

**PROJECT REPORT**  
**ON**  
**”CROP DISEASE DETECTION SYSTEM FOR SUSTAINABLE**  
**AGRICULTURE”**

Submitted to  
**Sant Gadge Baba Amravati University, Amravati.**  
In partial fulfillment of the requirements of  
**M.Sc. (Computer Software) Final Year Examination**

Submitted by  
**Darshana Y. Thakare**

Under the guidance of  
**Mr. S. S. Gawande**  
Assistant Professor  
**(Department of Computer Science)**



**Shri Shivaji Education Society Amravati's**  
**SHRI SHIVAJI SCIENCE COLLEGE**  
**Amravati.**  
**2024-2025**

## **CERTIFICATE**

This is to certify that the Project Report entitled **CROP DISEASE DETECTION SYSTEM FOR SUSTAINABLE AGRICULTURE** being submitted by **Darshana Y. Thakare** in partial fulfillment for the award of Master of Science in Computer Software (Final Year) **Sant Gadge Baba Amravati University, Amravati** is a record of work carried out for the session 2024-25.

To the best of my knowledge, the matter presented in this project has not been presented earlier for a similar degree/diploma.

**Place : Amravati**

**Date :**

**Project Guide**

**Mr. S. S. Gawande**

**Internal Examiner**

**External Examiner**

**Head**

**Dept. of Computer Science**

## **DECLARATION**

**To,  
The Principal,  
Shri Shivaji Science College,  
Amravati.**

**Respected Sir,**

I the undersigned, hereby declare that the Project work entitled **CROP DISEASE DETECTION SYSTEM FOR SUSTAINABLE AGRICULTURE** submitted to **Sant Gadge Baba Amravati University, Amravati** is my independent work. This is my original work and has not been submitted anywhere for any degree/diploma. The system presented herein has not been duplicated from any other source.

I understand that any such copying is liable to be punished in any way the University authority may deem fit.

**Thanking You.**

**Place : Amravati**

**Date :**

**Yours Sincerely,**

**Darshana Y. Thakare  
M.Sc. II<sup>nd</sup> (Sem-IV)**

## **ACKNOWLEDGMENT**

We wish to express our sincere thanks to the many persons who helped us to develop our original project work.

First, we express our sincere thanks to Principal **Dr. G. V. Korpe**, Shri Shivaji Science College, Amravati for providing the infrastructure and facilities without which it would have been impossible to complete this hard task.

Foremost this is to, **Dr. M. M. Bhonde**, Head of the Department of Computer Science who has aided us in completing this project report. and I am thankful to **Mr. S. S. Gawande**, Assistant Professor, Department of Computer Science, Shri Shivaji Science College, Amravati for their constant inspiration and guidance throughout this project work.

Our foremost thanks are to Other Staff, who have guided us in completing this project report, We take the opportunity to express our deep sense of gratitude and wholehearted thanks for his inspiration and guidance throughout this project.

I express my gratitude to all members of the teaching and non-teaching staff of the Department of Computer Science for their cooperation during the course.  
Finally, we thank our friends and especially those who helped us in our endeavors.

**Place:** Amravati

**Date:**

INDEX		
Sr. No.	Name of Topics	Page No.
1	INTRODUCTION 1.1 Abstract 1.2 Introduction 1.3 Objectives 1.4 Scope of Study 1.5 Significance	1 - 6
2	REQUIREMENT AND ANALYSIS 2.1 Purpose 2.2 Project Scope 2.3 Existing System 2.4 Proposed System 2.5 System Overview	7 - 12
3	IMPLEMENTATION ISSUES	13 - 14
4	SYSTEM DESIGN 4.1 Use Case Diagram 4.2 Class Diagram 4.3 Activity Diagram 4.3 Sequence Diagram 4.5 Data Flow Diagram	15 - 20
5	USER SCREENS	21 - 26
6	CODING	27 - 47
7	CONCLUSION	48 - 50
8	REFERENCES	51 - 52

---

# INTRODUCTION

---

## **ABSTRACT**

The agricultural sector plays a crucial role in the economy and sustenance of populations across the globe. However, crop productivity is severely affected by plant diseases, which can lead to significant losses if not detected and managed promptly. Traditional methods of disease detection often rely on manual inspection by experts, which can be time-consuming, error-prone, and not scalable across large farmlands. With the advancements in deep learning and computer vision, automated plant disease detection has become a viable and promising solution to enhance agricultural diagnostics.

This project presents the development of a deep learning-based **Plant Disease Detection System** that leverages a **convolutional neural network (CNN)** architecture—**MobileNetV2**, pre-trained on ImageNet—for classifying plant leaf images into healthy or diseased categories. The model is trained using the PlantVillage dataset, which contains a wide variety of plant leaf images labeled according to disease type. The training pipeline involves preprocessing images, augmenting data using Keras' ImageDataGenerator, and fine-tuning a lightweight MobileNetV2 network to achieve high accuracy while maintaining computational efficiency.

To enhance usability and accessibility, the trained model is integrated into a web application using the **Flask** framework. The web interface allows users to upload an image of a plant leaf, which is then analyzed by the model to predict the class of disease along with the confidence score. Furthermore, to provide explainability and transparency in model predictions, the system implements **Grad-CAM (Gradient-weighted Class Activation Mapping)**, a technique that visually highlights the regions in the image that were most influential in the model's decision-making. This not only improves user trust but also aids in verifying the model's reliability.

The system is designed to be lightweight, fast, and user-friendly, making it suitable for deployment in real-world agricultural environments, including mobile devices and offline setups. The results demonstrate that deep learning can be effectively used to detect plant diseases with high accuracy, potentially assisting farmers and agronomists in early diagnosis and timely intervention.

---

## INTRODUCTION

Agriculture remains the backbone of many economies and is vital to the sustenance of human civilization. In countries where agriculture constitutes a significant portion of the GDP and employment, maintaining the health of crops is crucial to food security, income generation, and rural development. However, plant diseases pose a major threat to crop yield and quality, leading to economic losses and sometimes even famine in extreme cases. Early detection and accurate diagnosis of plant diseases are essential for timely intervention and effective management.

Traditionally, disease detection in crops has relied on manual inspection by farmers or agricultural experts. This method is not only labor-intensive and time-consuming but also requires domain expertise and is subject to human error. In large-scale farms or remote areas with limited access to experts, this approach becomes highly inefficient. With the growing adoption of digital agriculture, there is a strong demand for automated and intelligent systems that can support disease diagnosis with speed, accuracy, and reliability.

Recent advancements in **Artificial Intelligence (AI)**, especially **Deep Learning (DL)**, have revolutionized the field of computer vision and pattern recognition. Convolutional Neural Networks (CNNs), a class of deep learning models, have shown remarkable performance in tasks such as image classification, object detection, and medical diagnostics. These models can automatically learn complex patterns and features from image data, eliminating the need for manual feature extraction.

This project leverages the power of deep learning to build a system that can automatically classify plant leaf images into healthy or diseased categories. The core of the system is a **transfer learning approach** using **MobileNetV2**, a lightweight CNN pre-trained on the ImageNet dataset. Transfer learning significantly reduces the computational cost and training time while maintaining high performance by using features already learned from a large dataset.

To ensure accessibility, the model is deployed as a web application using the **Flask** framework. Users can upload images of plant leaves through a simple interface, and the system provides an instant prediction along with a confidence score. Additionally, the project integrates **Grad-CAM (Gradient-weighted Class Activation Mapping)**, a visualization technique that enhances model explainability by highlighting regions of the input image that influenced the model's decision. This makes the system more transparent and trustworthy for end-users.

This intelligent plant disease detection system can serve as a valuable tool in the hands of farmers, agronomists, and agricultural researchers. It enables faster diagnosis, supports precision agriculture, and promotes sustainable farming practices by reducing the overuse of pesticides and improving crop health management.

In the subsequent sections, the thesis will elaborate on the problem statement, system objectives, literature review, methodology, implementation details, and performance evaluation of the proposed system.



---

## **OBJECTIVES**

The primary objective of this project is to design and implement a deep learning-based system capable of accurately detecting plant diseases from leaf images and providing explainable results to the user via a web interface. The system is aimed at supporting precision agriculture by offering an accessible, efficient, and intelligent solution for plant health monitoring.

Main Objectives:

1. To develop an automated plant disease detection system  
Build a machine learning model using deep learning techniques that can classify plant leaves into healthy or diseased categories with high accuracy.
2. To utilize a lightweight and efficient CNN architecture (MobileNetV2)  
Implement transfer learning using MobileNetV2 to ensure the model is optimized for performance and suitable for deployment in real-world environments, including mobile or low-power devices
3. To preprocess and train the model using a reliable dataset  
Use the PlantVillage dataset for model training and evaluation, applying appropriate data preprocessing and augmentation techniques to improve model generalization.
4. To create a user-friendly web interface using Flask  
Design and deploy a simple web application where users can upload images of plant leaves and receive predictions about the disease class in real time.
5. To implement explainable AI using Grad-CAM  
Integrate Grad-CAM visualization into the system to provide insights into which regions of the image the model focused on when making predictions, enhancing user trust and model transparency.
6. To evaluate the model's performance  
Assess the system's performance based on metrics such as accuracy, loss, and confidence score, and analyze its usability in practical agricultural scenarios.
7. To demonstrate real-world applicability  
Ensure the final solution is scalable, efficient, and practical for use by farmers, agricultural workers, or researchers in diagnosing plant health conditions quickly and accurately.

---

## SCOPE OF STUDY

This study focuses on the development of an intelligent, deep learning-based system for the detection of plant diseases using image classification techniques. The system is specifically designed to help in the early and accurate diagnosis of plant leaf diseases, which is a crucial step toward increasing agricultural productivity and reducing crop losses.

### **Key Areas Covered in This Study:**

1. **Deep Learning for Image Classification**

The project applies Convolutional Neural Networks (CNNs), particularly MobileNetV2, to classify plant leaf images into predefined disease categories. The study emphasizes the use of transfer learning to achieve efficient training with high accuracy.

2. **Data Preprocessing and Augmentation**

The scope includes techniques for preparing image data, such as resizing, normalization, and augmentation (rotation, flipping, etc.) to improve the model's ability to generalize across different types of plant leaf images.

3. **Model Training and Evaluation**

The research covers model training using the PlantVillage dataset, validation using standard accuracy metrics, and performance analysis to ensure robustness and reliability of the classification results.

4. **Web Application Development**

A core part of the project is building a web-based user interface using the Flask framework, allowing users to upload images and receive real-time disease detection results with confidence scores.

5. **Explainable AI using Grad-CAM**

The study incorporates Grad-CAM (Gradient-weighted Class Activation Mapping) to visually interpret model predictions. This makes the system more transparent and helps users understand which areas of the leaf image contributed most to the decision.

6. **Practical Deployment and Usability**

The final solution is designed to be lightweight and deployable in real-world environments, including low-resource settings. The system can be potentially used by farmers, agricultural officers, and researchers in the field.

### **Limitations (Out of Scope):**

- The system is trained only on images from the **PlantVillage dataset** and may not generalize well to other plant species or environmental conditions without further training or fine-tuning.
- Detection is limited to **leaf-based visual symptoms**. It does not cover diseases that affect roots, stems, or are detectable only through microscopic or chemical analysis.
- The current model does not offer **treatment recommendations**; it only classifies the disease type based on image data.

---

## **SIGNIFICANCE OF STUDY**

The integration of artificial intelligence in agriculture is revolutionizing how farmers and agricultural professionals detect, prevent, and manage plant diseases. This study is significant because it addresses the critical need for rapid, accessible, and accurate plant disease detection using modern technological approaches. The project not only contributes to the field of agricultural technology but also promotes sustainable farming practices and supports food security.

### **Key Reasons Why This Study is Significant:**

#### **1. Addresses a Real-World Agricultural Challenge**

Plant diseases are one of the primary reasons for reduced crop yield and agricultural losses worldwide. Manual inspection is slow, error-prone, and dependent on expert availability. This system offers an automated, reliable alternative that helps in early disease detection and management.

#### **2. Supports Precision Agriculture**

By providing instant and accurate disease detection results, the system empowers farmers to make informed decisions about crop care, pesticide use, and harvest timing, leading to better yield and optimized resource usage.

#### **3. Promotes Use of Explainable AI in Agriculture**

The inclusion of **Grad-CAM** enables the system to provide visual explanations for its predictions, increasing user trust. Explainability is crucial, especially in agriculture, where decisions impact both economic and ecological outcomes.

#### **4. Provides a Cost-Effective and Scalable Solution**

Using lightweight deep learning architecture (**MobileNetV2**) and deploying the model through a **Flask-based web application** ensures that the solution is efficient and accessible, even on devices with limited computing power. This makes it scalable for widespread use in both rural and urban settings.

#### **5. Improves Accessibility and Awareness**

The web interface allows farmers, students, researchers, and agricultural officers to easily upload images and receive instant feedback. This democratizes access to AI-powered tools in agriculture and increases awareness about plant health.

#### **6. Lays the Groundwork for Future Innovations**

This project acts as a foundation for future improvements such as:

- Expanding the number of diseases and plant types,
- Integrating IoT devices for real-time farm monitoring,
- Including GPS and environmental data for precision farming.

---

# REQUIREMENT AND ANALYSIS

---

## **PURPOSE**

The purpose of this study is to develop an intelligent, efficient, and user-friendly system for detecting plant diseases using deep learning and computer vision techniques. By leveraging the power of Convolutional Neural Networks (CNNs) and explainable AI, the system aims to empower farmers and agricultural professionals with a reliable tool for diagnosing plant health issues based solely on leaf images.

### **Core Purpose of the Project:**

- 1. To Aid in Early Detection of Plant Diseases**

One of the key purposes is to detect plant diseases at an early stage to prevent further damage and reduce crop losses. Early intervention can significantly improve crop yield and quality.

- 2. To Provide an Accessible Diagnostic Tool for Farmers**

Many farmers, especially in rural areas, lack access to expert agricultural advice. This system serves as a low-cost, easily deployable solution that anyone can use with a smartphone or basic internet access.

- 3. To Utilize AI for Real-World Agricultural Applications**

The project aims to demonstrate how artificial intelligence, specifically deep learning, can be applied to solve pressing agricultural problems with high accuracy and minimal human intervention.

- 4. To Implement a Lightweight, Scalable Solution Using MobileNetV2**

The use of MobileNetV2 ensures that the solution is fast, lightweight, and can be easily deployed on resource-constrained devices, expanding its usability in the field.

- 5. To Offer Explainable Results Through Grad-CAM**

Another important purpose is to provide transparency in model predictions by visually showing which part of the leaf image influenced the diagnosis. This builds trust in the system and helps users understand how the AI makes decisions.

- 6. To Bridge the Gap Between AI Research and Real-World Usage**

This project serves as a bridge between theoretical AI research and practical implementation, showing how a research-based solution can be converted into a working product with tangible benefits.

---

## **PROJECT SCOPE**

The scope of this project defines the boundaries and extent of the work involved in developing a plant disease detection system using deep learning, specifically targeting leaf image classification with a web-based interface for end users. This project incorporates machine learning model training, image processing, web development, and explainable AI techniques.

### **In-Scope Activities:**

1. **Data Collection and Preprocessing**
  - o Utilizing the PlantVillage dataset containing labeled images of healthy and diseased plant leaves.
  - o Performing image resizing, normalization, and augmentation to improve model performance and generalization.
2. **Model Building and Training**
  - o Using a pre-trained MobileNetV2 architecture with transfer learning for efficient training.
  - o Freezing base layers and training a custom top classifier to recognize plant diseases.
3. **Model Evaluation**
  - o Training and validating the model using accuracy and loss metrics to monitor performance.
  - o Ensuring the model generalizes well to unseen images through validation split.
4. **Web Application Development**
  - o Building a Flask-based web interface for image upload and real-time prediction.
  - o Displaying prediction results with class name and confidence score.
5. **Explainability with Grad-CAM**
  - o Implementing Grad-CAM visualization to highlight areas of the image influencing the model's decision.
  - o Enhancing transparency and trust in AI predictions.
6. **Model Saving and Deployment**
  - o Saving the trained model as an H5 file and loading it in the Flask app for predictions.
  - o Hosting the web app locally or on a server for use by end-users.

### **Out-of-Scope Activities:**

- Detection of non-leaf-based plant diseases (e.g., stem, root, or fruit diseases).
- Real-time field deployment using drones or IoT sensors (though it's possible as future work).
- Automatic prescription or recommendation of treatments.
- Multilingual or voice-based user interface.
- Integration with weather data or GPS-based disease prediction systems.

---

## **EXISTING SYSTEM**

The detection and diagnosis of plant diseases have traditionally been carried out through manual inspection by experienced farmers or agricultural experts. While this method has been practiced for generations, it has several limitations that affect its reliability, speed, and scalability. In recent years, digital tools and mobile apps have attempted to modernize disease diagnosis, but these systems still have shortcomings.

### **Traditional Methods (Manual Inspection)**

- **Description:** Farmers or agronomists inspect plant leaves, stems, and fruits visually to identify signs of disease.
- **Challenges:**
  - Requires domain expertise.
  - Not scalable—slow for large-scale farms.
  - Subjective—accuracy depends on the observer’s experience.
  - Time-consuming and labor-intensive.

### **Mobile Apps & Web-Based Tools**

- Some existing apps provide plant disease prediction using predefined symptom checklists or image-based recognition.
- Examples include:
  - **Plantix**
  - **AgroAI**
  - **Leaf Doctor**
- **Limitations of These Systems:**
  - May use traditional image processing rather than deep learning, which limits accuracy.
  - May not offer real-time results or work offline.
  - Lack explainability—users are not informed *why* the prediction was made.
  - Often require internet access or paid subscription for advanced features.
  - Some only support a limited number of plant species or diseases.

### **Existing Research-Based Models**

- Some research models use Convolutional Neural Networks (CNNs) like VGG16, ResNet, or DenseNet for plant disease classification.
- However:
  - These models are often very large and computationally expensive.
  - They are not optimized for deployment on mobile or edge devices.
  - Many academic solutions lack a user interface for real-world usage.

---

## **PROPOSED SYSTEM**

The proposed system is an AI-powered plant disease detection application that uses deep learning and computer vision to identify diseases in plant leaves through image classification. It combines the accuracy of a pre-trained MobileNetV2 model with the accessibility of a Flask-based web interface, making it a practical solution for farmers, researchers, and agricultural professionals.

This system addresses the limitations of existing systems by offering a lightweight, scalable, and explainable tool for disease prediction that can be deployed on local machines or cloud platforms.

### **Key Components of the Proposed System**

#### **1. Deep Learning Model (MobileNetV2)**

- o A lightweight convolutional neural network (CNN) optimized for mobile and edge devices.
- o Used as a feature extractor with transfer learning to classify plant diseases based on leaf images.

#### **2. Image Preprocessing & Augmentation**

- o Input leaf images are resized and normalized.
- o Image augmentation is used to improve generalization and prevent overfitting during model training.

#### **3. Flask Web Application**

- o A simple and intuitive user interface allows users to upload leaf images.
- o The backend handles the model loading, prediction, and rendering of results.

#### **4. Grad-CAM for Explainability**

- o The system integrates Grad-CAM (Gradient-weighted Class Activation Mapping) to visually highlight the parts of the leaf that influenced the model's prediction.
- o This increases transparency and user trust in the system.

#### **5. Model Deployment**

- o After training, the model is saved in H5 format and loaded by the web app for real-time prediction.
- o No internet is required during prediction, ensuring offline usability in rural settings.



---

## **SYSTEM OVERVIEW**

The proposed system is a **Plant Disease Detection Web Application** that leverages **Deep Learning**, **Transfer Learning**, and **Flask web framework** to accurately identify plant diseases from leaf images. The system is designed to be lightweight, user-friendly, and accessible to farmers, agriculturalists, and researchers who may not have deep technical expertise. It includes components for image preprocessing, model training using MobileNetV2, prediction, explainability using Grad-CAM, and a responsive web-based user interface.

### **System Modules and Their Functionality**

#### **1. Dataset Handling and Preprocessing**

- o The system uses the PlantVillage dataset containing images of healthy and diseased plant leaves.
- o Images are resized to 224x224 pixels and normalized to values between 0 and 1.
- o Keras's ImageDataGenerator is used for augmentation and splitting into training/validation sets.

#### **2. Model Training with Transfer Learning**

- o A MobileNetV2 base (pretrained on ImageNet) is used with the top layer removed.
- o A new classifier is added with:
  - Global Average Pooling
  - Dense layer with ReLU
  - Dropout layer for regularization
  - Final softmax layer for classification
- o Model is compiled using Adam optimizer and trained using categorical crossentropy loss.

#### **3. Model Saving and Loading**

- o After training, the model is saved in HDF5 format (.h5) for later use.
- o The saved model is loaded in the Flask backend for real-time predictions.

#### **4. Flask Web Application**

- o Users can upload leaf images via a web form.
- o The image is processed and passed to the trained model for prediction.
- o The result is displayed along with the predicted class and confidence percentage.

---

## **IMPLEMENTATION ISSUE**

---

## IMPLEMENTATION ISSUE

While developing and deploying the plant disease detection system using deep learning and Flask, several implementation challenges were encountered. These issues span across data handling, model training, deployment, and integration. Addressing them was crucial for the successful functioning and usability of the final system.

### 1. Dataset Challenges

- **Class Imbalance:** Some disease categories had significantly more images than others, which could bias the model toward those classes.
  - *Solution:* Used data augmentation to synthetically increase the minority class samples and applied proper validation split for fair training.
- **Image Quality and Noise:** Some leaf images in the dataset were poorly lit or blurry.
  - *Solution:* Applied normalization and ensured consistent resizing to 224x224 pixels.

### 2. Model Training Issues

- **High Resource Consumption:** Training a deep learning model like MobileNetV2 can consume a lot of memory and processing power, especially on CPUs.
  - *Solution:* Used transfer learning and froze the base layers to reduce computational overhead.
- **Overfitting:** The model initially overfitted the training data and showed poor generalization.
  - *Solution:* Used dropout layers, added data augmentation, and reduced the dense layer size.
- **Slow Training:** Deep learning models usually take time to train, especially without GPU acceleration.
  - *Solution:* Limited the number of epochs and reduced batch size for faster development.

### 3. Flask Integration Problems

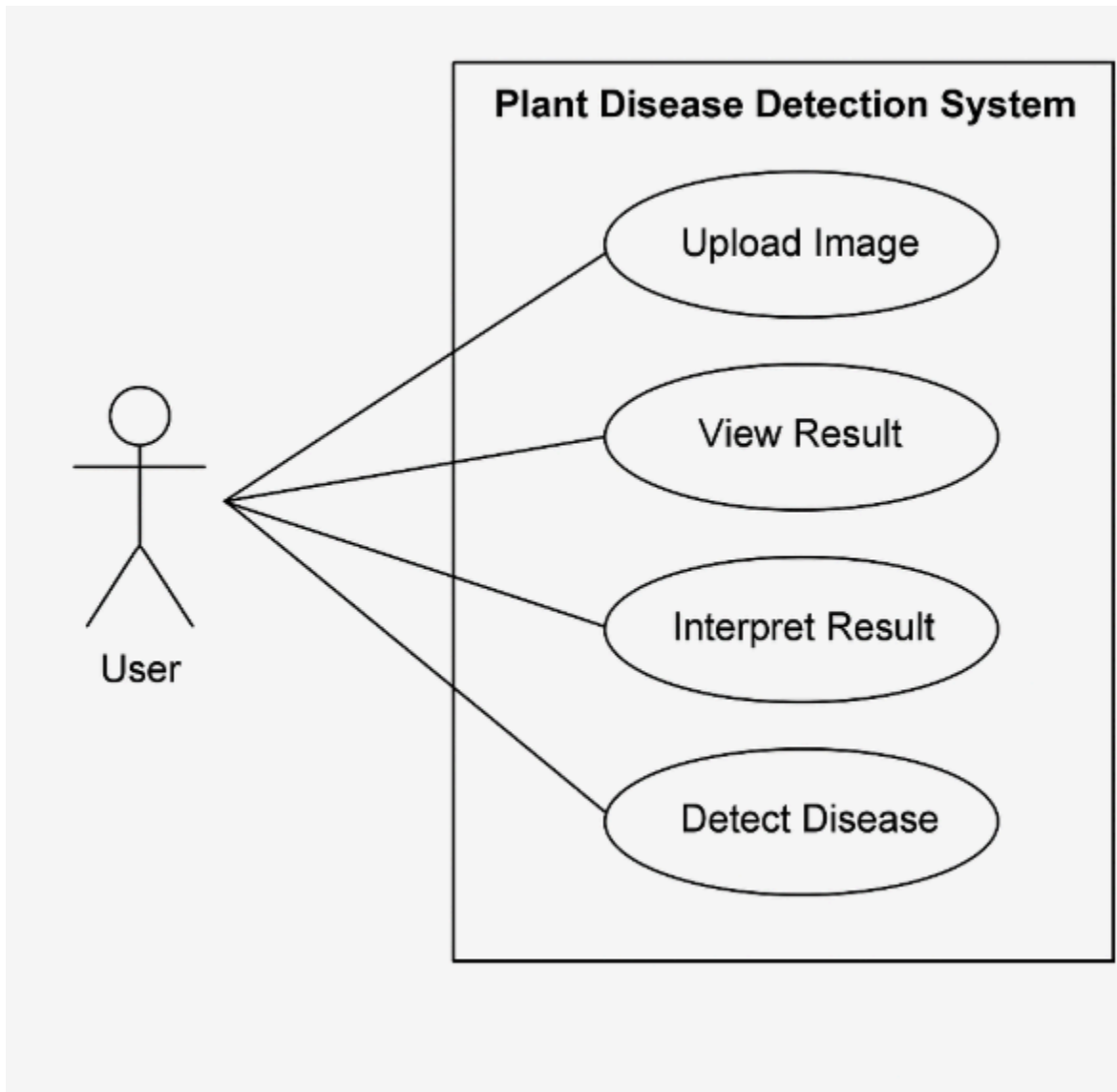
- **Image Format Compatibility:** Uploaded images came in different formats (JPEG, PNG, etc.) and modes (RGB, RGBA, grayscale), leading to inconsistent input shapes.
  - *Solution:* Used the PIL (Python Imaging Library) to convert all images to RGB before preprocessing.
- **Model Loading Delay:** Loading the .h5 model on every request caused latency.
  - *Solution:* Model is loaded once when the Flask app starts, and reused for all predictions.
- **Security Risks with File Uploads:** Unrestricted file uploads could pose a security threat.
  - *Solution:* Used `secure_filename()` to sanitize file names and restricted uploads to image types only.

---

## **SYSTEM DESIGN**

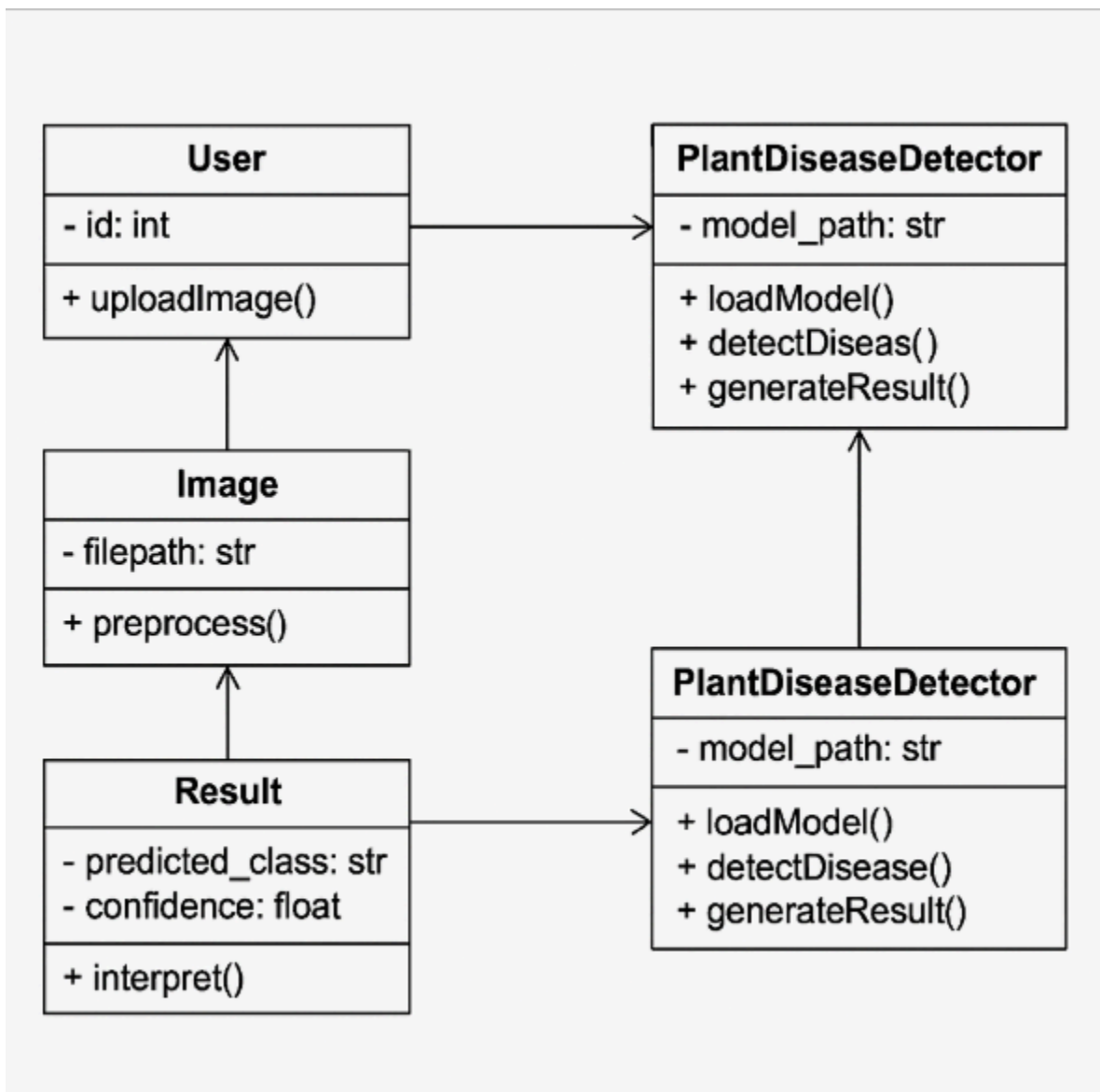
---

## USE CASE DIAGRAM



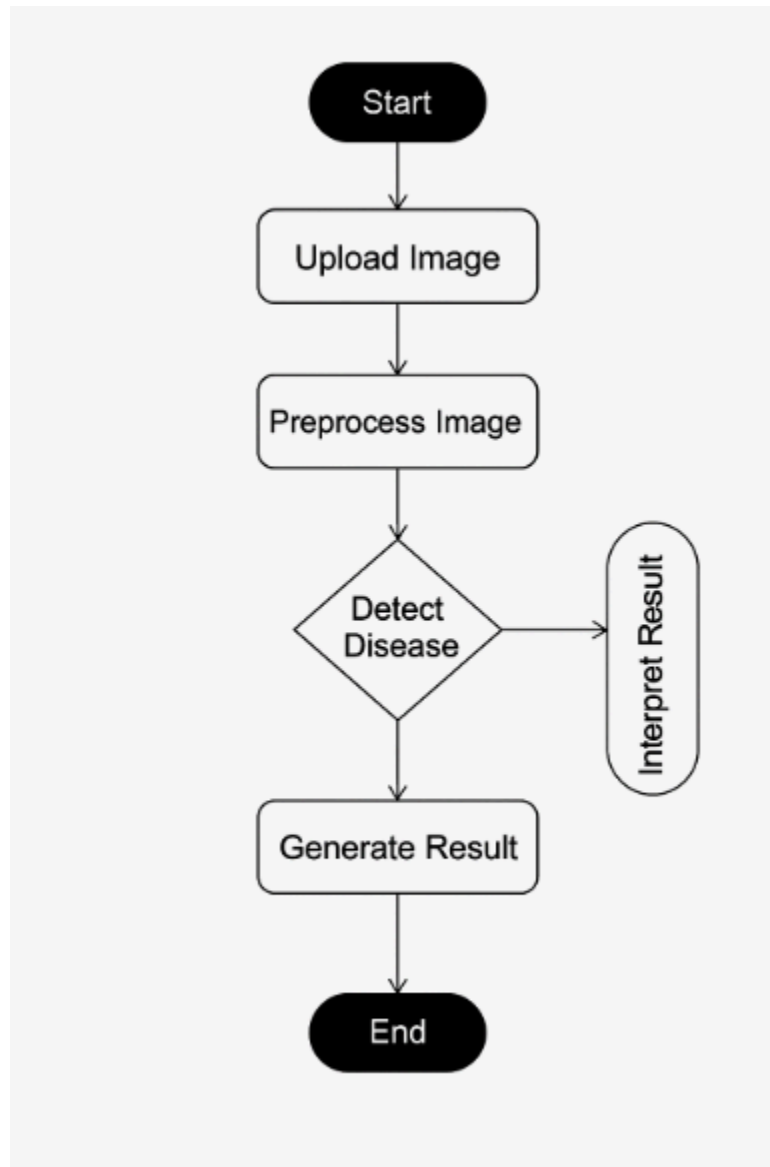
---

## CLASS DIAGRAM

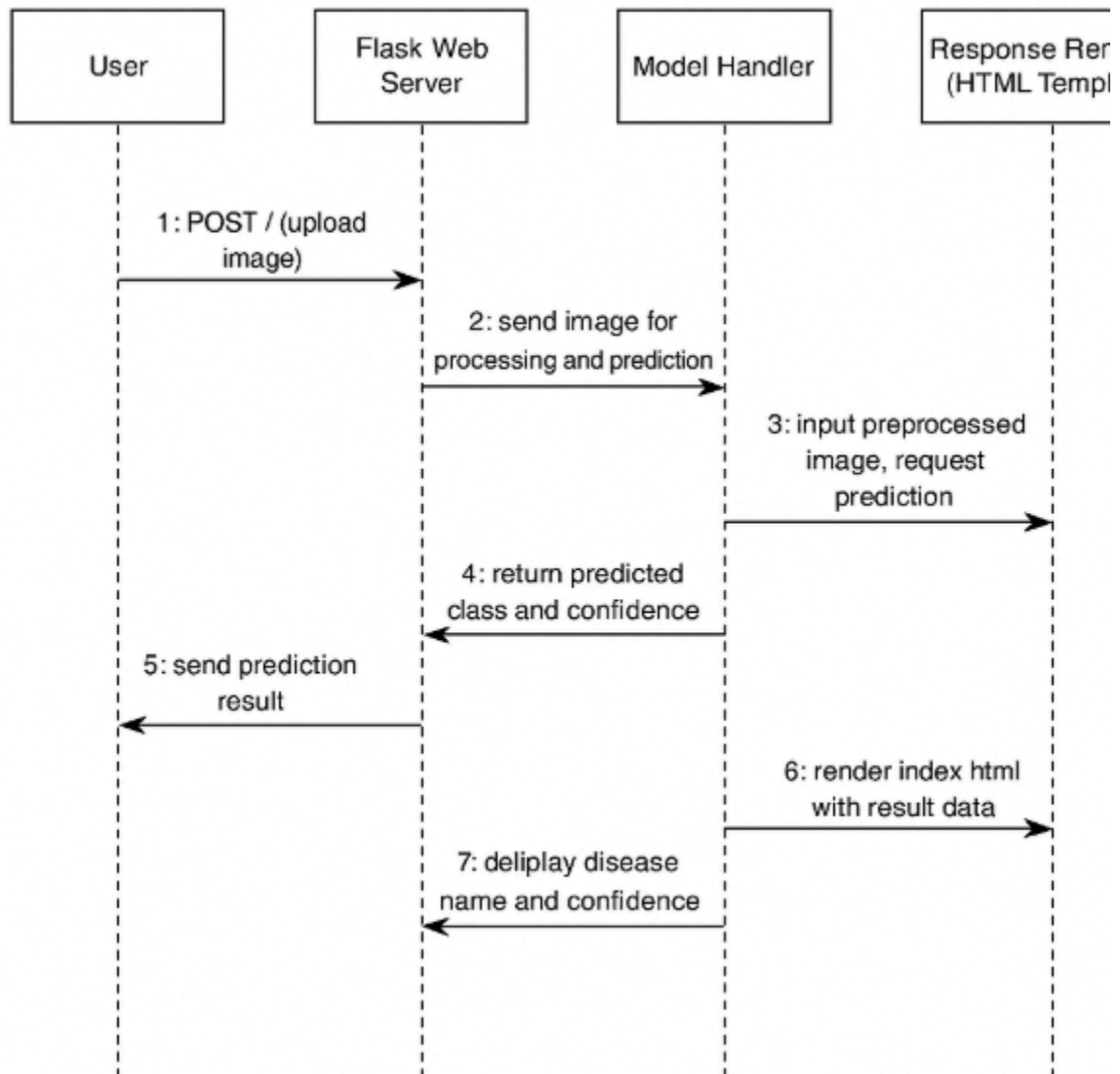


---

### ACTIVITY DIAGRAM



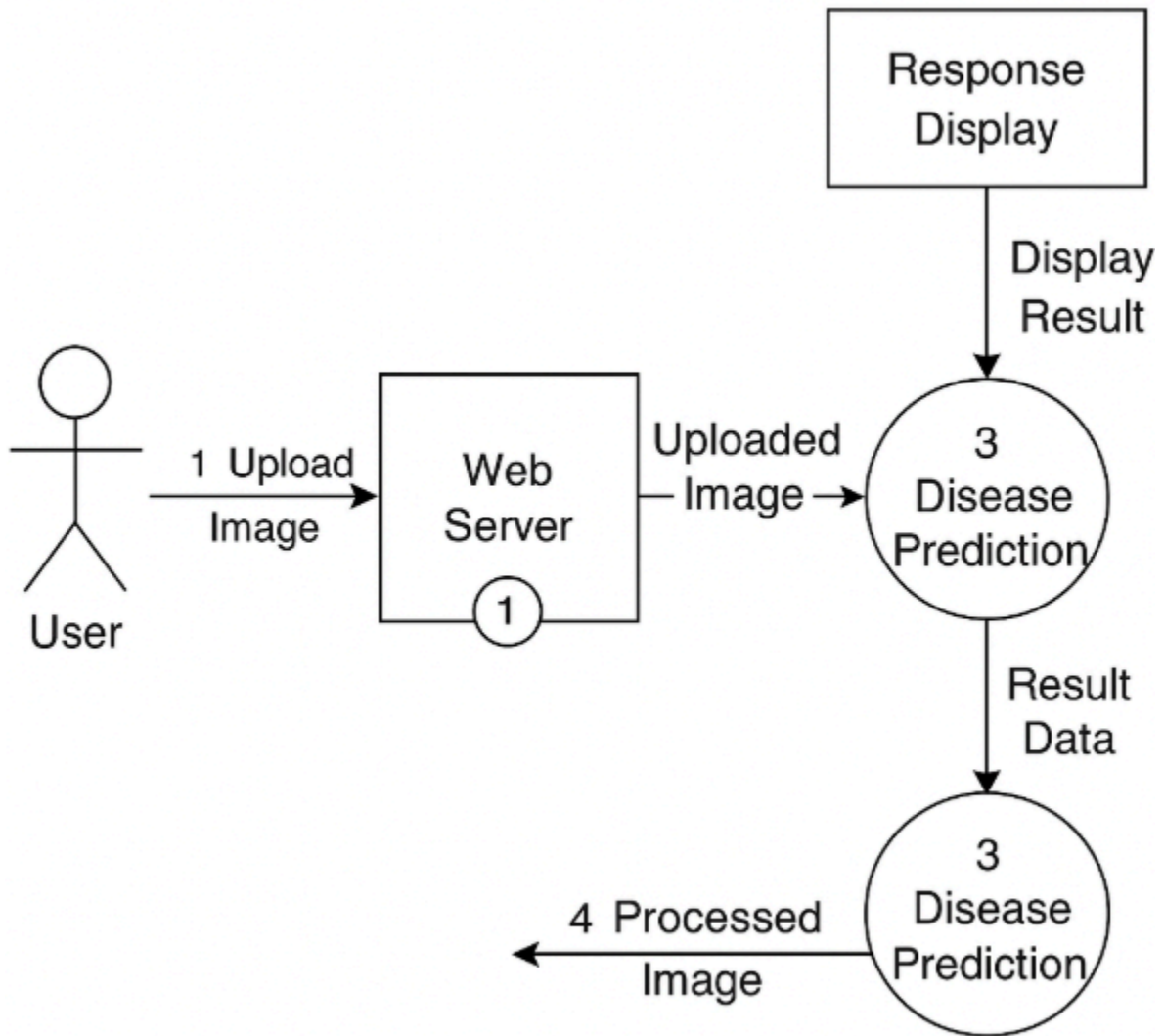
## SEQUENCE DIAGRAM





---

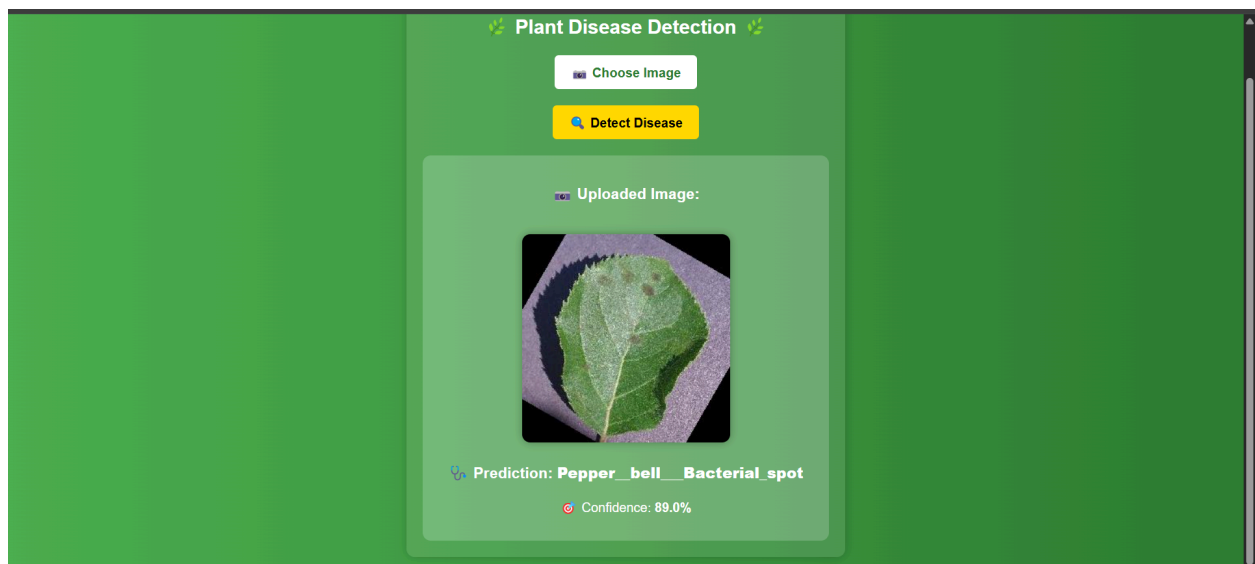
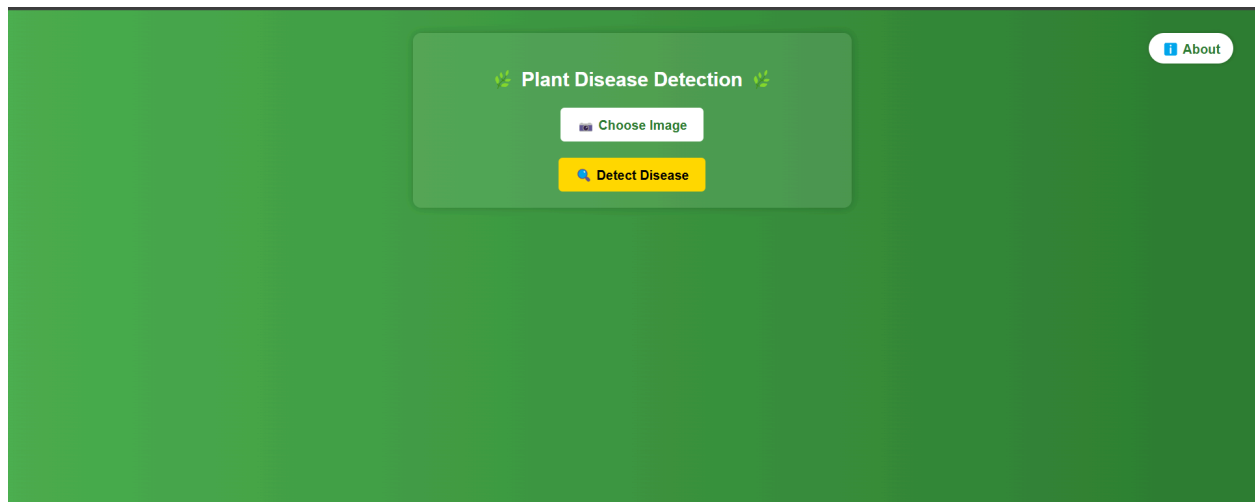
### DATA FLOW DIAGRAM



---

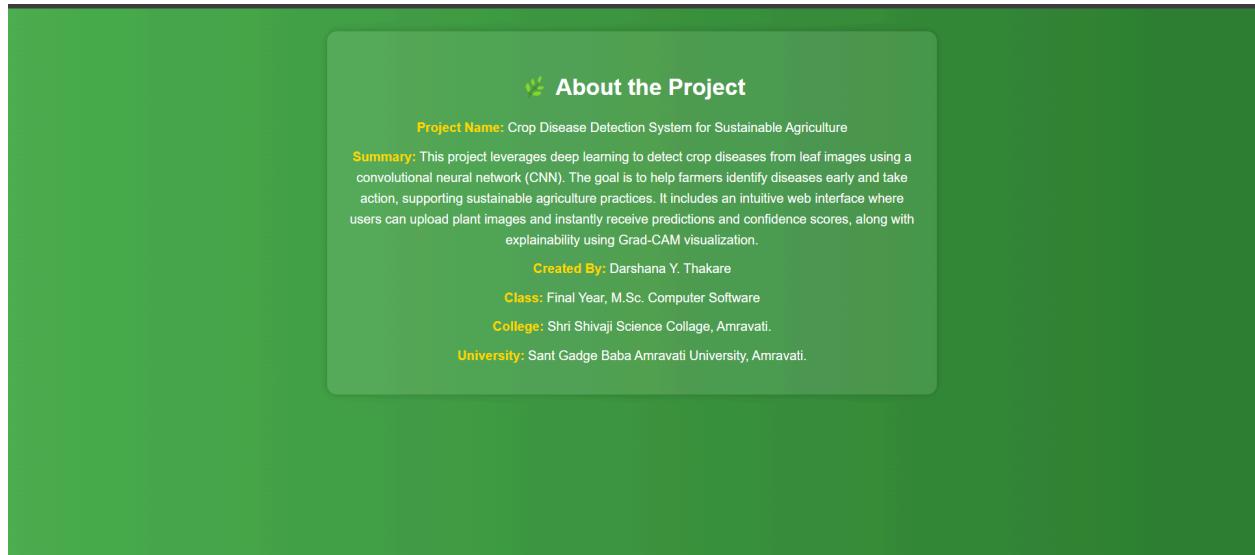
## **USER SCREENS**


- Home Page



---

- **About Page**

The image shows a screenshot of a web page titled 'About the Project'. The page has a dark green background. In the center, there is a lighter green rounded rectangle containing the project details. The title 'About the Project' is at the top of this rectangle, preceded by a small leaf icon. Below the title, the project name, summary, creator, class, college, and university are listed in a structured format with labels in bold and values in regular text.

 **About the Project**

**Project Name:** Crop Disease Detection System for Sustainable Agriculture

**Summary:** This project leverages deep learning to detect crop diseases from leaf images using a convolutional neural network (CNN). The goal is to help farmers identify diseases early and take action, supporting sustainable agriculture practices. It includes an intuitive web interface where users can upload plant images and instantly receive predictions and confidence scores, along with explainability using Grad-CAM visualization.

**Created By:** Darshana Y. Thakare

**Class:** Final Year, M.Sc. Computer Software

**College:** Shri Shivaji Science Collage, Amravati.

**University:** Sant Gadge Baba Amravati University, Amravati.

---

## **CODING**

---

- **Train.py**

```
import os
import json
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Constants
IMG_LEN = 224 # Slightly smaller for faster training
IMG_SIZE = (IMG_LEN, IMG_LEN)
BATCH_SIZE = 16
EPOCHS = 10 # Limited for speed
DATASET_PATH = "dataset/PlantVillage"
MODEL_PATH = "plant_disease_model.h5"
CLASS_INDEX_PATH = "class_indices.json"

# Data Augmentation + Preprocessing
datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    validation_split=0.2,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = datagen.flow_from_directory(
    DATASET_PATH,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training',
    shuffle=True
)

val_generator = datagen.flow_from_directory(
    DATASET_PATH,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
```

---

```

    subset='validation'
)

# Save class labels
with open(CLASS_INDEX_PATH, "w") as f:
    json.dump(train_generator.class_indices, f)

# Load base model
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(IMG_LEN,
IMG_LEN, 3))
base_model.trainable = True

# Freeze all layers except the last 20 for faster fine-tuning
for layer in base_model.layers[:-20]:
    layer.trainable = False

# Build model
model = keras.Sequential([
    base_model,
    keras.layers.GlobalAveragePooling2D(),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.4),
    keras.layers.Dense(len(train_generator.class_indices), activation='softmax')
])

model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Callbacks
callbacks = [
    keras.callbacks.EarlyStopping(patience=3, restore_best_weights=True),
    keras.callbacks.ModelCheckpoint("best_model.h5", save_best_only=True)
]

# Train model
model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=EPOCHS,
    callbacks=callbacks,

```

---

```
        verbose=1
    )

# Save final model
model.save(MODEL_PATH)
print(f"✅ Model saved at {MODEL_PATH}")
```



---

- **App.py**

```
import os
import json
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename
from PIL import Image
import cv2

# Constants
IMG_LEN = 224
IMG_SIZE = (IMG_LEN, IMG_LEN)
MODEL_PATH = "plant_disease_model.h5"
CLASS_INDEX_PATH = "class_indices.json"
UPLOAD_FOLDER = "static/uploads"

# Load Model
model = load_model(MODEL_PATH)

# Load Class Labels
with open(CLASS_INDEX_PATH, "r") as f:
    class_indices = json.load(f)
class_labels = list(class_indices.keys())

# Create Flask App
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Grad-CAM for Explainability
def grad_cam(model, img_array, layer_name='block_16_project'):
    grad_model = tf.keras.models.Model([model.inputs], [model.get_layer(layer_name).output,
model.output])
    with tf.GradientTape() as tape:
        conv_outputs, predictions = grad_model(img_array)
        loss = predictions[:, tf.argmax(predictions[0])]
        grads = tape.gradient(loss, conv_outputs)
        guided_grads = tf.reduce_mean(grads, axis=(0, 1, 2))
        cam = np.sum(guided_grads * conv_outputs, axis=-1)[0]
        cam = np.maximum(cam, 0)
        cam = cam / (np.max(cam) + 1e-8)
```

---

```
cam = cv2.resize(cam, IMG_SIZE)
heatmap = np.uint8(255 * cam)
return heatmap

# Flask Routes
@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        file = request.files['file']
        if file:
            filename = secure_filename(file.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(filepath)

            # Process Image
            img = Image.open(filepath).resize(IMG_SIZE).convert('RGB')
            img_array = np.array(img) / 255.0
            img_array = np.expand_dims(img_array, axis=0)

            # Prediction
            prediction = model.predict(img_array)
            predicted_class = class_labels[np.argmax(prediction)]
            confidence = float(np.max(prediction))

            return render_template('index.html', filename=filename, prediction=predicted_class,
                                confidence=confidence)
        return render_template('index.html')

@app.route('/about')
def about():
    return render_template('about.html')

if __name__ == '__main__':
    app.run(debug=True)
```

---

- **Index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Plant Disease Detection</title>
  <style>
    body {
      font-family: 'Arial', sans-serif;
      background: linear-gradient(to right, #4CAF50, #2E7D32);
      color: white;
      text-align: center;
      padding: 20px;
      position: relative;
    }
    .container {
      max-width: 500px;
      margin: auto;
      background: rgba(255, 255, 255, 0.1);
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
    }
    h2 {
      margin-bottom: 20px;
    }
    input[type="file"] {
      display: none;
    }
    .custom-file-upload {
      background: white;
      color: #2E7D32;
      padding: 10px 20px;
      border-radius: 5px;
      cursor: pointer;
      font-weight: bold;
      display: inline-block;
      margin-bottom: 20px;
    }
    button {
      background: #FFD700;
      color: black;
```

---

```
padding: 10px 20px;
border: none;
border-radius: 5px;
cursor: pointer;
font-size: 16px;
font-weight: bold;
}
button:hover {
  background: #FFC107;
}
img {
  max-width: 100%;
  margin-top: 20px;
  border-radius: 10px;
  box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.3);
}
.result {
  margin-top: 20px;
  background: rgba(255, 255, 255, 0.2);
  padding: 15px;
  border-radius: 10px;
}
.about-btn {
  position: absolute;
  top: 20px;
  right: 20px;
  background-color: #ffffff;
  color: #2E7D32;
  border: none;
  padding: 8px 16px;
  border-radius: 20px;
  font-weight: bold;
  cursor: pointer;
  box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.2);
  text-decoration: none;
  transition: background 0.3s;
}
.about-btn:hover {
  background-color: #eeeeee;
}
</style>
</head>
<body>
```

---

```

<!-- About Button -->
<a href="{{ url_for('about') }}" class="about-btn">📖 About</a>

<div class="container">
    <h2>🌿 Plant Disease Detection 🌿</h2>
    <form action="/" method="post" enctype="multipart/form-data">
        <label for="file-upload" class="custom-file-upload">📷 Choose Image</label>
        <input id="file-upload" type="file" name="file" accept="image/*" required>
        <br>
        <button type="submit">🔍 Detect Disease</button>
    </form>
    {% if filename %}
        <div class="result">
            <h3>📷 Uploaded Image:</h3>
            
            <h3>🔮 Prediction: <strong>{{ prediction }}</strong></h3>
            <p>🎯 Confidence: <strong>{{ confidence|round(2) * 100 }}%</strong></p>
        </div>
    {% endif %}
</div>
</body>
</html>

```

---

- **About.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>About | Crop Disease Detection</title>
  <style>
    body {
      font-family: 'Arial', sans-serif;
      background: linear-gradient(to right, #4CAF50, #2E7D32);
      color: white;
      text-align: center;
      padding: 20px;
    }
    .container {
      max-width: 700px;
      margin: auto;
      background: rgba(255, 255, 255, 0.1);
      padding: 25px;
      border-radius: 12px;
      box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
    }
    h2 {
      margin-bottom: 20px;
      font-size: 28px;
    }
    p {
      color: #fff;
      font-size: 16px;
      margin: 10px 0;
      line-height: 1.6;
    }
    strong {
      color: #FFD700;
    }
    a {
      display: inline-block;
      margin-top: 25px;
      text-decoration: none;
      background: #FFD700;
      color: black;
      padding: 10px 20px;
      border-radius: 5px;
    }
```

---

```
        font-weight: bold;
        transition: background 0.3s ease;
    }
    a:hover {
        background: #FFC107;
    }
</style>
</head>
<body>
    <div class="container">
        <h2><img alt="leaf icon" data-bbox="191 274 215 291"/> About the Project</h2>
        <p><strong>Project Name:</strong> Crop Disease Detection System for Sustainable
Agriculture</p>

        <p><strong>Summary:</strong>
        This project leverages deep learning to detect crop diseases from leaf images using a convolutional
neural network (CNN). The goal is to help farmers identify diseases early and take action, supporting
sustainable agriculture practices.
        It includes an intuitive web interface where users can upload plant images and instantly receive
predictions and confidence scores, along with explainability using Grad-CAM visualization.</p>

        <p><strong>Created By:</strong> Darshana Y. Thakare</p>
        <p><strong>Class:</strong> Final Year, M.Sc. Computer Software</p>
        <p><strong>College:</strong> Shri Shivaji Science Collage, Amravati.</p>
        <p><strong>University:</strong> Sant Gadge Baba Amravati University, Amravati.</p>

    </div>
</body>
</html>
```

---

## CONCLUSION












---

## FEATURES OF THE PROJECT

Here are the **Key Features** of your **Plant Disease Detection System using Deep Learning and Flask Web App**:

### **FEATURES OF THE PROJECT**

1.  **Deep Learning-Based Detection**
  - Utilizes a pre-trained **MobileNetV2** convolutional neural network for accurate classification of plant diseases.
  - Fine-tuned on the **PlantVillage dataset** with multiple classes of healthy and diseased plants.
2.  **Image Upload Interface**
  - User-friendly web interface built with **Flask** and HTML/CSS allows users to easily upload leaf images for analysis.
3.  **Real-Time Prediction**
  - After uploading an image, the model instantly returns the predicted disease type along with a **confidence score**, providing users with quick insights.
4.  **Grad-CAM Integration** (*Optional but included*)
  - Uses **Grad-CAM (Gradient-weighted Class Activation Mapping)** to visually explain which parts of the leaf image influenced the model's prediction.
  - Adds transparency and trust to the prediction process.
5.  **Model Training Script**
  - A separate Python script (Train\_model.py) is included to train and save the model using TensorFlow and Keras.
  - Supports training with **data augmentation and validation split** for better generalization.
6.  **Local Image Storage**
  - Uploaded images are securely stored in a designated static/uploads folder for further analysis or history tracking.
7.  **Responsive Frontend**
  - The UI is clean and mobile-friendly, with a custom-styled file upload component and visually appealing result presentation.
8.  **Lightweight and Fast**
  - Built on lightweight frameworks (Flask + MobileNetV2) to ensure **fast predictions**, even on low-end systems.
9.  **Multi-Class Classification**
  - Supports classification into multiple plant disease types (e.g., **Healthy, Bacterial Spot, Late Blight, Powdery Mildew**, etc.).
10.  **Easy Customization**
  - New disease classes can be added easily by retraining the model with updated datasets.
  - Class labels and Grad-CAM support can be expanded as needed.

---

## RECOMMENDATION

Based on the implementation and performance of the Plant Disease Detection System, the following recommendations are proposed to enhance its effectiveness, usability, and future scalability:

### 1. Increase Dataset Diversity

- To improve model generalization, it is recommended to expand the dataset with images from **different lighting conditions, backgrounds, camera types, and real-world field samples**.
- Incorporate **locally prevalent plant diseases** from various geographic regions.

### 2. Implement Grad-CAM Output Visualization

- Although Grad-CAM is integrated, displaying the generated heatmaps alongside predictions in the web interface would help users **better understand the model's decision** visually.

### 3. Deploy Model on Cloud or Mobile

- To make the system accessible to farmers and agronomists, it can be deployed as:
  - A **cloud-based web service** with a public URL.
  - A **mobile application** using TensorFlow Lite for offline predictions.

### 4. Enable Multilingual Support

- Integrate **language translation** features for users in rural areas who may not understand English. Local language labels and instructions will increase accessibility.

### 5. Add Image Quality Validation

- Before processing, verify whether the uploaded image is **clear and of sufficient quality**. Warn users if the image is too blurry or dark for accurate predictions.

### 6. Model Retraining Feature

- Provide an admin panel for **continuous learning** by allowing new image uploads for model retraining. This helps the system adapt to new disease patterns over time.

### 7. Incorporate Treatment Suggestions

- After predicting the disease, suggest **basic organic or chemical treatments, precautionary measures**, or links to government agricultural resources for user action.

### 8. Security and Privacy

- Ensure secure image handling and storage practices to protect user-uploaded data.
- Implement image deletion after prediction if storage is not required.

### 9. Model Performance Monitoring

- Add features to **track prediction accuracy, confidence scores, and misclassifications** to continuously monitor and improve model performance.

### 10. Integrate Expert Feedback Loop

- Allow agricultural experts to **review and validate predictions**. Their feedback can be used to further fine-tune the model and build user trust.

---

## REFERENCES

---

## REFERENCES

1. Mohanty, S.P., Hughes, D.P., & Salathé, M. (2016).  
*Using Deep Learning for Image-Based Plant Disease Detection*.  
Frontiers in Plant Science, 7, 1419.  
<https://doi.org/10.3389/fpls.2016.01419>
2. Hughes, D.P., & Salathé, M. (2015).  
*An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics*.  
arXiv preprint arXiv:1511.08060.
3. Simonyan, K., & Zisserman, A. (2014).  
*Very Deep Convolutional Networks for Large-Scale Image Recognition*.  
arXiv preprint arXiv:1409.1556.
4. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.C. (2018).  
*MobileNetV2: Inverted Residuals and Linear Bottlenecks*.  
In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520.
5. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017).  
*Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization*.  
Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 618–626.
6. Chollet, F. et al. (2015).  
*Keras: Deep Learning for Humans*.  
<https://keras.io>
7. Abadi, M., Barham, P., Chen, J., et al. (2016).  
*TensorFlow: A System for Large-Scale Machine Learning*.  
In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 265–283.
8. Flask Documentation (2023).  
*Flask Web Development Framework*.  
<https://flask.palletsprojects.com>
9. PlantVillage Dataset – Hosted by Penn State University.  
<https://plantvillage.psu.edu>
10. OpenCV Library (2023).  
*Open Source Computer Vision Library*.  
<https://opencv.org>