PLANT DISEASE DETECTION SYSTEM

A PROJECT REPORT

Submitted by

ADHARSH RAJAN(SNG22CS010)
AHSANA FATHIMA A(SNG22CS018)
ANEETTA K J(SNG22CS038)
ARATHY JEROME(SNG22CS042)

to

The APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

In

Computer Science and Engineering

Department of Computer Science and Engineering

Sree Narayana Gurukulam College of Engineering,

Kadayiruppu, 682311

APRIL 2025

DECLARATION

We undersigned hereby declare that the project report " **PLANT DISEASE DETECTION SYSTEM** ", submitted for partial fulfilment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Assistant. Professor. Soumya A.V. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Kadayiruppu

Date: 03/04/2025

Adharsh Rajan

Ahsana Fathima A

Aneetta K J

Arathy Jerome

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
SREE NARAYANA GURUKULAM COLLEGE OF ENGINEERING,
KADAYIRUPPU, 682311
(Affiliated to APJ Abdul Kalam Technological University & Approved by A.I.C.T.E)

CERTIFICATE

This is to certified that the project report, **"PLANT DISEASE DETECTION SYSTEM"** submitted by **ADHARSH RAJAN, AHSANA FATHIMA A, ANEETTA K J, ARATHY JEROME** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bona fide record of the project work carried out by them under our guidance and supervision .This report in any form has not been submitted to any other University or Institute for any purpose.

Professor. (Dr.) SMITHA SURESH	**Assistant. Professor. SOUMYA A .V**	**Associate. Professor. SAINI JACOB SOMAN**
HEAD OF THE DEPARTMENT	**PROJECT GUIDE**	**PROJECT COORDINATOR**

Submitted for the University Evaluation on ……………………………………….

University Register No………………………………………………………………

Internal Examiner External Examiner

ACKNOWLEDGEMENT

Dedicating this project to the Almighty God whose abundant grace and mercy enabled its successful completion, we would like to express our profound gratitude to all the people who had inspired and motivated us to undertake this project.

We wish to express our sincere thanks to our Head of the Department, **Professor. (Dr.) Smitha Suresh**, for providing us the opportunity to undertake this project. We are deeply indebted to our project coordinator **Associate. Professor. Saini Jacob Soman** and project guide **Assistant. Professor. Soumya A.V** in the Department of Computer Science and Engineering for providing us with valuable advice and guidance during the course of the project.

Finally, we would like to express my gratitude to Sree Narayana Gurukulam College of Engineering for providing me with all the required facilities without which the successful completion of the project work would not have been possible.

COURSE OUTCOME AND PROGRAM OUTCOME

Course Outcome	
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools &advanced programming techniques (Cognitive Knowledge Level: Apply)
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply

Program outcomes
Engineering Graduates will be able to:
PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems
PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9. Ethics: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES (PSO’s)
PSO1: Shall enhance the employability skills by finding innovative solutions for challenges and problems in various domains of CS..
PSO2: Shall apply the acquired knowledge to develop software solutions and innovative Mobile applications for various problems

CO PO PSO MAPPING

	PO1 (Engineering Knowledge)	PO2 (Problem Analysis)	PO3(Design/Development of Solution)	PO4 (Investigation of complex problem)	PO5 (Modern tool usage)	PO6 (The Engineer and Society)	PO7 (Environment and Sustainability)	PO8 (Ethics)	PO9 (Individual and team work)	PO10 (Communication)	PO11(Management and Finance)	PO12 (Lifelong learning)	PSO1 finding innovative solution	PSO2 Software envelopment
CO1	2	2	2	2		2	2	2	2	2	2	2	2	2
CO2	2	2	2	2	2	2		2	2	2	2	2	2	2
CO3	2	3	3	3	3	2	2	2	3	3	2	2	3	3
CO4	2	2	2	2	2			2	2	2	2	2		2
CO5	3	3	3	3	3	2	2	2	2		3	3	3	3
AVERAGE	2.2	2.4	2.4	2.4	2.5	2.0	2.0	2.0	2.2	2.25	2.2	2.2	2.5	2.4

PO	Attained Point (0/1/2/3)	Justification
PO1	3	Applied mathematical models and deep learning techniques (CNNs) for plant disease classification.
PO2	2	Developed an understanding of machine learning concepts and dataset processing
PO3	3	Designed and implemented the project using React.js (frontend) and Node.js with Express.js (backend).
PO4	2	Integrated TensorFlow models for automated plant disease classification
PO5	3	Implemented RESTful APIs for real-time disease prediction and frontend-backend communication.
PO6	3	Developed a system to assist farmers by providing disease identification and treatment recommendations.
PO7	3	Conducted research on existing plant disease detection systems and improved accuracy.
PO8	3	Maintained team collaboration via weekly discussions, video conferencing, and task assignments.
PO9	3	Organized regular project documentation updates, code reviews, and testing phases.
PO10	3	Managed resources efficiently, including dataset handling and training on CPU-based systems
PO11	3	Gained knowledge of ML models, database management, and full-stack development.
PO12	3	Collaborated with cross-functional teams improvement of the plant disease classification system.
POS1	3	Optimized plant disease detection models for improved accuracy and efficiency, using techniques such as hyperparameter tuning and cross-validation.
POS2	3	Developed and deployed a real-time disease prediction system using RESTful APIs and integrated TensorFlow models, enabling practical application in agriculture.

ABSTRACT

Develops a Plant Disease Detection system using advanced image processing and CNN-based machine learning to accurately identify and classify diseases from leaf images. Aims to promote sustainable farming practices by reducing reliance on chemical treatments and improving overall crop health. Focuses on enhancing agricultural productivity and minimizing economic losses caused by plant diseases.

The system is designed to be user-friendly and easily accessible, allowing farmers to upload leaf images through a mobile app or web platform for immediate disease detection results. By integrating this technology, farmers can quickly understand the health status of their crops, which ultimately leads to improved decision-making and better resource management. The tool can detect a wide range of plant diseases across different crops, providing valuable insights into plant health and offering suggestions for effective treatment or management practices

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

Abbreviation	**Definition**
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
RNN	Recurrent Neural Network
IoT	Internet of Things
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
DNN	Deep Neural Network
GUI	Graphical User Interface
CSS	Cascading Style Sheets
HTML	Hyper Text Markup Language
API	Application Programming Interface
ML	Machine Learning
DL	Deep Learning

CHAPTER 1

INTRODUCTION

Plant disease detection is crucial for preventing crop losses and ensuring healthy yields. With advancements in technology, such as image processing and AI, automated systems can quickly and accurately identify diseases from plant images. This enables early intervention, reduces pesticide use, and supports sustainable farming practices, ultimately improving crop productivity and food security. The user-friendly solution benefits farmers, researchers, and agricultural professionals.

1.1 Overview:

Plant disease detection plays a vital role in modern agriculture by helping prevent crop losses, ensuring healthier yields, and contributing to sustainable farming practices. With the rapid growth of technology, particularly image processing and artificial intelligence (AI), automated systems now offer efficient and accurate methods for detecting diseases in plants from images. These systems can identify various diseases, enabling early intervention to control outbreaks and reducing the need for excessive pesticide use. The primary goal is to promote healthier crops, enhance food security, and improve overall agricultural productivity.

1.2 Scope:

The scope of plant disease detection encompasses several key areas:

1. Early Detection and Diagnosis: Identifying diseases at early stages to prevent widespread damage.
2. Disease Prediction: Using historical data and AI models to predict disease outbreaks, allowing for proactive measures.
3. Automation in Monitoring: Employing drones, cameras, and sensors to monitor plant health in large-scale farming operations.
4. Disease Identification: Accurate identification of diseases based on images and symptoms of the plants.

1.3 Features:

1. Image-Based Disease Detection: The system uses images of plants (captured by drones, cameras, or smartphones) to detect visual signs of diseases, such as leaf spots, discoloration, or wilting.
2. Machine Learning and AI Models: The use of advanced AI algorithms, such as deep learning, enables the system to analyze images and classify diseases accurately. These models are trained on a large dataset of images representing different plant diseases.
3. Real-Time Monitoring: With AI-based processing, farmers and agricultural professionals can receive real-time feedback about the health of their crops, enabling quick action to be taken.
4. Automatic Disease Classification: Once an image is processed, the system can identify the specific disease affecting the plant, reducing the need for manual diagnosis and expert intervention.
5. Integrated Solutions: The disease detection system can be integrated into existing farming technologies, such as crop management software and IoT devices, to provide a more holistic view of plant health.

CHAPTER 2

LITERATURE SURVEY

In the field of plant disease detection, various studies have explored different methodologies and technologies to improve the accuracy and efficiency of disease identification.

One study focuses on vision-based AI techniques like Convolutional Neural Networks (CNNs) and Transfer Learning for plant disease detection. This methodology offers high accuracy through deep learning models and analyzes various datasets. However, it highlights the challenge of developing robust models suitable for real-world applications, especially for IoT systems, and the high computational cost. The future direction suggests developing lightweight models for real-time and IoT-based applications. [1]

Another study uses machine learning with color, texture, and morphology features for plant leaf disease detection. The advantage of this approach lies in the effective use of morphological features, making it adaptable for different plant types and diseases. Despite this, high computational costs and the lack of robust models for real-world use remain significant challenges. Future research may focus on utilizing larger datasets and improving feature extraction methods. [2]

A separate study uses automatic visual detection, machine learning algorithms, image processing, and sensor technology to classify plant diseases. It points out the benefits of faster, automated disease detection, with potential for large-scale use. However, challenges include limited data on the efficiency of these methods and the reliance on technology. Future research is expected to focus on improving accuracy and integrating IoT and AI for real-time monitoring. [3]

In another study, advanced image processing techniques and transfer learning with GoogleNet and AlexNet achieved high accuracy in detecting diseases across multiple crops. While GoogleNet achieved 99.35% accuracy, AlexNet showed a lower performance, and both models required significant computational resources. Future work may focus on exploring additional architectures and datasets to enhance results. [4]

A review of plant leaf disease detection using computer vision and AI emphasizes early disease detection, automation, and high accuracy. However, challenges such as data dependency and overfitting are mentioned. The future of this field involves integrating more advanced technologies and addressing the high costs of sophisticated equipment. [5]

A study focused on wheat leaf disease identification used transformer models with federated learning, achieving high accuracy. However, these models require substantial computational resources for initial training and optimization. The future scope is to enhance scalability and adapt these models to diverse environmental conditions. [6]

Another study developed a real-time dataset for rice, wheat, and maize diseases, achieving high accuracy with advanced models. Some challenges include preprocessing issues and limited real-life factors. Future work may include expanding datasets with more crops and exploring object detection models. [7]

Lastly, a study using transfer learning techniques like DenseNet, VGG, and ResNet on large datasets showed high accuracy, especially on the PlantVillage dataset. However, large datasets require significant computational resources. The future scope of research may involve exploring more efficient transfer learning methods that require fewer resources, making them suitable for diverse environments. [8]

.

CHAPTER 3

PROBLEM STATEMENT AND OBJECTIVE

3.1 Problem Statement:

- In modern agriculture, plant diseases significantly affect crop yield and quality, leading to substantial economic losses.
- Farmers and agricultural professionals often struggle to diagnose diseases in plants quickly and accurately, leading to delayed interventions and unnecessary chemical treatments.
- Traditional methods of disease detection are time-consuming, costly, and require expert knowledge, which is not always readily accessible to farmers.
- Therefore, there is a pressing need for an automated and efficient system that can identify plant diseases from leaf images with high accuracy, enabling timely intervention and reducing the reliance on harmful chemicals.

1. Widespread Impact of Plant Diseases**:**

Plant diseases are a major threat to global agriculture, causing significant damage to crops and leading to reduced productivity and quality.

2. Challenges in Early Detection**:**

Farmers face difficulties in diagnosing plant diseases at an early stage, which often results in delayed interventions and increased damage to crops.

3. Over-reliance on Chemical Treatments:

Traditional disease detection methods rely on the use of chemical treatments that are frequently applied without proper identification, leading to overuse, environmental pollution, and health risks.

4. Time-Consuming and Expensive:

Current disease diagnosis methods are often time-consuming, labor-intensive, and costly, requiring expert knowledge and manual labor.

5. Limited Access to Expert Knowledge**:**

Many farmers, particularly in rural or underserved areas, do not have easy access to experts or advanced tools for disease detection, which limits their ability to make timely and informed decisions.

6. Economic Losses for Farmers**:**

Delayed or incorrect diagnosis of plant diseases leads to crop loss, poor harvests, and financial instability for farmers, especially in developing regions.

7. Need for Cost-Effective and Accessible Solutions**:**

There is a growing need for affordable, accurate, and user-friendly solutions for plant disease detection that can be easily accessed and used by farmers, even with minimal technological expertise.

8. Environmental Concerns:

The indiscriminate use of chemicals to treat diseases without accurate diagnosis leads to negative environmental consequences, including soil degradation and contamination of water sources.

9. Need for Technological Advancement**:**

There is a need for an automated, technology-driven system that can provide quick, reliable, and accurate disease diagnosis to help farmers reduce losses and improve crop health.

3.2 Objective:

- The primary goal of this project is to design and implement a Plant Disease Detection System that leverages advanced image processing techniques and Convolutional Neural Networks (CNNs) for the automatic identification and classification of plant diseases from leaf images.

- The system will be capable of providing accurate, real-time diagnostics, enabling farmers to take timely action to prevent further disease spread and reduce reliance on harmful chemical treatments.

- This system will feature a user-friendly interface suitable for mobile or web platforms, ensuring accessibility for farmers with varying levels of technological expertise.

In addition to early disease detection, the project aims to:

- Enhance agricultural productivity by reducing crop loss and improving disease management, leading to healthier crops.
- Promote sustainable agricultural practices by enabling farmers to use precision agriculture techniques, including targeted treatment of affected plants, thereby reducing the need for widespread pesticide use.
- Empower farmers to make data-driven decisions based on reliable disease diagnoses, fostering greater confidence in adopting innovative technologies.
- Contribute to economic stability for farmers by reducing the costs associated with crop diseases and minimizing financial losses from untreated or misdiagnosed diseases.

CHAPTER 4
REQUIREMENT SPECIFICATION

4.1Functional Requirements:

- Image Input: Users (farmers and agricultural professionals) can upload leaf images via mobile or web applications.
- Disease Detection: The system analyzes uploaded images using advanced image processing and CNN-based machine learning to detect and classify plant diseases.
- Real-time Diagnosis: Provides real-time disease diagnosis, delivering results within seconds after image analysis.
- Disease Classification**:** Classifies detected diseases into categories based on pre-trained models, offering relevant disease information.
- User-friendly Interface: Features an intuitive, easy-to-use interface, requiring minimal technical expertise from users.
- Disease Prevention Recommendations: Suggests preventive measures and treatments, emphasizing sustainable practices.
- Mobile and Web Application **Access**: Accessible on both mobile devices (Android/iOS) and web platforms, offering flexible use.
- Data Storage: Stores uploaded images and diagnosis results for future reference and analysis, with cloud and local storage options.
- Multi-Language Support: Offers multi-language support to cater to a diverse user base from different regions.
- System Notifications: Notifies users about disease diagnosis, prevention tips, and updates for repeated use.

4.2 Non-Functional Requirements:

1. Performance:
   The system should process leaf images and return results in under 5 seconds to ensure timely diagnosis.

2. Scalability:
   The system should be scalable to handle large numbers of users, especially during peak usage seasons like planting or harvest time.

3. Security:

   User data (including images) should be securely stored and transmitted, adhering to data privacy regulations.

4. Reliability:

   The system should be highly reliable, with minimal downtime, ensuring availability when farmers need it most.

5. Usability:

   The user interface should be simple, intuitive, and easy to navigate for users with varying levels of technological expertise.

6. Accessibility:

   The system should comply with accessibility standards, making it usable for individuals with disabilities.

7. Maintainability:

   The system should be easy to update, maintain, and expand with new disease data and features.

8. Compatibility:

   The system should be compatible with various devices and platforms, including Android, iOS, and web browsers**.**

9. Energy Efficiency:

   The system should be optimized for efficient resource usage, ensuring minimal battery consumption on mobile devices.

10. Localization:

   The system should support localization to different regions and languages to cater to a global audience

4.2 Software Specification:

- Frontend**:** React(JavaScript)
- Backend**:** Node.js(Express)
- Database**:** MYSQL

4.3 Constraints and Assumptions:

1. Image Quality:
    - Constraint**:** Poor image quality affects accuracy.
    - Assumption: Farmers can capture clear images.
2. Device and Network**:**
    - Constraint: Low device power and poor internet may impact performance.
    - Assumption**:** Farmers have suitable devices and internet access.
3. Training Data**:**
    - Constraint**:** Limited data may reduce classification accuracy.
    - Assumption**:** Sufficient and diverse datasets are available.
4. Model Accuracy**:**
    - Constraint: Model may struggle with unseen diseases or bad images.
    - Assumption: Regular updates to the model ensure accuracy.
5. User Expertise**:**
    - Constraint**:** Limited tech knowledge may affect usage.
    - Assumption: The system will be user-friendly and require minimal training.
6. Sustainable Practices**:**
    - Constraint: Limited access to eco-friendly treatments in some areas.
    - Assumption**:** Farmers will have access to sustainable solutions.

.

CHAPTER 5

SYSTEM DESIGN

5.1 USE CASE DIAGRAM:

The image represents a use case diagram for a plant disease detection system, illustrating the interactions between three primary actors: the Farmer, the Expert or Plant Specialist, and the Detection System. The Farmer is responsible for uploading plant images, viewing database diagnoses, and receiving disease-related advice. The Expert or Plant Specialist plays a more analytical role, verifying diagnoses, updating the disease database, and providing recommendations based on their expertise. The Detection System is an automated component that detects plant diseases, provides initial diagnoses, and generates reports. Each use case represents a specific function within the system, emphasizing the collaboration between users and technology to ensure accurate and efficient disease identification and management in agricultural practices.

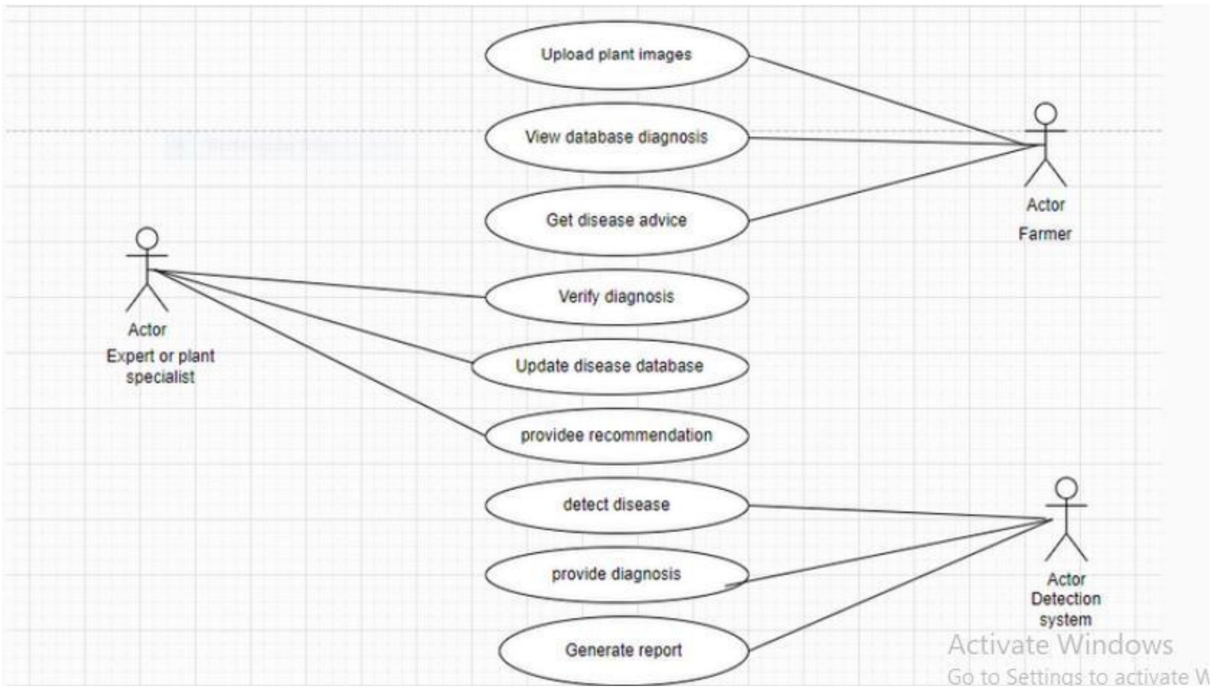


Fig 5.1:Use Case

5.2 SYSTEM ARCHITECTURE:

A **System Architecture Diagram** provides a high-level view of the system's components and how they interact with each other. It focuses on the overall structure of the system, showing its major parts and how they work together. The image illustrates a flowchart of a plant leaf disease detection system using deep learning and advanced optimization techniques. The process begins with an input leaf image from a new plant disease dataset. The image undergoes deep learning-based detection to identify potential disease symptoms. It then moves through pre-processing, specifically using sampling subset feature filtering, to refine the data. Following this, the system applies Ada Boosting Region-Based Segmentation for accurate region extraction, alongside a Spider Optimization-Based Maximum Feature Weighting technique to enhance relevant feature selection.

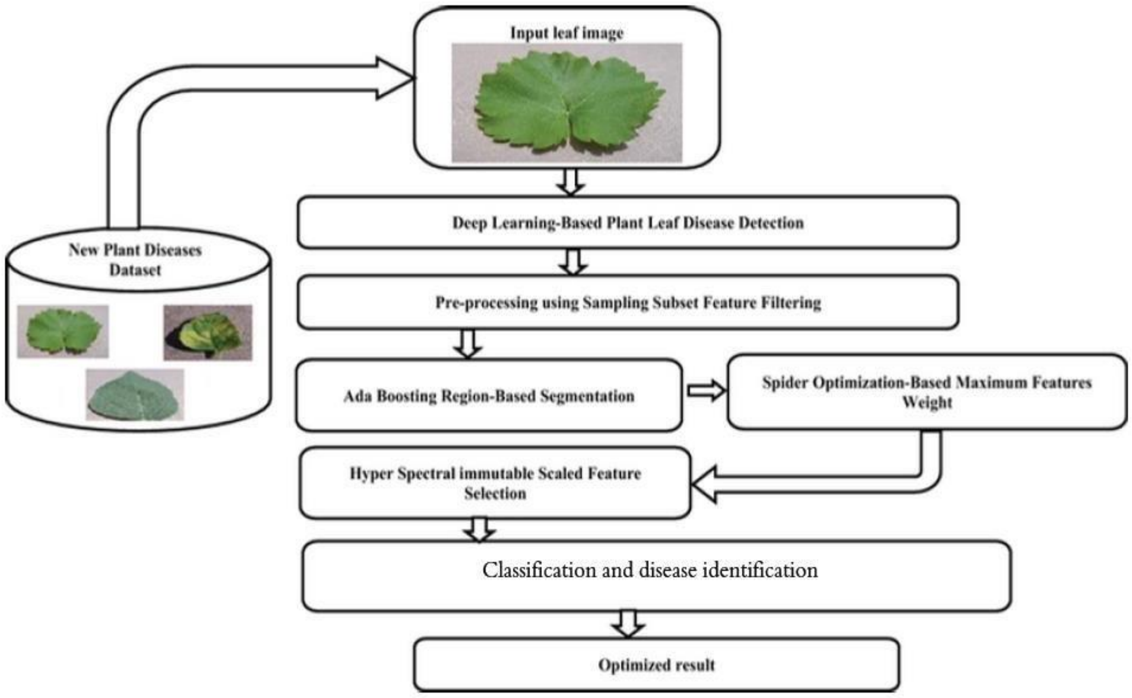

Input leaf image
Deep Learning-Based Plant Leaf Disease Detection
New Plant Diseases Dataset
Pre-processing using Sampling Subset Feature Filtering
Ada Boosting Region-Based Segmentation
Spider Optimization-Based Maximum Features Weight
Hyper Spectral immutable Scaled Feature Selection
Classification and disease identification
Optimized result

Fig 5.1:System Architecture

5.3 APPLICATION ARCHITECTURE DIAGRAM:

The Application Architecture Diagram focuses specifically on the software components and how they interact. It is a subset of the system architecture diagram, and it details the logical flow of the application

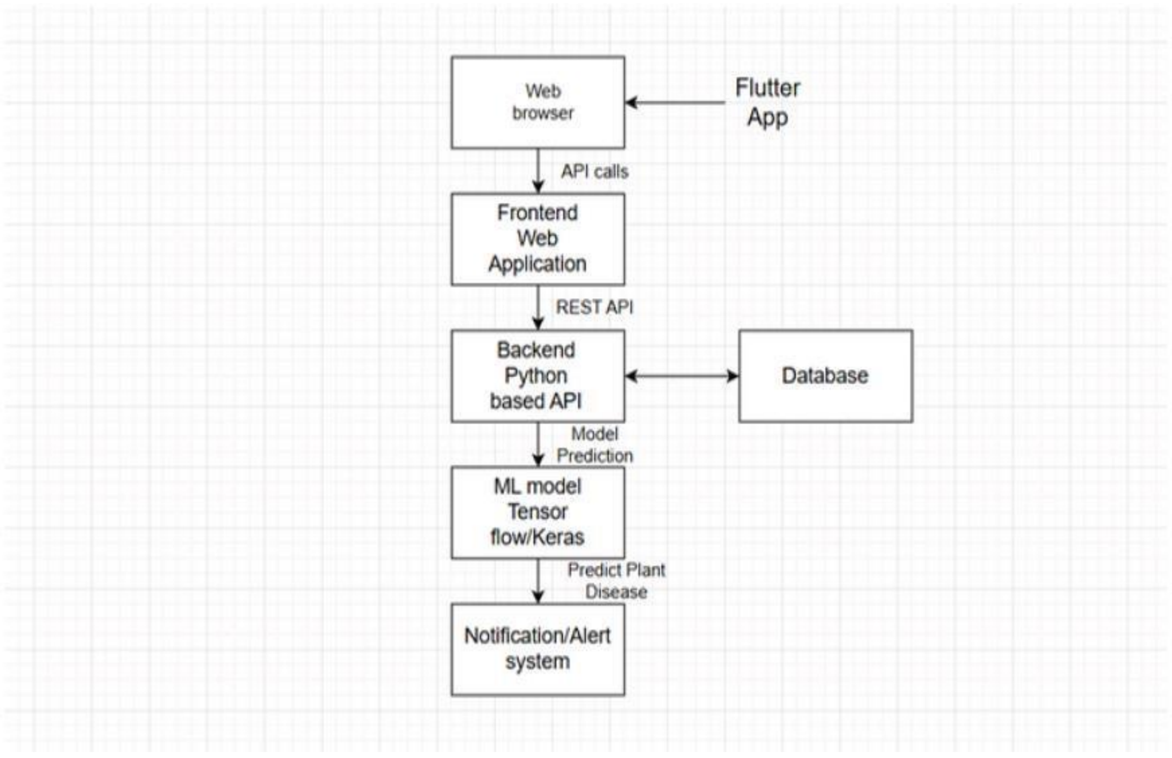


Fig 5.2:Application Architecture

5.4 DATABASE DIAGRAM:

A Database Diagram (often called an Entity-Relationship Diagram or ERD) shows the structure of a database. It highlights how data is organized within tables and how these tables relate to each other.

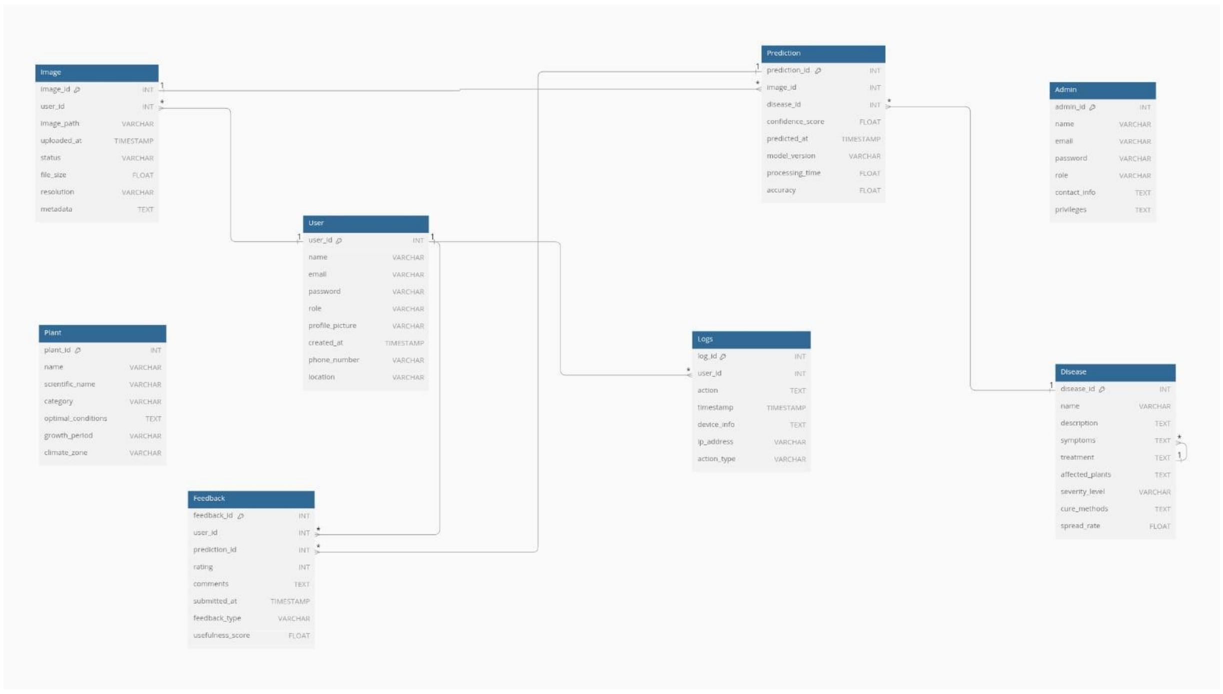


Fig5.3:Database diagram

5.5 ER DIAGRAM:

An Entity-Relationship (ER) diagram is a visual representation of the entities within a system and the relationships between them. The primary goal of an ER diagram is to illustrate how data is related within a system and how various pieces of information are interconnected.

The image is an Entity-Relationship Diagram (ERD) that outlines the database structure for a plant disease detection system. It features several key entities and their relationships. The **User** entity includes attributes such as name, email, role, and password, and is associated with uploading **Images**, providing **Feedback**, and generating **Logs** to track system activity. Each uploaded image can be linked to a **Prediction**, which contains information like confidence score, model version, and associated disease. The **Disease** entity includes detailed attributes such as name, description, treatment, severity level, and the plants it affects. It corresponds to predictions made from user-uploaded images. The **Plant** entity stores information like scientific name, category, and is linked to diseases that affect it. The **Admin** entity is responsible for managing plants and monitoring logs, with details like name, email, and role. This ERD provides a clear overview of how data flows through the system and how different components interact to support disease identification and user engagement.

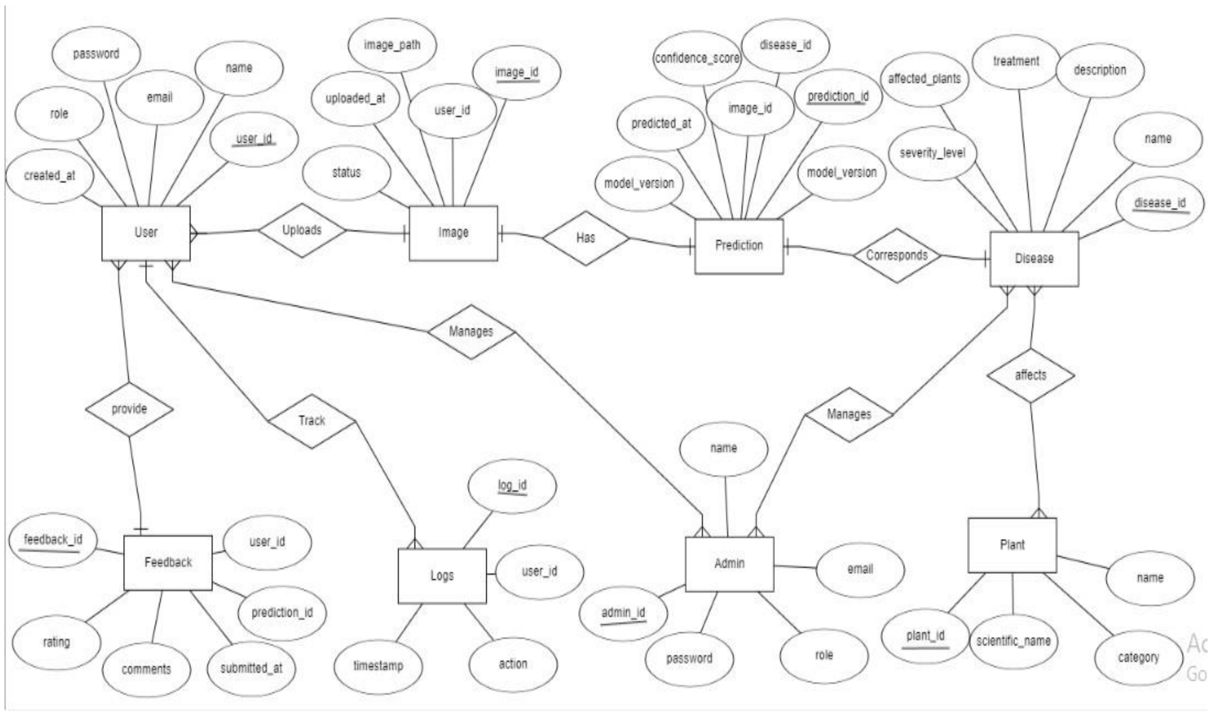


Figure 4.5:ER Diagram

CHAPTER 6

SYSTEM IMPLEMENTATION

PROGRAM CODES:

Frontend

Indexpage:

```
import React from "react";

import { Link } from "react-router-dom";

import "./styles/styles.css";


const Header = () => (

  <header>

    <div className="logo">PlantCare</div>

    <nav>

      <ul>

        <li><Link to="/">Home</Link></li>

        <li><a href="#about">About</a></li>

        <li><a href="#features">Features</a></li>

        <li><a href="#contact">Contact</a></li>

      </ul>

    </nav>

  </header>

);
```

```
const Hero = () => (
  <section className="hero">
    <h1>Detect Plant Diseases with AI</h1>
    <p>Upload an image and get instant analysis for plant diseases.</p>
    <Link to="/login" className="btn">Get Started</Link>
  </section>
);


const About = () => (
  <section id="about" className="about">
    <h2>About Our System</h2>
    <p>Our AI-powered plant disease detection system helps farmers and gardeners identify
plant diseases early, enabling them to take timely preventive measures.</p>
  </section>
);

const Features = () => (
  <section id="features" className="features">
    <h2>Features</h2>
    <div className="feature-list">
      <div className="feature">
        <h3>AI Detection</h3>
        <p>Analyze plant images with high accuracy.</p>
      </div>
      <div className="feature">
        <h3>Remedies & Suggestions</h3>
        <p>Get expert-approved solutions for detected diseases.</p>
      </div>
      <div className="feature">
        <h3>Easy to Use</h3>
        <p>User-friendly interface for seamless experience.</p>
      </div>
    </div>
  </section>
);

const Contact = () => (
  <section id="contact" className="contact">
    <h2>Contact Us</h2>
    <p>Email: support@plantcare.com</p>
  </section>
);

const Footer = () => (
  <footer>
    <p>&copy; 2025 PlantCare. All rights reserved.</p>
  </footer>
);
```

```
// 🟩 Fix: Rename indexPage to IndexPage (React component names should start with
uppercase)
const IndexPage = () => (
  <>
    <Header />
    <Hero />
    <About />
    <Features />
    <Contact />
    <Footer />
  </>
);

export default IndexPage; // 🟩 Fix the export
```

Login:

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import axios from "axios";  // 🟩 Import axios for API requests
import "./styles/styles.css";

const Login = ({ setIsAuthenticated }) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [showPassword, setShowPassword] = useState(false);
  const [rememberMe, setRememberMe] = useState(false);
  const navigate = useNavigate();
```

```
// 🟩 Load remembered email from localStorage
useEffect(() => {
  const rememberedUser = localStorage.getItem("rememberedUser");
  if (rememberedUser) {
    setEmail(rememberedUser);
    setRememberMe(true);
  }
}, []);

// 🟩 Handle login
const handleLogin = async (event) => {
  event.preventDefault();

  try {
    const response = await axios.post("http://localhost:5000/api/login", {
      email,
      password,
    });

    alert(response.data.message);

    if (response.status === 200) {
      setIsAuthenticated(true);

      if (rememberMe) {
        localStorage.setItem("rememberedUser", email);
      } else {
        localStorage.removeItem("rememberedUser");
      }

      navigate("/profile"); // 🟩 Redirect to profile
    }
  } catch (error) {
    alert(error.response?.data?.message || "Login failed");
  }
};

return (
  <div className="login-container">
    <h2>Login</h2>
    <form onSubmit={handleLogin}>
      <div className="input-group">
        <input
          type="email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          placeholder="Email"
          required
        />
      </div>

      <div className="input-group">
        <input
          type={showPassword ? "text" : "password"}
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          placeholder="Password"
          required
        />
        <label>
          <input
            type="checkbox"
            checked={showPassword}
            onChange={() => setShowPassword(!showPassword)}
          />
          Show Password
        </label>
      </div>

      <div className="remember-forgot">
        <label>
          <input
            type="checkbox"
            checked={rememberMe}
            onChange={(e) => setRememberMe(e.target.checked)}
          />
          Remember Me
        </label>
      </div>
```

```
        <button type="submit" className="btn">Login</button>

        <p>Don't have an account? <a href="/signup">Sign Up</a></p>
      </form>
    </div>
  );
};

export default Login;
```

Upload page:

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import "./styles/styles.css";

const UploadPage = () => {
  const [selectedFile, setSelectedFile] = useState(null);
  const [preview, setPreview] = useState(null);
  const [message, setMessage] = useState("");
  const navigate = useNavigate();

  // Handle file selection
  const handleFileChange = (event) => {
    const file = event.target.files[0];
    if (file) {
      setSelectedFile(file);
      setPreview(URL.createObjectURL(file));
      setMessage("");
    }
  };

  // Handle drag-and-drop
  const handleDragOver = (event) => event.preventDefault();
  const handleDrop = (event) => {
    event.preventDefault();
    const file = event.dataTransfer.files[0];
    if (file) {
      setSelectedFile(file);
      setPreview(URL.createObjectURL(file));
      setMessage("");
    }
  };

  // Handle Upload and Prediction
  const handleUpload = async () => {
  if (!selectedFile) {
      setMessage("¡ Please select an image to upload."); return;
    }

    const formData = new FormData();
    formData.append("image", selectedFile);

    try {
      const response = await fetch("http://127.0.0.1:5000/predict", {
        method: "POST",
        body: formData,
      });

      const data = await response.json();
      if (data.success) {
      navigate("/prediction", {
          state: {
            imageUrl: preview,
            disease: data.disease,
            suggestions: data.suggestions,
          },
        });
      } else {
        setMessage("+ Prediction failed. Try again.");
      }
    } catch (error) {
      console.error("Error uploading image:", error);
      setMessage("+ Error uploading image.");
    }
  };

  return (
    <div className="upload-container">
      <h2>🌿Upload an Image for Plant Disease Detection</h2>
      <p>Choose a clear image of the plant leaf to analyze for diseases.</p>
      {/* Drag & Drop or Click to Upload */}
      <div className="upload-box" onDragOver={handleDragOver} onDrop={handleDrop}>
        <label className="upload-label">
          📤 Drag & Drop or Click to Upload
          <input type="file" accept="image/*" onChange={handleFileChange} hidden />
        </label>
      </div>

      {/* Preview Image */}
      {preview && (
        <div className="preview-container">
          <h4> 🖼️Preview</h4>
          <img src={preview} alt="Preview" className="preview-image" />
        </div>
      )}

      {/* Upload Button */}
      <button onClick={handleUpload} className="btn">Upload & Predict</button>

      {/* Success/Error Message */}
      {message && <p className="message">{message}</p>}
    </div>
  );
 };

 export default UploadPage
```

Userprofile:

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import { Link } from "react-router-dom";
import "./styles/styles.css"; // 🟩 Ensure this path is correct

const UserProfile = () => {
  const navigate = useNavigate(); // 🟩 Use React Router navigation

  const [profileImage, setProfileImage] = useState(
    localStorage.getItem("profileImage") || "/default-profile.jpg"
  );
  const [name, setName] = useState(localStorage.getItem("userName") || "Your Name");
  const [email, setEmail] = useState(localStorage.getItem("userEmail") ||
"your.email@example.com");
  const [phone, setPhone] = useState(localStorage.getItem("userPhone") || "");
  const [gender, setGender] = useState(localStorage.getItem("userGender") || "");
  const [editMode, setEditMode] = useState(false);

  const handleImageChange = (event) => {
    const file = event.target.files[0];
    if (file) {
      const reader = new FileReader();
      reader.onload = (e) => {
      setProfileImage(e.target.result);
        localStorage.setItem("profileImage", e.target.result);
      };
      reader.readAsDataURL(file);
    }
  };

  const handleSaveProfile = () => {
    localStorage.setItem("userName", name);
    localStorage.setItem("userEmail", email);
    localStorage.setItem("userPhone", phone);
    localStorage.setItem("userGender", gender);
    alert("Profile Updated Successfully!");
    setEditMode(false);
  };

  const handleLogout = () => {
    localStorage.removeItem("loggedInUser");
    navigate("/"); // 🟩 Redirect to home instead of "index.html"
  };

  return (
    <div>
      <header>
        <div className="logo">Plant Disease Detection</div>
        <nav>
          <ul>
            <li><Link to="/dashboard">Dashboard</Link></li> {/* 🟩 Use React Router */}
            <li><Link to="/profile">Profile</Link></li>
            <li><button onClick={handleLogout} className="btn">Logout</button></li>
          </ul>
        </nav>
      </header>

      <div className="profile-container">
        <div className="profile-card">
          <label htmlFor="profileImageInput">
            <img src={profileImage} alt="Profile" className="profile-pic" />
          </label>
          <input
            type="file"
            id="profileImageInput"
            accept="image/*"
            onChange={handleImageChange}
            hidden
          />
          <h2 className="username">{name}</h2>
          <p className="user-email">{email}</p>
          <button className="btn edit-profile" onClick={() => setEditMode(true)}>
            Edit Profile
          </button>
        </div>

        {editMode && (
          <div className="edit-profile-section">
            <label>Full Name:</label>
            <input type="text" value={name} onChange={(e) => setName(e.target.value)} />
            <label>Email:</label>
            <input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />
            <label>Phone Number:</label>
            <input type="tel" value={phone} onChange={(e) => setPhone(e.target.value)} />
            <label>Gender:</label>
            <select value={gender} onChange={(e) => setGender(e.target.value)}>
              <option value="">Select Gender</option>
              <option value="Male">Male</option>
              <option value="Female">Female</option>
              <option value="Other">Other</option>
            </select>
            <button className="btn save-profile" onClick={handleSaveProfile}>
              Save Changes
            </button>
          </div>
        )}

        <div className="upload-section">
          <h3>Upload Plant Image</h3>
          <button className="btn upload-btn" onClick={() => navigate("/upload")}>
            Upload
          </button>
        </div>
      </div>

      <footer>
        <p>&copy; 2025 Plant Disease Detection. All rights reserved.</p>
```

```
      </footer>
    </div>
  );
};

export default UserProfile;
```

Prediction page:

```
import { useLocation, useNavigate } from "react-router-dom";
import { useState } from "react";
import "./styles/styles.css"; // Ensure this is linked for styling

const PredictionPage = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const { imageUrl, disease, suggestions } = location.state || {};
  const [feedback, setFeedback] = useState("");
  const [isSubmitting, setIsSubmitting] = useState(false);

  const handleFeedbackSubmit = async () => {
    if (!feedback.trim()) {
      alert("¡ Please enter feedback before submitting."); return;
    }

    setIsSubmitting(true);

    try {
      const response = await fetch("http://localhost:5000/feedback", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ feedback }),
      });

      if (response.ok) {
        alert("✅Feedback submitted successfully!");
        setFeedback("");
      } else {
        alert("+ Failed to submit feedback. Try again.");
      }
    } catch (error) {
      console.error("Error submitting feedback:", error);
      alert("+ Error submitting feedback.");
    } finally {
      setIsSubmitting(false);
    }
  };

  return (
    <div className="prediction-container">
      <h2>🔍 Prediction Result</h2>

      {/* Display Uploaded Image */}
      {imageUrl && (
        <div className="image-preview">
          <h4>🖼️ Uploaded Image</h4>
          <img src={imageUrl} alt="Uploaded Plant" className="uploaded-image" />
        </div>
      )}

      {/* Prediction Results */}
      <p><strong>     Disease:</strong> {disease || "No prediction available"}</p>
      <p><strong>💡 Suggestions:</strong> {suggestions || "No suggestions available"}</p>

      {/* Feedback Section */}
      <div className="feedback-section">
        <h3>< Provide Feedback</h3>
        <textarea
          value={feedback}
          onChange={(e) => setFeedback(e.target.value)}
          placeholder="Enter your feedback..."
          rows="3"
        />
        <button onClick={handleFeedbackSubmit} disabled={isSubmitting}>
          {isSubmitting ? "Submitting..." : "Submit Feedback"}
        </button>
      </div>

      {/* Navigation Buttons */}
      <div className="button-group">
      <button onClick={() => navigate("/upload", { state: { fromPrediction: true } })}>
```

```
        Try Another Image
</button>


      </div>
    </div>
  );
};


export default PredictionPage;
```

Backend:

Database:

```
const mysql = require("mysql");
const db = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "Adharsh@123",
    database: "plant_disease_db",
});
db.connect((err) => {
    if (err) {
        console.error("+ Database connection failed:", err);
        return;
    }
    console.log(" 🟩Connected to MySQL database!");
});
```

```
module.exports = db;
```

server.js

```
const express = require("express");

const cors = require("cors");

const multer = require("multer");

const { spawn } = require("child_process");

const path = require("path");

const db = require("./db");

const routes = require("./routes");


const app = express();


// 🟩 Middleware

Setup app.use(cors());

app.use(express.json()); // Properly parse JSON requests

app.use(express.urlencoded({ extended: true })); // Parse URL-encoded requests


// 🟩 Debugging Middleware to Log Incoming Requests

app.use((req, res, next) => {

   console.log("* Incoming request:", req.method, req.url);

   console.log("*  Headers:",  req.headers);

   console.log("* Body:", req.body);

   next();

});
```

```
// 🟩 Routes

app.use("/api", routes);


// 🟩 File Upload Setup

const upload = multer({ dest: "uploads/" });


// 🟩 Prediction Endpoint

app.post("/predict", upload.single("image"), (req, res) => {

   if (!req.file) {

      return res.status(400).json({ success: false, error: "No file uploaded" });

   }


   const imagePath = path.join(__dirname, req.file.path);

   console.log("🟩Image uploaded:", imagePath);


   // Run the Python script

   const pythonProcess = spawn("python", ["predict.py", imagePath]);


   let result = "";

   let errorOutput = "";


   pythonProcess.stdout.on("data", (data) => {

      result += data.toString();

   });
```

```
    pythonProcess.stderr.on("data", (data) => {
        errorOutput += data.toString();
    });

    pythonProcess.on("close", (code) => {
        if (code !== 0) {
            console.error(+ Python process exited with code ${code});
            console.error(+ Error: ${errorOutput});
            return res.status(500).json({ error: "Prediction failed", details: errorOutput });
        }
        console.log(🟩Prediction result: ${result.trim()});
        res.json({ result: result.trim() });
    });
});

// 🟩 Global Error Handler
app.use((err, req, res, next) => {
    if (err instanceof SyntaxError && err.status === 400 && "body" in err) {
        console.error("+ JSON Parse Error:", err.message);
        return res.status(400).json({ error: "Invalid JSON format" });
    }
    console.error("+ Server Error:", err.message);
    res.status(500).json({ error: "Internal server error" });
});
```

```
// 🟩 Start Server

const PORT = 5000;

app.listen(PORT, () => {

  console.log(✅ Server running on port ${PORT});

});
```

routes.js

```
const express = require("express");

const router = express.Router();

const db = require("./db");

const bcrypt = require("bcrypt");

const multer = require("multer");

const path = require("path");


// 🟩 Configure Multer for Image Uploads

const storage = multer.diskStorage({

destination: (req, file, cb) => {

  cb(null, "uploads/"); // Save files in 'uploads' directory

 },

 filename: (req, file, cb) => {

  cb(null, Date.now() + path.extname(file.originalname)); // Rename file with
timestamp

 },
});

const upload = multer({ storage: storage });

// 🟩 Signup Route (with hashed password)
router.post("/signup", async (req, res) => {
console.log("* Received Signup Request:", req.body);

  const { name, email, password } = req.body;

  if (!name || !email || !password) {
    return res.status(400).json({ success: false, message: "All fields are required" });
  }

  try {
    const hashedPassword = await bcrypt.hash(password, 10); // Hash password before
storing

    const query = "INSERT INTO users (name, email, password) VALUES (?, ?, ?)";
    db.query(query, [name, email, hashedPassword], (err, result) => {
      if (err) {
        console.error("+ Database Insert Error:", err);
        return res.status(500).json({ success: false, message: "Database error" });
      }
      res.json({ success: true, message: "User registered successfully" });
    });
  } catch (error) {
    console.error("+ Hashing Error:", error);
    res.status(500).json({ success: false, message: "Internal server error" });
  }
});

// 🟩 Login Route (with password verification)
router.post("/login", (req, res) => {
  console.log("* Received Login Request:", req.body);

  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(400).json({ success: false, message: "All fields are required" });
  }

  const query = "SELECT * FROM users WHERE email = ?";
  db.query(query, [email], async (err, results) => {
    if (err) {
      console.error("+ Database Error:", err);
      return res.status(500).json({ success: false, message: "Database error" });
    }

    if (results.length === 0) {
      return res.status(401).json({ success: false, message: "User not found" });
    }

    const user = results[0];

    try {
      const passwordMatch = await bcrypt.compare(password, user.password);
      if (!passwordMatch) {
        return res.status(401).json({ success: false, message: "Incorrect password" });
      }

      res.json({ success: true, message: "Login successful", user });
    } catch (error) {
      console.error("+ Password Comparison Error:", error);
      res.status(500).json({ success: false, message: "Internal server error" });
    }
  });
});

// 🟩 Prediction Route (Handles Image Upload)
router.post("/predict", upload.single("image"), (req, res) => {
console.log("* Received Image Upload Request");

  if (!req.file) {
    console.log("+  No  file  received");
    return res.status(400).json({ success: false, message: "No file uploaded" });

  }


  console.log(" 🟩 Image uploaded:", req.file.path);


  // TODO: Call the ML model for prediction (currently placeholder)

  const prediction = {

    disease: "Leaf Spot",

    suggestions: ["Use antifungal spray", "Maintain proper watering", "Prune infected
leaves"]

  };


  res.json({

    success: true,

    imageUrl: /uploads/${req.file.filename}, // Send back image URL

    disease: prediction.disease,

    suggestions: prediction.suggestions

  });

});

module.exports = router;
```

Model

```
import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import os
```

```
# Define paths
train_dir = "dataset/train"
val_dir = "dataset/valid"

img_size = (128, 128)
batch_size = 16  # Reduce to improve CPU performance

# Load datasets
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    image_size=img_size,
    batch_size=batch_size,
    label_mode="categorical",
    shuffle=True
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    val_dir,
    image_size=img_size,
    batch_size=batch_size,
    label_mode="categorical"
)

# Model definition
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), padding="same", input_shape=(128, 128, 3)),
    layers.BatchNormalization(),
    layers.LeakyReLU(negative_slope=0.1),
    layers.MaxPooling2D(),

    layers.Conv2D(64, (3, 3), padding="same"),
    layers.BatchNormalization(),
    layers.LeakyReLU(negative_slope=0.1),
    layers.MaxPooling2D(),

    layers.Conv2D(128, (3, 3), padding="same"),
    layers.BatchNormalization(),
    layers.LeakyReLU(negative_slope=0.1),
    layers.MaxPooling2D(),

    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(38, activation="softmax")
])

# Compile model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

# Train model
model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=10
)

# Save model
model.save("plant_disease_model.h5")
```

CHAPTER 7

SYSTEM TESTING

Test Case 1: Image Upload Functionality

This test case verifies whether the system allows users to upload images successfully. The image file should be properly accepted by the system, and the upload should complete without errors. A confirmation or visual indication should be displayed upon a successful upload.

Test Case 2: CNN-based Disease Classification

This test case checks whether the system correctly identifies plant diseases using a CNN-based model. Once the image is uploaded, the system should analyze it and accurately classify the disease based on the trained model.

Test Case 3: UI Responsiveness and Usability

This test case ensures that the user interface is responsive across various devices and resolutions. Additionally, it evaluates the ease of use, ensuring that all functionalities are accessible and user-friendly.

Test Case 4: Real-time Detection Performance

This test case validates the system's capability to process images and detect diseases within an acceptable time frame. It measures the performance and ensures that results are delivered quickly enough for practical use.

Test Case 5: Multiple Disease Identification

This test case evaluates whether the system can detect multiple diseases from a single image if present. The system should accurately identify and display all relevant diseases found in the uploaded image.

Test Case 6: Early Stage Disease Detection

This test case validates the system’s ability to detect diseases at an early stage. The model should be sensitive enough to recognize symptoms even when they are minimal or in the initial stages of development.

TEST CASES

Test Case ID	Description	Expected Result
TC1	Verify image upload functionality	Image is uploaded successfully.
TC2	Test CNN-based disease classification	The system correctly identifies the disease.
TC3	Check UI responsiveness and usability	The interface is user-friendly and responsive.
TC4	Validate real-time detection performance	The system processes images within an acceptable time.
TC5	Ensure identification of multiple diseases	All plant diseases are detected accurately.
TC6	Validate early stage disease detection	The system detects disease in early stage.

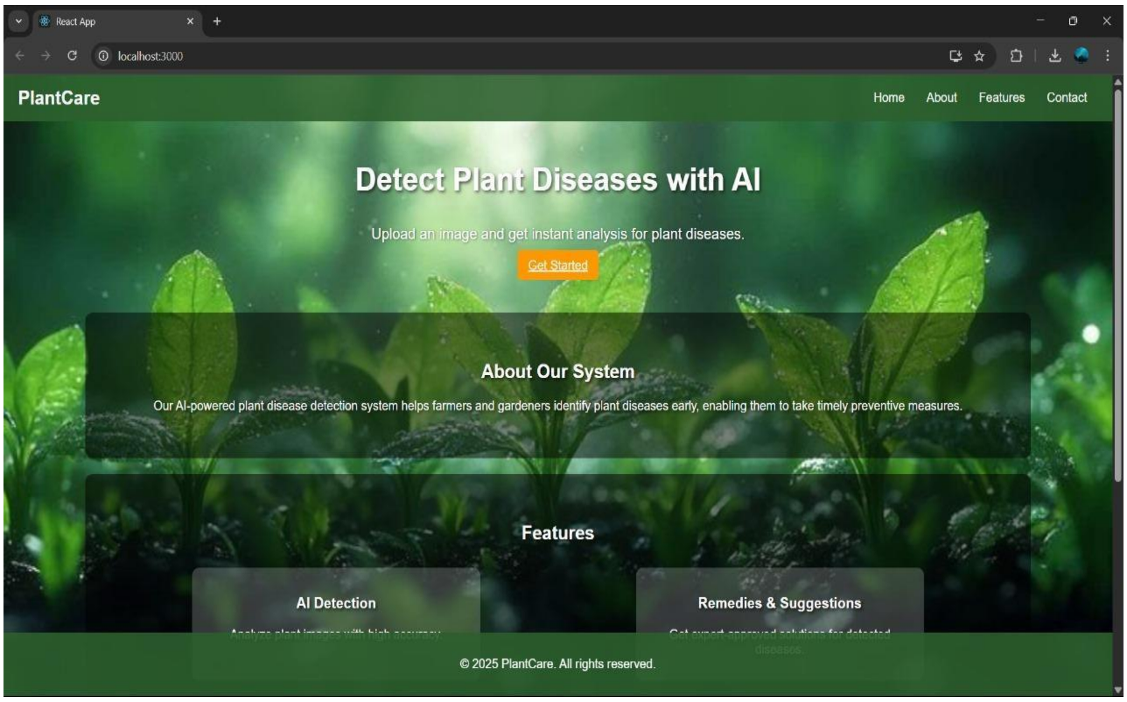
CHAPTER 8

RESULT

8.1 Index Page:

The image shows the homepage of a plant disease detection web application called PlantCare, running locally on localhost:3000. The page features a clean, green-themed design with a background image of healthy plants. At the center, there's a bold heading: "Detect Plant Diseases with AI", along with a subtext encouraging users to upload images for instant analysis. A prominent "Get Started" button is provided to initiate the process.

Below that, the "About Our System" section briefly explains that the AI-powered system helps farmers and gardeners detect plant diseases early for timely preventive measures. The Features section includes highlights such as AI Detection and Remedies & Suggestions, emphasizing the capabilities of the system. Navigation links (Home, About, Features, Contact) are located at the top right, and the footer notes the © 2025 copyright information for PlantCare.



Detect Plant Diseases with AI

Upload an image and get instant analysis for plant diseases.

About Our System

Our AI-powered plant disease detection system helps farmers and gardeners identify plant diseases early, enabling them to take timely preventive measures.

Analyze plant images with high accuracy.

Get expert-approved solutions for detected diseases.

© 2025 PlantCare. All rights reserved.

Fig 8.1:Index page

8.2 Login Page:

The image displays the Login page of the PlantCare web application, hosted locally at localhost:3000/login. The interface is clean and user-friendly, with a blurred, vibrant green background featuring healthy plants, maintaining visual consistency with the homepage.

At the center is a translucent login form that includes input fields for Email and Password, along with options to Show Password and Remember Me. A prominent orange “Login” button is provided for users to submit their credentials. Below the button, there is a link labeled “Sign Up” for users who do not yet have an account, guiding them to the registration page. The design emphasizes accessibility and a nature-inspired aesthetic suitable for an agricultural or plant-health-focused application

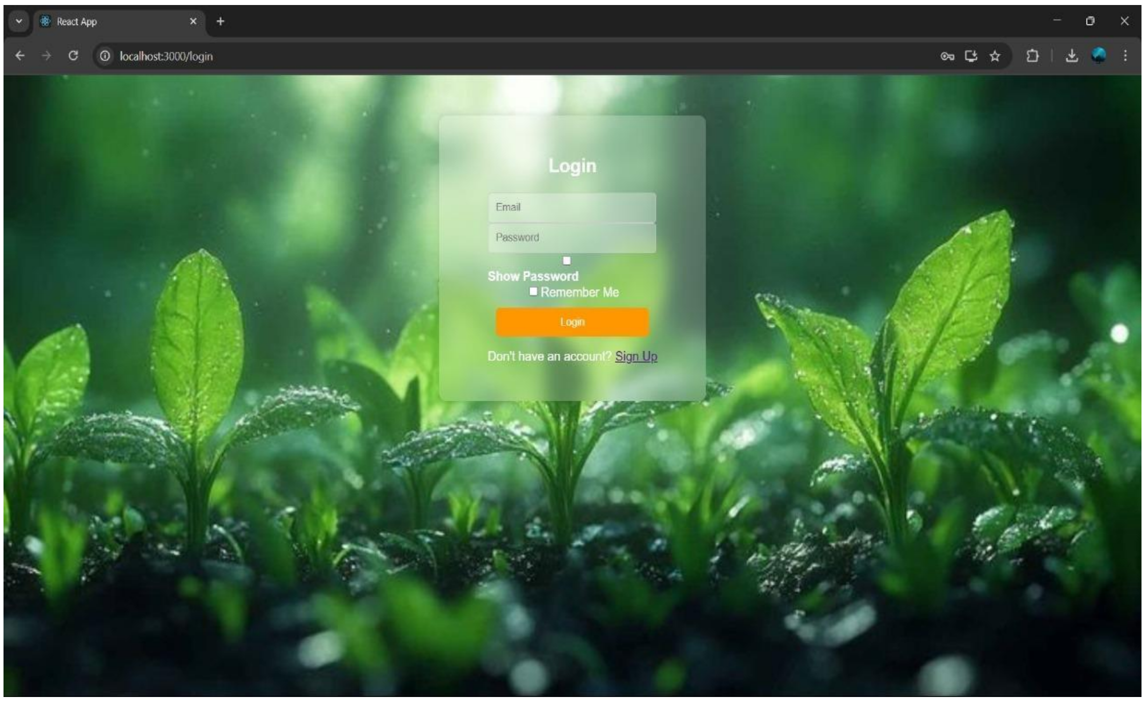


Fig 8.2:Login page

8.3Userprofile Page:

The image shows the User Profile page of the *Plant Disease Detection* web application, accessible at localhost:3000/profile. The interface features a nature-themed background with healthy green plants, maintaining visual harmony across the site.

At the center, a profile card displays the user's avatar, name ("Adharsh"), and email address. Below the user information, there are two key buttons:

- Edit Profile – Allows the user to update their personal information.

- Upload Plant Image – Enables the user to upload images of plant leaves for disease detection.

Navigation options like Dashboard, Profile, and a bright Logout button are placed in the top-right corner for easy access. The page is visually engaging and functionally designed to support plant disease identification workflows.

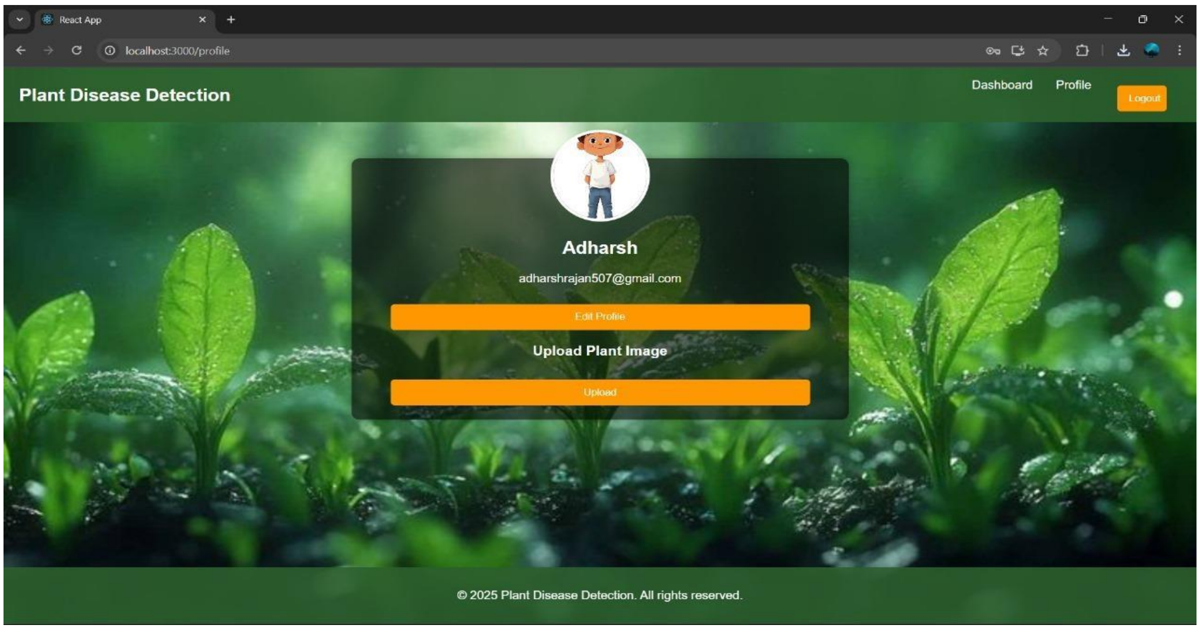


Fig 8.3:Userprofile page

8.4Upload Page:

The page invites users to "Upload an Image for Plant Disease Detection", offering a simple and user-friendly interface. At the center, there's a clickable upload area labeled "Click to Upload", where users can select a plant leaf image from their device. Below that, the bright "Upload & Predict" button allows users to initiate the AI-based analysis.

Consistent with the rest of the application, the background showcases fresh green plant leaves, reinforcing the theme of plant health and care. This page serves as the main entry point for performing disease predictions through image analysis.

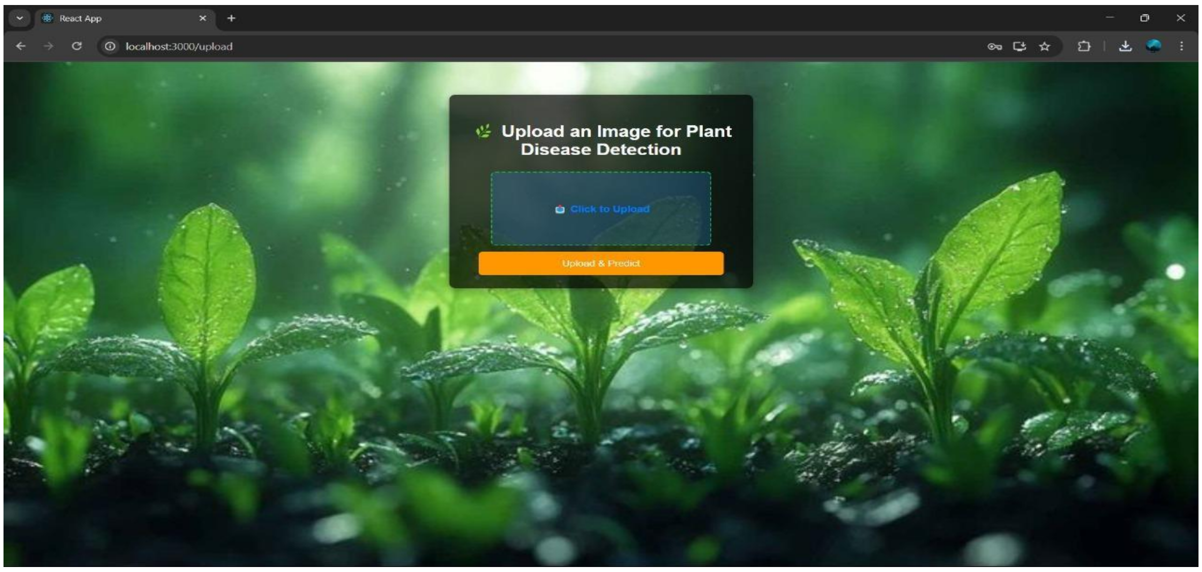
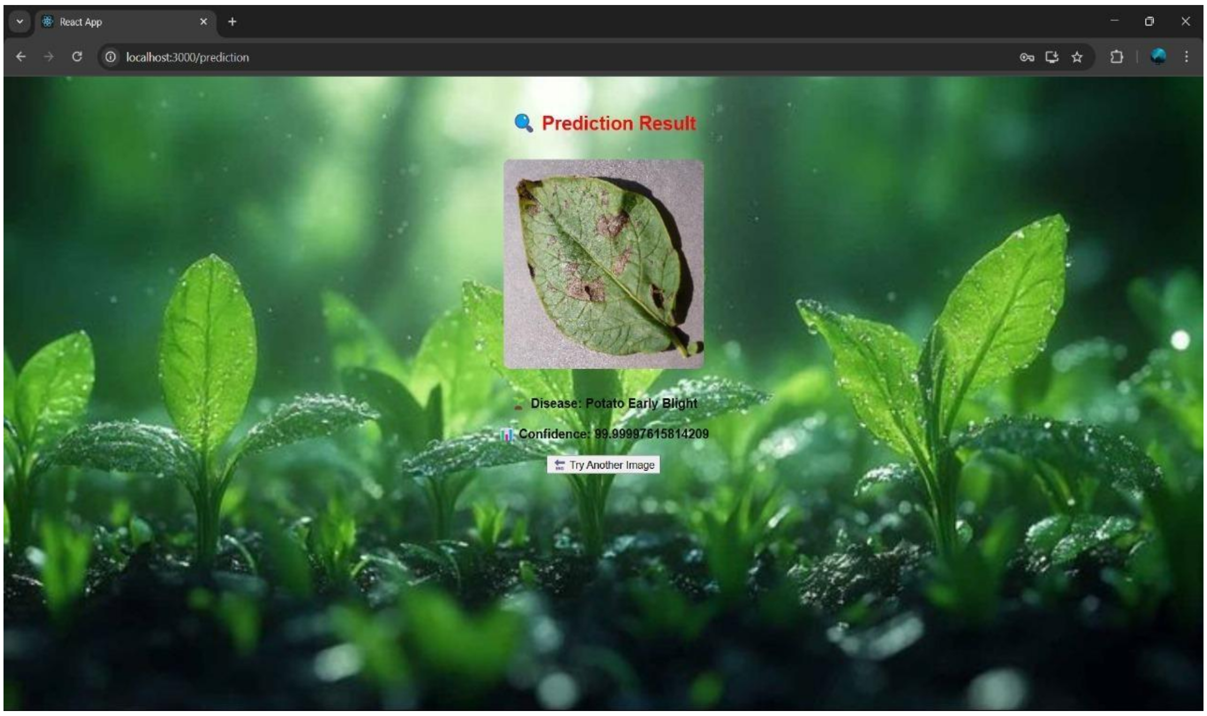


Fig 8.4:Upload Image

8.5Prediction Page:

A visually distinct red heading titled " Prediction Result" captures the user's attention, emphasizing the outcome of the AI analysis. Beneath the result, there is a button labeled "Try Another Image", which allows users to return and upload a new image for analysis.

The background maintains the application's consistent green aesthetic, reinforcing the theme of agriculture and plant health, and enhancing user engagement with a natural and calming visual design



Disease: Potato Early Blight

Confidence: 99.99997615814209

Fig 8.5:Prediction page

CHAPTER 9
CONCLUSION AND FUTURE SCOPE

The Plant Disease Detection System using advanced image processing and CNN-based machine learning represents a significant advancement in the field of agricultural technology. By accurately identifying and classifying diseases from leaf images, the system provides real-time, accessible diagnostics for farmers and agricultural professionals via mobile and web applications.

9.1 Conclusion:

1. Accurate Disease Detection: The system uses advanced image processing and machine learning to accurately identify and classify plant diseases from leaf images.
2. User-Friendly Interface: It provides a simple and easy-to-use platform for farmers and agricultural professionals to get real-time disease diagnosis via mobile or web apps.
3. Promotes Sustainability: The system helps reduce reliance on chemical treatments, promoting sustainable farming practices and healthier crops.
4. Improved Crop Health: Early disease detection leads to better management and healthier crops, improving overall agricultural productivity.
5. Economic Benefits: By reducing crop losses, the system helps farmers save money and avoid economic strain caused by undiagnosed diseases.
6. Accessible and Affordable: The system makes expert-level disease detection accessible to farmers, even in remote or underserved areas, improving their productivity and livelihood.
7. Overall Impact: This solution helps farmers make informed decisions, boosts agricultural efficiency, and contributes to a more sustainable and productive farming system.

9.2 Future Scope:

1. Integration with IoT: Use sensors and smart tools for continuous crop health monitoring.
2. Broader Disease and Plant Coverage: Expand detection to more plant diseases and species.
3. Offline Functionality: Enable disease detection without internet access for remote areas.
4. AI and Machine Learning Improvements: Enhance accuracy and provide personalized treatment recommendations.
5. Marketplace Integration: Connect with agricultural marketplaces for easy access to disease treatments.
6. Global Expansion: Localize the system for various languages, regions, and agricultural practices.
7. Affordable for Small Farmers: Ensure the system is cost-effective for small-scale farmers.
8. Government and Agricultural Program Collaboration: Partner with governments and programs for large-scale disease monitoring and management.

REFERENCES

[1] “A Systematic Literature Review on Plant Disease Detection”, Wasswa Shafeek et al., 2023

[2] “Plant leaf disease detection using informal learning and explainable AI”, Ammar Oad, Syid Shoaid, Abbar, Zafar Beenish,2024

[3] “Pathogen-based classification of plant diseases”, Safedine Kadry,2023

[4] “Detection of 26 diseases on 14 crop species”, Mohanty et al.,2023

[5] “Plant leaf disease detection classification and diagnosis using computer vision and artificial intelligence”, Anuja Bhargara, Muhammad H Alshaif,2024

[6] “A Comprehensive Approach Towards Wheat Leaf Disease Identification Leveraging Transformer Models & Federated Learning”,Md Fahim-ul-Islam,,Amitabha Chakrabarthy,2024

[7] “Real-time plant diseases dataset development and detection on plant disease” Diana Susan Joseph, 2024

[8] “Learning for plant disease detection”,Om Prakash Go Swami, Feng Dong,2023