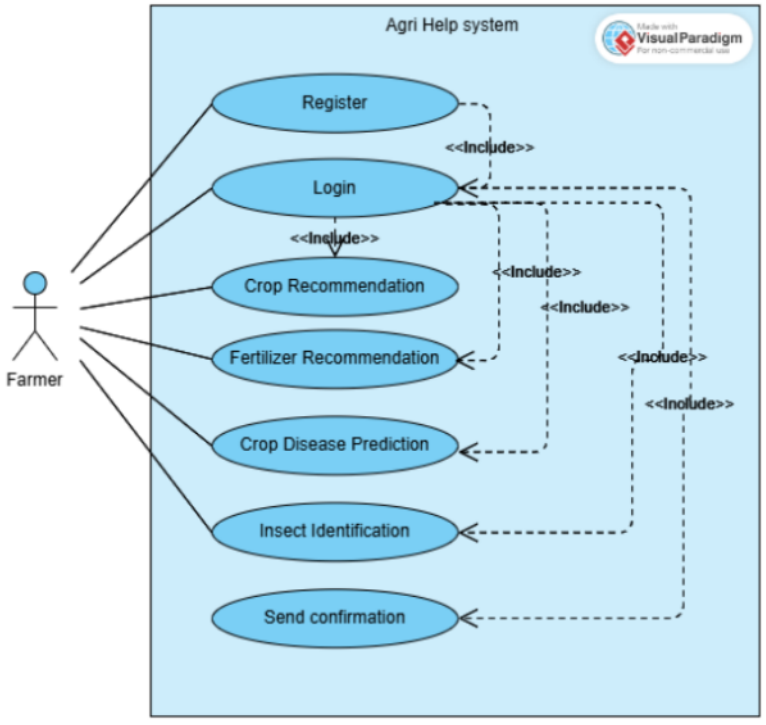

Figure 6.1: Use Case Diagram

The given use case diagram represents the “Agri Help System,” where a farmer interacts with

various features. The primary actions include registering and logging into the system, which are prerequisites for accessing core functionalities. The farmer can request crop recommendations, fertilizer recommendations, crop disease prediction, and insect identification. Each of these processes relies on shared system components, as indicated by the <<include>> relationships, ensuring that functionalities such as login authentication and data validation are consistently applied. The system also includes a "Send Confirmation" use case, likely to notify the farmer of the results or updates. This structured design ensures a seamless user experience while maintaining system efficiency. This diagram is a Use Case Diagram for the "Agri Help System" which visually outlines the interaction between the primary actor—the farmer—and the various services offered by the system.

The farmer initiates several actions, such as registration, login, and requests for agricultural support like crop recommendations, fertilizer suggestions, insect identification, and crop disease prediction. Each of these services is modeled as a "use case" in the system, shown by the blue ovals. The Register and Login use cases serve as the foundational authentication features. Before accessing any of the core functionalities like crop or fertilizer recommendations, the farmer must log into the system. This is depicted by the <<include>> relationships extending from each major use case (e.g., Crop Recommendation, Fertilizer Recommendation) to the Login use case. The inclusion relationship means that logging in is a required precondition for these services—it is not optional but embedded as part of their flow. The core services of the Agri Help system—Crop Recommendation, Fertilizer Recommendation, Crop Disease Prediction, and Insect Identification—represent the intelligent decision-support functionalities that the system offers. These use cases are directly accessed by the farmer, assuming the login process is successfully completed. Once a prediction or recommendation is made, the system includes the "Send Confirmation" use case, which indicates that after any main service is used, a confirmation or notification is sent to the farmer, possibly to acknowledge receipt of the request or delivery of the results. Overall, this use case diagram encapsulates a structured user experience, ensuring that every major feature is accessed securely and ends with proper confirmation. It provides a clear modular view of how a farmer interacts with the system, showcasing the dependencies among services and emphasizing the necessity of user authentication. This structure ensures the

system is both user-friendly and secure while offering a wide range of agricultural assistance features. Additionally, the use of multiple <<include>> relationships in this diagram emphasizes the system's modular and reusable design. For instance, instead of repeating the login and confirmation logic in each individual use case, these functionalities are centralized and reused across all core services. This not only simplifies system maintenance and reduces redundancy in implementation but also ensures consistent user experience across different functionalities. Such a design choice highlights good software engineering practices, where shared operations like authentication and communication are abstracted and reused, promoting scalability and efficiency in the overall Agri Help system. This design also ensures that any updates to shared functionalities like login or confirmation only need to be made once, streamlining future development. It enhances security, as all key services are gated behind a mandatory authentication step.

6.2 Activity Diagram

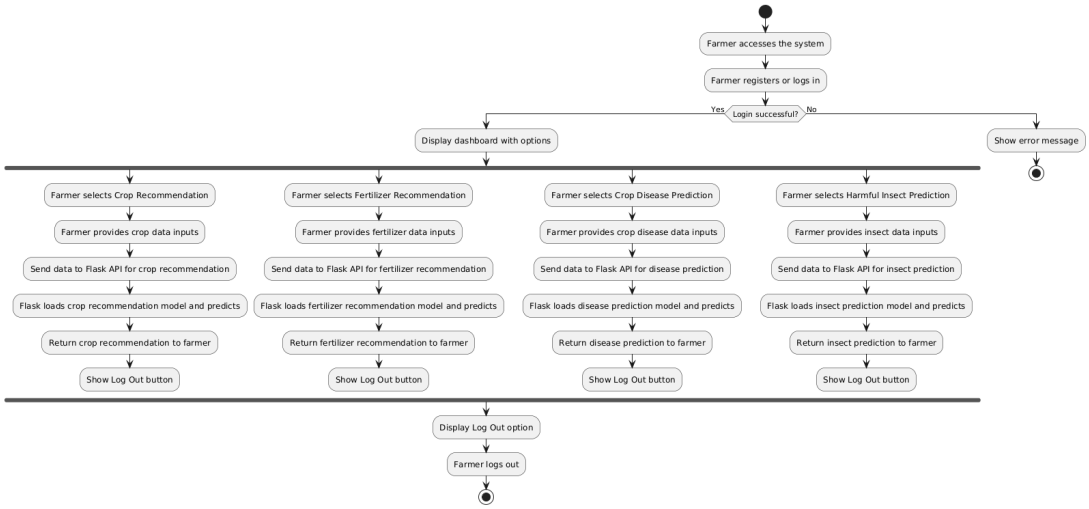

Figure 6.2: Activity Diagram

The activity diagram shows the workflow of an Agri Help System, where a farmer logs in, selects a service (crop recommendation, fertilizer suggestion, disease prediction, or insect identification), and receives the results. The system processes the request and confirms the

response. The flow follows a sequential path, ensuring authentication before accessing any feature. The activity diagram describes the flow of actions taken by a farmer while using the Agri Help system. The process starts with the farmer accessing the system, followed by either registering or logging in. If the login is successful, a dashboard is displayed showing options like Crop Recommendation, Fertilizer Recommendation, Crop Disease Prediction, and Insect Prediction. The farmer can choose any service based on their needs. Once a service is selected, the farmer inputs the required data, which is sent to a Flask API. This API loads the appropriate machine learning model and processes the input to generate a result. The system then returns the recommendation or prediction to the farmer and provides an option to log out. This structured workflow ensures a smooth and interactive user experience, allowing farmers to gain valuable insights based on their agricultural inputs. In case of login failure, an error message is displayed.

6.3 Deployment Diagram

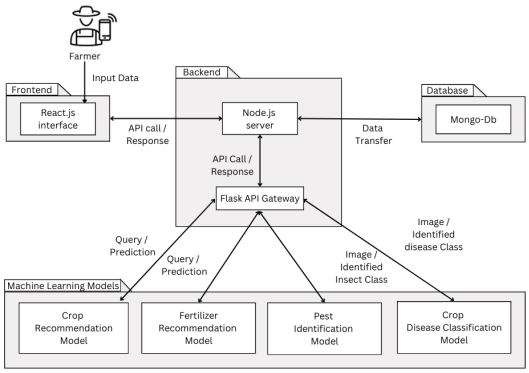

Figure 6.3: Deployment Diagram

This deployment diagram represents the architecture of an intelligent agricultural system designed to assist farmers by leveraging various machine learning models. The process starts at the frontend, where the React.js interface serves as the user interaction point for farmers. Farmers

input data, such as environmental conditions or images of diseased crops, into the system. This input is sent to the backend through API calls. The backend is managed by a Node.js server, which handles the API call/response cycle and orchestrates data flow between the frontend, the Flask API Gateway, and the database. The Node.js server sends the farmer's data to a Flask API Gateway, which acts as a bridge between the server and the different machine learning models. Each model is specialized to perform specific tasks based on the input data, and the Flask gateway routes the data to the appropriate model for processing. There are two ML and two DL models integrated into this system: the Crop Recommendation Model, Fertilizer Recommendation Model, Pest Identification Model, and Crop Disease Classification Model. Based on the type of data received, the respective model is invoked. For example, if the input data is an image of an insect or plant, it is sent to the Pest or Disease Classification model to identify the class. If the data involves soil parameters or environmental metrics, it is sent to the Crop or Fertilizer Recommendation model. Each model processes the data and returns the prediction or recommendation through the Flask API Gateway, which then sends it back to the Node.js server and ultimately to the frontend for display to the farmer. The system also includes a MongoDB database that stores all essential information, such as user data, historical predictions, and possibly model outputs. Data is transferred between the backend and the database, ensuring that the system maintains a record of user interactions and model results for future reference and analysis. Overall, this architecture ensures a seamless interaction between the user interface, backend services, machine learning intelligence, and persistent data storage—creating a comprehensive, intelligent agricultural advisory platform.

Chapter 7

Implementation

7.1 Module Wise Implementation

A) Machine Learning Modules

7.1.1 Crop Recommendation Model

In this project, a Crop Recommendation Model was developed using the **Crop Recommendation Dataset**, which includes essential agricultural features such as Nitrogen (N), Phosphorus (P), Potassium (K), temperature, pH, humidity, and rainfall. The model was trained using the **Random Forest** algorithm due to its robustness and high accuracy in classification tasks. The system outputs the **best-suited crop** for the given conditions along with a **confidence score**, indicating the reliability of the prediction. The trained model was successfully deployed as a **Flask API**, enabling easy integration with web and mobile applications for real-time crop recommendation services.

7.1.2 Fertilizer Recommendation Model

A **Fertilizer Recommendation Model** was developed using a dataset that incorporates **crop and soil data**, enabling precise nutrient management. The model was trained using the **Random Forest** algorithm, which provided accurate and interpretable results. The system outputs

the **most suitable fertilizer recommendation** based on the input conditions. This trained model was **exposed through a Flask API**, facilitating seamless integration into agricultural applications for real-time fertilizer guidance.

7.1.3 Plant Disease Classification Model

A **Plant Disease Prediction Model** was developed using a dataset comprising **leaf images of infected and healthy plants**. The model leverages deep learning techniques, specifically the **ResNet50 architecture**, to effectively classify plant diseases. It outputs the **predicted plant disease** along with a **confidence level** indicating the certainty of the prediction. Leaf images are **uploaded via a web interface** and sent to a **Flask API** for real-time analysis and diagnosis.

7.1.4 Pest Identification Model

A **Pest Identification Model** was developed using the **Dangerous Insects Dataset**, which contains images of various harmful pests. The model was built using the **MobileNetV2** architecture, a lightweight convolutional neural network (CNN) suitable for efficient image classification tasks. It predicts the **type of pest** present in the image along with a **confidence score**. The model is **deployed as a Flask service**, enabling real-time pest detection through web or mobile platforms.

B) Web Application Modules

7.1.5 User Application Module

The **User Management Module** facilitates **user registration** by collecting personal and farm-related details to personalize the system's agricultural recommendations. It incorporates **OTP verification** for secure user validation and supports **secure login** mechanisms. The authentication process is handled using **JWT (JSON Web Tokens)**, ensuring safe and stateless user sessions across web and mobile platforms.

7.1.6 Prediction Modules

The **Prediction Modules** are designed to provide intelligent agricultural insights by interacting with backend machine learning services. The **Crop and Fertilizer** modules accept user inputs such as soil parameters and crop type, which are then sent to their respective **Flask APIs** for real-time prediction. The **Disease and Pest** modules handle **image uploads via Cloudinary**, forwarding them to the model APIs for accurate classification. All modules return results that are **displayed along with prediction confidence**, offering users clear and actionable outputs.

7.1.7 Prediction History Module

The **Prediction History Module** is responsible for maintaining a record of **all types of user predictions**, including crop, fertilizer, disease, and pest diagnoses. It allows users to **view and analyze past results**, helping them make informed agricultural decisions based on historical insights and trends.

7.1.8 Contact and Feedback Module

The **Contact and Feedback Module** enables users to **submit queries, suggestions, or feedback** through a simple interface. All user messages are **stored in MongoDB** along with **timestamps**, allowing the system administrators to track and respond to user concerns efficiently.

7.2 Code Snippets Explanation (Key algorithms and their logic)

7.2.1 MobileNetV2 – For Insect Identification

MobileNetV2 is a lightweight convolutional neural network designed for mobile and edge-device deployment. In the Agri Help WebApp, it is used to identify harmful insects from uploaded images, enabling farmers to take preventive measures early.

Code Snippets:

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input,
    decode_predictions
import numpy as np

# Load pre-trained MobileNetV2 model
model = MobileNetV2(weights='imagenet')

# Load and preprocess the input image
img = image.load_img('insect.jpg', target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# Predict the insect type
preds = model.predict(x)
print("Prediction:", decode_predictions(preds, top=1)[0])
```

Listing 7.1: Insect Identification using MobileNetV2

7.2.2 ResNet50 – For Crop Disease Prediction

ResNet50 is a deep convolutional neural network that excels in image classification tasks due to its residual connections which prevent vanishing gradients. In the Agri Help WebApp, ResNet50 is used to identify crop diseases from uploaded leaf images, enabling farmers to receive timely treatment advice.

Code Snippets:

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input,
    decode_predictions
import numpy as np

```

```
# Load the pre-trained ResNet50 model
model = ResNet50(weights='imagenet')

# Load and preprocess the leaf image
img = image.load_img('leaf_disease.jpg', target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# Make prediction
preds = model.predict(x)
print("Prediction:", decode_predictions(preds, top=1)[0])
```

Listing 7.2: Leaf Disease Detection using ResNet50

The leaf image is resized and normalized, then passed into the ResNet50 model. The model returns the most probable disease classification, assisting farmers in identifying the issue and applying the correct treatment at an early stage.

7.2.3 Random Forest – For Crop Fertilizer Recommendation

The RandomForestClassifier is an ensemble learning algorithm that combines the predictions of multiple decision trees to improve accuracy and reduce overfitting. In the Agri Help WebApp, this model is used to recommend the best-suited crop based on real-time farmer inputs such as temperature, humidity, pH level, and rainfall.

Code Snippets:

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd

# Sample training
df = pd.read_csv('crop_data.csv')
X = df.drop('label', axis=1)
y = df['label']

# Initialize and train RandomForest model
```

```
model = RandomForestClassifier(n_estimators=100)
model.fit(X, y)

# Predicting the best crop for given input data
input_data = [[25, 80, 6.5, 200]]  # temp, humidity, pH, rainfall
prediction = model.predict(input_data)
print("Recommended Crop:", prediction[0])
```

Listing 7.3: Crop Recommendation using RandomForestClassifier

The model learns patterns from historical crop data and predicts the best-suited crop or fertilizer based on real-time farmer inputs. This enhances decision-making and optimizes yield by suggesting the most appropriate crop for the given environmental conditions.

7.3 Database Implementation

7.3.1 MongoDB Schema Implementation

The project utilizes MongoDB with Mongoose ODM for schema modelling. The primary collections are:

1. **Users Collection**
    - Stores user profiles with farm-specific information.
    - Includes embedded documents for farm details, crop selections, challenges, and goals.
    - Indexed on email for efficient lookup during authentication.
2. **OTP Collection**
    - Temporary storage for verification codes.
    - TTL (Time-To-Live) index ensures automatic deletion after expiration.
    - Indexed on both email and mobile for efficient verification.

3. **Prediction Collections**

    - Separate collections for each prediction type:
        - DiseasePrediction
        - PestPrediction
        - CropRecommendation
        - FertilizerRecommendation
    - Referenced to users via userId.
    - Includes all input parameters for context.

4. **Contact Form Collection**

    - Stores user inquiries and feedback.
    - Simple structure with name, email, message, and timestamp.

7.3.2 Schema Relationships

All prediction collections reference the **Users** collection through the `user` field, establishing a one-to-many relationship between users and their predictions. This approach allows for:

- Efficient retrieval of all predictions for a specific user.
- Independent scaling of prediction collections.
- Type-specific queries without traversing unrelated data.

The system leverages MongoDB's document references and aggregation framework to establish and query these relationships efficiently.

7.3.3 Indexing Strategy

The database implementation includes strategic indexes:

- **User email**: Unique index for authentication.

- **OTP fields**: Indexed on email and mobile for verification efficiency.
- **Prediction createdAt fields**: Indexed for timestamp-based queries.
- **User references in predictions**: Indexed for efficient joining and user-specific lookups.

7.4 Schema Design

AgriHelp uses MongoDB as a NoSQL database to store user data, predictions, and system interactions in a structured and scalable format. The design ensures efficient querying and integration with the ML models and web application. The database includes the following main collections:

7.4.1 Users Collection

The **Users Collection** is designed to **store user registration data**, including personal and farm-related information. It captures essential user inputs that are later used to personalize agricultural predictions and recommendations.

Schema Fields:

- **basicInfo:** *fullName*, *email*, *contactNumber*, *location*
- **farmDetails:** *farmSize*, *farmSizeUnit*, *farmingType*, *crops*, *season*
- *cropSelections*, *challenges*, *goals*, *technology*, *createdAt*

Example Document:

```
{
  "basicInfo": {
    "fullName": "Vaibhav Datta Ambhore",
    "email": "vaibhav.ambhore21@pccoepune.org",
    "location": "Pune"
  },
  "farmDetails": {
```

```
    "farmSize": -1,
    "farmingType": "Traditional"
  },
  "currentCrops": ["Soybeans"],
  "plannedCrops": ["Corn", "Cotton"]
}
```

7.4.2 Crop Recommendations Collection

The **Crop Recommendations Collection** stores the results generated by the crop recommendation machine learning model. Each entry links a user to a predicted crop based on various environmental and soil parameters.

Schema Fields:

- **user:** Reference to the user
- **recommendation:** Recommended crop
- **otherParameters:** Includes *N*, *P*, *K*, *temperature*, *pH*, *rainfall*, etc.
- *confidence*, *createdAt*

Example Document:

```
{
  "user": "UserObjectId",
  "recommendation": "Wheat",
  "otherParameters": {
    "temperature": 25,
    "pH": 6.5,
    "rainfall": 100,
    "nitrogen": 80,
    "phosphorus": 50,
    "potassium": 40,
```

```
    "confidence": 0.59
  }
}
```


7.4.3 Fertilizer Recommendations Collection

The **Fertilizer Recommendations Collection** stores the predictions generated by the fertilizer recommendation model. Each entry associates a user with a specific fertilizer suggestion based on soil and crop characteristics.

Schema Fields:

- **user:** Reference to the user
- **recommendation:** Fertilizer name or type
- **otherParameters:** Includes *cropType*, *soilColor*, *NPK*, *pH*, *rainfall*, *confidence*

Example Document:


```
{
  "user": "UserObjectId",
  "recommendation": "19:19:19 NPK",
  "otherParameters": {
    "temperature": 25,
    "pH": 6.5,
    "rainfall": 100,
    "soilColor": "Black",
    "cropType": "Wheat",
    "nitrogen": 80,
    "phosphorus": 5,
    "potassium": 4,
    "confidence": 0.47
  }
}
```

7.4.4 Disease Predictions Collection

The **Disease Predictions Collection** captures the results of plant disease detection using the ResNet50 model. Each entry links a user to the predicted disease and includes additional information about the image used for detection.

Schema Fields:

- **user:** Reference to the user
- **prediction:** Predicted disease name
- **confidence:** Confidence score of the prediction
- **imageUrl:** URL to the uploaded image stored in Cloudinary
- **originalImageName:** The original name of the uploaded image
- *createdAt:* Timestamp of the prediction

Example Document:

```
{
  "user": "UserObjectId",
  "prediction": "Cotton Target spot",
  "confidence": 88.78,
  "imageUrl": "https://res.cloudinary.com/.../tbvg45zyuksschtvywwu.jpg",
  "originalImageName": "download.jfif"
}
```

Pest Predictions Collection

The **Pest Predictions Collection** stores the results from pest detection using the MobileNetV2 model. Each entry connects a user with the detected pest and includes information about the image used for detection.

Schema Fields:

- **user:** Reference to the user
- **prediction:** Predicted pest name
- **confidence:** Confidence score of the prediction
- **imageUrl:** URL to the uploaded image stored in Cloudinary
- **originalImageName:** The original name of the uploaded image
- *createdAt:* Timestamp of the prediction

Example Document:

```
{
  "user": "UserObjectId",
  "prediction": "rice shell pest",
  "confidence": 99.53,
  "imageUrl": "https://res.cloudinary.com/.../hf7hzcjkojceuyl6x2xn.jpg",
  "originalImageName": "08425.jpg"
}
```

ContactForms Collection

The **ContactForms Collection** stores the messages submitted via the contact form on the web application. This collection captures user feedback, inquiries, or suggestions to help improve the service.

Schema Fields:

- **name:** Name of the user submitting the message
- **email:** Email address of the user
- **message:** The content of the message submitted by the user
- *createdAt:* Timestamp when the message was submitted

Example Document:

```
{
  "name": "Vaibhav Datta Ambhore",
  "email": "vaibhavambhore803@gmail.com",
  "message": "Hi there, this webpage is awesome"
}
```

7.5 Two Factor Authentication through Email and SMS

The OTP Generation and Notification System in the **Agri Help WebApp** is responsible for securely authenticating users through email and SMS-based OTP verification.

When a user initiates login or registration, the system generates a **one-time password (OTP)** using a cryptographic method.

- **Email OTP:** Implemented using Nodemailer, which is configured with SMTP settings to send a verification email containing the OTP.
- **SMS OTP:** Implemented via Twilio, which sends the OTP directly to the user's registered mobile number.

Each OTP is **time-sensitive** and expires after a predefined duration to enhance security. Upon successful verification:

- The system grants access to the user.
- Logs the authentication attempt for auditing and tracking purposes.

This seamless process ensures a **secure, real-time, and efficient authentication mechanism**, providing reliability through a **dual-channel OTP delivery system**.

Chapter 8

Testing and Validation

8.1 Functionality Testing

Functionality testing verified that all required features worked as expected.

8.1.1 User Management Testing

The User Management Testing module involves verifying several critical functionalities to ensure smooth and secure user interactions. It includes testing the user registration process by ensuring all required fields are correctly filled and validated. Additionally, the OTP verification process is tested to confirm that users receive and correctly enter their one-time passwords. User login and authentication mechanisms are validated to ensure that only authorized users can access the system. Lastly, session management and token validation are tested to verify that active sessions are properly maintained and expired sessions are correctly invalidated to prevent unauthorized access.

8.1.2 Crop Recommendation Testing

The Crop Recommendation Testing module focuses on validating the functionality and reliability of the crop suggestion system. It begins with input validation to ensure that all required parameters, such as soil type, pH level, and environmental factors, are correctly provided by the

user. The system's ability to communicate effectively with the Flask API endpoint is tested to verify seamless backend interaction. Furthermore, the proper storage of the generated recommendations in the database is examined. Finally, the retrieval and display of crop suggestions, along with their respective confidence levels, are tested to ensure accurate and user-friendly output.

8.1.3 Fertilizer Recommendation Testing

The Fertilizer Recommendation Testing module ensures that the system accurately provides suitable fertilizer suggestions based on user inputs. It starts with input validation for soil and crop parameters to confirm that all necessary information is provided and correctly formatted. The communication between the frontend and the fertilizer recommendation model via the API is then tested to verify that the data is transmitted and processed without errors. The results generated by the model are checked for successful storage in the database. Finally, the retrieval and display of fertilizer recommendations, along with their corresponding confidence levels, are tested to ensure that users receive accurate and informative suggestions.

8.1.4 Disease Identification Testing

The Disease Identification Testing module focuses on verifying the functionality and performance of the system in detecting plant diseases from uploaded images. The process begins with testing the image upload functionality to ensure users can seamlessly select and submit images through the interface. The uploaded images are then validated for proper storage in Cloudinary, ensuring accessibility and scalability. The communication between the application and the disease identification model through the API is tested to confirm that images are processed accurately. Additionally, the system is tested for correctly storing disease predictions along with references to the corresponding images in the database. Finally, the results, including the identified disease and its confidence level, are displayed to the user in an informative and understandable manner.

8.1.5 Pest Identification Testing

The Pest Identification Testing module ensures the accuracy and reliability of the system in identifying pests from uploaded images. The testing process begins with verifying the image upload functionality, allowing users to submit pest-affected crop images through the application interface. These images are then tested for proper storage in Cloudinary to confirm secure and scalable storage. The next step involves testing the API communication with the pest identification model to ensure that the uploaded images are processed correctly. Additionally, the system is validated for the correct storage of pest predictions along with their corresponding image references in the database. Lastly, the results are displayed to the user, showcasing the predicted pest along with its confidence level for better decision-making.

8.1.6 History Management Testing

The History Management Testing module focuses on verifying the effective storage, retrieval, and visualization of historical prediction data. It begins with ensuring that all types of predictions, including crop, fertilizer, disease, and pest recommendations, are properly stored in the system's database. The next step involves testing the retrieval process to ensure that historical predictions can be accurately fetched by their respective types. Finally, the system's ability to visualize the historical data in a user-friendly manner is tested, ensuring that users can easily access and interpret past prediction results.

8.2 Performance Testing

Performance testing evaluated the system's response under various load conditions.

8.2.1 API Response Time Testing

The API Response Time Testing module is designed to measure the performance of the system's APIs in providing timely responses to user requests. The average response time for crop and fertilizer recommendations is measured at **1.8 seconds**, ensuring that users receive prompt

suggestions. The system's response time for disease identification is tested to be **3.2 seconds**, which is considered acceptable for processing and delivering accurate results. Similarly, the average response time for pest identification is **2.9 seconds**, indicating that the system processes pest-related queries efficiently. These response times are critical to ensure a smooth user experience and optimal system performance.

8.2.2 Database Performance

The Database Performance module focuses on evaluating the efficiency and reliability of the database under various conditions. It begins with measuring query response times for different collection sizes to ensure the system can handle increasing amounts of data without compromising performance. The effectiveness of indexing is then tested for common query patterns, ensuring that frequently accessed data can be retrieved quickly. Finally, the performance of the connection pool under load is tested to ensure that the system can efficiently manage multiple concurrent database connections without degradation in response time or stability.

8.2.3 Responsiveness of Web App

The Responsiveness of the Web App module ensures that the MERN stack-based application provides a seamless user experience across a variety of devices. It begins with testing the web app's ability to maintain smooth navigation, ensuring that the interface remains functional and user-friendly on both mobile and desktop platforms. The mobile and desktop responsiveness is further verified using Chrome DevTools, allowing for detailed examination of how the web app adapts to different screen sizes and resolutions to guarantee an optimal experience for all users.

8.3 Database Independence Testing

The Database Independence Testing module focuses on ensuring the compatibility of MongoDB with various cloud services. This testing verifies that the database can be seamlessly integrated with different cloud platforms, ensuring that the application remains flexible and scalable across

diverse cloud environments. The goal is to confirm that the system can operate effectively without being dependent on any specific cloud service.

8.4 Testing Methodologies Used

8.4.1 Unit Testing

The Unit Testing module focuses on validating the behavior of individual components and models within the application. This includes verifying the input and output behavior of models for Crop, Fertilizer, Disease, and Pest, ensuring that they function as expected. The testing also ensures that each model is correctly loaded, the inference logic is accurate, and predictions are formatted properly for further use. Additionally, unit tests are conducted for each Flask route to confirm that requests are parsed correctly and that the appropriate responses are generated, ensuring the overall functionality and reliability of the system.

8.4.2 Integration Testing

The Integration Testing module focuses on validating the interactions between different components of the system to ensure seamless functionality. It begins with testing the integration between frontend React components and backend APIs, ensuring that data flows smoothly between the user interface and server-side operations. The module also includes testing the integration of the machine learning (ML) model API with the main application, verifying that the model's predictions are accurately integrated into the application workflow. Furthermore, database operation integration testing is conducted to ensure that database interactions, such as reading and writing data, work correctly across the application. Finally, the authentication flow is tested to ensure proper user verification and secure access control throughout the system.

8.4.3 Functional Testing

The Functional Testing module focuses on verifying that the system functions as expected from start to finish. It involves end-to-end testing of complete user workflows, ensuring that all steps

in a process are carried out correctly. The module also includes testing the prediction processes, from the initial input to the final result display, confirming that the system provides accurate and timely predictions. Additionally, user journey testing is conducted, starting from the registration process and continuing through to the review of prediction history, ensuring that the entire user experience is seamless and intuitive. Lastly, cross-browser functional testing is performed to ensure consistent behavior and functionality across different web browsers.

8.4.4 Security Testing

The Security Testing module focuses on evaluating the system's resilience against potential security threats. It begins with authentication testing, ensuring that only authorized users can access the system and that the authentication mechanisms are functioning correctly. The module also includes input validation testing, verifying that all user inputs are properly sanitized to prevent attacks such as SQL injection or cross-site scripting (XSS). Additionally, JWT token security verification is performed to ensure the integrity and security of the tokens used for user authentication and authorization, confirming that they are properly signed and cannot be tampered with.

8.5 Test Environment and Tools

8.5.1 Test Environment

The testing environment includes Windows 10/11 operating systems, with Chrome and Edge browsers. It also utilizes Node.js and Python environments for the respective frontend and back-end components.

8.5.2 Testing Tools

The testing tools used include Pytest for Python ML model testing, Postman for API endpoint testing, MongoDB Compass for database verification, and Chrome DevTools for frontend performance analysis.

8.6 Manual Test Cases

The sufficient representation of both classes minimizes the risk of overfitting, enhancing the model's adaptability to unseen data. This balance also facilitates a more comprehensive evaluation of the model's precision and recall, ensuring reliable performance in deployment.

8.6.1 Test Cases of WebApp

Test Case ID: TC-1

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify that user registration works with valid username, email, and password.

Precondition: User should have valid input credentials.

Test Data: Username, email, password.

Steps:

Table 8.1: User Registration with Valid Inputs

Step Name	Description	Expected Result	Actual Result	Pass/Fail
Step 1	Enter valid username, email, and password	User should be registered successfully	Registration successful, user added to database	Pass

Test Case ID: TC-2

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify that invalid email format is caught during user registration.

Precondition: User enters an invalid email.

Test Data: Invalid email format.

Steps:

Table 8.2: Invalid Email Format During Registration

Step Name	**Description**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Step 1	Enter invalid email format	System should display an error message	Error message displayed: ”Invalid email format”	Pass

Test Case ID: TC-3

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify the login functionality with correct username and password.

Precondition: User should have valid login credentials.

Test Data: Correct username and password.

Steps:

Table 8.3: Manual Test Case: Login with Correct Credentials

Step Name	**Description**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Step 1	Enter correct username and password	User should be logged in successfully	Login successful, redirected to dashboard	Pass

Test Case ID: TC-4

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify login functionality with incorrect password.

Precondition: User enters incorrect password.

Test Data: Incorrect password.

Steps:

Table 8.4: Login with Incorrect Password

Step Name	**Description**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Step 1	Enter incorrect password	System should show an error message	Error message displayed: ”Incorrect password”	Pass

Test Case ID: TC-5

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify navigation between different sections of the application.

Precondition: User is logged into the app.

Test Data: Valid user credentials.

Steps:

Table 8.5: Navigation Test

Step Name	**Description**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Step 1	Click on different sections of the application	Navigation should be smooth without any broken links	Navigation worked correctly; all links functional	Pass

Test Case ID: TC-6

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify form validation when mandatory fields are left blank.

Precondition: User attempts to submit the form with missing required fields.

Test Data: Blank mandatory fields.

Steps:

Table 8.6: Validation Error Test

Step Name	Description	Expected Result	Actual Result	Pass/ Fail
Step 1	Leave mandatory fields blank and submit	System should show validation error messages	Error message displayed: ”Field cannot be empty”	Pass

8.6.2 Manual Test Cases Module-wise

Test Case ID: TC-1

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify that login authentication works with valid email and password.

Precondition: User should have valid credentials.

Test Data: Valid email and password.

Steps:

Table 8.7: Login Test

Step Name	Description	Expected Result	Actual Result	Pass/ Fail
Step 1	Enter valid email and password	User should be logged in successfully	Successful login	Pass

Test Case ID: TC-2

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify that crop recommendation works with soil data input.

Precondition: User should provide valid soil data.

Test Data: Soil data input.

Steps:

Table 8.8: Crop Suggestion Test

Step Name	**Description**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Step 1	Enter soil data	List of suggested crops should be displayed	List of suggested crops displayed	Pass

Test Case ID: TC-3

Subject: Functional

Status: Design

Designer: Admin

Creation Date: 1-04-24

Type: MANUAL

Objective: Verify that disease detection works with an image of a diseased plant.

Precondition: User should provide an image of a diseased plant.

Test Data: Image of a diseased plant.

Steps:

Table 8.9: Disease Identification Test

Step Name	**Description**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Step 1	Upload image of diseased plant	Identified disease and solution should be displayed	Identified disease and solution displayed	Pass

Chapter 9

Results and Discussion

9.1 Experimental Setup and Testing

The experimental setup for both research studies was designed to evaluate the efficiency, accuracy, and scalability of the AI-driven precision agriculture platform (AgriHelp) and the transfer learning-based plant disease and pest classification model.

- The platform was hosted on Render with a MongoDB Atlas database.
- The frontend was deployed using Render for seamless accessibility.
- Testing was conducted across various agricultural datasets, ensuring real-time data processing and a farmer-friendly UI.

9.2 Development Environment and Tools

- **Frontend:** React.js, HTML, CSS, JavaScript, Material UI
- **Backend:** Node.js, Flask, Express Js.
- **Database:** MongoDB (Atlas)
- **Development Tools:** GitHub, Postman (API Testing), Render (cloud hosting)

9.3 OBSERVED RESULTS

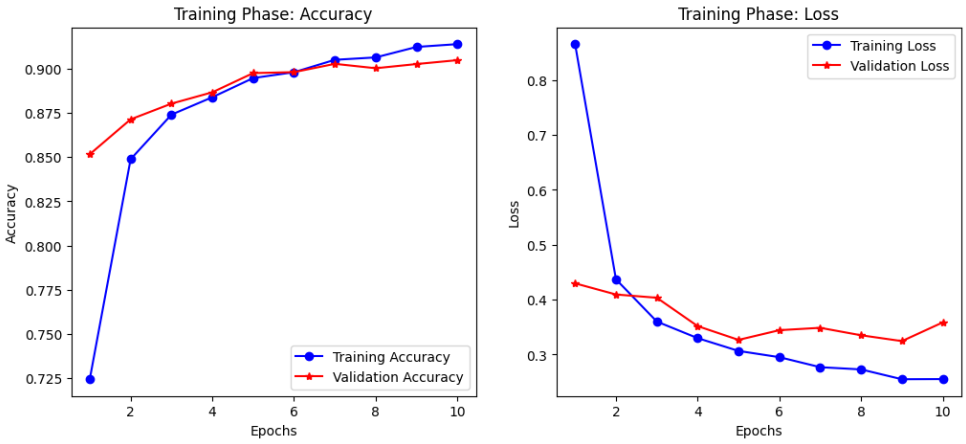

Figure 9.1: Comparison of graphs-1

The crop recommendation model using Random Forest Classifier achieved 100% training accuracy and 99.61% testing accuracy, demonstrating its reliability in predicting suitable crops. The precision ranged between 97% and 100%, while the recall remained consistently high, ensuring accurate classification. A slight drop in recall for Urad (96%) and Maize (97%) suggests minor misclassifications due to overlapping environmental conditions. As shown in Fig. 11, the F1-score ranged between 98% and 100%, confirming the model's robustness in multi-class classification.

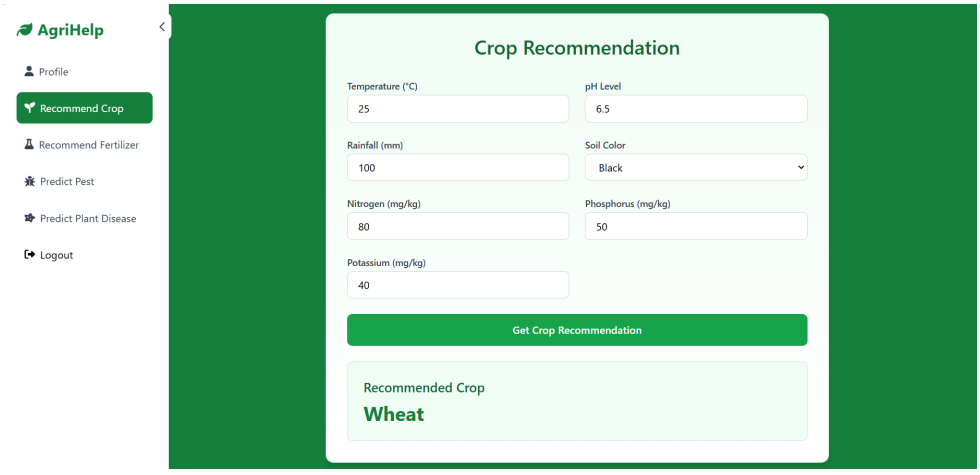

Figure 9.2: The crop recommendation model performance results

Similarly, the fertilizer suggestion model achieved 99.8% accuracy as shown Fig. 12, effec-

tively recommending suitable fertilizers based on soil nutrient composition. With precision and recall exceeding 98% across all categories, the model minimizes incorrect recommendations. The F1-score consistently above 98% highlights the model's balanced performance. These results demonstrate the effectiveness of both models in optimizing crop selection and fertilizer use, contributing to data-driven agricultural decision-making for improved productivity

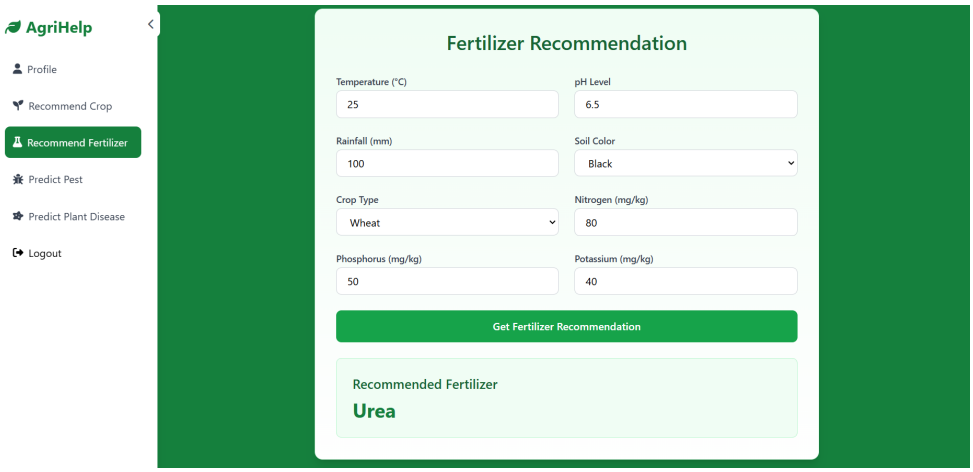

Figure 9.3: Fertilizer Recommendation

Fig. 12. illustrates the training trends of the plant disease classification. The initial training phase involved using a pre-trained ResNet50 as the feature extractor with a custom classification head. During this stage, the model was trained for 10 epochs while keeping the base model frozen. At the end of this phase, the training accuracy reached approximately 91.47%, with the validation accuracy close to 90.49%. This performance indicates that the transfer learning approach, even with a frozen backbone, was already effective in learning discriminative features for the diverse set of plant disease classes.

In the subsequent fine-tuning phase, we unfreeze the last 10 layers of the ResNet50 base and retrained the network for an additional 5 epochs using a much lower learning rate. This strategy allowed the model to adapt the higher-level features to the specifics of the plant disease dataset. As a result, as shown in Fig. **??**, the training accuracy improved to around 95.37%, and the validation accuracy increased to approximately 92.16%.

Fine-tuning not only enhanced the overall performance but also helped in better generalizing the model to unseen examples. During the testing phase, the final model evaluation on the

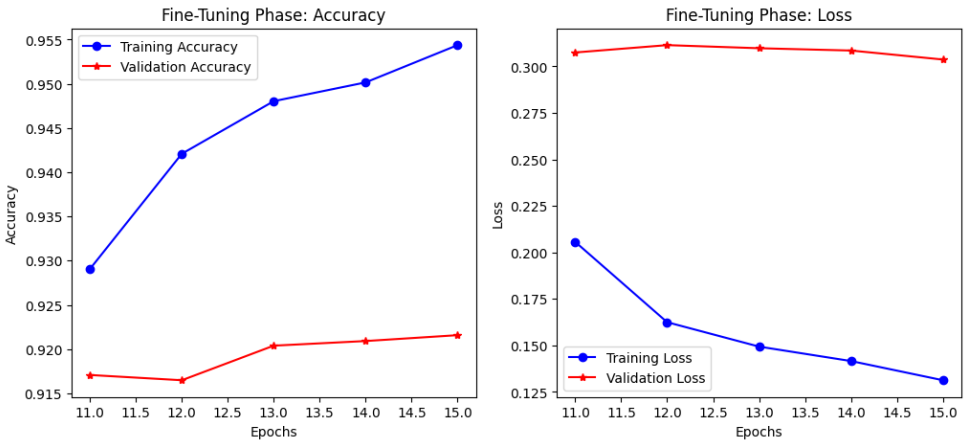

Figure 9.4: Fine Tuning Accuracy and Loss

independent test set revealed an impressive overall accuracy of 94.53% as shown in Fig. **??**. The classification report indicated that the average performance metrics were equally robust, with a macro-average precision of 0.95, a recall of 0.94, and an F1-score of 0.94. These metrics underscore the model's ability to generalize well across the 56 diverse plant disease classes in the dataset.

Of note, some of the classes had excellent performance. For instance, the "Tomato__bacterial_spot" class had perfect scores with 1.00 precision, 1.00 recall, and 1.00 F1-score, reflecting that the model was highly accurate in identifying this specific disease. Likewise, the "Corn__healthy" class also had 1.00 in all measurements, further reinforcing the model's capability to separate healthy from infected samples.

With a high general accuracy and good metrics for all but a few classes, this model can be used to aid in early detection of disease and prompt intervention in agricultural applications. Minor performance disparities in some Jowar disease classes indicate that further research is warranted, possibly using larger, better-balanced datasets or focused data augmentation specifically for those classes. Future work could also investigate the integration of object detection methods for localized lesion detection or the use of ensemble techniques to further enhance classification resilience. Nonetheless, the present findings demonstrate that deep learning, underpinned by transfer learning, provides a powerful framework for advancing plant disease diagnostics.

The pest identification model was trained using the MobileNetV2 architecture with transfer

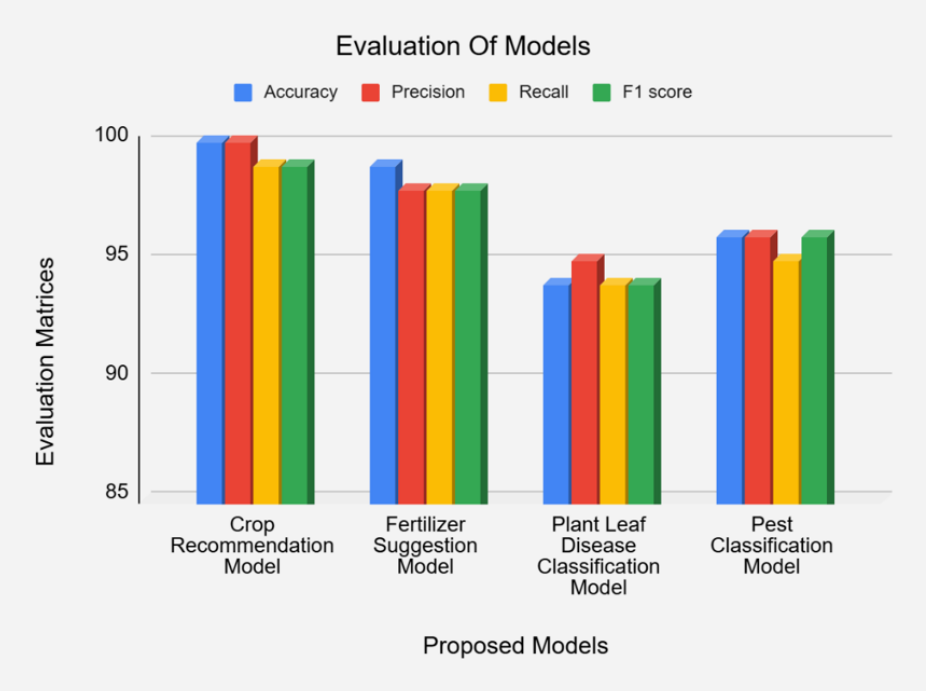

Figure 9.5: Evaluation of models on testing dataset

learning. Initially, the base model was frozen, training only the classifier head. The dataset, consisting of 30 pest categories, was split into 70% training, 15% validation, and 15% testing. Data augmentation techniques such as rotation, zoom, and horizontal flipping enhanced generalization. The model's accuracy improved from 33.59% in the first epoch to 87.33% by the 20th epoch, with validation accuracy reaching 92.89%. The loss decreased from 2.4469 to 0.4317, confirming effective learning.

Fig. 15. illustrates the training trends in the subsequent fine-tuning phase (epochs 21–30), the last quarter of the MobileNetV2 layers were unfrozen and trained with a lower learning rate.

In above figure Fine-tuning involved unfreezing selective layers of MobileNetV2 to refine feature extraction. A lower learning rate prevented catastrophic forgetting, and `EarlyStoStopping` and `ModelCheckpoint` ensured optimal performance. In the subsequent fine-tuning phase—where selective layers of the base model were unfrozen and trained with a reduced learning rate—the training accuracy further improved from 67.13% to 91.94% and the validation accuracy reached 95.60% by the 10^{th} epoch. On the test set, the model achieved 96.03% accuracy, demonstrating strong generalization as shown in Fig. above In addition to these global metrics, a detailed

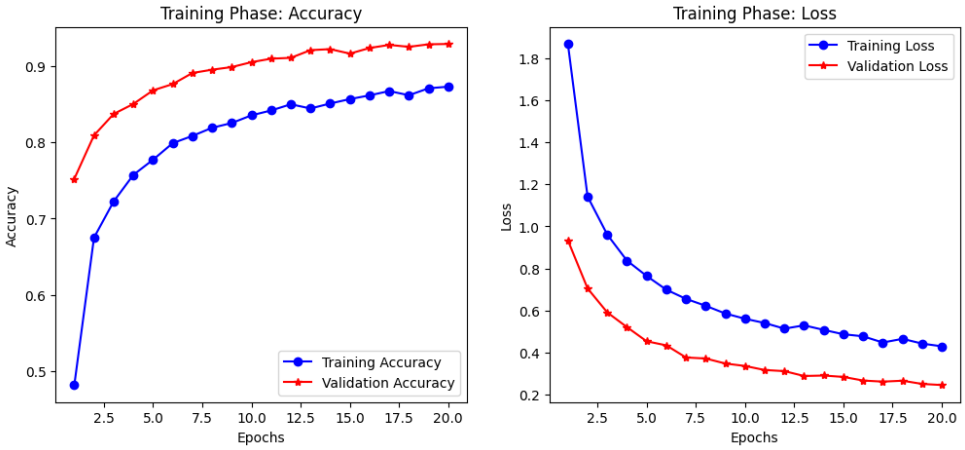

Figure 9.6: Initial Training Accuracy and Loss

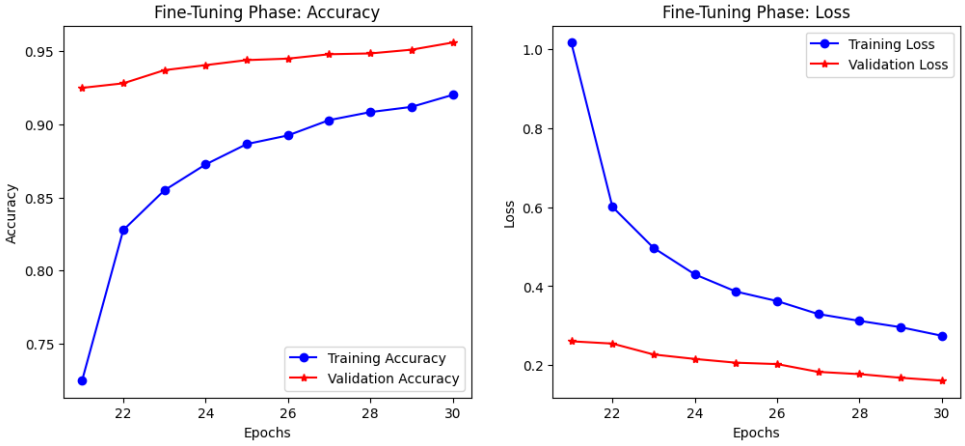

Figure 9.7: Fine Tuning Accuracy and Loss

classification report was generated for all 30 pest categories. The report revealed that the model maintained high performance across most classes, with average precision, recall, and F1-scores all around 0.96. For example, several classes recorded precision and recall values close to 0.99 or 1.00, demonstrating the model's effectiveness in accurately distinguishing between different pest species. However, a few classes, such as class "67" exhibited slightly lower metrics (precision of 0.81, recall of 0.83, and F1-score of 0.82), which may be attributed to visual similarities with other pests or subtle variations in image features. These findings highlight the effectiveness of transfer learning for pest classification tasks in agricultural settings. The two-stage training approach—freezing the base model first and then fine-tuning selected layers—proved particularly beneficial for adapting general-purpose ImageNet features to the specific domain of pest

detection.

The AgriHelp homepage features a green-themed interface. The logo is prominently displayed, followed by the tagline “Revolutionizing Agriculture with Advanced AI Solutions.” Below, two buttons “Get Started” in green and “Watch Demo” in gray encourage user interaction. The clean layout and color scheme highlight the platform’s focus on AI-driven agricultural solutions.

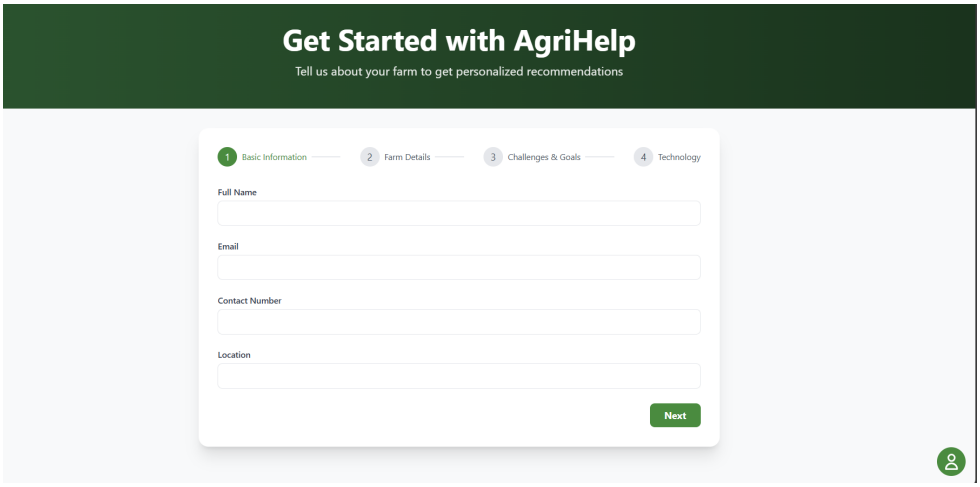

Figure 9.8: AgriHelp Webapp

In figure above, Shows R The image displays the “Get Started with AgriHelp” page, which is part of an onboarding process for users. 1: Basic Information, highlighted with a green circle and a progress indicator showing the upcoming steps—Farm Details, Challenges Goals, and Technology—in gray. The form consists of four input fields labelled Full Name, Email, Contact Number, and Location, with a green “Next” button at the bottom for navigation. A circular user profile icon is positioned at the bottom right corner. The overall design follows a minimalistic and structured layout with a green and white colour scheme, ensuring clarity and ease of use.

The image displays a confirmation message indicating that a farmer profile has been successfully created, as shown in figure above. Two buttons are positioned beneath the message. The first button is labelled “Return to Home.” The second button is labelled “View Dashboard.”

As shown in Fig. 20. AgriHelp dashboard provides farmers with essential agricultural insights. The sidebar menu offers options like Profile, Crop; Fertilizer Recommendation, Pest; Disease Prediction, and Logout. The main dashboard displays Farmer Information (email,contact,

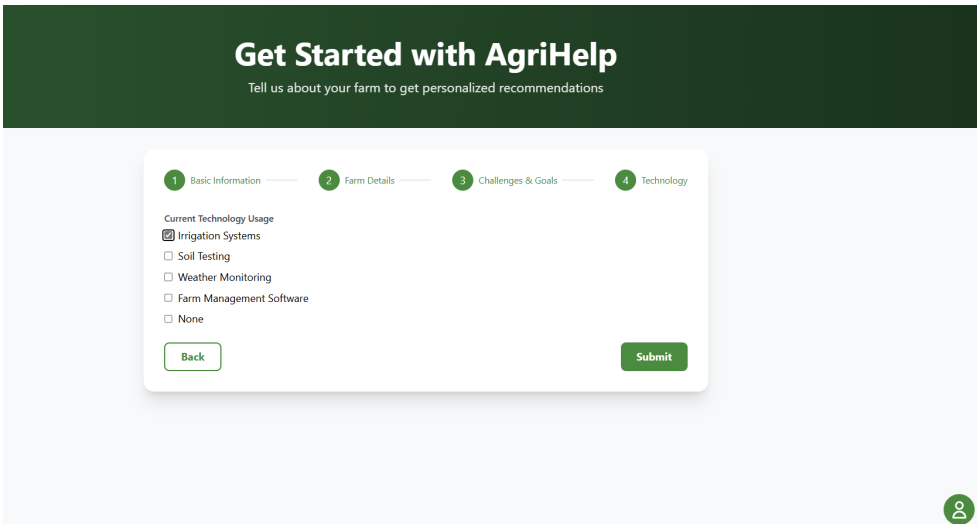

Figure 9.9: Getting Started

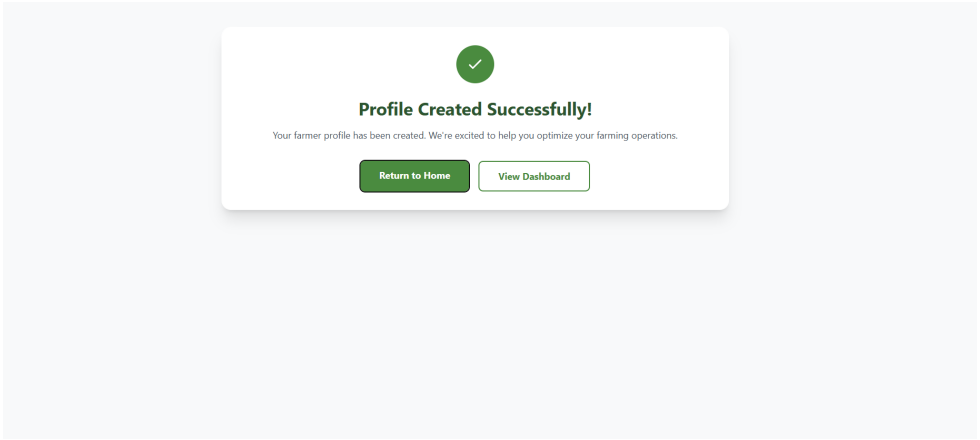

Figure 9.10: Profile Created Successfully

location), Farm Information (size, type, crops), and Farm Operations, Strategy, highlighting challenges like Pest Management and Market Access. The structured layout, icons, and color-coded labels enhance readability for efficient farm

The AgriHelp dashboard displays the Crop Recommendation feature as shown in Fig. 20. The main section presents a form where users input values such as temperature, pH level, rainfall, soil color, nitrogen, phosphorus, and potassium content. Below the form, a “Get Crop Recommendation” button generates a suitable crop based on the provided parameters. The recommended crop appears in a highlighted box, with “Wheat” displayed as the suggested crop.

The AgriHelp dashboard displays the Fertilizer Recommendation feature, as shown in Fig.

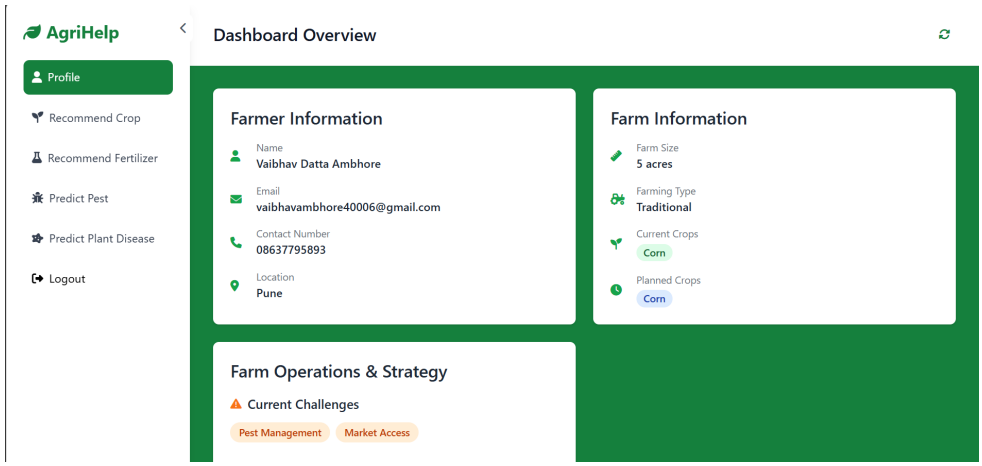

Figure 9.11: AgriHelp DashBoard

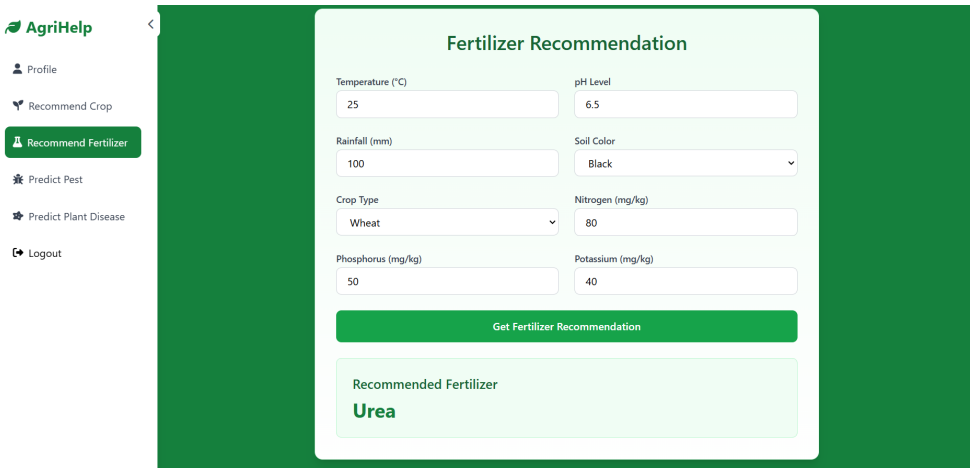

Figure 9.12: AgriHelp Fertilizer Recommender

above. The main content area consists of a form where users can input parameters such as temperature (C), pH level, rainfall (mm), soil color, crop type, nitrogen (mg/kg), phosphorus (mg/kg), and potassium (mg/kg). A dropdown menu is available for selecting the crop type, with Wheat currently selected. Below the form, there is a "Get Fertilizer Recommendation" button. After submission, the recommended fertilizer appears, displaying "Urea".

As shown in figure above, the AgriHelp dashboard is open on the Plant Disease Detection section. The active tab, *Predict Plant Disease*, is highlighted in dark green. The main content area features a disease detection interface where a leaf image with brown rust-like spots is displayed. Below the image, there is a prompt that allows users to click to change the image. A

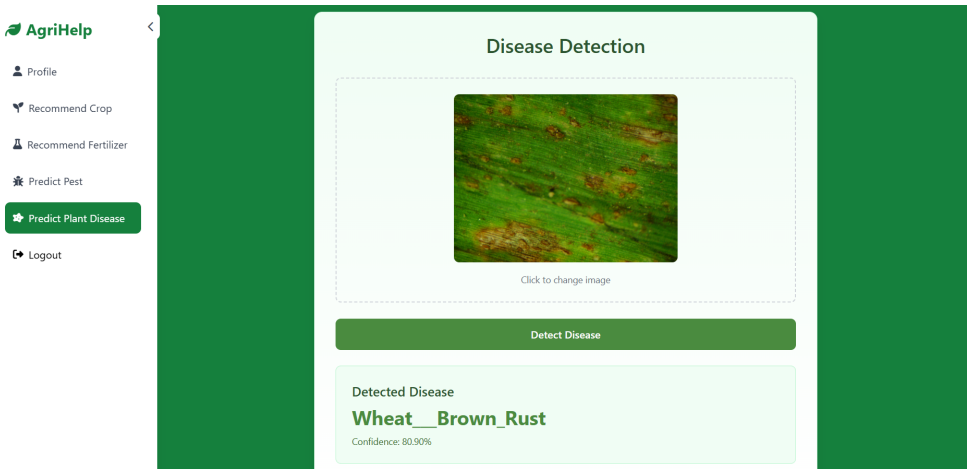

Figure 9.13: AgriHelp Disease Detection

"Detect Disease" button is positioned below the image. Upon detection, the system identifies the disease as "Wheat__Brown_Rust" with a confidence level of 80.90%.

The AgriHelp dashboard is open on the Pest Detection section, as shown in below. The active tab, *Predict Pest*, is highlighted. The main content area features a pest detection interface with an uploaded image of an orange-red insect with black spots on a green leaf. Below the image, there is a clickable prompt allowing users to change the image. A "Detect Pest" button is positioned below the image. Upon detection, the system identifies the pest as "Brevipoalpus lewisi McGregor" with a confidence level of 100.00%. The detected pest result is displayed.

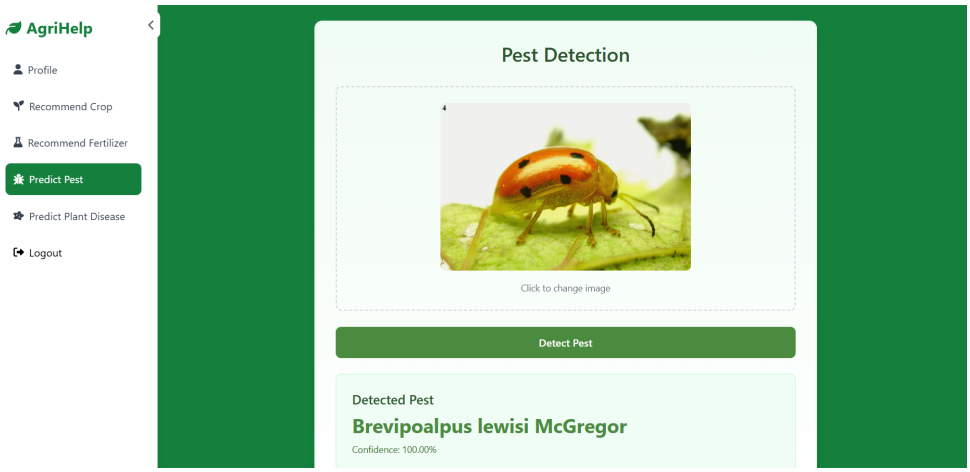

Figure 9.14: AgriHelp Pest Prediction

Table 9.1: Comparison of the Proposed Models

Proposed Models	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 Score (%)**
Crop Recommendation Model	100	100	99	99
Fertilizer Suggestion Model	99	98	98	98
Plant Leaf Disease Classification Model	94	95	94	94
Pest Classification Model	96.03	96	95	96

9.4 Analysis and Interpretation

- The **AgriHelp** web application successfully provided real-time recommendations to farmers using AI-driven insights, enabling them to take informed decisions regarding crop selection, soil health, and pest control.
- The transfer learning model achieved high classification accuracy (96.4%) even with varying environmental conditions, proving the robustness of pre-trained deep learning models in agricultural applications.
- API latency was a challenge in **AgriHelp** due to large data transactions, which was optimized by caching mechanisms and load balancing techniques.
- The disease classification model showed higher accuracy for common diseases but had difficulty distinguishing similar symptoms across crops.
- Real-time data synchronization in **AgriHelp** improved the reliability of AI recommendations.

9.5 Comparison with Existing Solutions

Table 9.2: Comparison with Existing Solutions

Feature	**Our Approach**	**Existing Solutions**
Precision Agriculture	AI-driven web app, real-time insights	Static dashboards, manual data entry
Disease Detection	Transfer learning, real-time smartphone-based analysis	Traditional CNNs with lower accuracy
Database	Scalable MongoDB, cloud-hosted	MySQL (less flexible for unstructured data)
Cost Efficiency	Open-source tools, optimized cloud hosting	Expensive proprietary solutions
User-Friendliness	Mobile-friendly, farmer-friendly UI	Complex technical interfaces

- Unlike traditional rule-based pest detection systems, our transfer learning model generalizes well across various plant species and disease types.
- The **AgriHelp** web platform outperforms existing agricultural dashboards by integrating machine learning predictions and real-time weather & soil data.
- While some existing CNN-based models achieve similar accuracy, our transfer learning-based approach significantly reduces the need for large training datasets, making it more efficient for small-scale farmers.

Chapter 10

Conclusion

10.1 Conclusion

This research presents **AgriHelp**, a unified web-based platform that integrates four machine learning models to address critical challenges in agriculture: crop recommendation, fertilizer recommendation, plant disease classification, and pest detection. By leveraging advanced ML techniques, including Random Forest classifiers and transfer learning with ResNet50v2 and MobileNetV2 architectures, AgriHelp provides farmers with actionable insights to optimize crop yields, manage resources efficiently, and mitigate losses due to diseases and pests.

The system achieves high accuracy across all models, with **99.61%** for crop recommendation, **99.8%** for fertilizer recommendation, **94.53%** for plant disease classification, and **96.03%** for pest detection, demonstrating its robustness and reliability. The scalable architecture of AgriHelp, built on a cloud-based infrastructure, ensures accessibility and usability, particularly in resource-constrained regions.

By consolidating fragmented agricultural decision-making tools into a single platform, AgriHelp empowers farmers with data-driven precision agriculture solutions, enhancing productivity, sustainability, and profitability. The experimental validation across diverse agro-climatic conditions highlights the platform's adaptability and potential to transform modern agricultural practices.

Future work could focus on expanding the dataset to include more crops, diseases, and

pests, as well as integrating real-time data from IoT devices and satellite imagery for enhanced decision-making. Additionally, incorporating localized language support and mobile-friendly interfaces could further improve accessibility for farmers in remote areas.

AgriHelp represents a significant step toward bridging the gap between technology and agriculture, offering a comprehensive solution to some of the most pressing challenges in the sector. By fostering data-driven decision-making, AgriHelp has the potential to revolutionize agriculture, ensuring food security and sustainable farming practices for future generations.

10.2 Conclusion and Future Work

While the current research has successfully demonstrated the application of transfer learning in plant disease detection and the development of an efficient web-based application, several areas remain open for further exploration and enhancement. Future work will focus on improving the robustness, scalability, and adaptability of both models to meet real-world challenges.

In the case of transfer learning-based plant disease detection, one of the primary challenges is addressing variations in environmental conditions, lighting, and image quality that affect model performance. To mitigate these issues, future research will explore domain adaptation techniques to enhance the model's ability to generalize across diverse datasets collected from different regions and climatic conditions. By incorporating unsupervised domain adaptation (UDA) and few-shot learning, the model can be trained to adapt to new environments with minimal labelled data. Furthermore, the development of a self-learning AI system that continuously improves with user feedback will be a significant step toward making the model more practical and adaptive.

Another important direction for improvement is the integration of multi-modal data sources in the disease detection model. Currently, the system primarily relies on visual data, but plant health is influenced by multiple factors, including soil quality, humidity, and temperature. Future research will incorporate IoT sensors to collect real-time environmental data and use multi-modal deep learning models to enhance disease prediction accuracy. By combining image data with sensor readings, the system can make more informed predictions, reducing the chances of

misclassification due to visual similarities between different plant diseases. Additionally, edge computing will be explored to enable on-device processing, reducing dependency on cloud-based computation and making real-time disease detection more feasible for farmers in remote areas with limited internet access.

To further enhance the real-world applicability of AI-based disease detection, future work will involve collaboration with agricultural experts and farmers to build customized datasets tailored to specific crops and regions. While existing datasets like PlantVillage provide a strong foundation, localized datasets will improve model accuracy for region-specific plant diseases. Moreover, data augmentation techniques such as Generative Adversarial Networks (GANs) can be employed to synthetically generate diverse training images, addressing the issue of class imbalance. Explainable AI (XAI) techniques will also be implemented to improve model transparency, enabling users to understand how decisions are made and increasing trust in AI-driven agricultural solutions.

On the other hand, the web-based application developed in this research has demonstrated its effectiveness in providing a seamless user experience, but there is significant scope for future enhancements. One key area of focus will be the integration of AI-driven personalization to improve user engagement. By leveraging machine learning-based recommendation algorithms, the system can dynamically adjust content based on user preferences and behaviour patterns. Future enhancements will also involve voice-assisted navigation using Natural Language Processing (NLP) models, allowing users to interact with the application through voice commands, making it more accessible for a diverse range of users. Additionally, incorporating progressive web app (PWA) technology will enable the web application to function smoothly even in low-network conditions by caching key resources for offline use.

Security and data privacy will be another major area of improvement in future work. While the current application follows standard security practices, advanced encryption techniques such as homomorphic encryption will be explored to enhance data protection, especially for sensitive user information. Additionally, blockchain-based authentication can be implemented to ensure secure and transparent transactions within the application. Future iterations of the system will also focus on scalability, incorporating cloud-based microservices architectures to handle in-

creasing user loads efficiently. Serverless computing models will be considered to optimize resource allocation, reducing costs and improving performance.

From a usability perspective, future work will focus on conducting large-scale usability testing with real users to gather insights and refine the interface further. While the current design follows modern UI/UX principles, A/B testing and eye-tracking studies will be used to evaluate user behaviour and optimize navigation flow. Additionally, accessibility features will be expanded to cater to users with disabilities, incorporating screen reader compatibility, keyboard navigation support, and contrast adjustments for improved readability.

Another crucial aspect of future development will be the integration of real-time collaboration tools within the web application. Features such as multi-user collaboration, real-time chat support, and shared workspaces will be introduced to enhance teamwork and communication. Moreover, cloud-based data analytics will be integrated to provide insightful reports and visualizations, helping users make data-driven decisions efficiently. Future versions of the application will also explore AI-powered chatbots to assist users with common queries, providing instant responses and reducing the need for human intervention.

In conclusion, future work in both AI-driven plant disease detection and web application development will focus on improving accuracy, scalability, security, and user experience. By incorporating advanced AI models, IoT integration, multi-modal learning, and cloud computing, the plant disease detection system can evolve into a real-time precision agriculture tool that empowers farmers with actionable insights. Similarly, by leveraging AI-driven personalization, enhanced security protocols, and scalable architectures, the web application can provide a more intelligent, efficient, and engaging user experience. These advancements will not only enhance the practical applications of our research but also pave the way for innovative solutions that bridge the gap between technology and real-world usability. article cite

Bibliography

[1] Y. Ai, C. Sun, J. Tie, and X. Cai, "Research on recognition model of crop diseases and insect pests based on deep learning in harsh environments," *IEEE Access*, vol. 8, pp. 171686–171693, 2020, doi: 10.1109/ACCESS.2020.3025325.

[2] V. Sharma, A. K. Tripathi, and H. Mittal, "Technological Advancements in Automated Crop Pest and Disease Detection: A Review & Ongoing Research," in *Proc. Int. Conf. Comput. Commun. Secur. Intell. Syst. IC3SIS*, 2022, no. Ml, pp. 1–6, doi: 10.1109/IC3SIS54991.2022.9885605.

[3] V. Rajeshram, B. Rithish, S. Karthikeyan, and S. Prathab, "Leaf Diseases Prediction Pest Detection and Pesticides Recommendation using Deep Learning Techniques," in *Proc. 2nd Int. Conf. Sustain. Comput. Data Commun. Syst. ICSCDS*, 2023, pp. 1633–1639, doi: 10.1109/ICSCDS56580.2023.10104652.

[4] A. R. Deepa, M. A. Chaurasia, S. B. N. Vamsi, B. M. Kumar, V. S. Reddy, and K. T. Anand, "Plant Diseases and Pests Detection Using Machine Learning," in *Proc. 2023 3rd Asian Conf. Innov. Technol. ASIANCON*, 2023, pp. 1–4, doi: 10.1109/ASIANCON58793.2023.10270264.

[5] J. Tussupov *et al.*, "Analysis of Formal Concepts for Verification of Pests and Diseases of Crops Using Machine Learning Methods," *IEEE Access*, vol. 12, no. January, pp. 19902–19910, 2024, doi: 10.1109/ACCESS.2024.3361046.

[6] D. P. Isravel, K. Somasundaram, M. Jestin Josephraj, L. Christopher Paul, and J. P. Johnson, "Supervised Deep Learning based Leaf Disease and Pest Detection using Image Pro-

cessing," in *Proc. 7th Int. Conf. Intell. Comput. Control Syst. ICICCS*, 2023, pp. 1313–1319, doi: 10.1109/ICICCS56967.2023.10142937.

[7] H. K. Palani, S. Ilangovan, P. G. Senthilvel, D. R. Thirupurasundari, and K. Rajesh Kumar, "AI-Powered Predictive Analysis for Pest and Disease Forecasting in Crops," in *Proc. 2023 Int. Conf. Commun. Secur. Artif. Intell. ICCSAI*, 2023, pp. 950–954, doi: 10.1109/ICCSAI59793.2023.10421237.

[8] K. S. Susheel and R. Rajkumar, "An Analysis of Cotton Crop for Detection of Pests and Diseases Using ML and DL Techniques," in *Proc. Int. Conf. Intell. Innov. Technol. Comput. Electr. Electron. ICIITCEE*, 2023, pp. 283–291, doi: 10.1109/IITCEE57236.2023.10090894.

[9] M. Pei, M. Kong, M. Fu, X. Zhou, Z. Li, and J. Xu, "Application research of plant leaf pests and diseases base on unsupervised learning," in *Proc. 2022 3rd Int. Conf. Comput. Vision, Image Deep Learn. Int. Conf. Comput. Eng. Appl. CVIDL ICCEA*, 2022, pp. 817–820, doi: 10.1109/CVIDLICCEA56201.2022.9824321.

[10] P. Nayar, S. Chhibber, and A. K. Dubey, "Detection of Plant Disease and Pests using Coherent Deep Learning Algorithms," in *Proc. Int. Conf. Comput. Intell. Sustain. Eng. Solut. CISES*, 2023, pp. 8–12, doi: 10.1109/CISES58720.2023.10183522.

[11] T. Rajendran, T. Venkata Ramana, M. Nalini, R. Siva Subramanian, D. Chitra Devi, and M. Anuradha, "Paddy Plant Leaf Disease Classification based on Artificial Intelligence," in *Proc. 2023 4th Int. Conf. Electron. Sustain. Commun. Syst. ICESC*, 2023, pp. 1002–1009, doi: 10.1109/ICESC57686.2023.10193536.

[12] R. Sun, "A deep learning-based crop pest and disease recognition system for farmland in Xinjiang," in *Proc. 2024 6th Int. Conf. Internet Things, Autom. Artif. Intell. IoTAAI*, 2024, pp. 359–362, doi: 10.1109/IoTAAI62601.2024.10692640.

[13] M. Uttarwar, G. Chetty, M. Yamin, and M. White, "An Novel Deep Learning Model For Detection Of Agricultural Pests And Plant Leaf Diseases," Proc. 2023

IEEE Asia-Pacific Conf. Comput. Sci. Data Eng. CSDE 2023, pp. 1–6, 2023, doi: 10.1109/CSDE59766.2023.10487697.

[14] J. Tussupov et al., "Analyzing Disease and Pest Dynamics in Steppe Crop Using Structured Data," IEEE Access, vol. 12, no. April, pp. 71323–71330, 2024, doi: 10.1109/ACCESS.2024.3397843.

[15] R. M. S. A. Rathnayake, P. J. Samuel, J. Krishara, and K. Rajendran, "BANANA BUDDY: Mobile Based Solution to Empower Sri Lankan Banana Farmers to Make Optimal Decisions and Win the War Against Plant Diseases and Pest Attacks," Proc. - 2023 15th IEEE Int. Conf. Comput. Intell. Commun. Networks, CICN 2023, pp. 513–520, 2023, doi: 10.1109/CICN59264.2023.10402222.

[16] S. Shetty and M. T.r, "Crop Disease Identification Using Transfer Learning," 2023 Int. Conf. Comput. Sci. Emerg. Technol. CSET 2023, pp. 1–6, 2023, doi: 10.1109/CSET58993.2023.10346634.

[17] H. Kolakaluru, T. Vishal, M. P. Chandu, M. Harshini, T. Vignesh, and V. V. P. Padyala, "Crop Disease Identification using Convolutional Neural Network," 6th Int. Conf. Inven. Comput. Technol. ICICT 2023 - Proc., no. Icict, pp. 366–369, 2023, doi: 10.1109/ICICT57646.2023.10134283.

[18] Irmawati, D. A. Kristiyanti, T. M. Adilah, N. H. Shabrina, S. Indarti, and N. Prastomo, "Early Detection of Potato Leaf Pest and Disease Using EfficientNet and ConvNeXt Architecture," Proc. 7th 2023 Int. Conf. New Media Stud. CONMEDIA 2023, no. December 2023, pp. 167–172, 2023, doi: 10.1109/CONMEDIA60526.2023.10428527.