KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

(APPROVED BY AICTE, NEW DELHI)

Department of Computer Science & Engineering

A Project Report on

SMART DISEASES DETECTION AND FIELD MONITORING SYSTEM

Submitted in the partial fulfillment for the award of the degree of

Bachelor of Engineering
in
Computer Science & Engineering

Submitted by

Irfan Hussian Lone	**2GI22CS401**

Guide
Dr. Prasad Pujar
Associate Professor, Department of CSE

2024-2025

KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

(APPROVED BY AICTE, NEW DELHI)

Department of Computer Science & Engineering

CERTIFICATE

Certified that the project entitled **"SMART DISEASES DETECTION AND FIELD MONITORING SYSTEM"** carried out by, **Irfan Hussian Lone(2GI22CS401)**, students of KLS Gogte Institute of Technology, Belagavi, can be considered as a bonafide work for partial fulfillment for the award of **Bachelor of Engineering** in Computer science and engineering of the Visvesvaraya Technological University, Belagavi during the year 2024-2025. It is certified that all corrections/suggestions indicated have been incorporated in the report. The project report has been approved as it satisfies the academic requirements prescribed for the said Degree.

Guide	HOD	Principal
Dr. Prasad Pujar	**Dr. Sanjeev S. Sannakki**	**Dr. M. S. Patil**

Final Viva-Voce

	Name of the examiners	Date of Viva -voce	Signature
1.			
2.			

DECLARATION BY THE STUDENT

I , **Irfan Hussian Lone (2GI22CS401)**, hereby declare that the project report entitled "**SMART DISEASES DETECTION AND FIELD MONITORING SYSTEM**" submitted by us to KLS Gogte Institute of Technology, Belagavi, in partial fulfillment of the Degree of **Bachelor of Engineering** in **Computer Science and Engineering** is a record of the project carried out at Gogte Institute of Technology. This report is for the academic purpose.

We further declare that the report has not been submitted and will not be submitted, either in part or full, to any other institution and University for the award of any diploma or degree.

Name of the student	**USN**	**Signature**
Irfan Hussian Lone	2GI22CS401	

Place: Belagavi

Date:

ACKNOWLEDGEMENT

I would like to express our deepest gratitude to our esteemed project guide, **Dr. Prasad Pujar** for their invaluable guidance and exceptional mentorship throughout the duration of our final year project.

We would like to thank all the people that worked along with us for the project for their patience and openness in creating an enjoyable working environment.

We would like to express sincere thanks to **Dr. M. S. Patil**, Principal, G.IT. Belagavi for his warm support throughout the B.E. program.

We are extremely thankful to **Dr. S. S. Sannakki**, Professor & Head Dept of CSE, G.I.T. Belagavi for her constant co-operation and support throughout this project

We would like to thank our mentor **Dr. Prasad Pujar** for his constant support and guidance throughout our project tenure. He has provided us with incessant support and has been a constant source of inspiration throughout the project.

We thank all our family members, friends, and all the Teaching, Non-Teaching, and technical staff of the Computer Science and Engineering Department, KLS. Gogte Institute of Technology, Belagavi for their invaluable support and guidance.

Sincerely,

Irfan Hussian Lone 2GI22CS401

ABSTRACT

In the face of rising global concerns about food security and agricultural productivity, particularly with the challenges posed by climate change and pest outbreaks, there is a growing need for efficient and smart farming techniques. This research aims to develop a **Smart Disease Detection and Field Monitoring System** using IoT (Internet of Things) sensors, coupled with AI/ML (Artificial Intelligence and Machine Learning) algorithms, to enhance plant health management and optimize field monitoring. The system integrates multiple sensors for temperature, humidity, and soil moisture, which are crucial for determining crop health. Additionally, AI-based disease detection models are employed to recognize plant diseases from images of leaves and other plant parts.

The core components of this system include the use of an ESP8266 microcontroller that connects various sensors to the internet, enabling real-time data transmission to a Firebase cloud database. The collected data is then used to train an AI/ML model, which helps identify potential diseases and anomalies in the plants' health. This model processes visual data from cameras, analyzes the environmental factors such as temperature, humidity, and soil moisture, and provides actionable insights into plant health, guiding the appropriate interventions such as irrigation or pesticide application.

The disease detection algorithm employed utilizes a Convolutional Neural Network (CNN), which has been trained to recognize patterns associated with specific plant diseases, ensuring accurate and early detection. The system also incorporates a dashboard interface, built using Flask or Django, where farmers and users can monitor real-time data from the field, track historical trends, and receive recommendations for managing plant health.

TABLE OF CONTENTS

		Content	Page No.
	i.	Declaration	i
	ii.	Acknowledgement	ii
	iii.	Abstract	iii
	iv.	Table of contents	iv
	v.	List of Tables	v
	vi.	List of Figures	vi
	viii	List of Symbols	viii
1.		**Chapter 1: Introduction**	**1**
	1.1	Introduction	1
	1.2	Objectives	1
	1.3	Methodology	2
		1.3.1 Problem Understanding and Requirement Analysis	2
		1.3.2 Hardware System Development	3
		1.3.3 Data Acquistion and Processing	3
		1.3.4 Cloud Integration	3
		1.3.5 Machine Learning Model for Diseases Detection	3
	1.4	Summary	3
2.		**Chapter 2: Literature Review**	**5**
	2.1	Literature	5
3.		**Chapter 3: System Analysis**	10
	3.1	Problem Identification	10
	3.2	System Requirement	10
		3.2.1 Software Requirement	10
		3.3.2 Functional Requirement	11
		3.2.2 Non-Functional Requirement	12
	3.3	Feasibility Study	12
		3.3.1 Technical Feasibility	12
		3.3.2 Economic Feasibility	12
		3.3.3 Operational Feasibility	12
	3.4	System Architecture	12

	3.5	Summary	14
4		**Chapter 4: System Design**	**15**
	4.1	System Architecture Overview	15
	4.2	Diseases Detection Using Ai/ML	16
		4.2.1Key Components of the Disease Detection Algorithm	18
	4.3	Use Case Diagram	20
	4.4	Hardware Requirement	22
	4.5	Software Requirement	23
5		**Chapter 5: Implementation**	24
	5.1	Smart Diseases Detection Using AI/ML	24
		5.1.1 Overview	24
		5.12 Requirements	24
		5.13 Step-by-steps Implementation	24
	5.2	Yield Monitoring Using Environmental Sensors	28
		5.2.1 Overview	28
		5.2.2 Requirements	28
		5.2.3 Step-by-steps Implementation	28
	5.3	Setting Up Firebase Libraries	30
	5.4	Add Firebase Credentials in Arduino IDE	30
	5.5	Set Up AI/ML Disease Detection System	31
		5.5.1 Install Python and Libraries	31
		5.5.2 Collect and Preprocess Data	31
		5.5.3 Train a CNN Model	31
		5.5.4 Disease Detection Using the Model	32
	5.6	Web Interface Setup	32
		5.6.1 Set Up a Simple Web Server	32
		5.6.2 Connect Web Interface to Firebase	32
6		**Chapter 6: Testing**	**33**
	6.1	Unit Testing	33
	6.2	Integration Testing	33
	6.3	System Testing	34

	6.4	Performance Testing	34
	6.5	Acceptance testing	34
7		**Chapter 7: Results**	**38**
	7.1	Results	38
8		**Chapter 8: Application And Conclusion**	**44**
	8.1	Applications	44
	8.2	Conclusion	45
	8.3	Future Scope	45
		References	47

LIST OF FIGURES

S. No	Fig. No	Figure	Pg. No
1	3.4.1	Soil Moisture Sensor	13
2	3.4.2	Humidity Sensor	13
3	3.4.3	Motor Pump	13
4	3.4.4	Node-MCU Diagram	13
5	3.4.5	Node MCU	13
6	3.4.6	Flow Chart	14
7	4.1	Circuit Diagram	15
8	4.2	Diseases Detection Working	18
9	4.2.1	Overall Model Architecture	19
10	4.3	Use case Diagram	20
11	5.1.3	CNN Architecture	25
12	5.2	Firebase Connection	28
13	7.1	Home Page	38
14	7.2	About Page	39
15	7.3	Supplements Page	40
16	7.4	Login Page	41
17	7.5	Forgot Password Page	41
18	7.5	Register Page	42
19	7.7	Dashboard Page	42
20	7.8	AI Engine	43
21	7.9	IOT Setup	43

LIST OF TABLES

Table No	Figure	Page No
6.1.1	Test Case 1 For CNN Model Disease Detection Accuracy.	35
6.1.2	Test Case 2 For IoT Sensor Data Transmission.	35
6.1.3	Test Case 3 User Authentication Functionality.	36
6.1.4	Test Case 4 Real-Time Monitoring Graph Display.	36
6.1.5	Test Case 5 End-to-End System Functionality	37
6.1.6	Test Case 6 End-to-End System Functionality	37

CHAPTER 1 INTRODUCTION

1.1 Introduction

Agriculture is a vital sector that sustains life and supports economies worldwide. It plays an indispensable role in ensuring food security, generating employment, and contributing to the GDP of many nations. However, the agricultural industry faces numerous challenges, including declining crop yields, increasing pest and disease outbreaks, and resource mismanagement. Among these, plant diseases remain one of the most significant threats, responsible for substantial yield losses and economic setbacks globally. These issues are further compounded by the unpredictable effects of climate change, which exacerbate the spread and intensity of diseases, making traditional agricultural practices less effective. Farmers, particularly those in small and medium-scale operations, struggle to identify diseases promptly, leading to delayed intervention, reduced productivity, and, in some cases, total crop failure.

The need for innovative, data-driven solutions to tackle these issues has never been more urgent. In this context, the convergence of modern technologies, such as the Internet of Things (IoT), Artificial Intelligence (AI), and cloud computing, has opened new possibilities for transforming traditional farming into a more efficient, intelligent, and sustainable practice. IoT enables real-time monitoring of critical environmental parameters like temperature, humidity, and soil moisture, providing continuous insights into field conditions. AI, on the other hand, brings precision and efficiency by leveraging machine learning algorithms to analyze plant images and accurately detect diseases at an early stage. Combined, these technologies provide a holistic approach to addressing the dual challenges of disease detection and field monitoring.

The proposed **Smart Disease Detection and Field Monitoring System** seeks to integrate IoT-based environmental sensing with AI-driven disease diagnosis to empower farmers with real-time, actionable insights. By deploying sensors in the field, the system continuously monitors key environmental conditions, ensuring that critical thresholds are identified and communicated promptly. Simultaneously, a machine learning-based disease detection module analyzes images of affected crops to identify diseases with high accuracy, enabling early intervention and minimizing losses

1.2 Objectives

- **Real-Time Monitoring:** Develop a system to continuously monitor environmental parameters such as temperature, humidity, and soil moisture using IoT sensors.
- **Early Disease Detection:** Implement a machine learning model to accurately identify plant diseases from images, enabling early intervention and reducing crop loss.

- **Automated Irrigation:** Integrate a soil moisture sensor with an automated water pump system to optimize irrigation and conserve water resources.
- **User-Friendly Interface:** Create an intuitive web or mobile dashboard for real-time data visualization, disease alerts, and system control.
- **Cloud-Based Data Management:** Utilize a cloud platform like Firebase for real-time data storage, retrieval, and synchronization across devices.
- **Affordable and Scalable Solution:** Design a cost-effective system that can be scaled for small to medium-sized farms, making it accessible to a wide range of users.
- **Sustainability and Efficiency:** Promote resource efficiency and sustainable farming practices through data-driven decision-making.
- **Integration and Adaptability:** Ensure the system can be adapted for various crops, regions, and environmental conditions.

1.3 Methodology: -

1.3.1 Problem Understanding and Requirement Analysis: -

- Conduct surveys or interviews with farmers to identify key challenges, such as:
  - Common plant diseases in the region.
  - Environmental parameters critical for crop health (e.g., temperature, humidity, soil moisture).
  - Challenges in irrigation and water management.
- Define functional and non-functional requirements, such as:
  - Real-time monitoring and control.
  - Scalability for small and large-scale operations.
  - Affordability for small farmers.

1.3.2. Hardware System Development: -

- ➢ Selection of Components:
  - Microcontroller/Processor: Use ESP8266 or ESP32 for IoT connectivity.
  - Sensors:
  - DHT11/DHT22: Measure temperature and humidity.
  - Soil moisture sensor: Monitor soil water content.

- Optional sensors: Light intensity (for photosynthesis monitoring), CO2 sensors (for greenhouse systems).
- Water Pump and Relay Module: For automated irrigation control.

➢ Integration and Circuit Design**:**

- Connect sensors and actuators to the ESP8266 microcontroller.
- Use proper pull-up resistors, voltage regulators, and connectors to ensure stability.
- Test communication protocols such as I2C, UART, or SPI for sensor interfacing.

1.3.3 Data Acquisition and Processing:

- Write firmware for ESP8266 using C++ (Arduino IDE or Platform IO).
- Collect data from sensors at regular intervals.
- Process raw sensor readings (e.g., convert soil sensor ADC values to percentage moisture).
- Implement threshold checks for real-time alerts (e.g., low soil moisture triggering pump activation).

1.3.4 Cloud Integration: -

- Platform Selection: Use Firebase for its real-time database, scalability, and compatibility with IoT systems.
- Data Flow:
  - Upload sensor data to Firebase in JSON format using REST APIs or Firebase SDK.
  - Enable real-time synchronization between devices (microcontroller and user dashboard).
- Storage and Retrieval:
  - Store historical data for environmental conditions and disease alerts.
  - Enable analysis of trends for better decision-making.

1.3.4 Machine Learning Model for Disease Detection:

- Model Selection: Use Convolutional Neural Networks (CNNs) for image-based disease classification.
- Dataset: Collect or use publicly available datasets like Plant Village.
- Preprocessing: Resize, augment, and normalize images for better training.
- Training: Train the model using frameworks like TensorFlow or PyTorch.
- Export the trained model to a lightweight format (e.g., TensorFlow Lite) for deployment.
- Deployment**: -** Host the disease detection model on a server or integrate it into the user dashboard for on-demand disease analysis.

1.4 Summary

The proposed ***Smart Disease Detection and Field Monitoring System*** integrates IoT-based real-time environmental monitoring with AI-driven disease detection to address critical challenges in agriculture. By leveraging sensors for temperature, humidity, and soil moisture, the system provides actionable insights to optimize irrigation and resource usage. A machine learning model accurately identifies plant diseases from crop images, enabling early intervention and minimizing crop losses. The data is seamlessly managed using cloud platforms like Firebase, ensuring real-time storage, retrieval, and synchronization across devices.

CHAPTER 2 LITERATURE REVIEW

The advancements in technology, particularly in IoT and machine learning, have enabled significant developments in smart agriculture. Several studies have explored the integration of IoT sensors and cloud platforms to monitor environmental parameters such as soil moisture, temperature, and humidity. These systems provide real-time insights to optimize irrigation and improve resource efficiency. For instance, research has shown that soil moisture sensors, combined with automated irrigation systems, can reduce water wastage by up to 40%.

In the domain of plant disease detection, machine learning, particularly Convolutional Neural Networks (CNNs), has demonstrated high accuracy in diagnosing crop diseases. Studies utilizing datasets like Plant Village have achieved classification accuracies exceeding 90% in identifying diseases in crops such as tomato, potato, and wheat. These systems have proven to be faster and more reliable than traditional manual inspection methods, which are often subjective and time-consuming.

Paper 1:

Ttile: Automatic plant pest detection & recognition using k-means clustering algorithm & correspondence filters

Summary

Plant pest recognition and detection is vital for food security, quality of life and a stable agricultural economy. This research demonstrates the combination of the k-means clustering algorithm and the correspondence filter to achieve pest detection and recognition. The detection of the dataset is achieved by partitioning the data space into Voronoi cells, which tends to find clusters of comparable spatial extents, thereby separating the objects (pests) from the background (pest habitat). The detection is established by extracting the variant distinctive attributes between the pest and its habitat (leaf, stem) and using the correspondence filter to identify the plant pests to obtain correlation peak values for different datasets. It is encouraging to note that the correspondence filter can achieve rotational invariance of pests up to angles of 360 degrees, which proves the effectiveness of the algorithm for the detection and recognition of plant pests.[1]

Paper 2:

Title: Plant Leaf Disease Detection using Inception Resnet V2 -CNN

Summary India is a agriculture country, most of the people are farmers. Still farmers are not able to increase their income, productivity due to different types of diseases in plant. It has negative impact on farming. Now a days number of diseases are increased, so identifying the name of disease is one of the challenges for farmers. If plant diseases are not discovered in early stage, then this can harm crop in large extent, so we need to create a system which can easily identify the name of disease. And also, in India most of the farmers are not educated so we also required to create a simple user interface either by using web development or by using Android app development. Usually, plant's leaf is primary source for identifying the name of the disease, so we required to create a CNN model which can easily identify the name of the disease by scanning the photo of leaf. If farmers are able to identify the disease in the early stage, they can take required action and loss of production can be reduced [2].

Paper 3:

Title: An Overview of the Research on Plant Leaves Disease detection using Image Processing Techniques

Summary

Diseases in plants cause major production and economic losses as well as reduction in both quality and quantity of agricultural products. Now a day's plant diseases detection has received increasing attention in monitoring large field of crops. Farmers experience great difficulties in switching from one disease control policy to another. The naked eye observation of experts is the traditional approach adopted in practice for detection and identification of plant diseases. In this paper we review the need of simple plant leaves disease detection system that would facilitate advancements in agriculture. Early information on crop health and disease detection can facilitate the control of diseases through proper management strategies. This technique will improve productivity of crops [3].

Paper 4:

Title: Apple Scab and Marsonina Coronaria Diseases Detection in Apple Leaves Using Machine Learning

Summary:

Agriculture plays an indispensable role in the development of the country especially in the growing country like India where most of the peoples' revenue is generated from agriculture. Disease affected crops leads to the loss of crop productivity. Therefore, leaf disease prediction in apple cultivation is of considerable importance to overcome these problems. The proposed work intends to predict apple scab and marsonina coronaria apple leave diseases using four different classification algorithms i.e. support vector machine (SVM), K nearest neighbor (KNN), classification decision and regression tree and Naive Bayes. From the simulation results, it was shown that KNN gives an accuracy of 99.4% in classifying apple scab and marsonina coronaria apple leave diseases as compared to the other classifiers [4].

Paper 5:

Ttile: An Introduction to Convolutional Neural Networks

Summary:

The field of machine learning has taken a dramatic twist in recent times, with the rise of the Artificial Neural Network (ANN). These biologically inspired computational models are able to far exceed the performance of previous forms of artificial intelligence in common machine learning tasks. One of the most impressive forms of ANN architecture is that of the Convolutional Neural Network (CNN). CNNs are primarily used to solve difficult image-driven pattern recognition tasks and with their precise yet simple architecture, offers a simplified method of getting started with ANNs. This document provides a brief introduction to CNNs, discussing recently published papers and newly formed techniques in developing these brilliantly fantastic image recognition models. This introduction assumes you are familiar with the fundamentals of ANNs and machine learning [5].

Paper 6:

Title: A REVIEW OF STUDIES ON MACHINE LEARNING TECHNIQUES

Summary:

This paper provides an extensive review of studies related to expert estimation of software development using Machine-Learning Techniques (MLT). Machine learning in this new era, is demonstrating the promise of producing consistently accurate estimates. Machine learning system effectively how to estimate from training set of completed projects. The main goal and contribution of the review is to support the research on expert estimation, i.e. to ease other researchers for relevant expert estimation studies using machine-learning techniques. This paper presents the most commonly used machine learning techniques such as neural networks, case-based reasoning, classification and regression trees, rule induction, genetic algorithm & genetic programming for expert estimation in the field of software development. In each of our study we found that the results of various machine-learning techniques depend on application areas on which they are applied. Our review of study not only suggests that these techniques are competitive with traditional estimators on one data set, but also illustrate that these methods are sensitive to the data on which they are trained [6].

Paper 7:

Title: IoT based temperature and humidity monitoring framework

Summary:

This study explored the use of Internet of Things (IoT) in monitoring the temperature and humidity of a data center in real-time using a simple monitoring system to determine the relationship and difference between temperature and humidity with respect to the different locations of measurements. The development of temperature and humidity monitoring system was accomplished using the proposed framework and has been deployed at the data center of Politeknik Muadzam Shah, where the readings were recorded and sent to an IoT platform of ATT M2X to be stored. The data was then retrieved and analyzed showing that there was a significant difference in temperature and humidity measured at different locations. X The monitoring system was also successful in detecting extreme changes in temperature and humidity and automatically send a notification to IT personnel via e-mail, short messaging service (SMS) and mobile push notification for further action [7].

Paper 8:

Ttile: IoT Based Moisture Control and Temperature Monitoring in Smart Farming

Summary: The Internet of Things (IoT) has made a revolution in all the fields of human life by making the work be smart and effective. The IoT devices like sensors, controller, Wi-Fi module and the cloud play a significant part in smart farming which acquires yield in the field of farming and lessens the wastage. The goal of this paper is to propose the IoT based framework for the farmers by analyzing the live information like (moisture, temperature) in the cloud. The agrarian device is equipped with Arduino innovation and can be received through web servers with different sensors and live information transmissions through Thingsspeak.com. The smart agriculture stick is proposed through this paper which is integrated with controller, sensor and live data that can be monitored through the cloud [8].

Paper 9:

Ttile: Traditional and current-prospective methods of agricultural plant diseases detection: A review

Summary:

As it is known, a significant part of the yield of agricultural crops is lost due to harmful organisms, including diseases. The article reveals the data on the widespread types of plant diseases (rot, wilting, deformation, the formation of tumors, pustules, etc.) and their symptoms. Early identification of the pathogen type of plant infection is of high significance for disease control. Various methods are used to diagnose pathogens of disease on plant. This article discusses the review of the literature data on traditional methods for diagnosis of plant pathogens, such as visual observation, microscopy, mycological analysis, and biological diagnostics or the use of indicator plants. Rapid and reliable detection of plant disease and identification of its pathogen is the first and most important stage in disease control. Early identification of the cause of the disease allows timely selection of the proper protection method and ensures prevention of crop losses. There are a number of traditional methods for identifying plant diseases, however, in order to ensure the promptness and reliability of diagnostics, as well as to eliminate the shortcomings inherent in traditional diagnostics, in recent years, new means and technologies for identifying pathogens have been developed and introduced into practice. As well as the article provides information on such innovative methods of diagnosis of diseases and identification of their pathogens, which are used widely in the world today, such as immunodiagnostics, molecular-genetic (and phylogenetic) identification, mass spectrometry, etc [9].

Paper 10:

Ttile: Agriculture monitoring system: A study

Summary:

This study is a review on controlling an electronic device (Arduino) apply for temperature and soil moisture process using Android based Smart phone application in order to address the issues of flexibility and functionality. Besides, this study in future will also develop a low cost and flexible for agriculture control due to not to incorporate with an expensive components such as high end personal computers. On peak of that, now anyone, from anytime and anywhere can have connectivity for anything and it is expected that these connections will extend and create an entirely advanced dynamic network of the internet of things. Thus, this study is to review several designs of smart monitoring system using an embedded micro-web server, with IP connectivity for accessing. There are three principal components in this study, which are an electronic device (Arduino), software development (eclipse), and system prototype internet protocol layer. The aim is to build the web organization and ultimately to combine all three components together [10].

CHAPTER 3 SYSTEM ANALYSIS

The **Smart Disease Detection and Field Monitoring System** is designed to address critical challenges in agriculture, such as early detection of plant diseases, efficient water management, and real-time monitoring of environmental parameters. The system's analysis is structured into problem identification, system requirements, and feasibility study.

3.1. Problem Identification

Agricultural productivity is often hampered by:

- **Plant Diseases**: Delayed or inaccurate identification of plant diseases leads to reduced yields and economic losses.
- **Inefficient Irrigation**: Over-irrigation or under-irrigation due to lack of real-time data results in poor crop health and resource wastage.
- **Manual Efforts**: Traditional methods of field monitoring are labor-intensive, time-consuming, and error-prone.
- **Data Accessibility**: Farmers lack actionable insights due to limited access to real-time and historical environmental data.

3.2 System Requirements

3.2.1 Software Requirement: -

The software components of the **Smart Disease Detection and Field Monitoring System** include tools and platforms for data collection, processing, visualization, and machine learning integration. Below is a brief overview:

1. Programming and Development Tools

- Arduino IDE: For writing and uploading the firmware to the ESP8266 microcontroller.
- Python: For machine learning model development and backend processing.
- React.js or Flutter: For building an interactive web or mobile dashboard for user interaction.

2. Cloud Platform

- Firebase:
  - Realtime Database: For storing and synchronizing sensor data and user actions.
  - Firebase Hosting: To deploy the web application for real-time accessibility.
  - Authentication: For user login and secure access.

3. Machine Learning Framework

- TensorFlow or PyTorch: For training and deploying a Convolutional Neural Network (CNN) model for plant disease detection.
- TensorFlow Lite: For deploying the lightweight model on the IoT platform or web interface.

4. Data Visualization Libraries

- Chart.js or D3.js: For creating real-time and historical data graphs on the dashboard.
- Gauge.js: For visualizing sensor values like soil moisture and humidity as interactive gauges.

5. Communication Protocols and APIs

- HTTP/REST APIs: For exchanging data between IoT devices and Firebase.
- MQTT Protocol: Optional for lightweight and efficient IoT communication.

6. Operating System and Dependencies

- Windows/Linux/MacOS: For development and deployment.
- Required libraries and dependencies for machine learning (e.g., NumPy, OpenCV, Scikit-learn) and IoT integration (e.g., Firebase SDK)

3.2.2 Functional Requirements

1. **IoT-Based Monitoring**:
   - Sensors for temperature, humidity, and soil moisture monitoring.
   - Automated data collection and real-time updates.
2. **Disease Detection**:
   - AI model for identifying diseases from crop images with high accuracy.
   - User-uploaded image processing for on-demand analysis.
3. **Water Pump Automation**:
   - Soil moisture-based control of irrigation systems.
   - Manual override through a web or mobile application.

4. **User Interface**:
   - Real-time data visualization using graphs and gauges.
   - Alerts for critical conditions and disease outbreaks.
   - Secure login and personalized dashboards.

3.2.2 Non-Functional Requirements

- Scalability: Support for different farm sizes and crops.
- Affordability: Cost-effective for small-scale farmers.
- Reliability: Accurate data readings and consistent performance under varying conditions.
- Ease of Use: Intuitive interface for users with minimal technical expertise.

3. 3 Feasibility Study

3.3.1 Technical Feasibility

- IoT Devices: Availability of affordable microcontrollers like ESP8266 and sensors (DHT22, soil moisture sensors).
- Cloud Integration: Firebase provides scalable and real-time database solutions.
- AI Models: Machine learning frameworks such as TensorFlow and PyTorch enable robust disease detection systems.

3.3.2 Economic Feasibility

- Hardware and software components are affordable and readily available, making the system cost-effective for small and medium-sized farmers.

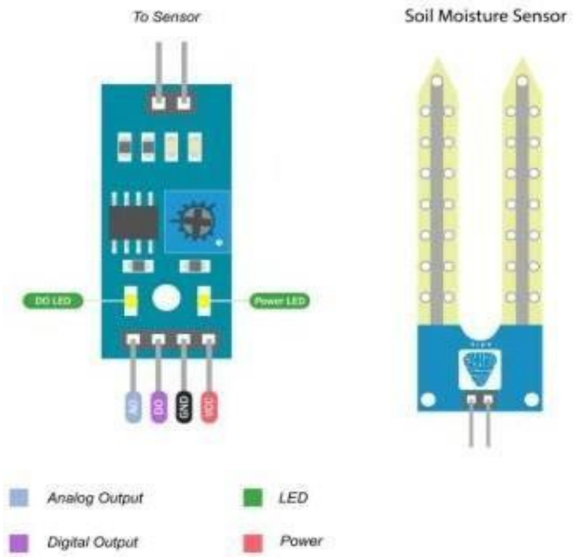
3.3.3 Operational Feasibility

- Automation reduces manual labor, while real-time data visualization empowers farmers to make timely decisions, ensuring operational efficiency.

3.4 System Architecture: -

The system architecture includes the following key components:

- **IoT Sensor Nodes**: Collect environmental data (e.g., temperature, humidity, soil moisture, water pump) from the field.



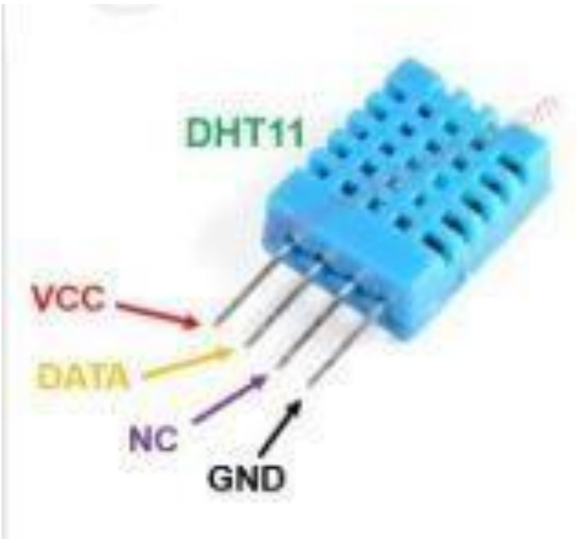


Figure No :3.4.1 Soil Moisture Sensor **Figure No :3.4.2 Humidity Sensor**

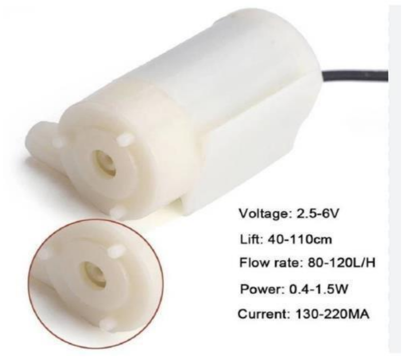


Figure No :3.4.3 Motor Pump

- **Microcontroller**: The ESP8266 or ESP32 processes data from the sensors and communicates with the cloud platform.

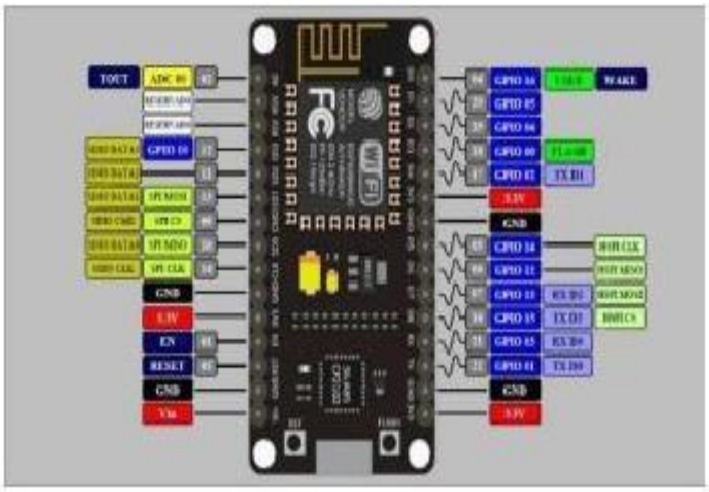


Figure No:3.4.4 NodeMCU Pin Diagram **Figure No:3.4.5 NodeMCU**

- **Cloud Platform**: Firebase serves as the backend, providing real-time data storage, synchronization, and user authentication. The cloud stores historical data and provides scalability for multiple users and devices.
- **Machine Learning Module**: A CNN-based disease detection model processes plant images uploaded by users to identify plant diseases with high accuracy.
- **User Interface**: A web or mobile-based dashboard displays real-time data, alerts, and recommendations. The dashboard allows farmers to monitor environmental conditions, detect diseases, control irrigation systems, and analyze historical data.

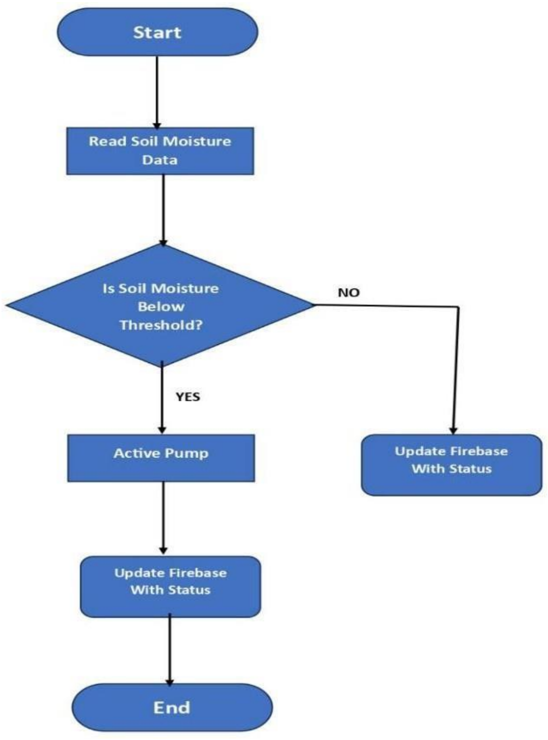

Start
Read Soil Moisture Data
Is Soil Moisture Below Threshold?
NO
YES
Active Pump
Update Firebase With Status
Update Firebase With Status
End

Fig No 3.4 Flow Chart

3.5 Summary: -

The **Smart Disease Detection and Field Monitoring System** presents an innovative solution to address key challenges in modern agriculture, such as disease management, water conservation, and real-time field monitoring. By combining IoT sensors, cloud computing, and machine learning, the system can provide real-time insights and automate critical tasks like irrigation, ultimately improving productivity and sustainability. The system is designed to be scalable, affordable, and user-friendly, making it accessible to small and medium-scale farmers.

CHAPTER 4 SYSTEM DESIGN

4.1 System Architecture Overview

The overall architecture of the system can be divided into four main component

1. **IoT Sensor Network**
2. **Microcontroller (ESP8266)**
3. **Cloud Platform (Firebase)**

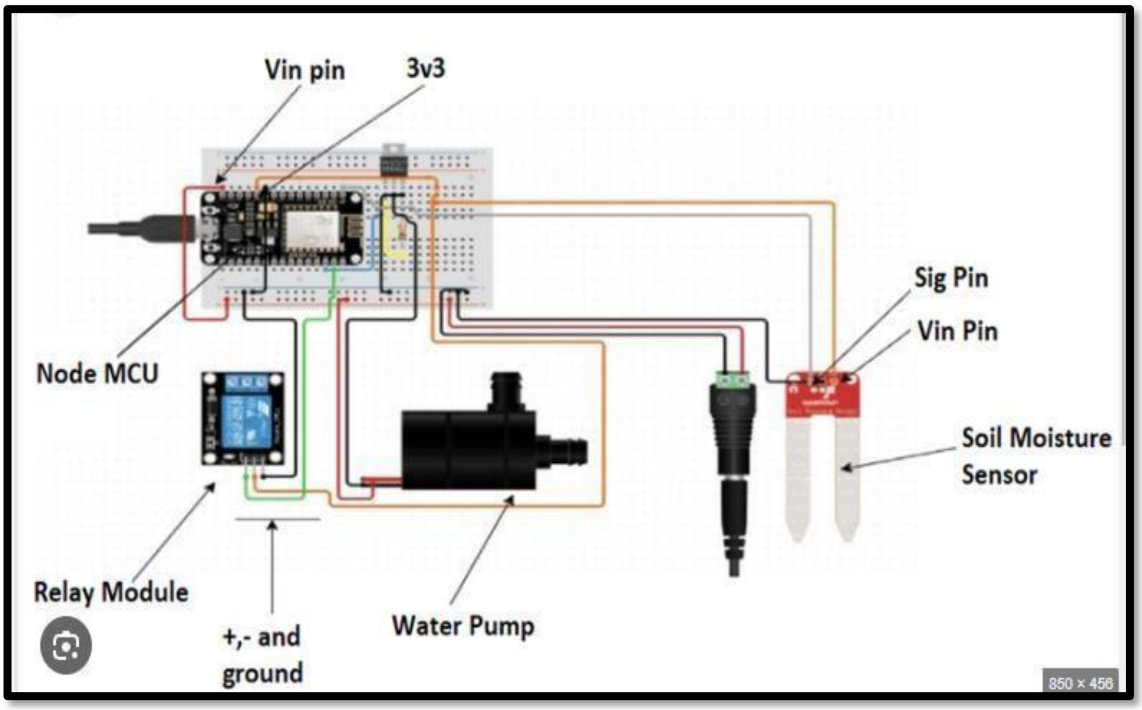


Fig No 4.1 Circuit Diagram

1. IoT Sensor Network

The IoT sensor network is the core of the system, collecting real-time environmental data necessary for monitoring the health of crops. The following sensors are part of the network:

- **Soil Moisture Sensor**: Measures the water content in the soil, enabling automated irrigation control.
- **Temperature and Humidity Sensors (e.g., DHT11, DHT22)**: Measure environmental conditions (temperature and humidity), which affect plant health and disease spread.

2. Microcontroller (ESP8266)

The **ESP8266/ESP32** microcontroller acts as the central unit for data acquisition and processing. It performs the following functions:

- **Data Acquisition**: The microcontroller collects data from the connected sensors (soil moisture, temperature, humidity).
- **Data Processing**: The microcontroller processes the sensor data (e.g., calculating average values, checking thresholds for soil moisture).
- **Communication**: It sends the processed data to the cloud platform (Firebase) over Wi-Fi.

3. Cloud Platform (Firebase)

Firebase serves as the backend for data storage and real-time synchronization. It handles:

- **Real-Time Database**: Stores sensor data and user actions (such as controlling irrigation).
- **User Authentication**: Ensures secure access to the system, with unique logins for different users.
- **Machine Learning Integration**: The cloud platform hosts and runs the disease detection model (e.g., TensorFlow or TensorFlow Lite) that processes user-uploaded plant images.

4.2 Disease Detection Using AI/ML: -

The disease detection workflow involves several stages, from image acquisition to classification and output generation The disease detection component of the **Smart Disease Detection and Field Monitoring System** is based on a **Machine Learning (ML) algorithm** designed to accurately classify plant diseases using image data. This algorithm is central to the system, allowing farmers to detect diseases at early stages, reducing the need for manual inspections, and providing actionable insights in real-time. The primary technique used for disease detection is **Convolutional Neural Networks (CNNs)**, a deep learning method particularly suited for image classification tasks.

Below is a detailed explanation of the disease detection algorithm, the steps involved, and the approach used to train and deploy the model.

. The steps involved are as follows:

1. **Image Acquisition**:
    - The system allows the farmer to upload images of plant leaves via a web or mobile interface. These images can be captured using smartphones or digital cameras.
    - The quality and clarity of the image are crucial for accurate disease detection. The image is pre-processed (if necessary) to improve quality before being fed to the model.
2. **Pre-Processing**:
    - **Image Normalization**: To ensure that all images are treated equally, the input images are normalized (resized, color normalized) so that they fit the expected dimensions for the model.
    - **Data Augmentation**: To increase the model's robustness and prevent overfitting, techniques such as rotation, flipping, cropping, and zooming are applied to the images.
3. **Feature Extraction (CNN-based)**:
    - **Convolutional Layers**: The core of the disease detection model is a CNN. These networks are designed to automatically detect and learn relevant features in an image, such as edges, textures, and shapes, without the need for manual feature engineering.
    - **Activation Functions**: After each convolution operation, an activation function (e.g., ReLU) is used to introduce non-linearity, enabling the network to learn more complex patterns.
4. **Model Training**:
    - A CNN model is trained using a large dataset of plant images, labeled with disease categories (e.g., healthy, leaf blight, rust, powdery mildew). The dataset should be representative of various diseases that affect the crops the farmer is interested in monitoring.
    - During training, the model adjusts the weights of the convolutional layers to minimize the error between the predicted and actual disease labels.
5. **Classification**:
    - Once the model has been trained, it can classify new, unseen images by passing them through the layers of the CNN. The network predicts the disease or status of the plant (healthy or diseased) based on learned features.
    - The output is typically a probability distribution over the classes (diseases), and the class with the highest probability is chosen as the prediction.
6. **Post-Processing**:
    - **Disease Confidence**: The system displays the disease prediction with the confidence score, indicating how confident the model is in its classification. If the confidence

score is low, the system may request a re-check or more data to improve prediction accuracy.

- **Visualization**: The results, including the disease name and a confidence score, are shown on the web or mobile interface for easy interpretation by the farmer.

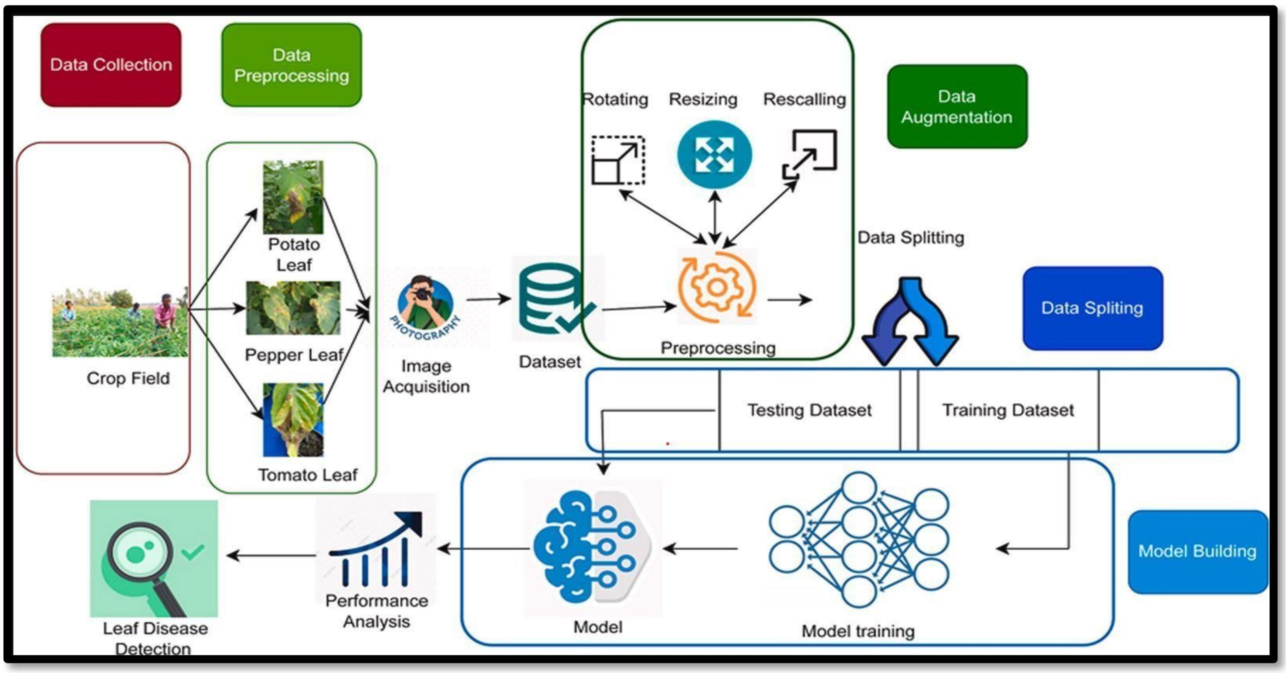


Fig No 4.2 Diseases Detection Working

4.2.1 Key Components of the Disease Detection Algorithm

a. Convolutional Neural Network (CNN)

CNNs are the most common approach for image classification problems. They consist of layers designed to process image data hierarchically:

- **Convolutional Layer**: This layer applies filters (kernels) to the input image to detect simple features (edges, colors, textures). The filter slides over the image, performing a convolution operation at each position.
- **Pooling Layer**: After convolution, the pooling layer reduces the dimensionality of the data (usually using max pooling) to preserve important features while minimizing computation.
- **Fully Connected Layer**: After several convolutional and pooling layers, the resulting features are flattened into a 1D vector and passed through fully connected layers to classify the image into predefined classes.

- **Activation Function**: The ReLU (Rectified Linear Unit) function is commonly used to introduce non-linearity and help the network learn more complex patterns.

b. Data Preprocessing and Augmentation

- **Normalization**: Images are resized to a standard size (e.g., 224x224 pixels) to ensure uniform input dimensions for the CNN.
- **Data Augmentation**: This helps the model generalize better by artificially increasing the size of the training dataset. Random transformations like rotation, scaling, flipping, and zooming are applied to training images.

c. Transfer Learning (Optional)

If training a CNN from scratch is computationally expensive or requires a large dataset, transfer learning can be used. Pretrained models such as ResNet, VGG, or Inception are fine-tuned on the specific plant disease dataset. This technique allows the model to leverage learned features from other domains (e.g., general object recognition) and adapt them to plant disease classification.

d. Model Evaluation Metrics

- Accuracy: The proportion of correctly predicted plant disease categories out of the total predictions.
- Precision: The percentage of correct disease predictions among all positive predictions.
- Recall (Sensitivity): The percentage of correctly predicted diseases among all actual positive cases.

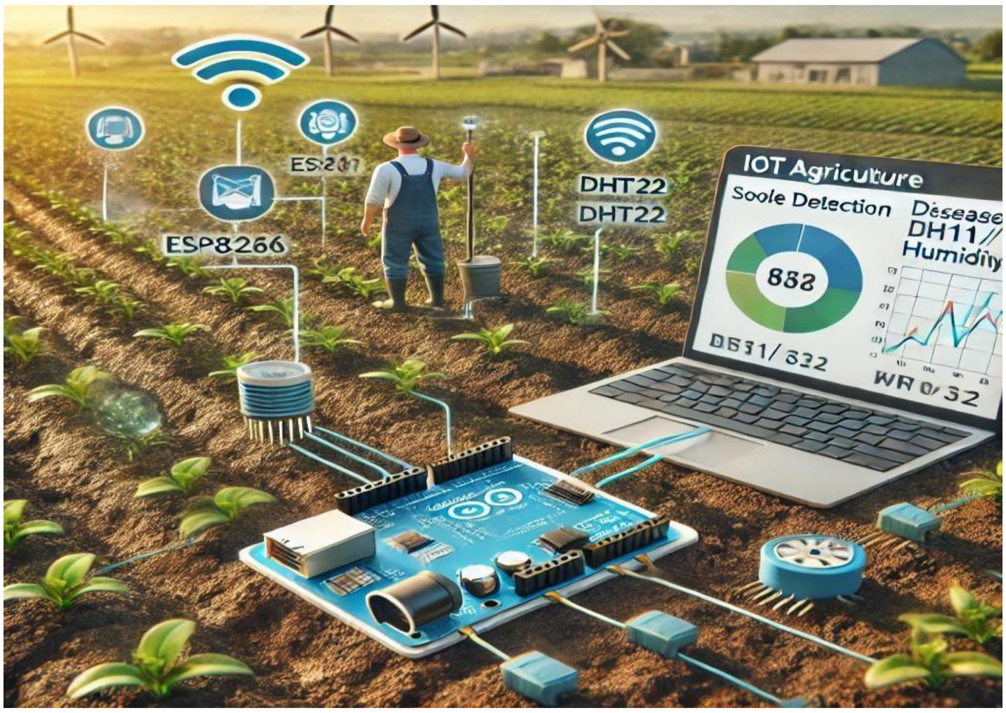


Figure No :4.2.1 Overall Model Architecture

4.3 Use Case Diagram

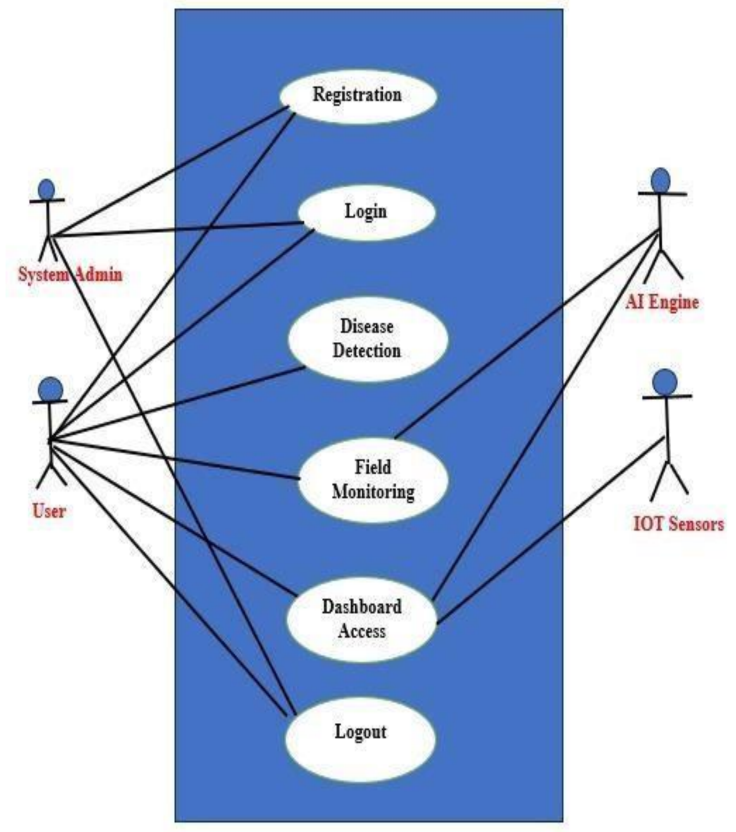


Figure No:4.3 Use case Diagram For Model

1. System Admin

- **Responsibilities**: The System Admin is responsible for managing users, overseeing the operation of the system, and ensuring proper functioning. The admin can access key features such as:
    - **Registration**: Admin can register new users in the system.
    - **Login**: Admin has the ability to log into the system, typically with higher privileges than the regular user.
    - **Disease Detection**: Admin may initiate disease detection processes or manage settings related to disease identification in crops.
    - **Field Monitoring**: Admin can access field monitoring features to analyze data from IoT sensors.

2. User

- **Responsibilities**: Users, which could be farmers or field operators, use the system for disease detection and monitoring crop conditions. Their interactions include:

  - **Registration**: Users can register themselves into the system to get access to the dashboard and monitoring tools.
  - **Login**: Users log into their accounts with their credentials to access personalized features.
  - **Disease Detection**: Users can upload images or sensor data for detecting diseases using AI models.
  - **Field Monitoring**: Users can access real-time data from IoT sensors to monitor the environmental conditions (such as temperature, humidity, and soil moisture) of the field.
  - **Dashboard Access**: Users can view graphical representations of sensor data and disease detection results in a dashboard format.
  - **Logout**: Once done, users can log out of the system for security purposes.

3. AI Engine

- **Responsibilities**: The AI engine plays a central role in processing data and making predictions about plant diseases. It interfaces with the disease detection component to:
  - **Disease Detection**: The AI engine processes the data (e.g., images, environmental readings) to detect and classify plant diseases using machine learning models. It uses various algorithms, such as CNNs (Convolutional Neural Networks), to identify disease patterns from images of plant leaves.
  -

4. IoT Sensors

- **Responsibilities**: IoT sensors are the hardware components responsible for collecting environmental data (temperature, humidity, soil moisture, etc.). These sensors are linked to the system to provide real-time field data. Key responsibilities include:
  - **Field Monitoring**: The sensors continuously monitor field conditions, such as moisture levels, temperature, and humidity, and send this data to the system for analysis.
  - The data from sensors is sent to a cloud or real-time database (such as Firebase), which can then be accessed by users for ongoing monitoring.

Diagram Interaction Details:

- **System Admin → Registration / Login**: Admin can create and manage user accounts.
- **System Admin → Disease Detection / Field Monitoring**: Admin can monitor and manage disease detection processes and field data analysis.

- **User → Disease Detection**: Users can upload images of plant leaves or access sensor data to detect diseases in plants.
- **User → Field Monitoring**: Users can monitor real-time environmental data from IoT sensors to track the health of crops.
- **User → Dashboard Access**: Users can visualize the disease status and environmental metrics on a dashboard interface, making it easy to understand crop health.
- **AI Engine → Disease Detection**: The AI engine analyzes the data from users (whether uploaded images or real-time sensor data) to classify the presence of plant diseases.
- **IoT Sensors → Field Monitoring**: The IoT sensors constantly send data to the system to ensure up-to-date monitoring of field conditions.

4.4 Hardware Requirement: -

Requirement	Details
ESP8266 Wi-Fi Module (NodeMCU)	Microcontroller for sensor integration and Wi-Fi communication to send data to Firebase.
DHT22 Sensor	Measures temperature and humidity levels with high accuracy.
Soil Moisture Sensor	Monitors soil moisture levels to determine irrigation needs.
Relay Module	Controls the water pump based on the moisture levels.
Water Pump	Automates irrigation based on soil moisture sensor readings.
Power Supply	5V USB adapter or power bank to power the ESP8266 module and sensors.
Jumper Wires	Connects the sensors, relay, and ESP8266.
Breadboard	For prototyping and sensor connections.
Plant Samples (Images)	Dataset of healthy and diseased plant leaves for training the AI/ML model.
Personal Computer/Laptop	Required for coding, training the AI/ML model, and setting up the Firebase database.

4.5 Software Requirement: -

Software	**Requirement**	**Details**
Arduino IDE	Arduino development environment for programming the ESP8266.	Install necessary libraries: FirebaseESP8266, DHT Sensor Library, and ESP8266WiFi.
Python	Python environment for training the AI/ML disease detection model.	Required libraries: TensorFlow, Keras, NumPy, OpenCV, and Matplotlib.
Firebase	Google Firebase Realtime Database	Used for storing sensor data and enabling communication between the ESP8266 and web interface.
Flask/Django	Web framework (Python-based) for creating a backend server for disease detection and Firebase integration.	Hosts the AI/ML model for plant disease predictions.
HTML/CSS/JavaScript	Technologies for building the front-end web interface.	Creates a dashboard for displaying real-time sensor data, historical trends, and control functionalities.

CHAPTER 5: - IMPLEMENTATION

This project integrates AI/ML-based Disease Detection **and** IoT-based Yield Monitoring using environmental sensors such as soil moisture, temperature, and humidity. The ESP8266 microcontroller collects sensor data and sends it to Firebase, a cloud platform that stores the data and allows real-time updates. The data is displayed on a web interface for monitoring and controlling irrigation systems.

5.1. Smart Disease Detection Using AI/ML

5.1.1 Overview

The disease detection system is built using machine learning algorithms, particularly Convolutional Neural Networks (CNNs). The system detects plant diseases by classifying plant images into categories such as "healthy" and various disease types (e.g., Leaf Blight, Powdery Mildew, Rust, etc.).

5.1.2 Requirements

- **Hardware**:
  - Camera (smartphone or webcam) to capture images of plant leaves.
  - ESP8266 Wi-Fi module for communication between the device and the cloud.
  - Firebase for cloud storage of data and images.
- **Software**:
  - Python Libraries: TensorFlow, Keras for deep learning model training, OpenCV for image processing.
  - Web Technologies: HTML, CSS, Django for the frontend.
  - Firebase SDK: For real-time database and image storage.

5.1.3 Step-by-Step Implementation

➢ **Data Collection**

- The dataset should consist of labeled images of healthy plants and various disease types. You can use datasets from platforms like **Kaggle** or **Plant Village**, or you can manually collect and label images.

➢ Data Preprocessing

- **Image resizing**: Images should be resized to a consistent size (224x224 pixels) to feed into the model.
- **Data Augmentation**: Apply random rotations, flips, and zooming to the images to improve the robustness of the model and reduce overfitting.

➢ CNN Model Architecture

Here's an example CNN architecture that works well for plant disease detection:

1. **Input Layer**: Image input (224x224x3).
2. **Convolutional Layer 1**: 32 filters (3x3), followed by **ReLU** activation.
3. **Max Pooling Layer 1**: Pooling window (2x2).
4. **Convolutional Layer 2**: 64 filters (3x3), followed by **ReLU**.
5. **Max Pooling Layer 2**: Pooling window (2x2).
6. **Flatten Layer**: Flatten the feature maps into a 1D vector.
7. **Fully Connected Layer**: 128 neurons with **ReLU** activation.
8. **Output Layer**: SoftMax activation to classify images into categories (Healthy, Blight, Rust, etc.).

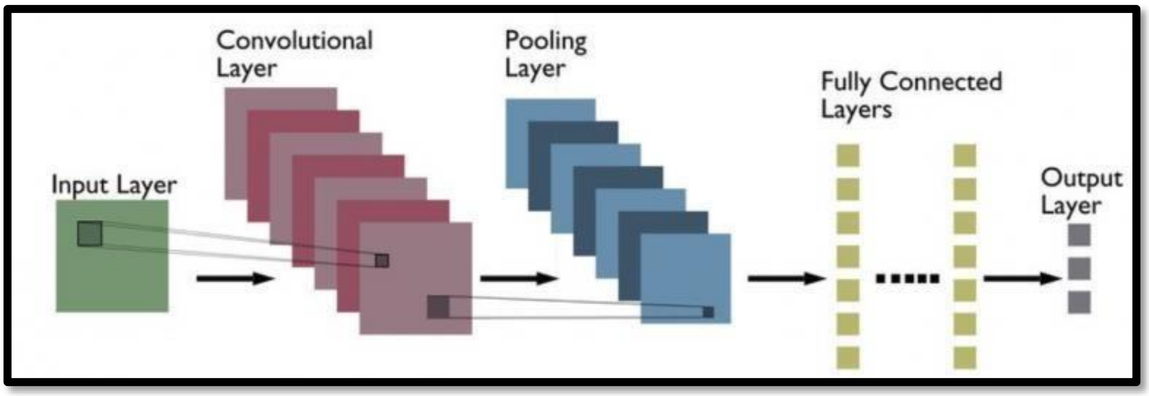


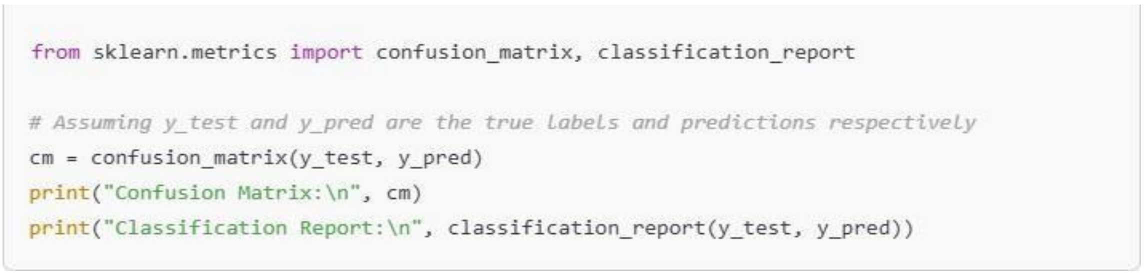
Figure No 5.1.3 CNN Architecture

➢ Model Training

- Use Keras with the Adam optimizer and cross-entropy loss to train the model.
- Split the dataset into training (80%) and validation (20%) sets.
- During training, monitor accuracy, precision, recall, and F1-score to ensure the model is generalizing well.

➢ **Model Evaluation**

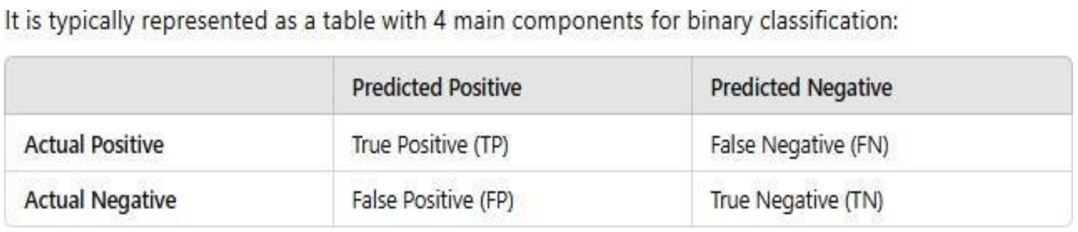
Evaluate the model using accuracy and confusion matrix to check its performance. For example:



```python
from sklearn.metrics import confusion_matrix, classification_report

# Assuming y_test and y_pred are the true labels and predictions respectively
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

A confusion matrix is a performance evaluation tool for classification models. It summarizes the results of predictions compared to the actual values, providing insights into how well the model is performing

- Structure of Confusion Matrix: -



It is typically represented as a table with 4 main components for binary classification:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Terms in Confusion Matrix

1. **True Positive (TP):**
   - Cases where the model correctly predicts the positive class.
2. **True Negative (TN):**
   - Cases where the model correctly predicts the negative class.
3. **False Positive (FP):**
   - Cases where the model incorrectly predicts the positive class (Type I error).
4. **False Negative (FN):**
   - Cases where the model incorrectly predicts the negative class (Type II error)

Role of Confusion Matrix In Smart Diseases Detection

1. Plant Disease Detection Evaluation:
   - Your machine learning model (e.g., a Convolutional Neural Network) analyzes images of plants to classify diseases.

- The confusion matrix helps evaluate the model's predictions against the actual disease labels.

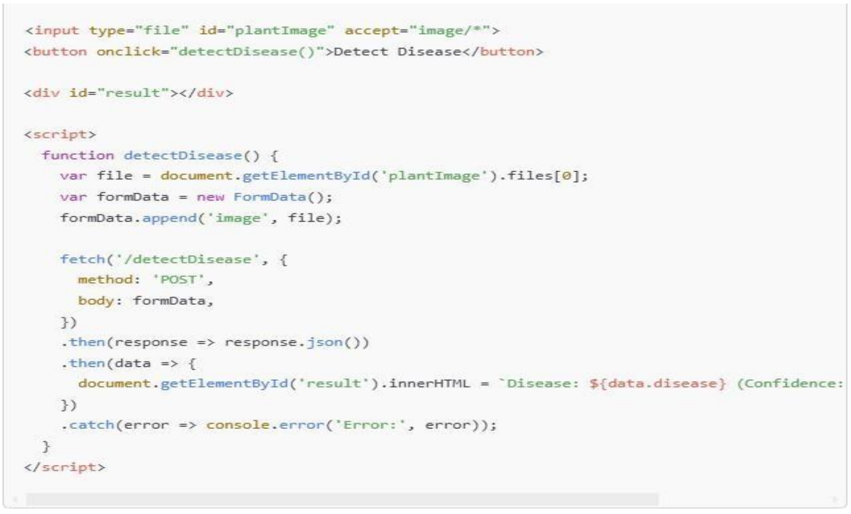
2. Performance Metrics:
   - The confusion matrix allows you to calculate:
     - Accuracy: How many predictions are correct overall.
     - Precision: The proportion of positive predictions (disease detected) that are correct.
     - Recall (Sensitivity): How well the model identifies diseased plants (no diseased plants missed).
3. Identifying Misclassifications:
   - By analyzing false positives (FP) and false negatives (FN) in the confusion matrix:
     - You can see where the model makes errors (e.g., incorrectly classifying a healthy plant as diseased or missing actual diseases).
     - This helps in fine-tuning the model and improving accuracy.

➢ **Model Deployment**

- Save the trained model using TensorFlow or TensorFlow Lite for real-time inference.
- Deploy the model on the cloud platform, or use Firebase to host the model for inference.

➢ **Disease Detection via Web Interface**

- Users can upload plant images via the web interface, where the image is processed by the model to predict whether it’s healthy or diseased.



```
<input type="file" id="plantImage" accept="image/*">
<button onclick="detectDisease()">Detect Disease</button>

<div id="result"></div>

<script>
  function detectDisease() {
    var file = document.getElementById('plantImage').files[0];
    var formData = new FormData();
    formData.append('image', file);

    fetch('/detectDisease', {
      method: 'POST',
      body: formData,
    })
    .then(response => response.json())
    .then(data => {
      document.getElementById('result').innerHTML = `Disease: ${data.disease} (Confidence:
    })
    .catch(error => console.error('Error:', error));
  }
</script>
```

5.2 Yield Monitoring Using Environmental Sensors:

5.3.1 Overview

The yield monitoring system uses IoT sensors to measure environmental conditions like **soil** moisture, temperature, and humidity. The system provides real-time insights into the field's status, helping farmers optimize irrigation.

5.3.2 Requirements

- **Hardware**:
  - Soil Moisture Sensor (capacitive or resistive) for measuring soil moisture.
  - DHT22 or DHT11 Sensor for reading temperature and humidity.
  - Relay and Water Pump for controlling irrigation based on sensor readings.
  - ESP8266 microcontroller for reading sensor data and sending it to Firebase.
- **Software**:
  - Arduino IDE for programming the ESP8266.
  - Firebase SDK for real-time data transfer to the Firebase database.
  - Web Interface for monitoring the sensor data

5.3.3 Step-by-Step Implementation

➢ **Hardware Setup**

- Soil Moisture Sensor: Connect the sensor's analog output to the ESP8266's analog input (A0 pin).
- DHT22 Sensor: Connect the sensor to a digital pin on the ESP8266.
- Water Pump: Use a relay to control the pump's ON/OFF state based on soil moisture levels.

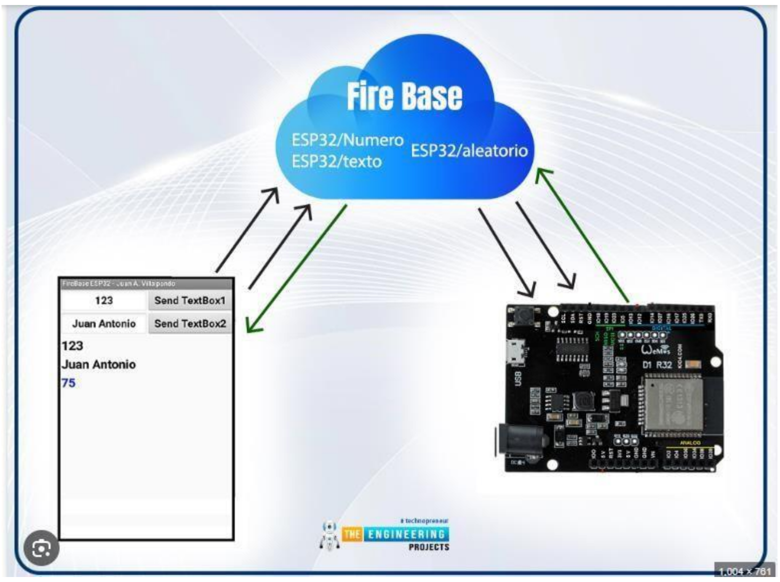


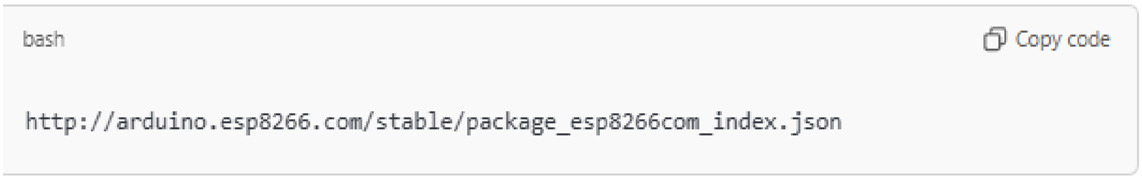
Fig No 5.2 Firebase Connection

- **Installation Of Libraries and Other Component: -**
  - Installations for Hardware

    Install Arduino IDE for ESP8266

    Step 1: Download and install Arduino IDE from the official Arduino website.

    Step 2: Open the Arduino IDE, go to File > Preferences and in the Additional Boards Manager URLs box, add the following URL:



```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

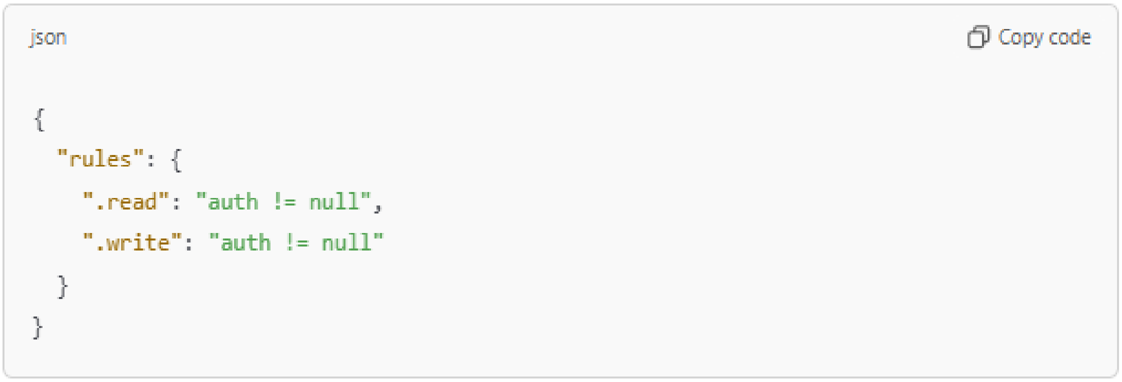
Step 3: Go to Tools > Board > Boards Manager. Search for ESP8266 and click Install.

Step 4: After installation, select your board by going to Tools > Board and selecting NodeMCU 1.0 (ESP-12E Module) or any other ESP8266 variant you are using.

- **Install Libraries for Sensors and Firebase**
  - DHT Sensor: For temperature and humidity readings, use the DHT sensor library:
    - Go to Sketch > Include Library > Manage Libraries.
    - Search for DHT sensor library and install it by Adafruit.
  - FirebaseESP8266: For Firebase communication, install the Firebase ESP8266 library:
    - Go to Sketch > Include Library > Manage Libraries.
    - Search for Firebase ESP8266 and click Install.
  - Other Libraries:
    - ESP8266WiFi: Used for Wi-Fi connectivity with the ESP8266.
    - Adafruit Unified Sensor: Required for the DHT sensor library.
- **Connect Sensors**
  - Soil Moisture Sensor: Connect the analog output pin of the sensor to the A0 pin on the ESP8266.
  - DHT22 Sensor: Connect the Data pin of the DHT22 sensor to any digital pin on the ESP8266 (e.g., D2).
  - Relay and Water Pump: Connect the relay module's input pin to a digital output pin (e.g., D1). Use the relay to control the water pump's ON/OFF state.

5.3 Setting up Firebase Libraries

- Create a Firebase Project
  - Step 1: Go to the Firebase Console, sign in with your Google account, and click Add Project.
  - Step 2: Follow the on-screen instructions to create a new project. After the project is created, navigate to the Database section.
- Firebase Database Setup
  - Step 1: Go to Realtime Database from the left sidebar.
  - Step 2: Select Create Database and set the security rules to test mode (for development purposes):

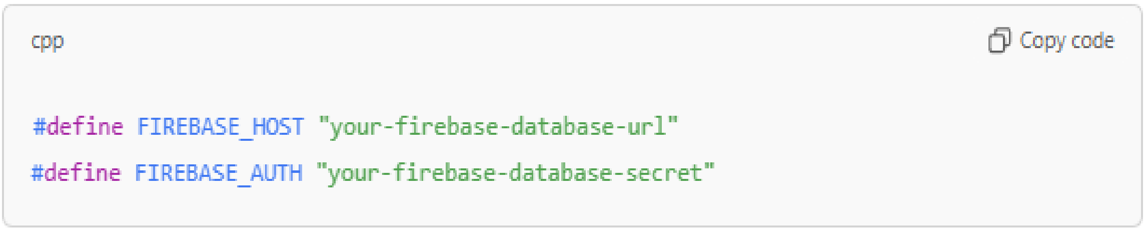


```json
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

- Firebase Authentication Setup
  - Step 1: Go to the Project Settings by clicking the gear icon next to your project name.
  - Step 2: Under Your Apps, select Web App and get your Firebase Web API key and Database URL.

5.4 Add Firebase Credentials in Arduino IDE

In your Arduino IDE, open the code for the ESP8266 device and add your Firebase details

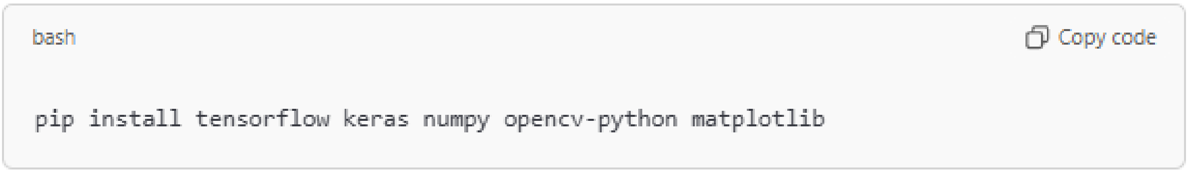


```cpp
#define FIREBASE_HOST "your-firebase-database-url"
#define FIREBASE_AUTH "your-firebase-database-secret"
```

5.5 Set Up AI/ML Disease Detection System

5.5.1 Install Python and Libraries

- Step 1: Download and install Python from the official website.
- Step 2: Install required Python libraries via pip:



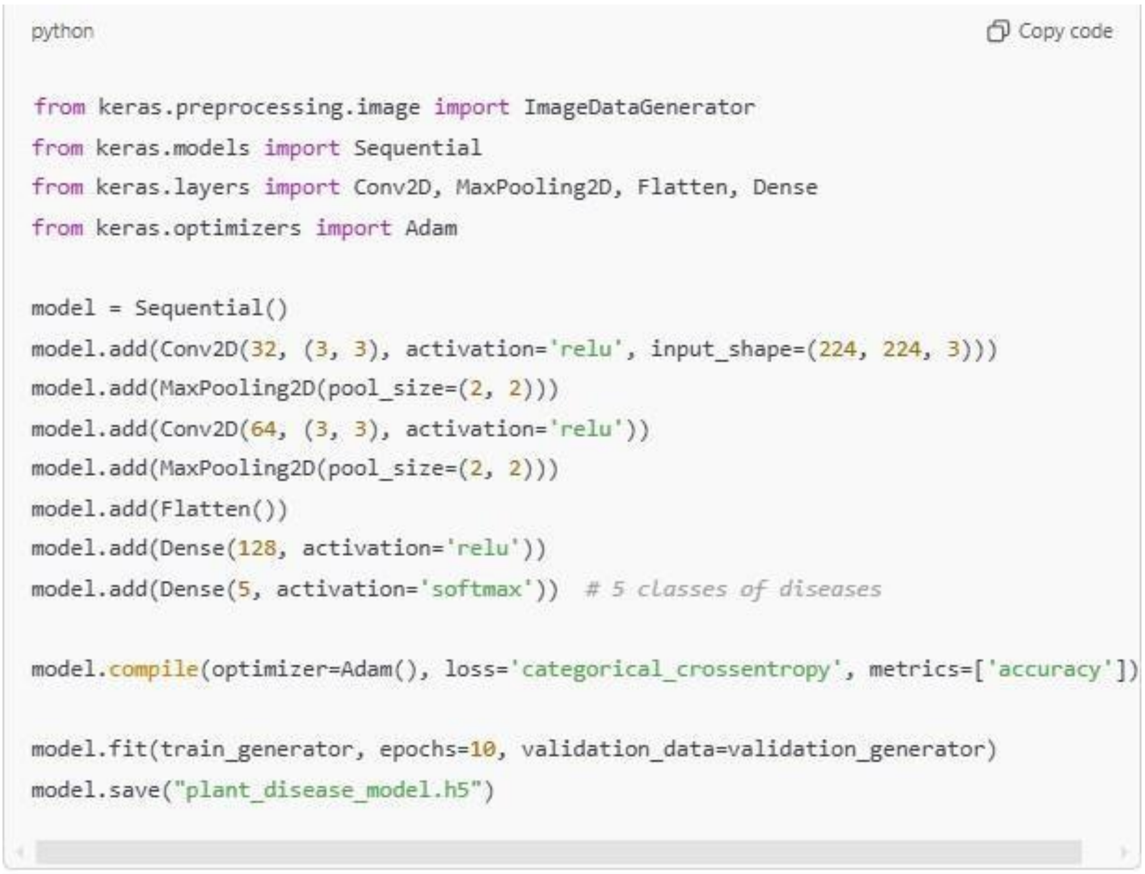
```bash
pip install tensorflow keras numpy opencv-python matplotlib
```

5.5.2 Collect and Preprocess Data

- Download a dataset for plant disease detection, such as the Plant Village dataset, or use your own set of labeled images.
- Ensure images are labeled properly (healthy or diseased with categories).
- Step 1: Resize images to a consistent size (e.g., 224x224 pixels).
- Step 2: Apply data augmentation techniques (like rotation, flip) for better generalization.

5.5.3 Train a CNN Model

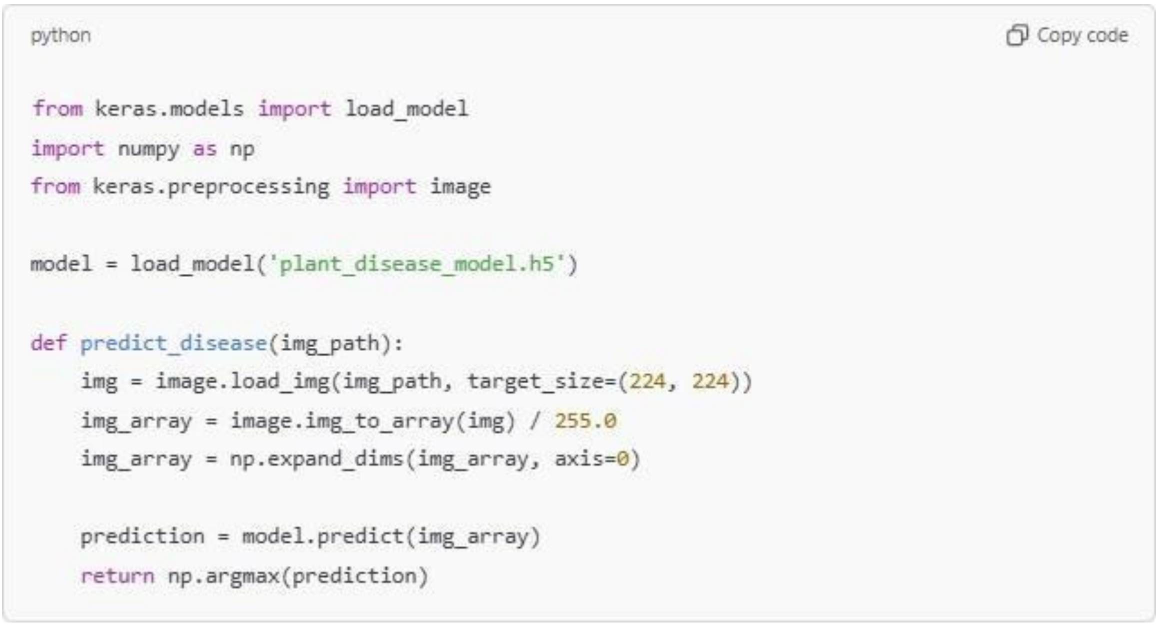
- Write the Python script for training the disease detection model:



```python
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.optimizers import Adam

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(5, activation='softmax'))  # 5 classes of diseases

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

model.fit(train_generator, epochs=10, validation_data=validation_generator)
model.save("plant_disease_model.h5")
```

5.5.4 Disease Detection Using the Model

- Use the trained model to predict the disease type by passing a new plant image.



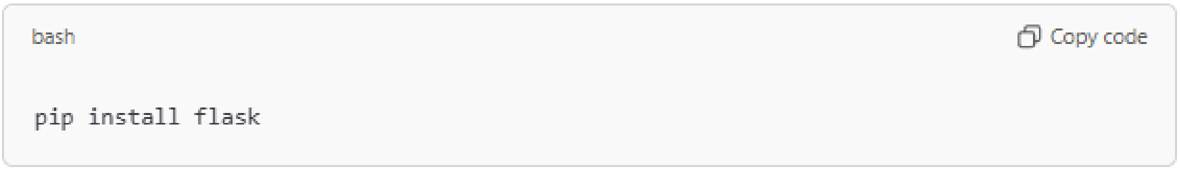
```python
from keras.models import load_model
import numpy as np
from keras.preprocessing import image

model = load_model('plant_disease_model.h5')

def predict_disease(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    return np.argmax(prediction)
```

Step 1: Deploy the model to Firebase or use a server to serve the model and make predictions based on plant image uploads.

5.6 Web Interface Setup

5.6.1 Set Up a Simple Web Server

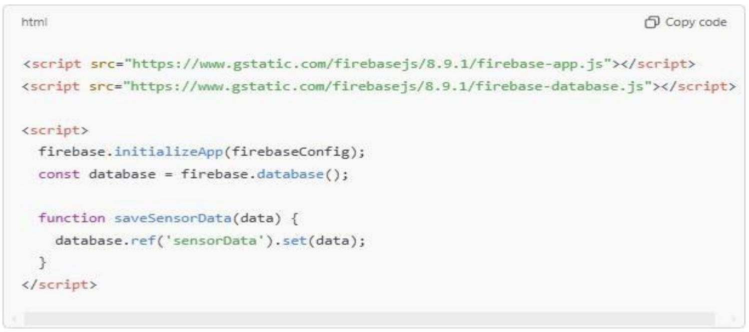
- Use Flask or Django (for Python) to set up a backend that serves the disease detection model and communicates with Firebase.
- Flask Setup:



```bash
pip install flask
```

5.6.2 Connect Web Interface to Firebase

- Use Firebase JavaScript SDK to read and write data on the web interface:



```html
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-database.js"></script>

<script>
  firebase.initializeApp(firebaseConfig);
  const database = firebase.database();

  function saveSensorData(data) {
    database.ref('sensorData').set(data);
  }
</script>
```

CHAPTER 6 TESTING

Testing is the process of executing a program to find errors. To make our software perform well it should be error-free. If testing is done successfully, it will remove all the errors from the software. In this article, we will discuss first the principles of testing and then we will discuss, the different types of testing.

Principles of Testing

- All the tests should meet the customer's requirements.
- To make our software testing should be performed by a third party.
- Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
- All the tests to be conducted should be planned before implementing it
- It follows the Pareto rule (80/20 rule) which states that 80% of errors come from 20% of program components.
- Start testing with small parts and extend it to large parts.

6.1 Unit Testing

Unit testing is a method of testing individual units or components of a software application. It is typically done by developers and is used to ensure that the individual units of the software are working as intended. Unit tests are usually automated and are designed to test specific parts of the code, such as a particular function or method. Unit testing is done at the lowest level of the software development process, where individual units of code are tested in isolation.

6.2 Integration Testing

Integration testing is a method of testing how different units or components of a software application interact with each other. It is used to identify and resolve any issues that may arise when different units of the software are combined. Integration testing is typically done after unit testing and before functional testing and is used to verify that the different units of the software work together as intended.

6.3 System Testing

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

6.4 Performance Testing

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size.

6.5 Acceptance Testing

Acceptance testing is software testing that evaluates whether a system meets its business and user requirements. It is the final testing stage before a system is released to production.

The main purpose of acceptance testing is to verify that a system:

- Meets all of its functional and non-functional requirements
- Easy to use and navigate
- Reliable and functions as expected
- Secure and comply with all applicable regulations

This testing is performed by end users or an expected group of users. This ensures that the system is tested from the perspective of the people using it daily.

Test Cases:

Test case number	1
Test case name	CNN Model Disease Detection Accuracy
Test Classification	Unit Testing
Feature to be tested	Disease prediction accuracy of the CNN model
Description	Tests the individual CNN model's ability to classify plant disease images.
Sample Input	Image of a diseased plant leaf (e.g., “Tomato Leaf with Blight”).
Expected output	Predicted disease label (e.g., “Tomato Blight”).
Actual output	“Tomato Blight.”
Remarks	Passed.

Table 6.1.1: Test Case 1 For CNN Model Disease Detection Accuracy.

Test case number	2
Test case name	IoT Sensor Data Transmission
Test Classification	Integration Testing
Feature to be tested	Integration between IoT sensors and backend data transmission
Description	Verifies data transmission and handling between sensors and the server
Sample Input	Temperature: 28°C, Humidity: 65%, Soil Moisture: 45%.
Expected output	Data displayed accurately on the dashboard.
Actual output	Data displayed correctly.
Remarks	Passed.

Table 6.1.2: Test Case 2 For IoT Sensor Data Transmission.

Test case number	3
Test case name	
Test Classification	System Testing
Feature to be tested	Complete user authentication and login system.
Description	Ensures the login and registration processes work as a whole
Sample Input	Username: “Irfan” Password: “Irfan@1234”
Expected output	Successful login for valid users, error for invalid inputs.
Actual output	Login Successful
Remarks	Passed.

Table 6.1.3: Test Case 3 User Authentication Functionality.

Test case number	4
Test case name	Real-Time Monitoring Graph Display
Test Classification	Acceptance Testing
Feature to be tested	Real-time data visualization on the web interface.
Description	Confirms if the system meets user requirements for dynamic graph updates.
Sample Input	Continuous sensor data - Temperature: 30°C, Humidity: 70%, Soil Moisture: 50%.
Expected output	Graph displays and updates data dynamically.
Actual output	Graph updates accurately in real-time.
Remarks	Passed.

Table 6.1.4: Test Case 4 Real-Time Monitoring Graph Display.

Test case number	5
Test case name	End-to-End System Functionality
Test Classification	System Testing
Feature to be tested	Complete system workflow (Disease detection, IoT data monitoring, and dashboard display).
Description	Verify if all system components (CNN model, IoT sensors, and web interface) work seamlessly together.
Sample Input	Input Image: Diseased leaf (e.g., “Tomato Leaf with Blight”). Sensor Data: Temperature: 29°C, Humidity: 60%, Soil Moisture: 50%.
Expected output	Disease Label: “Tomato Blight.” Sensor Data: Displayed correctly on the dashboard.
Actual output	Disease and sensor data displayed accurately on the interface.
Remarks	Passed. The system workflow is functioning correctly. .

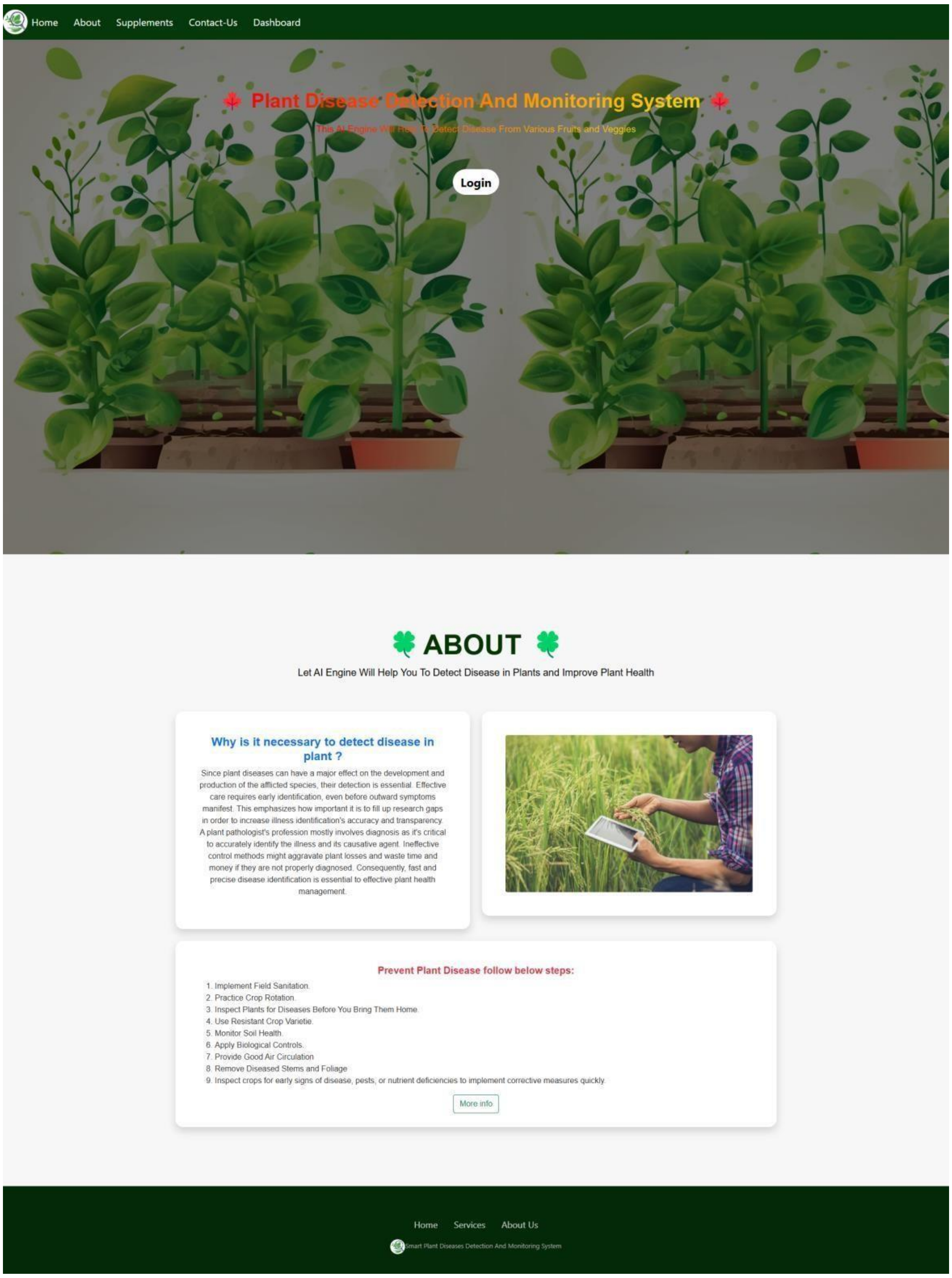
T
Table 6.1.5: Test Case 5 End-to-End System Functionality

Test case number	6
Test case name	System Performance Under Load
Test Classification	**Performance Testing**
Feature to be tested	System response time and stability under multiple user and data loads.
Description	Test system performance when multiple users access the system simultaneously and when large sensor data streams are processed.
Sample Input	50 concurrent user logins. Continuous sensor data: Temperature (30°C), Humidity (70%), Soil Moisture (45%).
Expected output	System remains stable with response time under 2 seconds.
Actual output	System handled 50 users, response time: 1.8 seconds
Remarks	Passed. The system performed efficiently under load. .

Table 6.1.6: Test Case 6 End-to-End System Functionality

CHAPTER 7 RESULTS

7.1Results


Home
About
Supplements
Contact-Us
Dashboard
Plant Disease Detection And Monitoring System
This AI Engine Will Help To Detect Disease From Various Fruits and Veggies
Login
ABOUT
Let AI Engine Will Help You To Detect Disease in Plants and Improve Plant Health
More info
Home
Services
About Us
Smart Plant Diseases Detection And Monitoring System

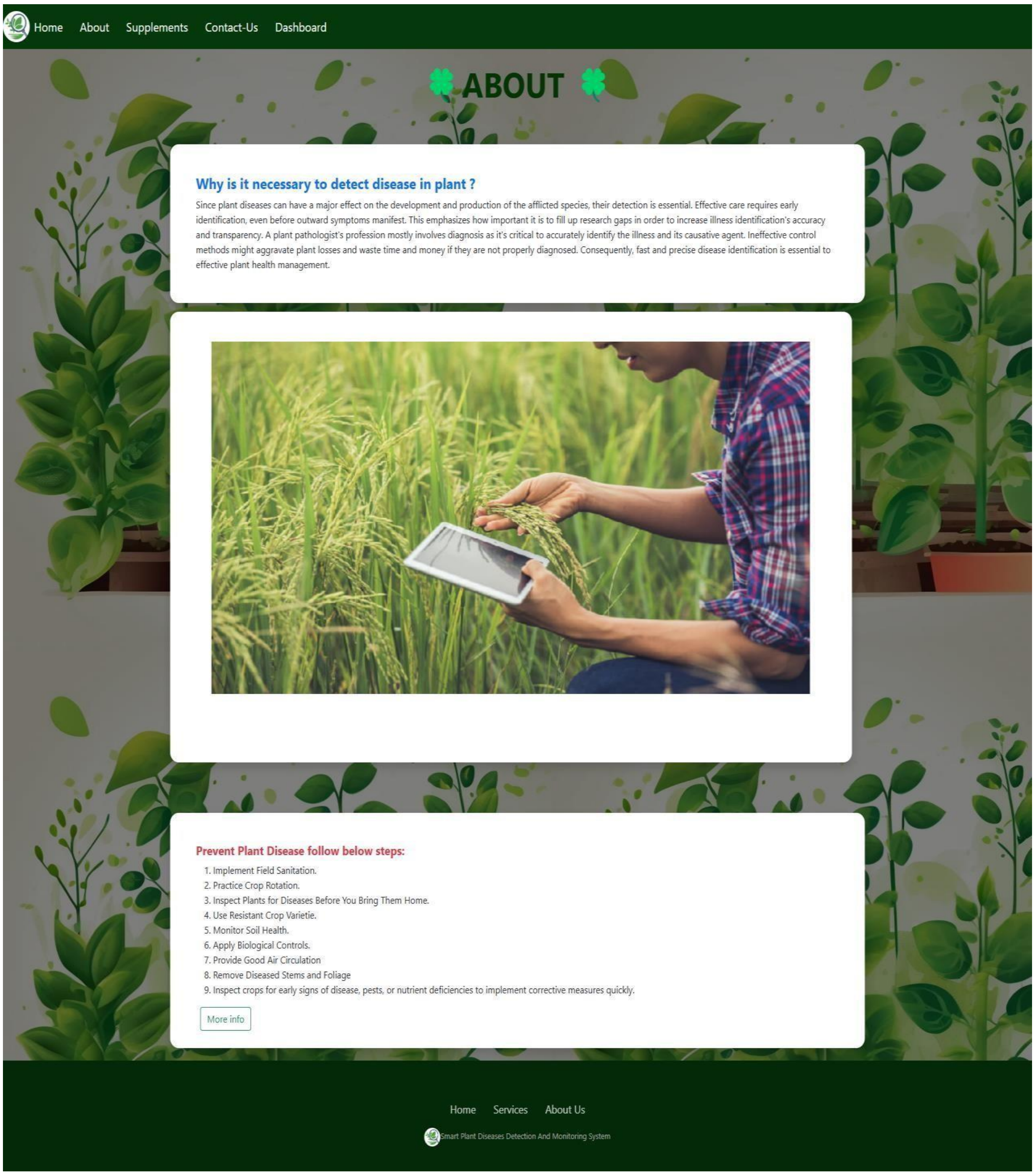
Why is it necessary to detect disease in plant ?

Since plant diseases can have a major effect on the development and production of the afflicted species, their detection is essential. Effective care requires early identification, even before outward symptoms manifest. This emphasizes how important it is to fill up research gaps in order to increase illness identification's accuracy and transparency. A plant pathologist's profession mostly involves diagnosis as it's critical to accurately identify the illness and its causative agent. Ineffective control methods might aggravate plant losses and waste time and money if they are not properly diagnosed. Consequently, fast and precise disease identification is essential to effective plant health management.

Prevent Plant Disease follow below steps:

1. Implement Field Sanitation.
2. Practice Crop Rotation.
3. Inspect Plants for Diseases Before You Bring Them Home.
4. Use Resistant Crop Varietie.
5. Monitor Soil Health.
6. Apply Biological Controls.
7. Provide Good Air Circulation
8. Remove Diseased Stems and Foliage
9. Inspect crops for early signs of disease, pests, or nutrient deficiencies to implement corrective measures quickly.

Figure 7.1 Home Page



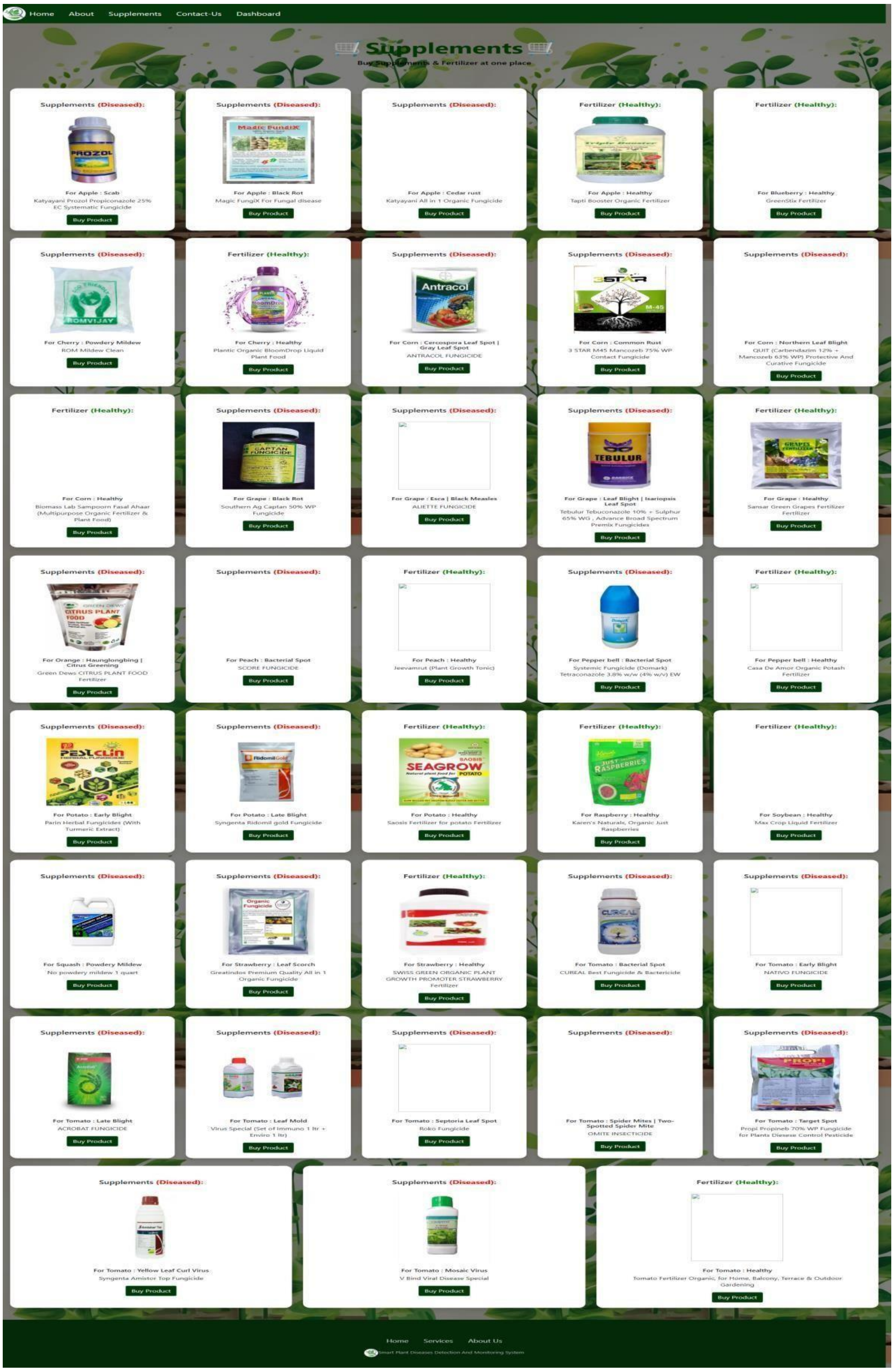
Why is it necessary to detect disease in plant ?

Since plant diseases can have a major effect on the development and production of the afflicted species, their detection is essential. Effective care requires early identification, even before outward symptoms manifest. This emphasizes how important it is to fill up research gaps in order to increase illness identification's accuracy and transparency. A plant pathologist's profession mostly involves diagnosis as it's critical to accurately identify the illness and its causative agent. Ineffective control methods might aggravate plant losses and waste time and money if they are not properly diagnosed. Consequently, fast and precise disease identification is essential to effective plant health management.

Prevent Plant Disease follow below steps:

1. Implement Field Sanitation.
2. Practice Crop Rotation.
3. Inspect Plants for Diseases Before You Bring Them Home.
4. Use Resistant Crop Varietie.
5. Monitor Soil Health.
6. Apply Biological Controls.
7. Provide Good Air Circulation
8. Remove Diseased Stems and Foliage
9. Inspect crops for early signs of disease, pests, or nutrient deficiencies to implement corrective measures quickly.

Figure No: 7.2 About Page



Home About Supplements Contact-Us Dashboard

Supplements

Buy Supplements & Fertilizer at one place

Supplements (Diseased):
For Apple : Scab
Katyayani Prozol Propiconazole 25% EC Systematic Fungicide
Buy Product

Supplements (Diseased):
For Apple : Black Rot
Magic FungiX For Fungal disease
Buy Product

Supplements (Diseased):
For Apple : Cedar rust
Katyayani All in 1 Organic Fungicide
Buy Product

Fertilizer (Healthy):
For Apple : Healthy
Tapti Booster Organic Fertilizer
Buy Product

Fertilizer (Healthy):
For Blueberry : Healthy
GreenStix Fertilizer
Buy Product

Supplements (Diseased):
For Cherry : Powdery Mildew
ROM Mildew Clean
Buy Product

Fertilizer (Healthy):
For Cherry : Healthy
Plantic Organic BloomDrop Liquid Plant Food
Buy Product

Supplements (Diseased):
For Corn : Cercospora Leaf Spot | Gray Leaf Spot
ANTRACOL FUNGICIDE
Buy Product

Supplements (Diseased):
For Corn : Common Rust
3 STAR M45 Mancozeb 75% WP Contact Fungicide
Buy Product

Supplements (Diseased):
For Corn : Northern Leaf Blight
QUIT (Carbendazim 12% + Mancozeb 63% WP) Protective And Curative Fungicide
Buy Product

Fertilizer (Healthy):
For Corn : Healthy
Biomass Lab Sampoorn Fasal Ahaar (Multipurpose Organic Fertilizer & Plant Food)
Buy Product

Supplements (Diseased):
For Grape : Black Rot
Southern Ag Captan 50% WP Fungicide
Buy Product

Supplements (Diseased):
For Grape : Esca | Black Measles
ALIETTE FUNGICIDE
Buy Product

Supplements (Diseased):
For Grape : Leaf Blight | Isariopsis Leaf Spot
Tebulur Tebuconazole 10% + Sulphur 65% WG , Advance Broad Spectrum Premix Fungicides
Buy Product

Fertilizer (Healthy):
For Grape : Healthy
Sansar Green Grapes Fertilizer Fertilizer
Buy Product

Supplements (Diseased):
For Orange : Haunglongbing | Citrus Greening
Green Dews CITRUS PLANT FOOD Fertilizer
Buy Product

Supplements (Diseased):
For Peach : Bacterial Spot
SCORE FUNGICIDE
Buy Product

Fertilizer (Healthy):
For Peach : Healthy
Jeevamrut (Plant Growth Tonic)
Buy Product

Supplements (Diseased):
For Pepper bell : Bacterial Spot
Systemic Fungicide (Domark) Tetraconazole 3.8% w/w (4% w/v) EW
Buy Product

Fertilizer (Healthy):
For Pepper bell : Healthy
Casa De Amor Organic Potash Fertilizer
Buy Product

Supplements (Diseased):
For Potato : Early Blight
Parin Herbal Fungicides (With Turmeric Extract)
Buy Product

Supplements (Diseased):
For Potato : Late Blight
Syngenta Ridomil gold Fungicide
Buy Product

Fertilizer (Healthy):
For Potato : Healthy
Saosis Fertilizer for potato Fertilizer
Buy Product

Fertilizer (Healthy):
For Raspberry : Healthy
Karen's Naturals, Organic Just Raspberries
Buy Product

Fertilizer (Healthy):
For Soybean : Healthy
Max Crop Liquid Fertilizer
Buy Product

Supplements (Diseased):
For Squash : Powdery Mildew
No powdery mildew 1 quart
Buy Product

Supplements (Diseased):
For Strawberry : Leaf Scorch
Greatindos Premium Quality All in 1 Organic Fungicide
Buy Product

Fertilizer (Healthy):
For Strawberry : Healthy
SWISS GREEN ORGANIC PLANT GROWTH PROMOTER STRAWBERRY Fertilizer
Buy Product

Supplements (Diseased):
For Tomato : Bacterial Spot
CUREAL Best Fungicide & Bactericide
Buy Product

Supplements (Diseased):
For Tomato : Early Blight
NATIVO FUNGICIDE
Buy Product

Supplements (Diseased):
For Tomato : Late Blight
ACROBAT FUNGICIDE
Buy Product

Supplements (Diseased):
For Tomato : Leaf Mold
Virus Special (Set of Immuno 1 ltr + Enviro 1 ltr)
Buy Product

Supplements (Diseased):
For Tomato : Septoria Leaf Spot
Roko Fungicide
Buy Product

Supplements (Diseased):
For Tomato : Spider Mites | Two-Spotted Spider Mite
OMITE INSECTICIDE
Buy Product

Supplements (Diseased):
For Tomato : Target Spot
Propi Propineb 70% WP Fungicide for Plants Diesese Control Pesticide
Buy Product

Supplements (Diseased):
For Tomato : Yellow Leaf Curl Virus
Syngenta Amistor Top Fungicide
Buy Product

Supplements (Diseased):
For Tomato : Mosaic Virus
V Bind Viral Disease Special
Buy Product

Fertilizer (Healthy):
For Tomato : Healthy
Tomato Fertilizer Organic, for Home, Balcony, Terrace & Outdoor Gardening
Buy Product

Home Services About Us

Smart Plant Diseases Detection And Monitoring System

Figure No 7.3 Supplements Page

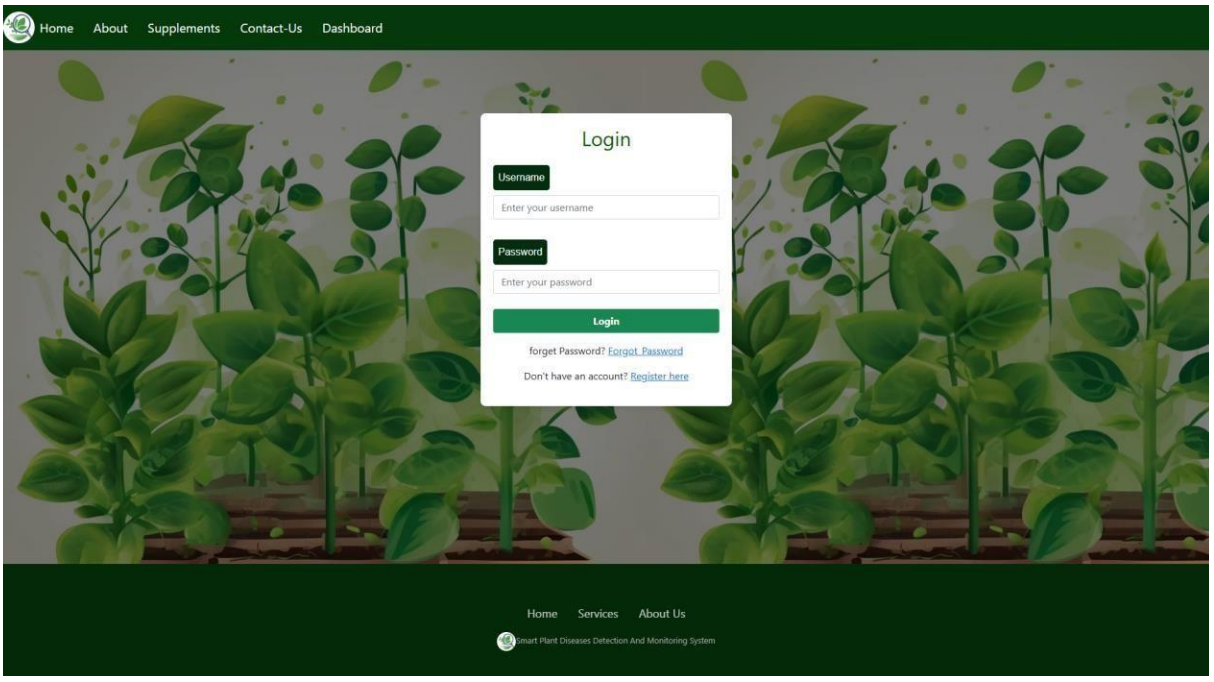


Figure No 7.4 Login Page

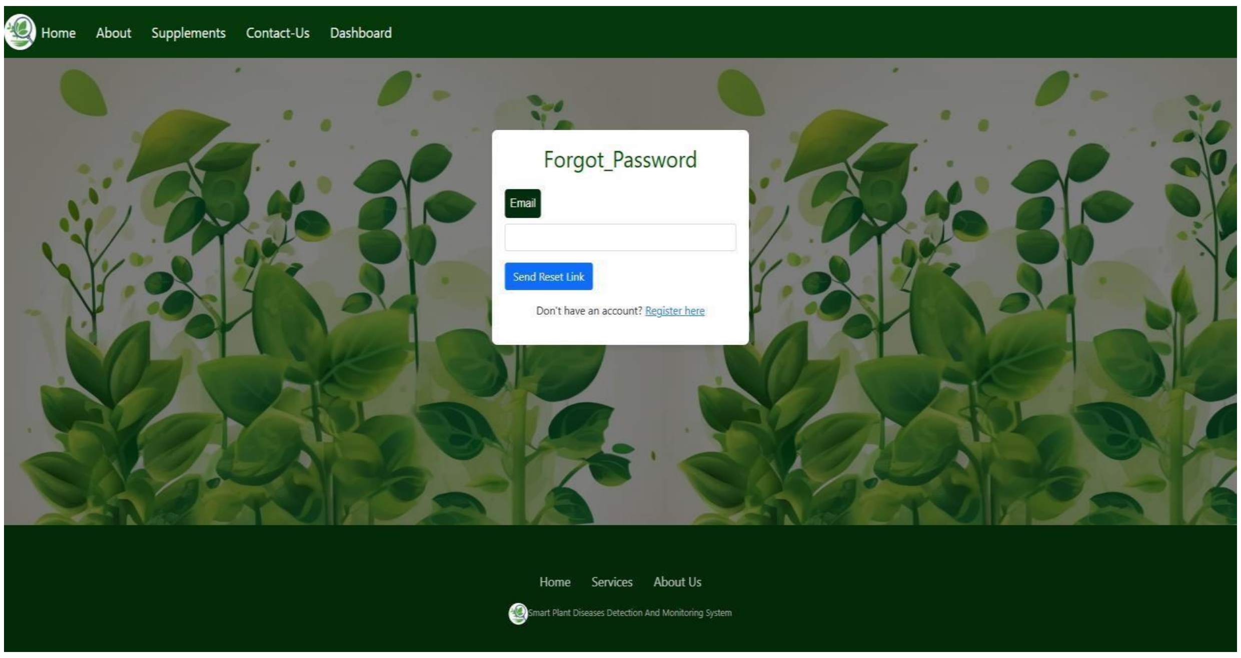


Figure No 7.5 Forgot_Password Page

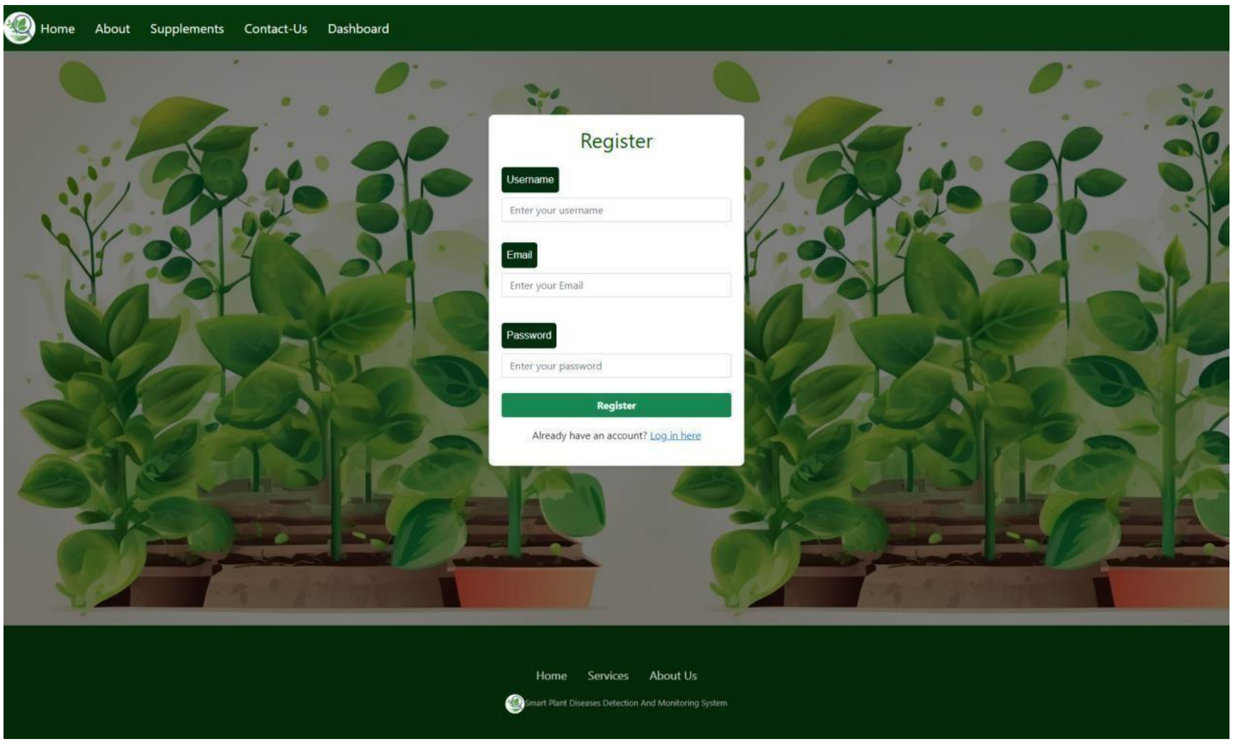


Figure No 7.6 Register Page

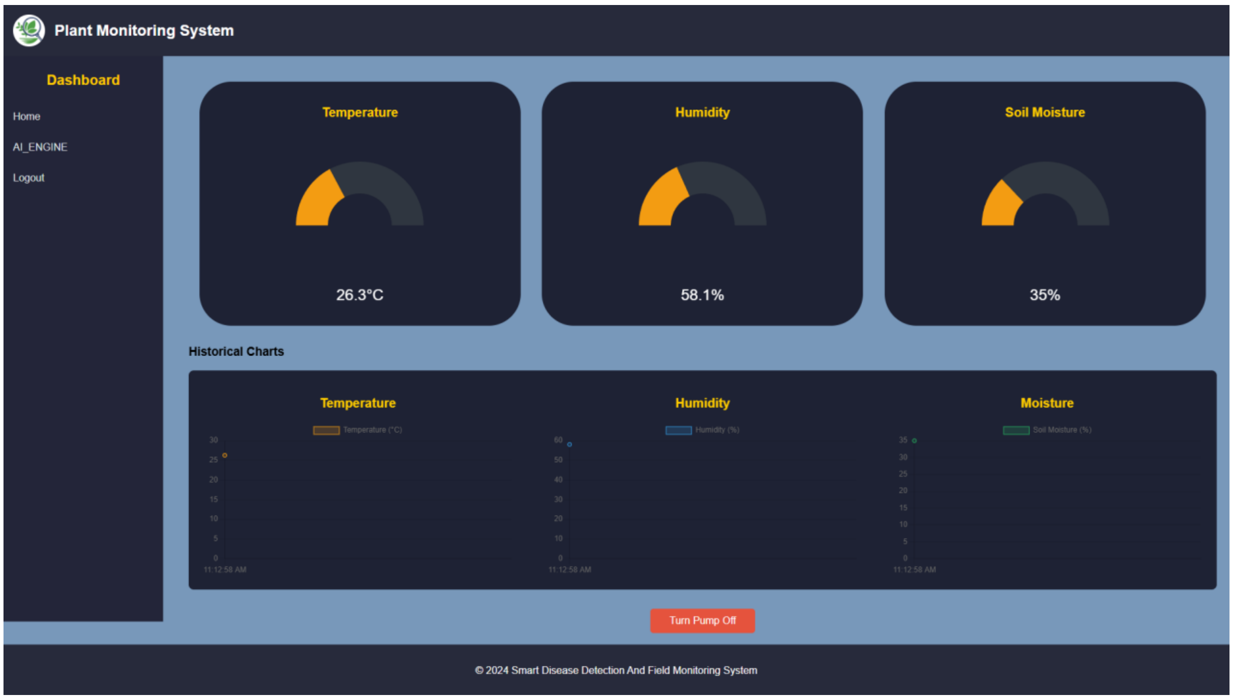
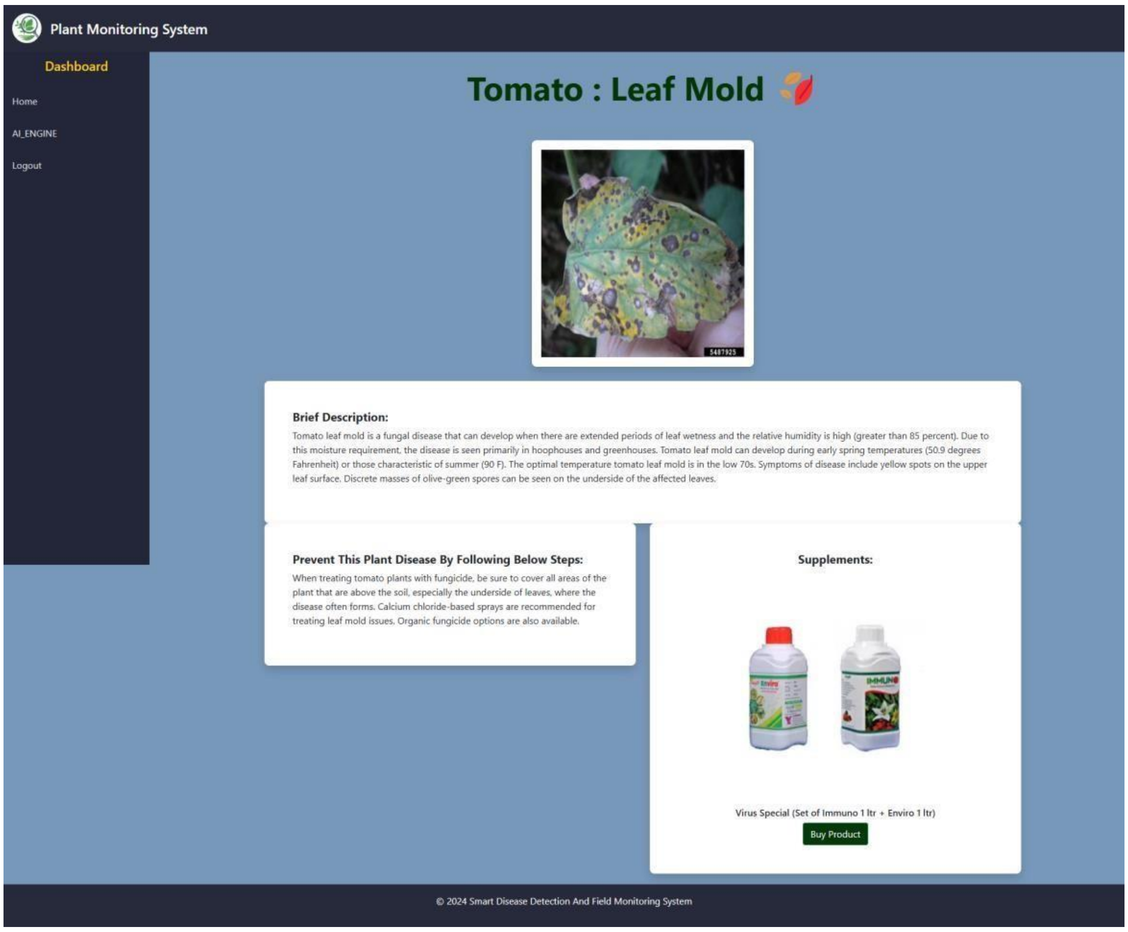


Figure No 7.7 Dashboard Page



Tomato : Leaf Mold

Brief Description:

Tomato leaf mold is a fungal disease that can develop when there are extended periods of leaf wetness and the relative humidity is high (greater than 85 percent). Due to this moisture requirement, the disease is seen primarily in hoophouses and greenhouses. Tomato leaf mold can develop during early spring temperatures (50.9 degrees Fahrenheit) or those characteristic of summer (90 F). The optimal temperature tomato leaf mold is in the low 70s. Symptoms of disease include yellow spots on the upper leaf surface. Discrete masses of olive-green spores can be seen on the underside of the affected leaves.

Prevent This Plant Disease By Following Below Steps:

When treating tomato plants with fungicide, be sure to cover all areas of the plant that are above the soil, especially the underside of leaves, where the disease often forms. Calcium chloride-based sprays are recommended for treating leaf mold issues. Organic fungicide options are also available.

Supplements:

Virus Special (Set of Immuno 1 ltr + Enviro 1 ltr)

© 2024 Smart Disease Detection And Field Monitoring System

Figure No :7.8 AI_Engine

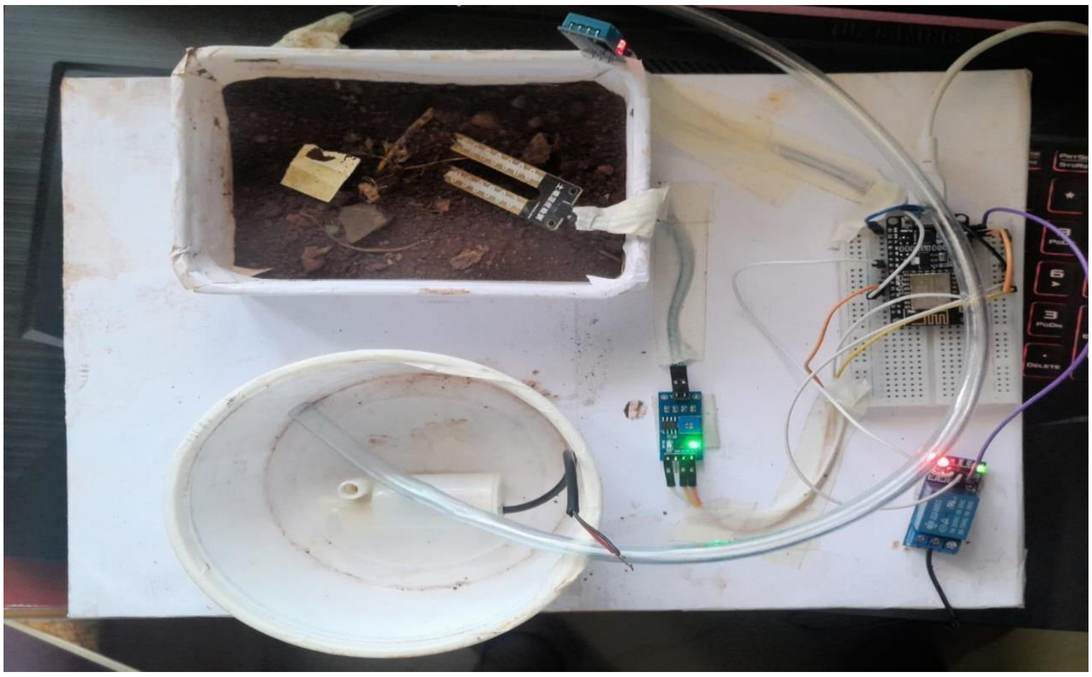


Figure No :7.9 IOT Setup

CHAPTER 8 APPLICATION AND CONCLUSION

8.1 Applications

- **Precision Agriculture**
    - Enables farmers to monitor environmental factors such as soil moisture, temperature, and humidity in real-time.
    - Facilitates automated irrigation control based on soil moisture levels, reducing water wastage and enhancing crop yield.
- **Early Disease Detection in Plants**
    - Utilizes AI/ML algorithms to detect plant diseases at an early stage through leaf image analysis, preventing the spread of diseases and minimizing crop loss.
    - Provides detailed feedback on the type and severity of the disease, allowing for targeted treatment.
- **Data-Driven Decision Making**
    - Historical data collected from sensors is stored in Firebase and displayed on dashboards, allowing farmers to analyze trends in environmental conditions and optimize farming practices.
- **Remote Monitoring and Control**
    - Farmers can remotely monitor field conditions via a web interface and control irrigation systems from anywhere, improving operational efficiency.
    - Alerts for critical conditions (e.g., low soil moisture) enable prompt action to protect crops.
- **Scalability for Large Farms**
    - Can be deployed across large agricultural fields with multiple sensors and nodes, providing comprehensive field monitoring.
- **Urban and Indoor Farming**
    - Suitable for monitoring environmental conditions in controlled environments such as greenhouses, hydroponics, and urban farming setups.

8.2 Conclusion

The **Smart Disease Detection and Yield Monitoring System** integrates IoT, AI/ML, and cloud technologies to provide an innovative solution for modern agriculture. By leveraging advanced sensors, machine learning, and real-time data visualization, the system addresses critical challenges in farming, including plant disease identification, resource optimization, and yield improvement. The combination of automated irrigation, disease detection, and an intuitive user interface ensures efficient resource utilization and enhances decision-making for farmers.

This system has proven effective in improving productivity by reducing water wastage, minimizing crop loss due to diseases, and enabling remote monitoring and control. By offering a user-friendly interface and real-time data insights, it empowers farmers with the tools needed to adopt sustainable farming practices and maximize profits.

8.3 Future Scope

- **Integration with Advanced Machine Learning Models**
  - Implementing more sophisticated AI/ML models such as convolutional neural networks (CNNs) with deeper architectures (e.g., ResNet, EfficientNet) for higher accuracy in plant disease detection.
  - Expanding the model to detect a broader range of plant diseases and deficiencies caused by nutrient imbalances.
- **Multispectral Imaging for Disease Analysis**
  - Incorporating multispectral or hyperspectral cameras to analyze plant health beyond the visible spectrum, detecting diseases and stress levels with higher precision.
- **Edge Computing for Faster Processing**
  - Deploying AI models on edge devices such as ESP32-CAM or Raspberry Pi for real-time disease detection without the need for constant internet connectivity.
- **Integration with Weather Forecasting**
  - Using weather prediction APIs to provide farmers with actionable insights on when to irrigate or apply disease prevention measures based on environmental conditions.
- **Mobile Application Development**
  - Developing a mobile application for farmers to access the dashboard, receive alerts, and control irrigation systems, enhancing accessibility and usability.

- **Advanced Analytics and Reports**
  - Incorporating advanced analytics, such as yield predictions and cost-benefit analysis, to help farmers make informed financial decisions.
- **Automated Pesticide Spraying Systems**
  - Adding automated pesticide or fertilizer dispensers that work in tandem with disease detection algorithms, reducing manual labor.
- **Scalability and Cloud Computing**
  - Extending the system's scalability to handle data from thousands of sensors and integrate with other agricultural IoT systems using platforms like AWS IoT or Google Cloud IoT.
- **Renewable Energy Integration**
  - Utilizing solar panels to power sensors, controllers, and water pumps, ensuring sustainable operation in off-grid locations.
- **Localization and Customization**
  - Customizing the system for specific crops, regional conditions, and languages to cater to a diverse range of farmers across the globe.

REFERENCES

[1] *F. Fina, P. Birch, R. Young, J. Obu, B. Faithpraise, and C. Chatwin, ''Automatic plant pest detection and recognition using k-means clustering algorithm and correspondence filters,'' Int. J. Adv.Biotechnol. Res., vol. 4, no. 2, pp. 189–199, Jul. 2013. The Robotic Process Automation Handbook by Tom Taulli*

[2] *Plant Leaf Disease Detection using Inception Resnet V2 -CNN Vishal Tanawade, Sarthak Thorat, Suraj Shirude, Hitesh Saswadkar, Pratik Karhekar, Prof. V. V. Waykule*

[3] *An Overview of the Research on Plant Leaves Disease detectionusing Image Processing Techniques", January 2014, Ms. Kiran R.Gavhale.*

[4] *Apple Scab and Marsonina Coronaria Diseases Detection in Apple Leaves Using Machine Learning January 2018 authors Swati Singh, Sheifali Gupta, International Journal of Pure and Applied Mathematics*

[5] *https://www.researchgate.net/publication/285164623_An_Introduction_to_Convolutional_Neural_Networks*

[6] *Yogesh, Singh & Bhatia, Pradeep & Sangwan, Om. (2007). A REVIEW OF STUDIES ON MACHINE LEARNING TECHNIQUES. International Journal of Computer Science and Security. 1.*

[7] *Rahman, Rafizah & Hashim, Ummi & Ahmad, Sabrina. (2020). IoT based temperature and humidity monitoring framework. Bulletin of Electrical Engineering and Informatics. 9. 10.11591/eei.v9i1.1557.*

[8] IoT based temperature and humidity monitoring framework *Rafizah Ab Rahman, Ummi Raba'ah Hashim, Sabrina Ahmad*

[9] A Khakimov et al 2022 IOP Conf. Ser.: Earth Environ. Sci. 951 012002.

[10] Hashim, Nik Mohd Zarifie & Mazlan, S. & Abd Aziz, Mohamad Zoinol & SALLEH, AZAHARI & Ja'afar, A. & Mohamad, Najmiah. (2015). Agriculture monitoring system: A study. Jurnal Teknologi. 77. 10.11113/jt.v77.4099.