Plant Leaf Disease Detection Using Deep Learning

A Mini Project Report Submitted

In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in

Computer Science and Engineering

by

U.Hari Chandana	**21N31A05P7**
V.Sruthi	**22N35A0526**
W.Rahul	**21N31A05R4**

Under the esteemed guidance of

Ms.P.L.Shailaja

Assistant Professor

Department of Computer Science and Engineering

Malla Reddy College of Engineering & Technology
(Autonomous Institution- UGC, Govt. of India(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

2024-2025

Malla Reddy College of Engineering & Technology
(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the application entitled "**plant leaf disease detection using deep learning**", submitted by Student **U.Hari Chandana, V.Sruthi, W.Rahul and 21N31A05P7, 22N35A0526, 21N31A05R4** of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2024-2025. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree.

Internal Guide	**Head of the Department**
MS.P.L. Shailaja	**Dr. S. Shanthi.,**
Assistant Professor	**Professor**

External Examiner

DECLARATION

We hereby declare that the application titled "**PLANT LEAF DISEASE DETECTION USING DEEP LEARNING**" submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of a degree or diploma.

U. Hari Chandana	**21N31A05P7**
V. Sruthi	**22N35A0526**
W. Rahul	**21N31A05R4**

ACKNOWLEDGEMENT

We feel honored to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) for allowing us to do this Project as part of our B. Tech Program. We are ever grateful to our Director **Dr. V. S. K Reddy** and Principal **Dr. S. Srinivasa Rao** who enabled us to have experience in engineering and gain profound technical knowledge.

We express our heartiest thanks to our HOD, **Dr. S. Shanthi** for encouraging us in every aspect of our course and helping us realize our full potential.

We would like to thank our Project Guide **P.L.Shailah** and Project Coordinator **Mr. P. Dastagiri Reddy** for his regular guidance, suggestions, constant encouragement, continuous monitoring, and unflinching cooperation throughout project work.

We would like to thank our Class Incharge **Mr. P.Dastagiri Reddy** who despite being busy with her academic duties took time to guide and keep us on the correct path.

We would also like to thank all the faculty members and supporting staff of the Department of CSE and all other departments who have helped directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project which gave us the strength to do our project.
with regards and gratitude

U. Hari Chandana	**21N31A05P7**
V. Sruthi	**22N35A0526**
W. Rahul	**21N31A05R4**

ABSTRACT

This project showcases a deep learning-based system for detecting plant leaf diseases, presented through a rendered HTML web page interface. The system employs convolutional neural networks (CNNs) to classify plant leaves as healthy or diseased based on visible symptoms, offering a reliable diagnostic tool for plant health. The model was trained on a robust dataset containing labeled images of various plant diseases, enabling it to accurately recognize common ailments. By harnessing CNNs for image analysis, the model achieves high accuracy in classifying plant diseases.

The HTML-rendered interface, developed with Flask, provides users with a straightforward platform to interact with the system. Users can upload an image of a plant leaf, which the model then processes to determine the presence of any disease. Once analyzed, the page displays the result along with a recommended remedy if a disease is detected. This interface offers a simple and accessible means for users to gain insights into their plants' health without requiring additional software.

This project underscores the potential of deep learning in agricultural diagnostics and aims to assist farmers, gardeners, and researchers in managing plant health. Future improvements could include adding more disease classes, optimizing the model for faster performance, or expanding the interface for use on mobile devices. By combining deep learning with an easy-to-use HTML interface, this project demonstrates the applicability of AI in practical agricultural solutions.

Keywords: Plant disease detection, HTML web page, deep learning, convolutional neural networks, CNN, Flask-rendered interface, plant health, agricultural diagnostics

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF OUTPUTS

1. INTRODUCTION

The **Plant leaf Disease Detection using Deep learning** project is an innovative solution aimed at automating the identification and classification of plant diseases using **Convolutional Neural Networks (CNNs)**. Traditional disease detection methods rely heavily on manual inspection, which is labor-intensive and prone to errors, particularly in large-scale farming. To address this challenge, the project utilizes a deep learning model trained on a large dataset of plant leaf images to detect multiple plant diseases with high accuracy. By incorporating CNNs, the system can effectively recognize patterns and anomalies in leaf structures that indicate the presence of diseases. The model processes images uploaded by users through an intuitive web interface built with **flask**, making it accessible to farmers and agricultural professionals. The system provides real-time diagnosis and suggests actionable disease management strategies. This automated approach significantly reduces the time and effort required for disease identification, thereby enabling farmers to act promptly to prevent widespread crop damage. The project has broad applications in the agricultural industry, as it can be adapted for use with various types of crops and can be integrated into farming operations, particularly in regions where expert human intervention may not be readily available. Future developments of this system include expanding its capabilities to recognize a wider range of diseases and adapting it for mobile platforms, further improving accessibility for farmers in remote locations. The project represents a significant advancement in the use of AI and machine learning in agriculture, contributing to sustainable farming practices by helping to ensure crop health and productivity.

1.1 PURPOSE, AIM AND OBJECTIVES:

The Plant Leaf Disease Detection project is designed to develop an advanced automated system utilizing Convolutional Neural Networks (CNNs) to accurately identify and classify diseases in plants from images of leaves. This system addresses the limitations of traditional agricultural practices, which rely heavily on manual inspections conducted by farmers or experts. Conventional methods are labor-intensive, time-consuming, and prone to errors, particularly in the early stages of disease manifestation. By employing CNN-based machine learning algorithms, the proposed system provides a more reliable, scalable, and efficient solution for detecting plant diseases, ensuring faster and more accurate diagnoses that can help prevent crop damage.

The aim of this project is to equip farmers and agricultural professionals with a user-friendly and accessible tool that automates disease detection in real-time. By allowing users to upload images of plant leaves through a web interface built using Streamlit, the system processes the images, compares them against a trained model, and provides immediate feedback on the presence and type of disease. This eliminates the need for expert knowledge, offering a solution that integrates seamlessly into day-to-day farm operations. Early detection of plant diseases is critical for managing crop health and preventing widespread damage, and this system empowers farmers to take proactive measures to safeguard their crops, leading to increased yields and reduced economic losses.

Additionally, the project promotes sustainable farming by reducing reliance on chemical treatments and minimizing crop losses through timely intervention. By providing accurate and immediate detection results, farmers can make informed decisions on treatment strategies, optimizing resource use and minimizing environmental impact. The system is designed to be adaptable and scalable, handling multiple crop types and disease categories, making it suitable for a wide range of agricultural applications. Overall, the project aims to revolutionize agricultural disease management by offering an advanced, practical, and sustainable tool that supports the broader goals of modern farming.

1.2 EXISTING AND PROPOSED SYSTEM:

Existing System:

In conventional agricultural practices, disease detection relies heavily on manual inspections carried out by farmers or agricultural experts. Typically, this involves visually inspecting crops for physical signs of disease, such as spots, lesions, or discoloration on plant leaves. However, this method poses several challenges. It is time-consuming, particularly when monitoring large fields or a high volume of plants. Additionally, manual detection tends to be less effective during the early stages of disease manifestation when symptoms may not be fully visible or obvious. As a result, significant damage can occur before the disease is accurately identified. Human error is another significant concern, especially in large-scale farming, where the volume of crops can overwhelm even the most experienced specialists. Misdiagnosis or delayed identification can lead to improper treatment, resulting in decreased crop yields and increased financial losses.

Proposed System:

The proposed system is an automated solution that uses Convolutional Neural Networks (CNNs) to detect plant leaf diseases, addressing the limitations of traditional manual methods. Instead of relying on labor-intensive and error-prone visual inspections, this system allows farmers to upload images of leaves through a web-based interface built using Streamlit. The images are processed in real-time, and the CNN model identifies whether the plant is diseased and, if so, classifies the specific disease. This automated approach eliminates the need for expert knowledge, making disease detection faster, more accurate, and easier to integrate into everyday farming operations. By providing early detection, the system helps farmers take timely action to prevent the spread of diseases and reduce crop loss. Additionally, the system offers recommendations for treatment, supporting more sustainable farming by reducing unnecessary chemical use and improving overall crop management.

1.3 SCOPE OF PROJECT:

The scope of the Plant Disease Detection system extends far beyond individual farm operations, positioning it as a versatile and adaptable tool for a wide range of agricultural applications. Designed to accommodate various crop types, the model is scalable and can be applied across different regions, enabling its use in diverse agricultural settings, from small-scale farms to large industrial agricultural enterprises. This adaptability ensures that the system can be tailored to meet the unique requirements of different crops, climates, and disease patterns in various parts of the world. Farmers, agricultural cooperatives, and companies involved in large-scale farming can benefit from this system, using it as a preventive measure to identify and manage diseases before they cause significant crop damage. The system's ability to process and analyze images of plant leaves in real-time allows for timely interventions, ensuring that diseases are detected early, thereby minimizing crop losses and improving overall agricultural productivity.

Furthermore, the scalability of the system offers vast potential for further adaptation and growth. As the system evolves, it can be expanded to cover an even broader range of plant species and diseases, making it a comprehensive solution for disease detection across multiple crops. Future updates will include continuous training of the CNN model, improving its accuracy and effectiveness over time as new disease data becomes available. This will enhance the system's ability to detect even rare or emerging plant diseases, ensuring that farmers are always equipped with the latest tools to combat threats to their crops. Another key aspect of the future scope involves integrating the system into mobile applications, making it more accessible to farmers in remote and rural areas. By enabling disease detection on mobile devices, the system will become more user-friendly and widely available, helping farmers in regions with limited access to agricultural experts and resources.

Overall, the Plant Disease Detection system is poised to play a crucial role in modernizing agriculture by offering an advanced, scalable, and accessible tool that supports sustainable farming practices. Through continuous improvements and expansion, it will become an indispensable resource for farmers worldwide, contributing to better disease management, increased crop yields, and a more secure global food supply.

2.LITERATURE SURVEY

Plant disease detection using image processing and machine learning has seen significant advancements in recent years. Traditional approaches relied on manual inspection of plants by agricultural experts, but these methods are limited in scalability and prone to human error. Early research focused on techniques like color-based segmentation and edge detection for identifying diseased areas on leaves. However, these methods struggled with complex variations in image quality, environmental conditions, and the variety of crops, which often led to low accuracy.

With the advent of machine learning, particularly Convolutional Neural Networks (CNNs), significant improvements have been made in plant disease detection. CNNs are designed to handle image classification tasks efficiently by automatically extracting relevant features from input images. Studies show that models like ResNet-50 and VGG-16 have been successfully applied to classify plant diseases with high accuracy. These models have been trained on large datasets of plant images, such as the PlantVillage dataset, which has contributed significantly to the field by providing a comprehensive collection of annotated images for training models.

Despite the success of CNNs in plant disease detection, integrating these models into accessible, real-time applications for farmers remains a challenge. Most existing research focuses on model performance, with limited emphasis on practical implementation in field conditions. Solutions like web-based or mobile applications that allow users to upload images for real-time disease diagnosis are still relatively rare. The use of Streamlit as an interface for deploying machine learning models offers a practical solution by providing a lightweight, easy-to-use platform that can be accessed remotely.

Recent studies also highlight the importance of dataset diversity in improving model generalization. Many models achieve high accuracy in controlled environments but struggle when applied to real-world images with varying lighting, background, and noise conditions. Techniques such as data augmentation, transfer learning, and domain adaptation have been explored to enhance model robustness. By addressing these challenges, projects like the Plant leaf Disease Detection system aim to provide a reliable and user-friendly tool for farmers, offering practical solutions for disease prevention and sustainable agriculture.

3.SYSTEM ANALYSIS

3.1 Hardware and Software Requirements

Hardware Requirements:

• Processor	:	Intel i5/i7/i9
• RAM	:	8 GB
• Storage	:	100 GB

Software Requirements:

• Operating System	:	Windows
• Programming Language	:	Python 3.7
• Development Environment	:	Visual Studio Code

3.2 SOFTWARE REQUIREMENTS SPECIFICATION:

The software requirements for the **Plant Leaf Disease Detection** project involve creating a user-friendly web interface using **Flask** that allows users to upload leaf images for disease classification. The system uses a **Convolutional Neural Network (CNN)** model to process and classify diseases, displaying the results with actionable recommendations. The backend is built with **Python** and **TensorFlow**, ensuring quick and accurate predictions. Key non-functional requirements include performance, scalability, and security, ensuring the system is responsive, secure, and able to handle multiple user requests concurrently.

Functional Requirements:

Image Upload: The system must allow users to upload images of plant leaves. The input images can be in formats like JPEG or PNG, and the interface should handle file validation and feedback in case of incorrect formats.

Disease Detection: The core functionality is disease detection using a trained **Convolutional Neural Network (CNN)**. The system processes the uploaded image, runs it through the model, and identifies the disease by comparing the image with known patterns. The system should support multiple disease types and provide a confidence score for each prediction.

Result Display: After detection, the system must display a detailed result to the user. This includes the detected disease, a confidence score (e.g., 90% confidence), and suggestions for managing the disease, such as recommended treatments or agricultural practices. The results must be presented in a user-friendly manner.

Image Storage and Logging: The system may log user interactions, including uploaded images and diagnostic results, for further analysis and improvement of the model. This feature should include user consent to comply with data privacy regulations.

Non-Functional Requirements :

Performance: The system should deliver results within a few seconds of image upload, ensuring real-time feedback. The detection model should be optimized for speed without compromising accuracy. The system should efficiently handle high traffic with multiple concurrent users.

Scalability: The system must be designed to scale across a growing user base. This includes the ability to handle increased image uploads, more disease types, and expanding geographic coverage. Cloud infrastructure such as AWS or Google Cloud may be used to ensure that the system scales effectively

Security: All data transmitted and stored must be secured. The system must ensure that uploaded images and any user information are handled according to privacy standards, such as GDPR. Encryption and access controls should be in place to prevent unauthorized access to the data.

The Plant leaf Disease Detection project aims to provide an efficient, user-friendly solution for identifying plant diseases using a Convolutional Neural Network (CNN) and a web interface powered by Streamlit. The Software Requirements Specification (SRS) outlines both the functional and non-functional requirements needed to achieve real-time disease detection with a focus on usability, scalability, and security. With these requirements, the system ensures accurate disease classification, ease of use for non-expert users, and the ability to scale as the agricultural demands grow.

4. SYSTEM DESIGN

DESCRIPTION:

The system for the **Plant Leaf Disease Detection** project is structured into several key modules to ensure efficiency and accuracy. The **image preprocessing** module is responsible for preparing uploaded leaf images by resizing them to a standardized dimension and normalizing pixel values, which helps ensure that all input data is consistent and can be processed effectively by the CNN. This step is crucial as it improves the performance of the model by eliminating discrepancies caused by varying image sizes and quality.

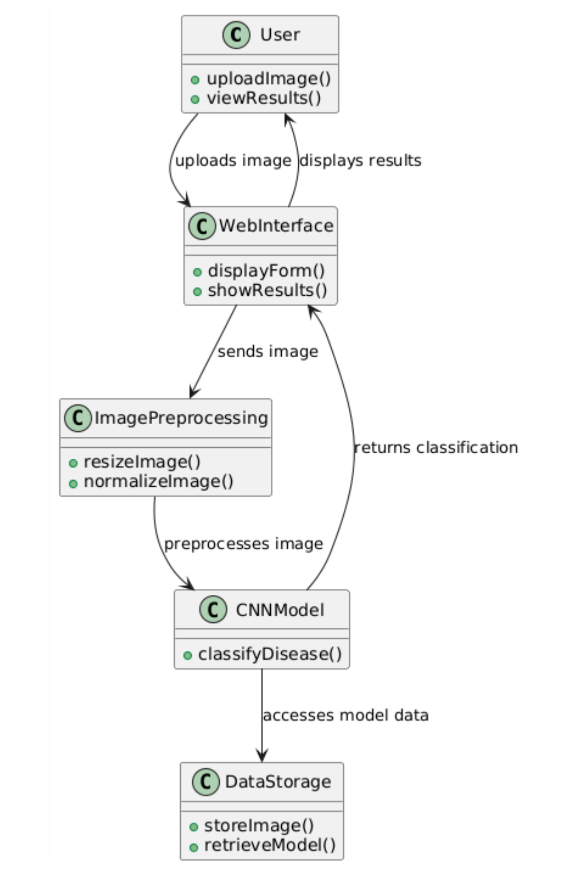
Once the image is preprocessed, it is passed into the **Convolutional Neural Network (CNN)**, which is trained to recognize and classify a wide range of plant diseases. The CNN extracts features from the images, such as texture, color, and shape, and uses this information to identify the presence of specific diseases. The model has been trained on a large dataset of plant images to ensure that it can accurately distinguish between healthy and diseased plants across various species.After the image has been processed by the CNN, the system generates an output that includes the predicted disease along with a confidence score, which indicates how certain the model is of its classification.

The results are then displayed through a **Flask**, which provides users with a seamless and intuitive experience. This interface allows users to easily upload images, view disease predictions, and receive recommendations for disease management. The modular design ensures that each component of the system functions independently while working together to deliver an efficient, real-time disease detection tool that is both scalable and user-friendly. This system can be applied to various crops and farming conditions, making it a valuable asset in modern agriculture.

4.1 UML Diagrams

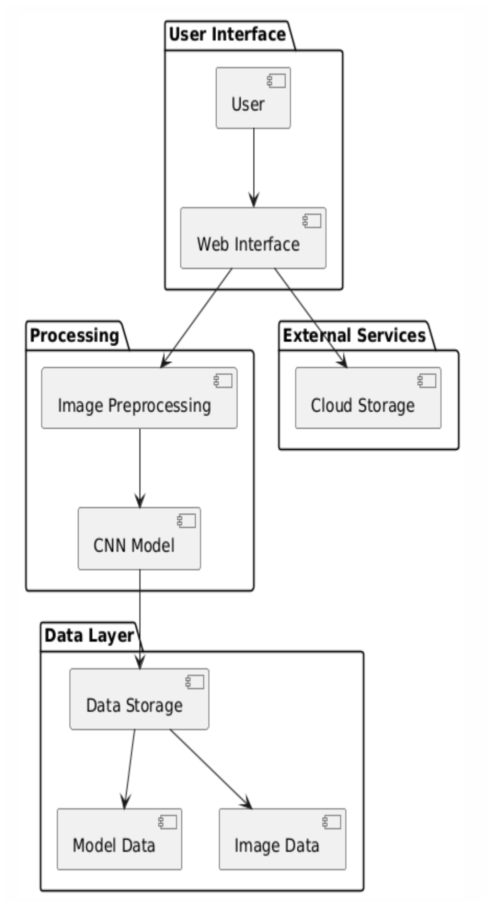
4.1.1 Class Diagram:

The class diagram depicts a plant disease classification system. The User interacts with the system by uploading an image via the uploadImage() method and viewing results with viewResults(). The web interface displays the form and sends the uploaded image for processing. The ImagePreprocessing class handles tasks such as resizing and normalizing the image. After preprocessing, the CNNModel classifies the image using its classifyDisease() method, with the DataStorage class providing access to stored images and model data through storeImage() and retrieveModel(). The system then returns the classification results to the user.



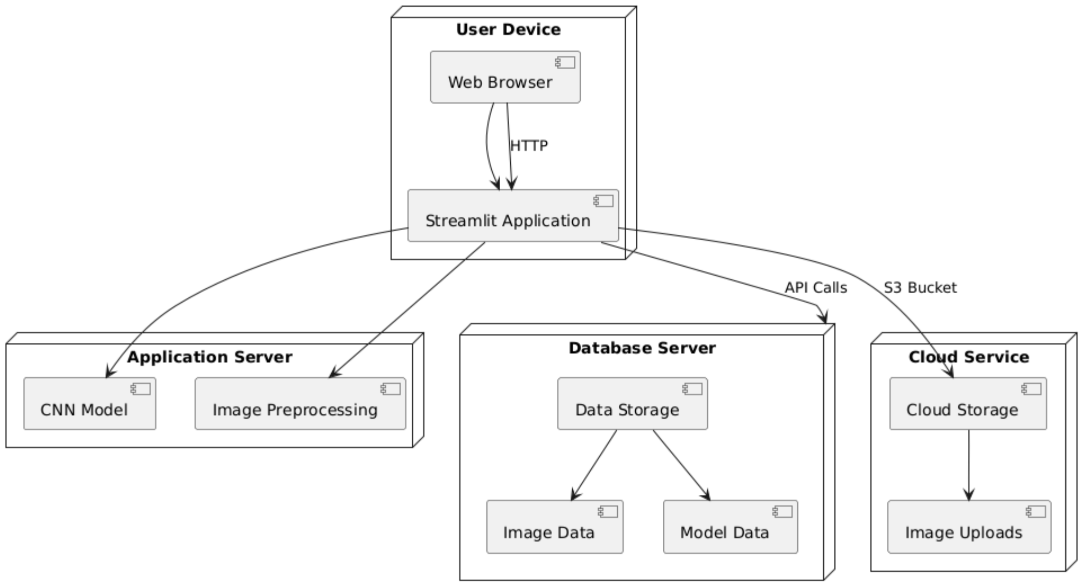
4.1.2 Component Diagram:

The component diagram outlines the architecture of a plant disease classification system. The User Interface involves the user interacting with the system via the Web Interface. The Processing layer includes Image Preprocessing and the CNN Model for handling image input and classification tasks. The External Services section provides Cloud Storage for additional storage and computational support. In the Data Layer, the Data Storage component manages both Model Data and Image Data, essential for the CNN model's functionality and result generation.



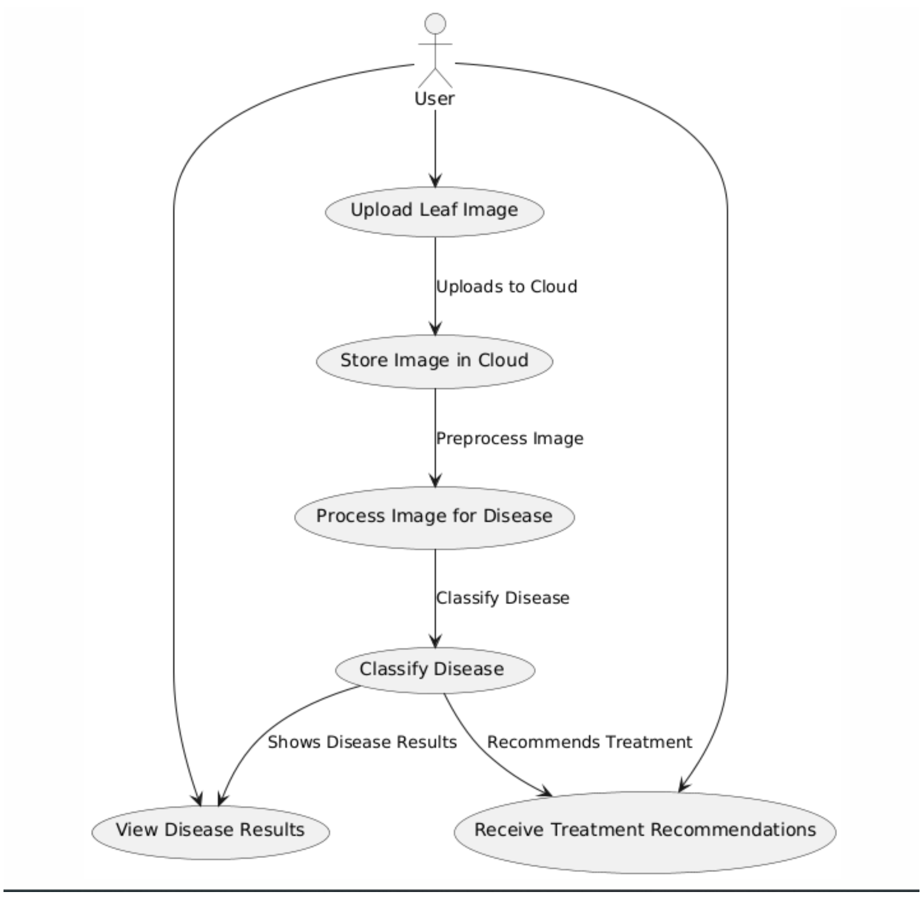
4.1.3 Deployment Diagram:

The deployment diagram illustrates the architecture of a plant disease classification system. The User Device interacts with the system via a Web Browser running a Streamlit Application over HTTP. The Application Server handles the CNN Model and Image Preprocessing tasks. The Database Server contains Data Storage, which manages both Image Data and Model Data. The Cloud Service component offers Cloud Storage and manages Image Uploads, with API calls facilitating communication between the different components and an S3 bucket used for cloud storage.



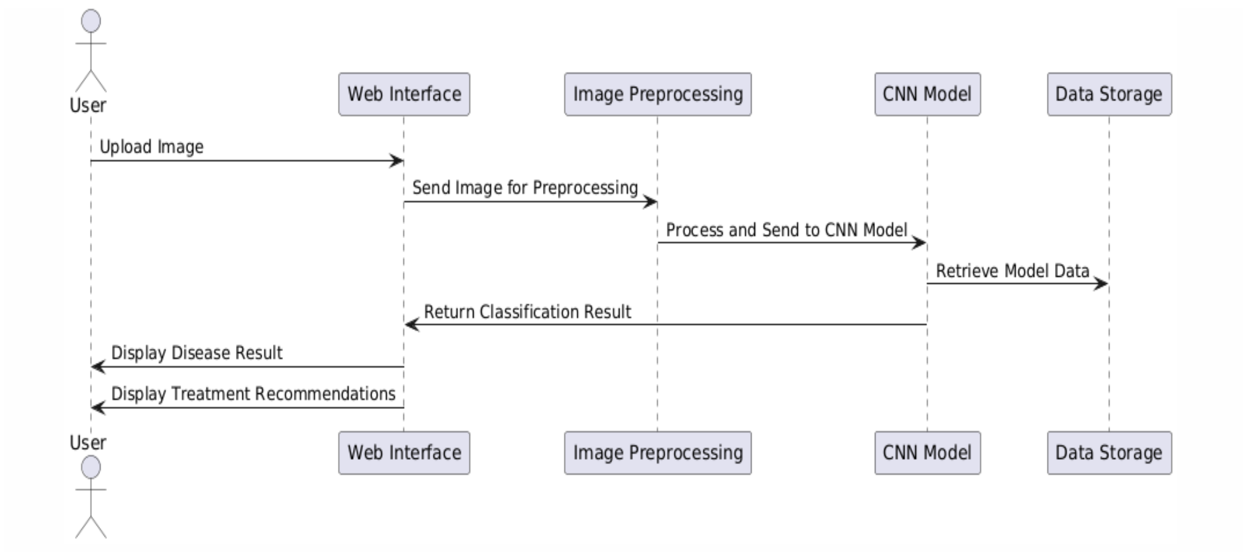
4.1.4 Use Case Diagram

The use case diagram outlines the interaction between the User and the system. The user uploads a Leaf Image, which is then uploaded to the cloud and stored. The system Preprocesses the Image for disease classification, using the model to Classify the Disease. Based on the classification, the system either Shows Disease Results or Recommends Treatment. The user can view the disease results or receive treatment recommendations as the final output of the process.



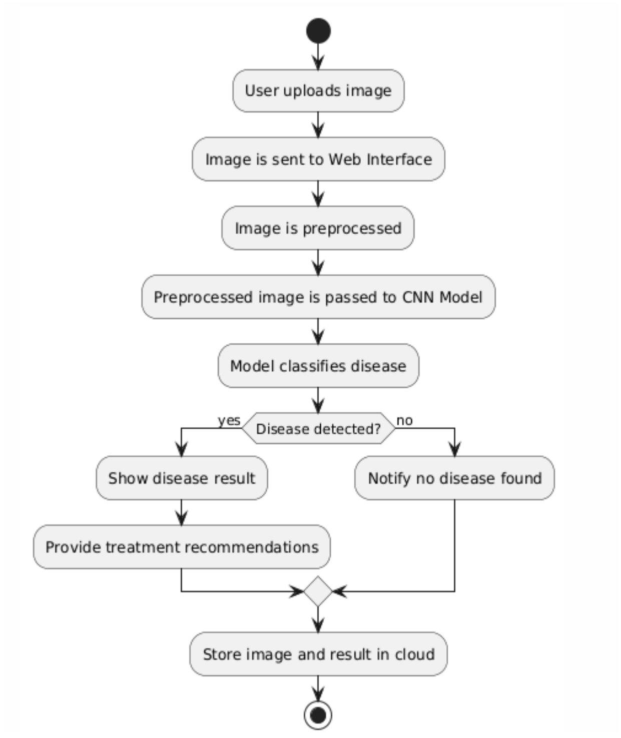
4.1.5 Sequence Diagram

The sequence diagram illustrates the interaction between the User and various system components. The user uploads an image through the Web Interface, which sends the image for preprocessing to the Image Preprocessing module. The preprocessed image is then processed by the CNN Model, which retrieves necessary data from Data Storage. The classification result is returned to the Web Interface, where the system displays the Disease Result or provides Treatment Recommendations to the user. This sequence outlines the step-by-step flow of information through the system.



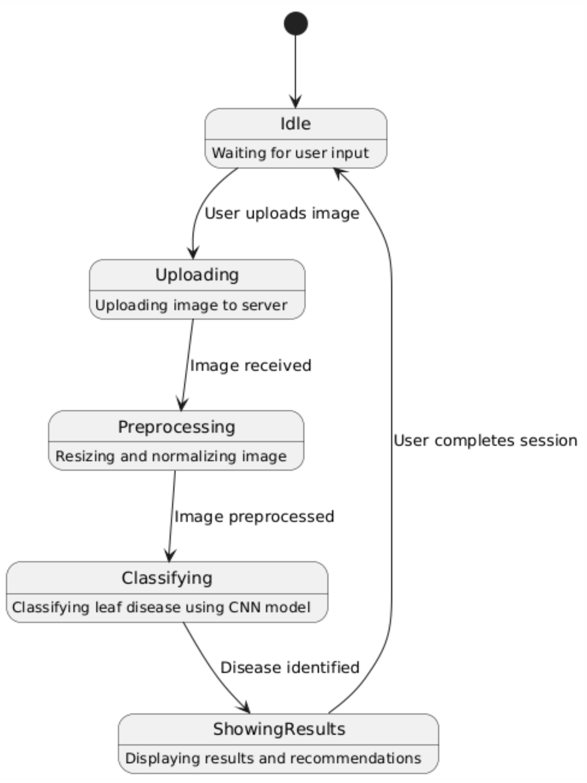
4.1.6 Activity Diagram

The activity diagram represents the workflow of the plant disease classification process. The user uploads an image, which is sent to the Web Interface. The image is then preprocessed and passed to the CNN Model for disease classification. If a disease is detected, the system shows the disease result and provides treatment recommendations. If no disease is found, the user is notified accordingly. Finally, the image and classification result are stored in the cloud, completing the process.


User uploads image
Image is sent to Web Interface
Image is preprocessed
Preprocessed image is passed to CNN Model
Model classifies disease
yes
Disease detected?
no
Show disease result
Notify no disease found
Provide treatment recommendations
Store image and result in cloud

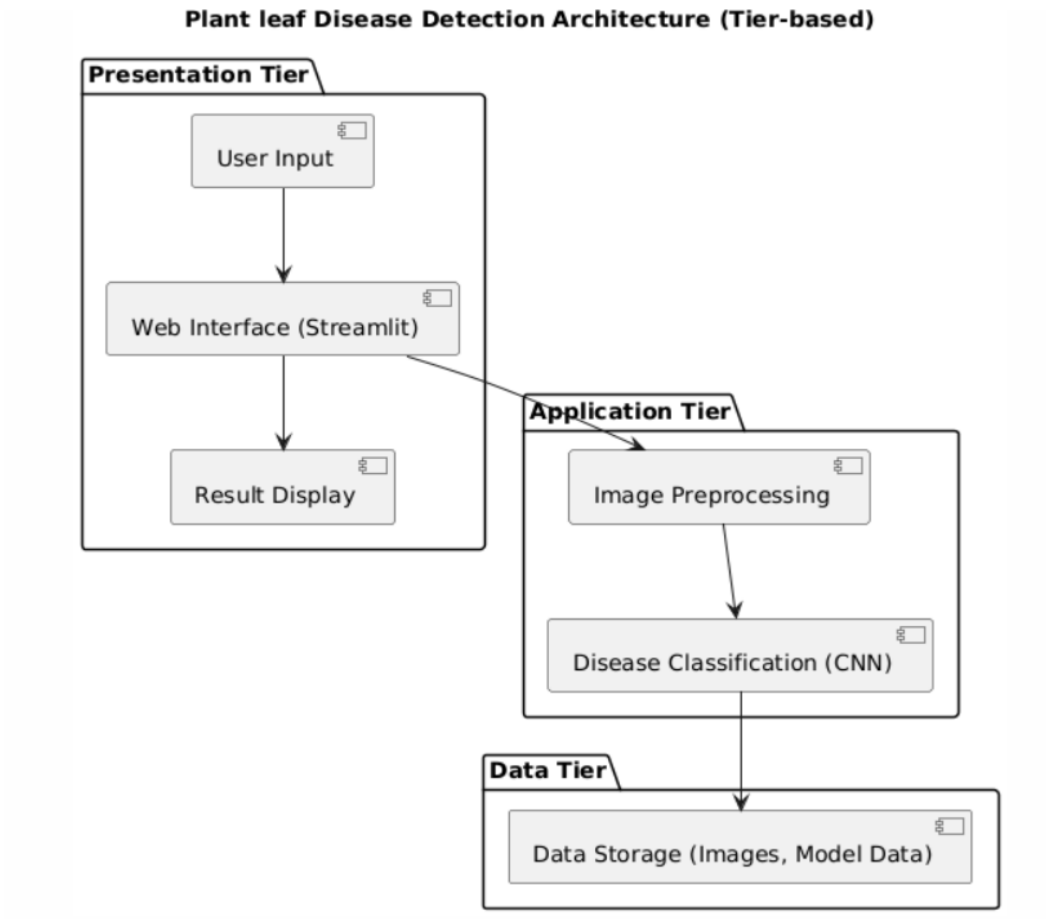
4.1.7 State Chart Diagram

The state chart diagram shows the various states the system goes through during the plant disease classification process. The system starts in an Idle state, waiting for user input. Once the User uploads an image, the system enters the Uploading state, where the image is sent to the server. After the image is received, it transitions to the Preprocessing state, where the image is resized and normalized. The preprocessed image is then classified in the Classifying state using the CNN model. If a disease is identified, the system moves to the ShowingResults state, where the classification results and treatment recommendations are displayed. The session ends once the user completes viewing the results**.**



4.2 Architecture

The architecture presented is a tier-based system for plant leaf disease detection. It consists of three main layers: the Presentation Tier, Application Tier, and Data Tier. In the Presentation Tier, user input is provided through a web interface, built using Streamlit, and results are displayed after processing. The Application Tier is responsible for processing the image, which involves image preprocessing followed by disease classification using a Convolutional Neural Network (CNN). Finally, the Data Tier handles storage, where images and model data are saved for future reference or further analysis. This layered approach ensures separation of concerns and scalability.



Plant leaf Disease Detection Architecture (Tier-based)
Presentation Tier
User Input
Web Interface (Streamlit)
Result Display
Application Tier
Image Preprocessing
Disease Classification (CNN)
Data Tier
Data Storage (Images, Model Data)

5. METHODOLOGY

5.1MODULE DESCRIPTION

This module focuses on the application of deep learning techniques to identify and diagnose plant diseases from images of leaves. It combines computer vision and machine learning to enhance agricultural productivity and ensure healthy crop yields by providing timely disease detection.

Learning Objectives:

1. Understand the fundamentals of deep learning and its applications in agriculture.
2. Explore various deep learning architectures suitable for image classification tasks.
3. Gain practical experience in preprocessing image data for training models.
4. Implement a convolutional neural network (CNN) to detect plant diseases.
5. Evaluate model performance using appropriate metrics and improve accuracy through techniques such as data augmentation.

1. Introduction to Plant Diseases:
    - Overview of common plant diseases and their impact on agriculture.
    - Importance of early detection and management strategies.
2. Deep Learning Fundamentals:
    - Overview of machine learning vs. deep learning.
    - Introduction to neural networks and CNNs.
3. Data Collection and Preprocessing:
    - Sources of plant disease images (public datasets, personal collection).
    - Image preprocessing techniques (resizing, normalization, augmentation).
4. Model Development:
    - Designing a CNN architecture.
    - Implementing transfer learning with pre-trained models (e.g., VGG16, ResNet).
    - Training the model on labeled datasets.
5. Evaluation and Optimization:
    - Techniques for model evaluation (accuracy, precision, recall, F1-score).
    - Hyperparameter tuning and model optimization strategies.
6. Deployment and Applications:
    - Deploying the model in a user-friendly application (web/mobile).

    - Future directions: integration with IoT devices for real-time monitoring.

7. Practical Component:

- Hands-on project where students will:
    - Collect or utilize an existing dataset of plant leaves.
    - Build and train a deep learning model for disease detection.
    - Present their findings and demonstrate their model’s performance.

8. Recommended Tools and Libraries:

- Python, TensorFlow/Keras o OpenCV for image processing, visual studio code for development.

6. IMPLEMENTATION

6.1 SAMPLE CODE

App.py:

```
import os
import sys
import io
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras._tf_keras.keras.preprocessing.image import load_img, img_to_array
from keras._tf_keras.keras.models import load_model
from flask import Flask, request, render_template, jsonify
from werkzeug.utils import secure_filename
app = Flask(__name__)
# Ensure 'uploads' directory exists
if not os.path.exists('uploads'):
os.makedirs('uploads')
# Load the pre-trained model
try:
model = load_model('model.h5')
print('Model loaded. Check http://127.0.0.1:5000/')
except Exception as e:
print(f"Error loading model: {e}")
model = None  # Ensure graceful handling if model fails to load

# Define labels for predictions
labels = {0: 'Healthy', 1: 'Powdery', 2: 'Rust'}
sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')
def getResult(image_path):
```

```
"""Preprocess the image and predict the result."""
try:
print(f"Processing image: {image_path}")

# Load and preprocess the image
img = load_img(image_path, target_size=(225, 225))
x = img_to_array(img)
x = x.astype('float32') / 255.0  # Normalize pixel values
x = np.expand_dims(x, axis=0)

# Predict the class probabilities
predictions = model.predict(x)
print(f"Predictions: {predictions}")

return predictions
except Exception as e:
print(f"Error in getResult: {e}")
return None

@app.route('/', methods=['GET'])
def index():
"""Render the index page."""
return render_template('index.html')

@app.route('/predict', methods=['GET','POST'])
def upload():
"""Handle file uploads and return the prediction."""
try:
# Check if a file is included in the request
if 'file' not in request.files:
```

```
print("No file part in the request.")
return jsonify({"error": "No file part in the request"}), 400

f = request.files['file']

# Check if the user selected a file
if f.filename == '':
print("No file selected for uploading.")
return jsonify({"error": "No file selected for uploading"}), 400

# Save the uploaded file
basepath = os.path.dirname(__file__)
file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
f.save(file_path)
print(f"File saved at: {file_path}")

# Get predictions
predictions = getResult(file_path)
if predictions is None:
print("Prediction failed.")
return jsonify({"error": "Prediction failed."}), 500
# Find the predicted label
predicted_label = labels[np.argmax(predictions)]
print(f"Predicted label: {predicted_label}")
return jsonify({"prediction": predicted_label})
except Exception as e:
print(f"Error in upload: {e}")
return jsonify({"error": str(e)}), 500
if __name__ == '__main__':
if model:
```

```
app.run(debug=True)
else:
print("Model failed to load. Exiting...")
```

index.html:

```
{% extends "import.html" %}
{% block content %}
<center>
<br>
<h2 style="color:White;">Plant Disease Prediction Using Deep Learning</h2><br>
<!-- Upload Form -->
<form id="upload-file" method="post" enctype="multipart/form-data">
<input type="file" name="file" class="btn btn-success" id="imageUpload" accept=".png,
.jpg, .jpeg">
</form>
<!-- Image Preview Section -->
<div class="image-section" style="display:none;">
<img id="imagePreview" class="img-responsive" src="#"
style="width:300px;height:300px;" /><br><br>
<div>
<button type="button" class="btn btn-info btn-lg" id="btn-predict">Predict!</button>
</div>
</div>
<!-- Loader -->
<div class="loader" style="display:none;">Loading...</div>
<!-- Prediction Result -->
<h3 id="result">
<span style="color:white;" ></span>
</h3>
</center><br><br>
<!-- JavaScript Section -->
<script>
```

```
$(document).ready(function () {
console.log("Document ready.");
// Hide elements initially
$('.image-section').hide();
$('.loader').hide();
$('#result').hide();
// Function to preview the uploaded image
function readURL(input) {
if (input.files && input.files[0]) {
const reader = new FileReader();
reader.onload = function (e) {
$('#imagePreview').attr('src', e.target.result);
$('.image-section').show();  // Show the image preview section
console.log("Image preview displayed.");
};
reader.readAsDataURL(input.files[0]);
}
}
// Trigger image preview when a file is selected
$('#imageUpload').change(function () {
console.log("File input changed.");
$('#result').text(''); // Clear previous results
$('#result').hide();   // Hide previous result section
readURL(this);         // Preview the selected image
});
// Handle the Predict button click
$('#btn-predict').click(function () {
console.log("Predict button clicked.");
const formData = new FormData($('#upload-file')[0]);  // Collect form data
// Show loader animation while making prediction
$(this).hide();
$('.loader').show();
// Make AJAX call to Flask backend
$.ajax({
type: 'POST',
url: '/predict',
data: formData,
contentType: false,
processData: false,
async: true,
success: function (response) {
console.log("Prediction received:", response);
// Hide the loader and display the result
$('.loader').hide();
$('#result').fadeIn(600);
$('#result span').text('Result: ' + response.prediction).css('color', 'white');
},
error: function (xhr, status, error) {
console.error("Prediction error:", error);
$('.loader').hide();
$('#result').fadeIn(600);
$('#result span').text('Error: ' + (xhr.responseJSON?.error || 'Prediction failed.'));
}
});
});
});
</script>
{% endblock %}
```

Import.html:

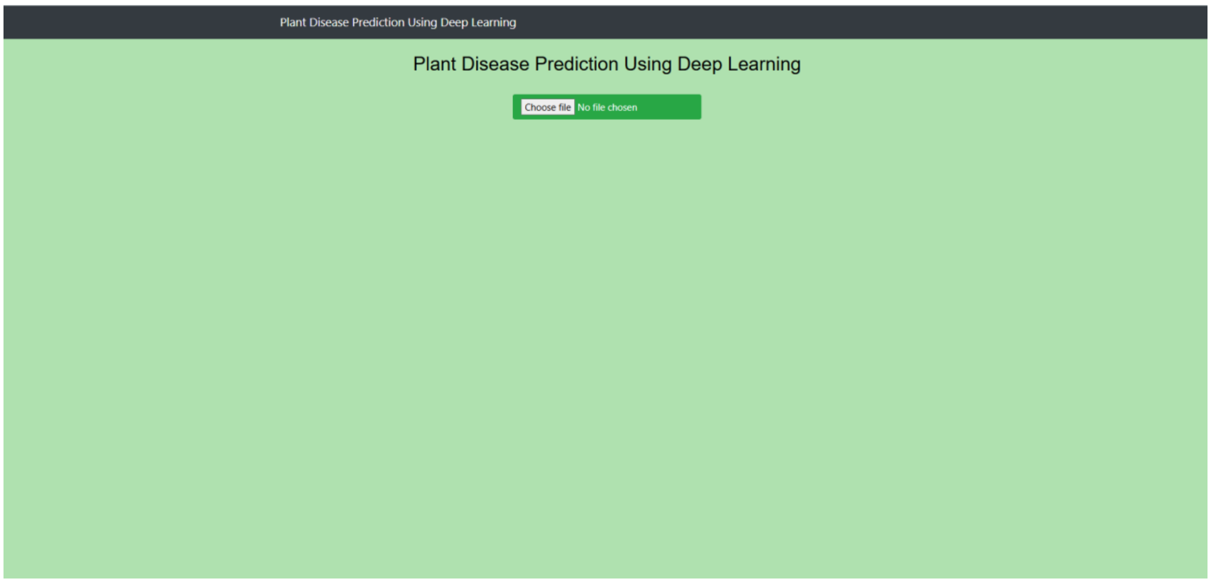
```
<html lang="en">
```

```
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Plant Disease Prediction Using Deep Learning</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}">
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
<link href="{{ url_for('static', filename='css/bootstrap.min.css') }}" rel="stylesheet">
<script src="{{ url_for('static', filename='js/jquery.min.js') }}"></script>
<script src="{{ url_for('static', filename='js/bootstrap.min.js') }}"></script>
<link href="{{ url_for('static', filename='css/test.css') }}" rel="stylesheet">
<script src="{{ url_for('static', filename='js/newjs.js') }}" type="text/javascript"></script>
<style>
@font-face {
font-family: 'CustomFont';
src: url('{{ url_for('static', filename='fonts/CustomFont.ttf') }}') format('truetype');
font-weight: normal;
font-style: normal;
}
#result span {
color: white;  / Set the result text color to white /
}
body {
background-image: url('{{ url_for('static', filename='image_back.jpeg') }}');
background-size: cover;
background-position: center;
background-repeat: no-repeat;
height: 100vh;
}
.content {
```

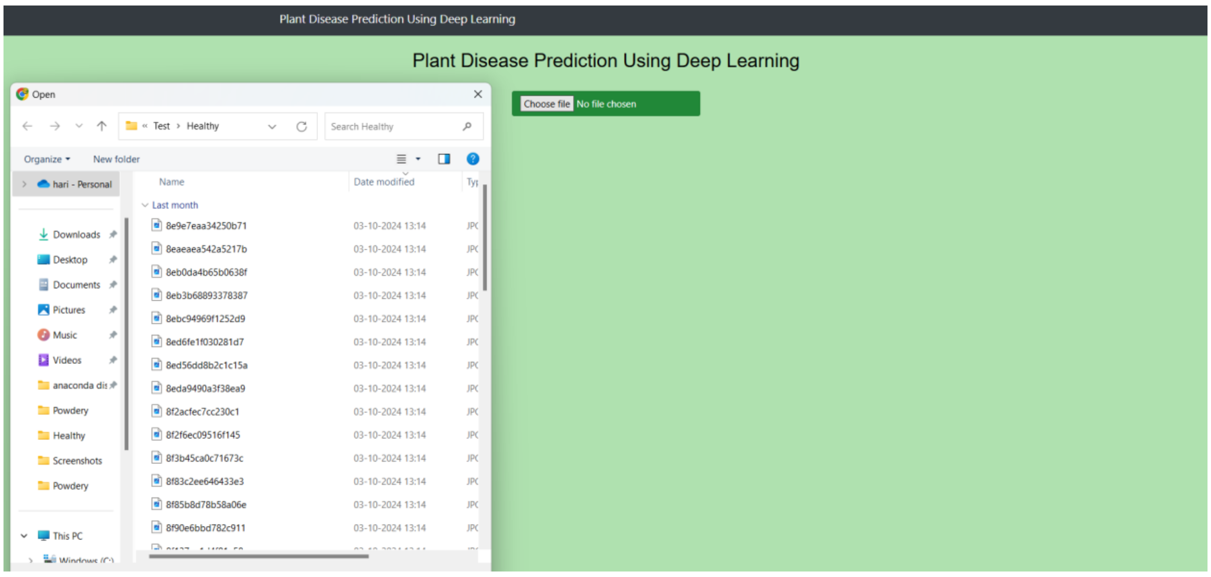
```
padding: 20px;
color: white; / Ensure text is visible against the background /
font-family: 'CustomFont', sans-serif; / Applying custom font to content /
}
h1, h2, h3, h4, h5, h6 {
font-family: 'CustomFont', sans-serif; / Apply the custom font to headings /
}
</style>
</head>
<body>
<nav class="navbar navbar-dark bg-dark" >
<div class="container">
<a class="navbar-brand" href="#" >Plant Disease Prediction Using Deep Learning</a>
</div>
</nav>
{% block content %}
{% endblock %}
</body>
</html>
```

6.2 OUTPUT SCREENS:

6.2.1.Rendered HTML page of Plant leaf disease classification using Deep learning:

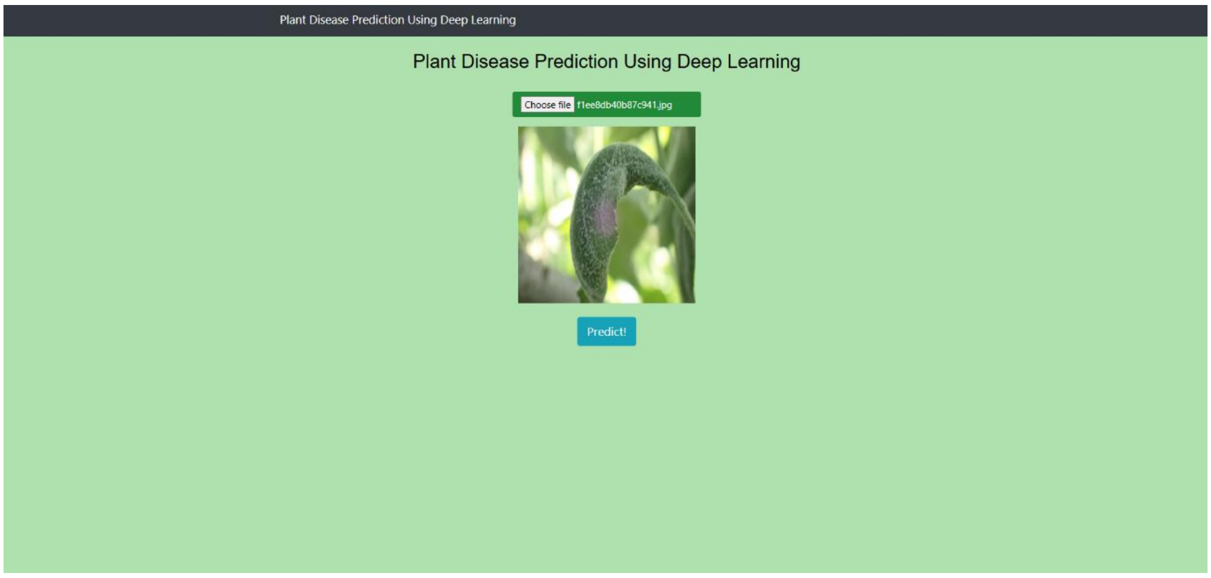

Plant Disease Prediction Using Deep Learning
Plant Disease Prediction Using Deep Learning
Choose file
No file chosen

6.2.2.Selecting the image of a diseased plant from the folder:

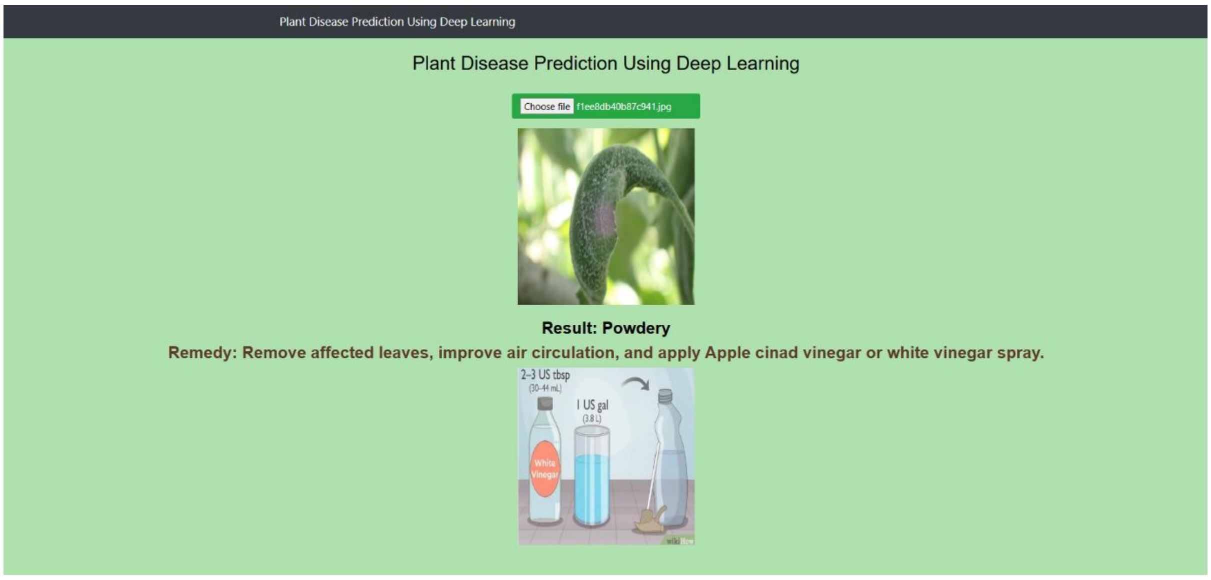


Name	Date modified	Typ
Last month		
8e9e7eaa34250b71	03-10-2024 13:14	JPG
8eaeaea542a5217b	03-10-2024 13:14	JPG
8eb0da4b65b0638f	03-10-2024 13:14	JPG
8eb3b68893378387	03-10-2024 13:14	JPG
8ebc94969f1252d9	03-10-2024 13:14	JPG
8ed6fe1f030281d7	03-10-2024 13:14	JPG
8ed56dd8b2c1c15a	03-10-2024 13:14	JPG
8eda9490a3f38ea9	03-10-2024 13:14	JPG
8f2acfec7cc230c1	03-10-2024 13:14	JPG
8f2f6ec09516f145	03-10-2024 13:14	JPG
8f3b45ca0c71673c	03-10-2024 13:14	JPG
8f83c2ee646433e3	03-10-2024 13:14	JPG
8f85b8d78b58a06e	03-10-2024 13:14	JPG
8f90e6bbd782c911	03-10-2024 13:14	JPG

6.2.3.Uploading the image for predicting the disease of the plant leaf:


Plant Disease Prediction Using Deep Learning
Plant Disease Prediction Using Deep Learning
Choose file f1ee8db40b87c941.jpg
Predict!

6.2.4. Predicting the output of the plant leaf disease with corresponding remedy:

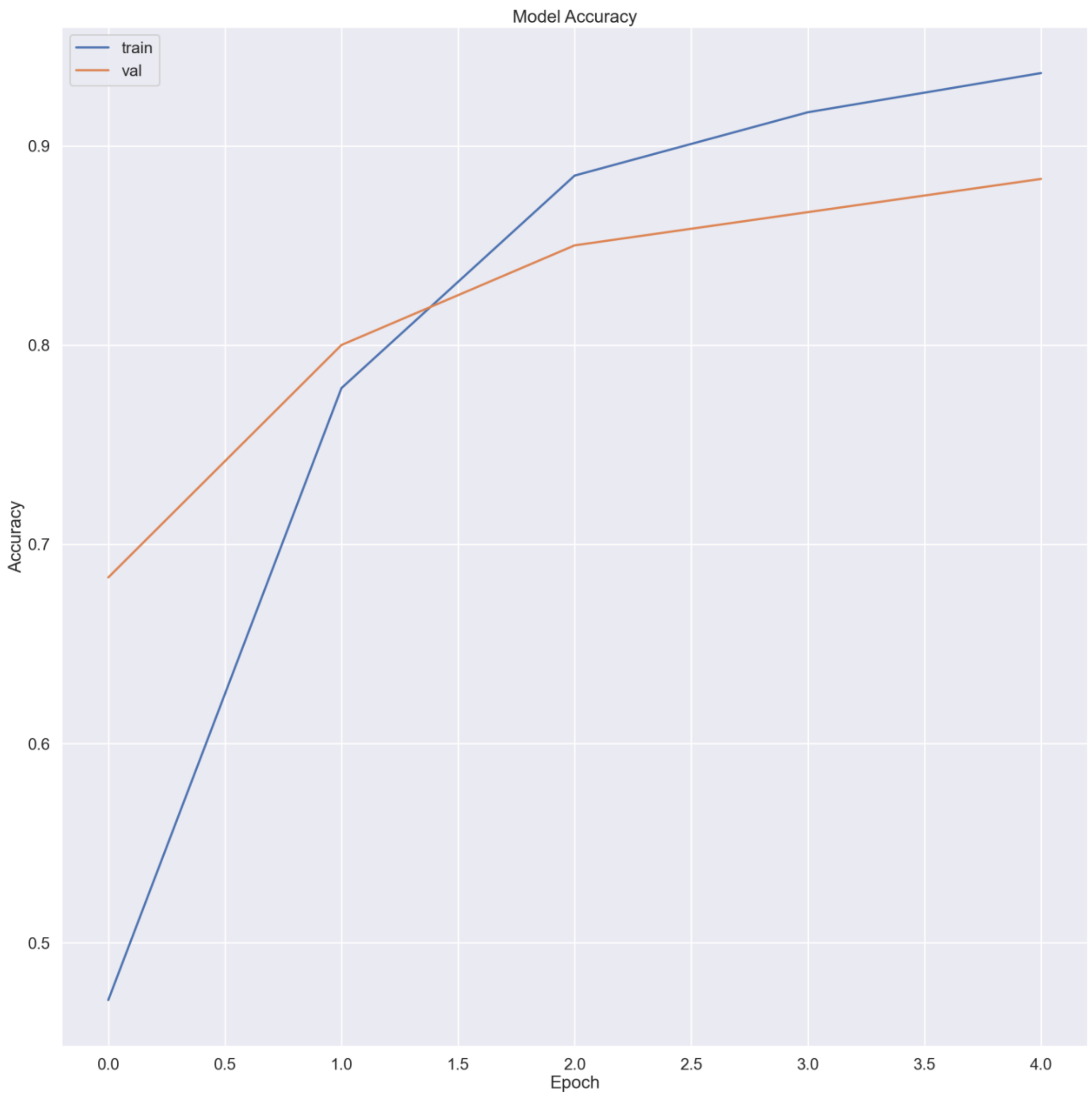

Plant Disease Prediction Using Deep Learning
Plant Disease Prediction Using Deep Learning
Choose file f1ee8db40b87c941.jpg

Result: Powdery

Remedy: Remove affected leaves, improve air circulation, and apply Apple cinad vinegar or white vinegar spray.

2-3 US tbsp
(30-44 mL)
1 US gal
(3.8 L)
White Vinegar

6.2.5 Accuracy of the Model:



7.CONCLUSION & FUTURE SCOPE

CONCLUSION:

The application of deep learning for plant disease detection represents a significant advancement in agricultural practices. By leveraging convolutional neural networks (CNNs) and other deep learning architectures, we can achieve high accuracy in identifying various plant diseases from images of leaves. This technology not only aids in early detection but also helps farmers make informed decisions, ultimately reducing crop loss and increasing food security. The integration of these techniques into agricultural systems can lead to more sustainable farming practices and improved management of resources**.**

FUTURE SCOPE:

The future scope of plant disease detection using deep learning is promising and multifaceted. Advancements in algorithms, including the exploration of generative adversarial networks (GANs) and transformer models, can lead to even higher accuracy and efficiency in disease identification. The integration of real-time monitoring systems utilizing drone technology and IoT devices could facilitate immediate responses to disease outbreaks, enhancing crop management. Developing user-friendly mobile applications will empower farmers to capture images and receive instant diagnoses, making advanced diagnostic tools more accessible. Additionally, combining deep learning with remote sensing and climate data analysis can provide holistic insights into plant health. The use of transfer learning can optimize model performance with limited datasets, particularly in diverse agricultural regions. Collaborative platforms that encourage data sharing among farmers, researchers, and agronomists can enhance collective knowledge and response strategies. Furthermore, educational initiatives utilizing deep learning technologies can empower farmers with essential skills for effective crop management. Overall, these developments have the potential to transform agricultural practices, promote sustainability, and significantly contribute to global food security.

8. BIBLIOGRAPHY

- Ferentinos, K. P. (2018). "Deep learning models for plant disease detection and diagnosis." *Computers and Electronics in Agriculture*, 145, 311-318. DOI: 10.1016/j.compag.2017.05.009
- Mohanty, S. P., Hughes, D. P., & Salathe, M. (2016). "Using Deep Learning for Image-Based Plant Disease Detection." *Frontiers in Plant Science*, 7, 1419. DOI: 10.3389/fpls.2016.01419
- Liu, L., & Wang, S. (2020). "A survey on deep learning for plant disease detection." *Artificial Intelligence Review*, 53(5), 3513-3535. DOI: 10.1007/s10462-019-09783-y
- Thakur, S. R., & Patil, P. D. (2021). "Plant Disease Detection Using Deep Learning Techniques: A Review." *Journal of Agricultural Informatics*, 12(3), 24-40. DOI: 10.17700/jai.2021.12.3.704
- Tian, H., & Guo, J. (2019). "A deep learning framework for the detection of plant diseases using convolutional neural networks." *Computers and Electronics in Agriculture*, 162, 235-245. DOI: 10.1016/j.compag.2019.04.020
- Pérez, J., & Alarcon, J. (2020). "Review on Deep Learning Techniques for Plant Disease Detection." *Sensors*, 20(9), 2612. DOI: 10.3390/s20092612
- Zhang, Z., & Liu, W. (2021). "Plant Disease Detection Using Deep Learning: A Comprehensive Review." *Journal of Plant Diseases and Protection*, 128(5), 431-443. DOI: 10.1007/s41348-021-00413-0
- Khan, M. A., & Khan, M. N. (2020). "Plant Disease Detection Using Transfer Learning: A Review." *Agricultural Sciences*, 11(1), 65-77. DOI: 10.4236/as.2020.111007