Tribhuvan University

Academia International College

Final Year Project Report
On
Plant Disease Detection System(PDDS)
[CSC 412]

Under the supervision of

"Er. Ganesh Bhatta"

Submitted by

Ashmita Timalsina (T.U. Exam Roll No. 26482/077)

Anjal Ghimire (T.U. Exam Roll No. 26478/077)

Ruth Ghising (T.U. Exam Roll No. 26511/077)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology Tribhuvan University

January, 2025

Tribhuvan University

Academia International College

Final Year Project Report

On

Plant Disease Detection System(PDDS)

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

Submitted by

Ashmita Timalsina (T.U. Exam Roll No. 26482/077)

Ruth Ghising (T.U. Exam Roll No. 26511/077)

Anjal Ghimire (T.U. Exam Roll No. 26478/077)

Submitted to

Department of Computer Science and Information Technology Academia International College

Institute of Science and Technology Tribhuvan University

January, 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Mrs. Ashmita Timalsina(26506/077), Mrs. Ruth Ghising (26509/077), and Mr. Anjal Ghimire(26503/077) entitled "Plant Disease Detection System(PDDS)" be accepted as fulfilling in partial requirements for the degree of Bachelors of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

.....

Er. Ganesh Bhatta

Project Supervisor

Department of Computer Science and Information Technology

Academia International College

Gwarko, Lalitpur

Tribhuvan University

Department of Computer Science and Information Technology

Academia International College

Certificate of Approval

This is to certify that this project prepared by Mrs. Ashmita Timalsina, Mrs. Ruth Ghising, and Mr. Anjal Ghimire entitled "Plant Disease Detection System (PDDS)" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. Er. Ganesh Bhatta Project Supervisor Department of Computer Science and IT Academia International College	…….. Mr. Bishwas Mathema HOD/Program Coordinator Department of Computer Science and IT Academia International College
…….. Internal Examiner Academia International College	…….. External Examiner Central Department of CSIT Tribhuvan University

Acknowledgement

We are deeply grateful to Academia International College for giving us the opportunity to work on this project as part of our academic curriculum. This project has been an incredible learning journey, allowing us to apply what we've learned in the classroom to solve a real-world problem.

Our heartfelt thanks go to our supervisor, Mr. Ganesh Bhatta, for his unwavering guidance, support, and constructive feedback throughout this project. His encouragement and insights have been invaluable in helping us stay on track and bring our ideas to life.

We would also like to extend our sincere gratitude to our families, friends, colleagues, and teachers who stood by us during this journey. Their support, motivation, and helpful feedback were instrumental in completing this project within the given timeframe.

This experience has been more than just a project—it's been an opportunity to grow, learn, and understand the practical applications of our studies. We truly appreciate everyone who contributed to making this a success.

Thanking You,

Ashmita Timalsina (T.U. Exam Roll No. 26482/077)

Ruth Ghising (T.U. Exam Roll No. 26511/077)

Anjal Ghimire (T.U. Exam Roll No. 26478/077)

Abstract

The Plant Disease Detection System is a web-based application designed to help users identify plant diseases by uploading images of plant leaves. Using advanced machine learning techniques, the system combines Convolutional Neural Networks (CNNs) for extracting critical features like texture, color, and patterns, with the Random Forest algorithm for highly accurate disease classification. Once a disease is detected, the system provides detailed information about the disease, including its causes, preventive measures, and treatment recommendations. The backend and frontend are seamlessly integrated using the Django framework, while MongoDB Atlas is employed to store and manage user data, plant details, and disease-related information securely. The system is designed to be user-friendly, responsive, and accessible for farmers, agricultural researchers, and experts. By leveraging modern technology, this project aims to address challenges in agriculture by promoting early disease detection and better crop management. This project demonstrates how machine learning and web technologies can be combined to provide practical solutions to real-world problems, particularly in the agricultural sector, contributing to sustainable farming practices.

Keywords: Plant Disease Detection System, Python-Django, MongoDB, Convolutional Neural Network, Random-forest.

Table of Contents

List of Figures

List of tables

List of Abbreviations

CNN	Convolutional Neural Networks
DFD	Data Flow Diagram
EDA	Exploratory Data Analysis
ER	Entity Relationship
FPR	False Positive Rate
PDDS	Plant Disease Detection System
SVM	Support Vector Machines
TPR	True Positive Rate
UI	User Interface
UX	User Experience
VS Code	Visual Studio Code

Chapter 1: Introduction

1.1 Introduction

The Plant Disease Detection System (PDDS) , a web application is a user-friendly platform designed to help you easily identify plant diseases by simply uploading a photo. The system uses advanced technology like Convolutional Neural Networks (CNNs) to analyze your plant image, looking for key features such as color, texture, and patterns that point to specific diseases.

Once these features are extracted, the data is passed through a Random Forest model, which uses a group of decision trees to make an accurate diagnosis. This combination of CNNs and Random Forest ensures the system can detect diseases with high precision, even catching subtle signs that may not be obvious at first glance.

After identifying the disease, the website provides detailed information, including the symptoms, causes, prevention tips, and treatment suggestions, helping you take quick and informed action.

Whether you're a farmer, gardener, or agricultural professional, this platform makes disease detection simple and accessible. By offering reliable, timely results, it helps promote healthier crops and supports sustainable farming practices, improving plant care and overall yield.

1.2. Problem Statement

Managing plant health can be a challenging task for farmers, gardeners, and agricultural professionals, especially when it comes to identifying diseases. Early detection is critical, but it can be difficult to recognize plant diseases without expert knowledge. Traditional methods of diagnosis often rely on visual signs or external help, which can be time-consuming, inefficient, and prone to error. Some key issues in plant disease detection include:

Delayed Diagnosis: Diseases may go unnoticed until they cause significant damage, leading to crop loss and reduced yield.

Lack of Expertise: Identifying plant diseases often requires specialized knowledge that not everyone has access to.

Manual and Slow Processes: Diagnosing plant diseases typically involves manual inspection, which is both slow and subject to human error.

Limited Access to Timely Information: Once a disease is detected, finding accurate, detailed information about it, including symptoms, prevention, and treatment options, can be a challenge.

1.3 Objective

The core objectives of Plant Disease Detection System (PDDS) are:

- To create a user-friendly platform where users are able to upload the image of plant leaves and get the results
- To provide information about the disease name, its description, preventive measure and treatment.

1.4 Scope and Limitations

1.4.1 Scope

The scope of PDDS include:

- Quick Detection: Users can upload a photo of a plant and press the "Detect" button. The system will provide basic details about the disease instantly.
- Full Details for Logged-in Users: To access complete information, like causes, symptoms, and treatment tips, users need to log in. After logging in, they can re-test the image to get the full diagnosis.
- User-Friendly and Helpful: Easy to use for anyone, from casual gardeners to professional farmers.
- Encourages Learning: Partial info encourages users to explore and engage further.

1.4.1.1 Future Scope

The future scope of PDDS include:

- User History: Users will be able to securely store and review their plant photos and disease predictions, making it easy to track plant health over time.
- More Resources: The platform will offer more helpful guides on plant care and pest management to help users better understand plant health.
- Expert Support: Users will be able to connect with agricultural experts for personalized advice, ensuring they get the right help when needed.

1.4.2. Limitations

- Limited Accuracy: The initial detection might not always be 100% accurate, as it depends on the quality of the uploaded photo and the system's ability to recognize the disease correctly.
- Re-testing Requirement: Users need to log in and re-upload the photo to access full information, which could be inconvenient for those seeking immediate, detailed answers.
- Internet Dependence: Since the system relies on online features like photo uploads and expert connections, users with slow or no internet access may face difficulties in using the platform effectively.
- Limited Plant and Disease Coverage: The system cover only limited plant and diseases, especially rare or less common ones, leading to incomplete diagnoses in some cases.
- Photo Quality Impact: The accuracy of the detection is highly dependent on the quality of the uploaded photo. Poor lighting, blurry images, or low resolution can affect the system's ability to correctly identify the disease.

1.5. Methodology

Agile Software Development is a software development methodology that values flexibility, collaboration, and customer satisfaction.[1] It is based on the Agile Manifesto, a set of principles for software development that prioritize individuals and interactions, working software, customer collaboration, and responding to change. [1]

We adopted Agile methodology to developed our project. It seems to be best methodology for our project to regularly supervised by our supervisor and implement feedbacks to improve project. This approach allowed us to work iteratively, adapt to changes, and incorporate regular feedback to ensure a high-quality outcome.

We started by discussing the project requirements with our supervisor, breaking them into smaller tasks, and prioritizing them. The project was divided into different tasks, with clear goals set for each tasks. Further we designed frontend according to the requirement which we were discussed in the begin of project. The backend involved integrating the CNN model and Random Forest model with our Django application. The detection model was trained using a dataset of 29,000 images across 22 classes, with ongoing improvements during each phase of model training process.

Regular meetings with our supervisor helped us refine the system based on feedback. Different aspects of improvement of project had been discussed such as user usability, model accuracy etc. in the meeting and implemented precisely to the system. Once all components were integrated, we conducted user testing and deployed the system.

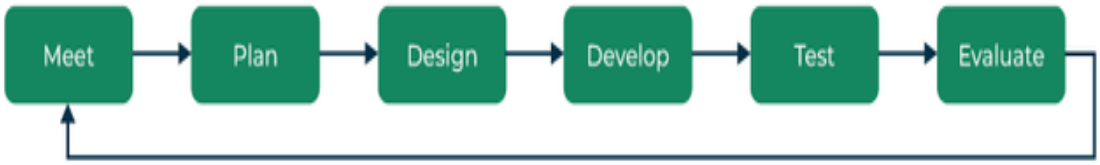


Figure 1.1: Agile Software Development Cycle

1.6. Report Organization

This report is structured into six key chapters, each focusing on a specific part of the project to ensure a logical and organized flow of information. The chapters are as follows:

Chapter 1: Introduction

The first chapter sets the stage for the entire report by introducing the project. It covers the background, defines the problem being addressed, states the objectives, and identifies the scope and limitations of the study. Additionally, it explains the methodology followed during the development process and provides an overview of how the report is structured.

Chapter 2: Background Study and Literature Review

Chapter 2 provides the theoretical foundation of the project by explaining key concepts, theories, and terminologies. It also includes a comprehensive review of similar projects and relevant studies by other researchers, helping to establish a connection between existing knowledge and the goals of this project.

Chapter 3: System Analysis

This chapter dives into the core analysis phase of the system development. It includes a detailed discussion on functional and non-functional requirements, along with a feasibility analysis that examines technical, operational, economic, and schedule-related factors. Moreover, this chapter utilizes models like ER diagrams and DFDs to analyze data and system processes.

Chapter 4: System Design

The design chapter focuses on the planning and blueprinting phase of the system. It covers database design (transforming ER diagrams into relational models and ensuring proper normalization), the design of forms and reports, and the creation of intuitive user interfaces. Additionally, this chapter provides insights into the algorithms that drive the system's functionality.

Chapter 5: Implementation and Testing

This chapter discusses how the system was implemented using specific tools and technologies. It explains the development modules in detail, highlighting the methods, classes, and processes involved. Furthermore, the testing process is covered, including unit testing and system testing, with results analyzed to ensure that the system performs as expected.

Chapter 6: Conclusion and Future Recommendations

The final chapter summarizes the key findings and achievements of the project. It provides a reflection on the project's objectives and the extent to which they were met. Additionally, this chapter offers suggestions and recommendations for future work, aiming to build on the project's findings and improve its outcomes.

Chapter 2: Background Study and Literature Review

2.1. Background Study

In the context of Nepal, Agriculture contributes 24.0 percent to the Gross Domestic Product (GDP) and about 67.0 percent of the total population resides within agricultural families.[2]The major method of plant disease detection in remote areas of developing country like Nepal is still ocular inspection. Like human, plants also suffer from many diseases that badly affect their normal growth. Identifying plant diseases is important in order to prevent the loses within the yield.

Because of huge number of plants and complexity, there are also many numbers of diseases. Its terribly problematic to observe the plant diseases manually. It requires a huge amount of labor, expertise within the plant diseases and collectively requires excessive time interval. Sometimes it also makes difficulties for experts to evaluate disease in plant by observing because of similarities scenarios of different disease in plants. Researcher also need software like plant disease detection to help and do research efficiently and timely. That's why there is a requirement of advance system that features plant disease detection by uploading picture.

In this project, we will be implementing the concept of image processing and machine learning for the detection of plant diseases with the help of their leaves pictures and will be recommending treatments and preventive measures. We will be using CNN model for feature extraction and random forest for classification. And a easily accessible and easily navigating design to make easy for users to detect disease with more accuracy.

2.2. Literature review

Plant Disease Detection Using Deep Learning and Attention Mechanism [3]. This study explores the use of deep learning models integrated with attention mechanisms to enhance the accuracy of plant disease detection. The researchers propose a convolutional neural network (CNN) model that leverages the attention mechanism to focus on the most relevant parts of plant images. The model is trained and tested on a diverse dataset of plant leaf images.

An Accurate Plant Disease Detection Technique Using Machine Learning [4]Gupta et al. propose a machine learning-based technique for the accurate detection of plant diseases. The approach involves extracting features from plant leaf images and using them as input

to various machine learning classifiers, including Support Vector Machines (SVM) and Random Forest. The study demonstrates that the combination of feature extraction methods like Histogram of Oriented Gradients (HOG) with advanced classifiers significantly improves the detection accuracy. The authors also highlight the importance of dataset quality and the role of data augmentation in enhancing model performance.

Plant Disease Detection Using Convolutional Neural Networks [5] This paper by Aslan et al. investigates the application of convolutional neural networks (CNNs) for plant disease detection. The authors designed a CNN model specifically tailored to identify diseases in various plants based on leaf images. They experimented with different CNN architectures and found that deeper networks tend to perform better but at the cost of increased computational complexity. The paper discusses the trade-offs between model accuracy and computational efficiency, emphasizing the need for balance in real-world applications where resources may be limited. The results indicate that CNNs are highly effective for plant disease detection, achieving impressive accuracy across multiple plant species.

Plant Disease Detection using Image Processing [6]. Dey et al. present an image processing-based approach for detecting plant diseases. The methodology involves segmenting plant images to isolate diseased areas and then applying feature extraction techniques to identify the type of disease. The study highlights the effectiveness of techniques like K-means clustering for segmentation and Gabor filters for feature extraction. The authors compare their image processing approach with machine learning-based methods and conclude that while machine learning offers superior accuracy, image processing techniques provide a faster and computationally less intensive alternative. This makes them suitable for use in low-resource environments where high computational power is unavailable.

Chapter 3: System Analysis

3.1. System Analysis

Our Plant Disease Detection System is designed to help users identify plant diseases accurately and efficiently. The system uses a machine learning model trained on a dataset of 29,000 images to predict diseases in plants. It features an intuitive web interface built using Django, allowing users to upload images and receive results quickly. The system aims to assist farmers, gardeners, and agricultural experts in diagnosing plant health issues and taking timely action.

3.1.1. Requirement Analysis

The system requirement can be functional requirements and non-functional requirements.

i. Functional requirements

The system considered following functional requirements:

- User Authentication: Users can register and log in to access the system.
- Image Upload: Users can upload plant images for disease detection.
- Disease Detection: The system processes uploaded images and provides predictions about the disease.
- Result Display: Detected disease information, preventive measures and treatment methods are displayed to the user.

ii. Non-functional requirements

The system considered following non-functional requirements:

- Latency: The system ensures predictions are generated within few seconds of image upload, providing users with real-time results.
- Usability: The interface offers seamless and user-friendly image upload functionality, prioritizing ease of use and accuracy.

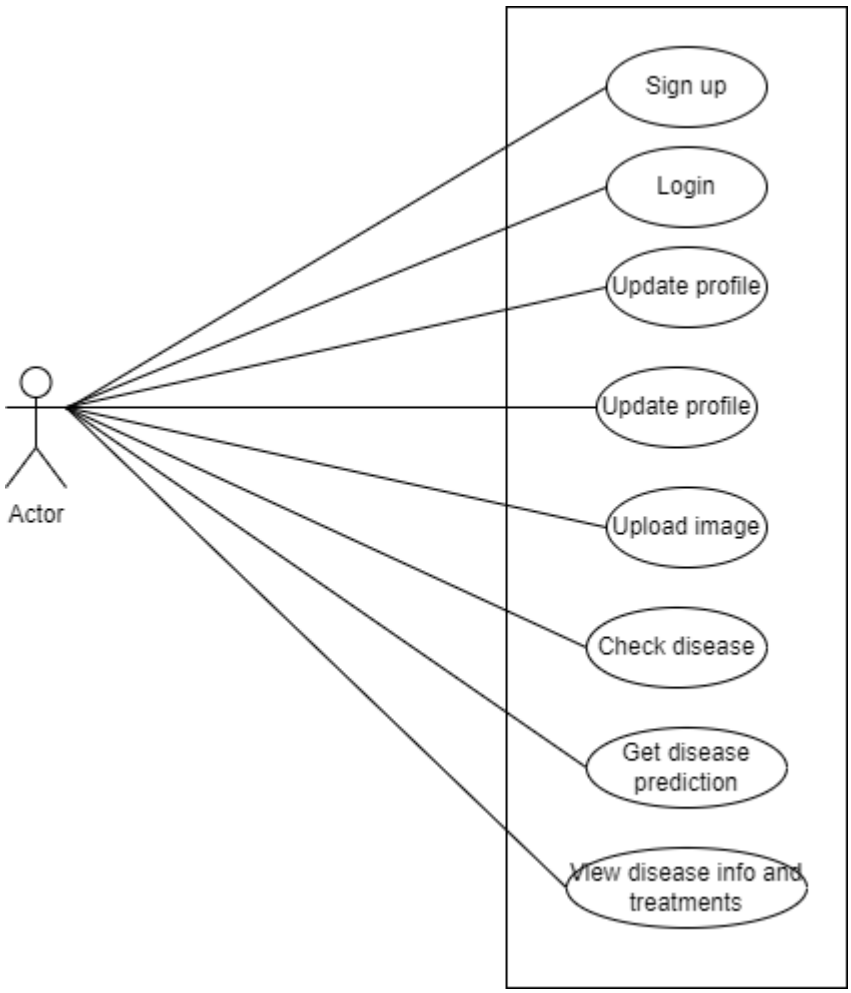


Figure 3.1 Use case diagram

3.1.2. Feasibility study

i. Technical Feasibility

Technical feasibility evaluates the extent of compatibility of the current technology with the requirements of plant disease detection system. Modern machine learning algorithms along with deep learning models such as Convolutional Neural Network(CNN) that performs well in image recognition of plant photos, these models can be used to accurately point out illnesses Due to applying web application development framework Django, and machine learning tools, such as TensorFlow or Keras, it has a solid technical foundation. Computational demands such as data processing and model training can be carried out with modern computational systems thus making the technical application of the project possible.

ii. Operational Feasibility

Operationality focuses on how effectively the plant diseases detection system will integrate and suggest the possible preventive measures and treatments. This involves developing a front end for a plant disease detection system that would allow users to capture pictures of plants leaves as well as get immediate responses on whether their plants are infected or not. Due to Django, the system can be easily used by people with various levels of technical background since Django is very tolerant to the creation of versatile and logical interfaces for web applications. The system is useful in activities carried out in the agricultural sector since it could also assist in the identification and management of diseases at an early stage which enhances it's operational applicability. [1]

iii. Economic Feasibility

The economic feasibility of the cost of developing as well as the development of the plant disease detection system is explored. The project is financially viable because the project will employ first Django and second Python-based machine learning tools that are open sources such as Tensorflow for model building and for the data set we will be using free available datasets from Kaggle /plant village.

iv. Schedule

Table 3.1. Project schedule

Tasks	Start date	End date	Duration
Paper Study/Discussion	8/3/2024	8/7/2024	4
Proposal Writing	8/7/2024	8/14/2024	7
System Design	8/15/2024	8/23/2024	8
Data Collection/Preprocessing	8/24/2024	8/26/2024	2
Model training / development	8/27/2024	10/7/2024	41
System Development	10/8/2024	11/21/2024	44
Testing	11/22/2024	12/1/2024	9
Final Review/Report writing	12/2/2024	12/11/2024	9

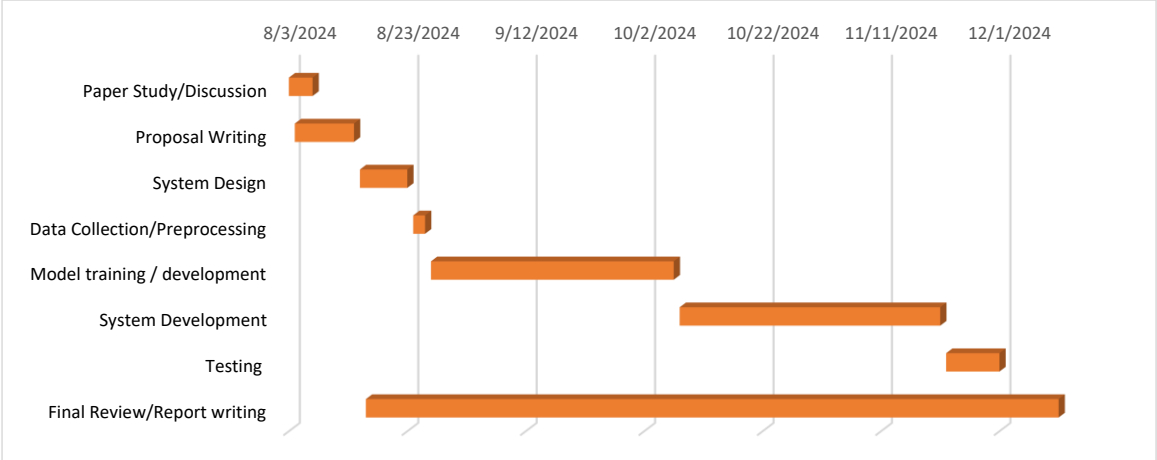


Figure 3.2 Project Gantt chart

3.1.3. Data Modeling using ER Diagrams

The ER diagram illustrates a robust system designed to empower users with detailed insights into plant diseases and related preventive measures. It features a well-structured architecture that connects users, their profiles, plants, and disease data through meaningful relationships.

Entities and Attributes

Users**:**

Attributes: Full name, Email, Phone, Address, and Gender.

This entity represents the core of the system, capturing essential user details required for interaction and personalization.

Profile_Details**:**

Attributes: id, user_id, bio, profile_picture, and occupation.

Linked to the Users entity via the has relationship, it enhances the user experience by storing supplementary profile information.

Plant_Info**:**

Attributes: Id, plant_name, image_url, and facts.

This entity holds a repository of plants, each accompanied by an image and interesting facts, creating a visual and educational database for users.

Disease_Data**:**

Attributes: _id, disease name, description, prevention, and treatment.

Central to the system, this entity provides comprehensive information about plant diseases, including their symptoms, preventive measures, and available treatments.

Relationships

Users and Profile_Details**:**

Relationship: has.

Highlights the connection between users and their personalized profile data.

Users and Plant_Info**:**

Relationship: View.

Establishes that users can explore plant-related information, enhancing their knowledge through slider in home page.

Users and Disease_Data**:**

Relationship: Can detect.

Emphasizes the system's functionality, enabling users to identify specific plant diseases and access critical details.

This ER diagram reflects an intelligent system aimed at bridging the gap between users and plant health knowledge. Its intuitive design ensures accessibility while prioritizing user engagement, making it a valuable tool for anyone interested in plants and their well-being.

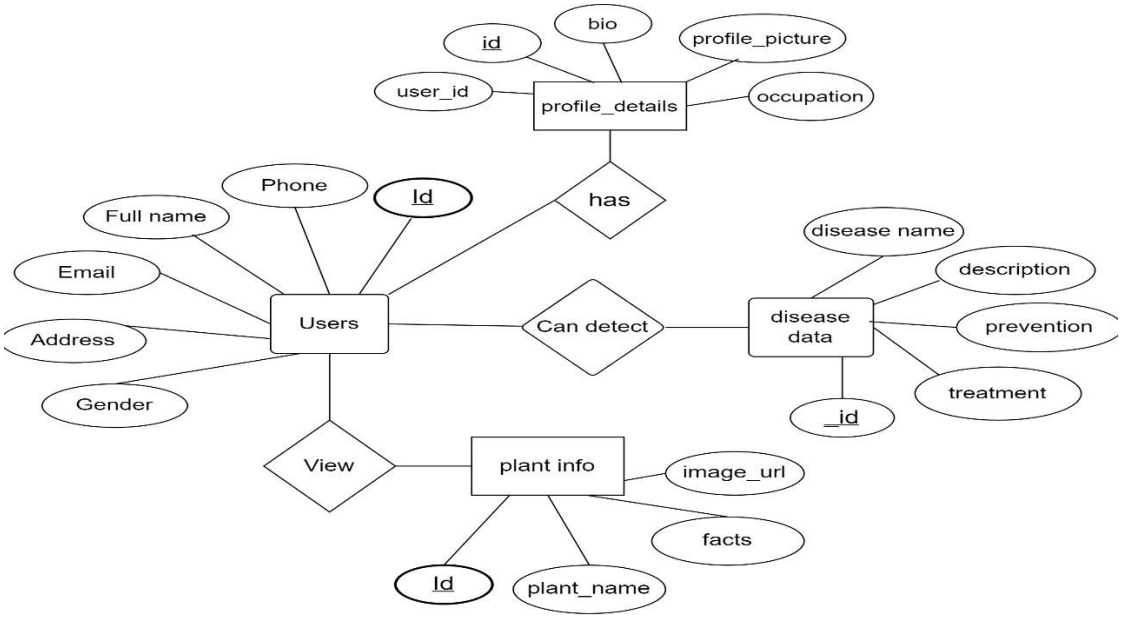


Figure 3.3 ER diagram of Plant Disease Detection System

3.1.4. Process Modeling using DFD

The processes like image upload, disease detection, and the flow of results between users and the system is shown below in form of Data Flow Diagram (DFD).

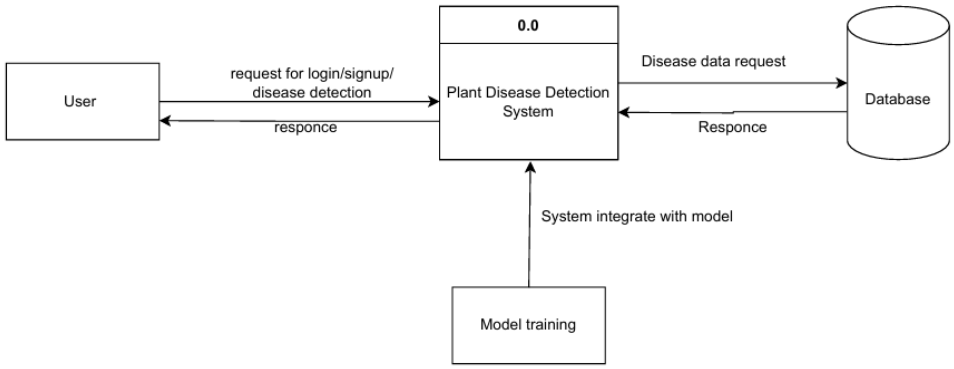


Figure 3.4 Level-0 DFD (Context Diagram) of Plant Disease Detection System

The Level 0 DFD provides an overview of the system as a single process. Here's how it works:

User Interaction:

Users can request operations such as login, signup, or plant disease detection by uploading an image.

The system processes the requests and provides appropriate responses, such as confirming login or displaying detection results.

Plant Disease Detection System:

This central process handles user requests, communicates with the database to fetch disease-related data, and integrates with a trained machine learning model for image-based disease detection.

Model Training:

The detection system integrates with a trained machine learning model to analyze images uploaded by users, enabling accurate disease prediction and classification.

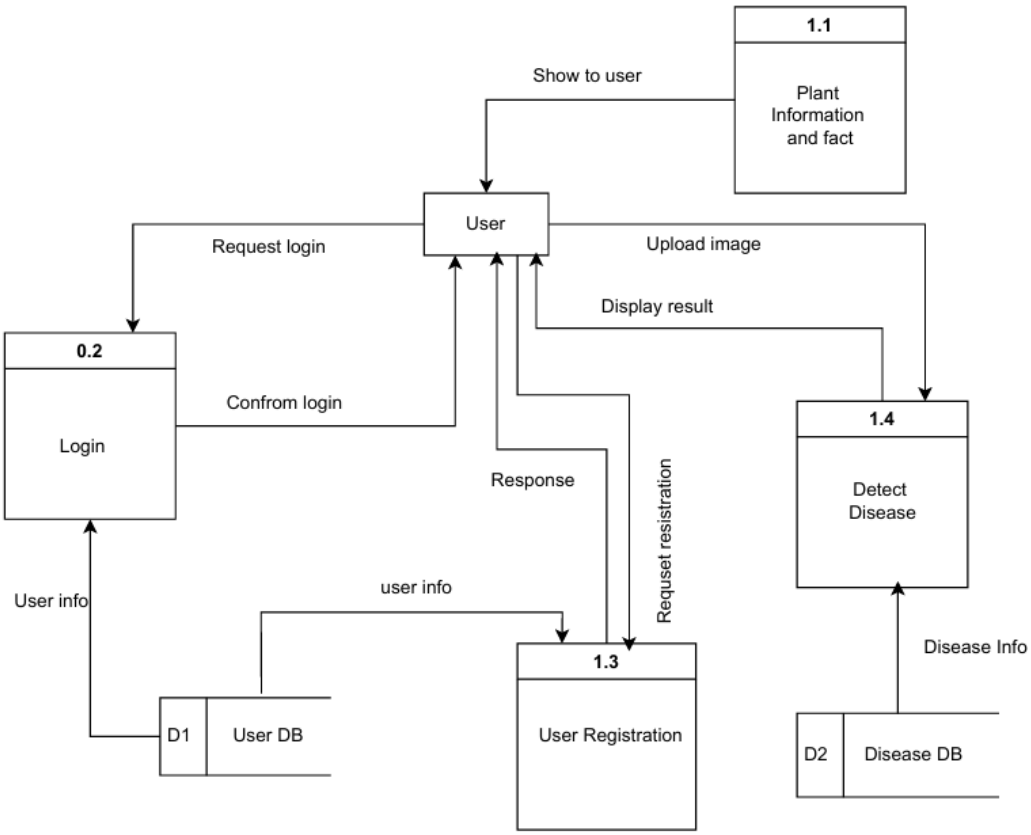


Figure 3.5 Level-1 DFD of Plant Disease Detection System

The Level 1 DFD dives deeper into specific processes and their data flows:

Login Process (1.2):

Users send a login request with their credentials.

The system validates the credentials using data from the User DB (D1).

If valid, the system confirms the login and grants access. Otherwise, it prompts the user to try again.

User Registration (1.3):

New users can register by providing their details.

The system stores this information in the User DB for future authentication.

Detect Disease (1.4):

Users upload an image of a plant.

The system uses the integrated machine learning model to analyze the image and detect diseases.

It fetches detailed disease information (e.g., details, prevention, and treatment) from the Disease DB (D2) and displays it to the user.

Plant Information and Facts (1.1):

The system also provides users with general plant-related information and facts, retrieved from the database.

This feature is designed to educate users about plants and their health.

Chapter 4: System Design

4.1 Design

The Plant Disease Detection System (PDDS) is built to make identifying plant diseases easy and accessible for everyone. We focused on creating a system that's simple to use, smart in its disease detection, and able to provide users with the right information when they need it.

4.1.1 Architectural Design

The PDDS, utilizes the three-tier architecture, implementing client-Server pattern. Here the presentation layer provides a user-friendly interface to client, and the application layer ensures the accuracy of the disease tested by the user whereas the data layer securely manages and retrieves all the important data related to the detected result. Together, these layers work harmoniously to provide users with fast, accurate results and valuable insights all in one easy-to-use system.

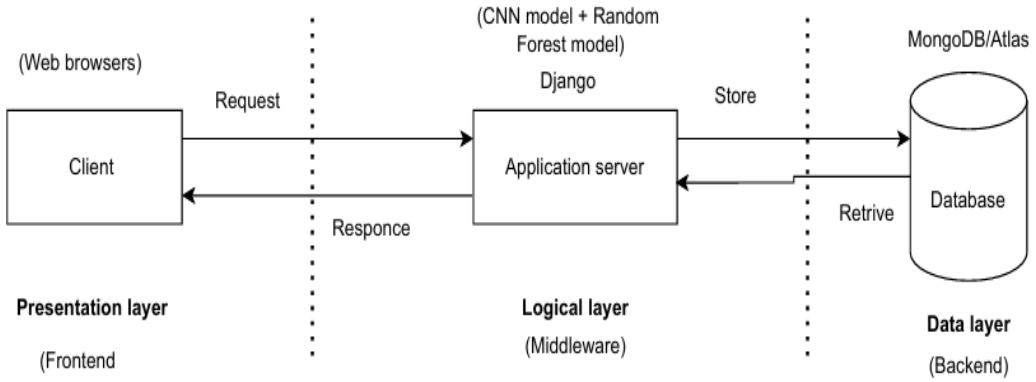


Figure 4.1 Three tier architecture of PDDS

4.1.2 Database Design

The accomplishment of the PDDS strongly relies on its effectively designed database, which is required to store user information, disease-name along with their causes prevention and treatment and facts about the plants. MongoDB Atlas a NoSQL database is used for the proper management of the information related to this project.

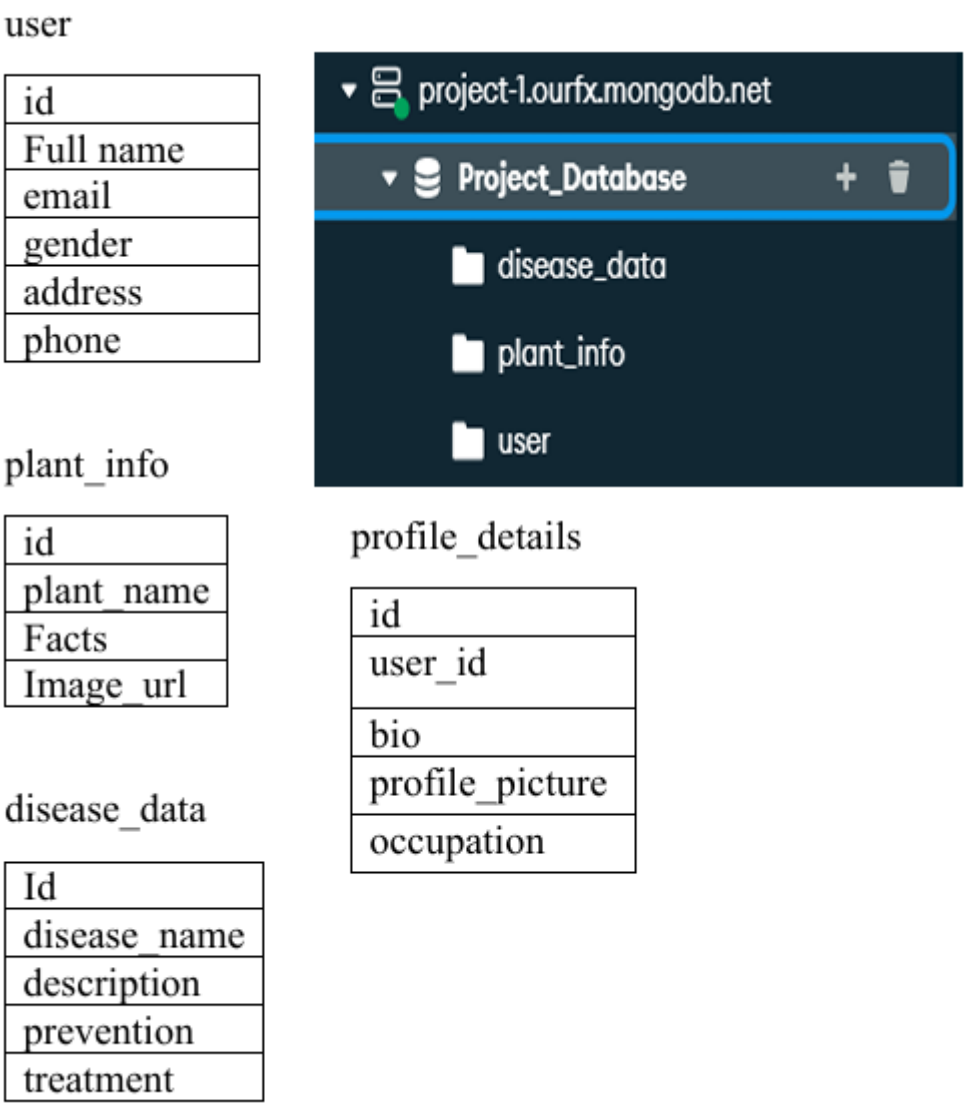


Figure 4.2 Database design of PDDS

4.1.3 Forms and Interface Design

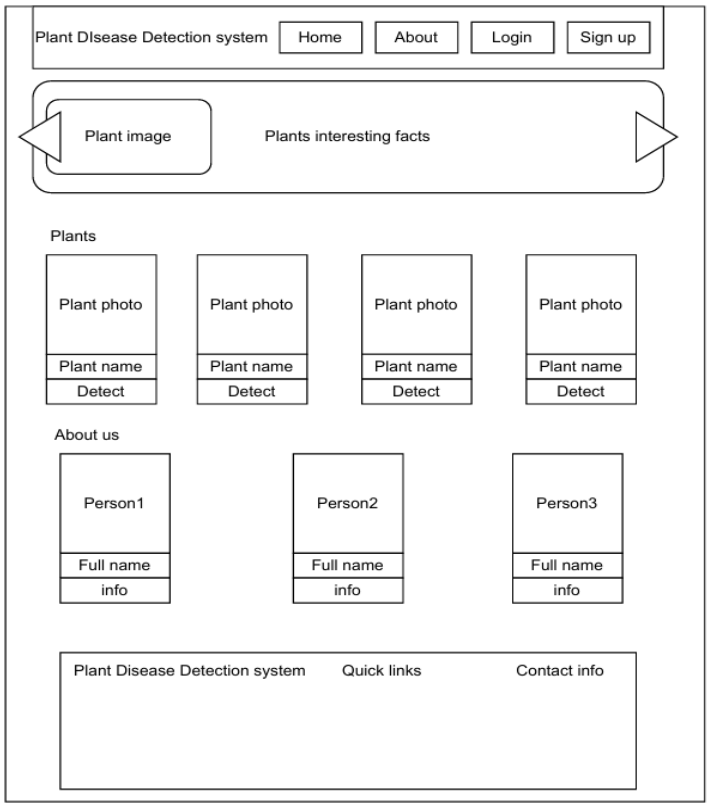


Figure 4.3 Home page design

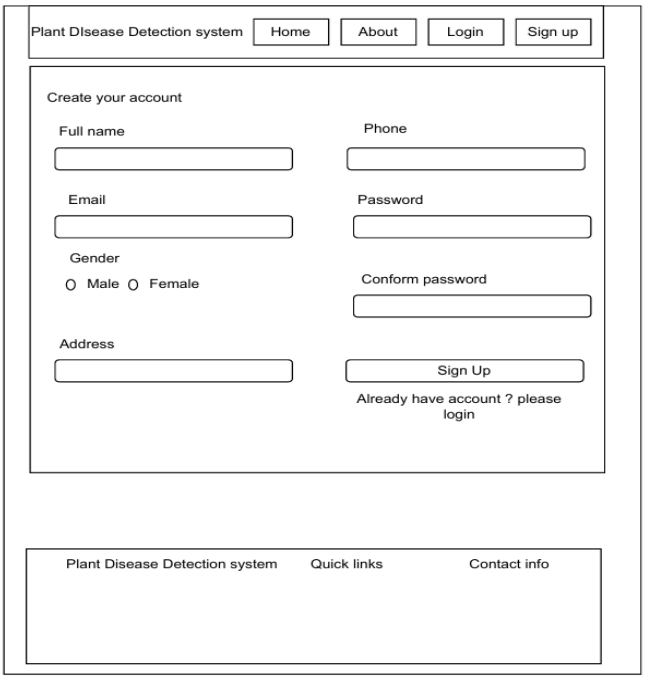


Figure 4.4 Sign up page design

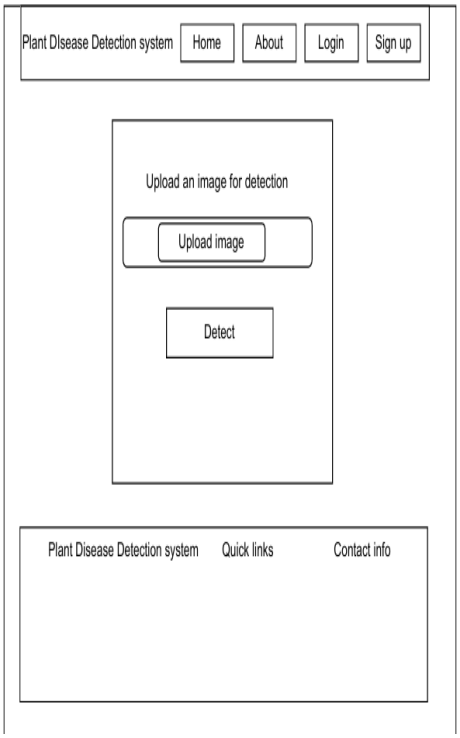


Figure 4.5 Upload image design

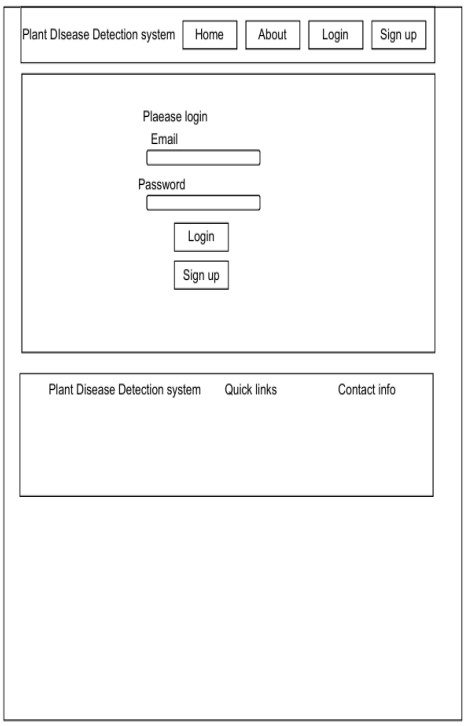


Figure 4.6 Login design

4.2 Algorithm Details

4.2.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. [7]For example visual datasets like images or videos where data patterns play an extensive role.[7]

4.2.1.1 CNN Architecture

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers[7]. The Convolutional layer applies filters to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction[7]. The network learns the optimal filters through backpropagation and gradient descent.[7]

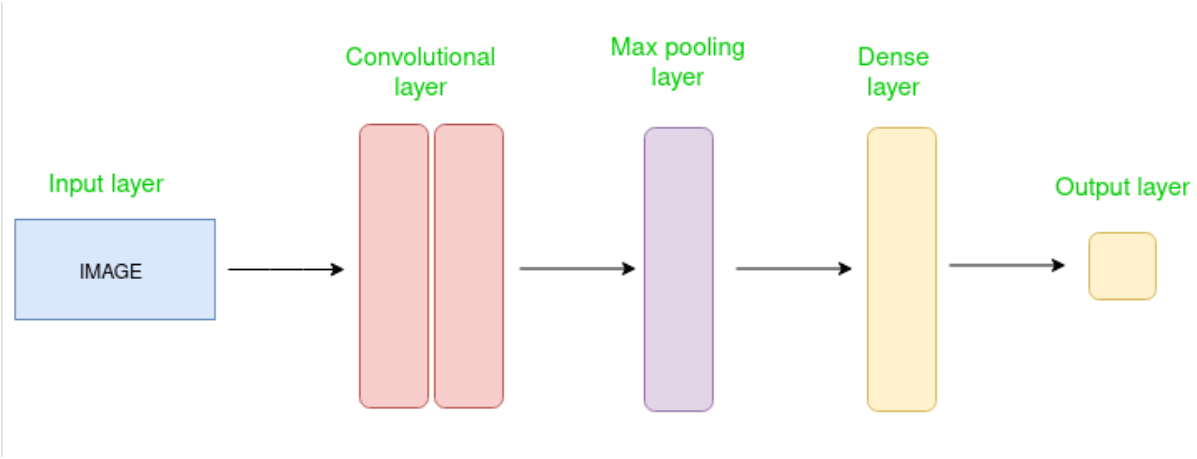


Figure 4.7 CNN Model Architecture

4.2.1.1.1 Build CNN Model

Here in PDDS, a Convolutional Neural Network (CNN) model is build using a sequential class , where each layer works together to process and classify images. The layer used in our CNN models are:

- Input Layer (shape=(256, 256, 3)): This is where the images enter the model. The model expects images that are 256 pixels tall, 256 pixels wide, and have 3 color channels (Red, Green, Blue).
- Convolutional Layers (Conv2D): These layers are like "filters" that scan the image to find important features like edges, shapes, or textures. The number of filters (32, 64, 128, 256) decides how many different features the model will look for in the image. Each filter scans the image in a 3x3 block.
- Activation Function (ReLU): After each convolution, the ReLU function is applied to add non-linearity to the model. This helps the model learn more complex patterns in the data.
- MaxPooling Layer: This layer helps reduce the size of the image while keeping important features. It takes the most important part (the maximum value) from each 2x2 block of pixels, making the model more efficient and reducing the amount of data to process.
- Batch Normalization: This helps the model train faster and more stably by normalizing the data before passing it to the next layer.

- Global Average Pooling: Instead of keeping the full feature map, this layer takes the average of all features, simplifying the data and making it easier for the model to make a decision.
- Fully Connected Layers (Dense): These layers take the processed features and start making sense of them by combining them into more complex representations. The first dense layer has 1024 neurons, helping the model understand deeper patterns.
- Dropout Layer: To avoid the model from memorizing the training data too well (which can lead to poor performance on new data), this layer randomly ignores 50% of the neurons during training, forcing the model to generalize better.
- Output Layer (Dense with SoftMax): Finally, the model has 22 neurons in the output layer, which corresponds to 22 possible classes. The SoftMax activation ensures that the model outputs a probability for each class, with the highest probability being the predicted class.

In simple terms, this model uses a series of layers to first look for basic features in images, then combines them to make a final prediction about what the image represents, choosing from 22 different possibilities. Here we have removed the prediction layer later as we are using random forest for classification.

4.2.2 Random Forest Algorithm

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset[8] Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.[8]

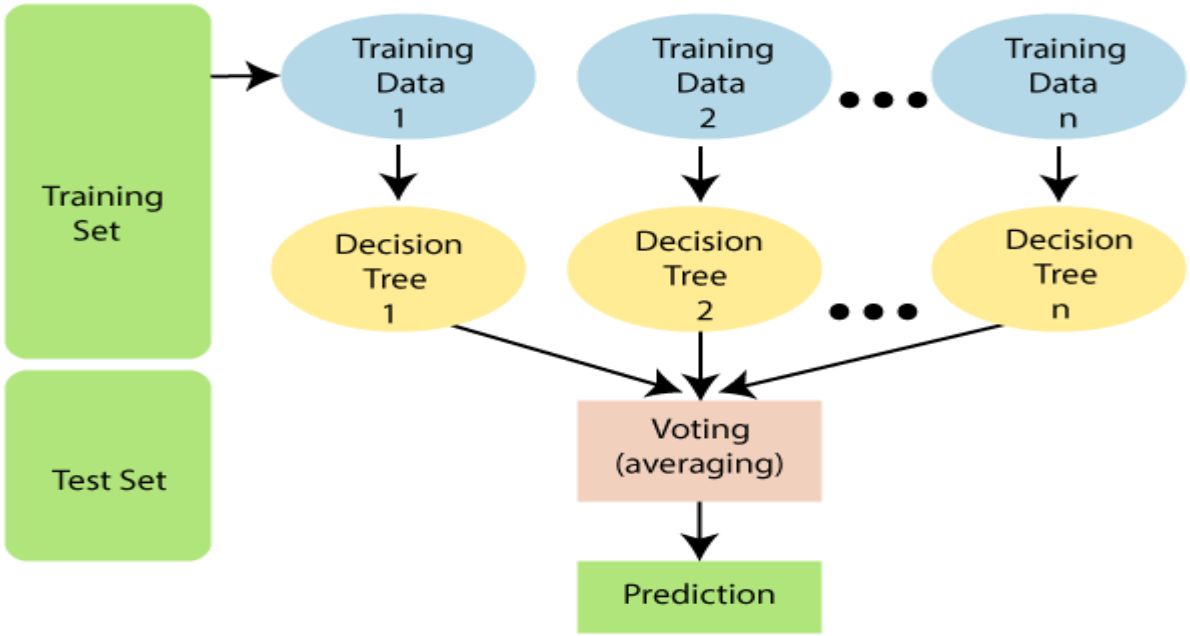


Figure 4.8 Working of Random Forest Algorithm

4.2.2.1 Random Forest Model

The Random Forest model is like a team of decision-makers, with each decision tree making its own choice. The model has used 200 of these decision trees to combine their individual decisions and come up with the final prediction. More trees generally help improve accuracy, but they also demand more computing power. Additionally, the model is set up so that the results are consistent, no matter how many times the code is run, ensuring reproducibility.

- Training the Model: During the training phase, the model learns from past examples. It looks at the input data (like images or features) and the correct answers (labels) that go with them. Over time, the Random Forest gets better at spotting patterns and trends in the data, which helps it make predictions on similar data in the future.
- Making Predictions: Once the model is trained and has learned from the data, it can be tested with new information that it hasn't seen before. The model uses the patterns it learned during training to predict the outcomes for this new data.
- Measuring Accuracy: To see how well the model is performing, we compare its predictions to the actual results. The closer the predicted values are to the real answers, the higher the accuracy of the model. This accuracy is then shown as a percentage, telling us how often the model gets things right.

4.2.3 Evaluation Matrix

When evaluating how well a classification model works, we rely on certain metrics to assess its consistency and correctness. Let's break them down in simple terms:

1. True Positive (TP):
   This is when the model gets it right and accurately predicts something as positive. For example, if the model identifies an image of a diseased plant correctly, that's a true positive.
2. False Positive (FP):
   Here, the model makes a mistake and predicts something positive when it's not. Imagine the model saying a healthy plant is diseased—that's a false positive.
3. True Negative (TN):
   This happens when the model correctly predicts something as negative. If the model correctly identifies that a plant is healthy, that's a true negative.
4. False Negative (FN):
   A false negative occurs when the model misses a positive and predicts it as negative. For instance, the model might fail to identify a diseased plant and label it as healthy.

4.2.3.1 Accuracy

Accuracy gives a general idea of how often the model is correct. It's the proportion of all correct predictions (both positives and negatives) to the total predictions.

Formula:

$$\text{Accuracy} = (TP + TN)/(TP+TN+FP+FN)$$

4.2.3.2 Pression

Precision tells us how often the model is correct when it predicts something as positive. For example, if the model flags 100 plants as diseased, precision tells us how many of those actually are diseased.

Formula:

$$\text{Pression} = TP/(TP+FP)$$

4.2.3.3 F1 Score

The F1 Score balances precision and recall. It's useful when there's an uneven distribution of positive and negative cases, as it considers both false positives and false negatives.

Formula:

F1 Score = 2*((Precision*Recall)/(Pression+Recall))

4.2.3.4 Recall

Recall focuses on how many actual positives the model identifies correctly. It's like asking: out of all diseased plants, how many did the model catch?

Formula:

Recall =TP/ (TP +FN)

4.2.3.5 ROC Curve

Receiver operating characteristics (ROC) curves are graphs showing classifiers' performance by plotting the true positive rate and false positive rate [9]. The area under the ROC curve (AUC) measures the performance of machine learning algorithms. ROC curves visually depict the statistical accuracy of classifier selection, but the graph's original use began in signal detection[9].

ROC curves are typically used in binary classification, where the TPR and FPR can be defined unambiguously [10]. In the case of multiclass classification, a notion of TPR or FPR is obtained only after binarizing the output[10]. This can be done in 2 different ways:

- The One-vs-Rest scheme compares each class against all the others (assumed as one)[10].
- The One-vs-One scheme compares every unique pairwise combination of classes[10].

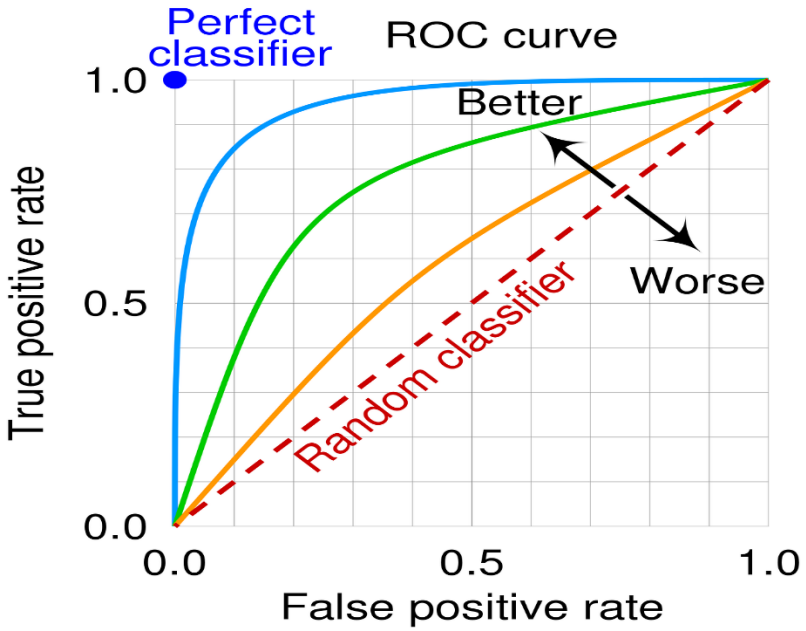


Figure 4.9 Roc Curve

4.2.3.6 Confusion Matrix

A confusion matrix visually represents the model's performance. It's a table showing how well the model predicted each category—correctly or incorrectly. This table not only highlights where the model succeeds but also where it struggles, helping us understand the types of errors being made.

For example:

- True Positives and True Negatives reflect the model's correct predictions.
- False Positives and False Negatives reveal where it went wrong.

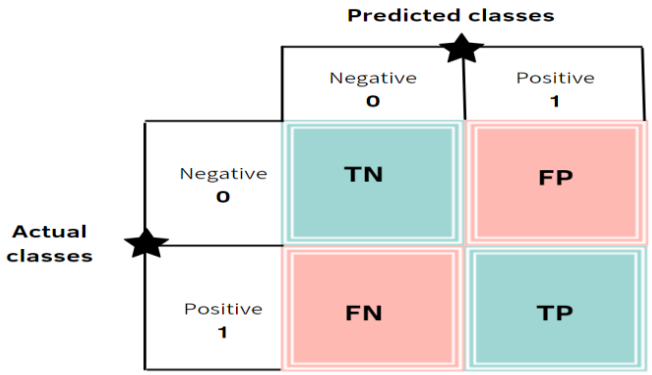


Figure 4.10 Confusion Matrix

Chapter 5: Implementation and Testing

5.1 Implementation

Implementation of PDDS involves various processes including defining requirements and planning, selecting methodology, choosing a framework, designing, building frontend components, implementing backend functionality, testing and quality assurance, and creating and optimizing content.

5.1.1 Tools Used

- Python: Backend Programming Language.
- Django: Framework used for development of the project.
- Draw.io: For creating diagrams like DFD and ER
- TensorFlow: For building and training CNN model and Random Forest Model
- MongoDB Atlas: A NoSQL cloud-hosted version of MongoDB database service that simplifies deploying and managing your databases.
- Google Colab: For training CNN and Random Forest Model.
- Vs Code: IDE for writing and running code.
- Microsoft Word: For preparing documentation.
- Microsoft Excel: For creating Gantt chart

5.1.2 Implementation details of Modules

The detail about some of the modules implemented in our system are:

- User Management Module:

  This module allows new users to signup by creating new account and allows exixting users to login securely. It also allows user to log out to end their session safely . This module allows user to test of disease as well. Stores all user details like username, email, and password in the user collection.

- Plant Testing Module:

  This Module allows users to upload images of plant leaves and detect diseases. Here the two most powerful machine learning modules are used, CNN for feature extraction from the uploaded image and Random Forest for taking those extracted features as input and detect the disease. This module not only detects the disease but also displays the disease results to users, giving them quick insights. Connect

with the diseases_data collection to retrieve information about the identified disease, such as symptoms, causes, and treatments.

- Disease Information Module:
  This module provides users with detailed insights into plant diseases, including how to prevent and treat them. Automatically displays these details after the system detects a disease.Uses the diseases_data collection to store and retrieve all disease-related information.
- Session Management Module(cookie-based):
  This module based on cookie-based session handling keeps users logged in so they can access features smoothly. It ensures only logged-in users can use certain features, like testing plants. This ends the session properly when users log out, clearing their login state.Uses cookies (instead of sessions) to manage and track user states securely.

5.2. Testing

Testing is an essential part of the software development process, ensuring that the system is reliable, functional, and meets the desired quality standards. It involves a thorough examination of both individual components and the system as a whole to identify and address any potential issues.

5.2.1. Test Cases for Unit Testing

Unit testing focuses on checking each module or component independently to ensure it performs its specific task correctly.

Table 5.1. Test Cases for User Login/Sign up

Test case Id	Test case description	Steps to perform	Expected Outcome	Actual Outcome	Status
TC-01	User login with valid input	Email : anjalghimire890@gmail.com Password : 1234	Login successfully	As expected	Pass

TC-02	User login with invalid input	Email : anjal@gmail.com Password : 1111	Login failed display error message	As expected	Pass
TC-03	Register new user with valid inputs	Provide mail : ashmitatimalsina12@gmail.com Created password : 1234	Sign up Successfully and enter to home page	As expected	Pass
TC-04	Login new user with new data to check register works or not	Email : ashmitatimalsina12@gmail.com Password : 1234	Login successfully	As expected	Pass

Table 5.2 Test cases for disease detection model

Test case Id	Test case description	Steps to perform	Expected Outcome	Actual Outcome	Status
TC-01	Test disease of plant which is on the test option of our system(apple, grape, corn, tomato)	Upload image of apple which is infected by Apple_scab	Accurate prediction returned	As expected	Pass
TC-02	Test disease of plant which out of our system options of testing disease	Upload image of potato	Random output which is irrelevant to input	As expected	Pass

5.2.2. Test Cases for System Testing

System testing examines the entire system to confirm that all components work together harmoniously. This phase ensures that the integration of modules is seamless and that the system provides a smooth user experience

Table 5.3 Tast cases for Responsiveness of System

Test case Id	Test case description	Steps to perform	Expected Outcome	Actual Outcome	Status
TC-01	Validate connection between backend and detection model	Send processed data to the prediction model → Check the model's response	Prediction is accurate	As expected	Pass
TC-02	Test data flow from model to result display	Submit data → Process backend → Fetch prediction results → Verify output on the screen	Results are displayed clearly and accurately.	As expected	Pass
TC-03	Verify data flow between input form and backend	Fill in the input form with valid data → Submit → Check backend processing	Data is correctly validated, processed, and stored.	As expected	Pass

Table 5.4 Test Cases for Responsiveness

Test case Id	Test case description	Steps to perform	Expected Outcome	Actual Outcome	Status

TC-01	Test concurrent user access	Simulate multiple users accessing the system simultaneously	System performs smoothly without noticeable delays.	As expected	Pass
TC-02	Check UI adaptability	Interact with the UI by resizing the browser or switching between devices	UI adjusts seamlessly without glitches or misalignment.	As expected	Pass

Table 5.5 Test Cases for Usability Testing

Test case Id	Test case description	Steps to perform	Expected Outcome	Actual Outcome	Status
TC-01	Test page navigation	Navigate between various pages or modules of the application	Navigation is smooth, with no delays or broken links.	As expected	Pass
TC-02	Verify clarity of error messages	Enter invalid data in forms or simulate incorrect actions	Error messages are clear, helpful, and easy to understand.	As expected	Pass

5.3 Result Analysis

The Plant Disease Detection System is designed to accurately identify plant diseases and provide detailed information to users. To assess its performance and usability, various metrics and real-world scenarios were analyzed, providing insights into its effectiveness and areas for improvement.

5.3.1 Exploratory Data Analysis (EDA)

Step 1: Dataset Overview

This step introduces us to the dataset by identifying the number of unique plant disease types (22 in this case) and listing them. It provides an immediate sense of the dataset's scope and diversity. For example, it includes diseases like Apple___Cedar_apple_rust and Tomato___Late_blight, along with healthy plant categories. This helps set expectations for model training and highlights the variety of classes.

Number of plant disease types: 22

Classes: ['Apple___Cedar_apple_rust', 'Apple___healthy', 'Grape___Black_rot', 'Apple___Apple_scab', 'Apple___Black_rot', 'Grape___Esca_(Black_Measles)', 'Grape___healthy', 'Tomato___Septoria_leaf_spot', 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)', 'Corn___Cercospora_leaf_spot Gray_leaf_spot', 'Tomato___Spider_mites Two-spotted_spider_mite', 'Corn___Common_rust', 'Tomato___Target_Spot', 'Tomato___Tomato_mosaic_virus', 'Corn___healthy', 'Corn___Northern_Leaf_Blight', 'Tomato___Tomato_Yellow_Leaf_Curl_Virus', 'Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___healthy', 'Tomato___Late_blight', 'Tomato___Leaf_Mold']

Step 2: Count of Images per Class

In this step, we explore how many images belong to each plant disease type. A bar chart visually represents the distribution, which makes it clear that the dataset is imbalanced, with some classes (e.g., Tomato___Tomato_Yellow_Leaf_Curl_Virus) having thousands of images while others, like Tomato___Tomato_mosaic_virus, have relatively few. Understanding this distribution is essential for deciding whether techniques like oversampling or data augmentation are needed to balance the data.

Table 5.6 Count of Images per Class

No of classes	Class type	Image count
1	Apple___Cedar_apple_rust	275
2	Apple___healthy	1645
3	Grape___Black_rot	1180
4	Apple___Apple_scab	630
5	Apple___Black_rot	621
6	Grape___Esca_(Black_Measles)	1383
7	Grape___healthy	423
8	Tomato___Septoria_leaf_spot	1771
9	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1076
10	Corn___Cercospora_leaf_spot Gray_leaf_spot	513
11	Tomato___Spider_mites Two-spotted_spider_mite	1676
12	Corn___Common_rust	1192
13	Tomato___Target_Spot	1404
14	Tomato___Tomato_mosaic_virus	373
15	Corn___healthy	1162
16	Corn___Northern_Leaf_Blight	985
17	Tomato___Tomato_Yellow_Leaf_Curl_Virus	5357
18	Tomato___Bacterial_spot	2127
19	Tomato___Early_blight	1000
20	Tomato___healthy	1591
21	Tomato___Late_blight	1909
22	Tomato___Leaf_Mold	952

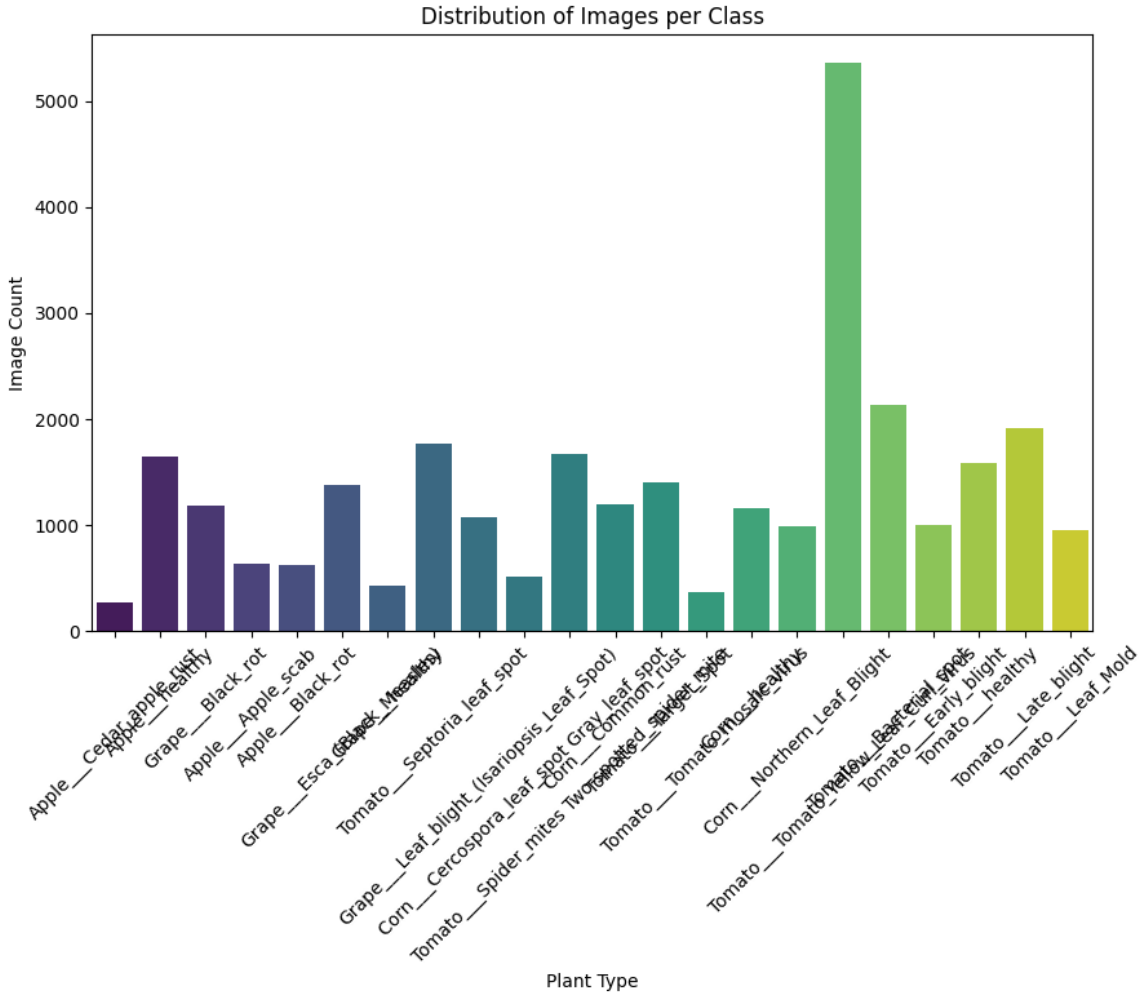


Figure 5.1 Distribution of image per classes

Step 3: Sample Images from the Dataset

Here, we visually inspect a few images from the dataset, showcasing the quality and appearance of the data. By plotting images from different classes, we get a quick sense of what the dataset looks like and whether any preprocessing (e.g., resizing, denoising) is necessary. Consistent image dimensions and clarity indicate a well-prepared dataset ready for further analysis.

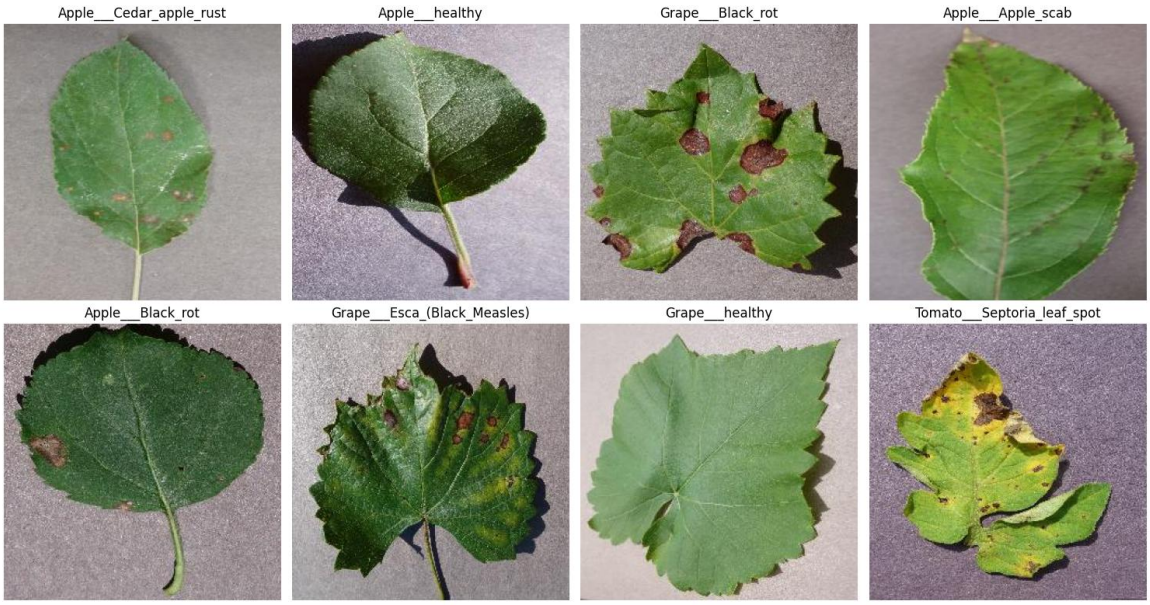


Figure 5.2 Sample image from dataset

Step 4: Image Dimensions and Color Channels Analysis

In this step we checks the technical properties of the images. All images have uniform dimensions of 256x256 pixels and 3 color channels (RGB). This consistency simplifies preprocessing since we won't need to handle varying image sizes or formats. Ensuring uniformity in dimensions and channels is critical for efficient model training.

	Plant Type	Dimensions (WxH)	\ Color Channels
0	Apple___Cedar_apple_rust	(256, 256)	3
1	Apple___healthy	(256, 256)	3
2	Grape___Black_rot	(256, 256)	3
3	Apple___Apple_scab	(256, 256)	3
4	Apple___Black_rot	(256, 256)	3
5	Grape___Esca_(Black_Measles)	(256, 256)	3
6	Grape___healthy	(256, 256)	3
7	Tomato___Septoria_leaf_spot	(256, 256)	3
8	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	(256, 256)	3
9	Corn___Cercospora_leaf_spot Gray_leaf_spot	(256, 256)	3
10	Tomato___Spider_mites Two-spotted_spider_mite	(256, 256)	3
11	Corn___Common_rust	(256, 256)	3
12	Tomato___Target_Spot	(256, 256)	3
13	Tomato___Tomato_mosaic_virus	(256, 256)	3
14	Corn___healthy	(256, 256)	3
15	Corn___Northern_Leaf_Blight	(256, 256)	3
16	Tomato___Tomato_Yellow_Leaf_Curl_Virus	(256, 256)	3
17	Tomato___Bacterial_spot	(256, 256)	3
18	Tomato___Early_blight	(256, 256)	3
19	Tomato___healthy	(256, 256)	3
20	Tomato___Late_blight	(256, 256)	3
21	Tomato___Leaf_Mold	(256, 256)	3

Figure 5.3 Image dimensions and color channel information

Step 5: Insights and Observations

This summary consolidates the key findings from the dataset:

- There are 22 distinct classes.
- The dataset contains 29,245 images in total.
- The image distribution shows imbalance on the basis of image count
- These observations guide the next steps, such as handling imbalance or verifying class labels.

Step 6: Pixel Intensity Distribution for a Sample Image

In this step, we analyze the grayscale intensity values of a sample image. The histogram illustrates how pixel intensities are distributed, giving insight into the image's brightness and contrast. If pixel values are heavily skewed, normalization or other preprocessing might be required to improve model performance.

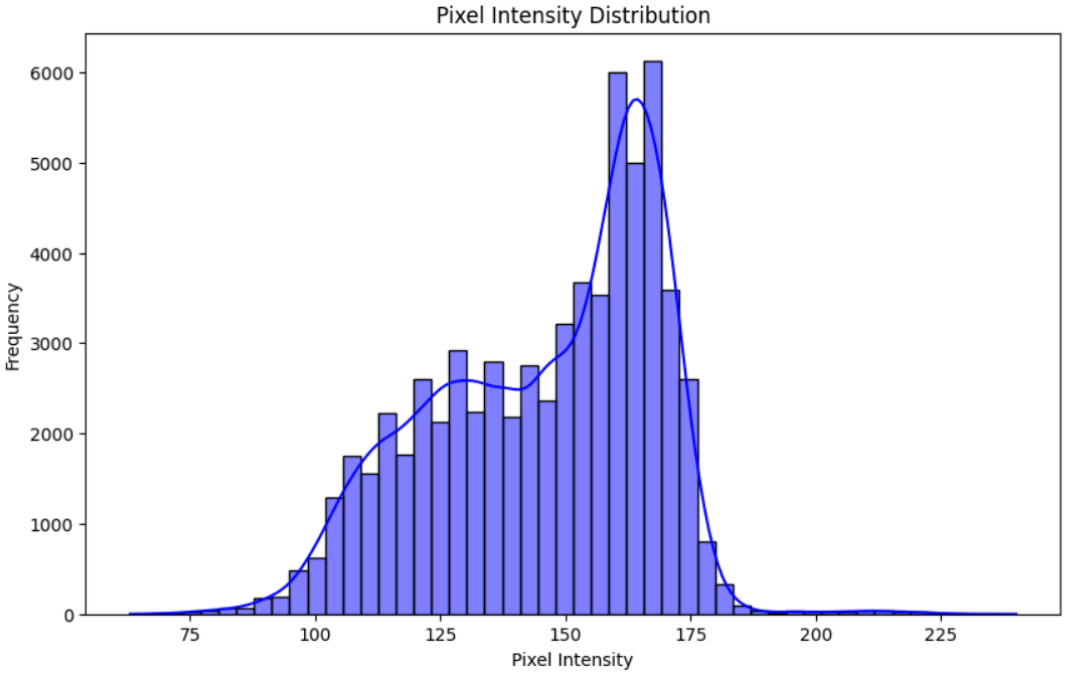


Figure 5.4 Pixel Intensity Distribution for a Sample Image

5.3.1 Evaluating Model Performance

- Accuracy

  The system achieves a high validation accuracy rate of 98%, indicating that it can correctly identify diseases in most cases.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

rf_classifier = RandomForestClassifier(n_estimators=200, random_state=42)
rf_classifier.fit(train_features, train_labels)

# Evaluate on validation set
val_predictions = rf_classifier.predict(val_features)
accuracy = accuracy_score(val_labels, val_predictions)

print(f"Validation Accuracy with Random Forest: {accuracy:.2f}")
```

```
Validation Accuracy with Random Forest: 0.98
```

Figure 5.5 Random Forest Detection Accuracy

The below shown graps are the graphs plot against CNN training accuracy and validation accuracy, CNN training loss and Validation loss, accuracy vs learning rate and loss vs learning rate, and the learning rate represented in each epoch.

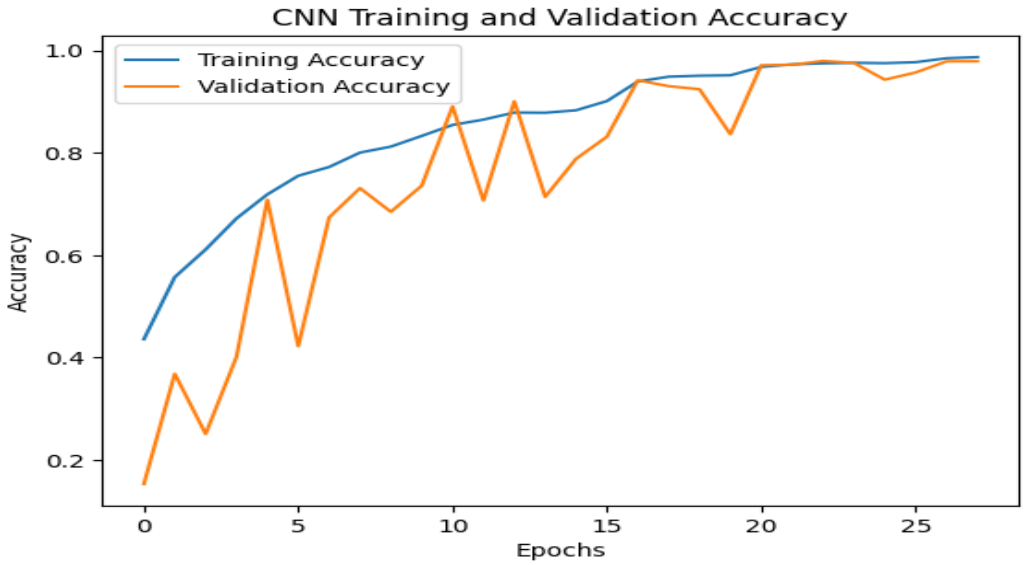


Figure 5.6 CNN Training and Validation Accuracy

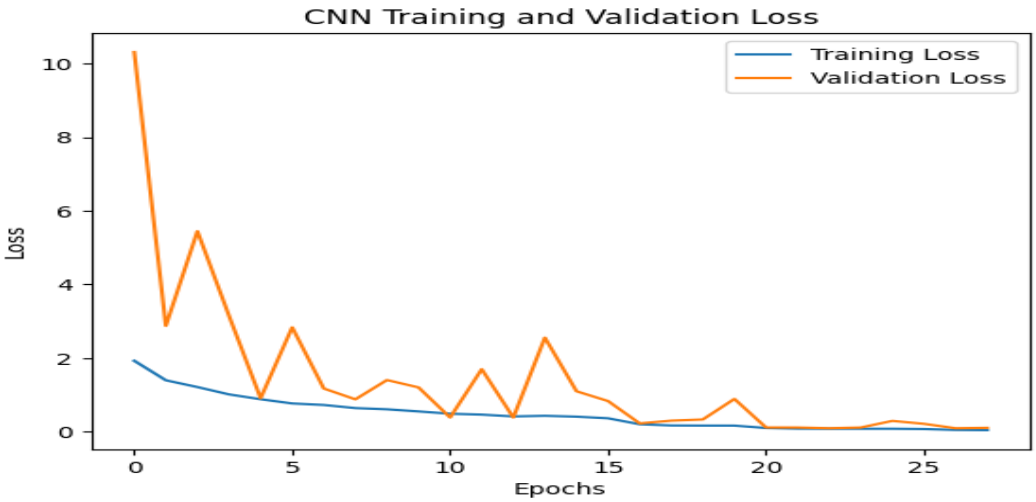


Figure 5.7 CNN Training Loss and validation Loss

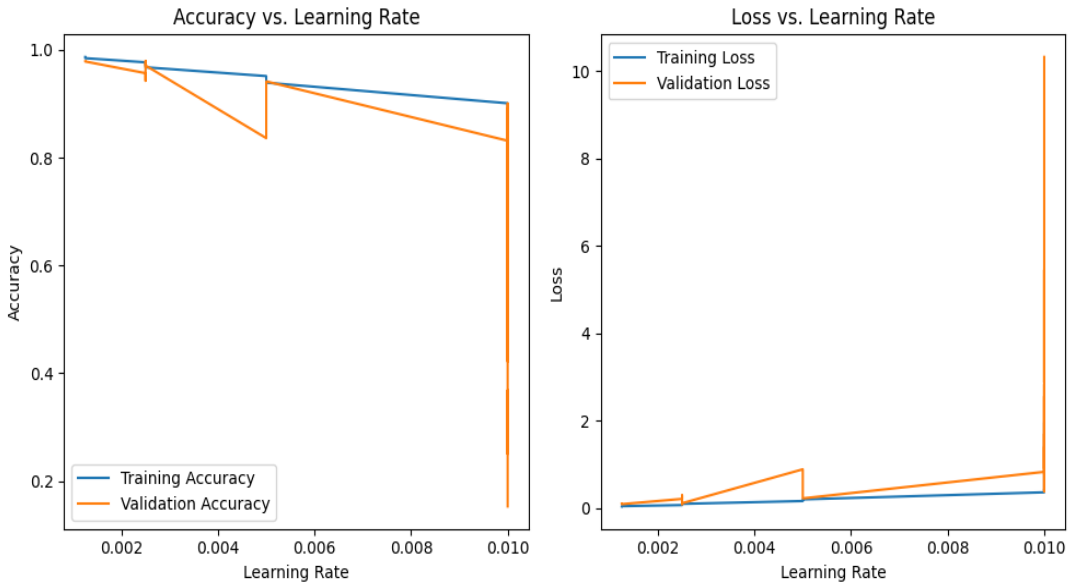


Figure 5.8 Accuracy Vs Learning Rate and Loss Vs learning Rate

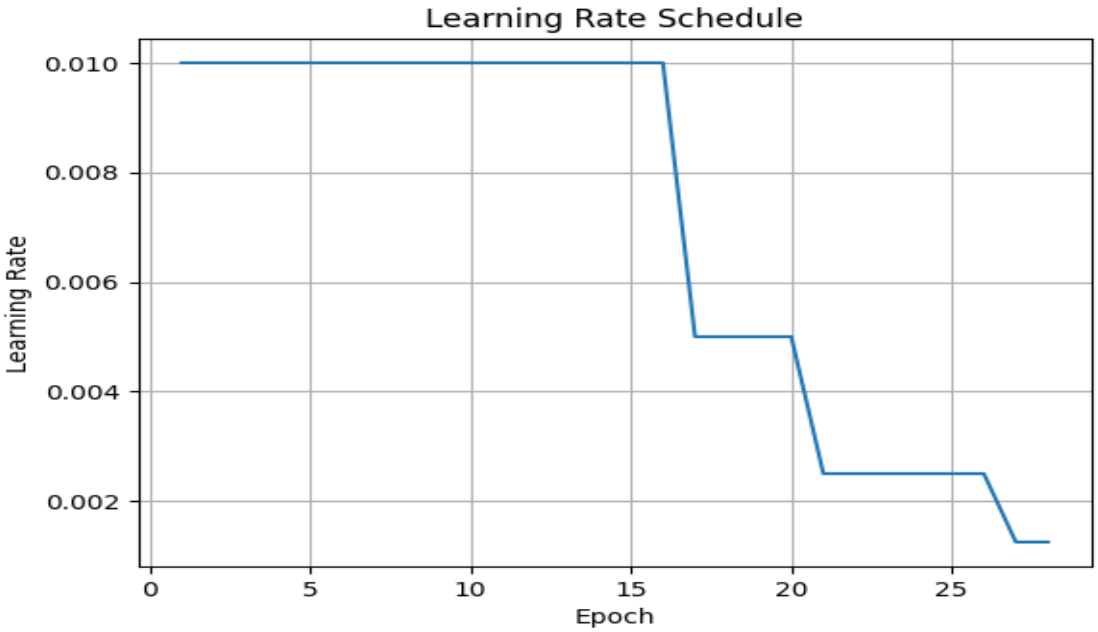


Figure 5.9 Learning Rate in Each Epoch

- Precision:

  Precision reflects how well the system avoids false alarms. For instance, if the model predicts a plant has a disease, it gets it right 97.78% of the time.

- Recall:

  This metric measures the system's ability to detect all actual disease cases, with a recall of 97.78%. It ensures that the model doesn't miss many cases.

- F1 Score:

  The F1 score, a balance between precision and recall, stands at 97.78%, showing the system's consistency in identifying diseases while minimizing errors.

```
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score


val_predictions_labels = val_predictions


# Calculate metrics:
accuracy = accuracy_score(val_labels, val_predictions_labels)
precision = precision_score(val_labels, val_predictions_labels, average='micro')
recall = recall_score(val_labels, val_predictions_labels, average='micro')
f1 = f1_score(val_labels, val_predictions_labels, average='micro')

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

```
Accuracy: 0.9778
Precision: 0.9778
Recall: 0.9778
F1 Score: 0.9778
```

Figure 5.10 Overall Accuracy, Precision, Recall and F1 Score of The System

The overall accuracy ,precision recall and F1 Score of the four plants i.e apple ,grape ,cornand tomato considered during model building is shown below:

```
Apple Metrics:
  Accuracy: 0.9865, Precision: 1.0000, Recall: 0.9865, F1-score: 0.9932
Corn Metrics:
  Accuracy: 0.9987, Precision: 1.0000, Recall: 0.9987, F1-score: 0.9994
Grape Metrics:
  Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000, F1-score: 1.0000
Tomato Metrics:
  Accuracy: 0.9975, Precision: 1.0000, Recall: 0.9975, F1-score: 0.9988
```

Figure 5.11 Precision Accuracy Recall and F1 Score of Each Category

- Confusion matrix:

  The system’s predictions were analyzed using a confusion matrix, which offered the following insights:

  - It excelled in detecting diseases with unique and visible symptoms.
  - However, diseases with subtle or overlapping features were more challenging to classify accurately.
  - While most predictions were spot-on, occasional errors arose, such as false positives (predicting a disease when none existed) or false negatives (failing to detect a disease that was present).

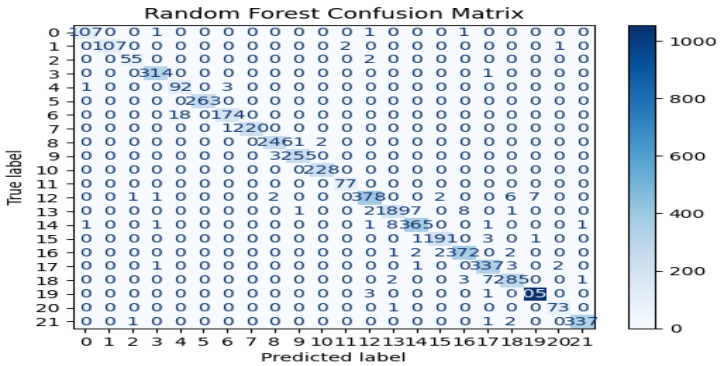


Figure 5.12 Confusion Matrix Generated by The Module

- ROC Curve

  The ROC curve visually shows how well the system balances catching diseased plants (sensitivity) while avoiding false positives. A higher Area Under the Curve (AUC) means the system makes confident and accurate decisions most of the time.Below shown figure is the ROC curve of our model.

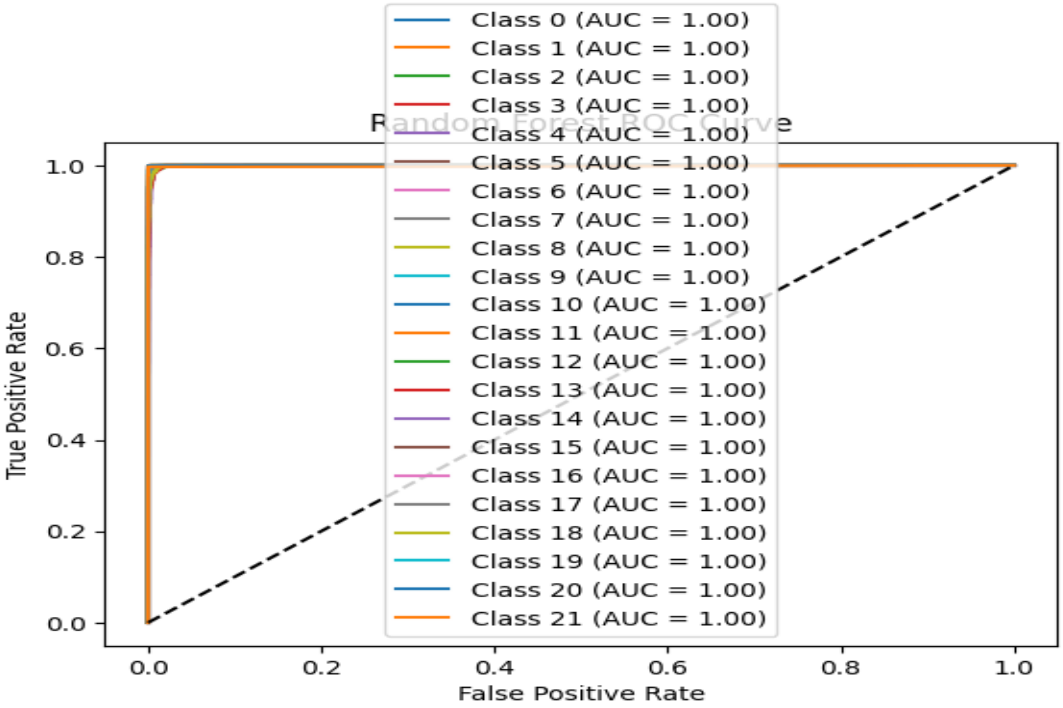


Figure 5.13 ROC Curve of The Model

Chapter 6: Conclusion and Future Recommendation

6.1 Conclusion

The Plant Disease Detection System is a powerful tool that uses cutting-edge machine learning techniques, like Convolutional Neural Networks (CNNs) and Random Forest algorithms, to provide a quick, accurate, and easy way to identify plant diseases. With its simple interface, users can upload photos of their plants and instantly get predictions about potential diseases, along with helpful information on symptoms, causes, prevention, and treatment. By using CNNs, the system carefully analyzes the images to pick up important details about the plant's condition, and the Random Forest classifier brings even more accuracy by combining the insights from multiple decision trees. The model's performance is solid, as shown by key metrics like accuracy, precision, recall, and the F1 score, proving it works well in real-world situations. Additionally, tools like the confusion matrix and ROC curve offer valuable insights into how the model makes its predictions, helping improve the system over time. This makes the system an invaluable resource for farmers, gardeners, and agricultural professionals who need reliable and timely information to keep their plants healthy and manage their crops effectively. In conclusion, this system marks a major leap forward in plant disease detection, making it easier for anyone to care for their plants while supporting sustainable farming practices. With its solid design and room for continuous improvement, it offers a dependable solution to the modern challenges faced in agriculture.

6.2 Future Recommendation

The future recommendation of PDDS include:

- User History: Users will be able to securely store and review their plant photos and disease predictions, making it easy to track plant health over time.
- More Resources: The platform will offer more helpful guides on plant care and pest management to help users better understand plant health.
- Expert Support: Users will be able to contact agricultural experts for personalized advice, ensuring they get the right help when needed.

References

[1] "Agile Software Development – Software Engineering." Accessed: Dec. 06, 2024. [Online]. Available: https://www.geeksforgeeks.org/software-engineering-agile-software-development/

[2] Kaatya Mishra, "Impediments in the Agriculture Sector Of Nepal." Accessed: Aug. 23, 2024. [Online]. Available: https://nepaleconomicforum.org/impediments-in-the-agriculture-sector-of-nepal/#:~:text=In%202022%2C%20the%20agriculture%20sector's,2022%20it%20decreased%20to%2066%25.

[3] P. Kulkarni, A. Karwande, T. Kolhe, S. Kamble, A. Joshi, and M. Wyawahare, "Plant Disease Detection Using Image Processing and Machine Learning."

[4] R. Sai Sharvesh, K. Suresh Kumar, and C. J. Raman, "An Accurate Plant Disease Detection Technique Using Machine Learning," *EAI Endorsed Transactions on Internet of Things*, vol. 10, Nov. 2024, doi: 10.4108/eetiot.4963.

[5] C. I. Sofuoglu and D. Birant, "Potato Plant Leaf Disease Detection Using Deep Learning Method," *Tarim Bilimleri Dergisi*, vol. 30, no. 1, pp. 153–165, Sep. 2024, doi: 10.15832/ankutbd.1276722.

[6] A. Mishra, P. Chaurasia, V. Arya, and F. J. G. Peñalvo, "Plant Disease Detection using Image Processing," in *Lecture Notes in Networks and Systems*, Springer Science and Business Media Deutschland GmbH, 2023, pp. 227–235. doi: 10.1007/978-3-031-22018-0_21.

[7] " Introduction to Convolution Neural Network." Accessed: Sep. 26, 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[8] " Random Forest Algorithm in Machine Learning." Accessed: Sep. 24, 2024. [Online]. Available: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

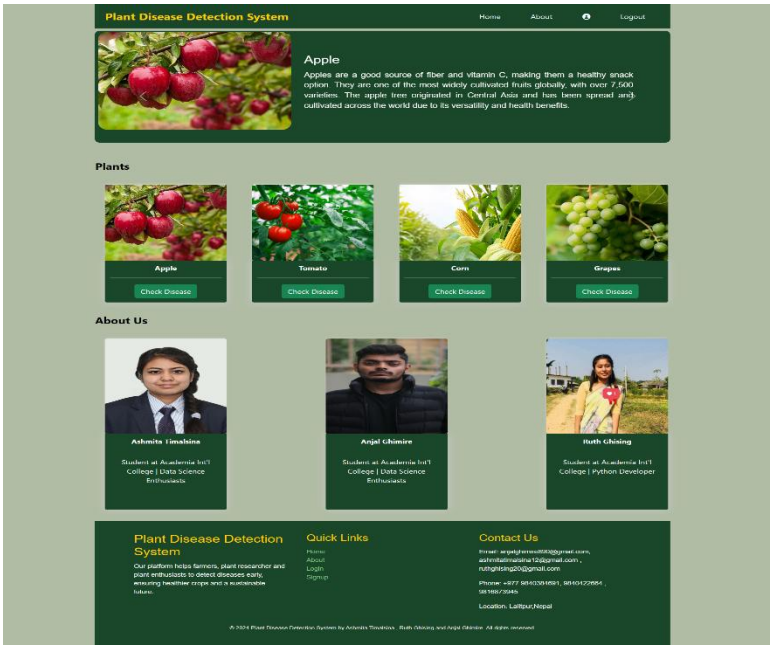
[9] "What Is ROC Curve in Machine Learning?" Accessed: Sep. 22, 2024. [Online]. Available: https://www.coursera.org/articles/what-is-roc-curve

[10] "Multiclass Receiver Operating Characteristic (ROC)." Accessed: Sep. 22, 2024. [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#multiclass-receiver-operating-characteristic-roc

Appendices

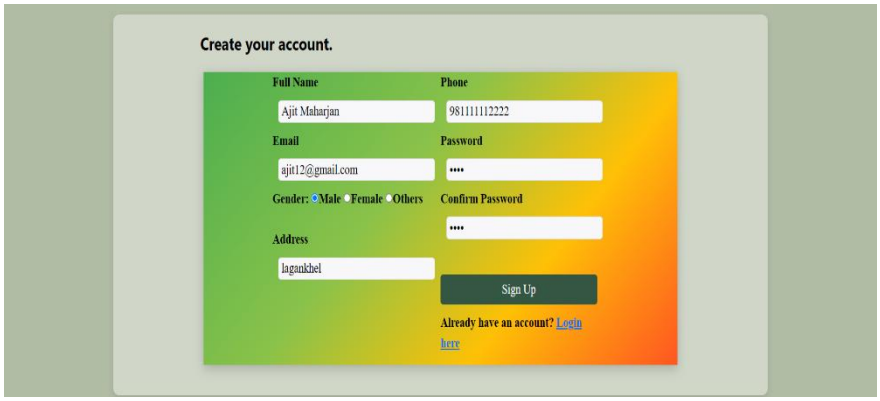
Screenshoot of websites

1. Home page

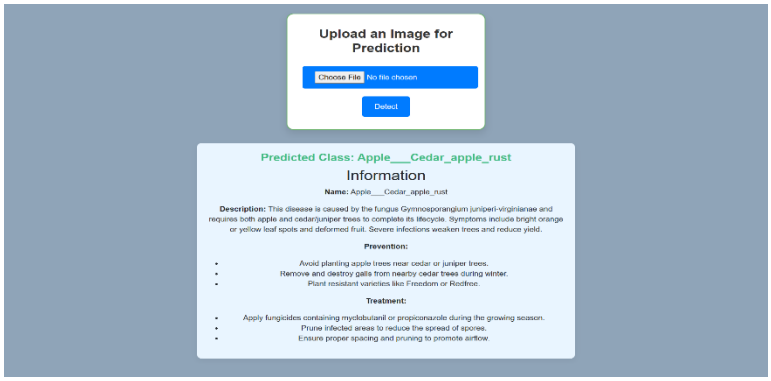


2. Login page

3. Sign Up page



4. Disease Detection page



5. Profile page