# Web/Mobile based advisory for identification and management of insect pest in agricultural crops

A Project Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR.

In Partial Fulfillment of the Requirements for the Award of the

degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
BY


| | |
|---|---|
| **D. Mahesh** | **15121A0535** |
| **B. Aishwarya** | **15121A0516** |
| **G. Vivek** | **15121A0565** |
| **G. Sai Vamsi** | **15121A0550** |


Under the Guidance of


**Dr. V. Anantha Natarajan, M.Tech., Ph.D**
**Associate Professor**
Department of Computer Science and Engineering




Department of Computer Science and Engineering
**SREE VIDYANIKETHAN ENGINEERING COLLEGE**
(Affiliated to JNTUA, Anantapuramu)
Sree Sainath Nagar, Tirupathi – 517 102


2015 - 2019

# SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Affiliated to Jawaharlal Nehru Technological University Anantapur)
Sree Sainath Nagar, A. Rangampet, Tirupati – 517 102, Chittoor Dist., A.P.

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## CERTIFICATE

This is to certify that the Project Work entitled

**Web/Mobile based advisory for identification and management of insect pest in agricultural crops**

is the bonafide work done by

| | |
|---|---|
| **D. Mahesh** | **15121A0535** |
| **B. Aishwarya** | **15121A0516** |
| **G. Vivek** | **15121A0565** |
| **G. Sai Vamsi** | **15121A0550** |

In the Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, A.Rangampet is affiliated to JNTUA, Anantapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during 2015 - 2019.

This is work has been carried out under my guidance and supervision.

The results embodied in this Project report have not been submitted in any University or Organization for the award of any degree or diploma.

**Internal Guide**

**Dr. V. Anantha Natarajan**
**M.Tech., Ph.D**
Dept of CSE
S.V.E.C
A.RANGAMPET

**Head**

**Dr. M. Sunil Kumar**
**Prof & Head**
Dept of CSE
S.V.E.C
A.RANGAMPET

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# DECLARATION

We hereby declare that this project report titled **Web/Mobile based advisory for identification and management of insect pest in agricultural crops** is a genuine project work carried out by us, in **B.Tech *(Computer Science and Engineering)*** degree course of **Jawaharlal Nehru Technological University Anantapur** and has not been submitted to any other course or University for the award of any degree by us.

Signature of the student

1.

2.

3.

4.

# ACKNOWLEDGEMENT

We are extremely thankful to our beloved Chairman and founder **Dr. M. Mohan Babu** who took keen interest to provide us the infrastructural facilities for carrying out the project work.

We take this opportunity to express our sincere thanks to **Prof. T. Gopala Rao**, Special Officer, SVET who encouraged us in every effort.

We are highly indebted to **Dr. P. C. Krishnamachary**, Principal of Sree Vidyanikethan Engineering College for his valuable support and guidance in all academic matters.

We are very much obliged to **Dr. M. Sunil Kumar,** Professor& Head, Department of CSE, for providing us the guidance and encouragement in completion of this project.

We would like to express our indebtedness to the project coordinator, **Mr. K. Siva Krishna Rao**, Assistant Professor(SL), Department of CSE for his valuable guidance during the course of project work.

We would like to express our deep sense of gratitude to **Dr. V. Anantha Natarajan**, Associate Professor, Department of CSE, for the constant support and invaluable guidance provided for the successful completion of the project.

We are also thankful to all the Faculty members and Non-Teaching staff of CSE Department, who have cooperated in carrying out our project. We would like to thank our parents and friends who have extended their help and encouragement either directly or indirectly in completion of our project work.

# ABSTRACT

A large volume of agricultural crops pathological information is available on the internet and various other resources, but it is not accessible to the farmers on demand. Hence to provide instant diagnosis of the infected crops and necessary advisory information to the farmers an application is designed. The app focuses on the detection of pests and diseases in the crop and delivers the right information immediately to the farmers. The app is loaded with crop disease library and deep learning model trained to identify pests/ diseases in the crop. The farmer can check the health of the crop or can detect disease/ pests present in the crop either by placing a query (symptoms) or by placing the image of the infected area of the crop. When a text-based query is given the app searches the disease library and delivers information including the ratio of the chemical substances to be used to control the disease. When the farmers upload an image of the infected crop the app predicts the disease/ pests using the pre-trained deep learning model and delivers the disease control information in the form of text SMS and Voice Message. The app can also provide information about the farming practices like crop rotation, poly culture and Water Harvesting. The app is designed to work in offline and is light weight. The app can provide information to the farmers in various regional languages making it so easy for farmers to interpret and understand the solution.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER-1

# INTRODUCTION

Agriculture is the major source and backbone of the income in India.70 percent of the population in India depends on farming. Over 58 percent of the rural people depend on farming as it is an essential mean of livelihood. Plant disease destroys 10 to 15 % of production in India. Fungus, bacteria are responsible for disease in the plant. Most widely occurring diseases in plants are Leaf blast, Brown spot, Aphids, Leaf scald.

Sometimes farmers are unable to pay attention to the diseases, which lead to the economic loss of the crop. When crops are affected by micro organisms and various pests farming in the country is affected. Generally, farmers observe the crops with the naked eye which is time taking and inaccurate. Each disease has a different remedy to work out. Inexperienced use of pesticides can cause the pathogens to develop long - term resistance, severely reducing their ability to fight back. It is crucial to prevent unnecessary waste of financial and other resources, thus achieving healthier production in this changing environment, appropriate and timely disease identification. The approach to disease detection is manual, meaning that farmers rely primarily on guide books or use their experiences to identify diseases and manage them. Each plant disease has different stages of growth. Whenever the disease occurs on a plant, farmers have to keep eyes on the infection. This approach of disease detection is time-consuming and requires some precaution during the selection of pesticides to prevent the diseases
crops. In particular, smartphones offer very new approaches to help ide ntify diseases due to their computing power, high resolution displays an d extensive builtin accessory sets, such as advanced HD cameras.

Capturing images of the leaves that seem to appear infected leaves and getting them processed by a system could become an attractive solution for farmers. Success of such system depends on how accurately the system carries out machine learning operations. The performance of a plant disease detection system can be evaluated by measuring the accuracy of the machine learning algorithms (classification) employed to detect the diseases in plant leaf images. At present Deep learning models are being used to detect and identify the diseases and pests in the plants. In this application we used the deep learning model to detect the diseases in the crop.

**Convolution Neural Networks:**

Convolution neural network is a class of deep learning methods that has become dominant in different tasks of computer vision and attracts interest across a variety of domains. These neural networks consist of multiple building blocks, such as convolution layers, pooling layers and fully connected layers, and are designed to learn spatial hierarchies of features automatically and adaptively

Convolution Neural Networks are analogous to traditional ANNs in that they consist of neurons which, through learning, optimize themselves. Each neuron will still receive an input and perform an operation (such as a scalar product followed by a non-linear function) on which countless ANNs are based. From the raw image vectors input to the class score final output, a single perceptive score function (the weight) will still be expressed throughout the network. CNNs are used primarily in the field of image pattern recognition. This allows us to encode image-specific features into the architecture, making the Introduction to Convolution Neural Networks 3 network more suitable for image-focused tasks–while further reducing the parameters required to set up the model. One of the greatest limitations of traditional ANN forms is that they tend to struggle with the computational complexity required to calculate image data.

## 1.1 Problem Statement:

Insect Pest take a heavy toll on agricultural crops causing severe loss to the farming community. Crop protection is one of the major components of crop management process. Crops are damaged by attack of disease, insect, nematodes and weeds. Managing them in the field and saving the crop from their attack is a major challenge for the farming community. Our crops are under threat from the day they are seeded til they are harvested causing significant damage to the crop affecting adversely to the farmer's economy. Many factors influence disease development and growth of insect that includes genetics of variety, plant growth stage, weather, soil etc. Most of these information on insects, disease etc has been identified and documented by the Scientists and available in various literatures. This documentation will have a better significance if they are reached out to the farming community whenever they need it. Developing a web or mobile app having a complete knowledge base of insect pest carrying its detail in the background will be of great help. It will help in identifying the insect pest in the farmers field based on the damaged symptom or by the image of the insects. On identifying the insect pest the system may provide the management.

## 1.2 Contribution:

This application accepts two types of inputs text based and image based from the user. Then it displays the disease and its control. The application works offline.

## 1.3 Scope and Objectives:

A large volume of agricultural crops pathological information is available on the internet and various other resources, but it is not accessible to the farmers on demand. Hence to provide instant diagnosis of the infected crops and necessary advisory information to the farmers an application is designed. The app focuses on the

detection of pests and diseases in the crop and delivers the right information immediately to the farmers. It also delivers the information about the ratio of the chemical substance to be used to control the disease. The app is light weight and works offline.

A database of about 600 diseases and their chemical and biological control is available. The application provides solutions to 51 varieties of plants.

The main objectives of the proposed system are as follows:
To detect the diseases at an early stage.

The farmer can identify the disease at an early stage by placing the image of the infected area of the crop. Farmer can also place a query to know about the disease and the respective measures. Therefore, farmers can apply the required fertilizer in the required amount so that the crop can yield good result.

To detect microorganism and the bacterial based diseases.

Plant disease library is loaded with different fungal, bacterial and microorganism-based diseases. The farmer can be able to detect the microorganism or bacteria that has caused the disease. It also suggests the respective measure for the disease treatment.

To detect the deficiency of chemicals in the leaf.

Disease in the leaf can be caused due to the deficiency of chemicals in the soil and other factors. The application will help the farmer to identify the deficient chemical by using image search as well as text-based search and respective measures will be displayed.

To detect and recognize the pests.

Plant disease library is loaded with different diseases that are caused due to the insects and pests. The application will enable farmer

can identify, which type of diseases are affected to leaf by using image search as well as text-based search.

To suggest the farmers with the suitable solutions and suggestions based on the detection and predictions made.

Plant Disease library is loaded with different diseases and their management is available in the form of biological control, chemical control and traditional control format. So that farmer can use the effective methods for disease treatment.

Mobile App Available in 10 Regional Languages and works offline

The application displays the measures and management in 10 Regional Languages (English, Hindi, Telugu, Kannada, Tamil, Malayalam, Bengali, Urdu, Marathi, Nepali). Hence the farmers can easily understand the solutions and can treat the plants.

### 1.4 Applications and Limitations:

The following are the application of the proposed system:

1. The application is used to detect the diseases in the leaf that are caused due to pests, microorganisms, Bacteria and Chemical deficiency.

2. Management/control of the disease is available in the native language.

3. Ratio of pesticide/fungicide to be used is specified.

4. Biological and chemical control methods are predicted.

5. Application available in offline.

### 1.5 Existing System:

- Existing systems uses the standard convolution neural networks, Computer vision techniques and colour co-occurrence method to detect the diseases in leaves.
- Standard convolution neural networks require more computational power and a greater number of parameters.
- Moreover, existing systems require internet availability to diagnose a leaf.

### 1.6 Proposed System:

- Proposed system accepts two types of inputs that is text based and image based from the user.
- It also has a built-in plant disease library having more than 600 diseases and their respective control mechanisms.
- Proposed system does not need internet availability to diagnose a crop. It works offline.

### 1.7 Datasets:

Appropriate data sets are required at all stages of disease recognition, beginning with the training phase to evaluate recognition algorithms performance. All the images collected for the dataset were taken from the dataset of the plant village and other sources, which were searched on different sources by disease and plant name. Images in the dataset were grouped into different classes representing visually determinable plant diseases from leaves. A further class was added in the dataset to distinguish healthy leaves from diseased leaves. It contains only images of healthy leaves. Thus, to identify the crops and diseases in the leaves, deep neural network could be trained. For training, verification and validation, images are divided by using K fold cross validation re sampling procedure. Finally, a database was created with an enormous number of images.

# CHAPTER-2
# LITERATURE SURVEY

A combination of two methods is used to detect the infected part disease. First segmentation is done, where segmentation removes the unwanted material from the image like soil, unaffected part of the leaf and anything else. The segmentation done based on edge detection. Where It takes the image, which is RGB model as input image and then feature values will be calculated based on the input image from RGB model and it will extract the required output. Then image analysis will be performed and after that classifier will be executed on it.

The software prototype system is illustrated and contains a description for disease detection and using image growing, image segmentation techniques on affected part of the leaf. They also studied the methods of image processing established to detect the diseases in leaves. The authors used the cucumber powdery mildew, speckle and downy mildews as samples of study and relate the effect details of simple and medium filter.

The support-vector machines, which is a machine learning algorithm and also in nonlinear in nature and set of features were extracted. Generally, these support-vector machines improve the accuracy of the disease detection. The main purpose of support-vector machine algorithms is used to detect the feature extraction. Suggested a technique for segmentation clustering k-means. RGB was converted to HIS, the colour co-occurrence method was used to extract colour characteristics [1].

In the scenario, the authors focused on having an established approach to automatically grading the defects on the plant leaves. A system based on Machine Vision Technology and Artificial Neural Network is of great use to detect the leaf plant automatically as well as to detect and grade the leaf disease. Automatic image processing and

neural network based approach has been studied and proposed for plant leaf disease deInsect Pest take a heavy toll on agricultural crops causing severe loss to the farming community. Crop protection is one of the major component of crop management process. Crops were damaged by attack of disease, insect, nematodes and weeds. Managing them in the field and saving the crop from their attack is a major challenge for the farming community. Crops are under threat from the day they are seeded till they are harvested causing significant damage to the crop affecting adversely to the farmer's economy. Many factors influence disease development and growth of insect that includes genetics of variety, plant growth stage, weather, soil etc. Most of these information on insects, disease etc has been identified and documented by the Scientists and various other sources. This documentation will have a better significance if they are reached out to the farming community whenever they need it. Developing a web or mobile app having a complete knowledge base of insect pest carrying its detail in the background will be of great help. It will help in identifying the insect pest in the farmers field based on the damaged symptom or by the image of the insects. On identifying the insect pest, the system may provide the management action. The method of color co-occurrence was used to extract color and texture characteristics specific to the type of leaf disease. The extracted set of features was used as input to train a neural network of feedback propagation and subsequent leaf disease detection.

An efficient, simple, cheap, fast and reliable system for the detection and classification of plant diseases can be developed on the basis of the proposed approach. An application of texture analysis in detecting and classifying the plant leaf diseases had been explained. Thus the proposed algorithm was tested on ten species of plants namely banana, lemon, mango, potato, tomato, and paddy. The diseases specific to those plants were taken for approach. The

experimental results indicate the proposed approach can recognize and classify the leaf diseases with a little computational effort. By this method, the plant diseases can be identified at the initial stage itself and the pest control tools can be used to solve pest problems while minimizing risks to people and the environment.

The techniques which were basically used for the detection and classification of leaf disease in plants which are K means clustering for segmentation, artificial neural network, Probabilistic Neural network and GLCM and SGLDM for texture analysis.[21]

Three diseases were detected using Eigen feature regularization and extraction technique (Red Spots, Leaf Crumple and White Spots). They can achieve 90 percent accuracy in detecting Red Spot (Fungal Disease) with the proposed algorithm. Fuzzy selection approach proposed by the authors, i.e. Fuzzy curves (FC) and surfaces (FS), leaves image on cotton disease. Using a fuzzy selection approach, they identify the best set of features. This research takes two steps to follow the process. FC isolates important and required features from the original feature set automatically in the first step and eliminates inaccurate features. Then use FS to get the feature depending on the significant feature in the second step. This approach is especially important in reducing the dimensionality of the feature space that simplifies practical implementation for classification applications.[6]

A groundnut leaf is taken to detect the disease by converting the image of the leaf. The texture and feature analysis of the hue value was analyzed and the groundnut leaf region was detected using neural networks. Paddy crop color prediction was done by feeding Different HSV values in the database (LCC) and then comparing the HSV test and reference image values to obtain the test image correlation value. The masking technique has been used to ignore the background.

On the basis of techniques such as image clipping, image segmentation such as K-Means clustering, Otsu threshold methods, particular diseases were detected. The extraction of the disease feature was used to detect the disease and the neural network was finally used to make the system self-efficient. The creation of an automatic moving robot captures the image of leaves on a farm. For few specific diseases, the captured images were analyzed and the exact amount of pesticide is sprayed on the basis of neural networks. The robot is also made up of a specific movement plan. Citrus leaves disease has been examined using clustering of k-means, extraction of texture features using GLCM, and the outcome is achieved using SVM. This helps the plant to reduce unnecessary pesticides or other chemical substances [7].

The authors presented a reduced approach to fruit identification and classification based on color and texture characteristics. Identifying normal and affected types of fruits by designing an elevator that can move fruits across camera that acquires side views of fruits and determines quality and fruit type. The proposed system performed is relevant to the fruit classification in the real world and involves techniques for image processing and pattern recognition. This method supports the fruit system's non-destructive quality measurement. The characteristics of color are reduced from 18 to 2. For 2 color characteristics, an average accuracy of 89.15 percent for affected type is obtained. The texture features are reduced from 30 to 2. Using 2 texture features, the average accuracy is achieved by 93.019 percent for normal type and 89.50 percent for affected type. To take advantage of both features, the combination of color and texture features is tested. The average accuracies for the normal type were increased to 96.85% and 93.89% for the affected type. In this work, a BPNN classifier is found to be appropriate[9].

The authors discussed a variety of plant diseases including Grape and wheat disease, sugar beet leaf disease, Cotton leaf disease, Rice plant disease, Orchid leaf disease, Rubber tree leaf disease, Apple fruit disease and chili plant disease. The symptoms, classification and detection of all these diseases have been depicted in peculiarity. Image segmentation was mainly performed on the leaves in order to check the disease present in it. The pathogens are basically responsible for the plant disease which destroys the leaf and stem of the plants. Back propagation (BP) networks, radial basis function (RBF) neural networks.[11]

The authors have proposed an approach which involves converting RGB image to HSV format and using H component for masking and removing green pixels at a threshold value. Colour co-occurrence matrix is used for comparing texture parameters to that of a normal leaf. Rupesh G. Mundada have proposed an approach where images are converted from RGB to Greyscale first, followed by resizing and filtering them. Later feature extraction with features like contrast and entropy is performed. Classification is done using a Support Vector Machine.

First conversion of an image from RGB to HSI was done and then green pixels are masked using threshold values. Texture analysis was then done using colour co-occurrence matrix (SGDM). The image was then classified using either minimum distance criterion or SVM classifier which has 86.77% and 94.74% accuracy respectively. Insect Pest take a heavy toll on agricultural crops causing severe loss to the farming community. Crop protection is one of the major components of crop management process. Crops are damaged by attack of disease, insect, nematodes and weeds. Managing them in the field and saving the crop from their attack is a major challenge for the farming community. Crops are under threat from the day they are seeded till they are harvested

causing significant damage to the crop affecting adversely to the farmer's economy. Many factors influence disease development and growth of insect that includes genetics of variety, plant growth stage, weather, soil etc. Most of this information on insects, disease etc has been identified and documented by the Scientists and various other sources. This documentation will have a better significance if they are reached out to the farming community whenever they need it. Developing a web or mobile app having a complete knowledge base of insect pest carrying its detail in the background will be of great help. It will help in identifying the insect pest in the farmers field based on the damaged symptom or by the image of the insects. On identifying the insect pest, the system may provide the management. Hrishikesh P. Kanjalkaret al, in their paper, the image is first converted from RGB to HIS. Segmentation is done using connected components labelling, thresholding is used to avoid unwanted regions. 11 features are used in this approach and classification is done using back propagation neural network. The accuracy of this method is 83%. proposed three methods for extracting features, histogram for colour, erosion concept morphology for obtaining boundaries of the images and colour coherence vector to classify pixels. K-means clustering is used for segmentation and SVM is used for classification of the images the accuracy is 81%. An approach where the processed image is first subjected to k-means clustering and image features are extracted and analysed using grey level co-occurrence matrix.[12]

**2.7** Manual method was used to monitor the farm, the farmer himself checked the parameters such as soil moisture, moisture, leaf disease. The authors surveyed smart farming to increase crop productivity and overall farming. Using IOT and sensors, it is possible to monitor the farm. You can find the farm's condition from your house or anywhere else. The author conducts various surveys on atmospheric change, disease detection and diagnosis, fertilizer calculator, soil estimates and

crop water estimates. It describes the smart farming technique's novel methodology. It uses some mechanical machinery, electronic circuits and sensors. To develop a smart system, the IR sensor, Optocoupler, crane system is used. This system consists of sensors such as soil moisture, temperature, humidity and sensors of obstacles. Using a microcontroller used to monitor the farm, a robot was developed.

This describes the farmers ' mobile application model that provides the farm's status. PIC microcontroller, sensors such as soil, temperature, humidity, and PIR are the hardware used in this model. For wireless communication with the sim card used to communicate with the owner, the GSM module is used. It describes how leaf disease can be detected by using texture statistics to detect plant disease. First, convert captured RGB image to HSV and then apply masking to the image's masking of green pixels. Segmentation is applied after masking and these segments are used for texture analysis and finally, the texture parameter is compared with the leaf's ideal texture parameter.

The authors present the study on techniques of image processing processed on various crop types such as fruit crops, vegetable crops, commercial crops, and cereal crops. The methodology proposed was to identify symptoms of fungal disease on different crop types. The database was maintained to store the texture on crops of fungal disease. The system used to monitor the crop remotely using GSM and to send a notification to the owner in order to control further loss. Arduino was used in this stem as the core component. Wireless network system has been designed using ZigBee. Each mode has connected a group of sensors to design the network of sensors and they are connected to Arduino and ZigBee. Using ZigBee transmitted Intelligence values are measured and the flaws in agriculture are found. This paper presents a technique used by image segmentation to detect leaf

disease; it also classifies the diseases of the plant leaf. This algorithm includes steps such as segmentation of images. Then the diseases are classified using expert techniques. It represents the technique that includes a comparison between an uploaded image of diseased leaf and images from the database. If the same feature image has been found, find the associated image information and detect a plant leaf disease. This paper presents the AgroTick application that includes technologies such as cloud computing, embedded firmware, hardware unit, and analysis of big data. This application has also been used to share knowledge about farming. Recognizing the previous research done in this field is important in order to be able to progress in the right direction. Detection of plant leaf disease has been a major area of research in which both image processing and deep learning techniques have been widely used to classify it accurately. It discusses the most popular techniques incorporated in literature in the relevant field. If done manually, monitoring a large crop field is a tedious task. The human effort put into plant supervision must be minimized. This is therefore a popular research domain that attracts a lot of researchers. In the literature, several works relating to plant diseases are observed.

An effective method has been proposed by the authors to identify whether a tomato leaf has been healthy or infected. The image given as input was first pre-processed by removing the background and with the help of erosion technique the present noise was eliminated. Gray Level Co-occurrence Matrix (GLCM) has been used to extract the enhanced image texture feature. Support Vector Machine (SVM) classifier was trained using various kernel functions and performance was evaluated using the cross-validation technique of N-fold. Using the linear kernel function with the SVM classifier, the proposed system achieved a 99.83 percent accuracy. Although the accuracy obtained is high, predicting or differentiating between healthy or diseased leaves is not sufficient. There was also no identification of the type of disease. The authors

have proposed various segmentation, extraction and classification techniques to overcome the problem of the above paper, which identify and detect the type of disease using the diseased image to conduct classification[14].

# CHAPTER-3

# OVERVIEW

The application works to deliver the right information to the farmer at the right time. The app is loaded with crop disease library and deep learning model trained to identify pests/ diseases in the crop. The farmer can check the health of the crop or can detect disease/ pests present in the crop either by placing a query (symptoms) or by placing the image of the infected area of the crop.

## 3.1 PROPOSED SYSTEM AND ARCHITECTURE:

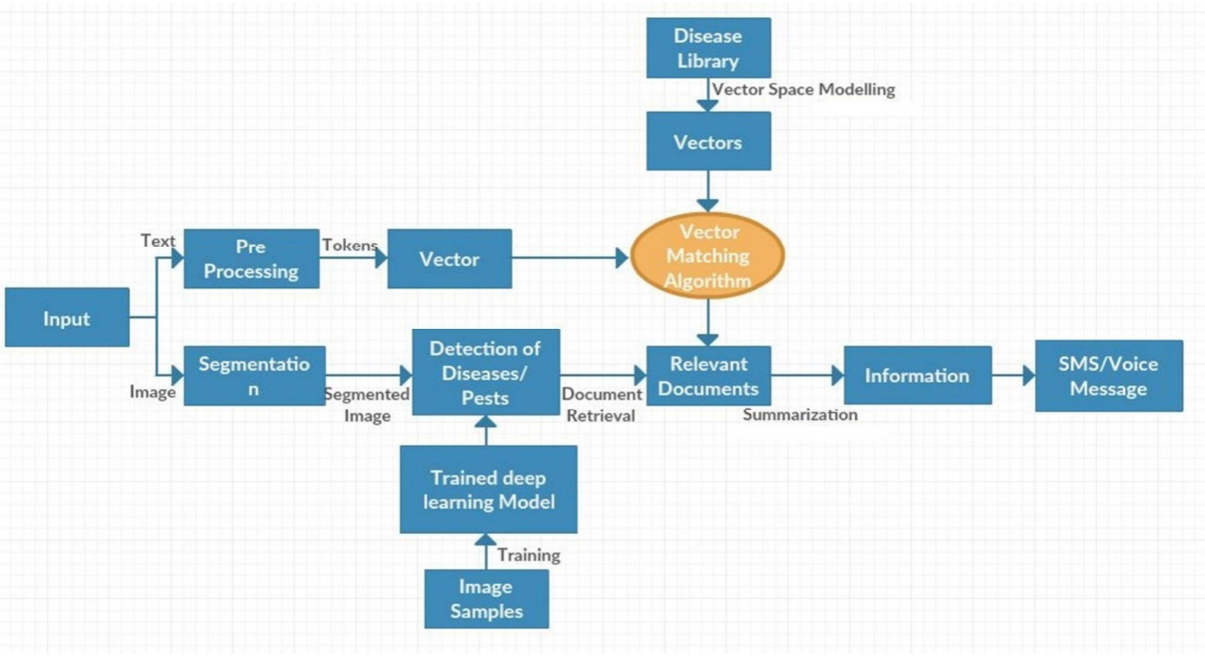


**Fig 3.1. Proposed System and Architecture**

### Text-based Approach:

When a text query is given as input to the app, it searches the disease library and provides necessary information including the ratio of the chemical substances to be used to control the disease.

**Image-based Approach:**

When the farmers upload an image of the infected area of the leaf the app predicts the disease/ pests using the pre-trained deep learning model and delivers the disease control information to the farmers.

### 3.2 Architecture:

**Input:**

Initially, this application accepts two types of inputs from the user **Text-based Approach** and **Image-based Approach**. The user required to choose one among these two approaches.

#### 3.2.1Text-based Approach:

When a text query is given by the user, the application searches the disease library with the given input by the farmer and delivers information including the ratio of the chemical substances (in various methods) to be used to control the disease. The input by the farmer goes through the pre-processing stage, vector matching stage and then to the summarization stage.

**Pre-processing:**

An efficient method has been proposed by the authors to identify whether a tomato leaf was healthy or infected. The image given as input was first pre-processed by removing the background and with the help of the erosion technique the present noise was eliminated. Gray Level Co-occurrence Matrix (GLCM) was used to extract the enhanced image texture feature. Support Vector Machine (SVM) classifier was trained using various kernel functions and performance evaluation was performed using N-fold cross-validation method. Using the linear kernel function with the SVM classifier, the proposed system achieved a precision of 99.83 percent. Although the accuracy obtained is high, predicting or distinguishing between healthy or diseased leaves is not sufficient. There has also been no identification of the type of disease.

The authors have proposed various segmentation, extraction and classification techniques to overcome the problem of the above paper that identify and detect the type of disease using the diseased image to conduct classification[14].

**Vector:**

Machines are better at understanding numbers that actual text passed on as tokens. This process of converting text to numbers is called Vectorization.Vectors then combine to form vector space which is continuous in nature, an algebraic model where rules of vector addition and similarity measures apply. Vectors are a foundational element of linear algebra. Vectors are used throughout the field of machine learning in the description of algorithms and processes such as the target variable (y).Word2vec is a particularly computationally-efficient predictive model for learning word embeddings from raw text. It comes in two flavours, the Continuous Bag-of-Words model and the Skip-Gram model.

**Disease Library:**

Disease library is composed of different diseases and their management is available including biological control, chemical control and traditional control Procedures.

**Vector Matching algorithm:**

Vector matching is needed to match the user input with the particular word from the disease library, so that the respective information can be displayed to the user. Most common vector matching algorithm is Similarity of Cosine. Cosine similarity is a measure of similarity of an inner product space between two non-zero vectors that measure the angle cosine between them. The cosine of 0 ° is 1 and is less than 1 for any angle in the interval (0, π] radians. It is therefore a judgment of orientation and not magnitude: two vectors

with the same orientation have a cosine similarity of 1, two vectors oriented at 90 ° relative to each other have a similarity of 0, and two diametrically opposed vectors have a similarity of-1, regardless of their magnitude. This is analogous to the cosine, which is unity (maximum value) when the segments are perpendicular to a zero angle and zero (uncorrelated).

**Relevant Information:**

By using vector matching algorithms, the relevant information is ranked according to similarity. Then the most similar documents are retrieved and documented.

**Summarization:**

Summarization is the process of shortening a text document with software, in order to create a summary with the major points of the original document. Retrieved relevant documents are then summarized and then presented to the user. The application provides different techniques to eradicate the disease for the farmer to prevent economic loss.

### 3.2.2 Image based search:

When the farmers upload an image of the infected area of the leaf, the application predicts and displays the disease/ pests management using the pre-trained deep learning model and delivers the disease prevention mechanism information to the farmers. The input given by the farmer goes through Segmentation, detection of diseases and relevant information retrieval stages.

**Segmentation:**

Image segmentation is the process of digital image partitioning into multiple segments (pixel sets, also known as super-pixels). The segmentation goal is to simplify and/or make the representation of an image more meaningful and easier to analyze. Typically, image

segmentation is used to locate objects and boundaries in images (lines, curves, etc.). More specifically, the process of assigning a label to each pixel in an image is image segmentation, so that pixels with the same label share certain characteristics.

**Detection of diseases/pest:**

**Image samples:**

These samples can be defined as affected part of image leaf (with different diseases and with different crops).These samples are collected by the application providers.

The image samples are the samples are used to retrieve the relevant information by using deep learning techniques and it displays the information to the farmers.

### 3.3 Deep learning Model:

**MobileNet:**

MobileNets are based on a streamlined architecture that builds light weight deep neural networks using depth-wise separable convolutions. It uses two simple global hyper-parameters that trade latency and accuracy effectively. These hyper-parameters allow the model builder to select the right model based on the constraints of the problem for their application. We present extensive resource and accuracy trade-off experiments and show strong performance on ImageNet classification compared to other popular models. We then demonstrate MobileNet's effectiveness across a wide range of applications and use cases including object detection, classification of fine grains, face attributes, and large-scale geo-location.

# CHAPTER-4

# DESIGN

## 4.1 MobileNet

We use a transfer learning mechanism to train the images. MobileNet is lightweight in architecture. Here we used MobileNet to classify the images of the leaves. MobileNets are the efficient convolution neural networks for mobile vision applications. It uses depthwise separable convolutions which basically means it performs a single convolution on each colour channel rather than combining all three and flattening it. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size and number of parameters.

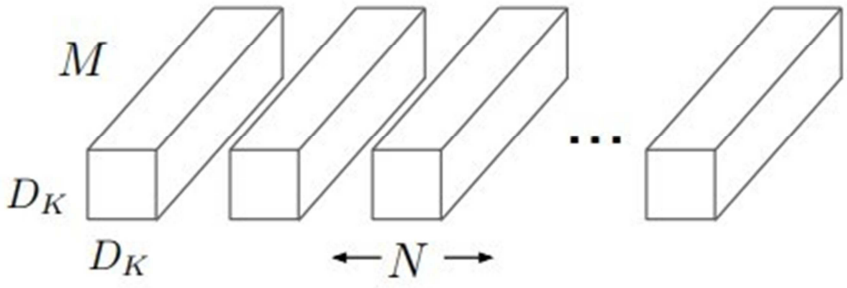


**Fig 4.1. Standard Convolution Filters**

Standard convolutional neural networks filters and combine inputs into a new set of outputs in a single step.

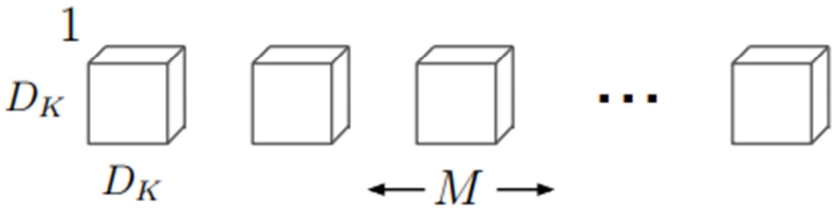


**Fig 4.2.Depthwise Convolutional Filters**

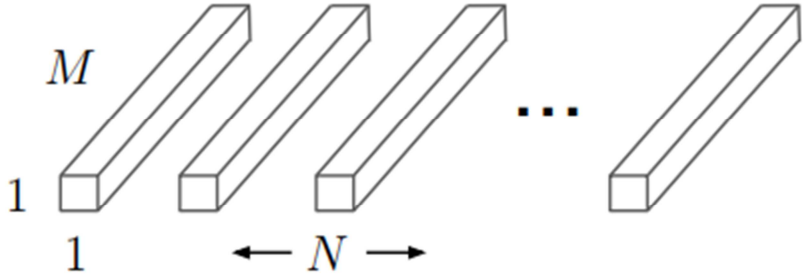


**Fig 4.3. 1x1 Convolution Filters called Pointwise Convolution in the Context of DepthwiseSeperable Convolution**

Depth wise Separable convolutional networks is done in two steps i.e., Depth wise and point wise.

### 4.2 MobileNet body Architecture:

The overall architecture of Mobile Net is as follows, having 30 layers with

1. convolutional layer with stride 2
2. Depth wise layer
3. pointwise layer that doubles the number of channels
4. Depth wise layer with stride 2
5. pointwise layer that doubles the number of channels

| Type / Stride | Filter Shape | Input Size |
|---|---|---|
| Conv / s2 | $3 \times 3 \times 3 \times 32$ | $224 \times 224 \times 3$ |
| Conv dw / s1 | $3 \times 3 \times 32$ dw | $112 \times 112 \times 32$ |
| Conv / s1 | $1 \times 1 \times 32 \times 64$ | $112 \times 112 \times 32$ |
| Conv dw / s2 | $3 \times 3 \times 64$ dw | $112 \times 112 \times 64$ |
| Conv / s1 | $1 \times 1 \times 64 \times 128$ | $56 \times 56 \times 64$ |
| Conv dw / s1 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 128$ | $56 \times 56 \times 128$ |
| Conv dw / s2 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 256$ | $28 \times 28 \times 128$ |
| Conv dw / s1 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 256$ | $28 \times 28 \times 256$ |
| Conv dw / s2 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 512$ | $14 \times 14 \times 256$ |
| $5\times$ Conv dw / s1<br>Conv / s1 | $3 \times 3 \times 512$ dw<br>$1 \times 1 \times 512 \times 512$ | $14 \times 14 \times 512$<br>$14 \times 14 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| Conv / s1 | $1 \times 1 \times 512 \times 1024$ | $7 \times 7 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 1024$ dw | $7 \times 7 \times 1024$ |
| Conv / s1 | $1 \times 1 \times 1024 \times 1024$ | $7 \times 7 \times 1024$ |
| Avg Pool / s1 | Pool $7 \times 7$ | $7 \times 7 \times 1024$ |
| FC / s1 | $1024 \times 1000$ | $1 \times 1 \times 1024$ |
| Softmax / s1 | Classifier | $1 \times 1 \times 1000$ |

**Table 4.1.Mobilenet body architecture**

It is also very low maintenance thus performing quite well with high speed. There are also many flavours of pre-trained models with the size of the network in memory and on disk being proportional to the number of parameters being used. The speed and power consumption of the network is proportional to the number of MACs (Multiply-Accumulates) which is a measure of the number of fused Multiplication and Addition operations.

## 4.3 Schematic view of Flow of Data through Model:

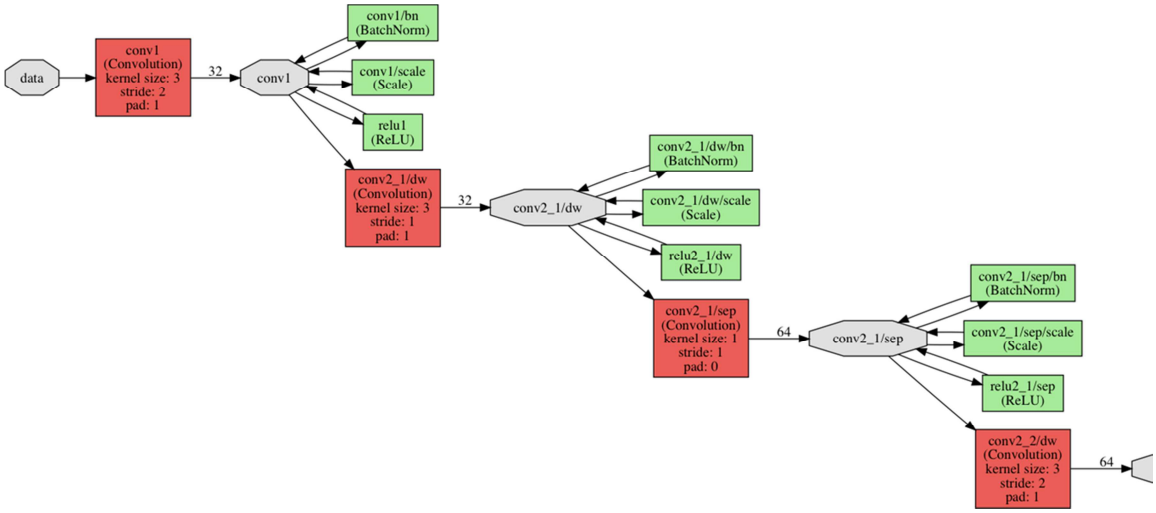


**Fig 4.4. Schematic view of flow of data through model**

## Depthwise separable convolution neural networks:

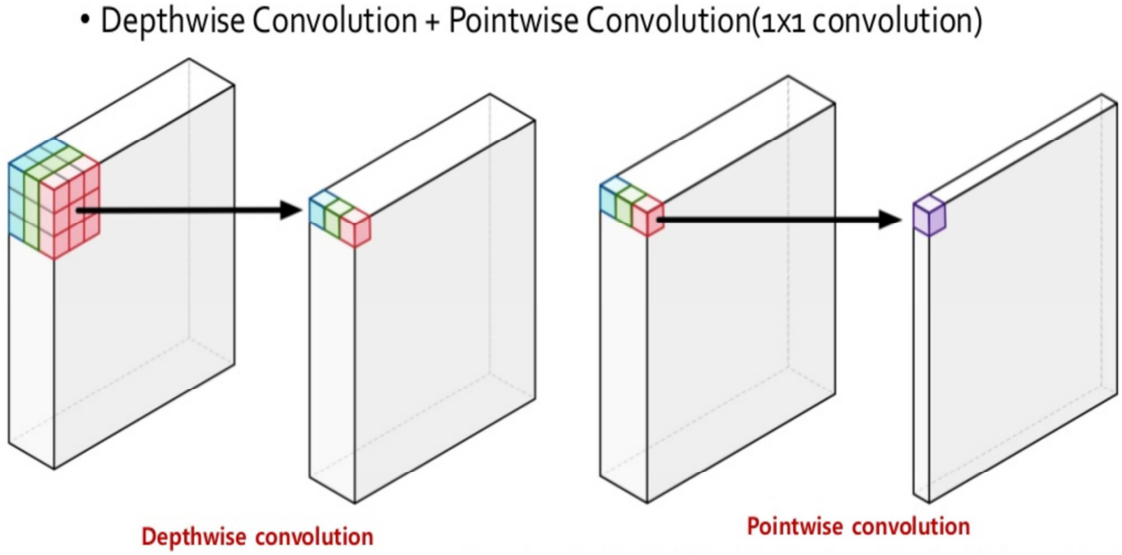


**Fig 4.5.Depthwise and Pointwise Convolution**

**Standard Convolution vs Depthwise Convolution:**

- Standard convolution has the computational cost of $D_k x D_k x M x N x D_F x D_F$
- Depthwise separable convolutions cost $D_k x D_k x M x D_F x D_F + M x N x D_F x D_F$
- Reduction in computations

$$1/N + 1/D_k^2$$

$D_k$ : Width/height of filters.

$D_F$ : Width/height of feature Maps.

M : Number of input channels.

N : Number of Output channels/filters.

If we use 3X3 depthwise separable convolutions, we get between 8 to 9 times less computations.

Number of computations and number of parameters in depthwise convolution networks reduces drastically.

## 4.4 UML Diagrams

UML is a standard language for software system artefacts to be specified, visualized, constructed and documented. The Object Management Group (OMG) created UML and the OMG proposed UML 1.0 specification draft in January 1997. OMG continually strives to create a standard that is truly industry. UML stands for Unified Modelling Language.

- UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints.

- UML can be described as a general purpose visual modelling language to visualize, specify, construct, and document software system.
- Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard.

### 4.4.1 Use case Diagram:

Use Case diagrams show the various activities the users can perform on the system. The System is something that performs a function. They model the dynamic aspects of the system. It provides a user's perspective of the system.

**Actor:** An actor is a user of the system playing a particular role.

**Use case:**Use case is a particular activity a user can do on the system.

**Relationship:**Relationships are simply illustrated with a line connecting actors to use cases.

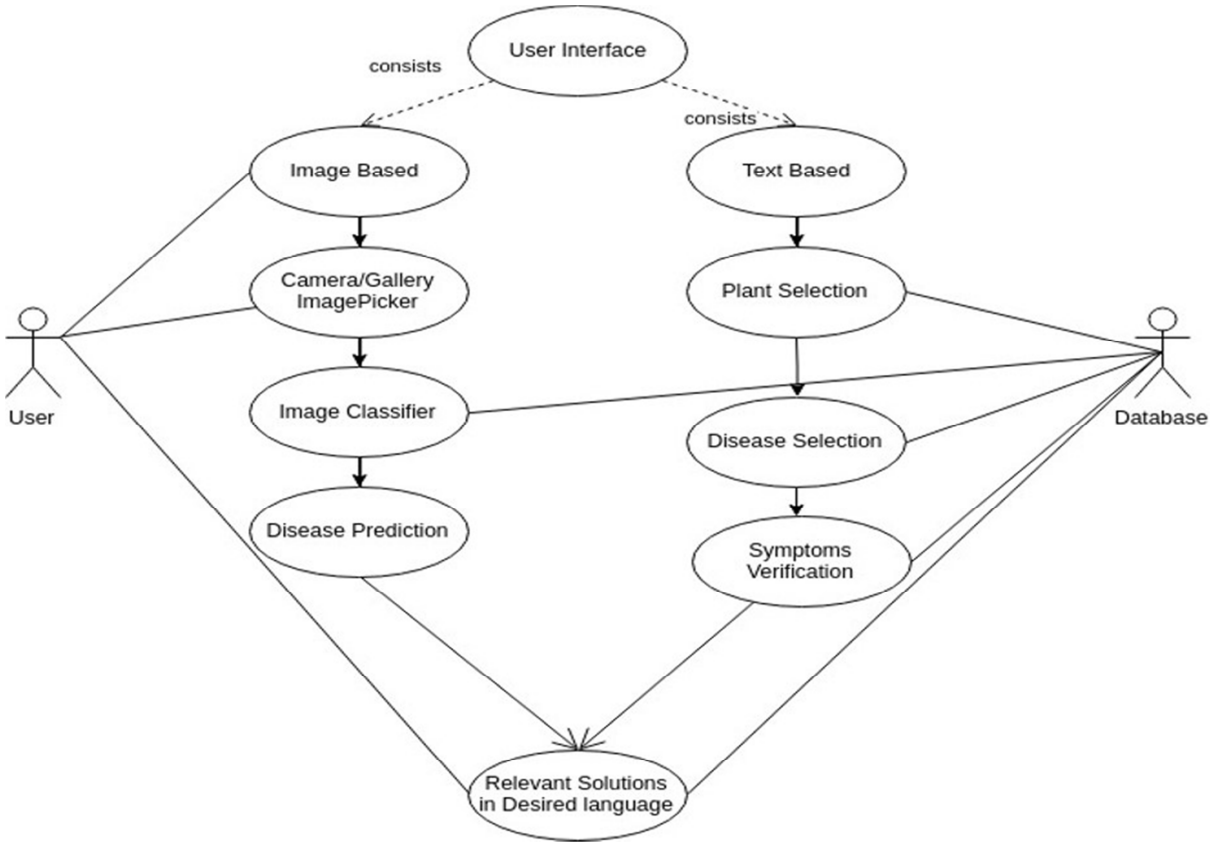


**Fig 4.6. Use case diagram for detection of diseases in plants using both image based and text-based methods.**

In the above diagram Actors are user and database. User can identify the disease in the leaf by placing a image. User can know more about a disease by placing a query.

### 4.4.2 Class Diagram:

A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. i.e., A graphical representation of a static view on declarative static elements. A class is the description of a set of objects having similar attributes, operations, relationships and behaviour.

**Classes:** User, identifying diseases in a leaf, tflite Model, Plant disease Library, Image based, Text based.

**Relationships:** Association.

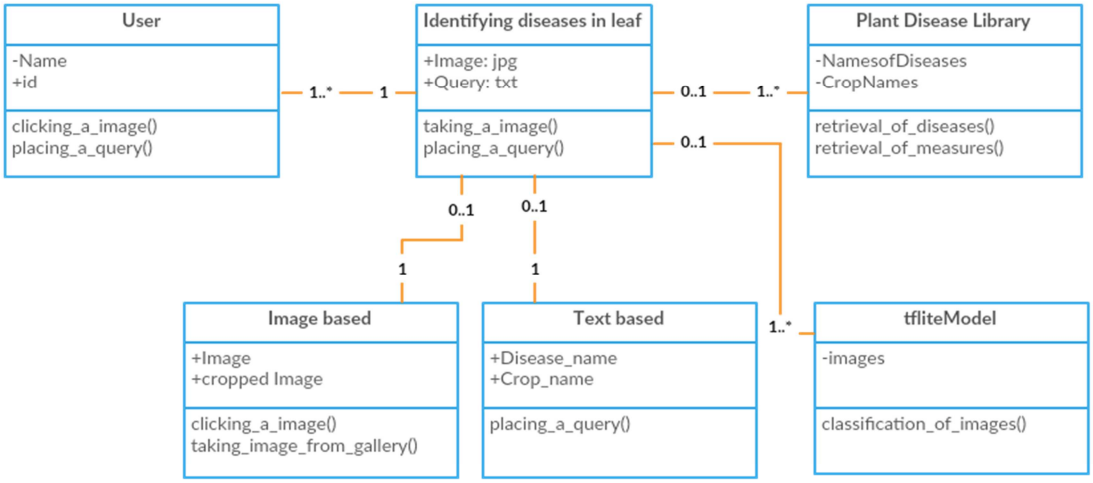


**Fig 4.7. Class Diagram for identifying the diseases in a leaf.**

### 4.4.3 Sequence Diagram:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. Sequence diagrams contain the following elements:

**Class roles:** It represent roles that objects may play within the interaction.

**Lifelines:** It represents the existence of an object over a period of time.

**Activations:** It represent the time during which an object is performing an operation

**Messages:** It represents communication between objects.

In the following figure the user places a query or the image of the infected area of the crop. When a image is placed it searches the model for classification. When a query is placed it searches the plant disease

library and displays the relevant information about the crop. It also displays the similar diseases.

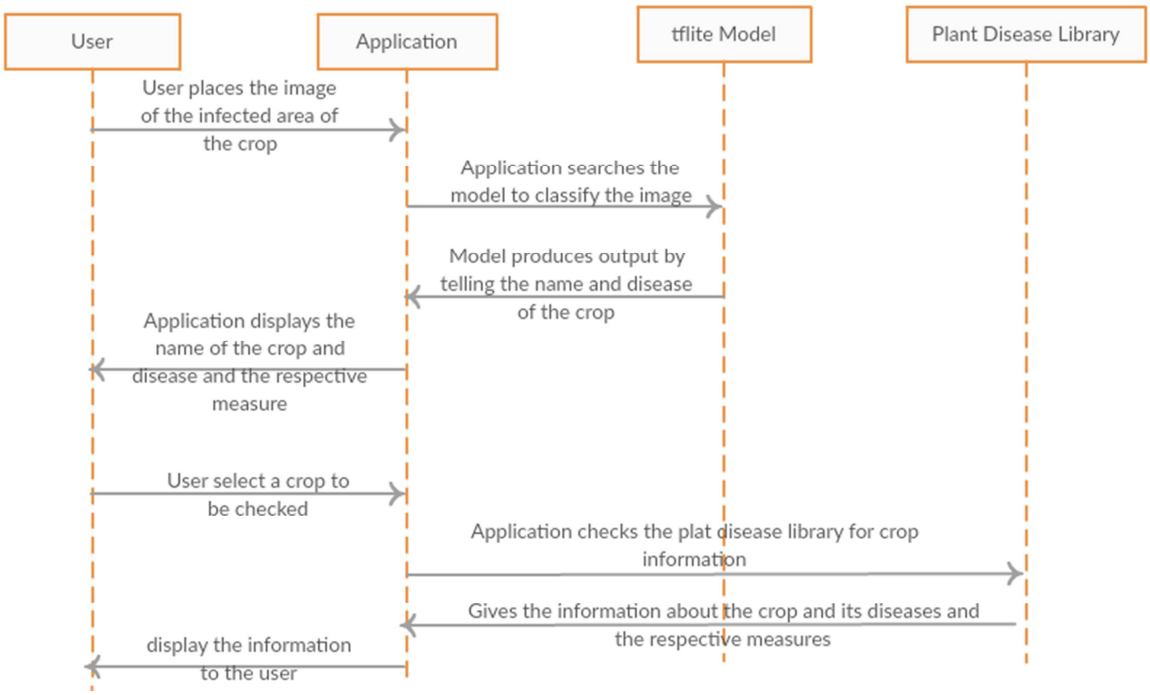


**Fig 4.8. Sequence Diagram for identifying the diseases in the leaf.**

### 4.4.4 Activity Diagram:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

**Activity:** An activity in Unified Modelling Language (UML) is a major task that must take place in order to fulfil an operation contract. Activities can be represented in activity diagrams. An activity can represent: The invocation of an operation. A step in a business process.In the following figure after farmer placing a query or an image an activity called searching is done to display the output for the user.

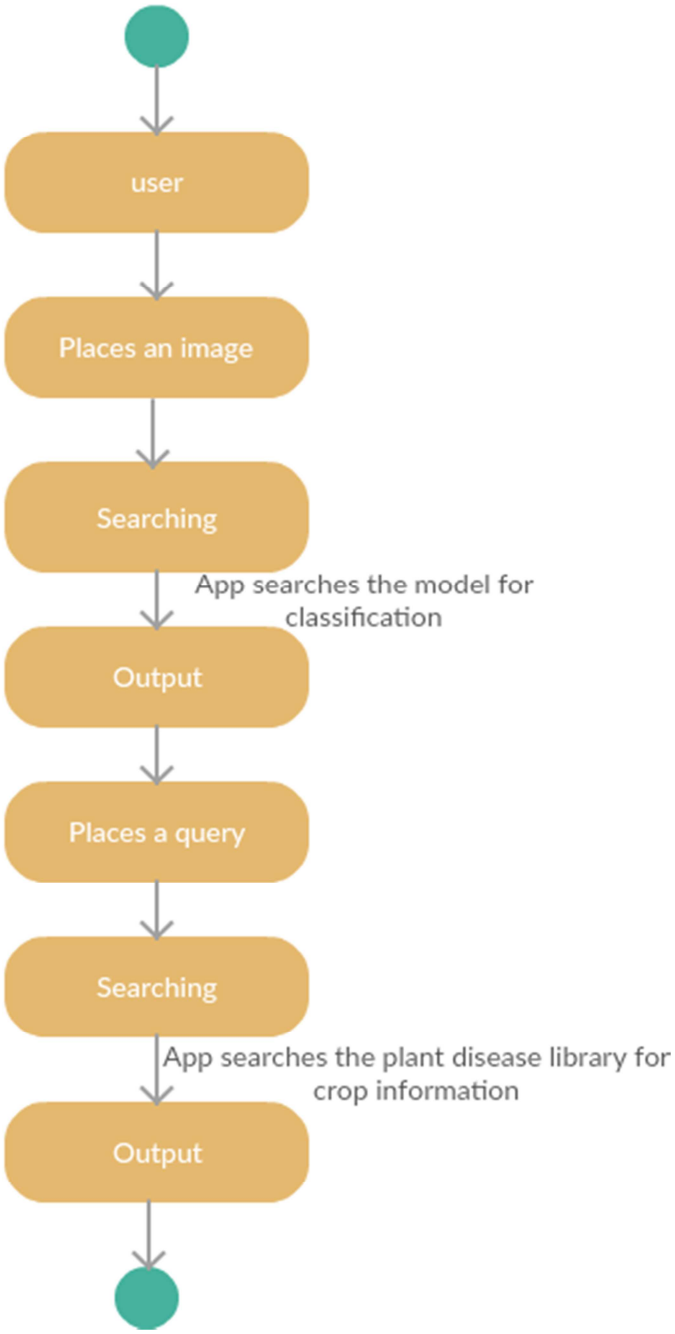


**Fig 4.9. Activity Diagram for identifying the diseases in a leaf.**

# CHAPTER-5
# IMPLEMENTATION

When training the datasets the classes of images may be imbalanced i.e., a class may have huge number of images than the other. When one class dominates the other class misclassification problem arises. To avoid this problem we does either over sampling or under sampling of the particular class. Oversampling means increasing the number of images in a class that has less number of images. Under sampling means decreasing the number of images in the class with more number of images. Oversampling of the minority may result in model over fitting as it will introduce duplicate instances, drawing from an already small pool of instances. Similarly, under the sampling of the majority, important instances that create significant differences between the two classes can be left out. In our dataset we used the over sampling method.

## 5.1 SMOTE Algorithm:

SMOTE is an over-sampling method. It creates synthetic (not duplicate) samples of the minority class. Hence making the minority class equal to the majority class. SMOTE does this by selecting similar records and altering that record one column at a time by a random amount within the difference to the neighbouring records.

## NearMiss:

NearMiss is an under-sampling technique. Instead of resampling the Minority class, using a distance, this will make the majority class equal to minority class.

**SMOTE Algorithm for Learning from Imbalanced Data:**

- The synthetic minimum oversampling Technique.
- creates Artificial Minority class data using feature space similarities.
- For all $x_i$ belongs to $S_{min,}$

  Randomly choose one of the k-nearest neighbor $x_{ij}$

  create a new sample $x_{new} = x_i + (x_{ij} - x_i) * Delta$
- For better results we can use Adaptive SMOTE algorithm.

**k-Fold Cross-Validation:**

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k. Here we consider the k value to be 3 ie., for training, verification and validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. It is a popular method because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
   - Take the group as a hold out or test data set

- Take the remaining groups as a training data set
- Fit a model on the training set and evaluate it on the test set
- Retain the evaluation score and discard the model

4. Summarize the skill of the model using the sample of model evaluation scores.

This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k − 1 folds.

## 5.2 Source code

### 5.2.1 Dart code for Capturing Image from Camera and Picking Image from Gallery:

```dart
class Screen1 extends StatefulWidget {
  @override
  _Screen1State createState() => new _Screen1State();
}
final cropKey = GlobalKey<CropState>();


class _Screen1State extends State<Screen1> {
  Future getImageGallery() async {

    var image = await ImagePicker.pickImage(source: ImageSource.gallery);
    recognizeImage(image);
    _cropImage(image);

    setState(() {
      // _image = image;
    });
  }
  Future getImageCamera() async{
    File image=await ImagePicker.pickImage(source: ImageSource.camera);
    _cropImage(image);
    //recognizeImage(image);
    setState((){
      //    _image=image;
    });
  }
  Future _cropImage(File imageFile) async {
    File croppedFile = await ImageCropper.cropImage(
      sourcePath: imageFile.path,
      ratioX: 1.0,
      ratioY: 1.0,
      maxWidth: 200,
      maxHeight: 200,
    );
    recognizeImage(croppedFile);
    setState(() {
      _image=croppedFile;
    });
  }
```

### 5.2.2 Dart code to Load tflite model and Recognizing the Disease in Leaves

```
void initState() {
  super.initState();
  loadModel();
}
Future loadModel() async {
  try {
    String res = await Tflite.loadModel(
      model: "assets/Disease2.tflite",
      labels: "assets/Disease.txt",
    );
    print(res);
  } on PlatformException {
    print('Failed to load model.');
  }
}
Uint8List imageToByteList(
    img.Image image, int inputSize, double mean, double std) {
  var convertedBytes = Float32List(1 * inputSize * inputSize * 3);
  var buffer = Float32List.view(convertedBytes.buffer);
  int pixelIndex = 0;
  for (var i = 0; i < inputSize; i++) {
    for (var j = 0; j < inputSize; j++) {
      var pixel = image.getPixel(i, j);
      buffer[pixelIndex++] = (((pixel >> 16) & 0xFF) - mean) / std;
      buffer[pixelIndex++] = (((pixel >> 8) & 0xFF) - mean) / std;
      buffer[pixelIndex++] = (((pixel) & 0xFF) - mean) / std;
    }
  }
  return convertedBytes.buffer.asUint8List();
}
Future recognizeImage(File image) async {
  var recognitions = await Tflite.runModelOnImage(
    path: image.path,
    numResults: 6,
    threshold: 0.05,
    imageMean: 127.5,
    imageStd: 127.5,
  );
  print(recognitions);
  setState(() {
    _recognitions = recognitions;
  });
}

Future recognizeImageBinary(File image) async {
  var imageBytes = (await rootBundle.load(image.path)).buffer;
  img.Image oriImage = img.decodeJpg(imageBytes.asUint8List());
  img.Image resizedImage = img.copyResize(oriImage, 224, 224);
  var recognitions = await Tflite.runModelOnBinary(
    binary: imageToByteList(resizedImage, 224, 127.5, 127.5),
    numResults: 6,
    threshold: 0.05,
  );
  print(recognitions);
  setState(() {
    _recognitions = recognitions;
  });
}
```

# CHAPTER- 6

# TESTING

The process or method of finding error/s in a software application or program so that the application functions according to the end user's requirement is called Testing.

## 6.1 Testing Strategies:

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

### Verification:

Verification is a process of evaluating a software development lifecycle's intermediate work products to check whether we are in the right track of the final product being created.

### Validation:

The system was successfully tested and implemented, ensuring that all the requirements listed in the software requirements specification are fully met. The corresponding error messages are displayed in case of erroneous input.

### Unit Testing:

Unit testing is performed on individual modules as they are completed and can be executed. It is confined to the requirements of the designer. The following two strategies can be used to test each module.

### Black Box Testing:

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

i. Incorrect or missing functions
ii. Interface errors
iii. Errors in data structure or external database access
iv. Performance errors
v. Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**White Box Testing:**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

i. Guarantee that all independent paths have been executed.
ii. Execute all logical decisions on their true and false sides.
iii. Execute all loops at their boundaries and within their operational bounds.

**Integration Testing:**

Integration testing ensures that a whole works together with software and subsystems. It tests the interface of all modules to ensure that when integrated together, the modules act properly.

**System Testing:**

Before delivery to the user, it involves in-house testing of the whole system. Its objective is to satisfy the user that the system meets all customer specification requirements. In the integration testing testers are concentrated on finding bugs/defects on integrated modules. But in the **Software System Testing** testers are concentrated on finding bugs/defects based on software application behaviour, software design and expectation of end user.

**User acceptance Testing:**

User acceptance testing (UAT) is the last phase of the software testing process. During UAT, actual software users test the software to make sure it can handle required tasks in real-world scenarios, according to specifications.

UAT is one of the final and critical software project procedures that must occur before newly developed software is rolled out to the market.

UAT is also known as beta testing, application testing or end user testing.

**Test Approach:**

Testing can be done in two ways:

☐ Bottom up approach

☐ Top down approach

**Bottom up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system.

When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down Approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper

level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

### 6.2 Test Cases:

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application.

### Test Scenario:

A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility. As a tester, we may put our self in the end user’s shoes and figure out the real-world scenarios and use cases of the Application under Test.

**Test cases for Mobile App:**

| Test Case ID | Pre-condition | Steps to reproduce | Input to the Testcase | Expected Output |
|---|---|---|---|---|
| Image_001 | Image should be taken from the gallery or should be captured | 1.Open the app.; 2. Select the language for the measures to be displayed.; 3. Select the Image based search.; 4.Observe the happening | Infected leaf | Name of the disease. |
| Text_001 | User should enter the text | 1.Open the app.;2. Select the language for the measures to be displayed.;3. Select the Text based search.;4. Observe the happening | Name of the disease | Measures to control the disease |
| Image_002 | Without registration Image should be taken from the gallery or should be captured | 1.Open the app.;2. Select the language for the measures to be displayed.;3. Select the Image based search.;4. Observe the happening | Healthy leaf | Healthy leaf |
| _002 | User should enter the text | 1.Open the app.;2. Select the language for the measures to be displayed.;3. Select the Text based search.;4. Observe the happening | Crop name | List of diseases and their measures |
| Image_003 | User should enter the text | 1.Open the app.;2. Select the language for the measures to be displayed.;3. Select the Text based search.;4. Observe the happening | Image of any object | Nothing |

| | | | | |
|---|---|---|---|---|
| Text_003 | User should enter the text | 1.Open the app.;2. Select the language for the measures to be displayed.;3. Select the Text based search.;4. Observe the happening. Open the App; 4.. On Login field, enter Username & Password; 5. Click on the Trash can;6. Observe the happening | Any text other than crop/disease | No results to display |

**Table 6.1. Test Cases for Mobile App**

# CHAPTER-7

# RESULTS

## 7.1 Experimental Results

### Selecting a language for displaying the results:

The application displays the measures in 10 native languages which makes the user to understand the measures in an easy and effective manner. While opening the app, first the user has to select the language in which he wanted to view the measures.

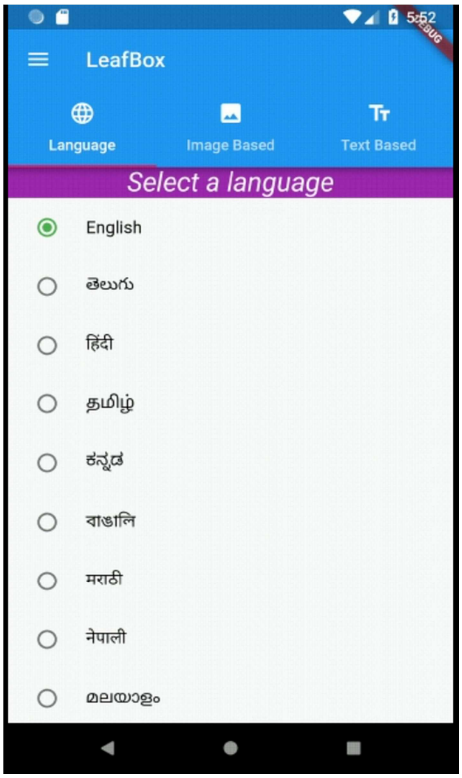


**Fig 7.1.Selection of language**

### Two-way approach for taking inputs:

The application accepts two inputs from the user that is text and the image input. The user has to select the image based or text based after selecting the language.

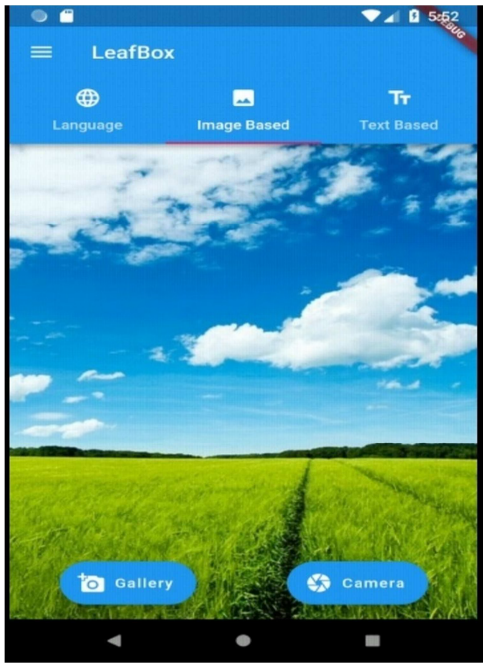


**Fig 7.2. Two way approach for taking inputs**

**Image based approach:**

Here the user provides the input in the form of an image. The image may be captured using the camera or taken from the gallery. Then the app uses the loaded deep learning model to detect the diseases in the leaf.

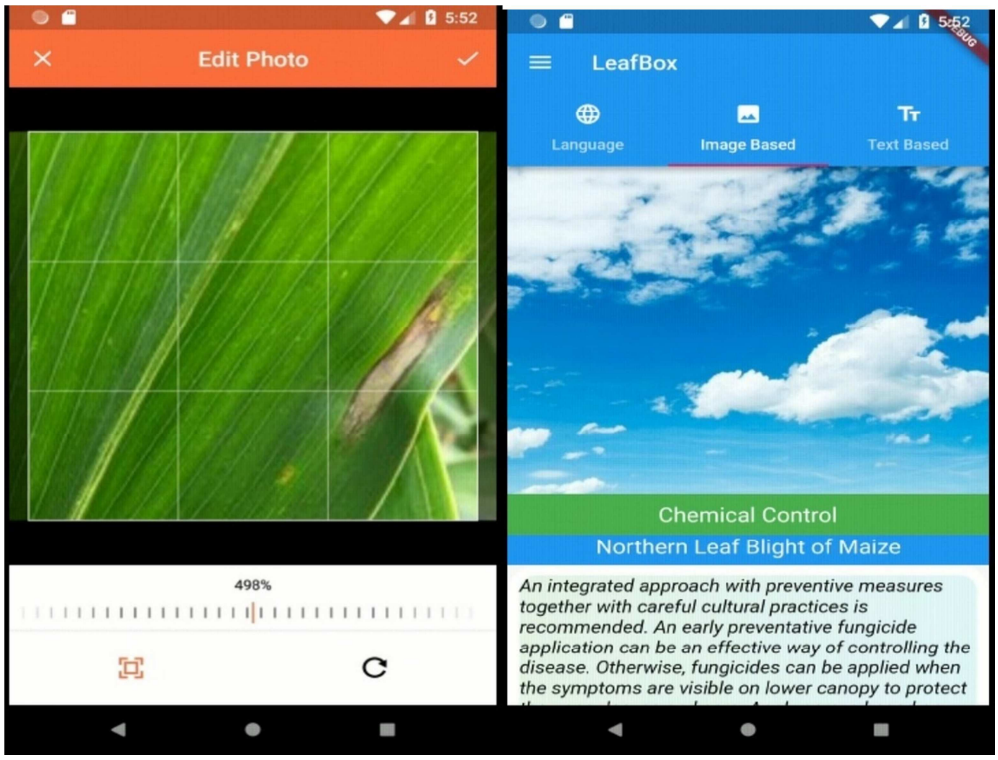


**Fig 7.3. Image based Approach**

**Text based approach:**

The user provides the name of the crop or the disease that the user wants to know. Then the app searches the preloaded disease library to display the respective measures. It displays the symptoms, chemical control and biological control.

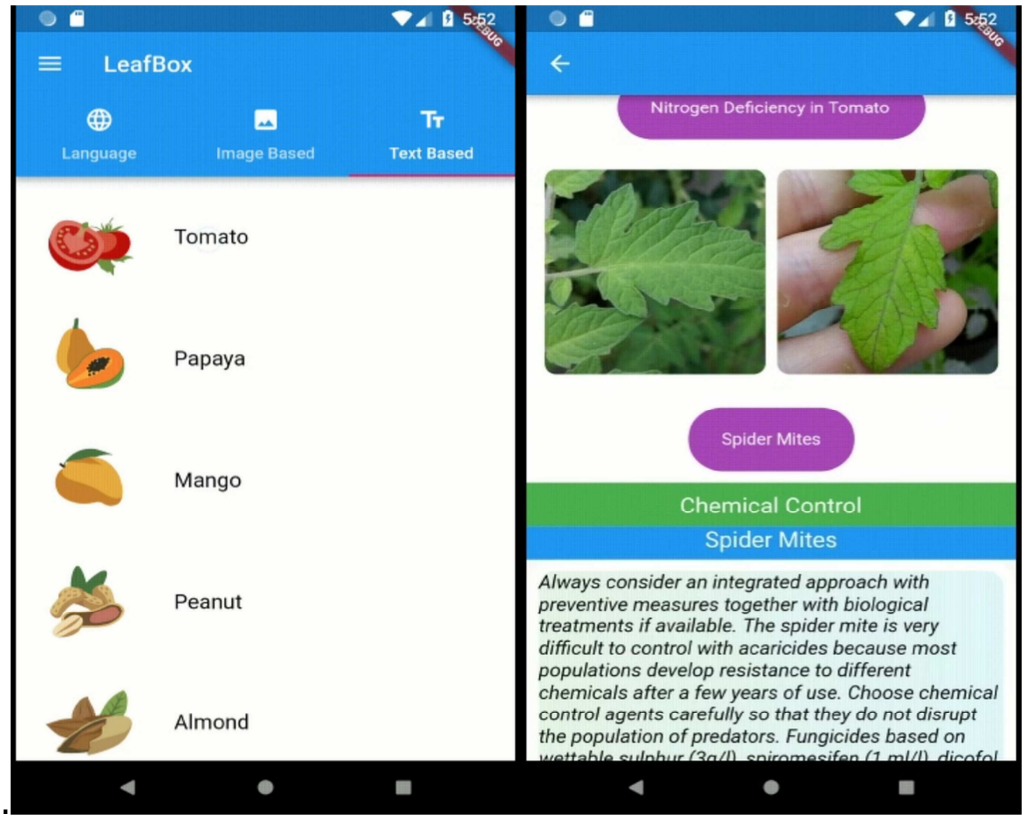


**Fig 7.4. Text based Approach**

Once the symptoms are matched, the checkbox is selected. Then it displays the measures and controls in the respective language.

**Displaying information in another language:**

The application displays the controls, crop names, diseases names in the selected language. Here we display the information in 10 different languages.


**Fig 7.5. Displaying Information in respective language**

# CHAPTER-8
# PERFORMANCE EVALUATION

Performance measurement is generally defined as regular management of outcomes and results, which generates reliable data on the effectiveness and efficiency of programs.To train a neural network you need some measure of error between computed outputs and the desired target outputs of the training data.

**Loss Function:**

Loss function is a method of evaluating how well your data set is modeling your algorithm. If the predictions are completely off, a higher number will be produced by the loss function. If they are pretty good, a lower number will be issued. As you change your algorithm pieces to try and improve your model, your loss function will tell you if you get anywhere. Our loss function will simply measure the absolute difference between our prediction and the actual value for every prediction we make. In mathematical notation, it might look something like abs(y_predicted – y). It doesn't matter if the predictions were too high or too low. All that matters is how incorrect the predictions were. This is not a feature of all loss functions, in fact, loss function will vary significantly based on the domain and unique context of the problem that you're applying machine learning to.Different flavours of loss functions are Mean squared error, Categorical cross entropy and likelihood error.

**Categorical Cross Entropy:**

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So, predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

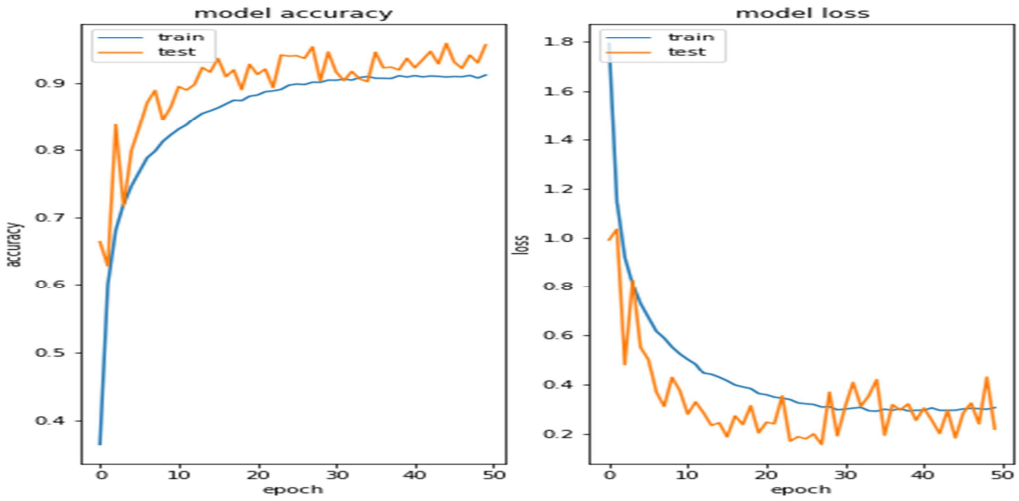


**Fig 8.1. Model accuracy and model loss**

The graph above shows the range of possible loss values given a true observation. As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Log loss penalizes both types of errors, but especially those predictions that are confident and wrong!

Cross-entropy and log loss are slightly different depending on context, but in machine learning when calculating error rates between 0 and 1 they resolve to the same thing.

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

$$-(y\log(p)+(1-y)\log(1-p))-(y\log\ (p)+(1-y)\log\ (1-p))$$

If M>2 (i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.

$$-\sum c=1Myo,\ c\log(po,c)$$

Where

- M - number of classes (dog, cat, fish)
- y - binary indicator (0 or 1) if class label cc is the correct classification for observation o
- p - predicted probability observation o is of class c

# CHAPTER-9
# CONCLUSION

Agricultural productivity is seriously affected by Diseases and Pests which leads to less profitability to the farmer. Farmers are unaware of a wide variety of diseases and proper measures needed to be taken to prevent them. This mobile application equips the farmer with a lot of information regarding the symptoms of the disease, controlling and preventive measures needed to be taken. It provides two major functionalities, Image-based and Text based. By using this application's Image-based functionality farmer can easily detect the disease by capturing the infected part of the leaf. After detecting the disease the application provides the symptoms of the disease for cross-checking and controlling and preventive measures are displayed then. Text-based features provide a list of plants and images of affected parts of the leaves of those plants. By clicking on a particular disease image it will display the preventive and control measures. It also offers users a privilege to change the language so that the application gets translated to a selected regional language which leads to a better understanding of the solution provided. Moreover, this application works offline so that the farmers in remote areas can use it. It is developed using latest technologies like Flutter framework and Dart programming so it is lightweight and handy. Coming to some of the setbacks of this application, providing upgrades for this application is difficult as it is offline. The size of the application is also somewhat high because data needs to be replicated in different languages for multi-language support. Once we switch to the cloud platform this overhead will not be there. As of now, this application works for some plants only as data for all the plants is not available. We can incorporate some other functionality into this application such as providing a platform for selling crops and also providing information regarding various government schemes to the farmers.

# CHAPTER-10

# REFERENCES


**[1]** Varshney, Sujeet, and TarunDalal. "Plant Disease Prediction using Image Processing Techniques-A Review." *International Journal of Computer Science and Mobile Computing* 5.5 (2016): 394-398.

**[2]** Mohanty, Sharada P., David P. Hughes, and Marcel Salathé. "Using deep learning for image-based plant disease detection." *Frontiers in plant science* 7 (2016): 1419.

**[3]** Sladojevic, Srdjan, et al. "Deep neural networks based recognition of plant diseases by leaf image classification." *Computational intelligence and neuroscience* 2016 (2016).

**[4]** Mohanapriya, K., and Mr M. Balasubramani. "A Study on Identification of Unhealthy Plant Leaves using Image Processing with PCA, LDA Techniques."

**[5]** Ferentinos, Konstantinos P. "Deep learning models for plant disease detection and diagnosis." *Computers and Electronics in Agriculture* 145 (2018): 311-318.

**[6]** Fuentes, Alvaro, et al. "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition." *Sensors* 17.9 (2017): 2022.

**[7]** Giriraja, C.V., et al. "Plant health analyser." 2017 International conference on Advances in Computing Communications and Informatics(ICACCI). IEEE, 2017.

**[8]** Wang, Guan, Yu Sun, and Jianxin Wang. "Automatic image-based plant disease severity estimation using deep learning." *Computational intelligence and neuroscience* 2017 (2017).

**[9]** Belsha, N., and N. Hariprasad. "An approach for identification of infections in vegetables using image processing techniques." *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. IEEE, 2017.

**[10]** Amara, Jihen, Bassem Bouaziz, and AlsayedAlgergawy. "A Deep Learning-based Approach for Banana Leaf Diseases Classification." *BTW (Workshops)*. 2017.

**[11]** Pinto, Loyce Selwyn, et al. "Crop disease classification using texture analysis." *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2016.

**[12]** Thorat, Apeksha, Sangeeta Kumari, and Nandakishor D. Valakunde. "An IoT based smart solution for leaf disease detection." *2017 International Conference on Big Data, IoT and Data Science (BID)*. IEEE, 2017.

**[13]** Ramcharan, Amanda, et al. "Deep learning for image-based cassava disease detection." *Frontiers in plant science* 8 (2017): 1852.

**[14]** Tm, Prajwala, et al. "Tomato Leaf Disease Detection Using Convolutional Neural Networks." *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE, 2018.

**[15]** Ubbens, Jordan R., and Ian Stavness. "Deep plant phenomics: a deep learning platform for complex plant phenotyping tasks." *Frontiers in plant science* 8 (2017): 1190.

**[16]** Kamilaris, Andreas, and Francesc X. Prenafeta-Boldu. "Deep learning in agriculture: A survey." *Computers and Electronics in Agriculture* 147 (2018): 70-90.

**[17]** Lu, Yang, et al. "Identification of rice diseases using deep convolutional neural networks." *Neurocomputing* 267 (2017): 378-384.

**[18]** Brahimi, Mohammed, Kamel Boukhalfa, and Abdelouahab Moussaoui. "Deep learning for tomato diseases: classification and symptoms visualization." *Applied Artificial Intelligence*31.4 (2017): 299-315.

**[19]** Hughes, David, and Marcel Salathé. "An open access repository of images on plant health to enable the development of mobile disease diagnostics." *arXiv preprint arXiv:1511.08060* (2015).

**[20]** DeChant, Chad, et al. "Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning." *Phytopathology* 107.11 (2017): 1426-1432.

**[21]** Meghana, K., et al. "GPU Accelerated Code Optimization: Leaf Disease Detection."

# CHAPTER-11

# APPENDIX

**Dart code for Capturing Image from Camera and Picking Image from Gallery:**

```
class Screen1 extends StatefulWidget {
  @override
  _Screen1State createState() => new _Screen1State();
}
final cropKey = GlobalKey<CropState>();
class _Screen1State extends State<Screen1> with
AutomaticKeepAliveClientMixin<Screen1> {
  @override
  bool get wantKeepAlive => true;
  String enDisease;
  List data;
  List lang;
  String disease;
    Future getImageGallery() async {
      var image = await ImagePicker.pickImage(source:
ImageSource.gallery);
      recognizeImage(image);
      _cropImage(image);

      setState(() {
      // _image = image;
      });
    }
    Future getImageCamera() async{
     File image=await ImagePicker.pickImage(source:
ImageSource.camera);
      _cropImage(image);
```

```
    //recognizeImage(image);
    setState((){
  //    _image=image;
    });
  }
  Future _cropImage(File imageFile) async {
    File croppedFile = await ImageCropper.cropImage(
      sourcePath: imageFile.path,
      ratioX: 1.0,
      ratioY: 1.0,
      maxWidth: 224,
      maxHeight: 224,
    );
```

**Dart code to Load tflite model and Recognizing the Disease in Leaves :**

```
void initState() {
    super.initState();
    loadModel();
  }
  Future loadModel() async {
    try {
      String res = await Tflite.loadModel(
        model: "assets/Disease2.tflite",
        labels: "assets/Disease2.txt",
      );
      print(res);
    } on PlatformException {
      print('Failed to load model.');
    }
  }
  Uint8List imageToByteList(
```

```
    img.Image image, int inputSize, double mean, double std) {
  var convertedBytes = Float32List(1 * inputSize * inputSize * 3);
  var buffer = Float32List.view(convertedBytes.buffer);
  int pixelIndex = 0;
  for (var i = 0; i < inputSize; i++) {
    for (var j = 0; j < inputSize; j++) {
      var pixel = image.getPixel(i, j);
      buffer[pixelIndex++] = (((pixel >> 16) & 0xFF) - mean) / std;
      buffer[pixelIndex++] = (((pixel >> 8) & 0xFF) - mean) / std;
      buffer[pixelIndex++] = (((pixel) & 0xFF) - mean) / std;
    }
  }
  return convertedBytes.buffer.asUint8List();
}
Future recognizeImage(File image) async {
  var recognitions = await Tflite.runModelOnImage(
    path: image.path,
    numResults: 6,
    threshold: 0.05,
    imageMean: 127.5,
    imageStd: 127.5,
  );
  setState(() {
    _recognitions = recognitions;
  });
}
Future recognizeImageBinary(File image) async {
  var imageBytes = (await rootBundle.load(image.path)).buffer;
  img.Image oriImage = img.decodeJpg(imageBytes.asUint8List());
  img.Image resizedImage = img.copyResize(oriImage, 224, 224);
  var recognitions = await Tflite.runModelOnBinary(
    binary: imageToByteList(resizedImage, 224, 127.5, 127.5),
```

```
      numResults: 6,
      threshold: 0.05,
    );
    print(recognitions);
    setState(() {
      _recognitions = recognitions;
    });
  }
recognizeImage(croppedFile);
   enDisease= _recognitions[0]["label"];
    if(enDisease=='2')
    {
      showDialog(context: context,
        builder:(BuildContext context)
        {
          return AlertDialog(
            shape: RoundedRectangleBorder(borderRadius:
            BorderRadius.all(Radius.circular(15))),
            content: Container(
              height: 100,
              child: Center(child: new Text(
                "Healthy leaf. No Diseases found ",
                style: TextStyle(fontSize: 18),),),
            ),
          );
        },
      );
    }
    else if(enDisease==null)
    {
      showDialog(context: context,
        builder:(BuildContext context)
```

```
    {
      return AlertDialog(
        shape: RoundedRectangleBorder(borderRadius:
        BorderRadius.all(Radius.circular(15))),
        content: Container(
          height: 100,
          child: Center(child: new Text(
            "Dataset Not found. Please refer in Text section",
            style: TextStyle(fontSize: 18),),),
        ),
      );
    },
  );
}
if(enDisease=='0' || enDisease=='1')
{
  showDialog(context: context,
    builder:(BuildContext context)
    {
      return AlertDialog(
        shape: RoundedRectangleBorder(borderRadius:
        BorderRadius.all(Radius.circular(15))),
        content: Container(
          height: 100,
          child: Center(child: new Text(
            "Disease Not available..",
            style: TextStyle(fontSize: 18),),),
        ),
      );
    },
  );
}
```

```
      else {
        data = Lib.getData(language[0]);
        lang = Lib.getData(language[_lanIndex]);
        SecondRoute s = new SecondRoute(en: data, data: lang);
        disease = lang[Lib.getIndex(enDisease,
'dis_name')[0]]['dis_name'];
        print(data);
        print(disease);
        print(enDisease);
        s.newTaskModalBottomSheet1(context, enDisease, disease);
      }
    }
```

**Dart code to Load the plant disease library and display the Diseases and the respective images for the crop :**

```
class ListTutorial extends StatelessWidget {
  final String sip;
  ListTutorial ({ this.sip});
  @override
  Widget build(BuildContext context) {
    return new Container(
      height: 75,
     child:Row(
     children:[
         new Expanded(
           child:Container(
             height: 200.0,
             width: 120.0,
        child: Column(
           children:[
              SizedBox(height: 5,),
           Chip(
```

```
            label: Text(sip,style: TextStyle(color: Colors.white,fontSize:
15),),
            backgroundColor: Colors.purple[400],
            padding: const EdgeInsets.only(left: 20.0,right: 20.0,top:
20.0,bottom: 20.0),
          ),],),
         ),),],),
   ); }
}
class ListTutorial1 extends StatelessWidget {
  final String sipp;
  ListTutorial1 ({ this.sipp});
  @override
  Widget build(BuildContext context) {
    return new Container(
      height: 220,
      child: Row(
        children: [
      new Expanded(
      child:Container(
      height: 200.0,
        width: 120.0,
        margin: EdgeInsets.all(5.0),
        decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(10.0),
            image: DecorationImage(
                image: AssetImage(sipp),
                fit: BoxFit.cover
            ),
        ),
     ),),],),
   ); }
```

```
}
class SecondRoute extends StatelessWidget
{
  final String plant;
  final List data;
  final List en;
  SecondRoute({this.plant,this.data,this.en});
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    List  diseases=Lib.getDiseases(plant,data);
    List enDiseases=Lib.getDiseases(plant,en);
    List images=Lib.getDiseaseImages(enDiseases,data);
    var bd=ListView.builder (
    itemCount: diseases.length ,
    itemBuilder: (BuildContext context,int index)
   {
            return new GestureDetector (
             onTap:(){

_newTaskModalBottomSheet(context,enDiseases[index],diseases[index]);
 },
child: new Container(
color: Colors.white,
margin: new EdgeInsets.only(top: 10.0, bottom: 10.0,left: 10.0,right:
10),
          child: new Column(
           children:[
              new ListTutorial(sip: diseases[index],),
              new SizedBox(height: 10,),

               GridView.builder(
```

```
                primary:false,
               gridDelegate:new
SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 2,
                 ),
                shrinkWrap: true,
                itemCount: images[index].length,
               //  scrollDirection: Axis.horizontal,
                itemBuilder: (BuildContext context,int i)
                {
                  return new Container (
                    height: 200,
                    child: new ListTutorial1(sipp: images[index][i],),
                  );
                }
             )
          ],
         ), ),
 );
 },
 );
return new  Scaffold(
backgroundColor: Colors.white,
appBar: AppBar(
 ),
     body: bd,
   );
  }
   void _newTaskModalBottomSheet (context,String disease,String
lanDisease){
    showBottomSheet(
        context: context,
        builder: (BuildContext bc){
```

```
        return Container(
          color: Colors.white,
          child: new Wrap(
            children: <Widget>[
              Container(
                color:Colors.green,
                alignment: Alignment.center,
                height:40,
                width:double.infinity,
                  child:  Text("Check Symptoms",style: TextStyle(color:
Colors.white,fontSize: 20),),
                 ),
              new SizedBox(
                height:10,
              ),
              Container(
                color: Colors.blue,
                alignment: Alignment.center,
                width:double.infinity,
                child:new SizedBox(
                  height:30,
                  child:Text(lanDisease,style: TextStyle(color:
Colors.white,fontSize: 20)),
                ),),
              new SingleChildScrollView(
             child: Container(
                  decoration: new BoxDecoration(
                    borderRadius:new  BorderRadius.circular(25.0),
                    border: new Border.all(
                      width: 10.0,
                     color: Colors.white,
                    ),
```

```
                    gradient: new LinearGradient(
                      colors: [Colors.orange[50], Colors.cyan[50]],
                    )
                  ),
child:Text(Lib.getProp(disease,data,'symptoms'),style:TextStyle(color:
Colors.black,fontSize: 17,fontStyle: FontStyle.italic)),
               ),),
              new CheckboxListTile(
                title: const Text('Check if symptoms are
matched',style:TextStyle(color: Colors.purple,fontSize: 20,)),
                value: false,
                onChanged: (bool value) {
                  newTaskModalBottomSheet1(context,disease,lanDisease);
                },
              ),
            ],
          ),  );
        }
    );
  }
  void finalsheet(context,String lanDisease,String prop,String
name,String disease){
    showBottomSheet(
        context: context,
        builder: (BuildContext bc){
          return Container(
            child: new Wrap(
              children: <Widget>[
                  Container(
                    color:Colors.green,
                    alignment: Alignment.center,
                    height:40,
```

```
                    width:double.infinity,
                    child:  Text(name,style: TextStyle(color:
Colors.white,fontSize: 20),),
                  ),
              new Wrap(
               children:[
                 Container(
                   color:Colors.blue,
                   alignment: Alignment.center,
                   width:double.infinity,
                   child:new SizedBox(
                     height:30,
                     child:Text(lanDisease,style: TextStyle(color:
Colors.white,fontSize: 20)),
                   ),),
                 Container(
                      decoration: new BoxDecoration(
                        borderRadius:new  BorderRadius.circular(25.0),
                        border: new Border.all(
                          width: 10.0,
                          color: Colors.white,
                        ),
                          gradient: new LinearGradient(
                            colors: [Colors.green[50], Colors.cyan[50]],
                          )
                      ),

child:Text(Lib.getProp(disease,data,prop),style:TextStyle(color:
Colors.black,fontSize: 17,fontStyle: FontStyle.italic)),
                    ),],),
          ],),);
          },
```

```
    );
  }
  void newTaskModalBottomSheet1(context,String disease,String
lanDisease){
    print(Lib.getProp(disease,data,'nutshell'));
    List nutshell=Lib.getProp(disease,data,'nutshell');
    final markDownData = nutshell.map((x) => "- $x\n").reduce((x, y) =>
"$x$y");
    showBottomSheet(
        context: context,
        builder: (BuildContext bc){
          return Container(
            color: Colors.white,
            child: new Wrap(
              children: <Widget>[
                Container(
                  alignment: Alignment.center,
                  height:40,
                  width:double.infinity,
                  color: Colors.green,
                  child:  Text("Summary",style: TextStyle(color:
Colors.white,fontSize: 20),),
                ),
                Container(
                  alignment: Alignment.center,
                  color:Colors.blue,
                  width:double.infinity,
                  child:new SizedBox(
                  height:25,
                  child:Text(lanDisease,style: TextStyle(color:
Colors.white,fontSize: 18)),
                ),),
```

```
                Container(
                  alignment: Alignment.centerLeft,
                  width:double.infinity,

                  color: Colors.white,
                  height:180,
                    child:Markdown(data: markDownData)),
              new SizedBox(width: 20,),
              new RaisedButton(
                  color: Colors.blue,
                padding: EdgeInsets.only(left: 10,right: 10),
                child: new Text("Chemical Control",style:
TextStyle(decorationColor: Colors.black,color: Colors.white,fontSize:
20),),
                shape: new RoundedRectangleBorder(borderRadius: new
BorderRadius.circular(10.0),),
                onPressed:
(){finalsheet(context,lanDisease,'c_control',"Chemical Control",disease);}
              ),
              new SizedBox(width: 20,),
              new RaisedButton(
                  color: Colors.blue,
                  child: new Text("Natural Control",style:
TextStyle(decorationColor: Colors.black,color: Colors.white,fontSize:
20),),
                  padding: EdgeInsets.only(left: 10,right: 10),
                  shape: new RoundedRectangleBorder(borderRadius: new
BorderRadius.circular(10.0),),
                  onPressed:
(){finalsheet(context,lanDisease,'b_control',"Natural Control",disease);}
              ),
              new SizedBox(width: 80,),
```

```
            new RaisedButton(
                color: Colors.green,
                child: new Text("Preventive Measures",style:
TextStyle(decorationColor: Colors.blue,color: Colors.white,fontSize: 20),),
                padding: EdgeInsets.only(left: 30,right: 30),
                shape: new RoundedRectangleBorder(borderRadius: new
BorderRadius.circular(10.0),),
                onPressed:
(){finalsheet(context,lanDisease,'measures',"Preventive
Measures",disease);}
             ),
            ],
           ),
        );}
    );
  }
}
class ThirdRoute extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Scaffold(
      appBar:new AppBar( title: Text("Text solution"),),
      body:new Container(
          width: 100,
          height: 100,
      ),);
  }
}
```

**Dart code for displaying the output in a selected language:**

```
class Lib {
```

```
static jsonToList(File f) {
  List x = JSON.jsonDecode(f.readAsStringSync());
  RegExp regExp = new RegExp(r"', '");
  for (int i = 0; i < x.length; i++) {
    List r = x[i]['sci_name'].split(regExp);
    r[0] = r[0].substring(1);
    r[r.length - 1] = r[r.length - 1].replaceAll("'", "");
    x[i]['sci_name'] = r;
    r = x[i]['nutshell'].split(regExp);
    r[0] = r[0].substring(1);
    r[r.length - 1] = r[r.length - 1].replaceAll("'", "");
    x[i]['nutshell'] = r;
    x[i]['file_name'] = x[i]['file_name'].split(r",");
  }
  return x;
}
static getDiseases(String plant, List data) {
  List diseases = new List();
  for (int i in getIndex(plant, 'sci_name')) {
    diseases.add(data[i]['dis_name']);
  }
  return diseases;
}
static getPlants(List x) {
  List plant = new List();
  for (int i = 0; i < x.length; i++) {
    x[i]['sci_name'].forEach((e) => plant.add(e));
  }
  return plant.toSet().toList();
}
static getProp(String disease, List data, String prop) {
  int index = getIndex(disease, 'dis_name')[0];
```

```
  return data[index][prop];
}
static getFromZip(String lang) {
 Stringpath='/data/user/0/sq.flutter.tfliteexample/app_flutter/.jsonFile/';
  List<int> bytes = new File(path + 'json.zip').readAsBytesSync();
  Archive archive = new ZipDecoder().decodeBytes(bytes);
  lang = lang + '.json';
  for (ArchiveFile file in archive) {
    String filename = file.name;
    if (filename == lang &&
        (FileSystemEntity.typeSync(path + 'json/' + filename) ==
            FileSystemEntityType.notFound)) {
      List<int> data = file.content;
      new File(path + 'json/' + filename)
        ..createSync(recursive: true)
        ..writeAsBytesSync(data);
    }
  }
  return path + 'json/' + lang;
}
static getData(String lang) {
  return jsonToList(new File(getFromZip(lang)));
}
static getIndex(String item, String col) {
  List data = getData('en');
  List index = new List();
  for (int i = 0; i < data.length; i++) {
    if (data[i][col].contains(item)) index.add(i);
  }
  return index;
}
static formIndex(int i, List data, String prop) {
```

```
    return data[i][prop];
  }
  static getDiseaseImages(List diseases, List data) {
    List images = new List();
    List newImages = new List();
    for (int i = 0; i < diseases.length; i++) {
      images.add(getProp(diseases[i], data, 'file_name'));
    }
    for (int i = 0; i < images.length; i++) {
      List list = new List();
      list = images[i];
      for (int j = 0; j < list.length; j++) {
        if (list[j].substring(0, 6) == "assets")
          break;
        list[j] = list[j].substring(1, list[j].length - 1);
        list[j] = 'assets/' + list[j];
      }
      newImages.add(list);
    }
    return newImages;
  }
  static getPlantImages(List plants) {
    List images =new List();
    for(int i=0;i<plants.length;i++){
      images.add('assets/plants/'+plants[i]+'.png');
    }
    return images;
  }
}
```