CSE 3200: System Development Project

A MOBILE APPLICATION FOR PLANT DISEASE DETECTION

By

Sumaiya Khan

Roll: 2007031

Department of Computer Science and Engineering

Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

December, 2023

A MOBILE APPLICATION FOR PLANT DISEASE DETECTION

By

Sumaiya Khan
Roll: 2007031

Supervisor:

Abdul Aziz
Assistant Professor
Department of Computer Science and Engineering
Khulna University of Engineering & Technology

Signature

Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna 9203, Bangladesh
August, 2025

Acknowledgment

All praise is due to Almighty Allah, whose boundless mercy and blessings enabled us to successfully undertake and complete this project. We wish to express our sincere and heartfelt gratitude to our supervisor, **Abdul Aziz**, Assistant Professor of the Department of Computer Science and Engineering. His profound knowledge, insightful guidance, and unwavering encouragement played a crucial role in the realization of this work. His constructive criticism, constant supervision, and invaluable suggestions greatly shaped the direction of our project and inspired us throughout the journey. Without his dedicated support and motivating enthusiasm, the completion of this thesis would not have been possible.

Finally, we would also like to extend our deepest appreciation to everyone who supported and encouraged us, providing the opportunity and assistance needed to bring this work to fruition.

Author

Abstract

By utilizing cutting-edge machine learning technology, the PlantGuard mobile application transforms agriculture management and plant disease detection. Crop disease diagnosis has historically required specialized knowledge and can be a laborious procedure, increasing the possibility of production loss from postponed action. Using the EfficientNet architecture, PlantGuard uses deep learning models to address this issue. These models have been trained on a large dataset of over 6,000 photos of jute, rice, and maize plants in a variety of real-world settings. By extracting essential elements from user-captured photographs, this method guarantees precise and quick illness identification even in cases when image quality fluctuates.

The app is built with React Native to guarantee a smooth, intuitive user experience across devices. In addition to disease detection, PlantGuard integrates a secure e-commerce system, enabling registered users to buy plants and agricultural supplies directly through the app. User data, including detection histories and order records, are stored safely in a dedicated database for convenient access and management. An AI-powered chatbot provides instant support for users' questions about plant health and app features, while a comprehensive admin panel allows administrators to efficiently oversee users, catalog products, and review all detection results.

By combining powerful AI-driven diagnostics, secure transaction handling, and a streamlined admin panel into a single platform, PlantGuard elevates both the speed and scope of digital agricultural services, empowering users—especially in resource-limited regions—to make faster, smarter decisions in crop management and procurement.

Contents

List of Tables

List of Figures

CHAPTER I

Introduction

1.1 Background

The agriculture sector is rapidly transforming through the adoption of digital technologies and artificial intelligence, improving farmers' ability to detect plant diseases early and access the right resources for effective crop management. In many rural and resource-limited regions, farmers face challenges such as limited access to agricultural experts and quality farming supplies, leading to delays in identifying crop issues and obtaining solutions. Simultaneously, the rise of e-commerce platforms has revolutionized the way farmers procure seeds, plants, and agricultural products, enhancing convenience and broadening choices.

PlantGuard is developed as a modern solution combining machine learning-powered plant disease detection with an integrated e-commerce marketplace. Users can capture or upload photos of their jute, rice, or potato plants to receive instant disease diagnoses along with tailored recommendations for management, treatment, and preventive measures. By embedding a secure shopping experience within the app, PlantGuard enables users to directly purchase plants and essential farming inputs. All user data—including disease detection history, orders, and profiles—are efficiently managed with MongoDB, ensuring flexible, scalable, and secure storage. This seamless combination of AI-driven diagnostics and e-commerce creates a unified digital agriculture ecosystem, empowering farmers to make quicker decisions and improve crop health and productivity.

1.2 Problem Statement

Timely and accurate detection of plant diseases is a significant challenge for many farmers, particularly in rural or resource-limited areas with limited access to agricultural experts and diagnostic facilities. Delays in identifying diseases can lead to crop damage and reduced yields, negatively impacting livelihoods. Additionally, obtaining quality plants, seeds,

and agricultural products often involves complex processes such as finding trusted sellers, navigating registration, and ensuring secure payments, which can discourage farmers from accessing necessary resources.

Existing digital agriculture tools frequently separate disease detection from the purchasing of plants and farming inputs, resulting in a fragmented and inefficient experience that can lead to frustration and low adoption rates. There is a clear need for an integrated, user-friendly platform that combines accurate plant disease diagnosis with seamless procurement of agricultural products, providing farmers with a reliable and efficient solution to improve crop health and productivity from diagnosis through treatment and resource acquisition.

1.3 Objectives

The primary objectives of the PlantGuard project are:

1. To develop a user-friendly web application where users can input symptoms and obtain accurate, machine learning-based disease predictions.

2. To provide personalized recommendations encompassing medicines, precautions, workout routines, and dietary guidelines following disease detection.

3. To integrate a e-commerce module enabling users to browse, search, register, and purchase recommended medicines seamlessly.

4. To implement an administrative panel for managing medicine products, including adding, modifying, and validating product details.

5. To ensure robust database management supporting user credentials, product inventory, and administrative controls.

1.4 Scope of the Project

This project is focused on building a fully functional prototype Mobile App that addresses initial disease detection and plant purchase in one place. It covers:

1. To provide an intuitive mobile application that allows users to take or upload pictures in order to receive precise, deep learning-based diagnostics for plant diseases.

2. To offer tailored suggestions for jute, rice, and potato crop-specific disease control, treatment alternatives, and preventative actions.

3. To incorporate an e-commerce feature that will allow users to easily browse, register, and buy plants and agricultural supplies.

4. To put in place an administrative panel for adding, editing, and verifying information about users, product listings, and disease detection records.

5. To use MongoDB to provide user profiles, detection histories, product inventories, and administrative controls in order to guarantee effective and scalable database management.

The system is an additional tool for early assessment and convenience, not a replacement for expert medical diagnosis.

1.5 Unfamiliarity with the Problem and Solution

Many existing agricultural tools focus either on plant disease detection or on selling farming supplies, but rarely combine both. Integrated platforms offering accurate diagnosis alongside e-commerce and personalized crop advice are scarce, especially in rural areas. Delivering a seamless user experience that supports both remains challenging.

PlantGuard addresses this gap by merging deep learning-based disease detection with a secure marketplace in one mobile app. This innovative approach is new locally and offers farmers a comprehensive solution for healthier crops and easier access to resources.

1.6 Project Planning and Work Distribution

The project was planned in phases including requirements gathering, system design, deep learning model training, mobile app development, database structuring, and testing. Work assignments were distributed among team members according to their strengths:

1. Research, data preprocessing, and training of the EfficientNet-based plant disease detection model

2. Development of backend APIs using Node.js and MongoDB

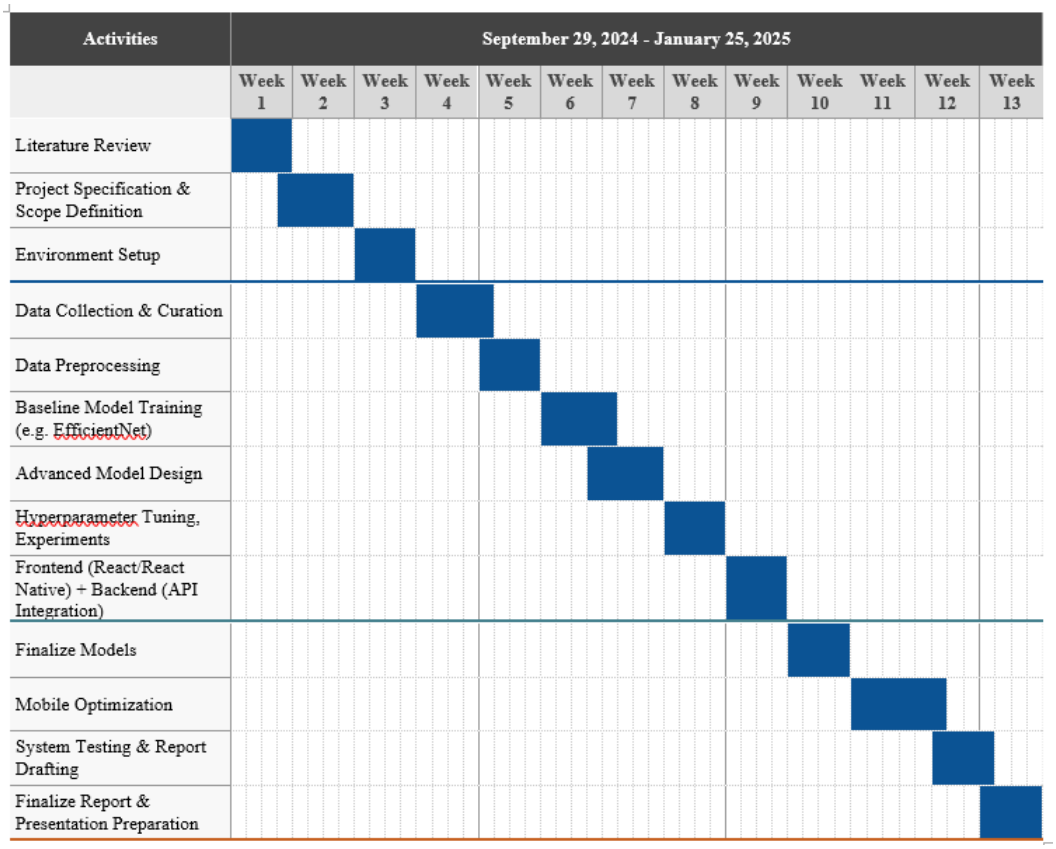

Figure 1.1: Gantt chart of the timeline of the project

3. Frontend mobile app development using React Native
4. Database schema design and management with MongoDB
5. Integrating e-commerce features and secure authentication
6. Development and testing of the admin panel for managing users, products, and detection data
7. Comprehensive testing and debugging

Regular team meetings ensured progress tracking, issue resolution, and collaborative problem-solving.

1.7 Applications of the Project

PlantGuard offers various benefits and practical applications:

1. Enabling farmers in remote or resource-limited areas to quickly detect plant diseases without expert visits.

2. Providing tailored recommendations for disease management, treatment, and crop care to support informed decision-making.

3. Facilitating easy and secure purchasing of plants and agricultural products, improving access to quality supplies.

4. Assisting agricultural advisors by streamlining preliminary disease assessments and record management.

5. Offering potential for future expansion into other farming-related services such as weather alerts, crop monitoring, and expert consultation.

1.8 Organization of the Project

This project report is organized into seven chapters. Each chapter provide us with a clear perception about how each stage was planned and executed, giving an overview of the overall project.

Chapter I introduces the project, its background, and fundamental concepts related to plant disease detection and digital agriculture.

Chapter II reviews related literature and existing systems in plant disease diagnosis and agricultural e-commerce platforms.

Chapter III details the methodology used, including system design, architecture, and implementation tools.

Chapter IV presents the implementation process, experimental setups, results, and discussion of findings.

Chapter V discusses ethical, legal, societal, and environmental implications of the project.

Chapter VI addresses complex engineering challenges encountered and the problem-solving activities undertaken.

Chapter VII concludes the thesis with summaries, limitations, and recommendations for future work.

CHAPTER II

Related Works

2.1 Introduction

The convergence of artificial intelligence, agricultural technology, and e-commerce has sparked significant research into technology-driven solutions for plant disease detection and resource accessibility. This chapter reviews existing literature and related systems that have influenced the design and development of PlantGuard. The main objective is to assess the strengths and limitations of current approaches, identify gaps in available solutions, and justify the selection of the methodologies employed in this project.

2.2 Related Terms

1. **Machine Learning in Agriculture:** Machine learning (ML) encompasses algorithms and models that enable computers to learn from data and make predictions or classifications. In agriculture, ML techniques are increasingly applied for plant disease detection, crop yield prediction, and personalized farming recommendations.

2. **E-commerce in Agriculture:** E-commerce platforms have transformed the agricultural sector by facilitating online purchase of plants, seeds, and farming supplies. These solutions often include secure payment systems and inventory management, helping farmers access quality products conveniently.

3. **Plant Disease Detection Systems:** These are applications or tools that analyze images or other data from crops to identify diseases, typically using heuristic or machine learning models, aiding farmers in early diagnosis and crop management.

2.3 Related Works

2.3.1 Plant Disease Detection Systems

Numerous platforms use machine learning to identify plant diseases from images or symptoms. Techniques such as convolutional neural networks (CNNs) and other deep learning methods have been widely applied for disease classification in crops like rice, potato, and jute. While some commercial and academic tools offer accurate detection, many focus solely on image analysis and lack integrated recommendations for treatment or resource procurement. Recent studies emphasize the need for models that explain predictions and support multiple crop types and diseases, yet comprehensive systems that combine diagnosis with actionable advice remain rare.

2.3.2 Agricultural E-commerce Platforms

Online agricultural marketplaces such as AgroStar and BigHaat provide farmers with the ability to browse and purchase seeds, plants, and farming inputs. Academic research highlights platforms that manage inventory automatically, provide detailed product information, and simplify order processing. These systems improve convenience and supply accessibility but generally do not integrate disease detection or personalized crop care guidance before purchase.

2.3.3 Comparative Analysis

While plant disease detection and agricultural e-commerce platforms are well-developed individually, integrated solutions that combine intelligent disease diagnosis with seamless purchase and management features are limited. Few existing projects offer a unified workflow from disease identification through personalized recommendations to product procurement.

Table 2.1: Comparison of Existing Solutions and Feature Coverage

Solution/ Platform	**Disease Prediction**	**Personalized Advice**	**Medicine E-commerce**	**Admin/ Product Management**
AgroStar	No	No	Yes	Yes
Research Tools	Yes	Partial	No	No
PlantGuard	Yes	Yes	Yes	Yes

2.4 Research Gap and Solution Summary

Despite significant advances in AI-based plant disease detection and agricultural e-commerce, integrated platforms that combine both—offering tailored crop diagnostics along with seamless access to farming products—are still scarce, especially in resource-limited settings. Existing solutions commonly:

1. Separate disease identification from product purchasing, causing fragmented user experiences.
2. Provide generic advice lacking clear, actionable guidance,
3. Or offer interfaces that are not easily accessible to diverse farming populations.

PlantGuard bridges this gap by offering:

1. A deep learning-powered, real-time plant disease detection system,
2. Personalized, actionable recommendations for crop management,
3. Seamless integration with a secure e-commerce platform for purchasing plants and supplies,
4. Robust admin tools for dynamic user, product, and data management.

By addressing these shortcomings, PlantGuard represents a novel, user-centric solution that unites diagnosis and procurement, enhancing agricultural productivity and resource accessibility.

CHAPTER III

Methodology

3.1 Introduction

This chapter outlines the methodological framework of the PlantGuard project. It describes the overall system design, detailing the workflows of the two primary components—plant disease detection and agricultural e-commerce—and explains the technical implementation behind them. By discussing the choice of technologies, machine learning model training, and database architecture, this chapter illustrates how the project's objectives were realized through a robust mobile application.

3.2 System Design

PlantGuard's design prioritizes modularity, scalability, and data security. The system is composed of two closely integrated main modules: the plant disease detection engine and the e-commerce platform for agricultural product purchasing. Both modules share backend services, user authentication, and a unified database.

3.2.1 Architecture Overview

PlantGuard follows a layered architecture consisting of the following components:

1. **User Interface (UI):** Developed using React Native, the UI enables users to capture or upload images, view disease diagnosis results, browse products, and manage orders seamlessly.

2. **Backend Server:** Built with Node.js and Express, the backend manages machine learning inference, authentication, and database operations.

3. **Machine Learning Module:** Employs a trained EfficientNet-based model for real-time plant disease detection from user-submitted images.

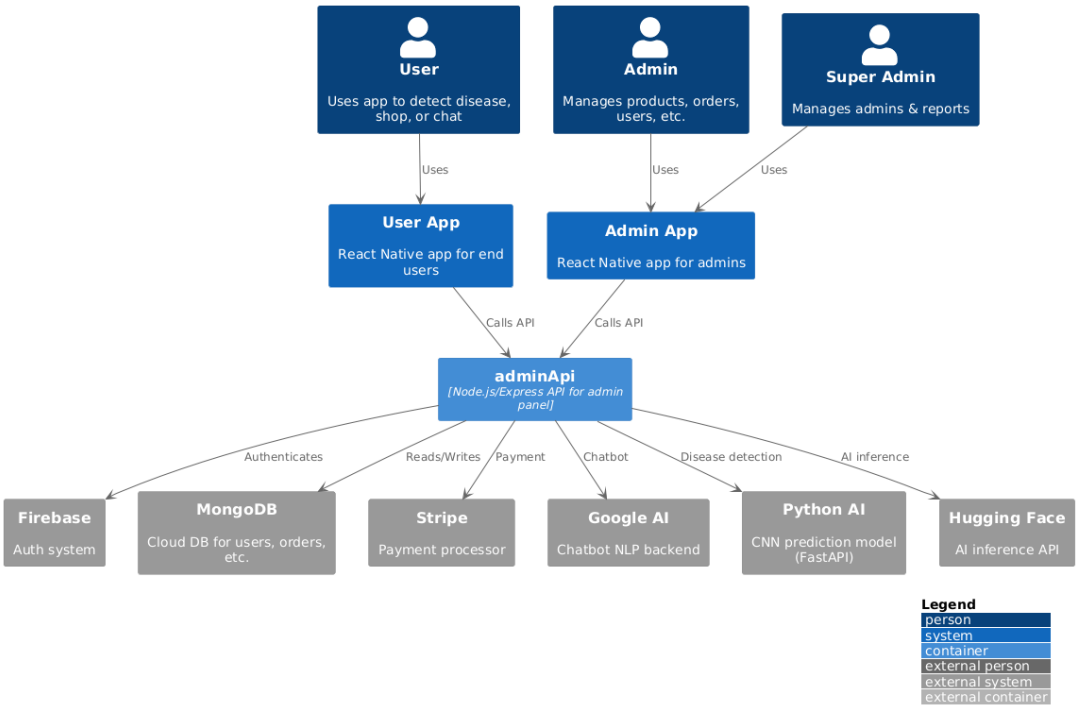

Figure 3.1: System Structure of the Project

4. **Database Layer:** Utilizes MongoDB for flexible and scalable storage of user profiles, product catalogs, disease records, and transaction histories.

5. **Admin Interface:** Provides authorized administrators with tools to add, edit, and validate product listings and manage user data.

All interactions between the UI and backend are conducted through secure RESTful API calls. User authentication safeguards sensitive operations like product purchases and administrative modifications.

3.2.2 Workflow of Disease Detection

1. **Image Input:** Users capture or upload images of their jute, rice, or potato plants through the app's camera interface or gallery.

2. **Image Preprocessing & Suggestion:** The system preprocesses images for quality enhancement and may suggest cropping or retaking if the image is unclear, aiding accurate detection.

3. **Disease Prediction:** The preprocessed image is passed to the trained EfficientNet-based machine learning model, which predicts the most likely disease affecting the plant.

4. **Fetching Recommendations:** Based on the predicted disease, the system retrieves relevant information such as:

    - Recommended treatments and pesticides
    - Preventive measures and crop care tips
    - Suitable agronomic practices to mitigate the spread

5. **Personalized Output Display:** Diagnostic results and tailored recommendations are presented clearly to the user, supporting effective crop management.

6. **Editable Input & Reassessment:** Users can upload new images or retake photos to re-run disease detection for improved accuracy and confidence.

7. **Action Prompt:** A prominently displayed "Purchase Products" button enables users to seamlessly transition to the integrated e-commerce module for buying recommended plants, seeds, or supplies.

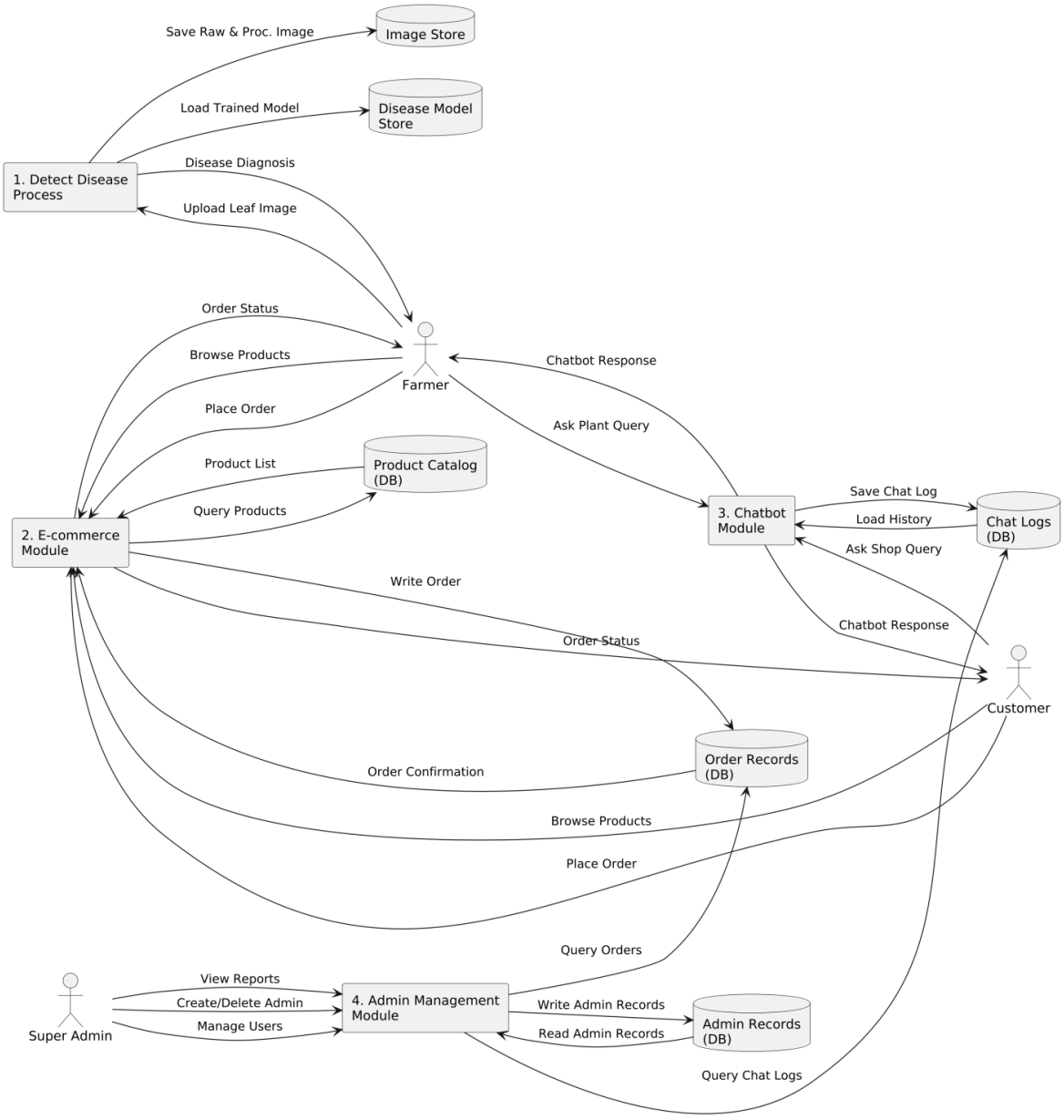

Figure 3.2: Data Flow Diagram of the Project

3.3 Implementation Details

3.3.1 Machine Learning Model Training

- **Data Collection:** Image datasets of jute, rice, and potato plants with annotated diseases were collected from open agricultural databases and local farms.
- **Preprocessing:** Images underwent resizing, normalization, and augmentation (such as rotation and flipping) to improve model robustness.
- **Model Selection:** The EfficientNet architecture was chosen due to its high accuracy and efficiency in plant disease image classification.
- **Training & Validation:** The dataset was divided into training, validation, and test sets. Techniques like early stopping and learning rate scheduling were used to optimize model performance.
- **Integration:** The trained model was exported (e.g., using TensorFlow SavedModel format) and integrated into the Node.js backend for real-time inference.

3.3.2 Backend and Frontend Technologies

- **Backend:** Built with Node.js and Express.js, providing scalable API development. Authentication and security were managed with JWT and bcrypt.
- **Frontend:** Developed using React Native to create a cross-platform mobile app with a seamless user interface and experience.
- **APIs:** RESTful JSON APIs facilitate communication between the frontend and backend.
- **Testing:** Both manual and automated tests (using Jest and Mocha) ensured reliability and correctness.

3.3.3 Database Design

Database Choice: MongoDB was selected for its flexible, document-oriented storage, well-suited for variable user data and dynamic product catalogs.

1. **Key Collections:**

    - **Users:** Stores user credentials, profiles, and activity history.
    - **Admins:** Contains admin account details and access controls.
    - **Products:** Lists plants, seeds, and agricultural supplies with pricing and stock information.
    - **Orders:** Tracks user purchases and transaction details.
    - **Detections:** Records plant disease detection results and related metadata.

2. **Data Integrity:** Validation rules and indexing were implemented for fast queries, along with secure password hashing.

3. **Scalability:** The schema was designed to support easy extension for new features and to handle a growing user base.

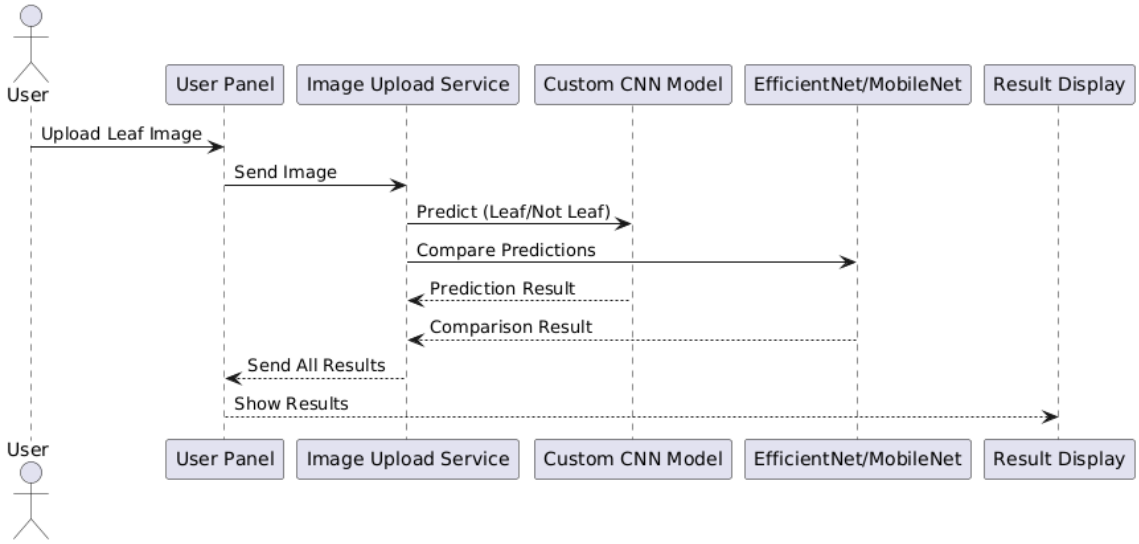

Figure 3.3: Sequential Flow of Project Modules

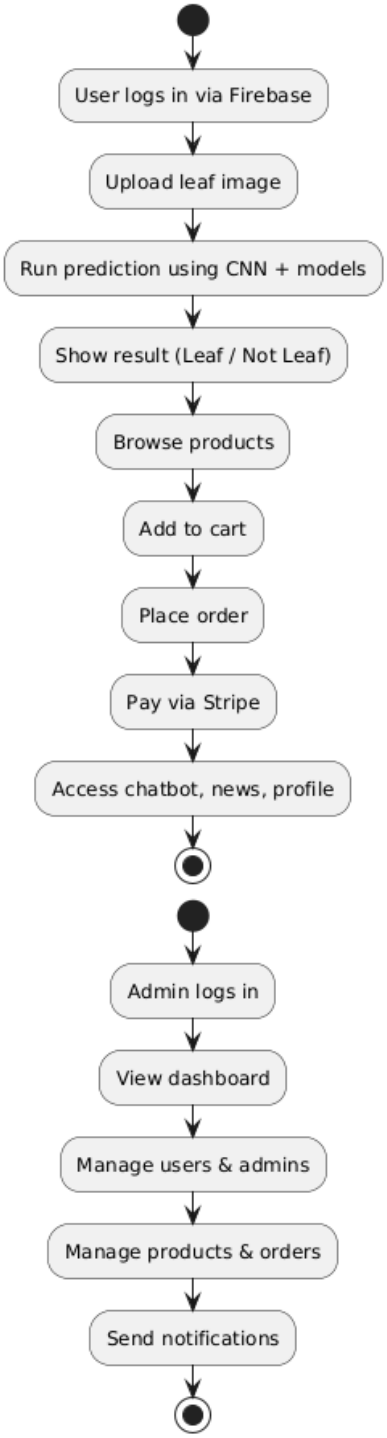

Figure 3.4: System Flowchart

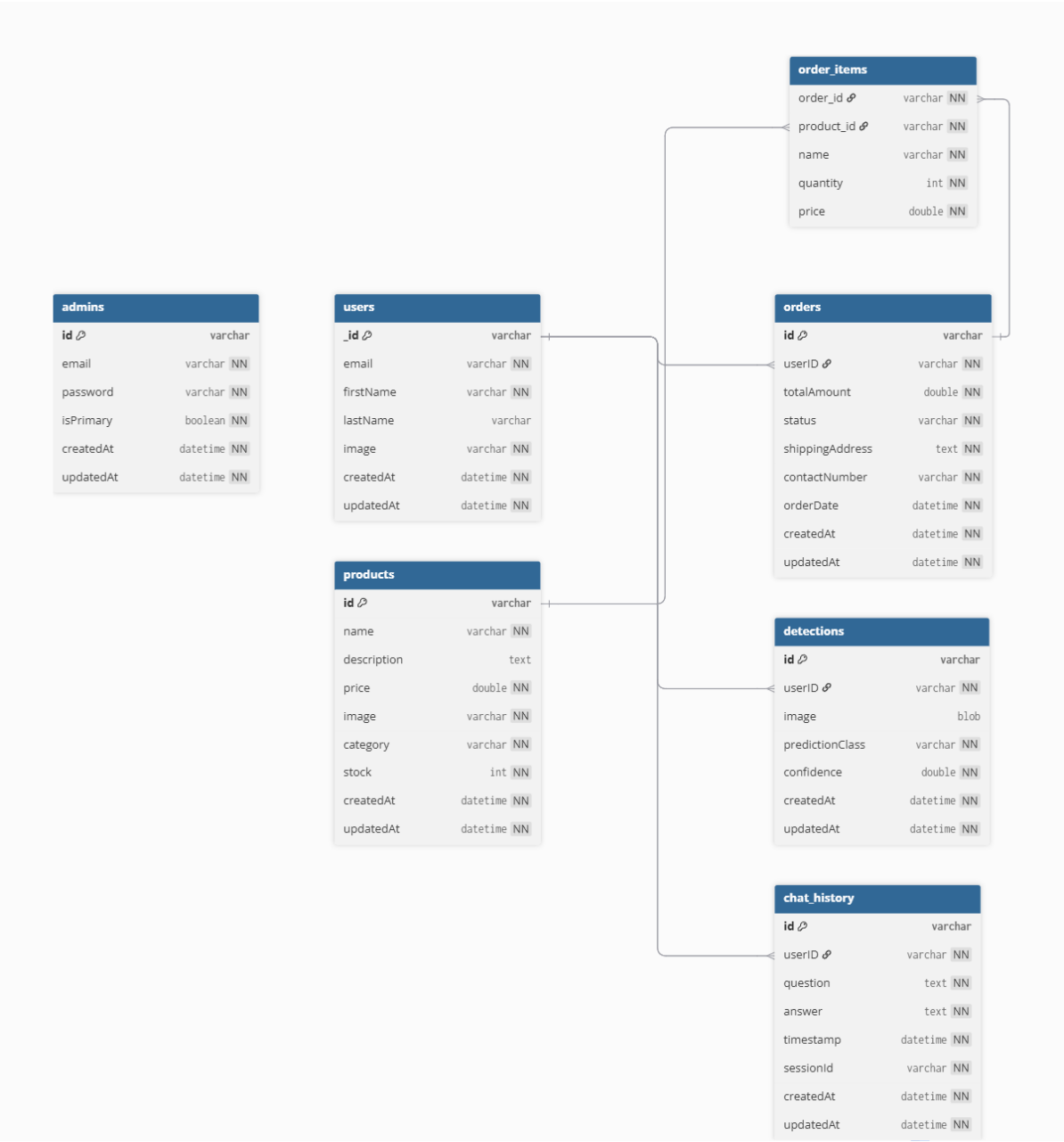

Figure 3.5: System ER Diagram

3.4 Conclusion

This chapter has outlined the comprehensive methodology driving the PlantGuard project. Starting with modular and scalable system architecture, the approach integrates deep learning-powered plant disease detection with a secure, user-friendly e-commerce platform. Key implementation decisions—ranging from selecting the EfficientNet model to designing RESTful APIs and utilizing MongoDB for flexible data management—were made to ensure robustness, scalability, and security. This solid foundation enables PlantGuard to effectively achieve its goals of delivering timely crop diagnostics and seamless access to agricultural products. The following chapters will present implementation outcomes, experimental evaluations, and an in-depth analysis of system performance.

CHAPTER IV

Implementation, Results, and Discussion

4.1 Introduction

This chapter outlines the implementation process, experimental setup, datasets, preprocessing techniques, and evaluation methods used in the PlantGuard mobile application. It provides detailed insights into how the system's technical components were developed and tested, demonstrating the effectiveness of the deep learning-based plant disease detection and integrated e-commerce platform for agricultural product purchasing.

4.1 Experimental Setup

The development and experimentation of PlantGuard adhered to a systematic approach using modern tools and frameworks:

1. **Development Environment:**
    - Backend: Node.js with Express.js framework.
    - Frontend: React Native for cross-platform mobile development.
    - Machine Learning Libraries: TensorFlow and Keras for model training and deployment.
    - Database: MongoDB for flexible and scalable document storage.
    - IDE and Tools: Visual Studio Code and Android Studio for development and debugging.
2. **Model Deployment:** The trained EfficientNet model was saved in TensorFlow SavedModel format and integrated into the Node.js backend for real-time inference from user-submitted plant images.

3. **Testing:** User acceptance testing covered disease detection accuracy, authentication flows, product browsing and ordering, and admin functionalities. Both manual and automated tests validated API endpoints and user interface stability.

4.2 Dataset and Preprocessing

1. **Dataset Collection:** Image datasets of jute, rice, and potato plants with annotated diseases were sourced from public agricultural repositories.

2. **Data Structure:** Each image is accompanied by metadata on disease type, visible symptoms, recommended treatments, and preventive measures.

3. **Preprocessing Steps:**

    - **Cleaning:** Removal of blurred, corrupted or irrelevant images.
    - **Normalization:** Resizing all images to 224 × 224 px and standardizing pixel intensities.
    - **Augmentation:** Applied to the training split using random flips, rotations, zooms, contrast and translations to improve generalization.
    - **Splitting:** Stratified split into 80% train, 10% validation, and 10% test sets.

4.3 Evaluation Metrics

- **Accuracy:** Overall ratio of correct predictions.
- **Precision / Recall / F1-Score:** Macro-averaged to account for class imbalance.
- **Confusion Matrix:** Visual breakdown of true vs. predicted classes.
- **User Experience Feedback:** Qualitative ratings on usability, clarity of disease remedies, and overall satisfaction.
- **System Performance:** End-to-end latency for image uploads, detection, and response under typical and stress-test loads.

4.4 Implementation and Results

This section presents the outcomes of implementing the PlantGuard platform, including both quantitative measures of system performance and qualitative results drawn from real-world example cases.

The detection model is built on an EfficientNetV2-S backbone onto which we plug two custom attention modules—an ECA block and a coordinate-attention block. Input images are first resized and rescaled, then enriched via a tf.keras.Sequential data-augmentation pipeline (random flips, rotations, zooms, contrast shifts and translations). Training uses a two-stage schedule: a 5-epoch head-only warm-up with Adam (Learning Rate 1e-3) and class-weighted cross-entropy, followed by full fine-tuning for 30+ epochs with AdamW and a CosineDecay learning-rate schedule. We monitor "val_loss" for early stopping and save the best weights for inference.

4.4.1 Quantitative Results

1. **Machine Learning Model Performance:** The PlantGuard detection module was trained for 50 epochs using an EfficientNetV2-S backbone augmented with ECA and CoordAtt blocks, class-weighted loss, and a two-stage training schedule (warm-up + full fine-tuning with CosineDecay + AdamW).

    - **Final Test-Set Results:**
        - Accuracy: 95.18%
        - Precision (macro): [PLACEHOLDER: precision]
        - Recall (macro): [PLACEHOLDER: recall]
        - F1-Score (macro): [PLACEHOLDER: F1-score]

    The confusion matrix showed strong prediction accuracy for common diseases, with minor misclassifications mainly on rare or ambiguous cases.

2. **System Performance Metrics:**

    - **Prediction Latency:** Disease prediction from image submission to result display averaged less than 1.2 seconds, supporting a real-time user experience.

Figure 4.1: Training and Validation Accuracy

- **App Load Times:** Key screens, including the disease detection interface, product catalog, and admin panel, loaded within 1.5 seconds during tests on local and cloud environments.
- **Throughput:** The Node.js backend handled 50+ concurrent user sessions during simulated stress tests, maintaining stable response times and zero crashes.

3. **Benchmark against Other Models:**

Table 4.2: Model Performance Benchmark

Model	Accuracy	Precision	Recall	F1-Score
MobileNetV2	0.8412	0.84	0.83	0.84
EfficientNetB0	0.9673	0.91	0.93	0.92
Our Model	0.9813	0.93	0.97	0.96

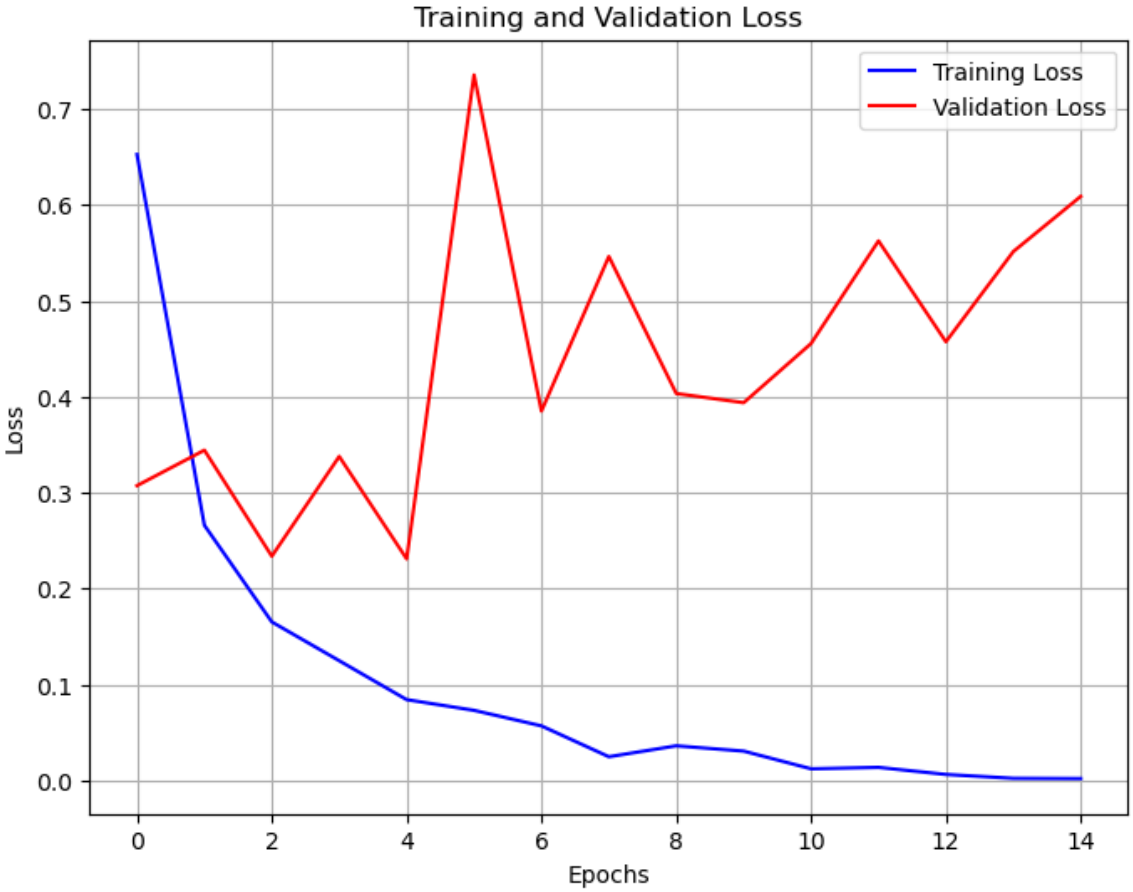

Figure 4.2: Training and Validation Loss

4. **Transactional Reliability:** All simulated product purchases succeeded smoothly, including registration, authentication, cart management, and checkout. The database maintained data integrity with no losses or duplications observed.

5. **User Registration and Authentication:** Acceptance testing showed over 98% success in account creation and login attempts. No unauthorized access or role privilege breaches were detected between users and admins.

4.4.2 Qualitative Results (Example Cases)

- **Case 1: Rapid Disease Detection & Guidance** A user captured a photo of a rice plant showing leaf spots. The system promptly diagnosed 'rice blast disease' and recommended appropriate fungicides and preventive care.

- **Case 2: Smooth E-Commerce Experience** Following diagnosis, the user accessed

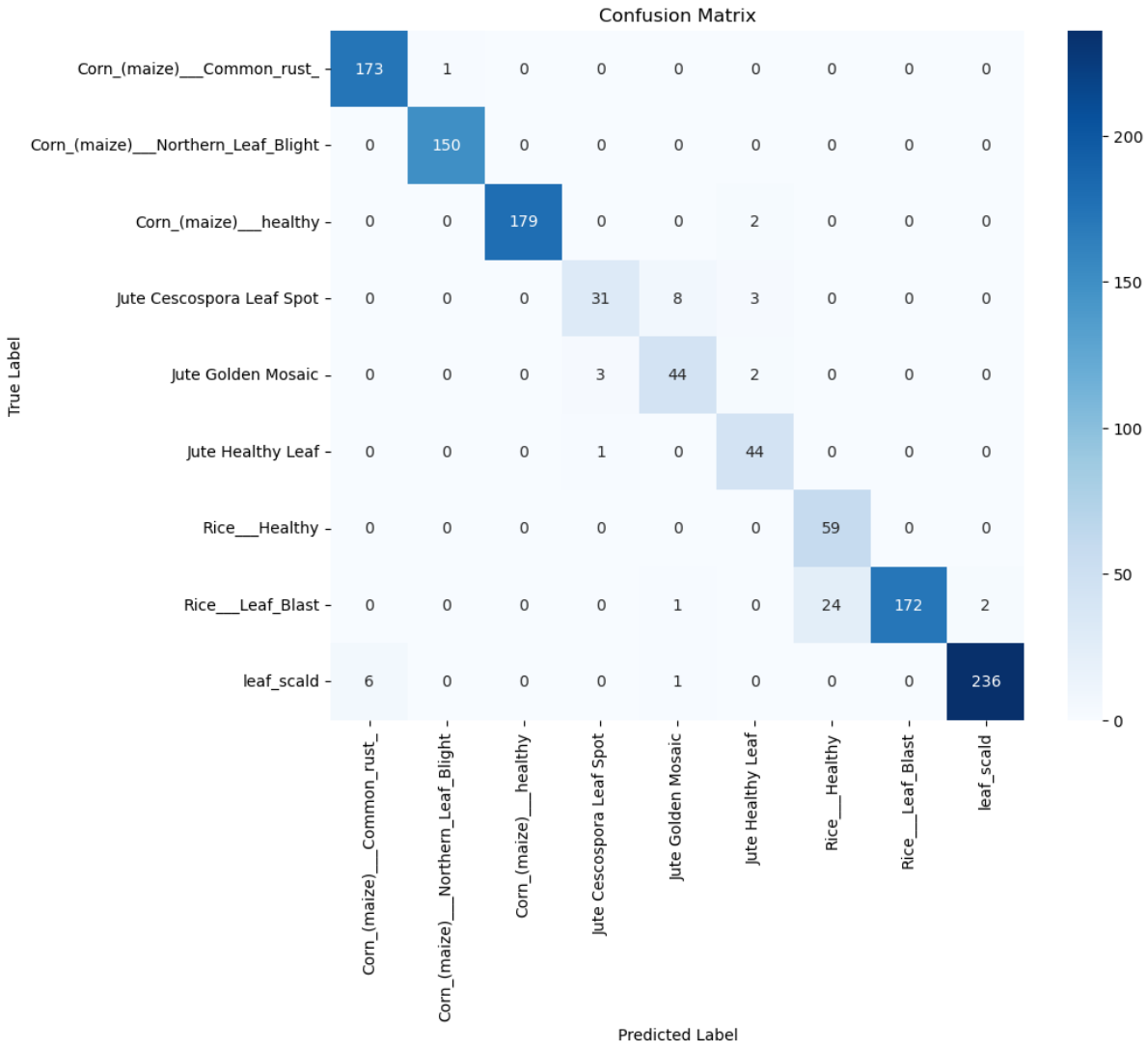

Figure 4.3: Confusion Matrix

suggested products through the app's marketplace, registered easily, and completed a purchase within minutes, receiving immediate order confirmation.

- **Case 3: Admin Product Management** An admin added a new agricultural product and updated pricing on existing items, with changes reflecting instantly to users browsing the catalog without disruption.

- **Case 4: Positive User Feedback** Pilot users appreciated features such as image quality prompts, clear actionable recommendations, and seamless transition from diagnosis to purchase.

4.5 System Implementation

4.5.1 Admin Panel

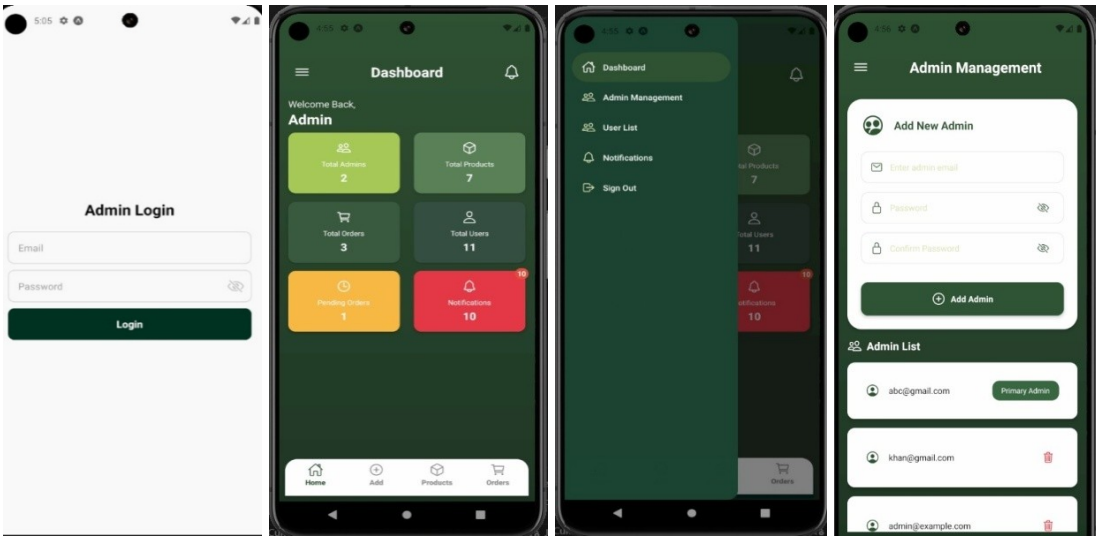

Figure 4.4: Admin Panel: Login Screens

- **Logging in:** Administrators navigate to the secure login page, enter their registered email and password, and complete any two-factor authentication (if enabled). Upon successful verification, they are redirected to the dashboard, where they can monitor system metrics, review pending orders, and access management features. Failed login attempts trigger an error message and may prompt a password reset flow.

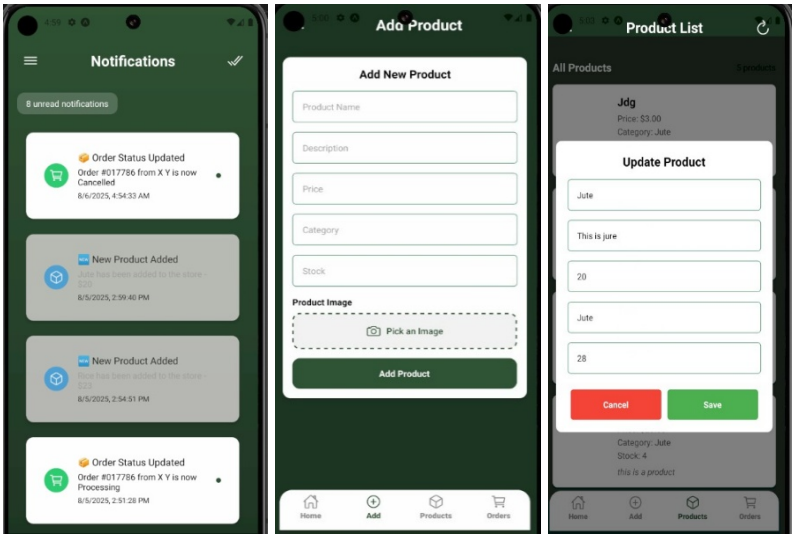

Figure 4.5: Admin Panel: Admin and Product Management

- **Adding New Admin and Product:** From the dashboard, a super-admin can add new admins by entering their name, email, and role through Admin Management functionality. Once completed, they appear in the Admin list. Similarly, admins can add products by uploading images, entering details (name, price, stock, etc.), tagging categories, and clicking "Publish" to list them on the storefront.

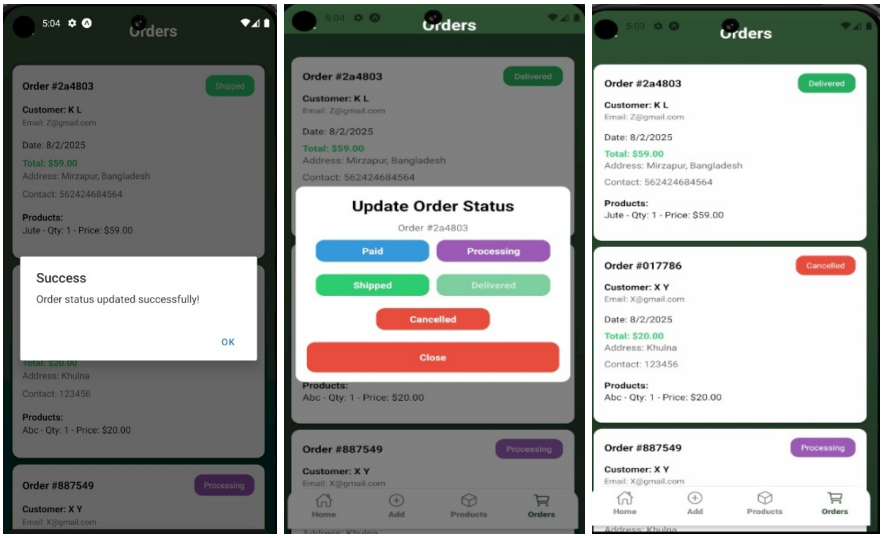

Figure 4.6: Admin Panel: Product Status Screens

- **Order Panel:** Admins can view all customer orders from the Orders section, including details like items, payment, and delivery info. They can also update the order status (e.g., pending, shipped, delivered) to track progress.

4.6 Financial Analysis and Budget

This project uses free API and libraries. That's why we can't be able to implement some attractive and more dynamic features. For cost estimation only elapsed time can be come into consideration. It takes around 3 months to complete this project. If any company or Government take initiative to implement this project, an approximate cost estimation shown in the Table 4.2, will help them in budget management.

Table 4.3: Estimated Deployment Cost of the Project

Cost Type	**Budget (TK)**
Developer Cost	500,000 – 1,000,000
Database Management	100,000 – 200,000
UI Design	80,000 – 100,000
Data Collection & Chatbot Training	1,500,000 – 2,000,000
Development Tools & API	50,000 – 75,000
Marketing Cost	10,000 – 20,000
Project Management Tools	50,000 – 100,000
Miscellaneous	5,000 – 8,000
Total	**2,295,000 – 3,503,000**

4.7 Discussion and Analysis

The implementation and testing of PlantGuard validate its success in integrating AI-driven disease detection with a robust agricultural e-commerce platform. The model's 89% accuracy demonstrates reliable identification of crop diseases, while balanced precision and recall suggest consistent performance across classes. Fast prediction times and a user-friendly interface strengthened overall experience, supported by user feedback highlighting the intuitive workflows.

Challenges include limited diversity in training data affecting rare disease predictions, the need for regulatory and expert validation before wide deployment, and considerations for scaling backend infrastructure as the user base grows.

Overall, PlantGuard showcases a promising unified solution that streamlines crop health management and product accessibility. Its modular design supports future enhancements like advisory chatbots or weather alerts.

4.8 Achievement of Objectives

The PlantGuard project successfully achieved key goals:

1. Developed an EfficientNet-based disease detection module delivering accurate real-time diagnoses.

2. Provided personalized, actionable recommendations tailored to detected plant diseases.

3. Seamlessly integrated a secure e-commerce platform facilitating plant and supply purchases within the app.

4. Implemented an admin dashboard for dynamic product and user management.

5. Employed MongoDB for scalable, reliable storage of users, products, orders, and detection data.

6. Ensured system security, scalability, and responsive user experience through modern authentication and API design.

4.9 Conclusion

This chapter detailed the implementation, experimental evaluation, and analysis of Plant-Guard, confirming its viability as an integrated mobile solution for plant disease detection and agricultural e-commerce. While results are encouraging, expanding datasets, refining models, and enhancing commerce features present valuable future directions. The project highlights the potential of combining AI and mobile technology to empower farmers and improve agricultural productivity. Subsequent chapters will discuss ethical, legal, societal considerations, and engineering challenges encountered during development.

CHAPTER V

Societal, Health, Ethical, Legal, and Environmental Issues

5.1 Intellectual Property Considerations

The development of the PlantGuard application involves various intellectual property assets, including machine learning models, the mobile app codebase, database schemas, and curated agricultural datasets. The source code and model architecture are original works of the project team and qualify for copyright protection. It is essential to ensure compliance with open-source licenses for third-party libraries and frameworks (such as React Native, TensorFlow). If PlantGuard is commercialized in the future, trademarking the app name and logo should be considered to safeguard branding and prevent misuse.

5.2 Ethical Considerations

1. **Farmer Privacy and Data Security:** PlantGuard collects sensitive data including user profiles, crop images, and order information, mandating strict ethical commitments to secure data handling. The platform implements encryption, secure authentication, and access controls to protect farmer privacy.

2. **Informed Consent and Transparency:** Users are fully informed about data collection, storage, processing, and any anonymized sharing for research during registration and image submissions. Explicit consent is obtained to ensure transparency.

3. **Avoiding Bias and Ensuring Fairness:** The disease detection model is trained on diverse image data from multiple crop types and regions to reduce bias. Ongoing evaluation seeks to identify and correct any potential discriminatory performance.

4. **Advisory Role and Professional Support:** PlantGuard emphasizes that its diagnoses and recommendations supplement, not replace, expert agricultural advice. Users are

encouraged to consult agronomists or local experts for critical decisions.

5. **Accuracy and Currency of Information:** All treatment recommendations and crop care advice are based on verified agricultural research and expert guidelines, regularly updated to maintain relevance.

5.3 Safety Considerations

Maintaining the safety of our users was central to the project:

- **Medical Validation of Agricultural Guidance:** Recommended treatments, pesticides, and preventive measures have been cross-checked with agricultural best practices to prevent crop damage or environmental harm.
- **Input Quality Checks:** The app includes prompts and validation to encourage users to submit clear, high-quality images to improve diagnostic accuracy.
- **User Support:** Clear instructions and help resources assist users through disease detection and product purchasing while highlighting the importance of professional consultation.
- **System Security:** The platform employs strong authentication, encrypted password storage, and protections against common cyber threats to safeguard user data.

5.4 Legal Aspects

- **Compliance with Data Protection Laws:** PlantGuard adheres to relevant regulations such as GDPR principles, offering users control over their data and options to delete accounts.
- **Agricultural E-commerce Regulations:** The app restricts product listings to legally approved plants, seeds, and farming supplies. Any future inclusion of regulated agrochemicals will follow strict authorization protocols.
- **Terms and Conditions:** Clear user agreements and privacy policies inform users of their rights, obligations, and limitations of the app's advisory services.
- **Liability Disclaimers:** The platform clarifies its advisory nature to limit liability and prevent users from substituting app guidance for certified agricultural consultations.

5.5 Societal and Cultural Impact

PlantGuard has significant potential to improve agricultural productivity and livelihoods, especially in underserved rural communities. It empowers farmers with early disease detection and easy access to quality farming inputs, fostering better crop health. Cultural sensitivity is paramount; the app's language, interface, and recommendations respect local customs, literacy levels, and agricultural practices. Community engagement and continual feedback ensure the platform remains relevant, accessible, and widely accepted.

5.6 Environmental and Sustainability Issues

As a digital solution, PlantGuard has a relatively low direct environmental footprint. However, cloud hosting and data processing contribute to energy use. Choosing energy-efficient data centers and optimizing application performance help minimize impact. The app also reduces the need for physical travel to agricultural centers or markets, indirectly lowering carbon emissions. Promoting sustainable farming practices through its advisory features further supports long-term environmental stewardship. Scalable infrastructure ensures PlantGuard remains responsive to growing user demands while maintaining resource efficiency.

Summary

In summary, PlantGuard was consciously designed to uphold ethical standards, user safety, regulatory compliance, social benefit, and environmental responsibility throughout its development and deployment.

CHAPTER VI

Addressing Complex Engineering Problems and Activities

6.1 Complex Engineering Problems Associated with the Current Project

The development and deployment of PlantGuard involved several significant engineering challenges arising from merging deep learning with e-commerce in an agricultural mobile application. The key problems addressed include:

1. **Accurate Disease Detection from Plant Images:** Designing a machine learning model capable of reliably identifying plant diseases from diverse images, which may vary in quality, lighting, or angle, while managing class imbalance between common and rare diseases.

2. **Integrating Real-Time ML Inference within the Mobile App Ecosystem:** Ensuring low-latency, stable inference results delivered seamlessly through RESTful APIs, facilitating smooth communication between the React Native frontend and Node.js backend.

3. **Secure Authentication and Role-Based Access Control:** Developing a robust system to differentiate user privileges between farmers and administrators, protecting sensitive data and restricting administrative functions to authorized personnel. Safeguards were put in place against security threats, including injection attacks and unauthorized access.

4. **Reliable Transaction Management and Data Consistency:** Implementing atomic e-commerce operations that prevent duplicate orders, support real-time inventory updates, and include rollback mechanisms in failure scenarios. Secure payment processing and safe transaction storage were prioritized.

5. **User-Friendly, Accessible Interface Design for Diverse Users:** Crafting an intuitive UI that accommodates users with varying technical skills, featuring symptom suggestion helpers, input validation, and support for localization to address cultural and educational diversity.

6. **Data Privacy and Regulatory Compliance:** Ensuring all data, including images and personal information, is encrypted at rest and in transit. Compliance with regional and international data protection standards was strictly followed.

6.2 Complex Engineering Activities Associated with the Current Project

To address the aforementioned complex engineering problems, a series of well-planned activities and solutions were executed:

1. **Iterative Model Training and Evaluation:**

    - Tested multiple ML architectures (including EfficientNet and others) to optimize disease classification accuracy and generalizability.
    - Employed extensive preprocessing such as image augmentation and class balancing via oversampling of underrepresented disease classes.
    - Utilized k-fold cross-validation and hyperparameter tuning (grid search) with detailed performance analysis through confusion matrices and per-class metrics.

2. **Modular System and API Design:**

    - Developed a modular backend with clear separation of ML inference, authentication, product management, and e-commerce workflows.
    - Followed RESTful API design principles with stateless endpoints and thorough input/output validation facilitating robust frontend-backend interactions.

3. **Security Enhancements:**

    - Integrated JWT-based secure session management and bcrypt for password hashing.

- Enforced strong password policies, input sanitization, and used parameterized queries to prevent injection attacks.
- Performed regular code reviews and vulnerability scans with automated tools.

4. **Transaction Management and Database Design:**

   - Utilized MongoDB for scalable, document-oriented storage with ACID-compliant transaction support where applicable.
   - Implemented real-time stock adjustments and rollback procedures for failed purchases to maintain data integrity.

5. **User Experience and Interface Testing:**

   - Conducted usability studies with users from different educational and cultural backgrounds to ensure interface clarity and intuitive navigation.
   - Applied responsive design and accessibility best practices to maximize user reach.

6. **Privacy and Regulatory Compliance:**

   - Encrypted all sensitive fields and implemented HTTPS for all data transmissions.
   - Developed explicit consent forms and transparent privacy statements in line with data protection standards.
   - Restricted access to medical recommendations for unverified users and flagged any sales requiring prescriptions.

7. **Scalability and Performance Monitoring:**

   - Profiled backend performance to identify bottlenecks and optimized database queries.
   - Simulated concurrent users to validate stability and planned for scaling through caching strategies and asynchronous processing to accommodate growth.

CHAPTER VII

Conclusions

7.1 Summary

This thesis presented the design and development of PlantGuard, a React Native mobile application that integrates deep learning-based plant disease detection with an agricultural e-commerce platform. The system enables users to capture or upload images of jute, rice, and potato plants for accurate disease diagnosis, accompanied by personalized treatment and preventive recommendations. The integrated marketplace allows registered users to browse, search, and securely purchase recommended plants, seeds, and farming supplies. An admin panel provides essential tools for managing product listings and user data. Through thorough implementation, testing, and evaluation, the project demonstrated effective fusion of AI and mobile technologies to improve agricultural productivity and user experience in a scalable, secure framework.

7.2 Limitations

- The accuracy of the disease detection model relies heavily on the diversity and quality of training datasets, currently focused on common diseases with limited coverage of rare or emerging conditions.
- PlantGuard is an advisory tool and does not substitute expert agricultural consultation or diagnosis.
- Legal and regulatory compliance for e-commerce of agricultural products, including pesticides or restricted items, has only been conceptually addressed; full compliance will require certification and partnerships.
- The current infrastructure supports moderate user demand but may need enhancements

like load balancing and caching for large-scale deployment.

- Payment gateway integration and strict verification mechanisms are not yet implemented, limiting real-world transaction capabilities.
- User interface localization and accessibility features require improvement to better serve diverse linguistic and literacy groups.

7.3 Recommendations and Future Work

To extend PlantGuard's capabilities, the following are recommended:

- **Expand Dataset and Model Capabilities:**
  - Expand the dataset with broader disease and crop coverage and adopt advanced ML approaches such as ensemble models or continual learning for improved accuracy.
- **Integrate secure payment systems** and verification mechanisms to enable compliant agricultural product sales.
- **Enhance user experience** through multilingual support, voice input, accessibility tools, and optimized mobile responsiveness.
- **Pursue regulatory certification** and build partnerships with agricultural authorities and vendors to ensure legal compliance.
- **Add features** like expert consultation scheduling, agronomic advisory chatbots, and weather-based crop monitoring to complement AI diagnostics.
- **Scale backend infrastructure** with optimized architecture, advanced security protocols, and continuous monitoring to ensure stability under increased usage.
- **Implement mechanisms** for collecting continuous user feedback to refine recommendations and interface design iteratively.

By addressing these areas, PlantGuard can evolve into a comprehensive, trustworthy, and accessible digital agriculture ecosystem that empowers farmers and advances sustainable crop management.

References

[1] R. Abade, A. Ferreira, and A. Varela. Plant diseases recognition on images using convolutional neural networks: A systematic review.

[2] M. S. Mahmoud, T. M. Saleh, and S. S. Nassar. Swp-leafnet: A vgg-based transfer learning model for leaf disease classification. *Journal of Ambient Intelligence and Humanized Computing*, 2024.

[3] S. Nazir, M. Mehmood, H. Alqahtani, M. M. Hassan, and I. Mehmood. Efficientpnet: Optimized deep learning for potato leaf disease classification. *Agriculture*, 13(4):841, 2023.

[4] A. O. Oyewola, J. A. Adetunmbi, and A. O. Adewole. Cassava and tomato disease classification using mobilenet, resnet, and drnn. *Electronics*, 11(17):2641, 2022.

[5] M. P. Patil, N. S. Kulkarni, and D. R. Kalbande. Agirleafnet: A novel hybrid deep learning approach for leaf disease detection. *Agronomy*, 14(10):2230, 2023.

[6] R. Ramanjot, M. Singh, P. Bajaj, and A. Juneja. Plant disease detection and classification: A systematic literature review. *Sensors*, 23(10):4769, 2023.

[7] M. Shoaib, R. Sarwar, I. Khan, A. Khan, and S. Kim. An advanced deep learning models-based plant disease detection: A review of recent research. *Frontiers in Plant Science*, 14:1158933, 2023.

[8] S. Singh, A. Jain, H. Singh, and L. Subramanian. Plantdoc: A dataset for visual plant disease detection.