2025

GreenNest

Detecting Leaf diseases using Deep Learning

Future Academy - Higher Future Institute for Specialized Technological Studies
School of Computer Science

Project report submitted to the Future Academy
for the degree of Bachelor of Computer Science

May 2025

Table of Content

Table of Figures

Chapter 1

Project Introduction and Overview

- Abstract
- Problem Definition
- Current Solutions
- Proposed Solution
- Research Criteria

Abstract

The agricultural sector, particularly plants, plays a vital role as it constitutes the majority of world industries and serves as the main food resource for approximately 8 billion people worldwide. Therefore, this project primarily focuses on contributing to the safe feeding of this vast population. Plant diseases are a significant threat to global food security. Due to this, developing a smart system that aids in disease recognition and assists farmers in enhancing plant productivity is crucial in both present times and the future.

Our solution leverages a deep learning technique called Convolutional Neural Network (CNN), which is the most suitable approach for this project. We apply preprocessing techniques and transfer learning using ResNet-50, building a model to recognize plant diseases with 98.78% accuracy. The solution is integrated with a mobile application as the user interface, built on the Flutter platform. The application takes plant images as input and determines whether the plant is infected or not. The model is trained on 26 disease classes and 14 healthy plant leaf types.

By harnessing advanced deep learning capabilities, GreenNest enables early detection of plant diseases, empowering farmers to mitigate risks and reduce crop losses effectively. The cross-platform mobile application ensures accessibility across diverse regions, delivering a user-friendly experience that requires minimal technical expertise. Real-time disease identification through image analysis allows for swift interventions, enhancing agricultural resilience. With its exceptional accuracy, GreenNest sets a new benchmark for automated disease detection systems. This project not only addresses immediate agricultural challenges but also holds potential for scalability, adapting to emerging diseases and plant types. Ultimately, GreenNest aims to bolster global food security and promote sustainable farming practices for future generations.

Introduction

Plant health is a cornerstone of global food security and sustainable agriculture. As the foundation of food systems and a vital source of non-food commodities—including energy, fiber, and horticultural products—plants face escalating threats from diseases exacerbated by climate change and intensified global trade. Over the past 45 years, an estimated 20–40% of global crop yields have been lost annually to pests and diseases, with staple crops such as wheat, rice, maize, potatoes, and soybeans suffering devastating losses ranging from 17% to over 30%. These losses not only jeopardize food supplies but also destabilize economies, degrade environmental resources, and endanger livelihoods, particularly in agrarian communities.

The rapid spread of pathogens, such as rice leaf smut and rice leaf scald, underscores the vulnerability of crops to large-scale outbreaks. Traditional disease management methods, often reliant on reactive measures, struggle to keep pace with the scale and speed of modern agricultural challenges. Early and accurate detection is critical to mitigating these risks, yet barriers like limited access to advanced diagnostics and delayed response mechanisms persist.

GreenNest emerges as a transformative response to these pressing issues. By integrating cutting-edge technology with user-centric design, this project aims to empower farmers, agronomists, and policymakers with real-time, data-driven tools to identify and combat plant diseases proactively. In doing so, GreenNest seeks to safeguard crop yields, enhance agricultural resilience, and contribute to a sustainable future for global food systems.

Problem Definition

Agriculture forms the backbone of global food security, feeding approximately 8 billion people worldwide. However, crop diseases pose a significant threat to agricultural productivity and food security, leading to several critical challenges:

Economic Impact

Annual crop losses of billions of dollars worldwide

Reduced farmer income and agricultural productivity

Increased food prices affecting consumer accessibility

Crop damage threatening food availability

Reduced yield affecting food supply chains

Risk to global food sustainability

Detection Challenges

Difficulty in early disease identification

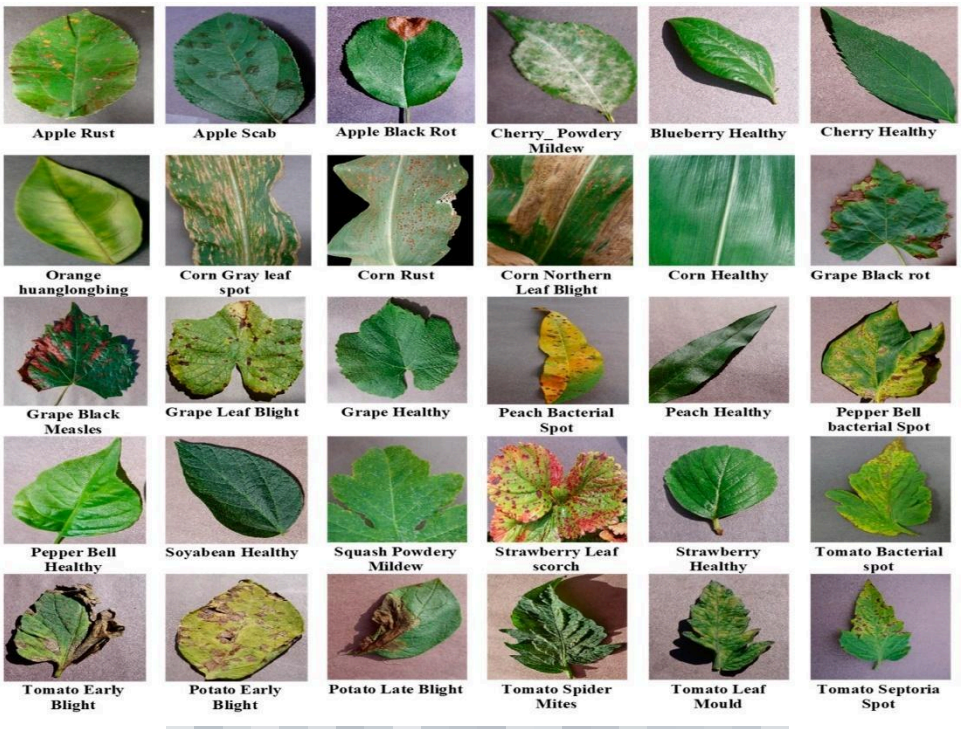
Limited access to agricultural experts

Time-consuming traditional diagnostic methods

Inconsistent diagnosis accuracy by manual inspection

Inefficient treatment of unidentified diseases

Wastage of resources on incorrect treatments



Detection Technologies

This table highlights the diverse range of methods available for plant disease detection, each with its strengths and weaknesses. The choice of method depends on factors like cost, specificity, and the stage of disease detection required.

Plant Disease Detection Technologies Comparison

Technology	Methodology	Advantages	Limitations
CNNs (AlexNet, VGG, Resnet, etc.)	Deep Learning	High accuracy, handles complex images	Requires large datasets, computationally expensive
Traditional ML	Feature Extraction & Classification	Easy to implement, widely used	Limited in detecting subtle symptoms
Cassava Plant Disease Identifier	Computer Vision	Monitors disease severity, improves with user input	Requires user knowledge
RGB Imaging	Optical Technique	Non-destructive, early detection	Limited specificity
Hyperspectral Sensors	Optical Technique	High specificity, accurate detection	Costly, complex data analysis
Manual Analysis	Molecular Methods	Direct detection, high specificity	Often invasive, requires expertise
Biosensors	Bio-recognition Elements	Early detection, selective	Developmental stage, cost

Current AI based Applications (Current Solutions)

In recent years, deep learning has revolutionized the field of agricultural diagnostics, enabling rapid and accurate identification of plant diseases. Below is a comparative summary of key AI-powered projects that have contributed to this domain, highlighting their respective algorithms, performance, and distinguishing features.

Project	Algorithm	Year	Accuracy	Unique Features / What It Does Better
PlantNet DL	CNN	2021	92%	Identifies plants via photos; intuitive UI; widely used for educational purposes
CropScan DL	VGG16	2022	88%	focused on crop disease classification using well-known VGG16 CNN architecture
AgriAI	MobileNet	2023	90%	Lightweight model for mobile deployment; trained on Indian crops; optimized for on-device inference
DiseaseNet	DenseNet121	2024	94%	benefits from DenseNet's deep feature propagation and efficiency in learning complex patterns
AgriVision	EfficientNet	2025	95%	Integrates ML, satellite imagery, IoT, offers real-time precision agriculture insights; boosts sustainability and economic impact

Proposed Solution

The GreenNest project addresses the limitations of existing solutions by integrating a Deep Learning based approach with transfer learning using RESNET-50. This innovative approach achieves an impressive 98.78% accuracy while minimizing computational resource requirements through the use of pre-trained models.

Key advantages of GreenNest include:

- **Enhanced specificity**: Capable of detecting a wide range of diseases across 26 disease classes and 14 healthy plant leaf variants
- **Real-time results**: Provides immediate diagnostic feedback to users
- **User-friendly interface**: Accessible to farmers regardless of technical expertise
- **Non-invasive methodology**: Overcomes the invasiveness of traditional and molecular methods
- **Independence from user knowledge**: Unlike basic Computer Vision tools, doesn't rely heavily on user expertise

These features position GreenNest as a scalable, cost-effective, and innovative solution for plant disease detection, directly contributing to the project's broader goal of enhancing global food security through improved crop health management.

Project Features

Technical Implementation

- Resnet50-based deep learning model for disease classification
- Transfer learning approach for improved accuracy
- Mobile and web applications for widespread accessibility
- Real-time disease detection capabilities

Key Features

- Automated disease detection via Resnet50

- Disease Remediation suggestions via Gemini LLM
- Leaf Health tracking Dashboard
- Cross-platform accessibility (mobile and web)
- User-friendly interface for farmers and agriculturists
- Historical tracking of disease detections
- Plant Species information database

Expected Outcomes

- Disease detection accuracy exceeding 98%
- Reduced time for disease identification
- Accessible to users through multiple platforms
- Decreased crop losses through early detection
- More efficient resource utilization

Unique Advantages

- Real-time results for immediate action
- Scalable to include new disease types
- Wide library of plant information sources and resources

Summary

- The GreenNest platform combines cutting-edge AI technology with practical usability features to create a comprehensive solution for plant disease management.
- By leveraging deep learning capabilities alongside user-friendly interfaces and actionable recommendations, the system empowers farmers and agricultural professionals to quickly identify, track, and address plant health issues, ultimately contributing to improved crop yields and sustainable farming practices worldwide.

Research criteria

1. Previous research on:
    a. Plant disease detection using machine learning/deep learning
    b. CNN applications in agriculture
    c. Resnet50 transfer learning implementations
    d. Mobile-based plant disease detection systems
    e. Similar Flutter applications in agriculture

2. Comparative studies of:
    a. Different CNN architectures for plant disease detection
    b. Various preprocessing techniques
    c. Alternative mobile development frameworks
    d. Different transfer learning approaches
    e. Accuracy rates of existing solutions
3. Papers discussing:
    a. Agricultural challenges in disease detection
    b. Impact of AI in agriculture, Dataset creation for plant diseases
    c. Mobile technology in farming, Real-world implementation challenges

Chapter 2

System Analysis

- Introduction
- Requirement Analysis
- User stories
- Context Diagrams
- Use Case Diagram
- Sequence Diagram
- ERD Diagram
- Class Diagram
- System Architecture
- Technologies and Tools
- Design Patterns

Chapter Introduction

In this chapter, we delve into the foundational aspects of the system, exploring its various dimensions and how it interacts with its users. The primary objective is to identify and understand the different user groups who will interact with the system, as well as the stakeholders who have a vested interest in its success. By clearly defining these roles, we can tailor the system to meet the specific needs and expectations of all parties involved.

To further enhance our understanding, we will employ a series of visual tools, including diagrams and design models. These visual aids are instrumental in summarizing the key components of the system, breaking it down into manageable sub-systems and functions. This decomposition not only facilitates a clearer understanding of the system's architecture but also ensures that the implementation process is smooth and free from unnecessary obstacles.

Our approach to project management is rooted in the Agile methodology, which allows us to perform project tasks concurrently. This iterative and incremental framework enables us to adapt quickly to changes, ensuring that we can respond effectively to new requirements or challenges as they arise. By selecting the most suitable frameworks and tools, we aim to implement the project in the most efficient and effective manner possible.

In summary, this chapter sets the stage for a comprehensive analysis of the system, focusing on user and stakeholder identification, system visualization, and the adoption of Agile methodologies to ensure a successful project outcome. Through this structured approach, we aim to create a system that is not only functional but also aligned with the needs and expectations of its users and stakeholders.

Requirements

- Through surveys conducted with farmers and by collecting essential information about plants from specialized websites.
- We analyzed the available requirements to choose suitable implementation steps that satisfy user needs across both mobile and web platforms.

Hardware Requirement

1. A mobile phone device/ Device with access to the internet
2. Internet connection
3. Plant leaf image (from the supported range)

Functional Requirements

User Requirement:

1. Users must be able to sign up and log in through both mobile app and web browser
2. Users must be able to modify their information and password on both platforms
3. Users must be able to browse the plant details tab and access plant information
4. Users must be able to capture plant photos for disease detection (mobile app)
5. Users must be able to upload photos from gallery/computer for disease detection (both platforms)
6. Users must be able to view their disease detection history across all platforms

System Requirement:

1. The system must process plant images for disease detection
2. The system must maintain stable internet connectivity
3. The system must securely access mobile phone camera (mobile app)
4. The system must maintain data consistency across web and mobile platforms
5. The system must provide responsive design for various screen sizes
6. The system must ensure secure data transmission between client and server
7. The system must support cross-platform compatibility for major browsers
8. The system must implement secure session management across devices

Non-Functional Requirements

Performance

The system should handle up to 100 concurrent users without significant degradation in performance.

Response time for user actions (e.g., login, registration, species management) should not exceed 2 seconds under normal load.

Scalability

The architecture should support horizontal scaling to accommodate an increasing number of users and data volume.

The system should be able to handle a 50% increase in user load without requiring major architectural changes.

Availability

The system should have an uptime of 99.9% to ensure that users can access it at any time.

Scheduled maintenance should be communicated to users in advance and occur during off-peak hours.

Security

User data must be encrypted both in transit and at rest to protect sensitive information.

The system should implement role-based access control to restrict access to sensitive functionalities.

Usability

The user interface should be intuitive and easy to navigate, allowing users to complete tasks with minimal training.

The system should provide help documentation and tooltips to assist users in understanding its features.

Maintainability

The codebase should follow industry best practices for coding standards and documentation to facilitate ease of maintenance.

The system should allow for modular updates, enabling new features to be added without impacting existing functionalities.

Compatibility

The system should be compatible with major web browsers (Chrome, Firefox, Safari, Edge) and mobile devices.

The application should support integration with third-party APIs for additional functionalities, such as external species databases.

User Stories

The Greenest app is designed to assist plant enthusiasts, farmers, and researchers in diagnosing plant diseases using AI-based image detection. To ensure accessibility and ease of use, the platform supports two types of users:

- **Guest Users**: Users who can access limited features without creating an account. This allows for a quick and easy way to test the app's core functionality.
- **Registered Users**: Users who sign up for an account, gaining access to personalized features such as saving past scans, receiving treatment recommendations, and participating in community discussions.

User Authentication and Management

1. User Registration:

As a new plant enthusiast, I want to register an account with my email address and a secure password, so that I can start tracking and managing my plant collection.

2. Email Verification:

As a newly registered user, I want to receive a verification email with a link to activate my account, so that I can be sure my account is active and secure and start using all the features.

3. User Login:

As a returning user, I want to log in using my registered email and password, so that I can quickly access my species information and manage my collection.

Species Management and Information

1. Browse All Species:

As a new or returning user, I want to see a list of all species available in the database, so that I can get familiar with the different types of plants that can be added to my collection.

2. Search Species by Name:

As a user browsing species, I want to quickly search for a species by its common name or scientific name, so that I can access information about specific plants.

3. View Detailed Species Information:

As a plant enthusiast/guest, I want to be able to view detailed information about any listed plant species including its common and scientific name, habitat, and care requirements, so that I can learn more about each plant.

4. Add a New Species to Favorites Collection

As a user browsing species, I want to add the details of a new species into my personal collection, including all its details, so that I can keep track of all plants that interest me depending on my needs.

5. Track a plant's Health

As a user maintaining my plants, I want to be able to view the detailed disease history and monitor my plant's health status, so that I can get actionable remediation steps for my plant.

Context Level 0 Diagram

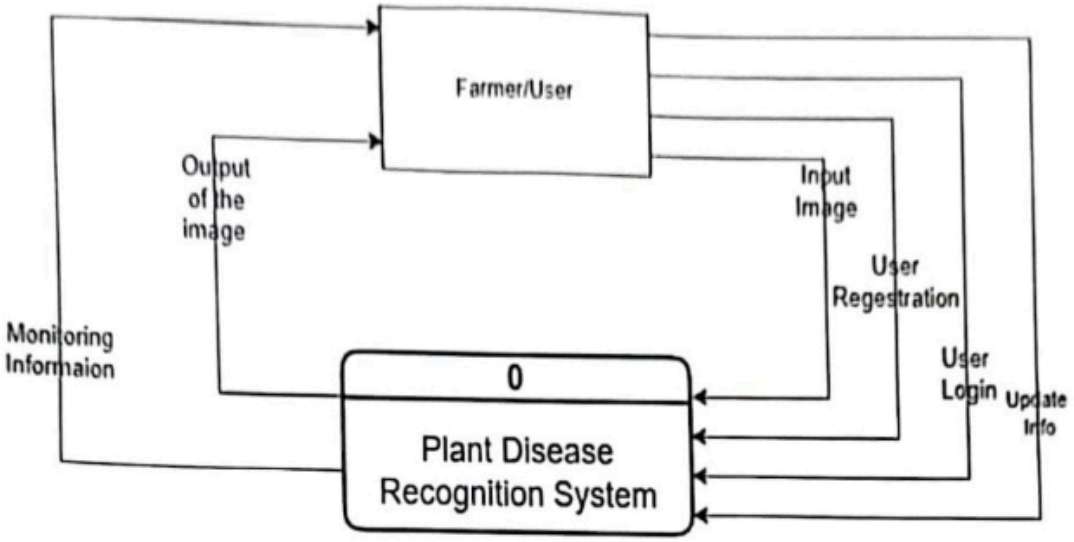


Figure 2 level 0 context diagram

The Context level 0 diagram provides a high-level overview of the GreenNest System and its interaction with the Farmer/User.

Key Interactions:

- Input Image: The user uploads a plant image for disease detection.
- User Registration & Login: Users create accounts and log in to access the system.
- Update Info: Users can modify their details.
- Output of the Image: The system returns results, identifying whether the plant is healthy or diseased.

Context Level 1 Diagram

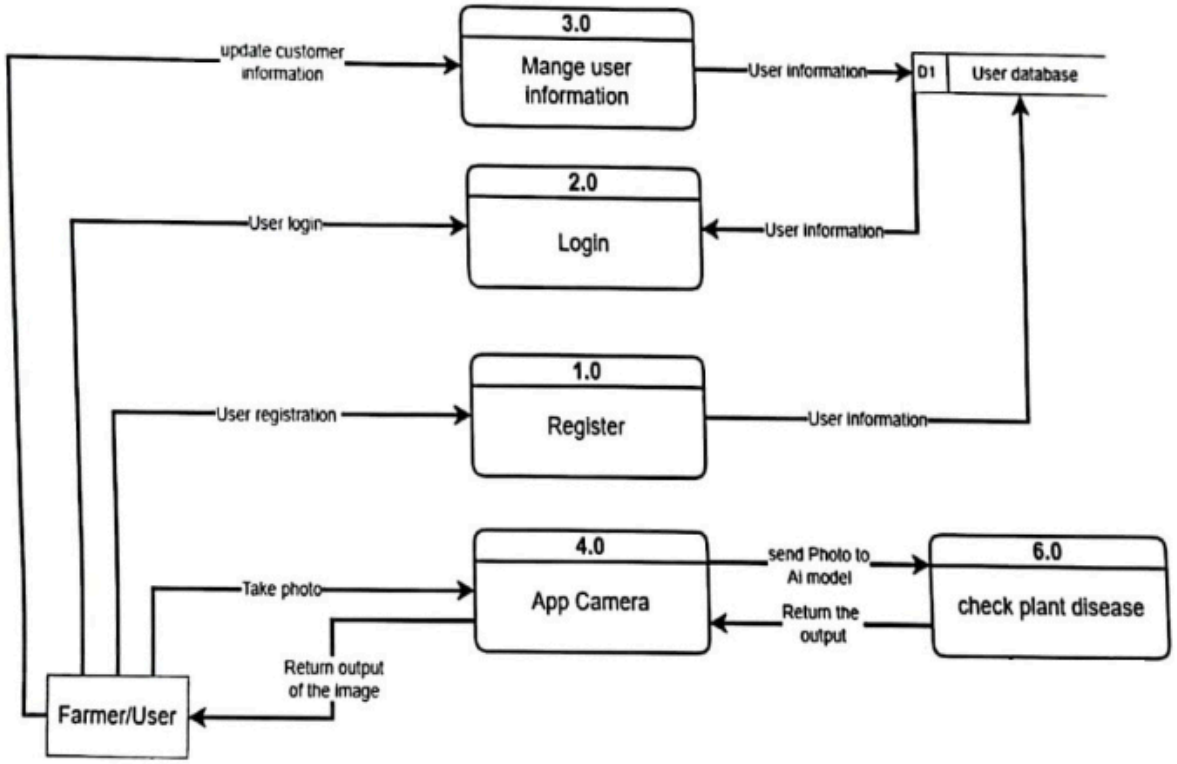


Figure 3 level 1 context diagram

Context level 1 diagram provides an In-Depth overview of the GreenNest System and its interaction with the Farmer/User.

Main Processes:

1. **Register (1.0):** Users register by providing information stored in the **User Database**.
2. **Login (2.0):** Registered users log in to access system features.
3. **Manage User Information (3.0):** Users can update their details in the database.
4. **App Camera (4.0):** Users take plant photos, which are sent for analysis.
5. **Check Plant Disease (6.0):** The AI model processes images and returns disease detection results.

Use case diagram

The GreenNest System caters to two types of users: **Guest Users** and **Users with an account**

The system allows basic browsing and information viewing for guests, with the ability to register. Registered users gain access to the main detection features, result saving, profile management, and a dashboard overview.

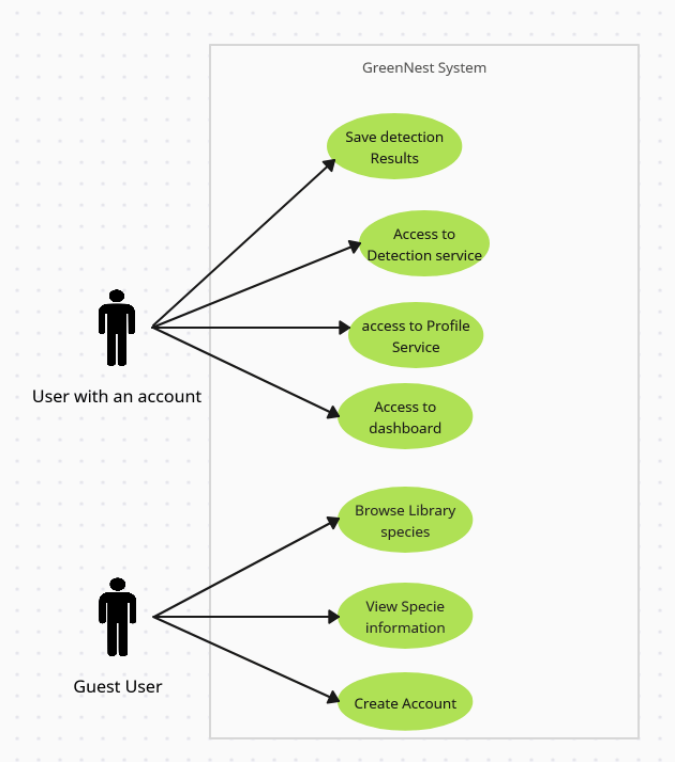
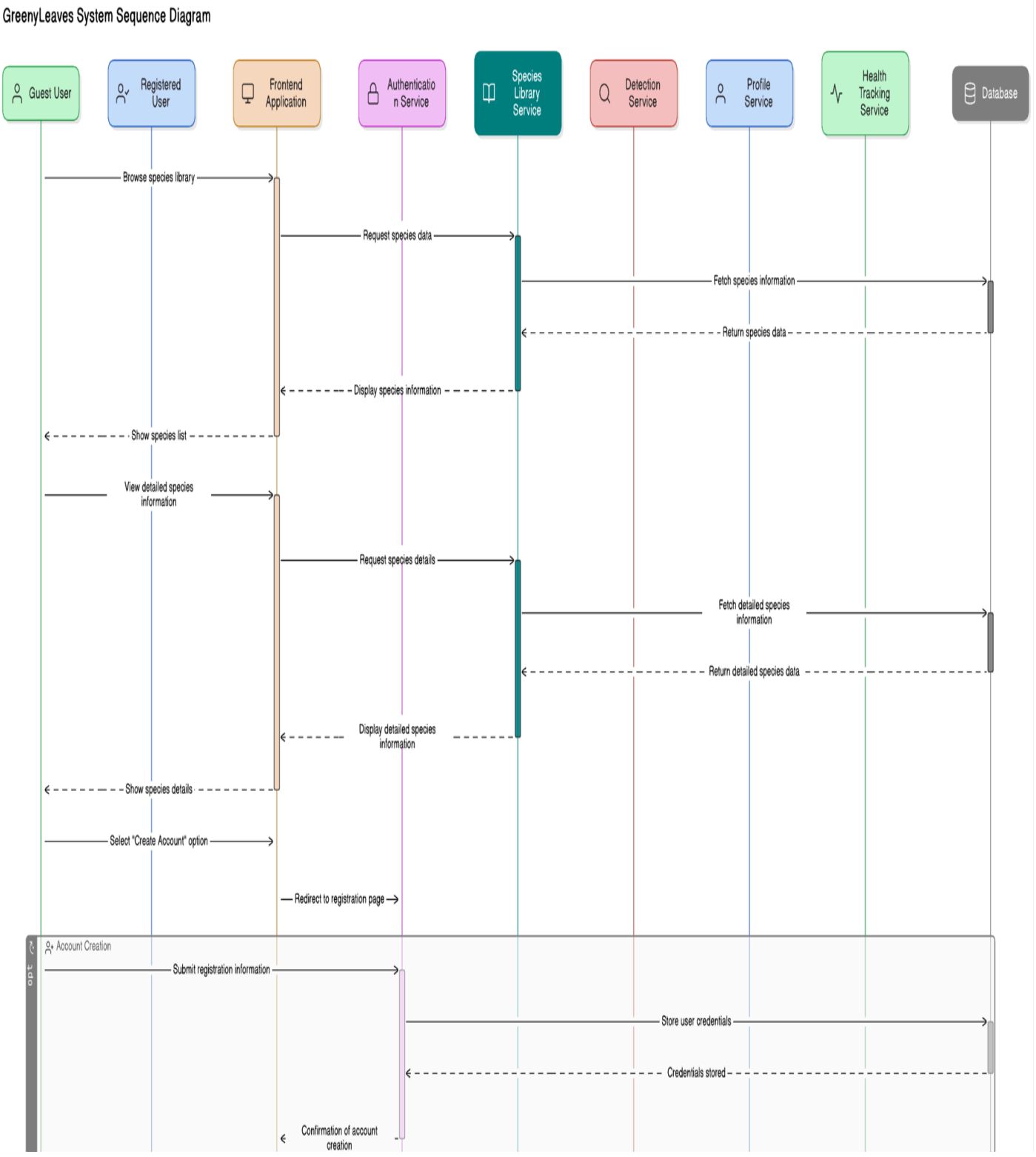


Figure 4 Use case Diagram

1. **Guest User:**
    - Can browse the library of species within the system.
    - Can view detailed information about specific species.
    - Has the option to create an account to gain more functionality.
2. **User with an account (Registered User):**
    - Can access the core "Detection service".
    - Can save the results obtained from the detection service.
    - Can access and manage their "Profile Service".

Sequence Diagram

This sequence diagram shows key interactions within the GreenNest Plant Leaf Disease Detection System, involving the User, the core System, and the database.



Sequence diagram components

The diagram shows several key components represented by colored boxes at the top and bottom

Key Interactions and Workflows

The sequence diagram illustrates multiple user-initiated processes flowing across the system architecture. The interactions primarily show:

1. **User Authentication Flow** - Initial login sequence with security verification
2. **Data Request Sequences** - How data is fetched from databases and external APIs
3. **API Integration** - Communication with external environmental data sources

Core Functionality:

- Monitor plant conditions
- Aggregate data from multiple sources
- Store historical environmental data for analysis
- Authenticate users with appropriate access levels

User Authentication (Registration/Login):

- A user initiates registration or login with the System.
- The System verifies credentials or creates a new user by interacting with the Database ("Read/Write user data").

On success: The System sends a verification email (via Email Service) for registration and confirms login/registration to the user.

Core System Interaction (Handling the "scan request" sequence):

- This part of the diagram represents different actions depending on the user type:
- **For Registered Users:** This sequence represents initiating the **disease detection feature**. The System interacts with the AI model through a flask API to detect the leaf disease, then the Database ("Read/Write detection data") to **save the detection results** to the user's history.
- Successful detection grants access to results and remediation features.

ERD Diagram

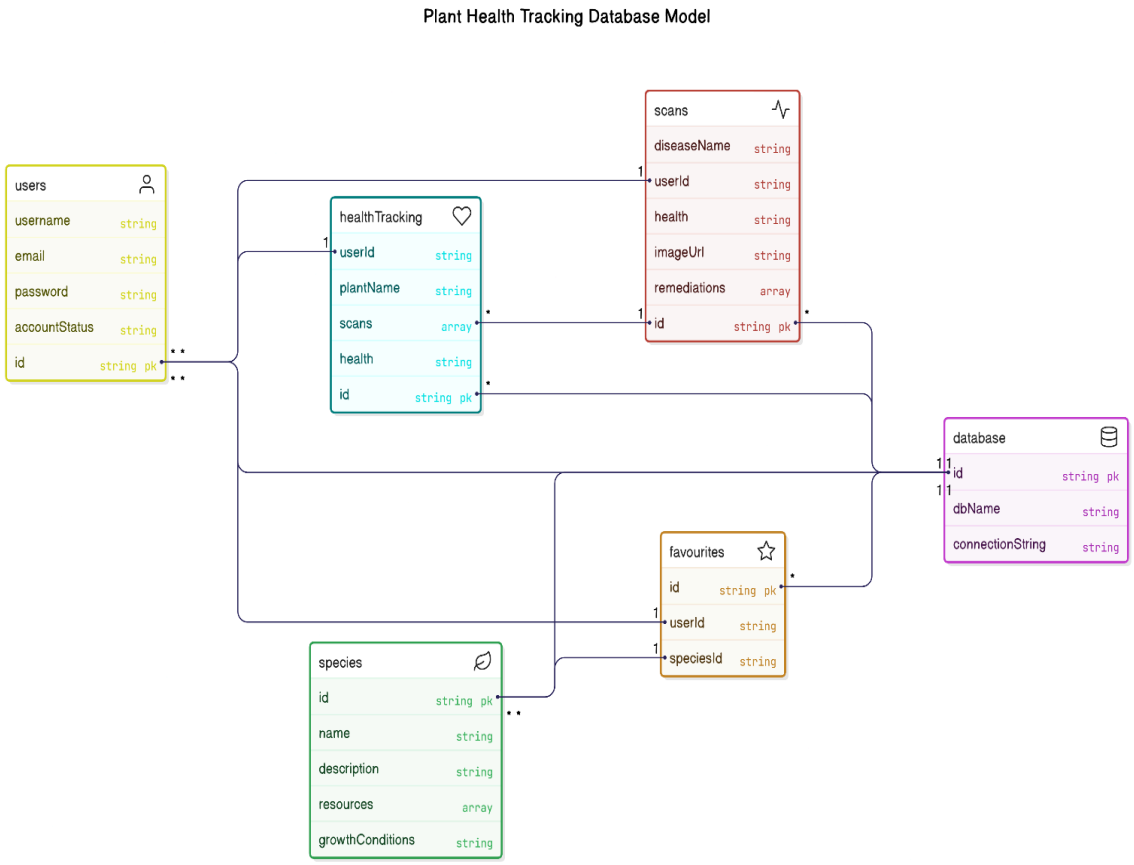


Figure 6 ERD Diagram

The Entity-Relationship Diagram (ERD) represents the database structure for the **GreenNest System**. It illustrates relationships between different entities and their attributes.

Summary of Entities and Core Functionality:

The data model is structured around six primary entities: users, species, healthTracking, scans, favorites, and database.

- The **user's** entity manages user account information, including credentials and account status, serving as the primary point of identification within the system.
- The **species** entity acts as a knowledge base, storing detailed information about various plant species, such as descriptions, resource links, and optimal growth conditions.
- The **healthTracking** entity allows users to monitor the ongoing health of specific plants they own or manage. Each entry is linked to a user and can aggregate multiple diagnostic scans over time. The plantName attribute allows for user-defined naming, distinct from the formal species name.
- Individual diagnostic events are captured in the **scan** entity. Each scan records details like the suspected disease, an image URL, assessed health status, and potential remediation steps. Scans are associated with a user and are also intended to be linked to a healthTracking instance.
- The **favorites** entity serves as a junction table, enabling a many-to-many relationship between users and species. This allows users to curate a personalized list of plant species they are particularly interested in.

Class Diagram

Figure 6 ERD Diagram

The Class Diagram represents The GreenNest System with three main entities:

1. **Users (Yellow)** – Stores user authentication and verification details (id, email, password, createdAt, etc.).

2. **Species (Green)** – Represents plant species with detailed attributes like scientific_name, genus, family, common_names, image_url, and Boolean fields (edible, vegetable).

3. **Scans (Blue)** – Stores disease-related data (name, description, symptoms, causes, treatment, healthy_images, diseased_images).

4. **Favourites (Orange)** Stores favorite species information (scientific name, resources, general useful information for farmers)

5. **HealthTracking (Purple)** Stores Tracked Plant's health and remediations by referencing scans (previous scans, name, disease, remediations)

Relationships:

1. Users ↔ Favourites
One-to-Many: A user can have many favourites.
Favourites references Users via user_id.

2. Users ↔ HealthTracking
One-to-Many: A user can track the health of many plants.
HealthTracking references Users via user_id.

3. Favourites ↔ Species
Many-to-One: Each favourite points to one species.
Favourites references Species via Class.

4. HealthTracking ↔ Species

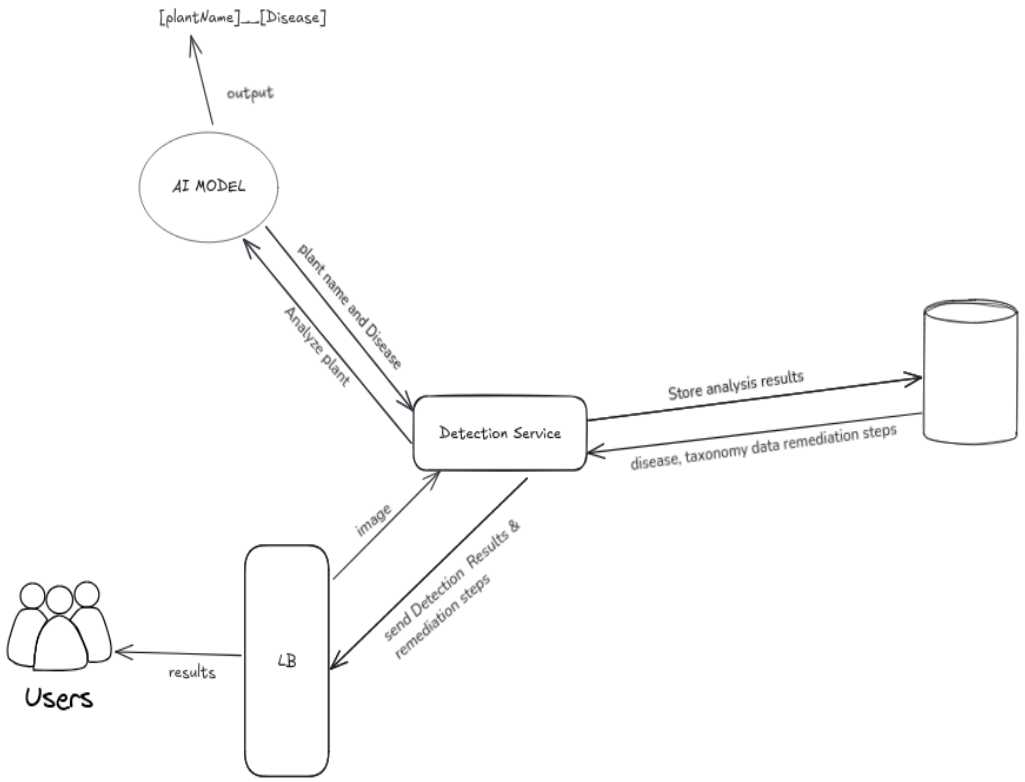
Many-to-One: Each tracked health entry is associated with one species.
HealthTracking references Species via Class.

5. HealthTracking ↔ Scans

Many-to-Many (or Embedded List): A health tracking entry can reference multiple scans (e.g., previous scans).
HealthTracking stores an array of scanIds or embeds scan summary
This design organizes user, plant, and disease data efficiently for the system.

System Architecture

The GreenNest System is designed to facilitate the management of users and plant species, including functionalities for user authentication, species information management, and disease remediation. The architecture follows a modular approach, separating concerns into distinct components that interact through well-defined interfaces.



Technologies and Tools

Web/Mobile

- **Node.js**: Backend for the main application
- **Flask**: Separate backend specifically for serving an API to the deep learning model
- **React**: Frontend JavaScript library for building the user interface
- **Flutter**: Cross-platform framework for mobile application development
- **Postman**: API testing platform and a documentation tool

Machine Learning & Data Science

- **Kaggle**: Platform used to obtain plant disease datasets and training the deep learning models
- **PyTorch**: Deep learning framework used for model training and development
- **Jupyter Notebooks**: Used for model Training, experimentation, and documentation
- **Gemini 2.5 LLM**: Used for obtaining actionable remediation steps and disease management guide.

Deep Learning Libraries

- **torchvision**: PyTorch package with datasets, model architectures, and image transformations
- **NumPy**: Library for numerical computing and array operations
- **Pandas**: Data manipulation and analysis library
- **Matplotlib/Seaborn**: Visualization libraries for model performance analysis
- **scikit-learn**: Used for evaluation metrics and preprocessing

Storage, Hosting, And Administration Services

- **Google Cloud Platform (GCP) Compute Engine:** Used for hosting the backend and serving the AI model via a publicly accessible server
- **Google Cloud Buckets**: used for storing user uploaded pictures
- **Nginx**: used as the main Web server and proxy
- **PM2**: Package manager to manage the backend servers
- **GitHub**: Repository where the project source code is stored and shared publicly

Communication

- **SendGrid**: Email delivery service integrated for notifications and alerts

Security Testing

- **BurpSuite CE:** suite of tools around proxying and request tampering
- **OWASP ZAP:** Proxy and open-source dynamic scanner (DAST) tool
- **Pentest Tools:** enterprise dynamic scanner (DAST) tool

Design patterns

The System's Backend follows the **Model-View-Controller (MVC)** architecture pattern to separate concerns and improve maintainability.

The MVC Design Pattern

MVC consists of models, views, and controllers, each serving a specific purpose.

- **Model (Data Layer)**
    - Represents the database and manages data-related logic.
    - Defined using schema models for users, plant species, scan history, etc.
- **View (Presentation Layer)**
    - Not implemented in this backend project but handled by a separate frontend.
    - The backend serves as an API, responding with JSON data for the frontend UI.
- **Controller (Application Logic Layer)**
    - Handles HTTP requests, processes data via services, and returns responses.
    - Ensures business logic is executed correctly.
- **Services (Business Logic Layer)**
    - Provides core functionality such as AI detection, authentication, Library, and file handling.
    - Separates business logic from controllers for reusability and cleaner code.

How MVC was Implemented

- **Models**: Mongoose Collection Schemas
    - Define Different Collection types
- **Controllers**: Manage API endpoints and requests
    - Define API routes and HTTP methods
    - Validate input parameters
    - Coordinate between model operations and response formatting

- **Views**: Generate structured API responses
    - Format model predictions into standardized JSON
    - Include prediction metadata
    - Handle error responses consistently

Structure Breakdown

Figure 9 Backend Implementation

Server Initialization

server.js

This is the main entry point for the backend

- Initializes Express server.
- Loads environment variables.
- Registers routes and middleware.
- Connects to the MongoDB database.

Models (Data Layer)

These are the schema or models that can be used to interact with the corresponding database collections or tables. The models help in structuring the data and defining relationships between different entities in the application.

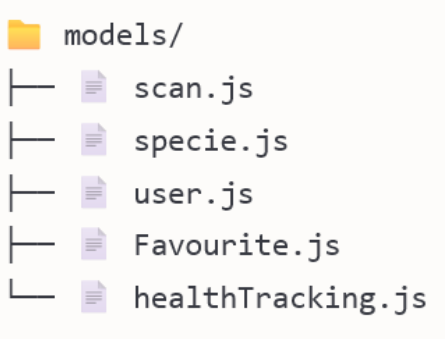


Figure 10 models MVC

Defined using **Mongoose ORM (MongoDB)**:

- **user.js** – Stores user information, authentication details.
- **specie.js** – Holds plant species and Library data.
- **scan.js** – Maintains a history of user scans.
- **Favourite**.js: maintains a list of favorited species
- **healthTracking**.js: holds tracked plants information such as diseases, and treatment options

Controllers (Application Logic)

Responsible for handling requests and interacting with services:

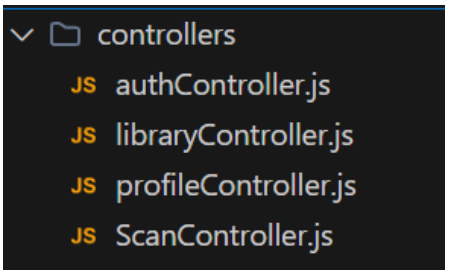


Figure 11 Controllers MVC

- **authController.js** – Manages user authentication and token issuance.
- **libraryController.js** – Fetches plant disease information, handles species related operations.
- **scanController.js** – Handles scan-related operations, plant health tracking and history.
- **profileController**.js handles user profile operations.

Routes (API Endpoints)

Routes map incoming requests to controllers:

- **AuthRoutes.js** – ***/api/v1/users/*** (Routes for Authenticated users, dashboard, history, Detection)
- **libraryRoutes.js** – ***/api/library/*** (Retrieve plant disease information)

Services (Business Logic Layer)

Services handle business logic such as the detection service, where uploaded images are sent to the model API by the detection service.

Figure 12 Services MVC

- **detection_service.js** – Calls the AI model to detect plant diseases from images.
- **user_service.js** – Handles user management tasks.
- **email_service.js** – Manages email notifications.
- **token_service.js** – Generates and verifies authentication tokens.
- **Library_Service**.js Manages the specie retrieval and
- **storage_service.js** – Manages file uploads (for storing plant leaf images).

Middlewares

Handles Security & Validation for example, user uploaded photos are checked for file extension, ensuring the users performing requests have valid session cookies, etc.

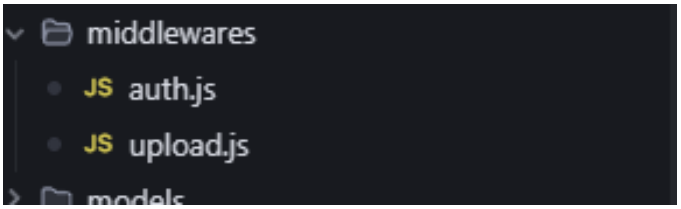


Figure 13 Middleware/Helper functions

- **Authentication middleware** – Protects API endpoints, Ensures correct input format for API requests
- **Upload**.js – Handles file uploads and storage

Library Parser

A CSV parser was made to Convert the library's dataset of 460k plant species from CSV into JSON objects for compatibility with MongoDB and to build the library Collection.

Quick Summary

Guest User Interaction

1. Guest users can browse our plant library with over 460k species
2. Guest users can view general plant details and health information

Registered User Interaction

- registered users upload a plant leaf image.
- Registered users can save scan history.
- Registered users can access dashboard and profile settings

Image Upload & Processing

- Image is uploaded via the upload middleware
- Detection service processes the image using an AI model.

Disease Detection & Response

- The AI model predicts disease probability.
- The response includes disease name, and possible treatments.

Data Storage & Retrieval

- Scan history is saved in MongoDB for registered users.
- Users can query disease details via *the /api/library/* endpoint.

Authentication & User Management

- authController.js handles login and JWT token generation.
- Secure endpoints require authentication via token_service.js.

The Greenest application follows the **MVC architecture** to ensure a clean, modular, and scalable structure. By separating concerns between models, controllers, and services, the system efficiently handles plant disease detection, user authentication, and data storage while maintaining flexibility for future improvements.

Chapter Summary

This chapter outlines the system analysis and design process for the GreenNest system, a solution aimed at tackling plant disease detection to support agricultural resilience. It begins with a focus on identifying key users—such as farmers, agronomists, and stakeholders, ensuring their needs shape the system's development. Visual tools play a critical role in this process, with diagrams like context diagrams, use case diagrams, sequence diagrams, and class diagrams used to define the system's boundaries, functionalities, interactions, and data structure.

The chapter also delves into the technical design, detailing the system architecture that integrates components like a deep learning model and a mobile interface. The database design is structured to efficiently manage plant disease data, supporting quick retrieval and analysis. Technologies such as DeepLearning tools for disease detection and the Flutter platform for cross-platform accessibility are highlighted as key enablers, enhancing the system's accuracy and scalability.

To manage the development process, the Scrum Agile methodology is employed, allowing for iterative progress and adaptability to evolving requirements. Together, these elements—user analysis, visual modeling, architectural design, and agile management—form a cohesive framework for building an innovative and practical system aligned with the project's mission to improve global food security.

Chapter 4

Model Training Methodology

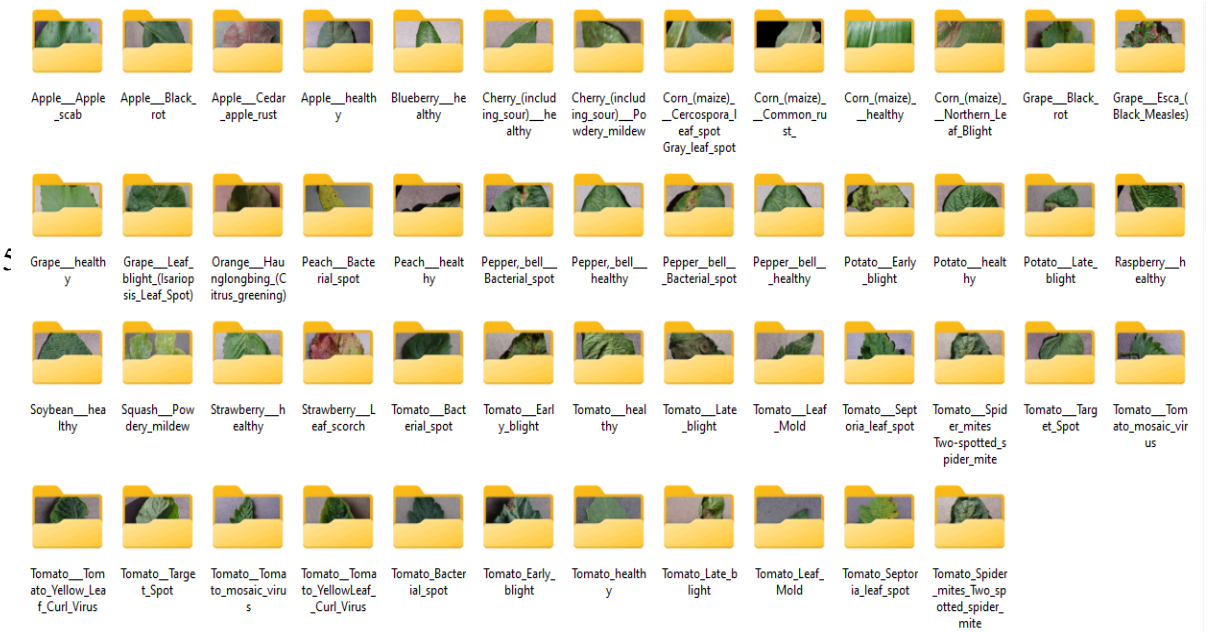
- Introduction
- Implementation Methodology
- Testing
- Evaluating the model Accuracy
- Future Recommendations

Chapter Introduction

This chapter details the design, implementation, and training Methodology of our AI based system. We begin by describing the dataset preprocessing steps and augmentation strategies employed to enhance model robustness. Next, we outline the Architecture of ResNet50 for multi-Class disease detection, followed by the training protocol, hyperparameter selection, and evaluation metrics. The results of this work demonstrate the feasibility of deploying deep learning for real-world agricultural applications, paving the way for scalable precision farming tools.

Dataset Preparation

Our model was trained on the expanded **new plant Diseases dataset**, which comprises 87,000+ high-quality RGB images of healthy and diseased plant leaves. The dataset encompasses **38 distinct disease classes** across multiple crop species including tomato, potato, apple, corn, and several others. The images were collected under controlled conditions with consistent lighting and background, making them ideal for initial model training.



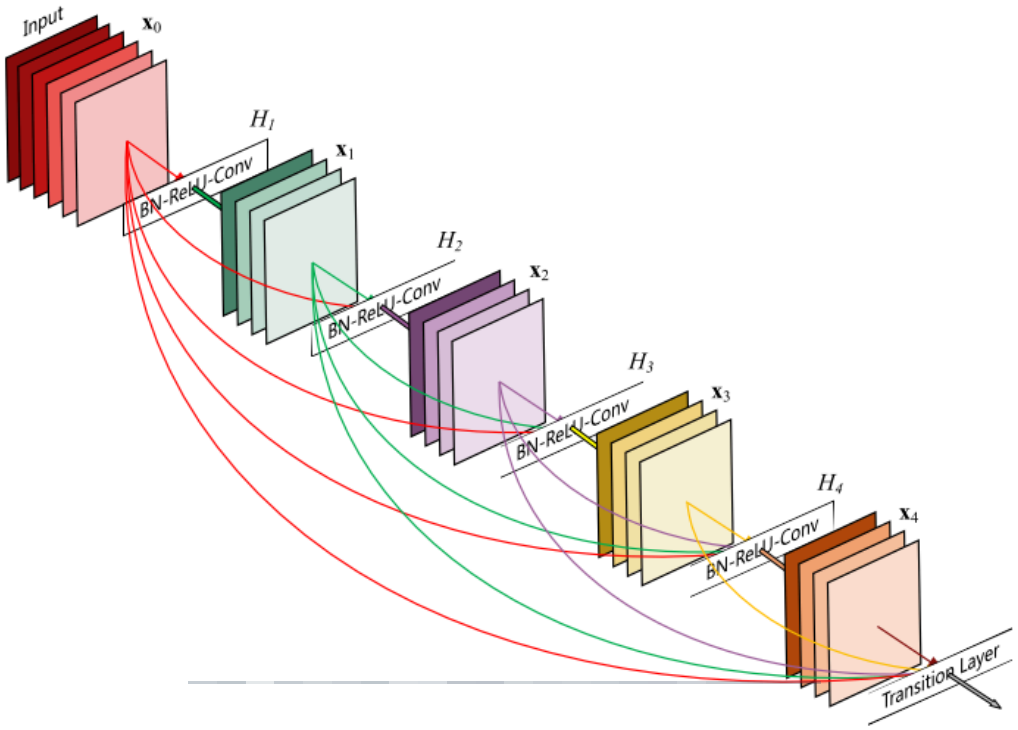
Model Architecture

ResNet-50 is a deep learning Architecture introduced by Microsoft Research Team, designed to improve the performance of deep neural networks by using residual blocks. These blocks allow the network to learn residual representations, which simplifies the optimization process and enables the training of much deeper networks compared to traditional architectures.

Why ResNet-50

ResNet50 is the ideal architecture for our disease classification needs due to its excellent balance of performance and efficiency. With a 92-93% top-5 accuracy on the ImageNet benchmark, it delivers the robust capabilities required for our application.

In Resnet-50, unlike in traditional neural networks, each layer feeds into the next layer, we use a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away, to avoid over-fitting (a situation when validation loss stops decreasing at a point and then keeps increasing while training loss still decreases). This also helps in preventing vanishing gradient problem and allow us to train deep neural networks



ResNet-50 is particularly effective for plant leaf disease detection due to several reasons:

1. **Hierarchical Feature Learning**: ResNet-50's deep architecture allows it to learn hierarchical representations of images, capturing both simple features like edges and textures, and complex features like shapes and whole objects. This capability is crucial for identifying diseases on plant leaves, which often involve subtle changes in texture and color

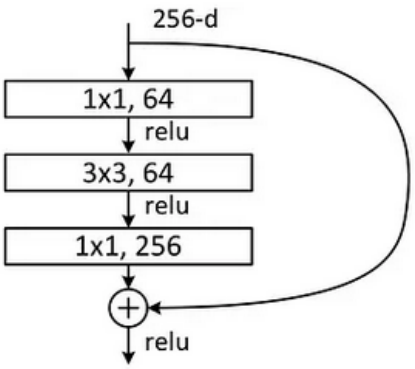
2. **Generalization Ability**: The model's ability to generalize well means it can perform accurately on unseen data, which is important for real-world applications where data variability is high. This helps in detecting diseases even when the model encounters new or slightly different leaf images.
3. **Adaptability**: ResNet-50 can be easily adapted for specific tasks through transfer learning, allowing developers to fine-tune the model on smaller datasets specific to plant leaf diseases. This reduces the need for large amounts of training data and speeds up the development process.

4. **High Accuracy**: Studies have shown that ResNet-50-based models can achieve high accuracy in plant leaf disease detection. For example, the ResNet50-DPA model achieved an accuracy of 99.28% in identifying tomato leaf diseases.

Breakdown of the Resnet-50 architecture

The architecture includes several key components:

- **Residual Blocks**:



- Each residual block consists of a series of convolutional layers followed by a direct connection (skip connection) that adds the initial input to the output of these layers. This design helps prevent the vanishing gradient problem, allowing gradients to propagate more efficiently through the network.
- **Skip Connections**: These are direct connections that bypass one or more layers, allowing the input to be added to the output of the convolutional layers. This facilitates better gradient flow and helps in training deeper networks.
- **Convolutional Layers**: These layers are used for feature extraction. In ResNet-50, they are organized into residual blocks to improve learning efficiency.
- **Fully Connected Layer:** A fully connected layer with SoftMax activation is used for final classification, outputting probability scores across 38 classes.
- Overall, ResNet-50's architecture and capabilities make it an excellent choice for plant leaf disease detection tasks, offering high accuracy and adaptability in a variety of scenarios.

The Training Process

The training pipeline was designed to optimize model performance while ensuring computational efficiency and generalization.

Several training algorithms were implemented to manage the end-to-end training workflow, including batch processing, forward/backward propagation, and parameter updates. Key components of the training process included:

Dataset Preparation

To enhance model robustness in real-world scenarios, by **augmenting** the original dataset with various transformations including rotations, flips, brightness adjustments, and minor color variations. This augmentation process not only

expanded our effective training data but also improved the model's ability to generalize across different environmental conditions.

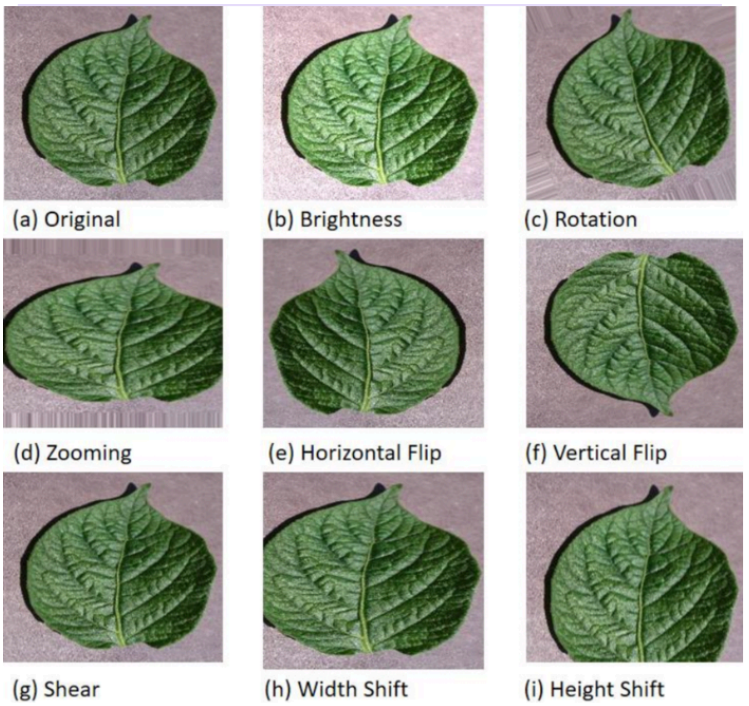


Figure 17 sample data augmentation

The modified dataset of 79,000 images was split using holdout validation: 80% for training, and 10% each for validation and testing.

Batch Processing

Images were loaded in batches of 128 to balance memory constraints and gradient stability.

Figure 18 Sample batch

Configurations & Optimization

- **Output Layer Adjustment**

  The initial model weights were frozen, and the final output layer was reconfigured to include 38 neurons corresponding to the 38 disease classes.

- **Learning Rate**

  Managed by the Adam optimizer, with a cosine annealing scheduler applied to dynamically adjust the learning rate, promoting smoother convergence.

- **Training Epochs**

  Training was set for a maximum of 10 epochs, with early stopping implemented to terminate training if the validation loss plateaued, ensuring efficient use of computational resources.

Validation Function

To monitor generalization and prevent overfitting, a validation step was executed after each epoch using a validation dataset.

Loss Function

Categorical Cross-Entropy was chosen as the primary loss function due to its effectiveness in multi-class classification tasks, such as plant disease identification across *K* distinct classes. Key advantages include:

- **Stronger Penalty for Confident Errors**: It penalizes confident yet incorrect predictions more heavily than uncertain ones.
- **Enhanced Gradient Signals**: It provides stronger gradient updates when the predicted probabilities deviate significantly from the ground truth, accelerating learning.
    - **Synergy with SoftMax:** It works seamlessly with the SoftMax activation function in the final layer of ResNet50, ensuring stable and interpretable probability outputs.

Training Process Steps

Forward/Backward Propagation: Input image batches were processed through the ResNet50 architecture. Predictions were evaluated using the categorical cross-entropy loss, followed by backpropagation to compute gradients.

Summary

- Batch Processing: Training images were loaded in batches of 128, striking a balance between GPU memory efficiency and gradient stability. This batch size helped ensure smoother updates and reduced the risk of overfitting due to noisy gradients.
- Forward Pass: Each batch of images was passed through the ResNet50 architecture. The network extracted hierarchical features through its
- convolutional layers, producing class probability distributions via the final SoftMax-activated output layer.
- Loss Computation: Categorical cross-entropy loss was calculated by comparing predicted probabilities against the true labels. This quantified the model's performance per batch and provided a scalar loss value to guide training.

- Backward Propagation: Gradients of the loss with respect to each model parameter were computed using backpropagation. This step allowed the model to learn which features and connections needed strengthening or weakening.
- Parameter Updates: The Adam optimizer, enhanced with a weight decay regularization term (1e-4), was used to update model weights, while weight decay discouraged excessively large weights, aiding generalization to promote more refined weight updates in later epochs and help the model converge to a better local minimum.

    - Validation Step: After each epoch, the model was evaluated on a separate validation set to monitor generalization performance. This provided insight into whether the model was improving or overfitting.
    - Early Stopping: To prevent unnecessary computation and overfitting, early stopping was triggered if the validation loss ceased to improve over successive epochs, thereby halting training once optimal performance was reached.

Accuracy Evaluation

Below is the calculated accuracy report and confusion matrix for the model across the 38 classes

Initial Performance

The model starts with very low accuracy (close to 0) at epoch 0, which is expected as the network begins with randomly initialized weights.

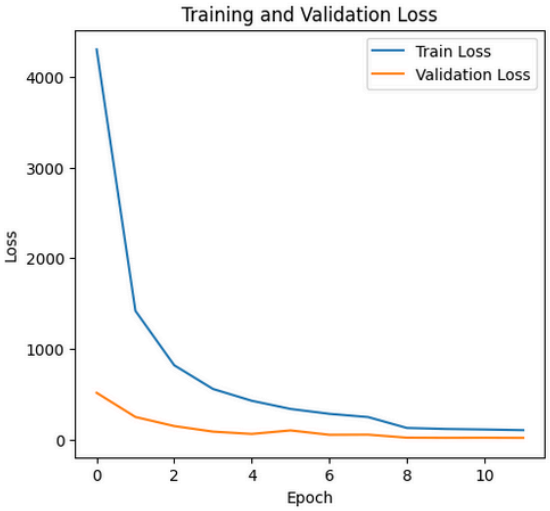


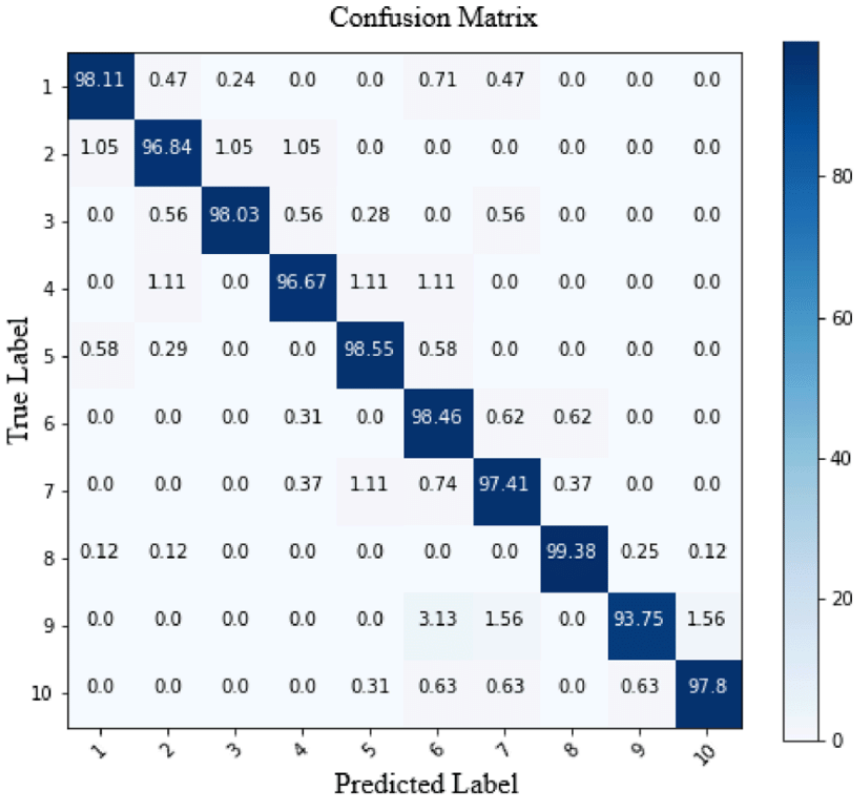
Figure 19 Training epochs and validation loss

Rapid Improvement

Figure 20 Training epochs and validation loss

There's a dramatic increase in accuracy during the first 5 epochs, with the accuracy jumping from near 0 to approximately 0.9 (90%).
This steep climb indicates that the model is quickly learning effective features for Classifying plant diseases.

Confusion matrix



Label	Class Name	Label	Class Name
1	Bacterial Spot	6	Spider mites
2	Earlyblight	7	Target spot
3	Lateblight	8	Tomato yellow leaf curl virus
4	Leaf mold	9	Tomato mosaic virus
5	Septoria spot	10	Healthy

The matrix shows how the model's predictions compare to the actual labels

- Rows represent the "True Labels" (the correct classes)
- Columns represent the "Predicted Labels" (what the model predicted)
- The numbers in each cell show how many examples fall into that combination
- The darker blue diagonal cells contain larger numbers - these represent correct predictions where the true label matches the predicted label

The matrix shows strong performance, as most predictions fall on the diagonal (correct classifications), with relatively few mistakes scattered elsewhere. The model achieves high accuracy on all 38 classes!

Classification report on Sample Test set

Class	Precision	Recall	F1-Score	Support
Apple – Apple Scab	0.92	1.00	0.96	3
Apple – Cedar Apple Rust	0.95	0.95	0.95	4
Corn (Maize) – Common Rust	0.96	1.00	0.98	3
Potato – Early Blight	0.89	1.00	0.94	5
Potato – Healthy	0.90	0.90	0.90	2
Tomato – Early Blight	0.94	0.83	0.88	6
Tomato – Yellow Leaf Curl Virus	0.91	0.83	0.87	6
Tomato – Healthy	0.96	1.00	0.98	4

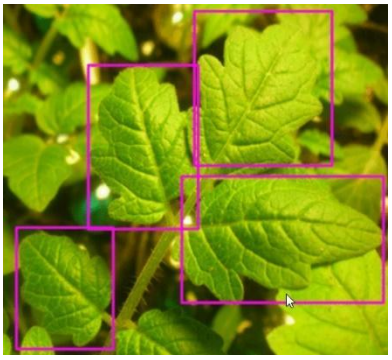
Integration of YOLOv8 for Object Detection

Motivation and Need for YOLOv8

In our current model pipeline, the primary objective is to classify plant health conditions based on input images. However, a significant limitation arises from the assumption that the input image contains only a relevant plant leaf. In practice, images captured in the field may contain background noise, multiple plant parts, or even irrelevant objects. This can mislead the classifier, especially if the leaf is partially visible or not centered.

To address this challenge, we integrate **YOLOv8 (You Only Look Once version 8)** as a **preprocessing stage for object detection**. YOLOv8 is a state-of-the-art object detection model that allows us to first **locate and identify the leaf** in the image.

By isolating only the relevant leaf region, we significantly improve the input quality for the classification stage (handled by our ResNet50-based model). This ensures that the classifier receives focused, meaningful visual data, enhancing its accuracy and robustness in real-world scenarios.



yolov8

YOLOv8 Architecture Overview

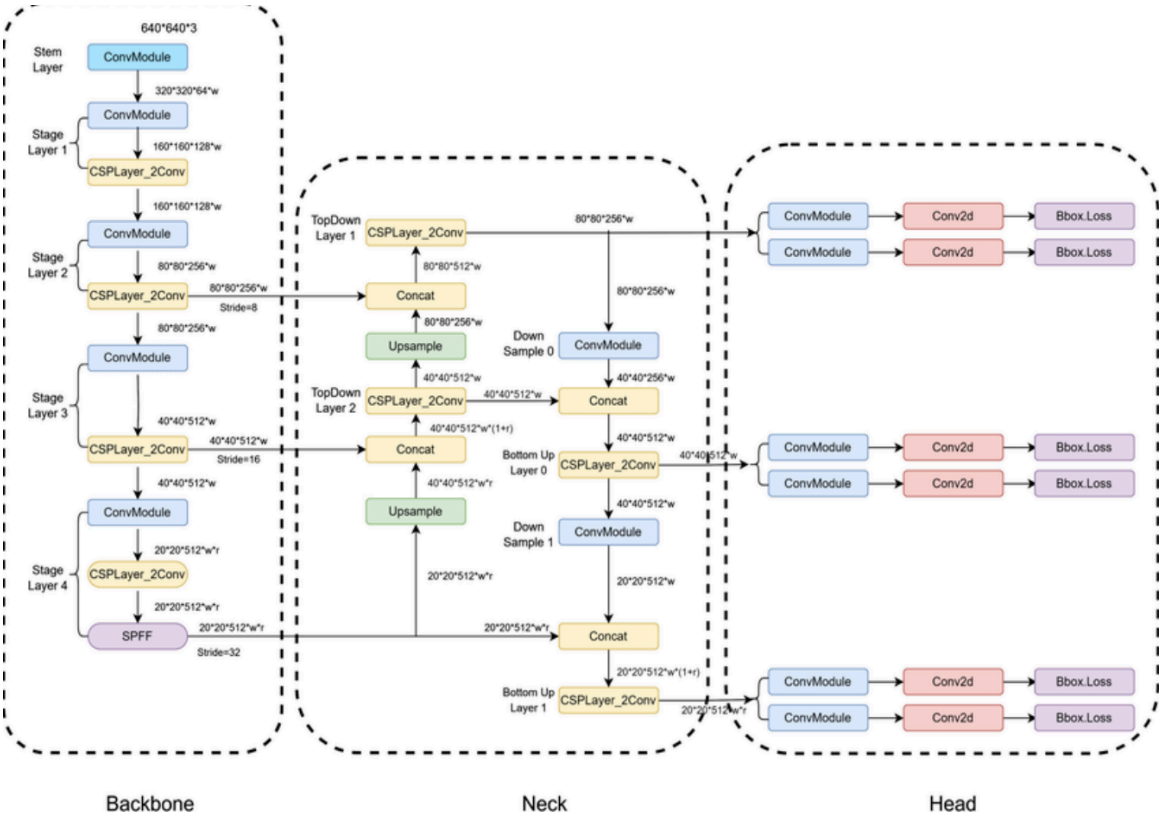


Figure 23 Yolov8 Architecture

1. **Backbone**:

   YOLOv8 uses a modern convolutional backbone (C2f-based) to extract rich, hierarchical features from input images. This component is responsible for detecting edges, textures, and complex patterns that are useful for identifying objects like leaves.

2. **Neck**:

   The neck is responsible for **multi-scale feature fusion**. YOLOv8 uses PAN (Path Aggregation Network) to combine features at different resolutions. This helps the model detect small and large objects effectively—especially useful for leaves that vary in size and position.

3. **Head**:

   YOLOv8 features a **decoupled head** that separately predicts:

   - Objectness (is there an object?)
   - Class probabilities
   - Bounding box coordinates

     This separation allows for more precise predictions and better training dynamics compared to older versions.

Integration with ResNet50

After YOLOv8 detects and crops the leaf region, the resulting image segment is passed to our **ResNet50 classifier**. This modular approach leverages the **strengths of YOLOv8 in localization** and **ResNet50 in fine-grained classification**.

By combining these two with Gemini prompt as a plant health expert, our system forms a **Three-stage pipeline**:

1. **YOLOv8** for leaf detection (object detection/localization)
2. **ResNet50** for disease classification (image classification)
3. **Gemini2.5 flash**: Researches the symptoms, treatment plan and retrieves them to the user as GreenNest's remediation service

This approach ensures scalability, better generalization to field data, and improved performance in challenging visual conditions.

Chapter Conclusion

The Project successfully implemented and evaluated a deep learning approach for plant disease classification using the ResNet50 architecture. Our methodology demonstrated exceptional performance in identifying plant diseases across multiple crop species. The key achievements of this work include:

1. **High Classification Accuracy**: The model achieved an impressive 98.79%, demonstrating its robustness and reliability for real-world applications. This performance exceeds many existing approaches in the literature and confirms ResNet50 as an effective architecture for this domain.
2. **Efficient Training Pipeline**: We developed a comprehensive training pipeline with optimized hyperparameters, this configuration enabled rapid convergence while preventing errors.
3. **Effective Knowledge Transfer**: By leveraging transfer learning from the pre-trained ResNet50 model, we successfully adapted its hierarchical feature learning capabilities to the specific characteristics of plant disease manifestations, despite having a relatively limited dataset size.

Chapter 6

Application Testing

- Performance testing
- Security assessment methodology
- Security assessment report
- Findings
- Conclusion

Table: Performance Testing

Test Suite	Test Case	Result	Notes
Error Handling	Invalid Uploads	✔	Proper error messages returned for invalid files
	Network Issues	✘	Partial uploads not retried correctly
	Security	✔	SQLi, XSS, and CSRF protection verified
Edge Cases	Upload Large Image	✔	Handled with error Message sent to user
	Upload Corrupt Image	✔	Proper error message displayed
	Unauthorized API Access	✔	Unauthorized users were not able to see limited data

Summary:

- **Error Handling:** Invalid file uploads were handled correctly, but network recovery needs improvement.
- **Edge Cases:** Large images caused crashes, and unauthorized access was possible under specific conditions.

Security Assessment Report

Executive Summary

A comprehensive security assessment was conducted on the application to identify potential vulnerabilities and security weaknesses. The assessment focused on common web application security issues including input validation, authentication mechanisms, and HTTP security configurations.

Two primary security issues were identified during the assessment: missing security headers and a Local File Inclusion (LFI) vulnerability in the username parameter. Both vulnerabilities have been successfully remediated as part of the security improvement process.

Assessment Methodology

The security assessment employed multiple testing approaches:

- **Static Code Analysis**: Review of source code for potential security flaws
- **Dynamic Testing**: Runtime testing of application functionality
- **Configuration Review**: Analysis of server and application security configurations
- **Manual Testing**: Targeted testing of input validation and authentication mechanisms

To guide the testing and remediation efforts, we followed the OWASP Top Ten as a framework for identifying and addressing common web application vulnerabilities. These standards helped prioritize testing around areas such as:

- Input validation (to prevent injection attacks)
- Authentication and session management
- Access control enforcement
- Data protection and error handling

we focused on ensuring the application's security by leveraging both Blackbox and Whitebox testing methodologies, with an emphasis on Whitebox approach

Summary of Findings

Finding ID	Vulnerability Type	Severity	Component	Status
SEC-001	Missing Security Headers	Medium	Web Server Configuration	**Remediated**
SEC-002	Local File Inclusion (LFI)	Low	Username Parameter	**Remediated**

Detailed Findings

SEC-001: Missing Security Headers

Severity: Medium
Component: Web Server Configuration
Status: Remediated

Description:
The application was missing critical HTTP security headers that help protect against common web-based attacks. The absence of these headers left the application vulnerable to various client-side attacks including cross-site scripting (XSS), clickjacking, and content type sniffing attacks.

Missing Headers Identified:

- X-Content-Type-Options: nosniff
- X-Frame-Options: DENY
- X-XSS-Protection: 1; mode=block
- Strict-Transport-Security
- Content-Security-Policy

Impact:
Without proper security headers, the application could be susceptible to:

- Cross-site scripting attacks
- Clickjacking attempts
- MIME type confusion attacks
- Man-in-the-middle attacks (for HTTPS connections)

Remediation Applied:
All missing security headers have been implemented in the web server configuration:

SEC-002: Local File Inclusion (LFI) in Username Parameter

Severity: Low

Component: Username Parameter Processing

Status: Remediated

Description:

A potential Local File Inclusion vulnerability was discovered in the username parameter handling mechanism. While the application was not properly validating path traversal sequences in the username field, testing revealed that the vulnerability was not exploitable to access actual system files.

Impact:

The impact of this vulnerability was limited due to its unexploitable nature:

- No actual file disclosure possible
- Potential for confusion in logging or display contexts
- Minor security hygiene issue requiring remediation

Remediation Applied:

The following security measures have been implemented:

1. **Input Validation**: Strict validation rules applied to username parameter
2. **Input Sanitization**: Removal of dangerous characters and path traversal sequences
3. **Whitelist Approach**: Only alphanumeric characters and specific safe characters allowed
4. **Path Restriction**: Implementation of secure file handling that prevents directory traversal
5. **Error Handling**: Improved error messages that don't reveal system information

Risk Assessment

Vulnerability	Likelihood	Impact	Risk Level
Missing Security Headers	**High**	**Medium**	**Medium**
LFI in Username	**Low**	**Low**	**Low**

Overall Risk Rating

Prior to remediation, the application presented a **Medium** overall security risk primarily due to the missing security headers, with the unexploitable LFI representing a minor security hygiene issue. Post-remediation, the risk level has been reduced to **Low** with the successful implementation of security controls.

Remediation Summary

All identified security vulnerabilities have been successfully addressed:

1. **Security Headers**: Comprehensive HTTP security headers have been implemented to protect against common web-based attacks
2. **Input Validation**: Robust input validation and sanitization libraries have been Integrated for the username parameter and other user inputs
3. **File Access Controls**: Secure file handling procedures have been implemented to prevent unauthorized file access

Verification Testing

Post-remediation testing was conducted to verify the effectiveness of the implemented security controls:

- **Security Headers**: Confirmed presence and proper configuration of all required security headers
- **LFI Testing**: Verified that path traversal attempts are properly blocked and sanitized
- **Input Validation**: Tested various malicious input patterns to ensure proper filtering

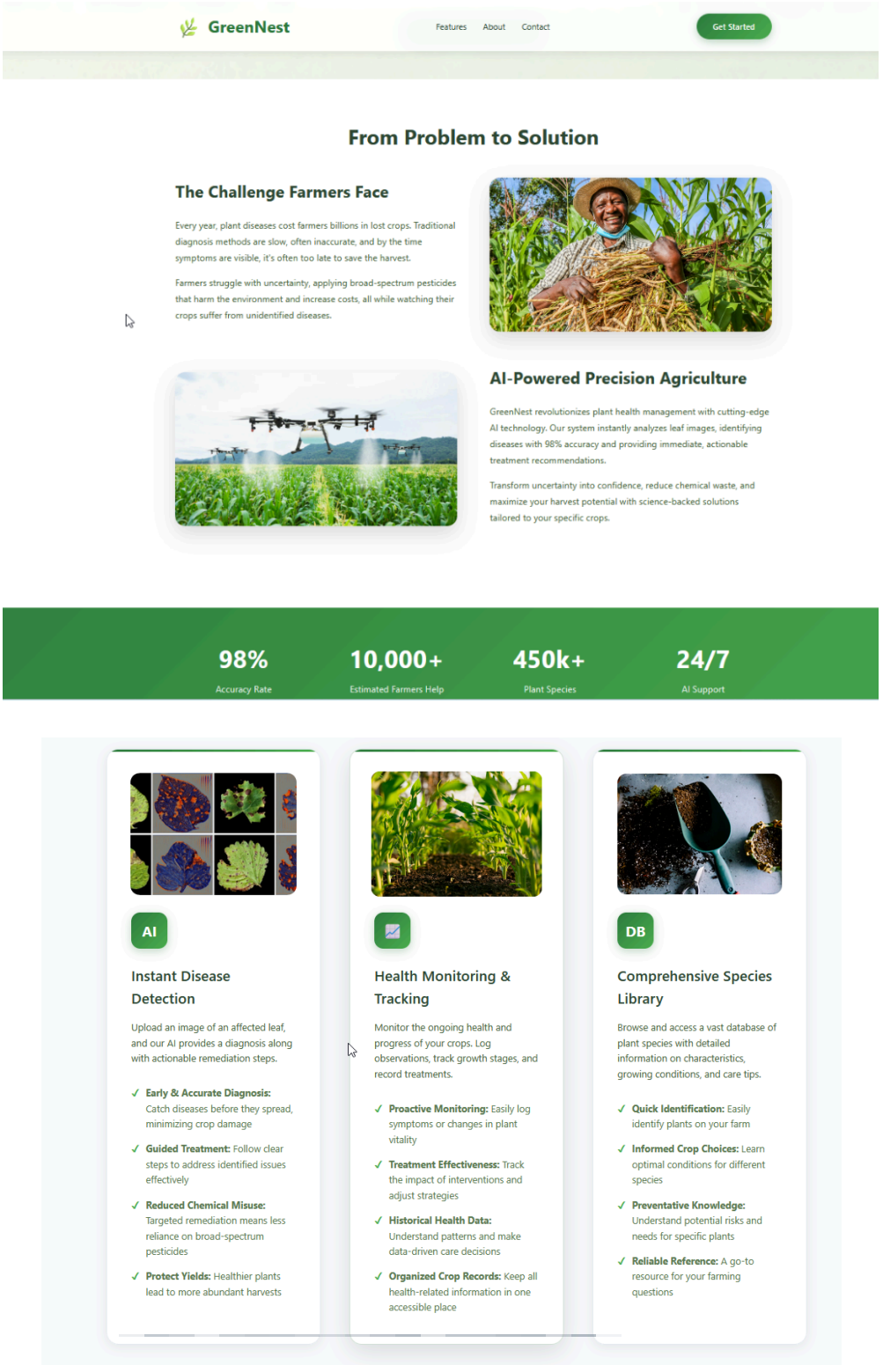
Conclusion

The testing phase for the GreenNest system has been successfully completed, confirming its readiness for deployment as a reliable and effective tool for plant disease detection. This conclusion is based on the outcomes of multiple testing methods, including, integration testing, system testing, and Security testing, each of which played a crucial role in ensuring the system's quality and usability.

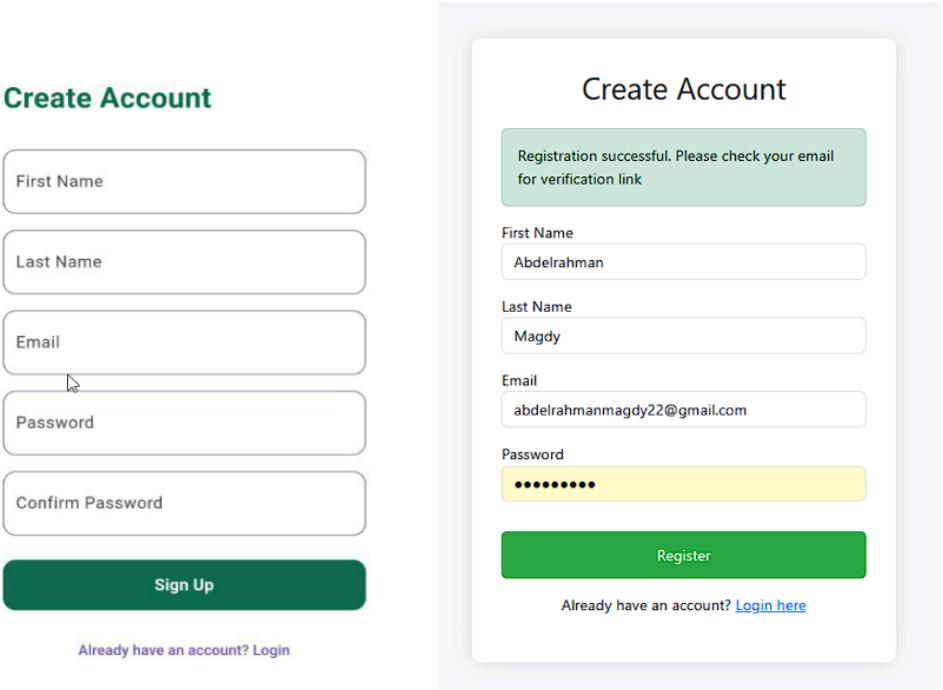
- AI model detected plant diseases accurately.
- Security mechanisms (SQLi, XSS, CSRF protections) are effective.
- Handling for Edge cases and business logic errors is correct

User Manual

Landing Page, browse to greennest.com



Registration



Email verification

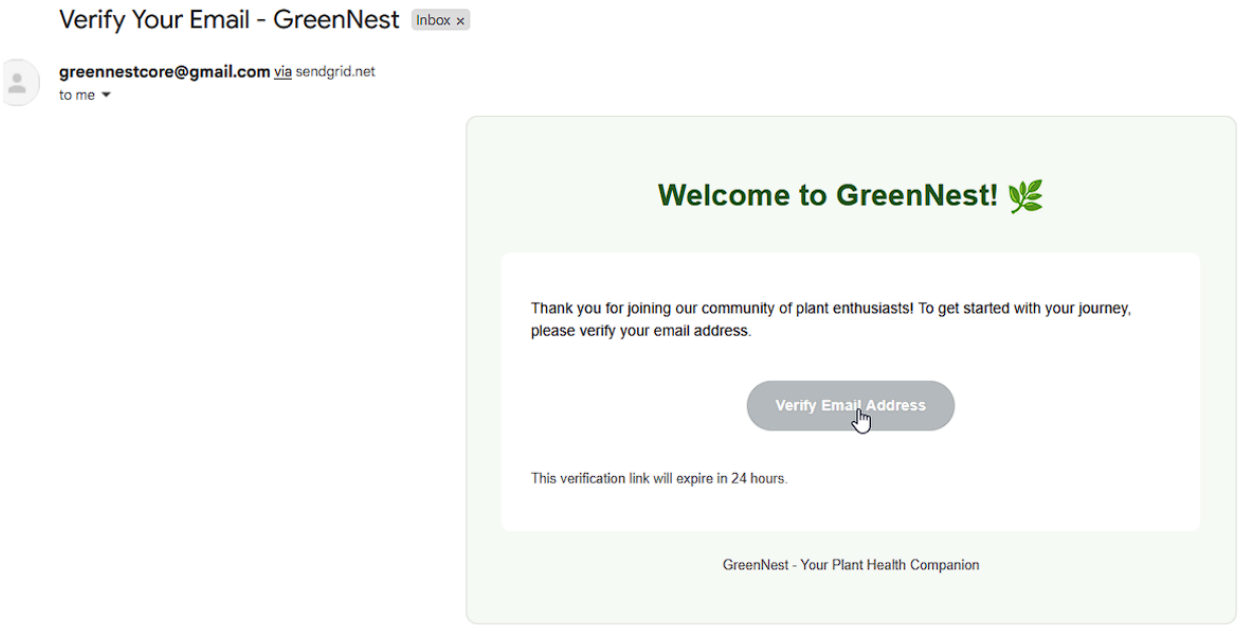


Figure 28 Email

Library

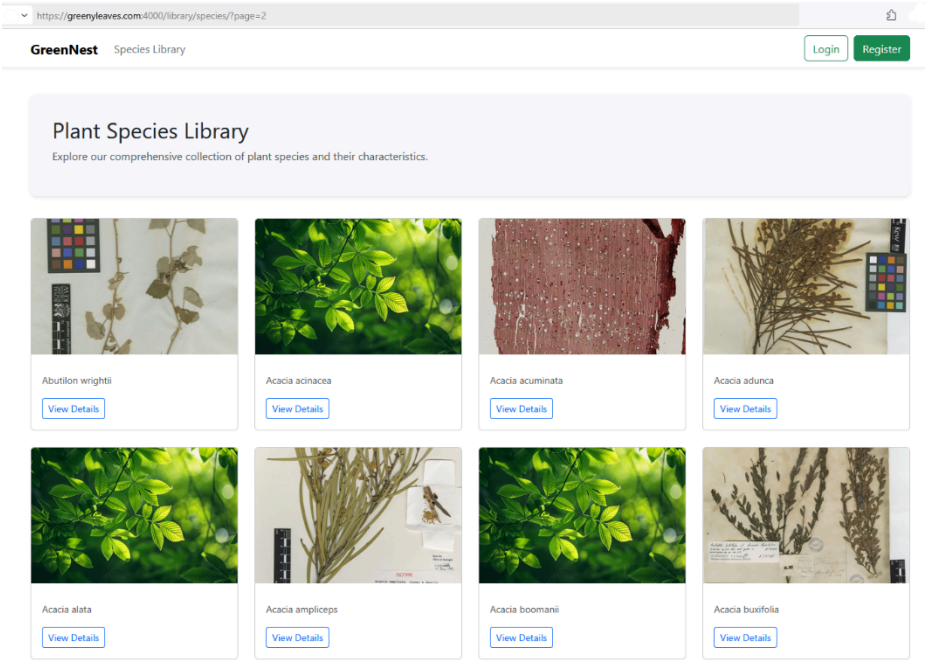


Figure 29 library

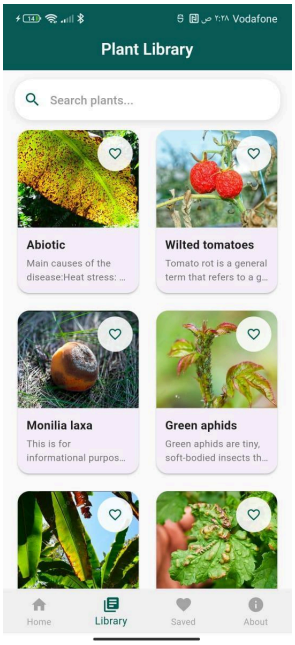


Figure 30 species Details

Library Species Details

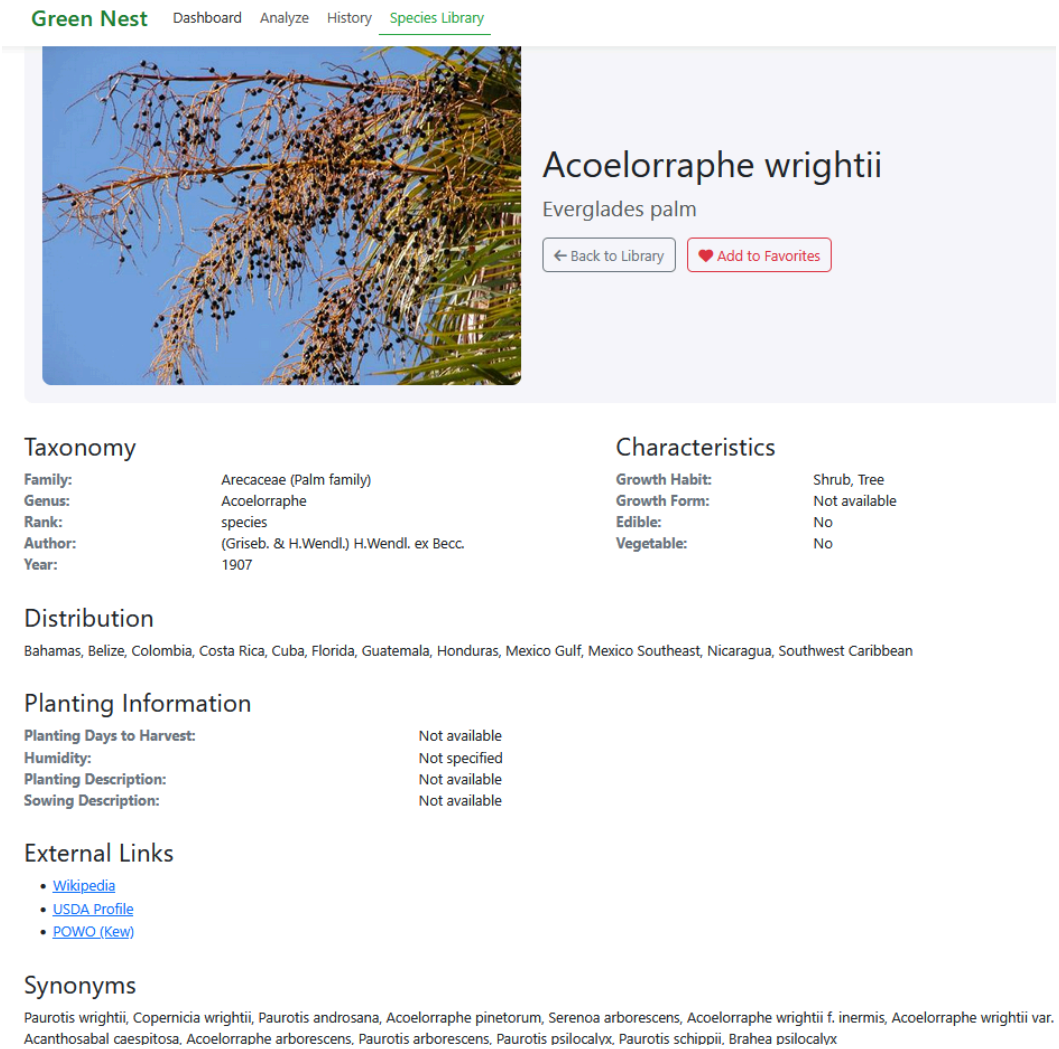


Figure 31 (Web) species Details

Detection Service

Scan page

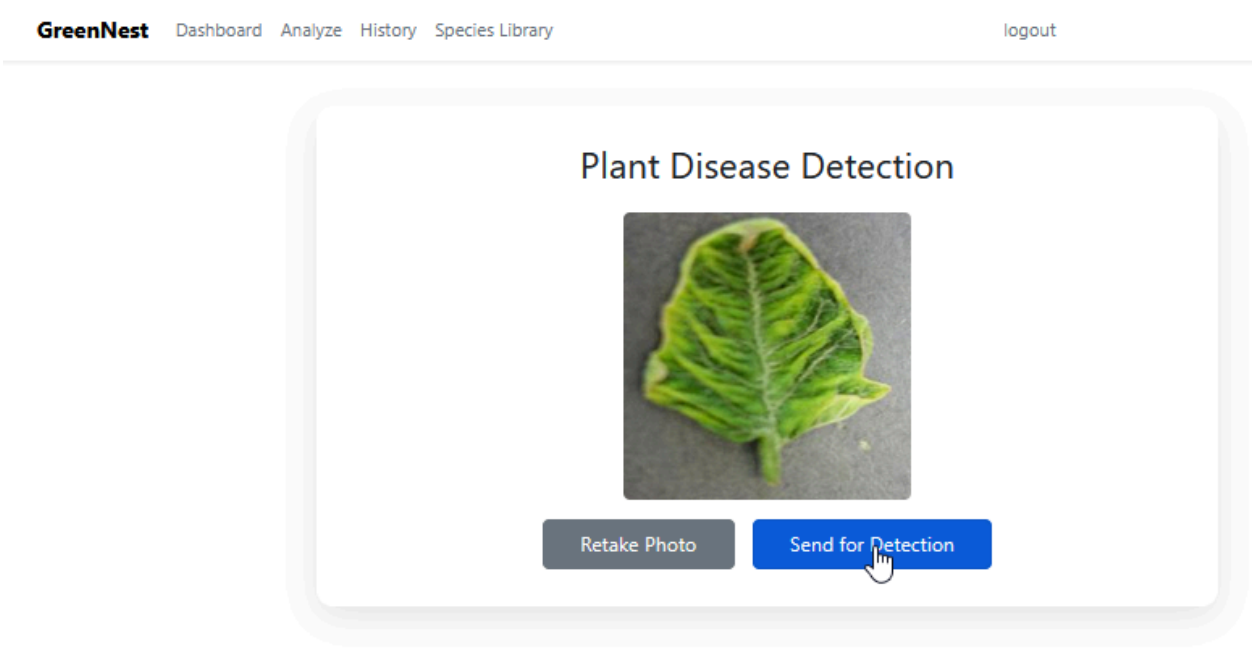


figure 32 scan page

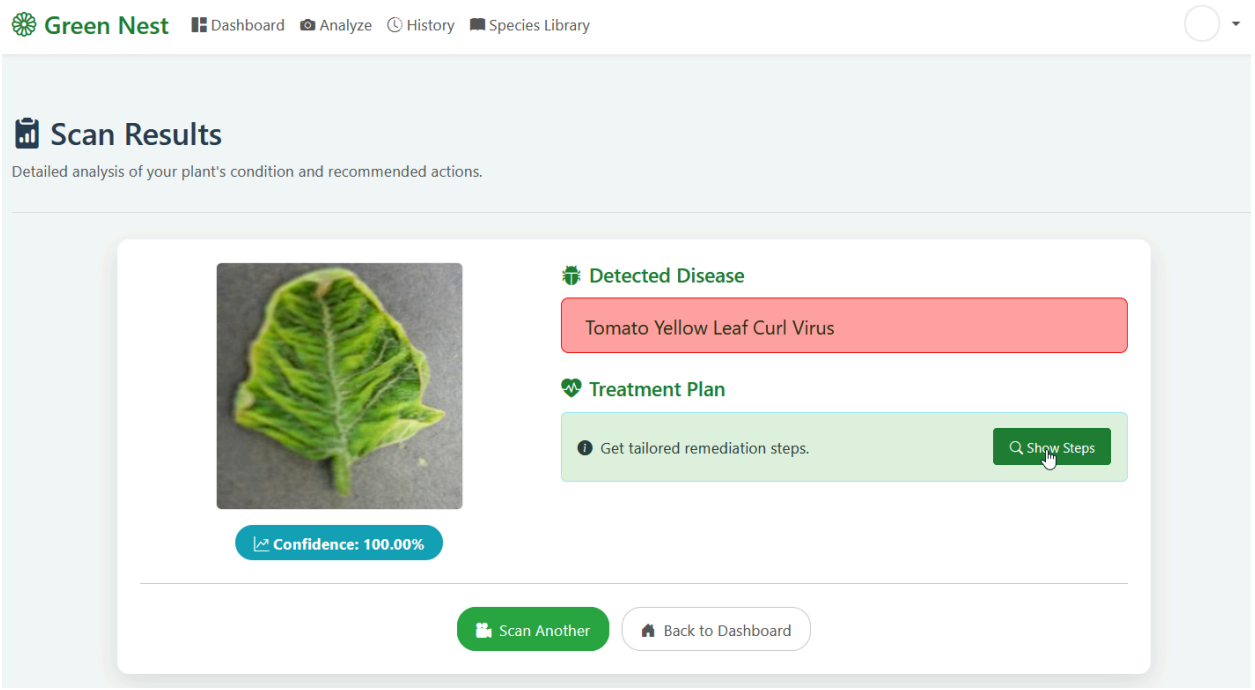


figure 33 results page

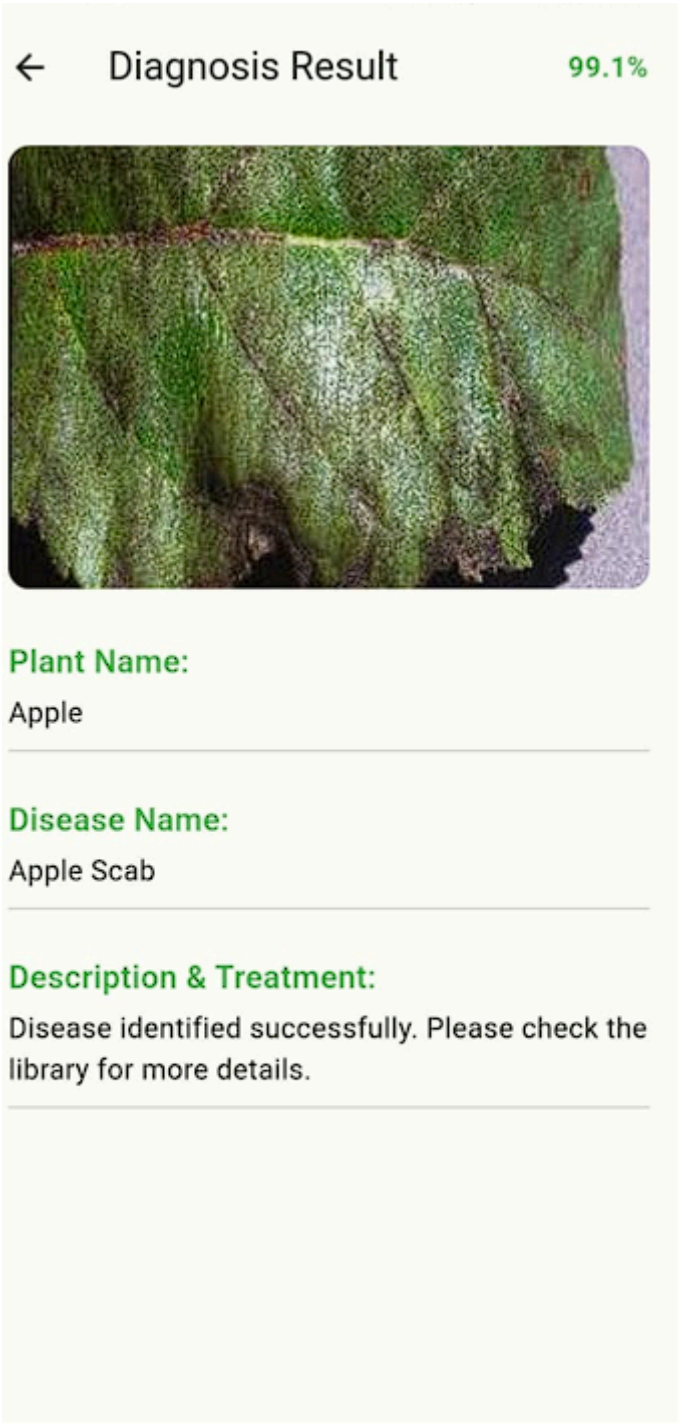


Figure 34 (Mobile) Results page

Symptoms & Treatment plan

Confidence: 100.00%

Detected Disease

Tomato Yellow Leaf Curl Virus

Treatment Plan

Tomato leaves are compound, odd-pinnate, and heavily veined.
Characterized by a strong, herbaceous scent when crushed or disturbed.

Tomato Yellow Leaf Curl Virus (TYLCV) Report

Observable Symptoms and Indicators:

- Upward curling of leaflets, particularly young leaves.
- Yellowing (chlorosis) of leaf margins and between veins.
- Stunted plant growth.
- Reduced fruit set and small, misshapen fruits.
- Thickened veins on affected leaves.
- Flower drop.

Underlying Causes and Contributing Factors:

- Transmission by whiteflies (Bemisia tabaci) feeding on infected plants.
- Introduction of infected transplants into the garden.
- Presence of alternate weed hosts harboring the virus.
- Favorable environmental conditions for whitefly proliferation: warm

Figure 35 treatment showcase

User Dashboard page

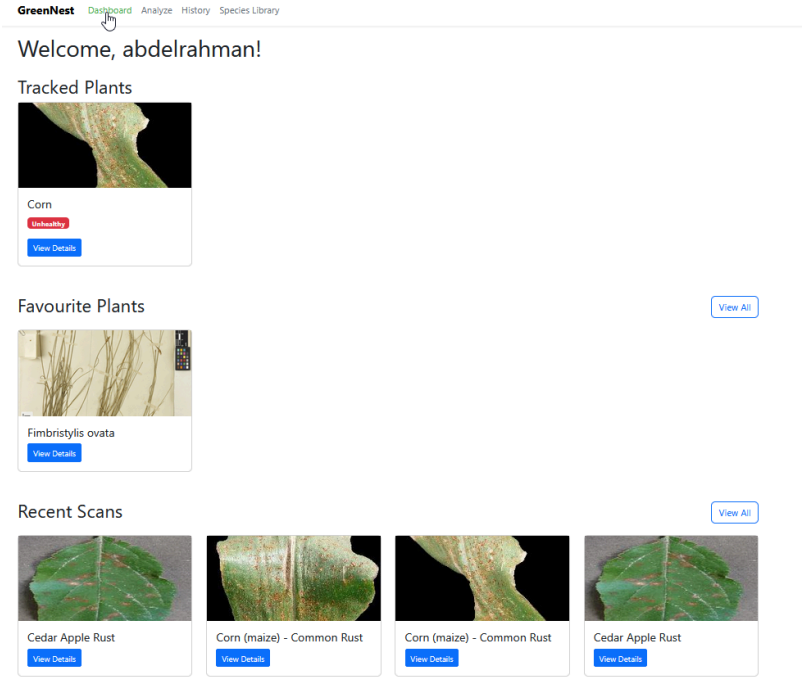


Figure 36 dashboard

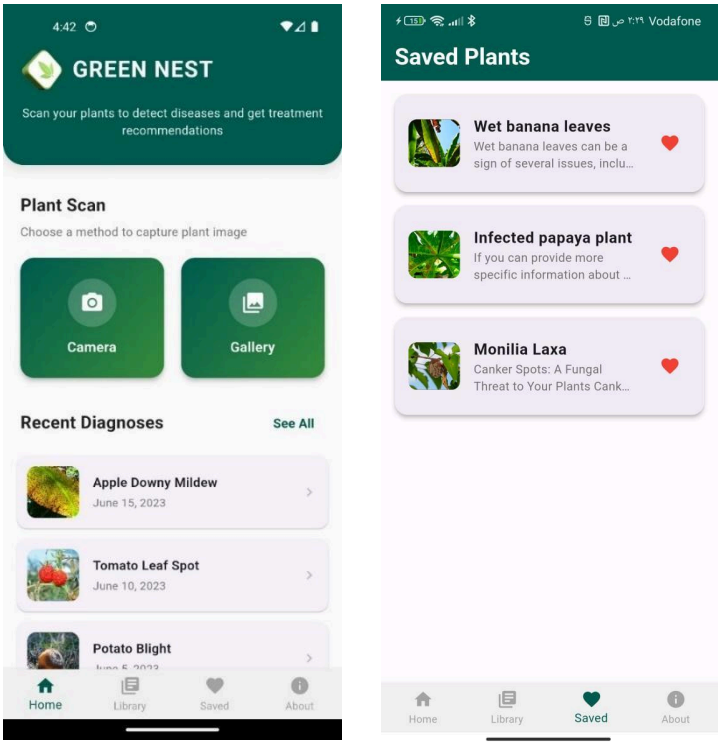


Figure 36 dashboard

Health Monitoring Service

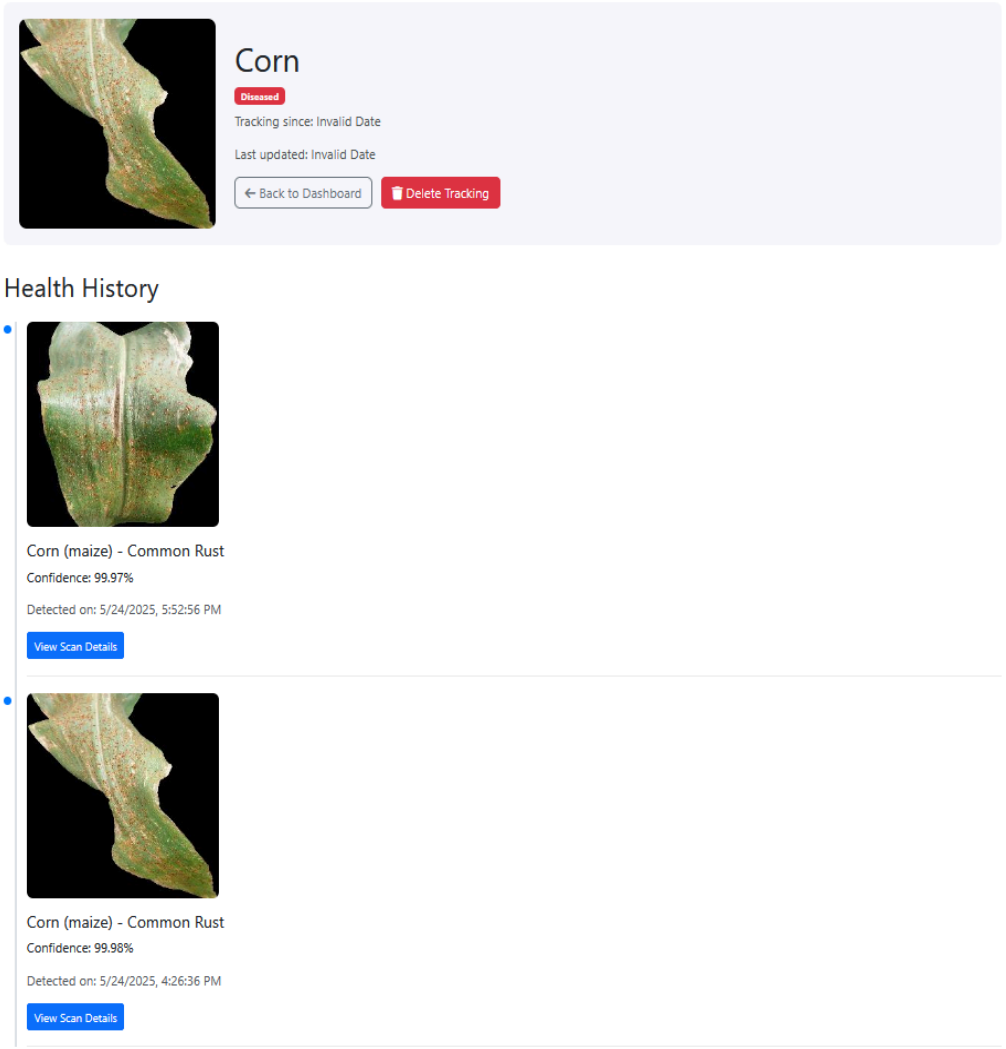


Figure 36 plant health monitoring

History Page

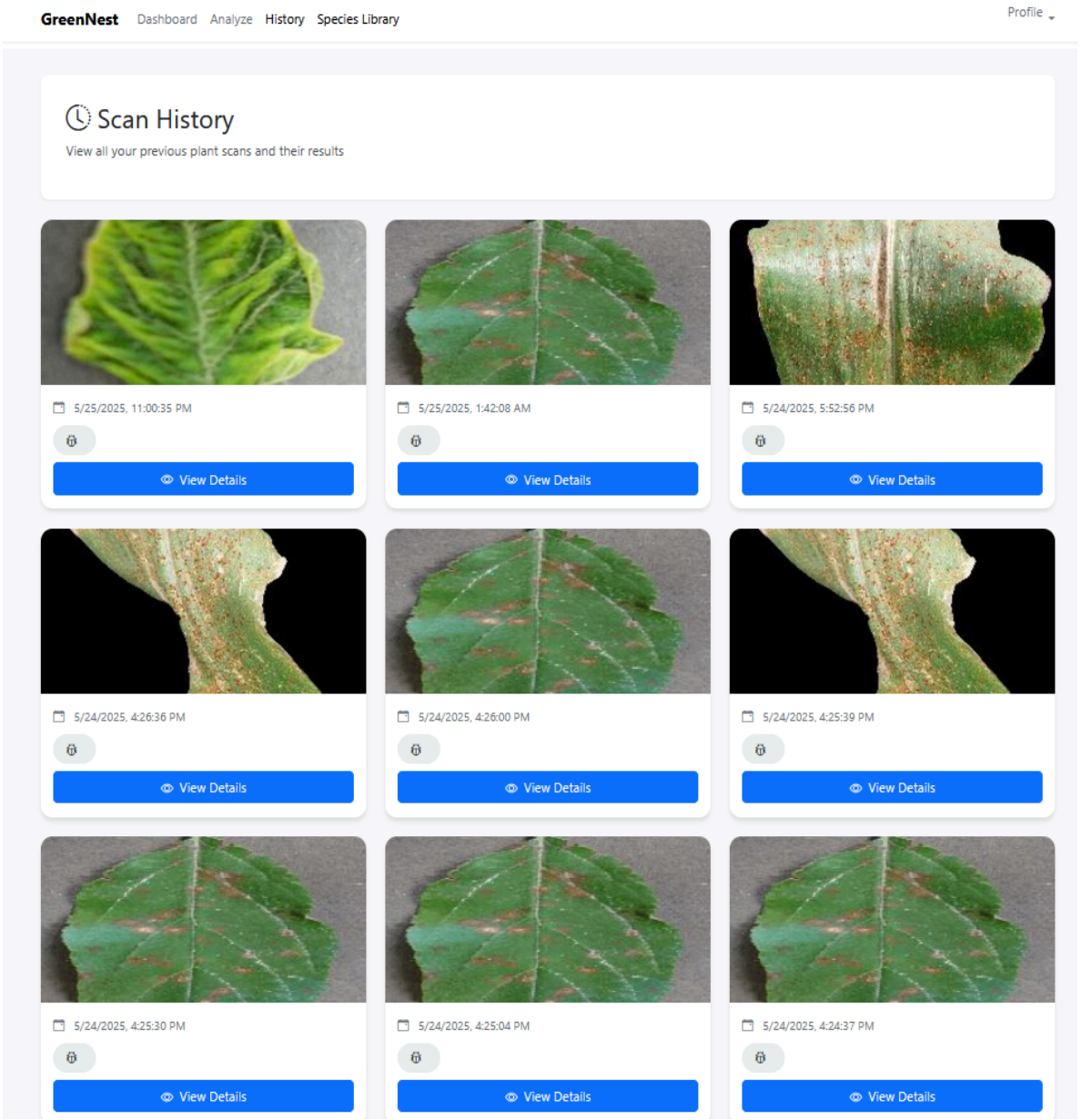


Figure 39 history page

Chapter 5

Summary and Recommendations

- Summary
- Main Findings
- Project Importance
- Limitations
- future recommendations
- Arabic Summary
- References

Summary

The GreenNest project focused on using AI, specifically deep learning with the Resnet50 architecture, to detect plant leaf diseases. This involved training a model to identify various diseases from plant images, aiming to help farmers with early detection and management.

Main Findings

The model achieved an accuracy of 98.78% across 14 disease categories, such as 5 disease classes specific to Apple leaves. Data augmentation was key, boosting accuracy to 98%, showing its value in enhancing model robustness.

Project Importance

This project is crucial for early disease detection, helping farmers prevent crop losses and increase yields. It offers a cost-effective, scalable solution that supports sustainable agriculture by reducing pesticide use. It also lays groundwork for further research and education in plant pathology and AI.

Project Limitations

Limitations include potential dataset bias, high computational requirements, and challenges in real-world applicability due to controlled image conditions.

Future Recommendations

focusing on dataset expansion, ensemble methods, and real-time applications can further enhance its impact, ensuring broader adoption and continued innovation in agricultural AI.

Conclusion

The GreenNest project marks a significant advancement in AI-driven plant disease detection, achieving notable accuracy and demonstrating the efficacy of data augmentation. Its importance lies in supporting sustainable agriculture and providing a scalable, cost-effective solution for farmers. However, limitations such as dataset bias and computational demands highlight areas for improvement.

Arabic Summary

ملخص بالعربيه

يهدف هذا المشروع إلى تطوير موقع ويب يستخدم تقنيات التعلم العميق لتوقع أمراض النباتات. الهدف هو توفير منصة سهلة الاستخدام للمزارعين وخبراء الزراعة للكشف المبكر عن الأمراض وتشخيصها بدقة، مما يعزز حماية المحاصيل وزيادة الإنتاجية.

يستخدم المشروع مجموعة بيانات "Modern Plant Disease" والتي تتضمن صورًا للنباتات السليمة والنباتات المتأثرة بأمراض مختلفة. من خلال تنفيذ شبكات التعلم العميق (CNNs)، يتم تدريب النماذج لتصنيف صور النباتات وتوقع وجود الأمراض. تم تصميم النماذج لاستخراج السمات المهمة من الصور وتعلم أنماط الأمراض.

تظهر النتائج التجريبية دقة ملحوظة، حيث يتم تحقيق أعلى دقة تبلغ 99.56% على مجموعة الاختبار. تعكس هذه الدقة العالية فعالية النماذج في تحديد الأمراض بدقة وتصنيفها بدقة. يقدم الموقع واجهة مستخدم سهلة الاستخدام، مما يتيح للمستخدمين تحميل الصور بسهولة واستلام التوقعات والوصول إلى معلومات إضافية.

من خلال توفير الكشف المبكر عن الأمراض وتشخيصها بدقة، يلعب الموقع دورًا حاسمًا في تمكين المزارعين من اتخاذ إجراءات سريعة، وبالتالي تقليل خسائر المحاصيل وتنفيذ استراتيجيات فعالة لإدارة الأمراض. يساهم ذلك في تعزيز الممارسات الزراعية المستدامة وتحقيق الأمن الغذائي.

في الختام، ينجح هذا المشروع في تطوير موقع ويب باستخدام تقنيات التعلم العميق لتوقع أمراض النباتات. تعكس الدقة العالية المحققة فعالية النماذج في تحديد الأمراض بدقة. تعزز واجهة المستخدم السهلة الاستخدام قابلية الاستخدام للمزارعين وخبراء الزراعة، مما يمنحهم أداة قيمة للكشف الفعال عن الأمراض واتخاذ القرارات

References

- **Backend SourceCode**

  :https://github.com/magdy3660/greenest-API
- **API Documentation:**
  https://www.postman.com/green-nest/greennest/documentation/tfub77m/gnes

- Using transfer learning-based plant disease classification and detection for sustainable agriculture | BMC Plant Biology | Full Text

- Plant Disease Detection using the PlantDoc Dataset and PyTorch

- Open Agriculture Journal Classification of Various Plant Leaf Disease Using Pretrained Convolutional Neural Network On Imagenet

- Transfer Learning Based Plant Diseases Detection Using ResNet50 | Request PDF

- Plant leaf disease classification using EfficientNet deep learning model - ScienceDirect

- Plant Village Dataset | TensorFlow Datasets

- Plant Disease Detection Using Transfer Learning with ResNet50

- https://yolov8.org/yolov8-architecture (yolov8 Architecture Deep Dive)
- Semantic segmentation for plant leaf disease classification and damage detection: A deep learning approach - ScienceDirect

- Detecting Pests & Plant Diseases with ResNet50 Model in the Application of Smart Farming and Agriculture | by Najma | Medium