Plant Disease Detection System

A PROJECT REPORT

Submitted in partial fulfillment of the requirements
for the award of degree of
BACHELOR OF COMPUTER APPLICATION

Approved by:

Prof. Pratap Sharma

Submitted by:

Sweta Singh (32301222033)

Sayan Golder (32301222031)

Kunal Gautam (32301222091)

Dr. B.C. Roy Academy of Professional Courses, Durgapur

(Formerly Dr.B.C. Roy Engineering College)

DURGAPUR | AFFILIATEF TO MAKAUT (FORMERLY WBUT)

Self-Certificate

This is to certify that the dissertation/project proposal entitled "**Plant Disease Detection System**" is done by us, is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of **Bachelor of Computer Application** under the guidance of **Prof. Pratap Sharma**. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Name of the Student(s):

Sweta Singh : - _____

Sayan Golder : - _____

Kunal Gautam :- _____

Certificate by Guide

This is to certify that this project entitled "**Plant Disease Detection System**" submitted in partial fulfilment of the degree of **Bachelor of Computer Application** by Sweta Singh, Kunal Gautam, Sayan Golder, students **of**
Dr. BC Roy Academy of Professional Courses, Durgapur is an authentic work carried out under my guidance & best of our knowledge and beliefs.

_____

(Signature with date)

Certificate of Approval

This is to certify that this documentation of Minor Project 2024, entitled "**Plant Disease Detection System**" is a record of bona-fide work, carried out by **Sweta Singh, Kunal Gautam and Sayan Golder,** under my supervision and guidance. In my opinion, the report in its present form is in fulfilment of all the requirements, as specified by the **Dr. B.C Roy Academy of Professional Courses, Durgapur** and as per regulations of this college. In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for **Bachelor of Computer Application**.

Prof. Pratap Sharma

PROFESSOR OF DR. BC ROY ACADEMY OF PROFESSIONAL COURSES, JEMUA ROAD, FULJHOR DURGAPUR – 713202

CONTENT

Acknowledgement

We would like to take this opportunity to thank our valued guide (**Prof. Pratap Sharma**) for their invaluable inspiration, astute direction, and tireless assistance, all of which made it possible for us to carry out and finish this effort.

We respect (**Prof. Rita Ganguly**) for her dedication to the job and the effort put forward, as well as for giving up her valuable time to hear our concerns and provide enlightening responses.

We sincerely thank (**Prof. Dr. Rajib Roy**) for providing all the facilities necessary for the course of study.

We would like to take this opportunity to offer our sincere gratitude to all of the faculty members at (**B.C.A**) for their assistance and encouragement.

We also want to express our gratitude to all of our friends who directly or indirectly assisted us in finishing the dissertation.

Signature of Student(s):

1. **Sayan Golder (32301222031)**

2. **Sweta Singh (32301222033)**

3. **Kunal Gautam (32301222091)**

Date –

CHAPTER-1

Introduction

A unique kind of technology called the **Plant Disease Detection System** is designed to identify plant diseases through automated image analysis using deep learning models. Agriculture is the backbone of the global economy, providing essential resources such as food and raw materials for an ever-growing population. Ensuring the health of crops is crucial to maintaining food security and sustainable agricultural practices. However, one of the most significant challenges in agriculture is the early detection and accurate diagnosis of plant diseases. These diseases not only reduce crop yields but also result in substantial economic losses for farmers and agribusinesses worldwide.

Traditional disease detection methods often rely on manual inspections and expert knowledge. While effective in some cases, these methods exhibit low performance and face several challenges, primarily due to the following reasons:

- **Subjectivity:** Diagnosis often depends on the expertise and experience of the inspector, leading to potential inconsistencies.
- **Time-Consuming:** Manual inspection of large fields can be labour-intensive and slow.
- **Limited Accessibility:** Access to skilled experts is not always feasible, especially in remote or underdeveloped regions.
- **Error-Prone:** Factors such as human fatigue and environmental conditions can affect the accuracy of diagnoses.

With the rapid advancements in artificial intelligence (AI) and machine learning (ML), there is an opportunity to revolutionize the way plant diseases are detected and managed. Automated systems powered by deep learning algorithms can provide fast, reliable, and objective solutions, overcoming the limitations of traditional methods.

This project, **"Plant Disease Detection Using Machine Learning,"** aims to address these challenges by developing an AI-powered system capable of identifying and classifying plant diseases with high precision. Leveraging convolutional neural networks (CNNs) and a curated dataset of plant images, the system is designed to detect diseases at an early stage. The proposed solution ensures faster and more accurate diagnosis, empowering farmers to take timely action and minimize losses.

Moreover, by reducing reliance on excessive pesticide use, this system promotes eco-friendly farming practices. Through this project, we aim to bridge the gap between modern technology and traditional agriculture, contributing to sustainable development and food security on a global scale.

Furthermore, traditional methods struggle to scale with the increasing demand for agricultural output. As global food demand rises, the ability to monitor vast agricultural lands efficiently becomes more critical. Factors such as climate change and the emergence of new plant pathogens add further complexity to disease management. Manual inspections are often insufficient to handle these challenges, emphasizing the need for innovative technological solutions.

The integration of machine learning and computer vision into agriculture presents a transformative opportunity. Advanced algorithms can process large volumes of image data, identifying subtle patterns that are often undetectable to the human eye. These systems are not only scalable but also adaptable, enabling real-time disease monitoring and proactive decision-making.

By equipping farmers with accessible and user-friendly tools, this project aims to democratize advanced technology, making it available even in resource-constrained settings. The implementation of such systems can lead to higher crop yields, reduced losses, and a significant decrease in environmental impact, ensuring a more sustainable future for agriculture.

CHAPTER- 2

PROJECT CATEGORY:

The Plant Disease Detection System belongs to the AI-Based Systems category, specifically focusing on applications in smart agriculture and precision farming. This system harnesses the power of artificial intelligence (AI) and machine learning (ML) to revolutionize the way plant diseases are detected, classified, and managed.

AI-BASED SYSTEM OVERVIEW:

The **Plant Disease Detection System** is an example of an AI-based application tailored specifically for agriculture. It integrates technologies such as machine learning, deep learning, and computer vision to address critical challenges in crop health management. The system processes images of plants, extracts features, and diagnoses diseases, all while reducing the reliance on traditional, manual methods.
Key features of AI-based systems include:

1. **Automation**: Automates repetitive and time-consuming tasks, such as manual inspection of crops, enabling farmers to focus on strategic decision-making.
2. **Scalability**: Handles large-scale operations, making it suitable for monitoring extensive agricultural lands.
3. **Adaptability**: Continuously improves over time by learning from new data, ensuring that the system remains effective in changing conditions.
4. **Accuracy and Precision**: Identifies subtle patterns in data that are often undetectable to humans, ensuring accurate disease diagnosis and intervention.
5. **Real-Time Insights**: Delivers results almost instantaneously, enabling timely decision-making, especially in critical scenarios like disease outbreaks.

   In agriculture, AI-based systems are revolutionizing traditional practices by introducing automation, enhancing productivity, and enabling precision farming. These systems leverage deep learning techniques such as **Convolutional Neural Networks (CNNs)** and frameworks like **TensorFlow** to process image data, recognize diseases, and provide targeted solutions.

FUNCTIONALITIES PROVIDED:

The **Plant Disease Detection System** offers the following key functionalities:

1. **Image-Based Detection**
   - Users can upload high-quality images of plant leaves through the web interface or WhatsApp.
   - The system processes these images using pre-trained deep learning models to identify diseases.
   - It highlights the diseased areas and provides a classification of the specific disease.
2. **Accurate Predictions**
   - Leveraging CNN-based models like CNN, the system ensures precise and reliable detection.
   - It eliminates human biases and inconsistencies, providing consistent results.
3. **Real-Time Feedback**
   - The system delivers instant disease predictions and insights to users.
   - Results are accessible through web platforms or via WhatsApp integration, enabling immediate corrective actions.
4. **User-Friendly Interface**
   - Designed to be intuitive and accessible for farmers and agricultural professionals, regardless of their technical expertise.
   - Simple navigation for uploading images and accessing reports.
5. **Record Management**
   - Maintains a history of previously diagnosed diseases and their associated data.
   - Users can access these records for future reference and better crop management.
6. **Multi-Platform Accessibility**
   - Offers flexibility to access the system via desktop browsers or mobile devices.
   - Integration with Twilio's WhatsApp API ensures wide accessibility, especially in regions with limited internet bandwidth.
7. **Eco-Friendly Practices**
   - Promotes targeted pesticide usage by identifying specific areas affected by diseases.
   - Reduces the environmental impact of farming by minimizing the overuse of chemicals.
8. **Scalability and Adaptability**
   - Capable of monitoring large-scale agricultural lands.
   - Adapts to different crop types and disease variations by learning from updated datasets.

CHAPTER- 3

TOOLS AND PLATFORMS

SOFTWARE SPECIFICATONS

A major element in building a system is the selection of compatible software since; selected software in the market is experiencing a geometric progression and should be acceptable by the firm and one user as well as it should be feasible for the system.

Front-end tool: VS Text Editor

Back-end tool: Python

Operating system: Windows 10, 11

Client Side: Hyper Text Markup

Language Server Side: Flask

The purpose of having a software requirement specification document is to have understandable, unambiguous and complete specification documentation for the project.

HARDWARE SPECIFICATIONS

The section of hardware configuration is an important task related to the software development. Insufficient Random Access Memory may affect adversely on the speed and efficiency of the entire system. The process should be powerful to handle the entire operations. The hard disk should have sufficient capacity to store the file and applications.

Processor: Any processor

System Memory: 128Mb Minimum (for development)

RAM: 4 GB (Minimum)

Hard disk: 20 GB (Minimum)

CHAPTER- 4

SYSTEM DESIGN:

The flow chart below shows the overall model of the proposed **Plant Disease Detection System** and the steps are as follows:

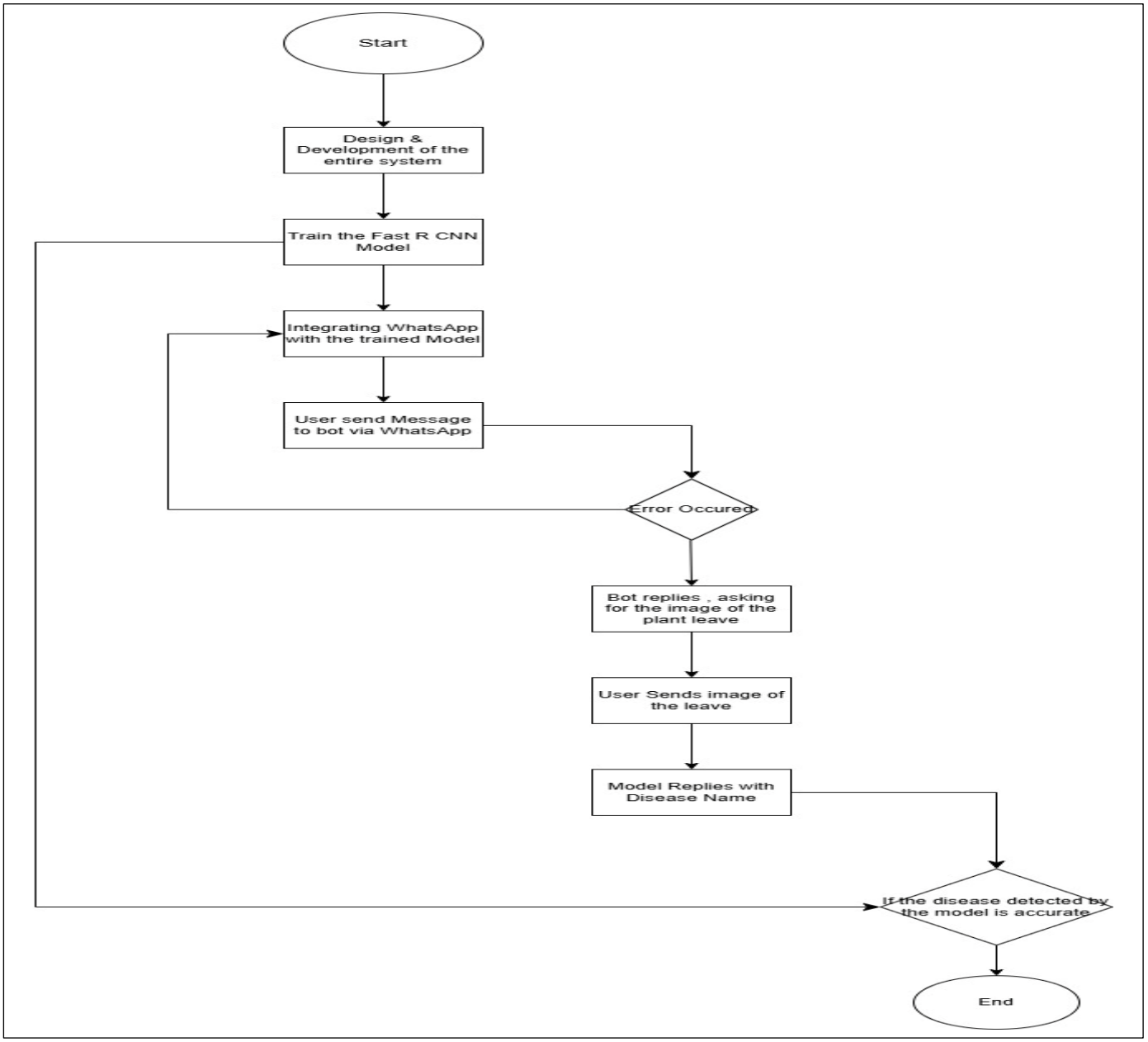
1. **User Image Upload**
   - The user accesses the web page or WhatsApp interface to upload an image of a plant leaf that needs to be analysed for potential diseases.
2. **Image Preprocessing**
   - Once the image is uploaded, the system performs necessary preprocessing tasks such as resizing, normalization, and augmentation (rotation, flipping, etc.) to ensure the image is ready for analysis.
3. **Image Analysis**
   - The processed image is sent to the **Deep Learning Model** (using TensorFlow and Keras), where the **Convolutional Neural Network (CNN)** analyses the image to detect any disease.
   - The model uses its trained parameters to identify patterns, such as discoloration, spots, or irregularities, and classifies them as specific diseases.
4. **Disease Diagnosis**
   - After analysing the image, the system generates a diagnosis, identifying the disease and providing its severity level.
   - The result is then interpreted by the backend system, and relevant treatment suggestions are prepared.
5. **Result Feedback**
   - The disease diagnosis, along with treatment recommendations, is displayed to the user through the web interface.
   - If the user has interacted via WhatsApp, the diagnosis and treatment suggestions are sent back through the chat.
6. **Reporting**
   - A report detailing the disease diagnosis, recommended treatments, and other relevant information is generated.
   - The report is either displayed on the web page or sent directly to the user via WhatsApp.
7. **Scalability and Adaptability**
   - The system can handle a large number of users, process multiple

images simultaneously, and can be retrained with new datasets to detect more diseases as required.

FLOW CHART DIAGRAM:

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

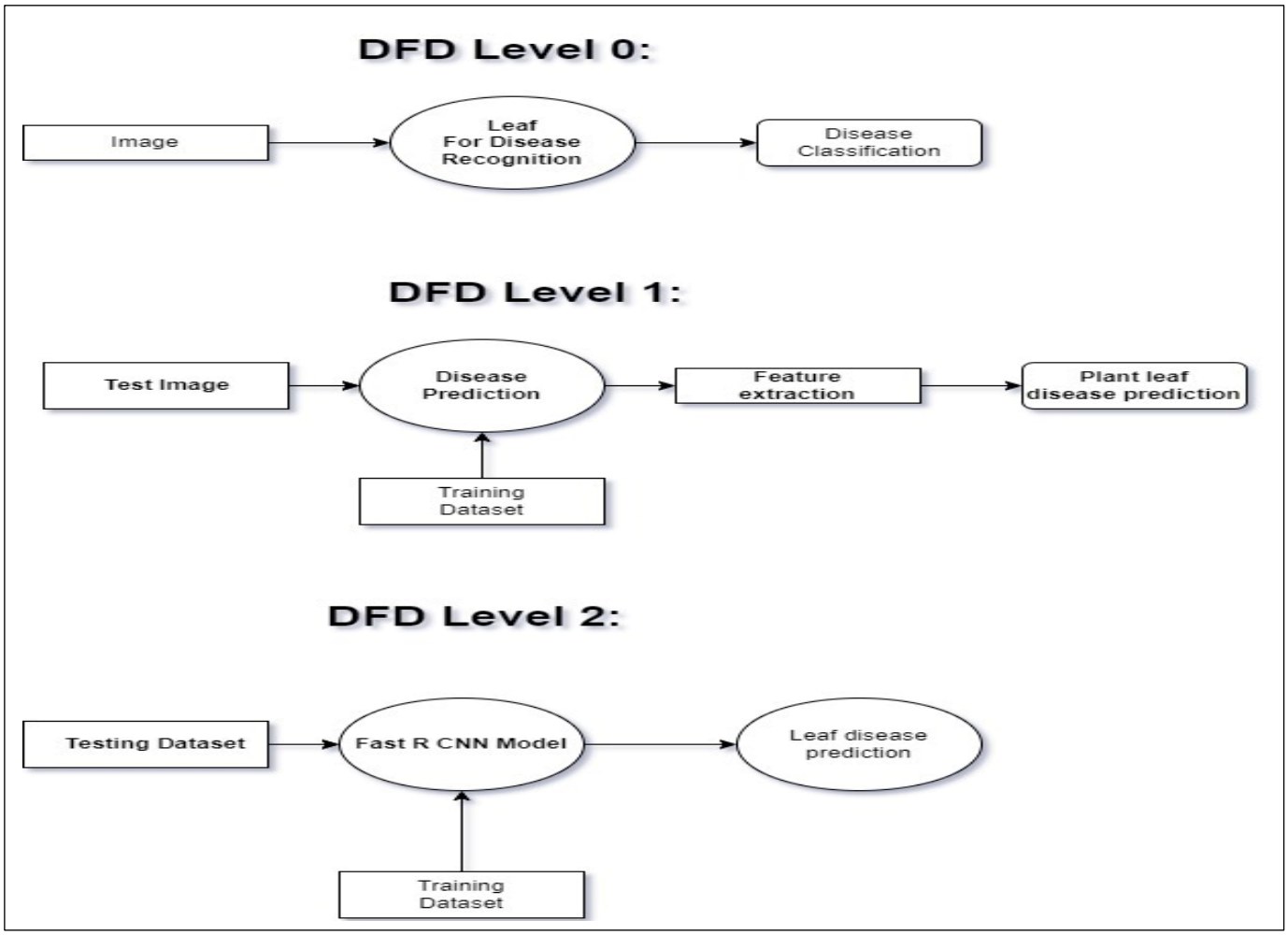


FLOW CHART OF PROPOSED SYSTEM

DFD DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation of the flow of data within a system. It visualizes the system's processes, data stores, data flows, and external entities, helping to understand the movement of information and the interactions between system components. DFDs are useful for analysing, designing, and documenting processes in a system.

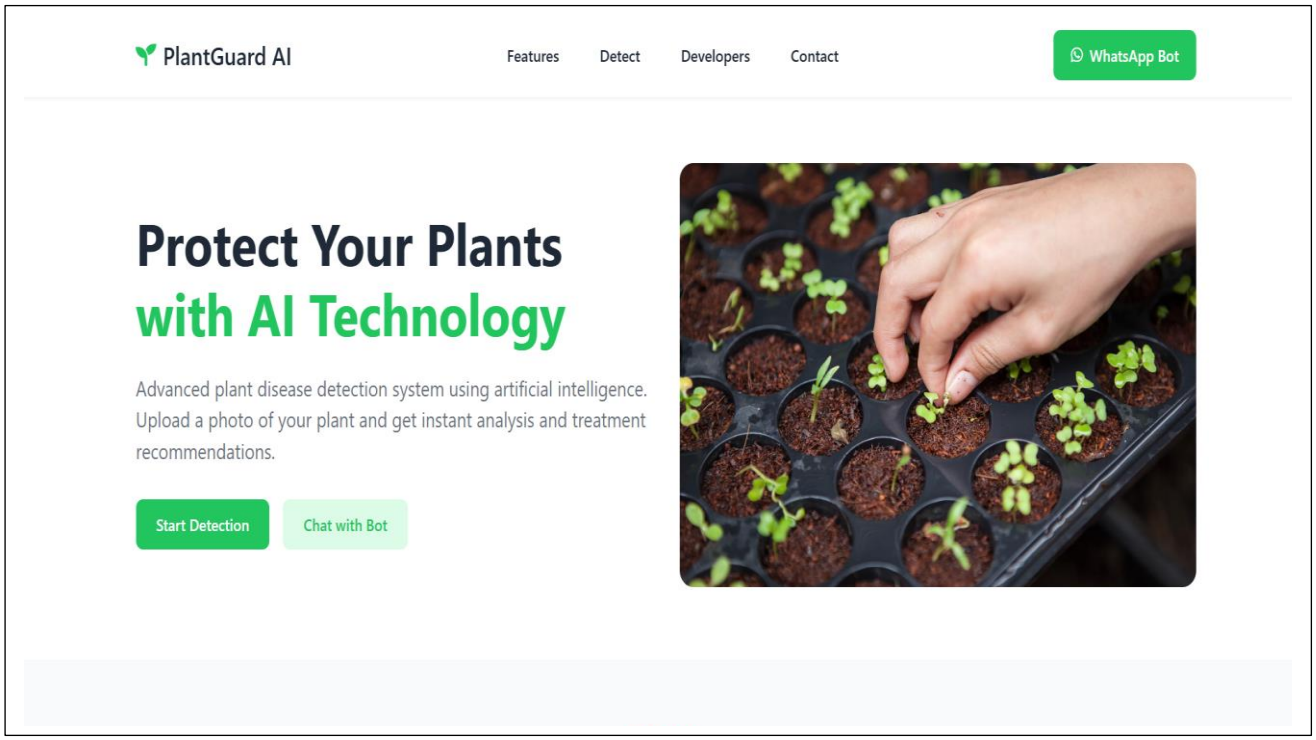
DFD Stages 0, 1, and 2 represent different levels of detail in the system's design, with each stage breaking down the system into more granular components.

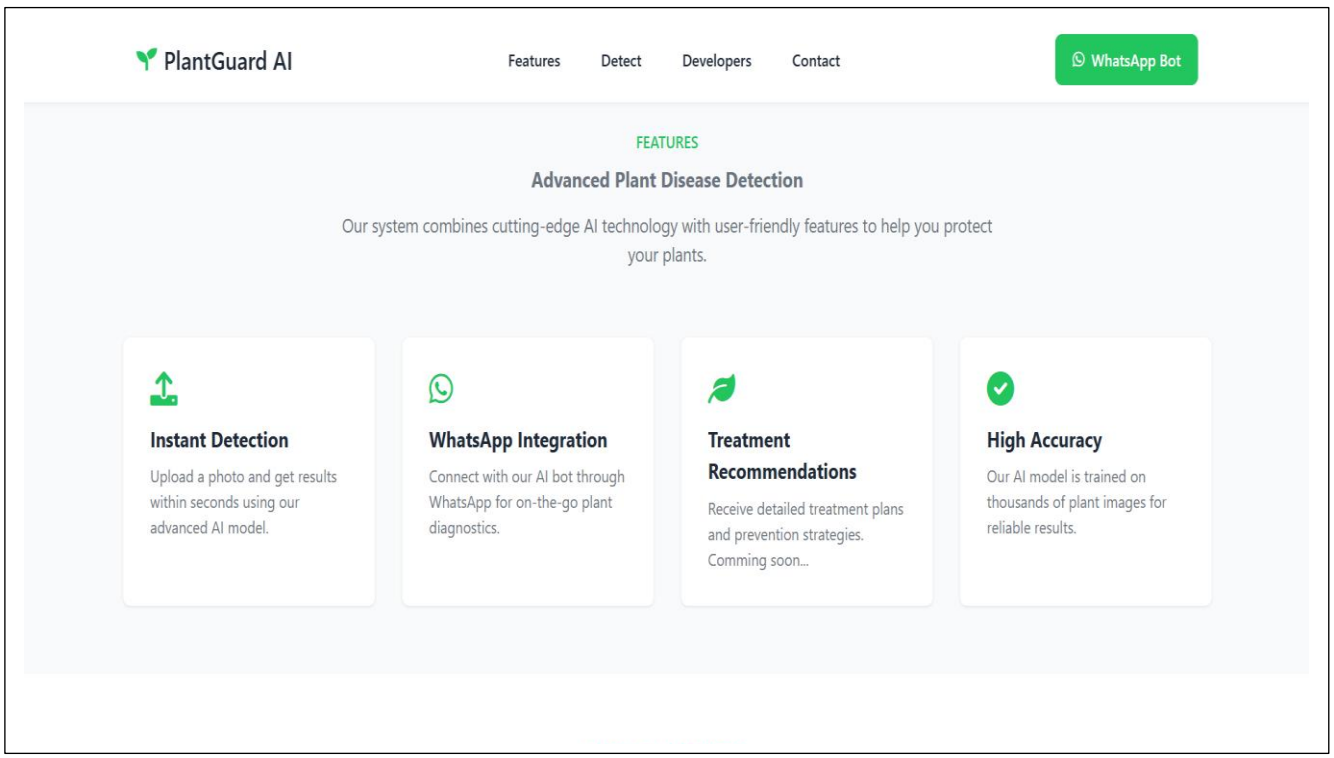


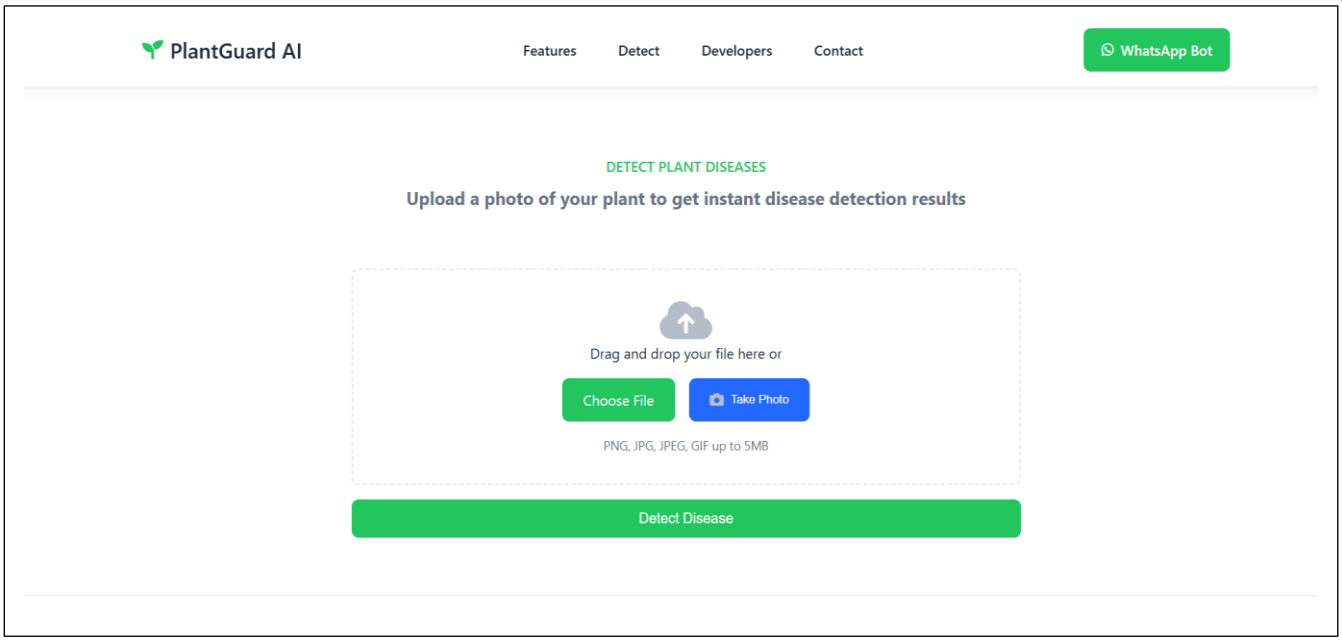
DATA FLOW DIAGRAM OF PROPOSED SYSTEM

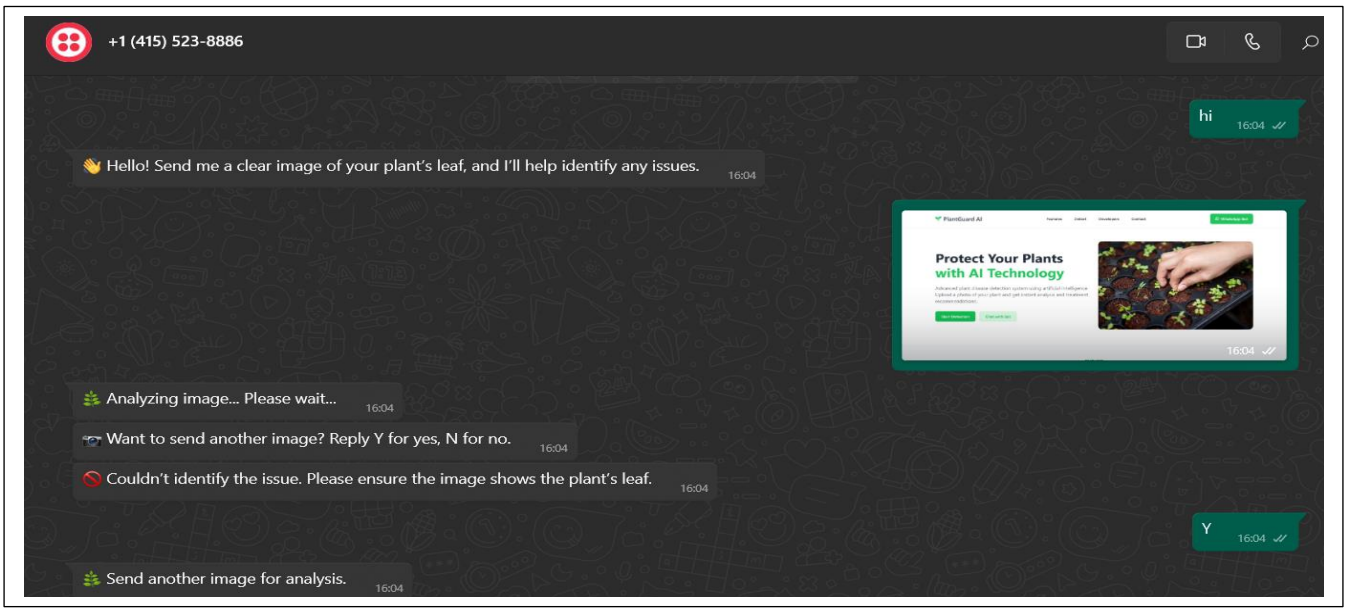
CHAPTER- 5

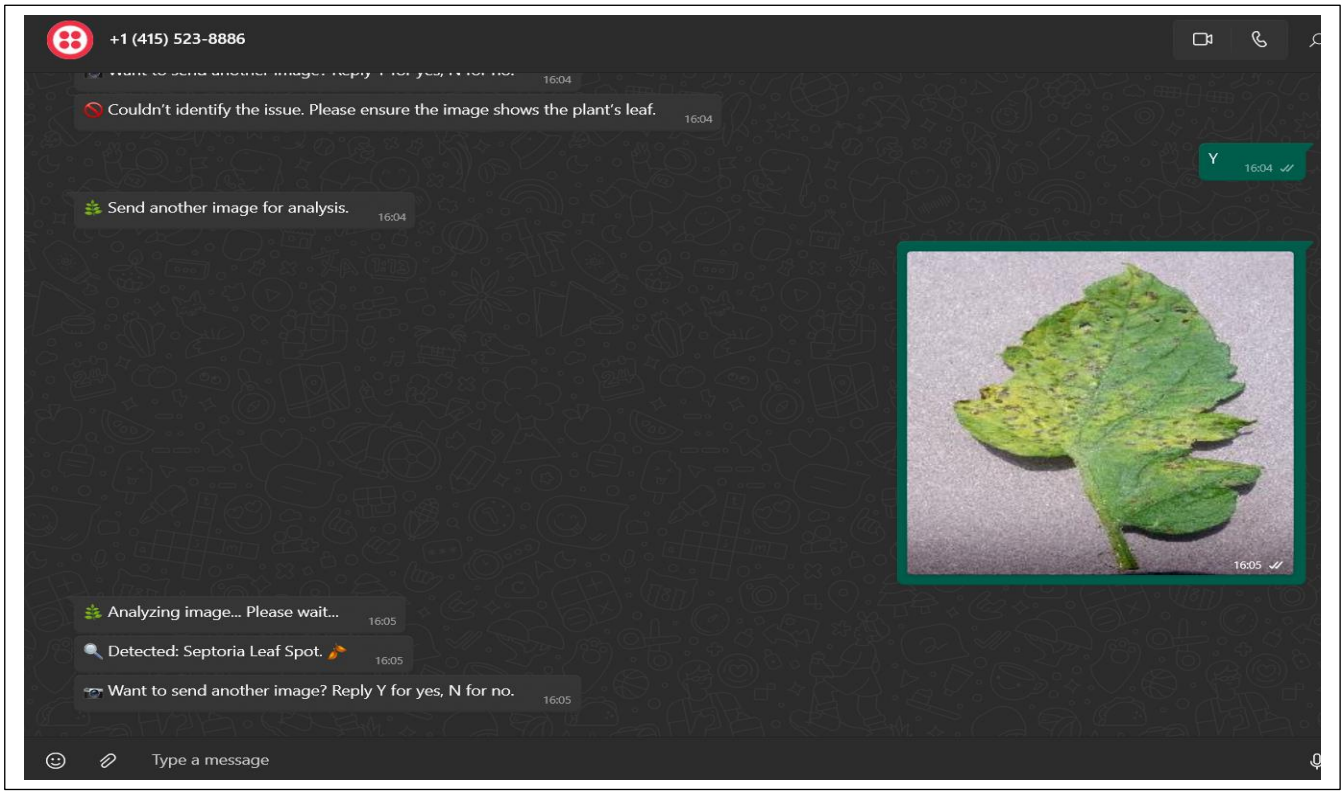
USER INTERFACE DESIGN



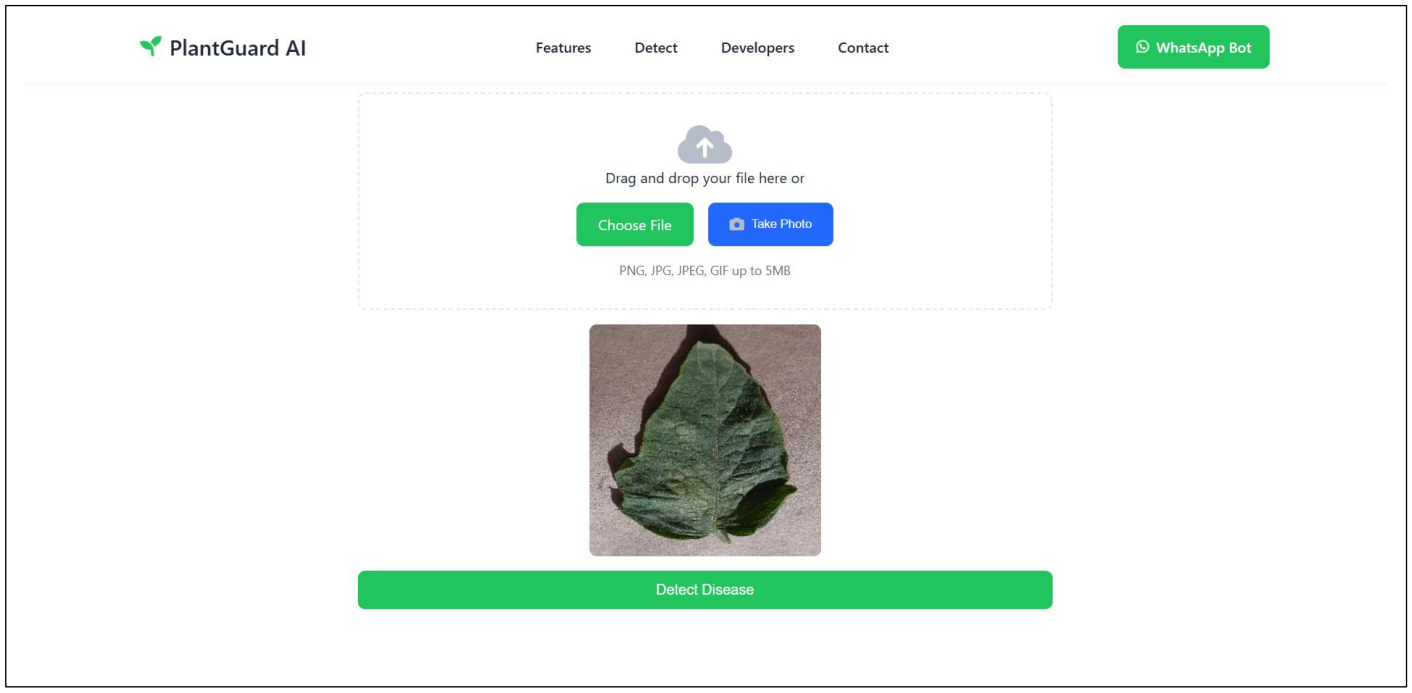


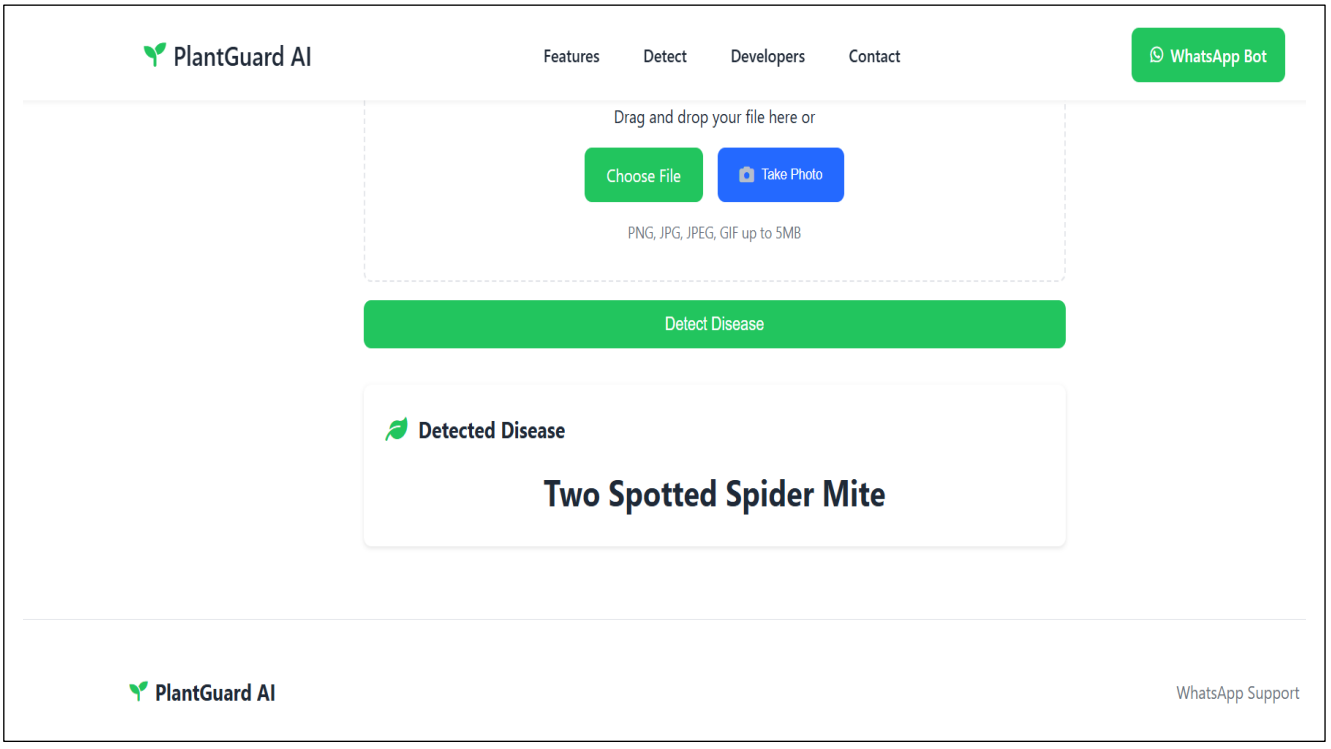

PlantGuard AI
Features
Detect
Developers
Contact
WhatsApp Bot
DETECT PLANT DISEASES
Upload a photo of your plant to get instant disease detection results
Drag and drop your file here or
Choose File
Take Photo
PNG, JPG, JPEG, GIF up to 5MB
Detect Disease


+1 (415) 523-8886
hi
16:04
Hello! Send me a clear image of your plant's leaf, and I'll help identify any issues.
16:04
Protect Your Plants
with AI Technology
16:04
Analyzing image... Please wait...
16:04
Want to send another image? Reply Y for yes, N for no.
16:04
Couldn't identify the issue. Please ensure the image shows the plant's leaf.
16:04
Y
16:04
Send another image for analysis.
16:04


+1 (415) 523-8886
16:04
Couldn't identify the issue. Please ensure the image shows the plant's leaf.
16:04
Y
16:04
Send another image for analysis.
16:04
16:05
Analyzing image... Please wait...
16:05
Detected: Septoria Leaf Spot.
16:05
Want to send another image? Reply Y for yes, N for no.
16:05
Type a message

DISEASE DETECTION RESULT





CHAPTER- 6

CODING SECTION

Front-End Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PlantGuard AI - Plant Disease Detection</title>
    <link rel="stylesheet" href="static\styles.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css">
    <script>
        // Scroll to the results section if there's a prediction
        window.onload = function () {
            const predictionSection = document.getElementById('prediction-
section');
            if (predictionSection && predictionSection.innerText.trim()) {
                predictionSection.scrollIntoView({ behavior: 'smooth' });
            }
        };
    </script>
</head>
<body>
    <header>
        <nav>
            <div class="logo">
                <i class="fas fa-seedling"></i>
                <span>PlantGuard AI</span>
            </div>
            <ul class="nav-links">
                <li><a href="#features">Features</a></li>
                <li><a href="#upload">Detect</a></li>
                <li><a href="#about">About</a></li>
                <li><a href="#contact">Contact</a></li>
            </ul>
            <a href="https://wa.me/+1(415)523-8886" class="whatsapp-btn"
target="_blank" rel="noopener noreferrer">
                <i class="fab fa-whatsapp"></i> WhatsApp Bot
            </a>
        </nav>
    </header>

    <section class="hero">
        <div class="hero-content">
            <h1>Protect Your Plants <span>with AI Technology</span></h1>
```

```
            <p>Advanced plant disease detection system using artificial
intelligence. Upload a photo of your plant and get instant analysis and
treatment recommendations.</p>
            <div class="hero-buttons">
                <a href="#upload" class="primary-btn">Start Detection</a>
                <a href="https://wa.me/+1(415)523-8886" class="secondary-btn"
target="_blank" rel="noopener noreferrer">Chat with Bot</a>
            </div>
        </div>
        <div class="hero-image">
            <img src="https://images.unsplash.com/photo-1530836369250-
ef72a3f5cda8?ixlib=rb-1.2.1&auto=format&fit=crop&w=1950&q=80" alt="Healthy
plants in a greenhouse">
        </div>
    </section>

    <section id="features" class="features">
        <div class="section-header">
            <h2>Features</h2>
            <p>Advanced Plant Disease Detection</p>
            <span>Our system combines cutting-edge AI technology with user-
friendly features to help you protect your plants.</span>
        </div>
        <div class="feature-grid">
            <div class="feature-card">
                <i class="fas fa-upload"></i>
                <h3>Instant Detection</h3>
                <p>Upload a photo and get results within seconds using our
advanced AI model.</p>
            </div>
            <div class="feature-card">
                <i class="fab fa-whatsapp"></i>
                <h3>WhatsApp Integration</h3>
                <p>Connect with our AI bot through WhatsApp for on-the-go plant
diagnostics.</p>
            </div>
            <div class="feature-card">
                <i class="fas fa-leaf"></i>
                <h3>Treatment Recommendations</h3>
                <p>Receive detailed treatment plans and prevention
strategies.</p>
            </div>
            <div class="feature-card">
                <i class="fas fa-check-circle"></i>
                <h3>High Accuracy</h3>
                <p>Our AI model is trained on thousands of plant images for
reliable results.</p>
            </div>
        </div>
    </section>

    <section id="upload" class="upload-section">
        <div class="section-header">
            <h2>Detect Plant Diseases</h2>
```

```
            <p>Upload a photo of your plant to get instant disease detection
results</p>
        </div>
        <div class="upload-container">
            <form action="/upload" method="POST" enctype="multipart/form-data"
id="upload-form">
                <div class="upload-box">
                    <i class="fas fa-cloud-upload-alt"></i>
                    <p>Drag and drop your file here or</p>
                    <label for="file-upload" class="file-label">Choose
File</label>
                    <input type="file" id="file-upload" name="file_"
accept="image/*" hidden>
                    <p class="file-info">PNG, JPG, GIF up to 10MB</p>
                </div>
                <div id="preview" class="preview-container" hidden>
                    <img id="preview-image" src="" alt="Preview">
                </div>
                <button type="submit" class="submit-btn">Detect
Disease</button>
            </form>
        </div>
        {% if prediction %}
        <div class="result-container" id="prediction-section">
            <h3 class="result-title">Detected Result:</h3>
            <p class="result-text">{{ prediction }}</p>
        </div>
        {% endif %}

    </section>

    <footer>
        <div class="footer-content">
            <div class="footer-logo">
                <i class="fas fa-seedling"></i>
                <span>PlantGuard AI</span>
            </div>
            <div class="footer-links">
                <a href="https://wa.me/+1(415)523-8886" target="_blank"
rel="noopener noreferrer">WhatsApp Support</a>
                <a href="#">Privacy Policy</a>
                <a href="#">Terms of Service</a>
            </div>
        </div>
        <div class="footer-bottom">
            <p>&copy; <span id="year"></span> PlantGuard AI. All rights
reserved.</p>
        </div>
    </footer>

    <script>
        document.getElementById('year').textContent = new Date().getFullYear();

        const fileUpload = document.getElementById('file-upload');
```

```
        const previewContainer = document.getElementById('preview');
        const previewImage = document.getElementById('preview-image');

        fileUpload.addEventListener('change', function(e) {
            const file = e.target.files[0];
            if (file) {
                const reader = new FileReader();
                reader.onload = function(e) {
                    previewImage.src = e.target.result;
                    previewContainer.hidden = false;
                }
                reader.readAsDataURL(file);
            }
        });
    </script>
</body>
</html>
```

Back-End Code

```
from flask import Flask, render_template, request, jsonify
import os
import tensorflow as tf
import numpy as np
import cv2
import json

app = Flask(__name__)

# Configurations
UPLOAD_FOLDER = os.path.join(os.getcwd(),'uploads')
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Ensure the upload folder exists
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

def predict_image(image_path):

    model = tf.keras.models.load_model('plant_disease_model.h5')
    with open('class_indices.json', 'r') as f:
        class_indices = json.load(f)
    print("Loaded Class Indices:", class_indices)
    class_indices = {int(v): k for k, v in class_indices.items()}
    img = cv2.imread(image_path)
    img = cv2.resize(img, (150, 150))
    img = img / 255.0
    img = np.expand_dims(img, axis=0)
    predictions = model.predict(img)
    predicted_class_index = np.argmax(predictions)
```

```
    predicted_class = class_indices[predicted_class_index]

    return predicted_class
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

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

@app.route('/upload', methods=['POST', 'GET'])
def upload_file():
    print(request.files)
    if 'file_' not in request.files:
        return jsonify({'error': 'No file uploaded'}), 400
    file = request.files['file_']
    if file and allowed_file(file.filename):
        print("hi")
        filename = file.filename
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)
        result = predict_image(filepath)
        # return jsonify({'message': 'File uploaded successfully', 'file_url':
f'/static/uploads/{filename}'})
        return render_template('index.html', prediction=result)
    else:
        return jsonify({'error': 'File type not allowed'}), 400

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

Twilio- Code

```
import datetime
import logging
from flask import Flask,request
import requests
from twilio.twiml.messaging_response import MessagingResponse
from twilio.rest import Client
import os
import json
from dotenv import load_dotenv
import requests
url = "https://web-production-0cb7.up.railway.app/predict/"

def detect(file):
    file = {"file": open(file,'rb')}
    response = requests.post(url, files=file)
    result = json.loads(response.text)['class_name']
    return result
load_dotenv()
app = Flask(_name_)
account_sid = os.getenv('TWILIO_ACCOUNT_SID')
auth_token = os.getenv('TWILIO_AUTH_TOKEN')
client = Client(account_sid, auth_token)

UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
session = {}
def save_image_from_url(media_url, from_number):
    static_dir = app.config['UPLOAD_FOLDER']
    image_name = f"whatsapp_{from_number.replace(':',
'')}{datetime.datetime.now(datetime.timezone.utc).strftime('%Y%m%d%H%M%S')}.pdf
"
    image_path = os.path.join(static_dir, image_name)

    try:
        message_sid = media_url.split('/')[-3]
        media_sid = media_url.split('/')[-1].split('.')[0]

        media = client.messages(message_sid).media(media_sid).fetch()
        media_url = f"https://api.twilio.com{media.uri.replace('.json', '')}"

        response = requests.get(media_url, auth=(account_sid, auth_token))
        with open(image_path, 'wb') as f:
            f.write(response.content)

        logging.info(f"Image saved successfully: {image_path}")
        return image_path
    except Exception as e:
        logging.error(f"Error downloading image: {e}")
        return None

@app.route('/', methods=['GET', 'POST'])
```

```
def chatbot():
    media_url = request.form.get('MediaUrl0')
    user_message = request.form.get('Body').lower()

    bot_message = MessagingResponse()

    if "state" not in session:
        session["state"] = "initial"

    if user_message in ['hi', 'hello'] and session["state"] == "initial":
        bot_message.message('👋 Hello! Send me a clear image of your plant’s
leaf, and I’ll help identify any issues.')
        session["state"] = "processing"

    elif media_url and session["state"] == "processing":
        bot_message.message('🌿 Analyzing image... Please wait...')

        image_path = save_image_from_url(media_url, request.form.get('From'))
        if image_path:
            dis = detect(image_path)
            if dis == 'Unknown':
                bot_message.message('🚫 Couldn’t identify the issue. Please
ensure the image shows the plant’s leaf.')
            else:
                bot_message.message(f'🔍 Detected: {dis}. 🌱')
        else:
            bot_message.message('🚫 Couldn’t download the image. Try again
later.')

        bot_message.message('📸 Want to send another image? Reply Y for yes, N
for no.')

    elif user_message == 'y':
        bot_message.message('🌿 Send another image for analysis.')
        session["state"] = "processing"

    elif user_message == 'n':
        bot_message.message('😊 Thanks for using the bot! Say "hi" anytime for
help.')
        session["state"] = "initial"
        session.clear()

    else:
        bot_message.message('😵 I didn’t understand. Reply "hi" to start
over.')
        session["state"] = "initial"

    return str(bot_message)
```

API -Code

```
from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.responses import JSONResponse,RedirectResponse
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np
import logging

logging.basicConfig(level=logging.INFO)

app = FastAPI(title="Plant Disease Analysis API")

# Load model and labels at startup
MODEL_PATH = "keras_model1.h5"
LABELS_PATH = "labels.txt"

model = load_model(MODEL_PATH, compile=False)
class_names = open(LABELS_PATH, "r").readlines()


@app.get("/",include_in_schema=False)
def index():
    return RedirectResponse("/docs", status_code=308)

@app.post("/predict/")
def predict_image(file: UploadFile = File(...)):
    try:
        # Open and preprocess the image
        print(file.content_type)
        image = Image.open(file.file).convert("RGB")
        size = (224, 224)
        image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)
        image_array = np.asarray(image)
        normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1
        data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
        data[0] = normalized_image_array

        # Perform prediction
        prediction = model.predict(data)
        index = np.argmax(prediction)
        class_name = class_names[index].strip()  # Strip newline characters
        confidence_score = float(prediction[0][index])
        return JSONResponse(
            content={
                "class_name": class_name,
                "confidence_score": confidence_score
            }
        )

    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

CHAPTER- 7

COMPONENTS AND MODULES

The **Plant Disease Detection System** relies on a set of well-defined components and modules that form the foundation of its architecture. These components ensure the system operates seamlessly, from data acquisition to disease diagnosis and feedback. Below are detailed descriptions of the key elements:

A. Dataset Description

A high-quality dataset is a fundamental requirement for building any AI-based system. For the **Plant Disease Detection System**, the dataset serves as the backbone for training and testing the deep learning model. Here's a detailed breakdown of the dataset used:

1. **Source of Dataset**:
   The dataset was sourced from **Kaggle**, an open-source platform for datasets, often used for machine learning and data science projects. Kaggle is known for hosting large, well-labelled datasets, making it an ideal choice for this project.

2. **Initial Dataset Size**:
   The original dataset contained a total of **60,000 images** of plant leaves. These images represented various states of plant health, including:

   - Healthy leaves.
   - Leaves affected by common diseases, such as rust, blight, mildew, and spots.

3. **Dataset Refinement Process**:
   After acquiring the dataset, a rigorous refinement process was conducted to ensure the quality and relevance of the data. The refinement steps included:

   - **Duplicate Removal**: Eliminating duplicate images to avoid redundancy in the dataset.
   - **Quality Check**: Filtering out corrupted, blurred, or poorly labeled images.
   - **Standardization**: Ensuring all images had consistent resolutions and formats suitable for model training.
   - **Label Verification**: Cross-checking that each image was correctly labeled

according to the disease category it represented.
Following this process, the dataset was refined to a manageable and high-quality subset of **10,000 images**, representing the most accurate and useful samples for the project.

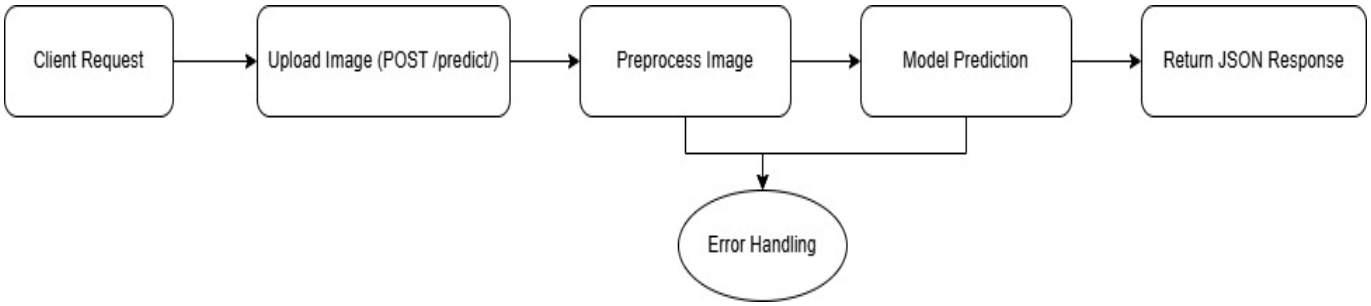
4. **Categories and Coverage**:
   The refined dataset is categorized into various disease types and healthy leaves. Each category is well-represented, ensuring a balanced and comprehensive training dataset. Examples include:
   - **Rust**: Yellowish or reddish spots on leaves.
   - **Blight**: Brownish-black necrotic lesions.
   - **Mildew**: White or gray fungal growth on leaf surfaces.
   - **Spots**: Circular or irregular discolorations caused by pathogens.
5. **Dataset Visual Example**:
   - *Include an image named "dataset_example.jpg" here* showcasing a few sample images, each labelled with its respective disease category or "Healthy." This visual provides a clear idea of the dataset diversity and quality.

B. API for Disease Detection

The API, built using Fast API, handles the prediction process by taking uploaded images as input and returning the detected disease and confidence score.

- Endpoints:
  - /predict/: Accepts images and returns predictions.
- Workflow:
  - Preprocesses uploaded images (resize, normalize).
  - Sends them to a trained deep learning model for analysis.
  - Responds with disease class and confidence score.

Diagram Reference:



C. **Twilio Integration**

The **Twilio Integration** enables users to interact with the system via WhatsApp, making it accessible and user-friendly. This module connects the WhatsApp messaging platform with the Flask backend and the disease detection API.

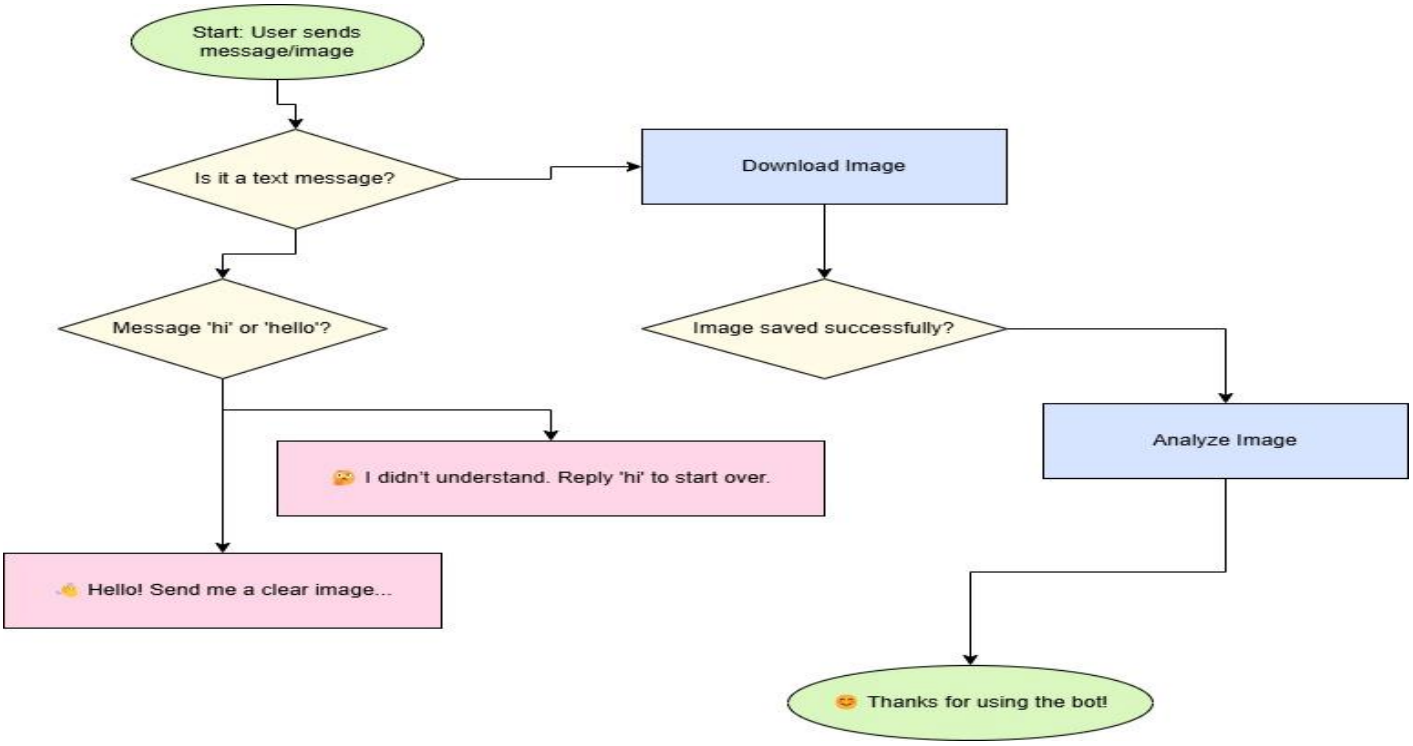
1. **Overview**:
   Twilio's API allows the system to send and receive messages through WhatsApp, making it easy for users to upload plant images and receive results directly on their mobile devices.

2. **Key Features**:
   - **Image Submission**: Users can send plant images as WhatsApp messages.
   - **Image Handling**: The system downloads images from the message, processes them, and prepares them for prediction.
   - **Disease Detection**: The images are sent to the disease detection API, which predicts the disease.
   - **Response Delivery**: Results are sent back to the user in an easy-to-understand format, including the detected disease and recommendations.

3. **Workflow**:
   - A user sends an image through WhatsApp.
   - The Flask server receives the message and extracts the image URL.
   - The image is downloaded using Twilio’s API and saved locally.
   - The image is passed to the prediction API for analysis.

- The result is sent back to the user as a WhatsApp message.

Example Workflow Image:



These detailed components and modules form the backbone of the **Plant Disease Detection System**, enabling it to deliver accurate and efficient results. The combination of a high-quality dataset and an advanced deep learning architecture ensures the system meets the needs of modern agriculture effectively.

CHAPTER- 8

WORKING PRINCPLE:

The **Plant Disease Detection System** operates on the principle of utilizing **deep learning and image processing** to analyze plant images and identify diseases. The system uses advanced machine learning models like **CNN**, trained on a curated dataset of plant images, to recognize patterns and classify diseases accurately. The workflow of the system can be described as follows:

1. **Image Input**:
   The user uploads an image of a plant leaf via a web interface or sends it using WhatsApp (via Twilio integration).
2. **Image Preprocessing**:
   The uploaded image undergoes preprocessing using resizing, normalization, and augmentation techniques to ensure compatibility with the deep learning model. The image is resized to a standard size (e.g., **224x224 pixels**) and normalized for better learning by the AI model.
3. **Prediction Model**:
   The preprocessed image is sent to the **Deep Learning Model** (CNN) for disease detection. This CNN-based architecture analyzes the image patterns and features to identify if the plant has a disease or is healthy.
4. **Model Inference**:
   The trained model processes the image, extracts patterns, and generates predictions with confidence scores for possible diseases.
5. **Result Generation**:
   The system interprets the model's prediction and returns the disease name and confidence score. This information is sent to the user via a web interface or WhatsApp.

Advantages of Plant Disease Detection System

The system offers numerous advantages for modern agricultural practices:

1. **Early Disease Detection**:
   The system can identify diseases at an early stage, reducing the spread of infection and potential crop losses.
2. **Time-Saving**:
   Automated disease detection significantly reduces the time required compared to manual inspection.
3. **Cost-Efficient**:
   By enabling early detection, it minimizes the need for widespread pesticide use, thus reducing costs.
4. **User-Friendly Interface**:
   Designed to be simple and intuitive, the system allows farmers, even with minimal technical expertise, to use it effectively.
5. **Scalability**:
   The system can process a large number of plant leaf images .

6. **Environmentally Friendly**:
   By targeting specific diseases and minimizing the need for overuse of chemicals, the system supports eco-friendly farming practices.

Limitations

Despite its benefits, the **Plant Disease Detection System** has certain limitations:

1. **Dependence on Dataset Quality**:
   The system’s accuracy depends on the quality of the images in the training dataset. Any bias, noise, or missing data in the dataset can affect prediction results.
2. **Environmental Variability**:
   Factors such as lighting conditions, background interference, or poor image quality can impact the system’s ability to analyze images accurately.
3. **New or Rare Diseases**:
   The system may struggle to recognize diseases that are not part of its training data or are rare.
4. **Internet Dependency for Communication**:
   Integration with WhatsApp using Twilio requires a stable internet connection. Connectivity issues could limit the system's usability.
5. **Complexity of Disease Patterns**:
   Some diseases may present similar visual symptoms, leading to misclassification or uncertainty in prediction results.
6. **Hardware Requirements**:
   Effective deployment depends on access to hardware with sufficient memory, processing power, and storage capabilities.

CHAPTER- 9

IMPLEMENTATION AND TESTING

OBJECTIVES OF TESTING:

The goal of the testing process is to identify and rectify as many defects as possible to enhance the reliability of the **Plant Disease Detection System**. While exhaustive testing is impractical, a diverse range of tests will be conducted to achieve this objective. The system's user interface has been designed for simplicity and intuitive use, enabling users to interact seamlessly with the disease detection features. Testing ensures that the system's usability and performance remain consistent across different platforms and environments.

Process Overview

The testing process follows these steps:

1. **Test Planning**:
   - Identify specific tests to be conducted for each module.
2. **Validation of Test Cases**:
   - Ensure test cases adequately verify the correct functionality of each module.
3. **Expected Outcomes**:
   - Define the anticipated results for each test.
4. **Setup and Execution**:
   - Record test setup, input data, and expected outcomes.
   - Perform tests on the system modules.
5. **Documentation**:
   - Maintain detailed documentation of test data, test cases, and test configurations.
   - Compile results in a **Unit/System Test Report**.
6. **Defect Handling**:
   - If a test fails, a bug report is generated, detailing the issue, possible causes, and conditions leading to the defect.
7. **Integration and System Testing**:
   - Modules passing unit tests are qualified for integration testing and further validation.

TEST CASES: Test cases are written descriptions of specific inputs, actions, or events, along with their expected outcomes. These are designed to evaluate the functionality of individual application features. Creating test cases aids in uncovering potential design or requirement issues and ensuring system reliability.

Testing Steps

1. Unit Testing:
   - Focuses on testing the smallest units (modules) of the system independently.
   - Each module is tested against its expected output to verify correct functionality.
2. Integration Testing:
   - Tests interactions between modules to ensure data flows correctly across interfaces.

- Detects issues arising from the integration of unit-tested modules into the overall system.

3. Validation Testing:
   - Conducted on the fully integrated system to ensure it meets user requirements and specifications.
   - Validation outcomes can result in:
     - Acceptance of functionality.
     - Identification of deficiencies.

TEST CASE REPORT:-

Tested By	Sweta Singh, Sayan Golder & Kunal Gautam
Test Type	Unit Testing

Test Case Detail:

Test Case Number	**1**
Test Case Name	Uploading Plant Image via Web Interface
Test Case Description	The user uploads an image, and the system predicts the plant disease along with confidence scores.
Item to be Tested	1. Image upload functionality via the web interface. 2. Correct disease prediction with confidence score display.

Test Case Number	2
Test Case Name	WhatsApp Image Analysis
Test Case Description	The user sends a plant image via WhatsApp, and the system provides disease diagnosis through Twilio integration.
Item to be Tested	1. Image reception and download via Twilio API. 2. Prediction results sent back to the user on WhatsApp.

This structured approach to testing ensures that the **Plant Disease Detection System** is robust, reliable, and user-friendly, meeting both functional and non-functional requirements.

CHAPTER- 10

CONCLUSION

- The **Plant Disease Detection System** underwent a thorough testing process to ensure its reliability, accuracy, and usability. Through unit, integration, and validation testing, all modules were evaluated for functionality and performance, both individually and as part of the overall system. The use of test cases helped identify and rectify potential issues, enhancing the system's efficiency and user experience. This robust testing framework ensures the system is well-prepared to meet user needs, providing accurate and timely plant disease diagnoses across various platforms.

.

FUTURE SCOPE

- The **Plant Disease Detection System** holds significant potential for future advancements, including the integration of IoT devices for real-time monitoring, drone-based imaging for large-scale farms, and the addition of more plant species and diseases to the dataset. Developing a multilingual mobile app, predictive analytics for disease outbreaks, and personalized treatment plans will further enhance its usability. With continuous AI model upgrades and cloud-based reporting, the system can evolve into a comprehensive tool for precision agriculture, improving crop health management and sustainability worldwide.