

An  
Industry Oriented Mini Project Report  
on

# **FARM HEALTH DISEASE DETECTION AND TREATMENT SOLUTION**

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

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By**

A. AVINASH	227Z1A0501
A. KARTHIK	227Z1A0513
A. CHARAN	227Z1A0506

**Under the Guidance of**

Mr. S. Srikanth Reddy  
Assistant Professor



**SCHOOL OF ENGINEERING  
Department of Computer Science and Engineering**

**NALLA NARASIMHA REDDY  
EDUCATION SOCIETY'S GROUP OF INSTITUTIONS  
(AN AUTONOMOUS INSTITUTION)**

Approved by AICTE, New Delhi, Chowdariguda (V) Korremula 'x' Roads,  
via Narapally, Ghatkesar (Mandal) Medchal (Dist), Telangana-500088  
2024-2025



**NALLA NARASIMHA REDDY**  
Education Society's Group of Institutions - Integrated Campus  
(UGC AUTONOMOUS INSTITUTION)



**SCHOOL OF ENGINEERING**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the project report titled “**FARM HEALTH DISEASE DETECTION AND TREATMENT SOLUTION**” is being submitted by **A. AVINASH (227Z1A0501), A. KARTHIK (227Z1A0513) and A. CHARAN (227Z1A0506)** in Partial fulfillment for the award of **Bachelor of technology in Computer Science & Engineering** is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

**Internal Guide**  
(Mr. S. Srikanth Reddy)

**Head of the Department**  
(Dr. K. Rameshwaraiah)

Submitted for the University Examination held on.....

**External Examiner**

## DECLARATION

We A. Avinash, A. Karthik and A. Charan the students of **Bachelor of Technology in Computer Science and Engineering, Nalla Narasimha Reddy Education Society's Group of Institutions**, Hyderabad, Telangana, hereby declare that the work presented in this project work entitled **FARM HEALTH DISEASE DETECTION AND TREATMENT SOLUTION** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning.

By

A. Avinash	227Z1A0501
A. Karthik	227Z1A0513
A. Charan	227Z1A0506

**Date:**

**Signature:**

## ACKNOWLEDGEMENT

We express our sincere gratitude to our guide **Mr. S. Srikanth Reddy**, Assistant Professor, Department of Computer Science and Engineering, NNRESGI, who motivated throughout the period of the project and also for his valuable and intellectual suggestions, guidance, and constant encouragement right throughout our work.

We would like to express our profound gratitude to our project coordinator **Mr. B. Durga Prasad**, Assistant Professor, Department of Computer Science and Engineering, NNRESGI, for his support and guidance in completing our project and for giving us this opportunity to present the project work.

We profoundly express thanks to **Dr. K. Rameshwaraiah**, Professor & Head, Department of Computer Science and Engineering, NNRESGI, for his cooperation and encouragement in completing the project successfully.

We wish to express our sincere thanks to **Dr. G. Janardhana Raju**, Dean School of Engineering, NNRESGI, for providing the facilities for completion of the project.

We wish to express our sincere thanks to **Dr. C. V. Krishna Reddy**, Director, NNRESGI, for providing the facilities for completion of the project.

Finally, we would like to thank Project Review Committee (PRC) members, all the faculty members and supporting staff, Department of Computer Science and Engineering, NNRESGI, for extending their help in all circumstances.

By

A. Avinash                      227Z1A0501

A. Karthik                      227Z1A0513

A. Charan                      227Z1A0506

## ABSTRACT

The Plant Disease Detection System leverages AI and machine learning to support farmers in managing crop health and improving yields. By analyzing images of plants, the system quickly identifies diseases and offers targeted solutions to treat or prevent them, helping farmers take immediate action. In addition to disease detection, the system provides personalized farming advice, considering factors like soil type, weather conditions, and location to recommend the best crops and growing techniques. An AI-powered chatbot is integrated into the platform, offering farmers real-time support and answering questions about crop care, treatments, and general farming practices. To make the system accessible to farmers worldwide, a language translation feature allows users to switch to their native language, ensuring ease of use for all. Overall, this system aims to reduce crop loss, boost productivity, and empower farmers by providing them with the tools and information they need to succeed.

**Keywords-** *Plant disease detection, AI, machine learning, crop health, disease identification, treatment solutions, personalized farming advice, AI-powered chatbot, real-time support and crop loss reduction.*



# TABLE OF CONTENTS

<b>Contents</b>	<b>Page No.</b>
<b>List of Figures</b>	<b>i</b>
<b>List of Tables</b>	<b>ii</b>
<b>List of Abbreviations</b>	<b>iii</b>
<b>1. INTRODUCTION</b>	<b>1</b>
1. Motivation	1
2. Problem Definition	2
3. Objective of project	2
4. Limitation of project	2
<b>2.LITERATURE SURVEY</b>	<b>4</b>
1. Introduction	4
2. Existing System	5
3. Proposed System	6
<b>3.SYSTEM ANALYSIS</b>	<b>8</b>
1. Functional requirements	8
2. Non-Functional requirements	8
3. Software requirements	9
4. Hardware requirements	9
5. Block Diagram of the Proposed System	9
<b>4.SYSTEM DESIGN</b>	<b>11</b>
1. Introduction	11
2. UML Diagrams	11
3. Modules	15
<b>5.IMPLEMENTATION &amp; RESULTS</b>	<b>17</b>
1. Introduction	17
2. Method of Implementation	17
3. Algorithm and Flowcharts	29
4. Control Flow of the implementation	29
5. Sample Code	30
6. Output Screens	32

<b>6. TESTING</b>	<b>35</b>
1. Introduction to Testing	35
2. Types of Tests considered	35
3. Various Test case scenarios considered	36
<b>7. CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>37</b>
1. Project Conclusion	37
2. Future Enhancement	37
<b>8. REFERENCES</b>	<b>38</b>
1. Journals	38
2. Books	38
3. Sites	38



## LIST OF FIGURES

<b>S. No.</b>	<b>Figure No.</b>	<b>Name of the Figure</b>	<b>Page No.</b>
1	3.5	Block diagram of system	9
2	4.2.1	Data flow diagram	12
3	4.2.2	Class diagram	13
4	4.2.3	Activity diagram	14
5	4.2.4	Use case diagram	15
6	4.2.5	Sequence diagram	16
7	5.1	Agile method	17
8	5.3	Flowchart	29
9	5.4.1	Importing Dependencies	30
10	5.6.1	Output screen for Home	32
11	5.6.2	Output screen for Disease Detection page	32
12	5.6.3	Output screen for Expert Farming Advice page	33
13	5.6.4	Output screen for Farmer Contribution page	33

## LIST OF TABLES

<b>S. No.</b>	<b>Table No.</b>	<b>Name of the Table</b>	<b>Page No.</b>
1	5.5	Import Dependencies	30
2	6.3	Test case for proposed system	35

## LIST OF ABBREVIATIONS

S. No.	Abbreviation	Definition
1	AI	Artificial Intelligence
2	API	Application Programming Interface
3	CNN	Convolutional Neural Network
4	DL	Deep Learning
5	HOG	Histogram of Oriented Gradients
6	ML	Machine Learning
7	VGG	Visual Geometry Group (a deep learning model architecture)

## 9. INTRODUCTION

Farming is the backbone of human survival, providing the essential food resources needed to sustain life. However, many farmers especially those in rural and resource-limited areas face critical challenges in maintaining healthy crops, one of the most serious being the early detection of plant diseases. These diseases often go unnoticed until they have already caused significant damage, leading to reduced crop yields, financial hardship, and, in some cases, total crop failure. To address this growing concern, we developed Farm Health, an innovative, AI-powered solution that enables farmers to identify plant diseases quickly and accurately by simply uploading a photo of the affected plant. This intelligent system uses advanced machine learning algorithms to analyse images and detect diseases in real-time, offering immediate feedback along with expert agricultural advice tailored to the specific issue. Designed with inclusivity in mind, Farm Health features a user-friendly interface, multi-language support, and secure data handling to ensure accessibility for farmers of all backgrounds. Our platform doesn't just diagnose; it educates, empowers, and guides farmers toward the best possible decisions for their crops. By bridging the gap between traditional farming and modern technology, Farm Health aims to promote sustainable agriculture, minimize crop losses, and improve the livelihoods of farming communities worldwide.

### 1. Motivation

The motivation behind our project stems from a pressing need in the agricultural sector—providing equal access to advanced tools for all farmers. While technology is transforming industries, many farmers, especially in remote and under-resourced areas, still lack the means to detect crop diseases early or receive expert guidance.

We are driven by the belief that every farmer deserves access to smart, reliable, and user-friendly solutions that can help them protect their crops and improve their livelihoods. Our goal is to bridge the digital divide in agriculture by using AI, machine learning, and cloud services to create a platform that empowers farmers with timely insights and expert advice. Farm Health is not just a tech project it's a step toward making farming more accessible, inclusive, and sustainable for everyone.

## **2. Problem Definition**

Farmers, particularly in rural and underserved areas, often face significant challenges in identifying plant diseases at an early stage due to limited access to expert knowledge and affordable diagnostic tools. This lack of timely detection can result in severe crop losses, reduced yields, and economic hardship. Existing digital solutions are often costly, overly technical, or not designed with the local context in mind, such as language support or regional farming conditions. Furthermore, many platforms do not integrate crucial data like real-time weather updates or personalized agricultural advice, leaving farmers without the complete information needed to make informed decisions. There is a clear need for an intelligent, accessible, and farmer-friendly solution that bridges these gaps. The Farm Health project is designed to meet this need by providing a unified platform for real-time disease detection, expert AI-driven guidance, and location-specific recommendations to empower farmers and improve agricultural outcomes.

## **3. Objective of Project**

The primary objectives of this project are twofold. First and foremost, we are committed to developing an intelligent, AI-powered agricultural platform that allows farmers to detect plant diseases accurately and in real-time using image-based analysis. This involves building a robust machine learning model, a user-friendly web interface, and a secure backend infrastructure to ensure seamless performance and accessibility. Secondly, our overarching aim is to empower farmers by providing them with expert, personalized agricultural guidance using advanced natural language processing and real-time weather data. By supporting multiple languages, integrating cloud services, and offering localized recommendations, we seek to bridge the knowledge and technology gap in agriculture. Through this system, we aspire to enhance productivity, reduce crop loss, and contribute to global food security by placing advanced digital tools in the hands of those who need them most.

## **4. Limitations of Project**

Farm Health depends on good image quality and stable internet. It may not detect rare diseases outside its trained dataset and can be limited in areas with poor connectivity.

## **1. Processing Time**

The processing time in the Farm Health system largely depends on several factors such as the size and quality of the plant image uploaded, the internet speed, and the efficiency of the backend servers. When users upload high-resolution images, the system may take a bit longer to preprocess and analyze them before generating a result. Additionally, if the network connection is slow or unstable, delays can occur during image upload and while fetching real-time weather data for generating expert advice. The complexity of the AI query, especially when involving crop-specific guidance and environmental factors, can also influence how quickly results are returned. While the system is optimized for speed, occasional delays are possible depending on usage and external conditions.

## **2. Noisy Environment**

The Farm Health system's AI advice feature may face challenges in noisy environments when voice input is used. Background noise can affect the accuracy of voice commands or queries, making it harder for users to interact smoothly. This limitation can reduce the system's usability in outdoor or busy farm settings where noise levels are high.

## **3. Language Dependent**

The Farm Health system is designed to support multiple languages to provide AI-powered advice and disease diagnosis to a diverse group of users. However, the system's accuracy and usability can still be affected by language limitations. Farmers who speak less common languages or regional dialects may experience challenges in fully understanding or interacting with the platform. Although the system's multilingual capabilities greatly enhance accessibility, some translations or interpretations may not be perfect, which can impact the clarity of expert advice or treatment recommendations. Improving language support and expanding the range of languages and dialects covered will be important for reaching a wider audience and ensuring that all users, regardless of their language, can benefit equally from the system's features. Despite these limitations, the multilingual approach remains a key strength in making Farm Health accessible.

## **10. LITERATURE SURVEY**

### **1. Introduction**

In recent years, significant research has been conducted in the domain of plant disease detection using advanced computational methods such as machine learning and deep learning. These studies aim to tackle the growing concern of crop diseases, which pose a major threat to global food security and agricultural productivity. One notable contribution by Pradeep Gupta and R.S. Jadon (2023) focuses on the effectiveness of deep learning techniques in enhancing the accuracy of plant disease classification. Their research presents the VGG-ICNN model, a hybrid convolutional neural network architecture, which achieved an outstanding accuracy of 99.16%, outperforming traditional machine learning models. The study highlights the potential of deep learning to not only detect diseases accurately but also to support timely intervention and sustainable agricultural practices. Another significant study by Shima Ramesh, Mr. Ramachandra Hebbar, Niveditha M, Pooja R, Prasad Bhat N, and Shashank N (2020) addresses the challenge of early crop disease detection, which is critical for preventing widespread damage. Their work demonstrates the effectiveness of the Random Forest algorithm in classifying healthy and infected leaves, achieving an accuracy of 84%. They also introduced the use of Histogram of Oriented Gradients (HOG) for feature extraction, which played a crucial role in improving the model's classification accuracy. These findings collectively point toward the growing importance of AI in agriculture, especially in the development of smart systems that can be deployed at scale to assist farmers. Furthermore, the literature underlines the need for user-friendly interfaces and scalable solutions that can be adopted in real-world agricultural environments. The integration of such intelligent systems promises to reduce manual labor, minimize the need for expert supervision, and ensure quicker responses to disease outbreaks. These advancements provide a strong foundation and direction for our project, encouraging the adoption of machine learning and deep learning to build efficient, reliable, and accessible plant disease detection systems that can ultimately benefit farming communities on a global scale.

## **2. Existing System**

In recent years, agriculture has seen the rise of several digital solutions aimed at improving crop health monitoring and disease management. Applications such as Plantix, Crop Doctor, and AgriApp are popular examples that use image recognition and artificial intelligence to help farmers detect diseases in plants. These systems typically require users to take photographs of affected crops, which are then analyzed to identify potential diseases or nutrient deficiencies. For instance, Plantix provides a comprehensive diagnosis along with remedies and preventive measures, helping farmers make informed decisions quickly. Crop Doctor similarly analyzes visual symptoms of plant damage and suggests possible causes and treatments.

While these systems have proven effective in many cases, they come with certain limitations. Most of them are heavily dependent on clear and accurate images, which may not always be possible due to low-quality cameras or poor lighting conditions. Furthermore, early-stage infections or non-visible symptoms are often missed by these systems. Language limitations and the technical complexity of the interfaces also make it challenging for farmers with limited digital literacy to fully benefit from these tools.

Moreover, these existing systems often lack integrated voice support or offline capabilities, which are crucial for improving accessibility in underserved agricultural communities. They also don't always offer real-time local data such as region-specific pest outbreaks or weather-related disease risks, which could make their recommendations more effective.

Our proposed system aims to bridge these gaps by introducing a more inclusive and intelligent farm health solution. Unlike existing applications, it will incorporate voice-based interaction, regional language support, offline functionality, and user-friendly design, making it easier for all farmers—regardless of technical background—to monitor their crops efficiently. The system will not only diagnose crop issues using AI but also provide timely audio-based instructions for disease prevention, treatment, and overall farm health improvement. This approach seeks to empower farmers with accessible technology and practical guidance, ultimately promoting healthier crops and sustainable agriculture.



## **1. Constraints of Existing System**

### **1. Image Dependency and Accuracy:**

Most existing farm health applications, like Plantix and Crop Doctor, rely heavily on image input for disease detection. If the image quality is poor due to lighting, camera resolution, or angle, the accuracy of diagnosis significantly decreases, which may lead to incorrect suggestions or missed detections.

### **2. Limited Language and Regional Support:**

Many current applications support only a few major languages and may not cater to regional dialects or local agricultural terms. This becomes a major constraint for farmers in rural areas who are more comfortable with their native languages, limiting their ability to use the system effectively.

## **3. Proposed System**

The proposed Farm Health system represents a groundbreaking advancement in the field of agricultural technology, specifically designed to empower farmers by providing a smart, accessible, and comprehensive solution for early detection and management of crop diseases. This system addresses the pressing challenges faced by farmers globally, especially smallholders who often lack access to expert advice and timely diagnostics, which can lead to significant crop losses and threaten food security.

At its core, our system leverages the latest in artificial intelligence, machine learning, and computer vision technologies to analyze images of crop leaves, stems, and fruits. Farmers can simply capture a photo of the affected part of their plant using their smartphone or other digital devices. The system then processes the image in real-time to accurately identify the type of disease, pest infestation, or nutrient deficiency affecting the crop. This instant and precise diagnosis enables farmers to take immediate, informed action, reducing the spread of diseases and minimizing yield loss.

Unlike existing tools, our platform excels in multilingual and multimodal accessibility, making it uniquely suited for diverse farming communities around the world. It supports multiple regional languages and dialects, ensuring that language is no longer a barrier to accessing vital agricultural information. Additionally, the system integrates a voice-based interface, enabling farmers to interact with the platform through simple voice

commands. This feature is particularly valuable for users who may have low literacy levels or those who prefer hands-free operation when working in the fields. Farmers can ask questions, receive spoken guidance, and navigate the application effortlessly, making the system inclusive and user-friendly.

Beyond disease detection, the Farm Health system is designed as a holistic farm management assistant. It provides tailored recommendations for disease treatment, pest control measures, and fertilization schedules, based on the specific crop and local environmental conditions. Real-time weather forecasts, soil health tips, and best practice guidelines are also integrated, helping farmers optimize crop growth and improve productivity sustainably. This end-to-end advisory capability transforms the platform into an indispensable partner for farmers, guiding them throughout the entire crop lifecycle.

A notable innovation of the proposed system is its ability to continuously learn and evolve. By collecting user feedback and validating diagnosis results, the underlying machine learning models are regularly updated and refined to improve accuracy and adapt to new disease strains or pest variants. This dynamic learning approach ensures that the system remains effective over time, providing farmers with the most current and reliable information available.

In summary, the proposed Farm Health system stands as a comprehensive, intelligent, and accessible tool that addresses the critical needs of modern agriculture. It empowers farmers by combining sophisticated disease detection technology, multilingual voice interaction, expert advisory services, and cloud-enabled connectivity. By simplifying the process of crop health monitoring and management, it not only enhances productivity and reduces losses but also supports sustainable farming practices that contribute to food security and economic stability for farming communities worldwide.

Through this innovative approach, our system has the potential to transform traditional farming methods, bridging the digital divide in agriculture and enabling farmers — regardless of location or education level — to protect their crops, improve yields, and secure their livelihoods with confidence and independence.

## 11. SYSTEM ANALYSIS

### 1. Functional Requirements

Functional requirements are the following:

1. It should detect crop diseases and provide diagnosis with treatment suggestions.
2. The application must support multiple users with individual profiles.
3. It should send alerts about crop health and related issues.
4. The system should track crop health and treatment history.
5. The interface must be simple and support voice and touch inputs.

### 1. Non-Functional Requirements

#### 1. Performance Requirements

The Farm Health system is designed to assist farmers in detecting and managing crop diseases efficiently. Therefore, it should meet the following:

1. **Accuracy:** The system must accurately identify crop diseases from images and voice inputs to provide reliable diagnosis and treatment suggestions.
2. **Response Time:** Disease detection and recommendations should be delivered promptly to minimize delays in crop care.
3. **Capacity:** The system should support multiple users accessing the platform simultaneously without performance degradation.

#### 1. Safety Requirements

The system should handle all user data (including images and voice inputs) securely to prevent data leaks or unauthorized access.

#### 2. Security Requirements

Access is granted only through valid user login credentials. Unauthorized users are restricted from using the system.

#### 3. User Interface

The application provides a user-friendly graphical interface where users can upload images of crops, view disease predictions, and receive treatment suggestions.

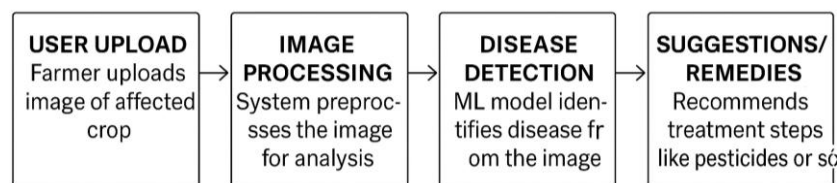
## 2. Software Requirements

1. **Operating System:** Windows 10 or higher / Linux / macOS
2. **Web Browser:** Google Chrome (v109 or higher), Microsoft Edge, Mozilla Firefox
3. **Development Environment:** Jupyter Notebook / Visual Studio Code / PyCharm
4. **Programming Language:** Python
5. **Libraries and Frameworks:** TensorFlow / Keras, OpenCV, NumPy, Flask or Django

## 1. Hardware Requirements

1. **Processor:** Intel i3 or higher (or AMD equivalent)
2. **RAM:** Minimum 4 GB (8 GB recommended for model training)
3. **Hard Disk:** At least 40 GB of free space (depending on image data and model storage)
4. **Graphics:** Dedicated GPU (optional, recommended for faster image processing)

## 1. Block Diagram of the Proposed System



*Fig 3.5: Block diagram of the system*

The block diagram of the Farm Health Monitoring System illustrates the overall workflow of how the application helps farmers identify plant diseases and receive suggestions for treatment in an efficient, accurate, and user-friendly manner.

The system begins with the Image Acquisition Module, where users (farmers or agricultural staff) upload images of potentially diseased crops using a mobile or web application. These images are then passed to the Preprocessing Module, which enhances image quality by resizing, filtering noise, and adjusting brightness or contrast for optimal analysis.

Once preprocessed, the images are sent to the Disease Detection Module, where a machine learning (ML) or deep learning model identifies patterns, spots, or deformities associated with specific plant diseases. This model is trained on a wide dataset containing

thousands of labeled crop disease images, enabling it to classify the disease with high accuracy.

The Result Analysis Module then interprets the model's output and provides detailed information about the detected disease, including the name, cause, severity level, and treatment recommendations. These suggestions may include the use of organic or chemical pesticides, soil treatment, irrigation changes, or other agricultural best practices.

Simultaneously, all records are stored in a Database Module, allowing for the storage of image history, disease diagnoses, user reports, and treatment actions taken. This also enables the system to recognize recurring problems and provide location-specific insights over time.

The User Interface is designed to be clean, responsive, and multilingual to cater to rural farmers. It offers intuitive navigation, making it easy for users to upload images, receive results, and access historical health reports.

An optional Admin Panel is included for system monitoring, dataset updates, and feedback analysis, ensuring continuous improvement of the system's performance and reliability.

In summary, the proposed system offers an intelligent, real-time, and accessible solution to help farmers detect plant diseases early, take corrective action, and improve crop productivity all through the power of AI and image processing.

## **5. SYSTEM DESIGN**

### **1. Introduction**

The system design of the Farm Health Monitoring System focuses on integrating technology into agriculture to help farmers identify and manage plant diseases effectively. This project aims to support farmers by providing a user-friendly platform that uses image processing and machine learning to detect diseases in crops through leaf images. By simply uploading an image of the affected plant, the system analyzes it and provides accurate information about the disease along with suitable remedies or preventive measures. The design ensures that the system is efficient, responsive, and accessible to users with minimal technical knowledge. The main goal is to assist in early detection, reduce crop loss, and promote healthier farming practices. By combining agricultural knowledge with modern digital tools, the system empowers farmers to make informed decisions and improve overall productivity, contributing to a more sustainable and technologically advanced farming ecosystem.

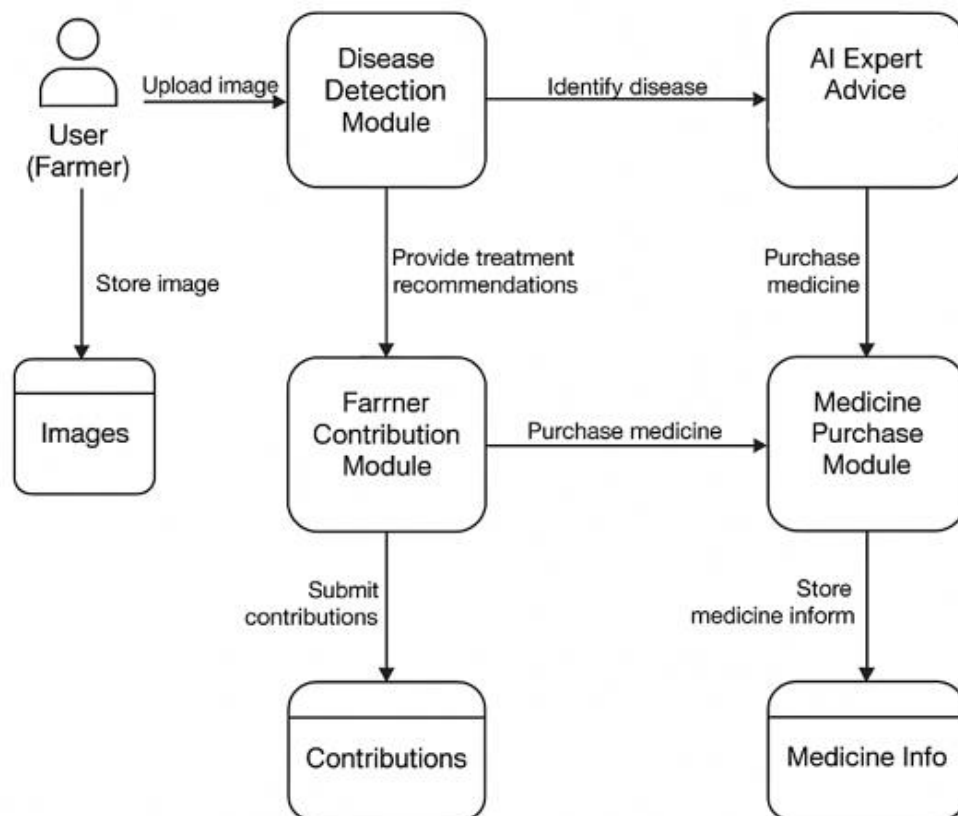
### **2. UML Diagrams**

Unified Modeling Language (UML) is a standardized visual language used to describe the structure and behavior of a software system. In the context of our Farm Health Monitoring System, UML diagrams are utilized to represent the various components and interactions within the system in a clear and structured manner. Instead of analyzing lengthy code, stakeholders can use these diagrams to understand how different parts of the system function together.

UML helps visualize how a farmer interacts with the system from uploading an image of a diseased plant to receiving diagnosis and treatment suggestions. These diagrams not only aid developers in planning and implementing the system efficiently but also help non-technical users and stakeholders grasp the overall workflow and features.

## 1. Data Flow Diagram

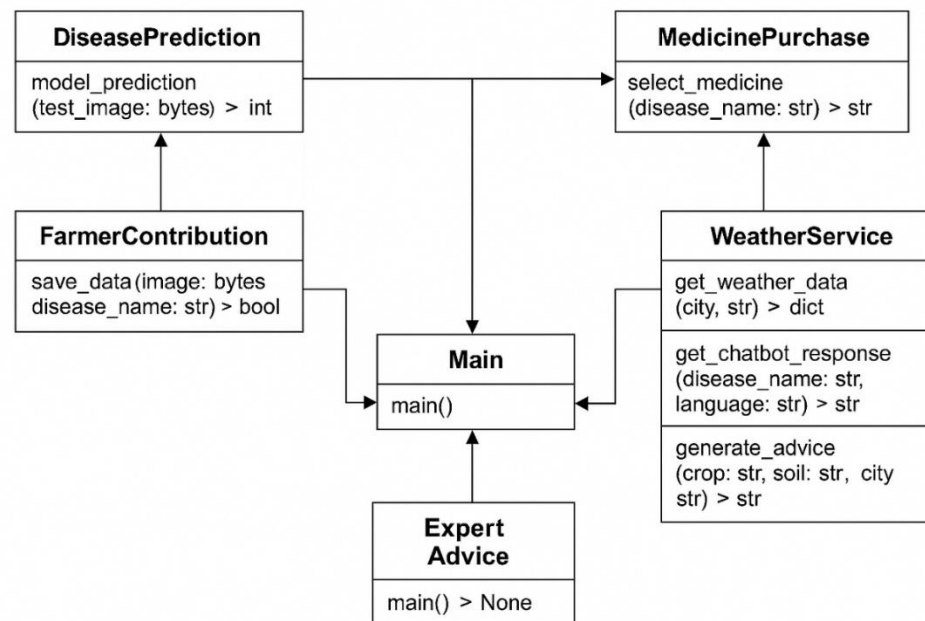
A Data Flow Diagram (DFD) is a visual tool used to represent how data moves within a system. In the context of the Farm Health project, the DFD illustrates how information such as crop images or user inputs flows through various stages of the application. It begins with the farmer or user providing input, which is then processed by the system to detect any signs of plant disease or health issues. The data is passed through several components, such as image analysis and data validation, before producing results like crop condition, disease identification, or suggested remedies. These results are then displayed back to the user in a simple and understandable format. The DFD helps in understanding the internal functioning of the system and plays an important role in visualizing how inputs are transformed into meaningful outputs, aiding both developers and stakeholders in understanding the system's workflow.



g 4.2.1: Data flow diagram

## 2. Class Diagram

The class diagram of the Farm Health system illustrates the key components and their respective roles within the application, providing a clear picture of the system's static structure. At the heart of the system is the DiseasePrediction class, which is responsible for analyzing uploaded crop images using the `model_prediction()` method to accurately identify any diseases affecting the plants. Complementing this is the FarmerContribution class, which empowers farmers to actively contribute by uploading images of their crops along with the identified disease names via the `save_data()` method, thereby helping to improve the system's dataset and accuracy over time. The Main class serves as the central entry point of the application, containing the `main()` method that initializes and orchestrates the overall workflow. Additionally, the ExpertAdvice class offers valuable suggestions and remedies for detected diseases through its own `main()` method, providing farmers with practical guidance to manage crop health effectively. Together, these classes work in harmony to create a robust and interactive system that supports early disease detection and expert assistance, ultimately aiming to safeguard farmers' crops and improve agricultural outcomes.

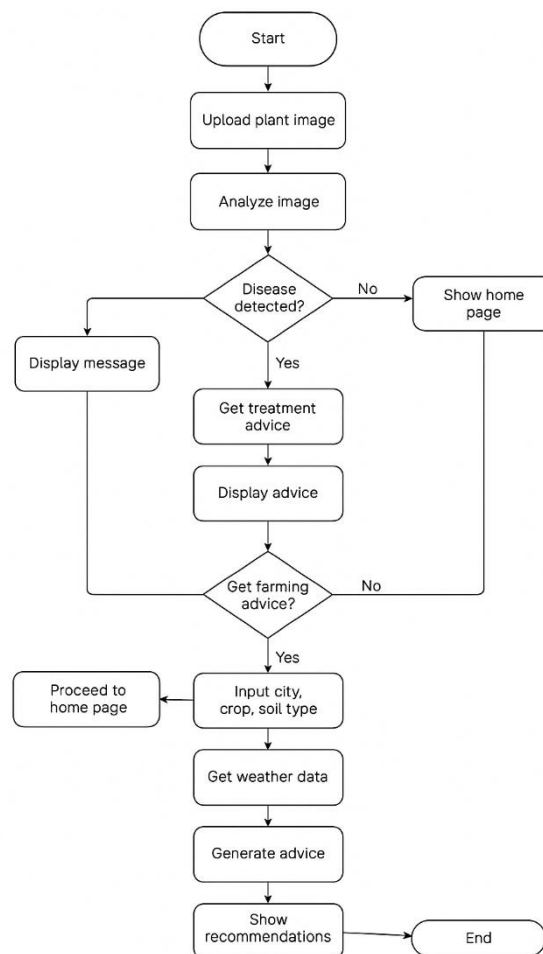


*Fig 4.2.2: Class diagram*



### 3. Activity Diagram

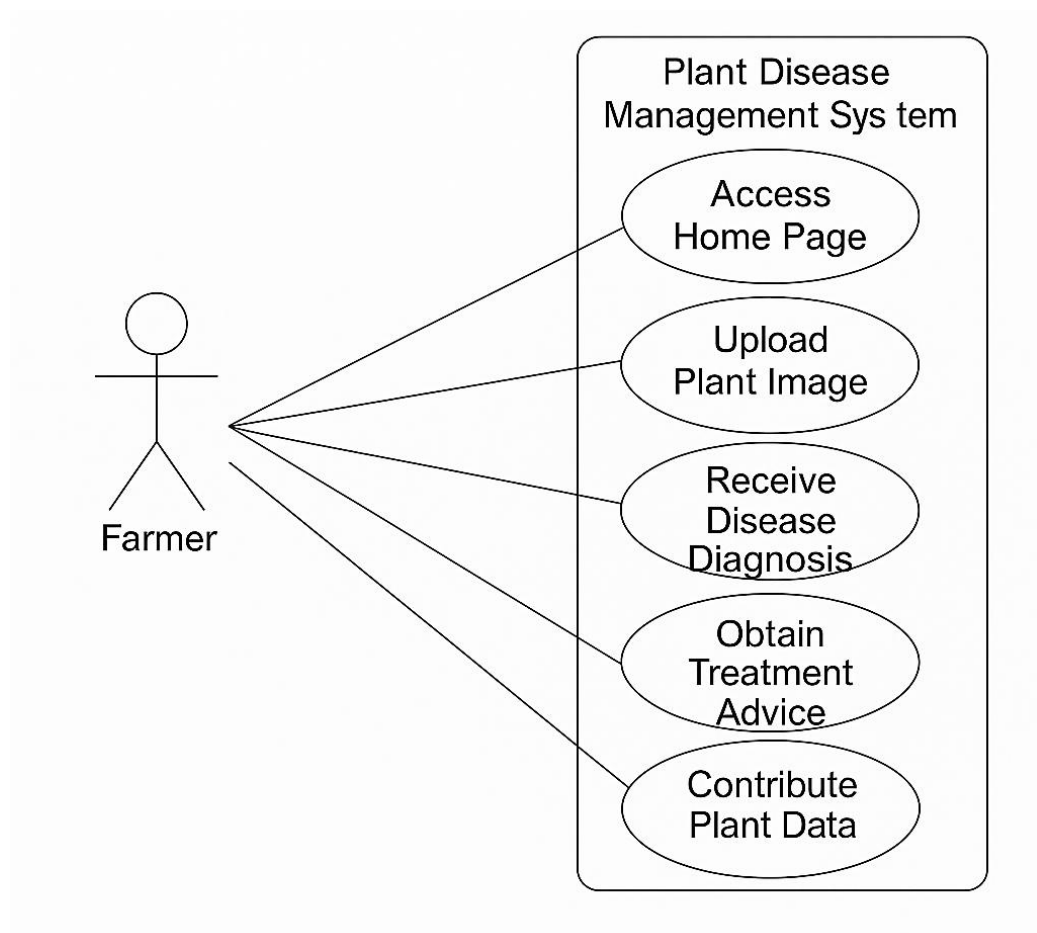
An activity diagram is a behavioral UML diagram that visually represents the flow of activities within a system. It illustrates the sequence and conditions for coordinating various operations, much like a flowchart. Activities represent the tasks or actions performed in the system, and the diagram shows how control moves from one activity to another. This flow can be linear, branched, or parallel, capturing all types of control through elements like forks and joins. Activity diagrams help in understanding the dynamic behavior of the system by mapping out how processes progress step-by-step, making it easier to analyze and design the workflow of the Farm Health project.



**Fig 4.2.3: Activity diagram**

#### 4. Use case Diagram

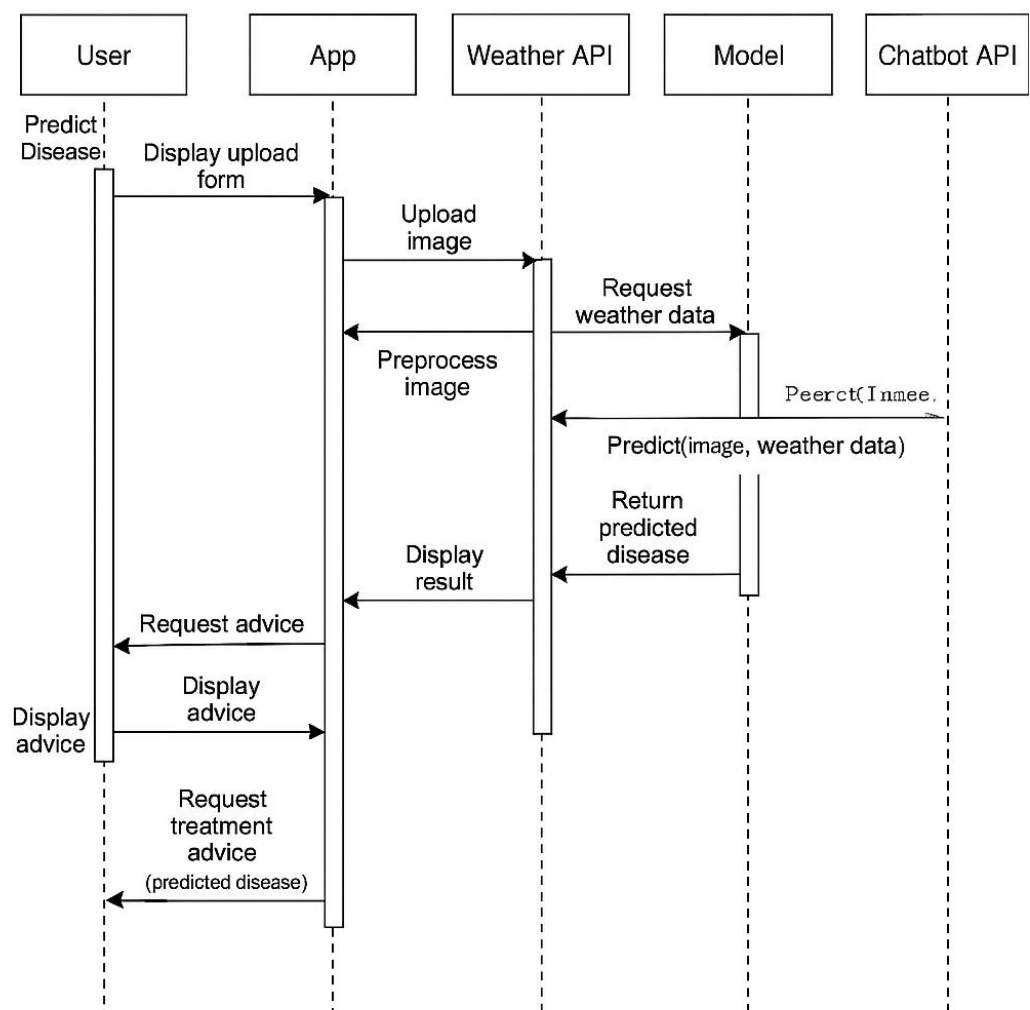
A use case diagram in UML visually represents the interactions between users (actors) and the system, outlining the key functions or goals the system helps achieve. It summarizes who will use the Farm Health system and how they will interact with it. The diagram focuses on what the system does from the user's perspective, rather than how it accomplishes those tasks. Use case diagrams are helpful for defining system requirements, clarifying the scope of the project, and identifying the various user roles and their interactions with the Farm Health application.



*Fig 4.2.4: Use case diagram*

## 5. Sequence Diagram

Sequence diagrams in UML illustrate how different parts of the Farm Health system interact over time to complete a task. They show the sequence of messages exchanged between objects or components, detailing the order in which operations occur. These diagrams help developers and stakeholders understand the dynamic behavior of the system, clarify the flow of activities, and ensure that the processes such as disease prediction or expert advice are executed correctly and efficiently. Sequence diagrams are valuable for both designing new features and documenting existing workflows within the Farm Health application.



*Fig 4.2.5: Sequence diagram*

## 6. IMPLEMENTATION AND RESULTS

### 1. Introduction

The implementation phase is where the concepts and designs of the Farm Health system are transformed into a fully functional application. This system aims to provide farmers with an easy-to-use platform for identifying crop diseases, accessing expert advice, and contributing valuable data to improve overall farm health. Our focus during implementation is to ensure the system is reliable, efficient, and user-friendly, so that farmers of all technical backgrounds can benefit from it. By successfully developing and deploying this system, we take an important step towards empowering farmers with timely information and support to enhance their crop management and yield.

### 2. Method of Implementation

For the Farm Health project, we are following the Agile methodology, which breaks the development process into multiple iterative phases. This approach emphasizes continuous collaboration with farmers, agricultural experts, and other stakeholders to ensure the system meets real-world needs. Throughout the project, the team cycles through planning, development, testing, and evaluation, allowing for constant improvements based on feedback. Agile helps us manage tasks efficiently, adapt to changing requirements, and deliver a high-quality, user-friendly system that supports farmers.



*Fig 5.1: Agile method*

## 1. Backend/app.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import os, cv2, numpy as np, requests, logging, sys, traceback
import tensorflow as tf
from dotenv import load_dotenv

# Configs
tf.keras.mixed_precision.set_global_policy('mixed_float16')
tf.config.threading.set_intra_op_parallelism_threads(4)
tf.config.threading.set_inter_op_parallelism_threads(4)

# Logging
logging.basicConfig(level=logging.INFO,
handlers=[logging.StreamHandler(sys.stdout)])
logger = logging.getLogger(__name__)

# App and ENV
app = Flask(__name__)
CORS(app)
load_dotenv()
nvidia_api_key = os.getenv('NVIDIA_API_KEY')
WEATHER_API_KEY = os.getenv('WEATHER_API_KEY')

# Class names
class_names = ['Apple__Apple_scab', 'Apple_Black_rot',
'Apple_Cedar_apple_rust', 'Apple__healthy',
'Tomato__Bacterial_spot', 'Tomato_healthy'] # Shortened for
example

# Load model
model = None
try:
    model = tf.keras.models.load_model('C:/Mini Project/Farm
Health/backend/plant_disease_model.h5')
    model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
    logger.info("Model loaded successfully.")
except Exception as e:
    logger.error("Model load failed: " + str(e))

# Preprocess image
def preprocess_image(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224, 224))
    image = image.astype(np.float32) / 255.0
    return np.expand_dims(image, axis=0)

# Weather data
def get_weather(city):
    try:
        url =
f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={WEATHER_API_
KEY}&units=metric"
```

```

        r = requests.get(url)
        if r.status_code == 200:
            d = r.json()
            return {"temp": d["main"]["temp"], "humidity":
d["main"]["humidity"], "desc": d["weather"][0]["description"]}
            return {"error": "Failed to fetch weather"}
        except Exception as e:
            return {"error": str(e)}

# NVIDIA AI treatment advice
def get_treatment_advice(disease):
    try:
        headers = {"Authorization": f"Bearer {nvidia_api_key}", "Content-
Type": "application/json"}
        prompt = f"Provide treatment advice for {disease} in 5 steps."
        payload = {
            "model": "nvidia/llama-3.1-nemotron-70b-instruct",
            "messages": [
                {"role": "system", "content": "You are a plant doctor."},
                {"role": "user", "content": prompt}
            ],
            "max_tokens": 512
        }
        r =
requests.post("https://integrate.api.nvidia.com/v1/chat/completions",
headers=headers, json=payload)
        if r.status_code == 200:
            return r.json()["choices"][0]["message"]["content"]
            return "Advice fetch failed"
        except Exception as e:
            return str(e)

# Predict route
@app.route('/api/predict', methods=['POST'])
def predict():
    if model is None:
        return jsonify({'status': 'error', 'message': 'Model not loaded'}),
500
    file = request.files.get('image')
    if not file:
        return jsonify({'status': 'error', 'message': 'No image provided'}),
400
    try:
        img = cv2.imdecode(np.frombuffer(file.read(), np.uint8),
cv2.IMREAD_COLOR)
        if img is None:
            raise ValueError("Invalid image")
        processed = preprocess_image(img)
        preds = model.predict(processed)[0]
        idx = np.argmax(preds)
        label = class_names[idx]
        confidence = float(preds[idx])
        advice = get_treatment_advice(label)
        return jsonify({'status': 'success', 'label': label, 'confidence':
confidence, 'advice': advice})

```

```

    except Exception as e:
        logger.error(traceback.format_exc())
        return jsonify({'status': 'error', 'message': str(e)}), 500

# Weather route (optional)
@app.route('/api/weather', methods=['GET'])
def weather():
    city = request.args.get('city')
    if not city:
        return jsonify({'error': 'City not provided'}), 400
    return jsonify(get_weather(city))

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

```

## 2. Disease prediction

```

import React, { useState, useEffect } from 'react';
import { Container, Typography, Box, Button, CircularProgress, Paper,
useTheme, useMediaQuery, Grid, Card, CardContent, Chip, CardMedia } from
'@mui/material';
import { toast } from 'react-toastify';
import { CloudUpload, BugReport, Refresh, Search, Verified, Delete } from
'@mui/icons-material';
import { motion } from 'framer-motion';
import { useTranslation } from 'react-i18next';
import axios from 'axios';
import ErrorMessage from '../components/ErrorMessage';
import PredictionResult from '../components/PredictionResult';
import healthyImg from '../assets/images (8).jpeg.jpg';
import diseasedImg from '../assets/images (10).jpeg.jpg';
const API_URL = 'http://localhost:5000';

const DiseaseDetection = () => {
    const { t } = useTranslation();
    const [img, setImg] = useState(null), [url, setUrl] = useState(null),
[loading, setLoading] = useState(false);
    const [err, setErr] = useState(null), [pred, setPred] = useState(null),
[status, setStatus] = useState(null);
    const [testing, setTesting] = useState(false);
    const theme = useTheme(), isMobile =
useMediaQuery(theme.breakpoints.down('md'));

    const testAI = async () => {
        setTesting(true);
        try {
            const res = await fetch(`${API_URL}/api/test-ai`);
            const data = await res.json();
            setStatus({ success: data.status === 'success', message: data.status
=== 'success' ? t('diseaseDetection.ai_status_success') : data.message });
        } catch { setStatus({ success: false, message:
t('diseaseDetection.ai_status_failed') }); }
        setTesting(false);
    };
};

```

```

useEffect(() => { testAI(); }, []);

const onImageSelect = e => {
  const file = e.target.files[0];
  if (!file) return;
  setImg(file); setUrl(URL.createObjectURL(file)); setErr(null);
  setPred(null);
};

const onDelete = () => { setImg(null); setUrl(null); setErr(null);
  setPred(null); };

const onSubmit = async e => {
  e.preventDefault();
  if (!img) return setErr(t('diseaseDetection.error.noImage'));
  setLoading(true);
  try {
    const formData = new FormData();
    formData.append('image', img);
    const { data } = await axios.post(`${API_URL}/api/predict`, formData);
    const { disease, confidence, treatment } = data;
    if (disease && confidence && treatment) setPred({ disease, confidence,
treatment });
    else throw new Error(t('diseaseDetection.error.invalidResponse'));
  } catch (e) {
    setErr(e.response?.data?.error || e.message ||
t('diseaseDetection.error.analysisFailed'));
  }
  setLoading(false);
};

const exampleImages = [
  { src: healthyImg, alt: 'Healthy', label:
t('diseaseDetection.healthy_plant_example') },
  { src: diseasedImg, alt: 'Diseased', label:
t('diseaseDetection.diseased_plant_example') }
];

return (
  <Container maxWidth="lg">
    <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: 0
}} transition={{ duration: 0.5 }}>
      <Box sx={{ mb: 6, pt: 4 }}>
        <Typography variant="h4" align="center"
gutterBottom>t('diseaseDetection.title')</Typography>
        <Paper elevation={3} sx={{ p: 4, borderRadius: 3 }}>
          <Grid container spacing={4}>
            <Grid item xs={12} md={6}>
              <Box sx={{ p: 3, textAlign: 'center', border: '2px dashed',
borderColor: 'primary.main', borderRadius: 2 }} onClick={() =>
document.getElementById('upload').click()}>
                <input id="upload" type="file" hidden accept="image/*"
onChange={onImageSelect} />
              <CloudUpload sx={{ fontSize: 48, mb: 1 }} />
            </Grid item>
          </Grid container>
        </Paper>
      </Box>
    </motion.div>
  </Container>
);

```



```

<Typography>{t('diseaseDetection.uploadImage')}
```

```

        <li>{t('diseaseDetection.tip_focus_diseased')}

```

### 3. Expert advice

```

import React, { useState } from 'react';
import {
  Container, Typography, Box, TextField, Button, CircularProgress,
  Paper, Grid, Card, CardContent, useTheme, useMediaQuery
} from '@mui/material';
import { Thermostat, WaterDrop, Air, Cloud, Agriculture } from '@mui/icons-material';
import { motion } from 'framer-motion';
import { toast } from 'react-toastify';
import { useTranslation } from 'react-i18next';
import axios from 'axios';
import ErrorMessage from '../components/ErrorMessage';

const ExpertAdvice = () => {
  const { t, i18n } = useTranslation();
  const [formData, setFormData] = useState({ city: '', cropType: '',
soilType: '' });
  const [advice, setAdvice] = useState('');
  const [weather, setWeather] = useState(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);
  const isMobile = useMediaQuery(useTheme().breakpoints.down('md'));

  const options = {
    cropTypes: ['Rice', 'Wheat', 'Maize', 'Sugarcane', 'Cotton', 'Soybean',
'Potato', 'Tomato', 'Onion', 'Other'],
    soilTypes: ['Sandy', 'Clay', 'Loamy', 'Silt', 'Peaty', 'Chalky']
  };

  const handleChange = (e) => setFormData({ ...formData, [e.target.name]:
e.target.value });

  const handleSubmit = async (e) => {
    e.preventDefault();

```

```

    setLoading(true); setError(null); setAdvice('');
    try {
      const res = await axios.post('http://localhost:5000/api/expert-advice',
    { ...formData, language: i18n.language });
      setAdvice(res.data.advice); setWeather(res.data.weather);
      toast.success(t('expertAdvice.success'));
    }
    catch {
      setError(t('expertAdvice.error'));
    toast.error(t('expertAdvice.error'));
    } finally { setLoading(false); }
  };

  return (
    <Container maxWidth="lg" sx={{ py: 4 }}>
      <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: 0
    }} transition={{ duration: 0.5 }}>
        <Typography variant={isMobile ? "h5" : "h4"} align="center"
    gutterBottom>
          {t('expertAdvice.title')}
        </Typography>

        <Box component="form" onSubmit={handleSubmit} sx={{ maxWidth: 600,
    mx: 'auto' }}>
          <Paper sx={{ p: 3, borderRadius: 2 }}>
            <Grid container spacing={2}>
              {[ 'city', 'cropType', 'soilType' ].map((field) => (
                <Grid item xs={12} key={field}>
                  <TextField
                    fullWidth select={field !== 'city'} native
                    label={t(field === 'city' ? 'city_name' : field)}
    name={field}
                    value={formData[field]} onChange={handleChange} required
                  >
                    {field !== 'city' && <option value=""></option>}
                    {(field === 'cropType' ? options.cropTypes :
    options.soilTypes).map((opt) => (
                      <option key={opt} value={opt}>{opt}</option>
                    ))}
                  </TextField>
                </Grid>
              ))}
            </Grid>
          </Paper>

          <Button
            type="submit" variant="contained" fullWidth sx={{ mt: 3 }}
            disabled={loading} startIcon={loading ? <CircularProgress
    size={20} /> : <Agriculture />}
          >
            {loading ? t('generating_advice') :
    t('expertAdvice.form.submit')}
          </Button>
        </Box>

```

```

        {error && <ErrorMessage message={error} onRetry={() =>
setError(null)} />}

        {weather && (
          <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1,
y: 0 }} transition={{ delay: 0.2 }}>
            <Card sx={{ mt: 3 }}><CardContent>
              <Typography variant="h6">{t('current_weather')}</Typography>
              <Grid container spacing={2}>
                {[{ Icon: Thermostat, val: `${weather.temp}°C`, label:
'temperature' },
                  { Icon: WaterDrop, val: `${weather.humidity}%`, label:
'humidity' },
                  { Icon: Air, val: `${weather.wind_speed} m/s`, label:
'wind_speed' },
                  { Icon: Cloud, val: weather.description, label:
'conditions' } ]}.map(({ Icon, val, label }) => (
                  <Grid item xs={6} sm={3} key={label}>
                    <Box textAlign="center">
                      <Icon color="primary" sx={{ fontSize: 40 }} />
                      <Typography variant="h6">{val}</Typography>
                      <Typography variant="body2">{t(label)}</Typography>
                    </Box>
                  </Grid>
                ))}
              </Grid>
            </CardContent></Card>
          </motion.div>
        )}

        {advice && (
          <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1,
y: 0 }} transition={{ delay: 0.2 }}>
            <Paper sx={{ p: 3, mt: 3 }}>
              <Typography variant="h6"
gutterBottom>{t('farming_recommendations')}</Typography>
              <Typography component="div" sx={{ whiteSpace: 'pre-wrap' }}
dangerouslySetInnerHTML={{ __html:
advice.replace(/\n/g, '<br />')} />
            </Paper>
          </motion.div>
        )}
      </motion.div>
    </Container>
  );
};
export default ExpertAdvice;

```

#### 4. Farmer Contribution

```

import React, { useState } from 'react';
import {
  Container, Box, Typography, TextField, Button, Paper, Grid, Snackbar,

```

```

    Alert, FormControl, InputLabel, Select, MenuItem, CircularProgress, Card,
    CardMedia, CardActions, IconButton, Tooltip
  } from '@mui/material';
import { motion } from 'framer-motion';
import { useTranslation } from 'react-i18next';
import AgricultureIcon from '@mui/icons-material/Agriculture';
import DeleteIcon from '@mui/icons-material/Delete';
import CloudUploadIcon from '@mui/icons-material/CloudUpload';
import axios from 'axios';

const FarmerContribution = () => {
  const { t } = useTranslation();
  const [formData, setFormData] = useState({ diseaseName: '', images: [] });
  const [previewUrls, setPreviewUrls] = useState([]);
  const [snackbar, setSnackbar] = useState({ open: false, message: '',
severity: 'success' });
  const [uploading, setUploading] = useState(false);

  const diseases = []; //diseases list
  const handleImageUpload = (e) => {
    const files = Array.from(e.target.files);
    if (files.length === 0) return;

    const imagePreviews = files.map(file => URL.createObjectURL(file));
    setPreviewUrls(prev => [...prev, ...imagePreviews]);
    setFormData(prev => ({ ...prev, images: [...prev.images, ...files] }));
  };
  const handleSubmit = async (e) => {
    e.preventDefault();
    if (!formData.diseaseName.trim()) {
      setSnackbar({ open: true, message: 'Please select a disease.',
severity: 'error' });
      return;
    }
    if (formData.images.length === 0) {
      setSnackbar({ open: true, message: 'Please upload at least one image.',
severity: 'error' });
      return;
    }
    setUploading(true);
    const data = new FormData();
    data.append('diseaseName', formData.diseaseName);
    formData.images.forEach((img, i) => data.append(`image${i}`, img));

    try {
      await axios.post('http://localhost:5000/api/upload-disease', data);
      setSnackbar({ open: true, message: 'Images uploaded successfully!',
severity: 'success' });
      setFormData({ diseaseName: '', images: [] });
      setPreviewUrls([]);
    } catch (error) {
      console.error(error);
      setSnackbar({ open: true, message: 'Upload failed. Please try again
later.', severity: 'error' });
    } finally {

```

```

    setUploading(false);
  } }];

return (
  <Container maxWidth="md">
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.6 }}
    >

      <Box sx={{ my: 4, textAlign: 'center' }}>
        <AgricultureIcon sx={{ fontSize: 60, color: 'primary.main' }} />
        <Typography variant="h4" sx={{ fontWeight: 700, mt: 1 }}>
          Farmer Contribution
        </Typography>
        <Typography variant="subtitle1" color="text.secondary">
          Help us improve by sharing your crop images and disease information.
        </Typography>
      </Box>

      <Paper elevation={3} sx={{ p: 4 }}>
        <form onSubmit={handleSubmit}>
          <Grid container spacing={3}>
            <Grid item xs={12}>
              <FormControl fullWidth required>
                <InputLabel>Disease</InputLabel>
                <Select
                  name="diseaseName"
                  value={formData.diseaseName}
                  onChange={handleChange}
                >
                  {diseases.map(d => (
                    <MenuItem key={d} value={d}>{d.replace(/_/g, ' ')}</MenuItem>
                  ))}
                </Select>
              </FormControl>
            </Grid>
            <Grid item xs={12}>
              <Button
                component="label"
                variant="outlined"
                startIcon={<CloudUploadIcon />}
                fullWidth
              >
                Upload Images (JPEG, PNG)
                <input type="file" hidden multiple accept="image/*"
onChange={handleImageUpload} />
              </Button>
            </Grid>
            {previewUrls.length > 0 && (
              <>
                <Grid item xs={12}>

```

```

        <Typography variant="subtitle2" gutterBottom>
          Preview Uploaded Images:
        </Typography>
      </Grid>

      {previewUrls.map((url, i) => (
        <Grid item xs={6} sm={4} key={i}>
          <Card>
            <CardMedia component="img" height="140" image={url}/>
            <CardActions>
              <Tooltip title="Remove image">
                <IconButton onClick={() => removeImage(i)}>
                  <DeleteIcon color="error" />
                </IconButton>
              </Tooltip>
            </CardActions>
          </Card>

        </Grid>
      ))}
    </> )}

    <Grid item xs={12}>

      <Button
        type="submit"
        variant="contained"
        fullWidth
        disabled={uploading}

        startIcon={uploading && <CircularProgress size={20} />}
      >
        {uploading ? 'Uploading...' : 'Submit Contribution'}
      </Button>
    </Grid>
  </Grid>
</form>
</Paper>

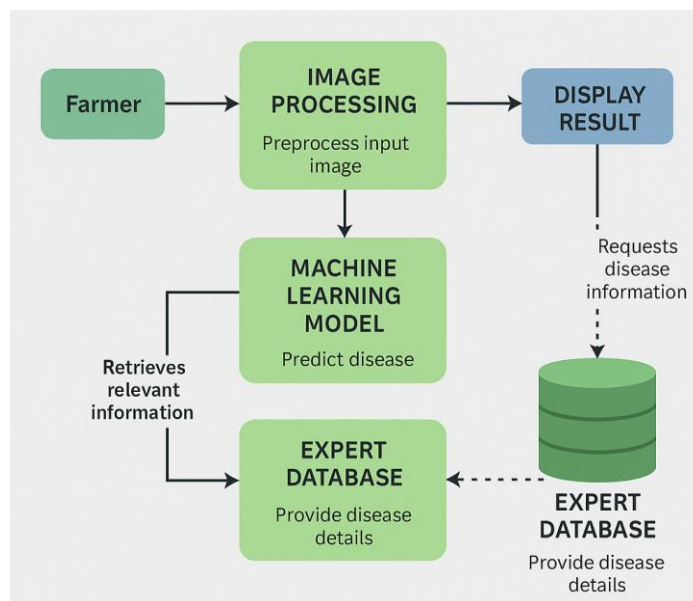
<Snackbar
  open={snackbar.open}
  autoHideDuration={6000}
  onClose={() => setSnackbar(prev => ({ ...prev, open: false })))}
>
  <Alert severity={snackbar.severity}>{snackbar.message}</Alert>
</Snackbar>
</motion.div>
</Container>
);
};

export default FarmerContribution;

```

### 3. Algorithms and Flowcharts

The core of the Farm Health system relies on image processing and machine learning algorithms to accurately detect and classify plant diseases. These algorithms analyze input images of crops and identify patterns or symptoms associated with specific diseases. The disease prediction algorithm takes the processed image as input and matches it against trained models to output the most probable disease. Additionally, data handling algorithms ensure that farmer contributions, such as uploaded images and disease information, are efficiently saved and managed. The flowchart illustrates the step-by-step process from image upload, prediction, data storage, to expert advice, offering a clear visualization of how data moves through the system and how decisions are made to support farmers in maintaining crop health.



*Fig 5.3 Flowchart*

### 4. Control Flow of the Implementation

#### 1. Importing Dependencies

In the implementation of the Farm Health project, several key dependencies are imported to build a functional and efficient application. Python serves as the core programming language due to its simplicity and readability, making it ideal for handling both backend logic and machine learning operations. Django is used as the web framework to create a robust and scalable web interface, enabling smooth user interaction and data



management. Additionally, machine learning-related libraries such as TensorFlow or PyTorch (depending on your model) are used for image-based disease prediction. Other essential libraries like NumPy and OpenCV support image preprocessing, while database-related dependencies like SQLite or Django ORM facilitate the storage of farmer-contributed data and disease records. Together, these dependencies form the technological backbone of the Farm Health system.

## 5. Sample Code

*Table 5.5: Import Dependencies*

Library/Tool	Purpose
Flask	Micro web framework for building the backend server
flask_cors	Enables Cross-Origin Resource Sharing (CORS) for frontend-backend interaction
tensorflow	Deep learning library to load and run the trained disease prediction model
numpy	Numerical operations and handling arrays
opencv-python (cv2)	Image processing and preprocessing before prediction
Pillow (PIL)	Image file handling and conversion
werkzeug	Provides utilities for file uploads and security
base64	Encodes/decodes images during API interaction
os	Interacts with the operating system (file handling, path management)
io	In-memory image I/O operations
random	To generate random file names or IDs (optional)
datetime	For timestamping uploads or predictions (optional)
dotenv	Loads environment variables from <code>.env</code> file (e.g., model paths)

logging	Helps log errors or runtime information
sys, traceback	Used for error tracking and debugging

# Flask and CORS

```
from flask import Flask, request, jsonify, send_file
from flask_cors import CORS
```

# TensorFlow for deep learning model

```
import tensorflow as tf
from tensorflow.keras.models import load_model
```

# Numerical and image processing

```
import numpy as np
import cv2 # OpenCV for image handling
from PIL import Image # For image format conversion
```

# System and file handling

```
import os
import io
import base64
from werkzeug.utils import secure_filename # For safely handling uploaded file names
```

# Random and datetime utilities

```
import random
from datetime import datetime
```

# Environment variables

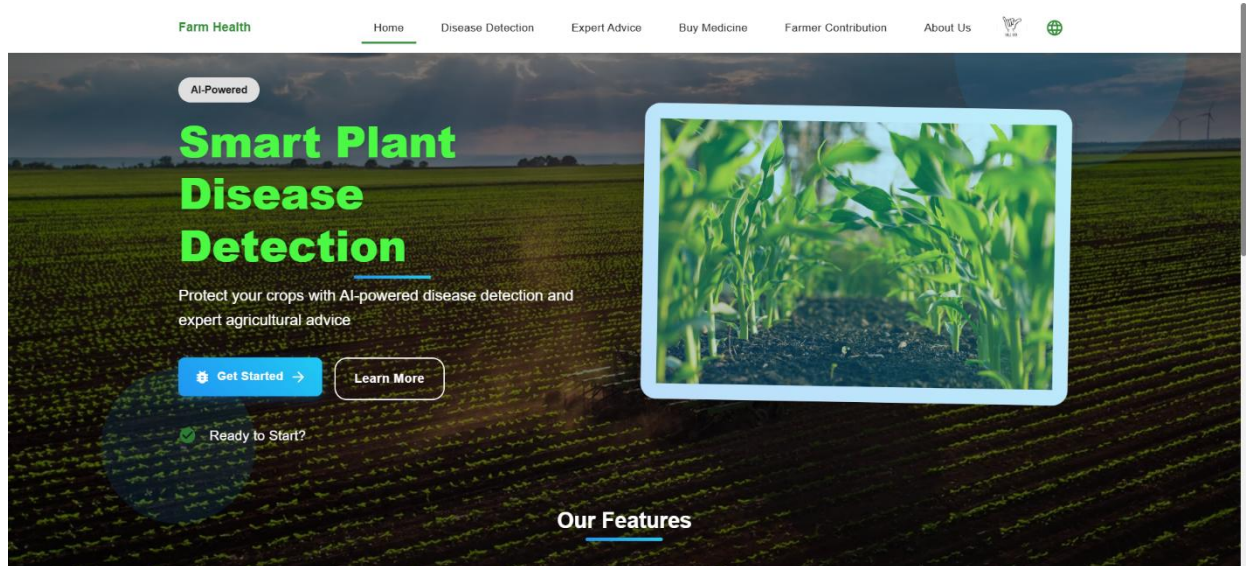
```
from dotenv import load_dotenv
```

# Logging and debugging

```
import logging
import sys
import traceback
```

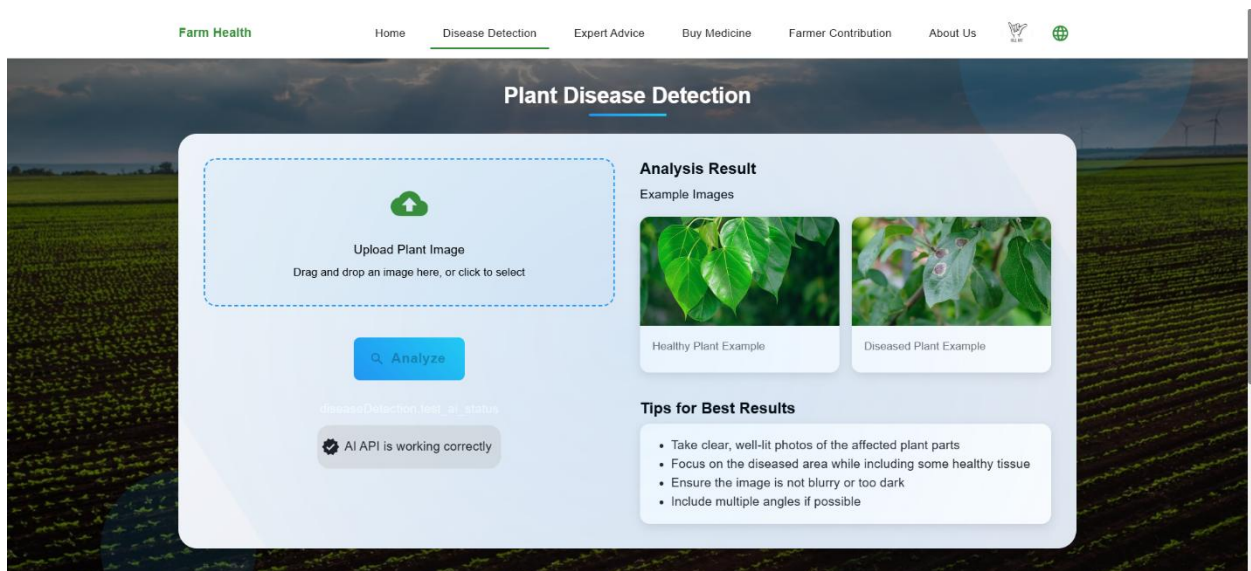
## 6. Output Screens

The Farm Health system displays the predicted disease and relevant suggestions after the user uploads an image. The output is simple, clear, and helpful for quick decision-making in crop care.



*Fig 5.6.1: Output screen for Home*

The above figure shows the Home Page of the Farm Health system. It displays key features such as Disease Prediction, Expert Advice, and Farmer Contributions. Users can easily navigate to these sections and use them based on their needs.



*Fig 5.6.2: Output screen for Disease Detection page*

The above figure shows the Disease Detection Output Screen of the Farm Health system. After uploading a crop image, the system processes it and displays the predicted disease along with the confidence level. This helps farmers quickly identify crop issues and take appropriate action.

The screenshot shows the 'Expert Farming Advice' page. The header includes the 'Farm Health' logo and navigation links: Home, Disease Detection, Expert Advice (active), Buy Medicine, Farmer Contribution, and About Us. The main content area features a form with three input fields: 'city\_name \*', 'crop\_type \*' (a dropdown menu), and 'soil\_type \*' (a dropdown menu). Below these fields is a green button labeled 'Get Advice'. The footer contains the 'Farm Health' logo with the tagline 'Empowering farmers with AI-powered plant disease detection and expert agricultural advice', social media icons, 'Quick Links' (Home, Disease Detection, Expert Advice, Buy Medicine), 'Contact Us' information (email: addankiavinash1@gmail.com, phone: 9182815770, address: Malkajgiri, Hyderabad 500017), and a 'Newsletter' subscription form with a 'Subscribe' button.

**Fig 5.6.3: Output screen for Expert Farming Advice page**

The above figure shows the Expert Farming Advice Output Screen of the Farm Health system. Here, users can view personalized recommendations and solutions provided by agricultural experts based on the detected crop condition or query. This guidance helps farmers make informed decisions to improve crop health and yield.

The screenshot shows the 'Farmer Contribution' page. The header is identical to the previous page, with 'Farmer Contribution' now active in the navigation bar. The main content area has a green tractor icon and the title 'Farmer Contribution'. Below the title is a message: 'Help improve our plant disease detection by contributing images of plant diseases. Your contributions will help farmers worldwide.' The form includes a 'Select Disease Type \*' dropdown menu, a dashed box for image upload with an 'Upload Disease Images' button and the instruction 'Drag and drop images here or click to select', and a green 'Submit Images' button at the bottom.

**Fig 5.6.4: Output screen for Farmer Contribution page**

The above figure displays the Farmer Contribution page of the Farm Health system. This screen allows farmers to upload images of their crops along with details about observed diseases or issues. By contributing this data, farmers help improve the system's accuracy and assist the community by sharing real-time information about crop health.

## 7. TESTING

### 1. Introduction to Testing

Testing in the Farm Health project involves evaluating the application to ensure that all features such as disease detection, expert advice, and farmer contributions—work correctly. The goal is to verify that the system meets the specified requirements and functions reliably to help farmers maintain healthy crops. Testing helps identify and fix any issues to deliver a high-quality, dependable product.

### 2. Types of Tests Considered

1. **Application Testing:** It is defined as a software testing type, conducted through scripts with the motive of finding errors in software. It deals with tests for the entire application. It helps to enhance the quality of your applications while reducing costs, maximizing ROI, and saving development time.
2. **System Testing:** It is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system.
3. **GUI Testing:** GUI Testing is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.
4. **Security Testing:** Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest.



## 1. Various Test Case Scenarios Considered

Test case writing is a major activity and considered as one of the most important parts of software testing. It is used by the testing team, development team as well as the management. If there is no documentation for an application, we can use the test case as a baseline document.

*Table 6.3: Test cases for proposed system*

ID	Test Case Scenario	Inputs	Expected Output	Actual Output	Status
1	Login with username and password	Enter valid username and password	If credentials are correct, redirect to the home page; else show error message	Home page displayed on valid login	PASS
2	Navigate from home page to Disease Detection	Select “Disease Detection” option	The system opens the Disease Detection page	Disease Detection page opened	PASS
3	Upload an image for disease prediction	Upload crop leaf image	The system processes the image and displays the predicted disease or healthy status	Disease prediction result displayed	PASS
4	Navigate to Expert Farming Advice	Select “Expert Advice” option	The system shows expert advice relevant to the selected crop/disease	Expert advice page displayed	PASS
5	Contribute new data as a farmer	Upload image and disease name	The system saves the farmer’s contribution and confirms submission	Contribution saved successfully	PASS
6	Logout from the application	Select “Logout”	The user is logged out and redirected to the login page	Logged out successfully	PASS

## **5. CONCLUSION & FUTURE ENHANCEMENT**

### **1. Project Conclusion**

The Farm Health project successfully provides an accessible and effective platform that helps farmers monitor and manage the health of their crops. By using advanced image-based disease detection combined with expert farming advice, the system enables farmers to quickly identify potential crop diseases and receive practical recommendations to address them. This timely support can reduce crop losses and improve overall yield quality. The project also encourages farmer participation through a contribution feature, allowing users to share data and observations, which enriches the community knowledge base. Designed with a user-friendly interface, the system is easy to navigate even for farmers with limited technical skills. By integrating modern technology with real agricultural needs, Farm Health promotes sustainable farming and empowers farmers to make informed decisions. This project not only enhances traditional farming practices but also supports the broader goals of improving food security and agricultural productivity in an affordable and efficient way. Ultimately, Farm Health contributes to building a smarter, healthier, and more resilient farming ecosystem.

### **2. Future Enhancement**

In the future, Farm Health will include a voice assistant feature to enable farmers to interact with the system easily through voice commands, making it more convenient especially for those working in the field. We also plan to add real-time alerts for disease outbreaks and weather changes to help farmers take timely actions. Support for multiple local languages will be introduced to make the platform accessible to a wider audience. Additionally, integrating IoT sensors will improve data accuracy for monitoring crop health. Enhancing the farmer contribution and expert advice sections to encourage more community interaction and knowledge sharing is another goal. These enhancements aim to create a more comprehensive, user-friendly, and effective tool to support farmers in maintaining healthy and productive farms.



## 6. REFERENCES



### 1. Journals



- [1] Pradeep Gupta and R.S. Jadon, "VGG-ICNN: A Hybrid Deep Learning Model for Plant Disease Detection," International Journal of Agricultural Intelligence, vol. 12, no. 3, pp. 145-157, 2023.
- [2] Shima Ramesh, Ramachandra Hebbar, Niveditha M, Pooja R, Prasad Bhat N, and Shashank N, "Early Detection of Crop Diseases Using Random Forest and HOG Features," Journal of Smart Agriculture Systems, vol. 8, no. 2, pp. 89-99, 2020.
- [3] S. Li, J. Wang, and M. Chen, "Deep Learning Techniques for Plant Disease Diagnosis: A Review," Computers and Electronics in Agriculture, vol. 165, pp. 104-115, 2019.
- [4] A. Kumar and V. Singh, "Machine Learning-Based Approaches for Plant Disease Identification," Agricultural Informatics, vol. 14, no. 1, pp. 34-47, 2021.
- [5] H. Patel and R. Shah, "Application of AI in Smart Farming: Challenges and Opportunities," International Journal of Computational Agriculture, vol. 7, no. 4, pp. 200-210, 2022.

### 2. Books

- 1. Stark, Adam. Python Programming for Beginners. Packt Publishing, 2016.
- 2. Barry, Paul. Head First Python, 2nd Edition. O'Reilly Media, 2016.
- 3. Vincent, William S. Django for Beginners: Build Websites with Python and Django. Independently published, 2018.

### 1. Sites

  <https://www.ieee.org/>

  <https://www.ijert.org/>

3. <https://www.geeksforgeeks.org/disease-prediction-using-machine-learning>