DIVISION OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

Academic Year 2024-2025
EVEN SEMESTER

MINIPROJECT
(MP2911)

Submitted by
J JENOLIN JEBA
(URK22CS5026)

in partial fulfillment for the award of the degree of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING
(Artificial Intelligence)

for the project titled

Plant Disease Diagnosis and Chatbot Assistant using MobileNetV2 and NLP

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

APR 2025

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved

NAAC A++ Accredited

DIVISION OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "**Plant Disease Diagnosis and Chatbot Assistant using MobileNetV2 and NLP**" is a bonafide work done during the Even semester of the academic year 2024-2025 by

J JENOLIN JEBA (Reg.No: URK22CS5026)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (AI/AIML) of Karunya Institute of Technology and Sciences.

Signature of Head of the Division

Dr. D. SUJITHA JULIET
Associate Professor & Head
Division of Artificial Intelligence and Machine Learning

Signature of the Supervisor

Dr Christina Magneta
Assistant Professor
Division of Artificial Intelligence and Machine Learning

Submitted for the Viva Voce held on _____

Examiner

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

School of Computer Science and Technology

DIVISION OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

MP2911-MINIPROJECT

CHECKLIST

MiniProject Review

S.No	Review Checklist	Yes/No/NA	Remarks by Project Guide
1	100% Project Completion- Ready for Demo		
2	PPT-Verified		
3	Project Outcome Proof Verified (Scopus Conference Registration/ Journal Acceptance /Winning the national or international level project or product competition/Patent filed)		
4	Project Review Report verified		

Verified the above checklist

Signature of the Project Guide

*S.No-3 to be attached in the last page of the project report,

DECLARATION

We, the undersigned stakeholders, hereby affirm our commitment to the development and implementation of the project described in this report. We are responsible for the creation and development of the project outlined in this Project report, hereby declare the following:

1) I hereby declare that this project is my original work and has not been copied or submitted elsewhere for any academic or non-academic purposes.
2) I confirm that this project has been carried out in accordance with the guidelines and regulations set by KITS and under the supervision of our assigned faculty.
3) I declare that all research, data collection, and implementation in this project have been conducted ethically, without any form of plagiarism or misconduct.
4) I acknowledge the valuable guidance and support received from our faculty, peers, and other contributors in the successful completion of this project.
5) I take full responsibility for the content, findings, and conclusions presented in this project and understand that any discrepancies or violations may lead to necessary actions as per college policies.

Signature of student

Name: J JENOLIN JEBA
Reg. No.: URK22CS5026
Division of Artificial Intelligence and Machine Learning

Signature of the Supervisor

Dr Christina Magneta
Assistant Professor
Division of Artificial Intelligence and Machine Learning

TABLE OF CONTENTS

Signature of the Supervisor

Dr Christina Magneta
Assistant Professor
Division of Artificial Intelligence and Machine Learning

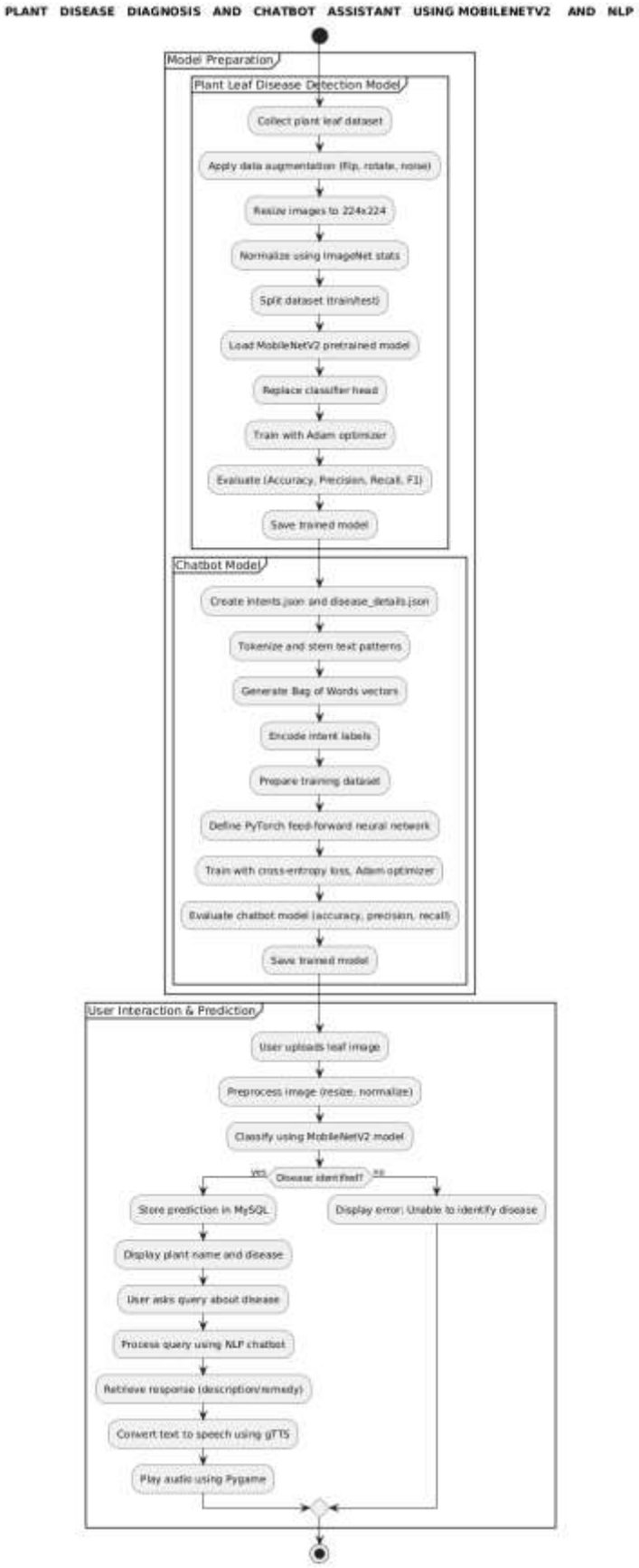
INTRODUCTION

Agriculture forms the backbone of a nation's economy, with approximately 70-75% of India's population relying on it [1]. Traditional disease identification methods, primarily relying on naked-eye observations are time-consuming, expensive, and require significant effort to identify infected leaves [2]. Many farmers lack formal education, which hampers their ability to identify plant leaf diseases promptly and effectively, exacerbating agricultural losses [3, 4]. Accurate identification of these diseases is essential to prevent financial losses and conserve resources [5]. With the advancement of AI technology, agricultural detection based on AI is now widely utilized for tasks including predicting crop production, processing weed identification, and finding plant diseases [6].

Identifying diseases from images of plant leaves is one of the most important research areas in precision agriculture [7]. This work focuses on Deep learning techniques, particularly convolutional neural networks (CNNs) such as pre-trained models like ResNet50, MobileNetV2, EfficientNet are implemented to find the model that can accurately identify the disease in the plant while providing remedies to prevent/cure the disease. It also integrates a Chatbot Assistant that can provide more information on the plant disease on how is it caused, it’s cure if available and how it can be prevented in the future along with a audio that reads the content delivered by the chatbot.

A web-based platform has been created for an AI-powered Plant Disease Detection and Assistance System. Plant diseases significantly impact agricultural productivity, making prompt and accessible identification and expert guidance essential. This project features an integrated system that utilizes image classification, Natural Language Processing (NLP), and text-to-speech (TTS) technologies. The system employs MobileNetV2 for classifying leaf images, a neural network chatbot for interactive assistance, and gTTS for generating audio responses. A distinctive feature of this project is the inclusion of real-time audio feedback, which merges deep learning, NLP, and TTS within a single Flask-based interface. The system boasts an impressive classification accuracy of 99.28%, providing users expert-level assistance.

ANALYSIS AND DESIGN



The Plant Disease Diagnosis and Chatbot Assistant using MobileNetV2 and NLP is a comprehensive system designed to assist users in identifying plant leaf diseases and provide relevant information and remedies through chatbot interaction. The architecture is divided into three main phases: Model Preparation, User Interaction & Prediction, and Response Delivery.

In the Model Preparation phase, two core models are developed. The first is the Plant Leaf Disease Detection Model, which involves collecting a dataset of diseased plant leaves, applying data augmentation techniques (such as flipping, rotating, and adding noise), resizing the images to 224x224 pixels, and normalizing them using ImageNet statistics. The dataset is split into training and testing sets. A pretrained MobileNetV2 model is loaded, and its classifier head is replaced to suit the number of disease classes. This model is trained using the Adam optimizer and evaluated using metrics like accuracy, precision, recall, and F1-score. The final trained model is saved for future inference.

Parallelly, the Chatbot Model is developed to enable natural language interaction using Natural Language Processing (NLP) techniques. The dataset consists of two JSON files: intents.json for user input patterns and intents, and disease_details.json for disease-specific information. NLP is applied to preprocess the text data by tokenizing user queries into individual words and applying stemming to reduce words to their root forms. These processed texts are then converted into numerical format using the Bag of Words (BoW) model, which transforms the textual data into fixed-size vectors suitable for machine learning. Intent labels corresponding to each input pattern are encoded, and a training dataset is prepared. A feed-forward neural network is defined using PyTorch and trained on this NLP-processed data using cross-entropy loss and the Adam optimizer. The trained model is evaluated based on accuracy, precision, and recall, and saved for deployment upon achieving satisfactory performance.

In the User Interaction & Prediction phase, users interact with the system by uploading images of plant leaves. These images are preprocessed and passed to the trained MobileNetV2 model for classification. If a disease is identified, the prediction is stored in a MySQL database, and the plant and disease names are displayed to the user. The user can then ask questions about the disease, which are processed by the NLP-based chatbot model. The chatbot retrieves an appropriate response, which may include a disease description and suggested remedies.

To enhance user experience, the system includes a Text-to-Speech (TTS) feature. The chatbot’s textual response is converted into speech using the gTTS library and played through the Pygame module, enabling audio feedback. In cases where no disease is detected, the system gracefully handles the situation by displaying an appropriate error message indicating that the disease could not be identified.

This architecture leverages the power of deep learning for image classification, natural language processing for conversational interaction, and text-to-speech for accessibility, making it a robust and user-friendly assistant for plant disease diagnosis.

IMPLEMENTATION

PLANT DISEASE DIAGNOSIS USING MOBILENETV2

Dataset

61486 plant leaf photos afflicted by various illnesses are included in the dataset produced by J. Arun Pandian et al. [8]. Image flipping, gamma correction, noise injection, principle component analysis (PCA) color augmentation, rotation, and scaling were the six categories of data augmentation techniques that were employed.

Data Preprocessing

Each image was downsized to 224 × 224 pixels in order to comply with the input dimensions specified by the pretrained model. Since the model was pretrained on this dataset, pixel intensity normalization was carried out using the provided means and standard deviations that are consistent with ImageNet. While creating the dataset, various data augmentation techniques, including rotation, zoom, and horizontal flip, were used to reduce the likelihood of overfitting and enhance the capacity for generalization. A random division of the pre-processed dataset was then made, with 80% set aside for training and 20% for testing.

Classification

MobileNetV2 served as the foundational architecture for categorization, offering the advantages of deep learning at a cheap computing cost. To achieve transfer learning benefits, the model was initialized using pretrained ImageNet weights. To maintain the total number of plant disease classes in the dataset, the final fully connected layer was swapped out for the classifier head of the network. The leaf disease dataset was then used to refine the model. The training approach employed the Adam optimization algorithm, a learning rate of 0.001, and Cross Entropy Loss as the loss function. 32-image batches were used for training throughout ten epochs. To expedite training, GPU acceleration was used wherever feasible.

Evaluation

The suggested model's parameters, including F-measure [9], Precession, and Recall, are computed and provided in Equations. 1, 2, and 3.

$$\text{Precision Measure (\%)} = \frac{\text{True Positives}}{\text{True Positives + False Positives}} \text{ X } 100 \quad \text{- Equation 1}$$

$$\text{Recall Measure (\%)} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}} \text{ X } 100 \quad \text{- Equation 2}$$

$$\text{F} - \text{measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision + Recall}} \text{ X } 100 \quad \text{- Equation 3}$$

CHATBOT ASSISTANT USING NLP

Dataset Collection

Two large datasets were used to train the chatbot. The intents.json dataset holds pre-defined user intent like greeting, farewell, and general queries. For every intent, it holds user input examples (patterns), potential chatbot responses, and other metadata such as confidence levels. The disease_details.json file holds structured data of different plant diseases. Every entry holds a description of the disease in detail and a set of suggested treatments. While processing data, intent patterns were automatically created from these entries.

Data Preprocessing

The input text patterns were cleaned and normalized through the following steps. Tokenization was done with nltk.word_tokenize() to divide each sentence into words. Stemming was performed to reduce the vocabulary size by reducing words to their base form through the Porter Stemmer (stemmer.stem()). Filtering was performed to remove common punctuation characters, providing a clean input to train the model. These processes standardized the input and removed query variations by users.

Feature Extraction

The text data was represented as numerical inputs using the Bag-of-Words (BoW) paradigm. Each phrase in this model is represented by a binary vector that indicates whether the recognized stemmed words are present (1) or absent (0). Based on the combined vocabulary of all the stemmed terms from both collections, each user query is represented as a vector.

Label Encoding

Python indexing was employed to assign each unique purpose tag (e.g., "General_Hello" and "explain_Apple plant with Scab") to its unique numerical label. Since each category is a unique intent, the model handled the problem as though it were a multi-class classification problem.

Dataset Preparation

The obtained X_train and y_train data were then encapsulated within a custom ChatDataset class following the input data transformation into BoW vectors and label encoding. This class was utilized during training alongside PyTorch's DataLoader to provide batches of data in an efficient manner.

Model Architecture

A straightforward but efficient feedforward neural network was implemented using PyTorch. An input layer with a dimension (BoW vector size) equal to the number of features, two hidden layers with ReLU activations to add nonlinearity, and an output layer that provides

raw scores (logits) for each intent with a dimension of the number of intent classes make up the network structure. The cross-entropy loss function, which is particularly well-suited for multiclass classification, was employed in the model's training process. The Adam optimizer was employed, which is renowned for its ability to modify learning rates throughout training. To promote balanced learning, 350 epochs of training were conducted using the training data, with a batch size of 16 samples per batch. Every ten epochs, the loss function was supplied in order to track the model's training.

Evaluation

The chatbot is evaluated with its Accuracy, precision, recall, F1-score and Support.[26] These parameters for the proposed model are calculated and is given in Equations 1,2,3 and 4.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{False Negatives} + \text{True Negatives} + \text{False Negatives}} \text{ X } 100 \quad \text{- Equation 4}$$

Audio response with gTTS

The chatbot's generated text responses are converted to speech using Google Text-to-Speech (gTTS). The audio was played back to the user using the Pygame module, enabling a voice-based interaction interface.

TEST RESULTS

The plant leaf disease dataset was used to train and evaluate the suggested model. With 49189 and 12297 photos, the dataset was divided into training and testing. 39 kinds of healthy and sick plant leaves are included in the dataset. CNN, Resnet, EfficientNet, and Vision Transformers (ViT) were compared to the suggested model. Furthermore, the models' performance is assessed on the ensuing test datasets. Finally, the results show that the proposed model performs better than all of the models that were previously described. The average correctness of the test findings will be assessed in the subsection that follows. The testing accuracy found while comparing the various models is shown below:

MODEL	ACCURACY	MODEL SIZE
Resnet 18	93.18	205.99 MB
Resnet 50	97.55	94.67 MB
MobileNet	99.28	9.34 MB
EfficientNet	99.06	16.54 MB
Vision Transformers (ViT)	50.02	343.38 MB
CNN	93.18	205.99 MB

The graph in Fig 2 displays a model's training loss over several epochs, demonstrating a constant and smooth reduction from an initial value of around 0.200 to a final value near 0.025. This consistent reduction implies that the model is efficiently learning from the training data, as there are no symptoms of instability, such as unexpected spikes or plateau. The absence of unpredictable behavior indicates that the learning rate and optimization method are fine-tuned, allowing the model to converge quickly.

Figure 2

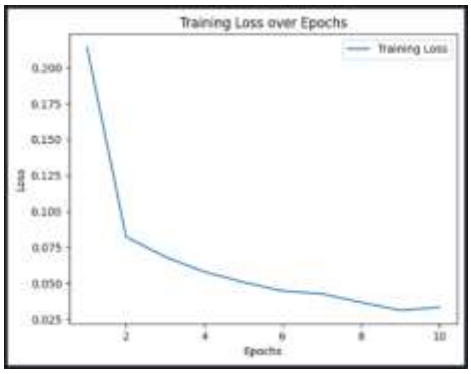
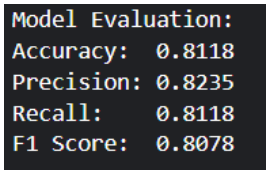


Figure 3 shows that many classes, particularly those with adequate test samples, have excellent precision, recall, and F1-scores, demonstrating that the model is effectively learning and differentiating between several categories. The presence of numerous classes with perfect or near-perfect F1 ratings indicates significant predictive power in those areas.

Figure 3



Performance of the Audio Response

The enforced voice response system significantly enhances the availability and usability of the chatbot by furnishing real- time audible feedback, exercising the gTTS(Google Text- to- Speech) machine for text- to- speech conversion and the Pygame library for audio playback, the system delivers immediate and comprehensible verbal responses(e.g.," The splint belongs to Tomato Early Blight") and conversational relations. Also, the system ensures contextual delicacy by reacquiring the last entry from the MySQL database when responding to follow-up queries, thereby maintaining applicability in individual feedback.

CONCLUSION AND FUTURE SCOPE

The proposed system effectively integrates image-based plant disease detection with an intelligent chatbot assistant to offer a comprehensive diagnostic solution. By leveraging the power of the MobileNetV2 model, the system achieves an impressive accuracy of 99.28% in identifying plant diseases from leaf images. This high-performance image classification is complemented by a responsive chatbot built using Natural Language Processing (NLP) techniques, which allows users to interact with the system and receive meaningful insights. The chatbot retrieves structured information such as disease descriptions, causes, and remedies, thereby enhancing user engagement and understanding. Furthermore, the integration of Text-to-Speech (TTS) using gTTS provides a voice-based interface, making the system more accessible and user-friendly.

As for the future scope, several enhancements can be considered to further improve the system's capabilities. Incorporating transformer-based NLP models like BERT can significantly enhance the chatbot's understanding and response accuracy. Expanding the plant and disease database will allow the system to support a wider variety of crops. Additionally, enabling real-time mobile or drone-based disease detection in the field, along with multilingual voice support, can make this tool highly valuable for farmers in diverse regions. Integration with IoT sensors and weather data APIs could also allow predictive diagnostics and timely alerts, paving the way for a smarter and more proactive agricultural support system.

REFERENCES

[1] Chen J, Chen J, Zhang D, Sun Y, Nanehkaran YA. Using deep transfer learning for image-based plant disease identification. Comput Electron Agric 2020; 173: 105393

[2] Angel Sheril J, Mary Eugine J, Dikshna U. Deep learning based disease detection in tomatoes. 3rd International Conference on Signal Processing and Communication (ICPSC). 13-14 May 2021; Coimbatore, India. 2021.

[3] Nanehkar Y A, Zhang Defu, Chen Junde, Yuan Tian. Recognition of plant leaf diseases based on computer vision. In . 1-5. J Amb Intell Humanized Comput 2020; 2020: 1-5.

[4] Ashok S, Kishore G, Rajesh V. Tomato leaf disease detection using deep learning techniques. Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES). 10-12 June 2020; Coimbatore, India. 2020.

[5] Hukkeri G, Soundarya B, Gururaj H, Ravi V. Classification of Various Plant Leaf Disease Using Pretrained Convolutional Neural Network On Imagenet. Open Agric J, 2024; 18: e18743315305194.

[6] Albattah, W., Nawaz, M., Javed, A., Masood, M., and Albahli, S. (2022). A novel deep learning method for detection and classification of plant diseases. Complex Intelligent Syst., 1–18. doi: 10.1007/s40747-021-00536-1

[7] Khirade SD, Patil AB. Plant disease detection using image processing. In: 2015 International Conference on Computing Communication Control and Automation.; 2015. p. 768–71. doi:10.1109/ICCUBEA.2015.153

[8] J, ARUN PANDIAN; GOPAL, GEETHARAMANI (2019), “Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network”, Mendeley Data, V1, doi: 10.17632/tywbtsjrjv.1

[9]K.V Gokula, J. D, R. Pinagadi Venkateswara, V. D, S. K, An automated segmentation and classification model for banana leaf disease detection, J. Appl. Biol. Biotechnol. 2022 (2022) 1–12.

APPENDIX
APPENDIX A- SAMPLE CODE

```
#Training the Plant disease diagnosis model
def train_model(model, train_loader, criterion, optimizer, epochs=10):
    model.train()
    train_loss = []
    for epoch in range(epochs):
        running_loss = 0.0
        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        epoch_loss = running_loss / len(train_loader)
        train_loss.append(epoch_loss)
        print(f"Epoch {epoch + 1}/{epochs}, Loss: {epoch_loss:.4f}")
    return train_loss
train_loss = train_model(model, train_loader, criterion, optimizer, epochs)

#Training the chat bot model
def train_model(X_train, y_train, input_size, hidden_size, output_size, num_epochs=350,
batch_size=16, learning_rate=0.001):
    dataset = ChatDataset(X_train, y_train)
    train_loader = DataLoader(dataset=dataset, batch_size=batch_size, shuffle=True)
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = NeuralNet(input_size, hidden_size, output_size).to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
    for epoch in range(num_epochs):
        for words, labels in train_loader:
            words, labels = words.to(device), labels.to(dtype=torch.long).to(device)
            outputs = model(words)
            loss = criterion(outputs, labels)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
        if (epoch+1) % 10 == 0:
            print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
    return model, criterion, optimizer
model, criterion, optimizer = train_model(X_train, y_train, input_size, hidden_size, output_size)
```

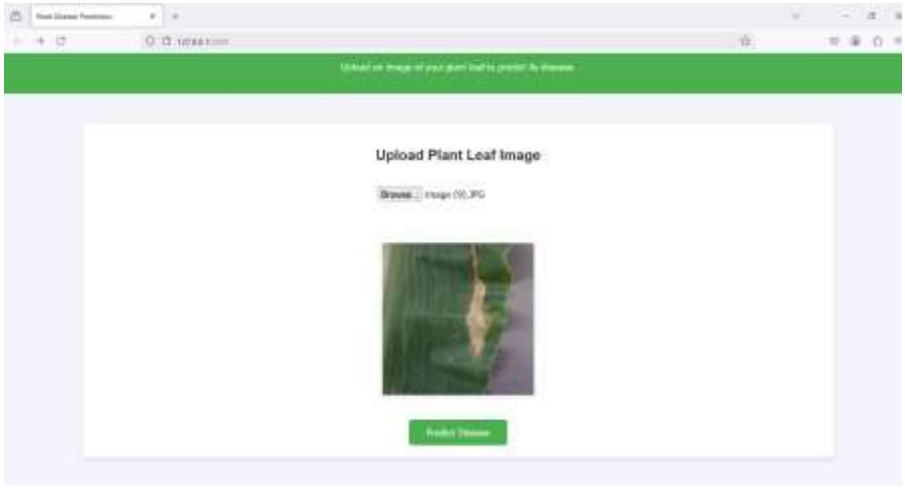
```
#SQL Connection,insertion and fetching
def create_connection():
    try:
        connection = mysql.connector.connect(
            host='localhost',
            database='plant_disease_db',
            user='root',
            password='root'
        )
        if connection.is_connected():
            return connection
    except Error as e:
        print(f"Error: {e}")
        return None

def insert_prediction(image_filename, predicted_class):
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            query = "INSERT INTO predictions (image_filename, predicted_class) VALUES (%s, %s)"
            cursor.execute(query, (image_filename, predicted_class))
            connection.commit()
            cursor.close()
            connection.close()
        except Error as e:
            print(f"Error: {e}")

def get_last_prediction():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            query = "SELECT predicted_class FROM predictions ORDER BY timestamp DESC LIMIT 1"
            cursor.execute(query)
            result = cursor.fetchone()
            cursor.close()
            connection.close()
            return result[0] if result else None
        except Error as e:
            print(f"Error: {e}")
            return None
```

```
# Load MobileNetV2 Model
model = models.mobilenet_v2(weights=models.MobileNet_V2_Weights.DEFAULT)
num_features = model.classifier[1].in_features
model.classifier[1] = nn.Linear(num_features, len(class_name_mapping)) # Adjust output to class count
model_path = "plant_village_mobilenet.pth"
model.load_state_dict(torch.load(model_path, map_location=device))
model = model.to(device)
model.eval()

# From text to speech
def text_to_speech(text):
    timestamp = int(time.time() * 1000)
    filename = f"audio_{timestamp}.mp3"
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    tts = gTTS(text=text, lang='en')
    tts.save(filepath)
    pygame.mixer.music.load(filepath)
    pygame.mixer.music.play()
    return filename

#Predicting the disease
def predict_image(image_path, model, transform, device, class_mapping):
    image = Image.open(image_path).convert("RGB")
    input_tensor = transform(image).unsqueeze(0).to(device)
    with torch.no_grad():
        outputs = model(input_tensor)
        _, predicted = torch.max(outputs, 1)
        predicted_class_idx = predicted.item()
        predicted_class = list(class_mapping.keys())[predicted_class_idx]
        label = class_mapping.get(predicted_class, "Unknown Class")
    audio_filename = text_to_speech(f"The leaf belongs to a {label}.")
    return label, audio_filename

#Chatbot
def load_chatbot_model(filename='data.pth'):
    data = torch.load(filename)
    model = NeuralNet(data["input_size"], data["hidden_size"], data["output_size"]).to(device)
    model.load_state_dict(data["model_state"])
    model.eval()
    return model, data['all_words'], data['tags']

def chatbot_response(model, sentence, all_words, tags, intents, device):
    sentence = tokenize(sentence)
    X = bag_of_words(sentence, all_words)
```

```
    X = X.reshape(1, -1)
    X = torch.from_numpy(X).to(device)

    output = model(X)
    _, predicted = torch.max(output, dim=1)
    tag = tags[predicted.item()]
    probs = torch.softmax(output, dim=1)
    prob = probs[0][predicted.item()]

    if prob.item() > 0.5:
        for intent in intents['intents']:
            if tag == intent["tag"]:
                if tag == "explain_disease":
                    last_disease = get_last_prediction()
                    if last_disease:
                        with open('disease_details.json', 'r') as f:
                            disease_details = json.load(f)

                        if last_disease in disease_details:
                            details = disease_details[last_disease]["details"]
                            remedies = "\n".join(disease_details[last_disease]["remedies"])
                            response = f"Details about {last_disease}:\n{details}\n\nRemedies:\n{remedies}"
                        else:
                            response = f"No details found for {last_disease}."
                    else:
                        response = "No recent predictions found."
                    return response, text_to_speech(response)
                else:
                    response = random.choice(intent['responses'])
                    return response, text_to_speech(response)

    return "I'm sorry, I couldn't understand that.", text_to_speech("I'm sorry, I couldn't understand that.")
```

APPENDIX B – OUTPUT SCREENSHOTS

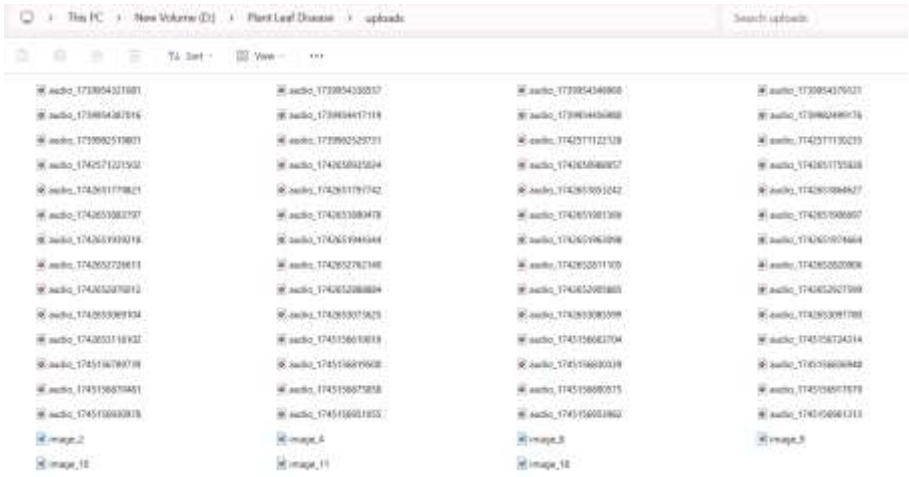
Uploading an image



The disease prediction along with chatbot

Chatbot queries

Saved Voice Responses from text to speech with input images



Saved Prediction from MySQL

```
mysql> use plant_disease_db;
Database changed
mysql> select * from predictions;
+----+----------------+-----------------------------------+---------------------+
| id | image_filename | predicted_class                   | timestamp           |
+----+----------------+-----------------------------------+---------------------+
|  1 | image_1.JPG    | Corn plant with Northern Leaf Blight | 2025-04-20 19:27:21 |
|  2 | image_9.JPG    | Tomato plant with Late blight     | 2025-04-20 19:27:39 |
|  3 | image_10.JPG   | Peach plant with Bacterial spot   | 2025-04-20 19:27:57 |
|  4 | image_5.JPG    | Grape plant with Black rot        | 2025-04-20 19:28:16 |
|  5 | image_8.JPG    | Apple plant with Black Rot        | 2025-04-20 19:28:59 |
|  6 | image_9.jpg    | Corn plant which is  healthy      | 2025-04-20 19:29:18 |
+----+----------------+-----------------------------------+---------------------+
6 rows in set (0.00 sec)
```

PROJECT OUTCOME- SCOPUS PUBLICATION/PATENT PROOF