

Fertilizer Recommendation System For Disease Prediction

PROJECT REPORT

Submitted by

TEAM ID: PNT2022TMID35918

Shakeel Mohammed G	2019504584
Thanish Malai P	2019504598
Jayaraj D	2019504530
Saranraj A K	2019504580

TABLE OF CONTENTS

1. INTRODUCTION

- a. Project Overview
- b. Purpose

2. LITERATURE SURVEY

- a. Literature Review
- b. Existing Problem
- c. References
- d. Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- a. Empathy Map Canvas
- b. Ideation & Brainstorming
- c. Proposed Solution
- d. Problem Solution fit

4. REQUIREMENT ANALYSIS

- a. Functional requirement
- b. Non-Functional requirements

5. PROJECT DESIGN

- a. Data Flow Diagrams
- b. Solution & Technical Architecture
- c. User Stories

6. PROJECT PLANNING & SCHEDULING

- a. Sprint Planning & Estimation
- b. Sprint Delivery Schedule
- c. Reports from JIRA

7. CODING & SOLUTIONING

- a. Feature 1
- b. Feature 2

8. TESTING

- a. Test Cases
- b. User Acceptance Testing

9. RESULTS

- a. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques.

1.1. Project Overview:

An Automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases' changes in cultivation method and inadequate plant protection techniques and suggest all the precautions that can be taken for those diseases.

1.2. Purpose:

- To Detect and recognize the plant diseases and to recommend fertilizer, it is necessary to identify the diseases and to recommend getting different and useful features needed for the purpose of analyzing later.
- To provide symptoms in identifying the disease at its earliest. Hence the authors proposed and implemented new fertilizers Recommendation System for Crop Disease Prediction

2. LITERATURE SURVEY

2.1. Literature Review:

[1] The proposed method uses SVM to classify tree leaves, identify the disease and suggest the fertilizer. The proposed method is compared with the existing CNN based leaf disease prediction. The proposed SVM technique gives a better result when compared to existing CNN. For the same set of images, F-Measure for CNN is 0.7 and 0.8 for SVM, the accuracy of identification of leaf disease of CNN is 0.6 and SVM is 0.8.

Advantages: The prediction and diagnosing of leaf diseases depend on segmentation such as segmenting the healthy tissues from diseased tissues of leaves.

Disadvantages: This further research is implementing the proposed algorithm with the existing public datasets. Also, various segmentation algorithms can be implemented to improve accuracy. The proposed algorithm can be modified further to identify the disease that affects the various plant organs such as stems and fruits.

[2] Detection of Leaf Diseases and Classification using Digital Image Processing International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), IEEE, 2017.

Advantages: The system detects diseases on citrus leaves with 90% accuracy.

Disadvantages: System only able to detect disease from citrus leaves. The main objective of this paper is image analysis & classification techniques for detection of leaf diseases and classification. The leaf image is firstly preprocessed and then does further work. K-Means Clustering is used for image segmentation and then the system extracts the GLCM features from disease detected images. The disease classification was done through the SVM classifier.

Algorithm used: Gray-Level Co-Occurrence Matrix (GLCM) features, SVM, K-Means Clustering.

[3] Semi-automatic leaf disease detection and classification system for soybean culture IET Image Processing, 2018.

Advantages: The system helps to compute the disease severity.

Disadvantages: The system uses leaf images taken from an online dataset, so it cannot be implemented in real time. This paper mainly focuses on the detecting and classifying the leaf disease of soybean plants. Using SVM the proposed system classifies the leaf disease in 3 classes like i.e., downy mildew, frog eye, and Septoria leaf blight etc. The proposed system gives maximum average classification accuracy reported is ~90% using a big dataset of 4775 images. **Algorithm used:** SVM.

[4] Cloud Based Automated Irrigation and Plant Leaf Disease Detection System Using an Android Application. International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017.

Advantages: It is a simple and cost-effective system for plant leaf disease detection.

Disadvantages: Any H/w failures may affect the system performance. The current paper proposes an android application for irrigation and plant leaf disease detection with cloud and IoT. For monitoring irrigation systems, they use soil moisture and temperature sensor, and sensor data send to the cloud. The user can also detect plant leaf disease. K-means clustering used for feature extraction. **Algorithm used:** K-means clustering, Other than this there are some other levels which can be used for sentimental analysis these are- document level, sentence level, entity and aspect level to study positive and negative, interrogative, sarcastic, good and bad functionality, sentiment without sentiment, conditional sentence and author and reader understanding points.

[5] The author proposes a method which helps us predict crop yield by suggesting the best crops. It also focuses on soil types in order to identify which crop should be planted in the field to increase productivity. In terms of crop yield, soil types are vital. By incorporating the weather details of the previous year into the equation, soil information can be obtained.

Advantages: It allows us to predict which crops would be appropriate for a given climate. Using the weather and disease related data sets, the crop quality can also be

improved. Prediction algorithms help us to classify the data based on the disease, and data extracted from the classifier is used to predict soil and crop.

Disadvantages: Due to the changing climatic conditions, accurate results cannot be predicted by this system.

[6] The current work examines and describes image processing strategies for identifying plant diseases in numerous plant species. BPNN, SVM, K-means clustering, and SGDM are the most common approaches used to identify plant diseases.

Disadvantages: Some of the issues in these approaches include the impact of background data on the final picture, optimization of the methodology for a specific plant leaf disease, and automation of the technique for continuous automated monitoring of plant leaf diseases in real-world field circumstances.

[7] The proposed method uses SVM to classify tree leaves, identify the disease and suggest the fertilizer. The proposed method is compared with the existing CNN based leaf disease prediction. The proposed SVM technique gives a better result when compared to existing CNN. For the same set of images, F-Measure for CNN is 0.7 and 0.8 for SVM, the accuracy of identification of leaf disease of CNN is 0.6 and SVM is 0.8.

Advantages: The prediction and diagnosing of leaf diseases depend on segmentation such as segmenting the healthy tissues from diseased tissues of leaves.

Disadvantages: This further research is implementing the proposed algorithm with the existing public datasets. Also, various segmentation algorithms can be implemented to improve accuracy. The proposed algorithm can be modified further to identify the disease that affects the various plant organs such as stems and fruits.

2.2. Existing Problem:

- Adequate mineral nutrition is central to crop production. However, it can also exert considerable Influence on disease development. Fertilizer application can increase or decrease development of diseases caused by different pathogens, and the mechanisms responsible are complex, including effects of nutrients on plant growth, plant resistance mechanisms and direct effects on the pathogen. The effects of mineral nutrition on plant disease and the mechanisms responsible for those effects have been dealt with comprehensively elsewhere.
- Most research articles use humidity, moisture, and temperature sensors near the plant's root, with an external device handling all of the data provided by the sensors and transmitting it directly to an Android application. It was created to measure the approximate values of temperature, humidity and moisture sensors that were programmed into a microcontroller to manage the amount of water.

2.3. References:

[1] Semi-automatic leaf disease detection and classification system for soybean culture IET Image Processing, 2018.

[2] Cloud Based Automated Irrigation and Plant Leaf Disease Detection System Using an Android Application. International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017.

[3] Ms. Kiran R. Gavhale, Ujwalla Gawande, Plant Leaves Disease detection using Image Processing Techniques, January 2014.

https://www.researchgate.net/profile/UjwallaGawande/publication/314436486_An_Overview_of_the_Research_on_Plant_Leaves_Disease_detection_using_Image_Processing_Techniques/links/5d3710664585153e591a3d20/An-Overviewof-the-Research-on-PlantLeaves-Diseaedetection-using-Image-ProcessingTechniques.pdf

[4] Duan Yan-e, Design of Intelligent Agriculture Management Information System Based on IOT, IEEE,4th, Fourth International reference on Intelligent Computation Technology and Automation, 2011.

<https://ieeexplore.ieee.org/document/5750779>

- [5] R. Neela, P. Fertilizers Recommendation System For Disease Prediction In Tree Leave International journal of scientific & technology research volume 8, issue 11, November 2019.<http://www.ijstr.org/final-print/nov2019/Fertilizers-Recommendation-System-ForDisease-PredictionIn-Tree-Leave.pdf>
- [6] Swapnil Jori¹, Rutuja Bhalshankar², Dipali Dhamale³, Sulochana Sonkamble , Healthy Farm: Leaf Disease Estimation and Fertilizer Recommendation System using Machine Learning, International Journal of All Research Education and Scientific Methods (IJARESM), ISSN:2455-6211.
- [7] Detection of Leaf Diseases and Classification using Digital Image Processing International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), IEEE, 2017.
- [8] Shloka Gupta, Nishit Jain, Akshay Chopade, Farmer's Assistant: A Machine Learning Based Application for Agricultural Solutions.

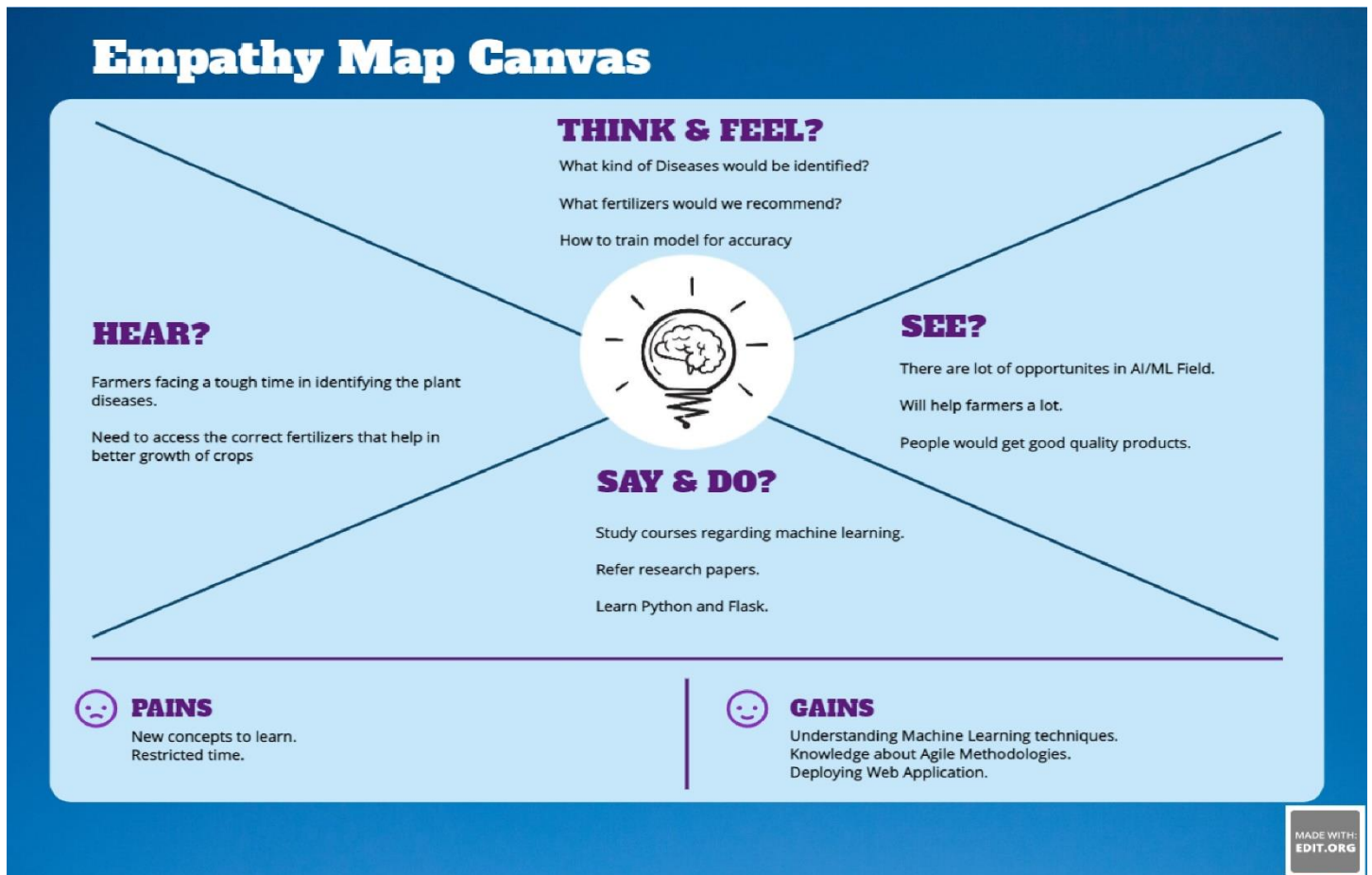
2.4. Problem Statement Definition:

Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality.

Who does the problem affect?	Farmers
What are the boundaries of the problem?	What are the boundaries of the problem?
What is the issue?	In agricultural aspects, if the plant is affected by leaf disease, then it reduces the growth and productiveness. Generally, plant diseases are caused by the abnormal physiological functionalities of plants.
When does the issue occur?	During the development of the crops as they will be affected by various diseases.
Where is the issue occurring?	The issue occurs in agriculture practicing areas, particularly in rural regions
Why is it important that we fix the problem?	It is required for the growth of better-quality food products. It is important to maximize the crop yield.
What solution to solve this issue?	An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant.
What methodology is used to solve the issue?	Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

3. IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map Canvas:



3.3. Proposed Solution:

S.No	Parameter	Description
1.	Problem Statement (problem to be solved)	In recent years, due to the spread of multiple diseases in the farming and the availability of multiple fertilizers in the market, the farmers are getting confused in predicting the diseases and choosing the correct fertilizers for treating them.
2.	Idea / Solution Description	<ul style="list-style-type: none">- When the user inputs an image of a diseased plant leaf, the application predicts the type of disease, displays the result along with a little background about the disease and suggestions to cure it.- Deep learning techniques are used to identify the diseases and suggest appropriate fertilizers that can be taken for those diseases.
3.	Novelty / Uniqueness	<p>The classification technique is divided into the following steps:</p> <ol style="list-style-type: none">1. Image acquisition2. Pre-processing <ul style="list-style-type: none">- Instant solutions for farmers' queries.
4.	Social Impact / Customer satisfaction	<ul style="list-style-type: none">- Fertilizers are a supplementary to nourish and build up the soil fertility in order to support plant nutrients and increase their productivity.- Nowadays, artificial intelligence and sensor technology plays a vital role in the agriculture field. The use of excess insecticides and

		fertilizers in farming poses a risk to human health.
5.	Business Model (Revenue Model)	<ul style="list-style-type: none"> - Helpline support for resolving app related issues. - Service availability depends on the plan subscribed to by the farmers.
6.	Scalability of the solution	<ul style="list-style-type: none"> - Positive is an outcome where the model correctly predicts positive class. - Future diseases that are found and the preventative fertilizers for them can easily be incorporated into the current model, making it highly scalable.

3.4. Problem Solution Fit:

Problem-Solution fit canvas 2.0

FERTILIZER RECOMMENDATION FOR DISEASE PREDICTION

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) Who is your customer? Farmers are the customers to use this application to predict the unknown disease by adding the images to this application.	6. CUSTOMER CONSTRAINTS What constraints prevent your customers from taking action or limit their choices of solutions? We farmers to use this application with the basic knowledge. Customers are asked to check their fertilizers as we recommended with agricultural experts.	5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem or need to get the job done? In the past there is an application but they are not effective as we recommended. But if there is any new unknown diseases the trained model gets confused.	Explore AS, differentiate
	Focus on J&P, tap into BE, understand RC	2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? The must has to image of the plant leaves and upload to the application that the trained model gets process the image and predict the disease.	9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? In the recent days new diseases enters in the farming and becomes unpredictable and there are too many fertilizers in the market. That's the reason that the problem exists	
Identify strong TR & EM		3. TRIGGERS What triggers customers to act? The customers can see efficiency of the application where it can used by neighbors.	10. YOUR SOLUTION If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. It predicts the disease exactly and also to recommend the fertilizer it used to recover from the disease earlier	8. CHANNELS of BEHAVIOUR 8.1 ONLINE What kind of actions do customers take online? The customer add an image to the portal and disease gets predicted, fertilizer recommended.
	4. EMOTIONS: BEFORE / AFTER How do customers feel when they face a problem or a job and afterwards? When there is unknown new disease customer face the problem	8.2 OFFLINE What kind of actions do customers take offline? No actions there in offline that can be implemented in future.		

4. REQUIREMENT ANALYSIS

4.1. Functional Requirements:

Following are the functional requirements of the proposed solution.

Fr.no	Functional requirement	Sub requirement (story/subtask)
Fr-1	User Registration	Registration through form Registration through Gmail
Fr-2	User confirmation	Confirmation via OTP Confirmation via Email
Fr-3	Capturing image	Capture the image of the leaf and check the parameter of the captured image.
Fr-4	Image processing	Upload the image for the prediction of the disease in the leaf.
Fr-5	Leaf identification	Identify the leaf and predict the disease in the leaf.
Fr-6	Image Description	Suggesting the best fertilizer for the disease

4.2. Non – Functional Requirements:

Following are the non-functional requirements of the proposed solution.

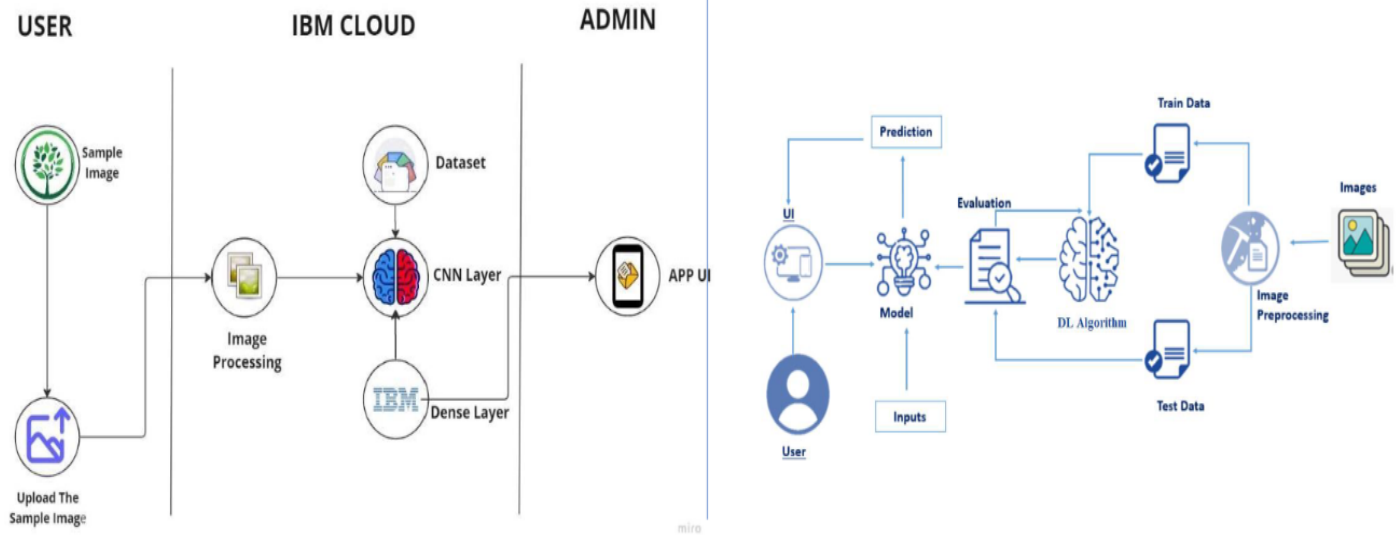
NFr.no	Non-functional requirement	Description
Nfr-1	Usability	Datasets of all the leaves is used to detect the disease that is present in the leaf
Nfr-2	Security	The information belongs to the user and leaves are highly secured.
Nfr-3	Reliability	The leaf image quality is important for predicting disease in the leaf.
Nfr-4	Performance	The performance is based on the quality of the leaf image used for disease prediction.
Nfr-5	Availability	It is available for all the users to predict the disease in the plant.
Nfr-6	Scalability	Increasing the accuracy of prediction of the disease in the leaf.

5. PROJECT DESIGN

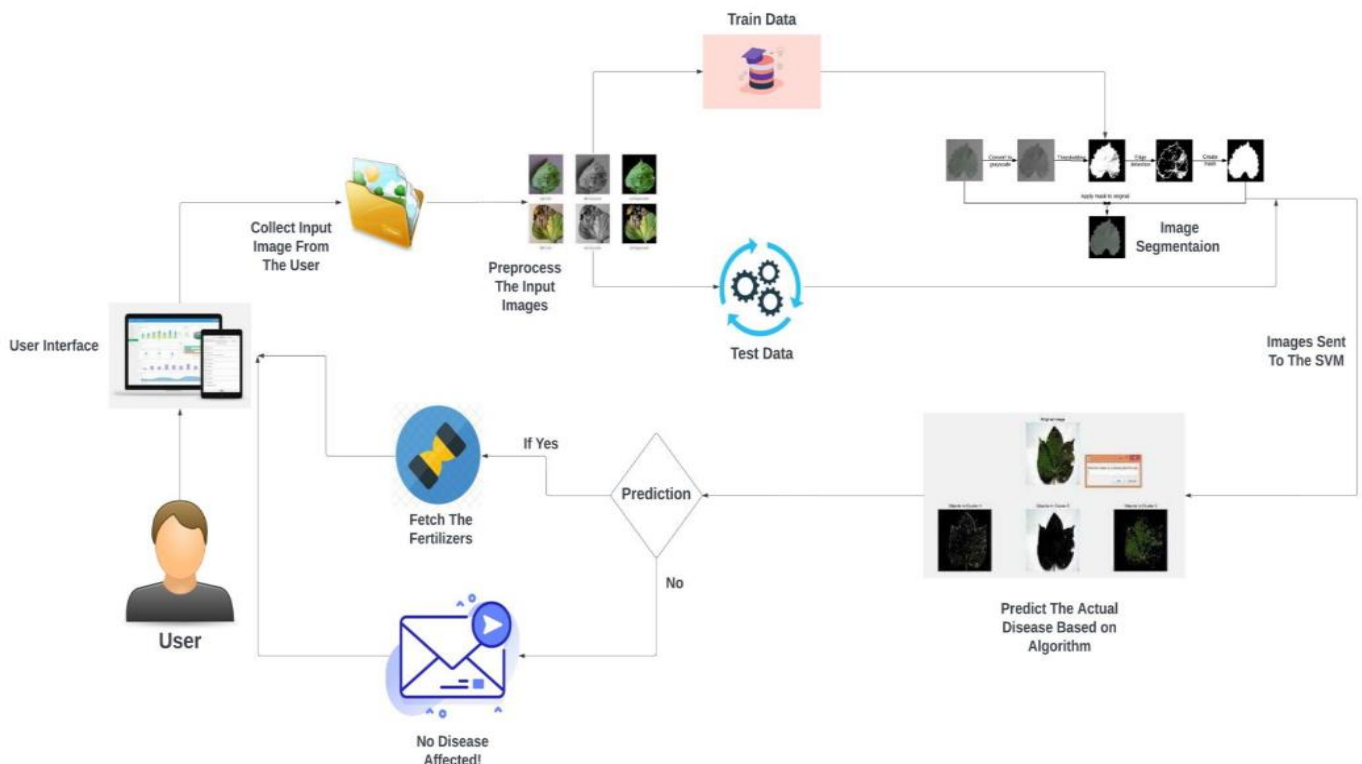
5.1. Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Example: [\(Simplified\)](#)

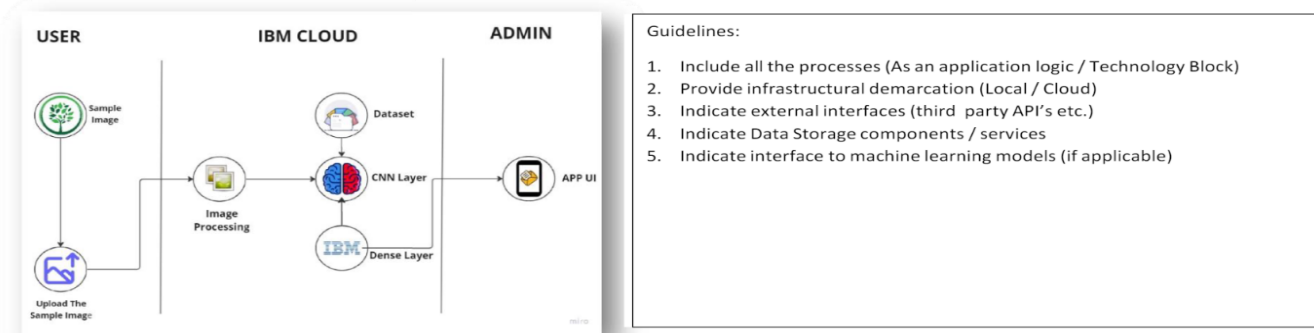


5.2. Solution Architecture:



Technical Architecture:

FERTILIZERS RECOMMENDATION SYSTEM FOR DISEASE PREDICTION:



5.3. User Stories:

The user stories for the Fertilizers recommendation system for disease prediction.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register though my Gmail.	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can log into the application by the verification code I received in my Gmail.	High	Sprint-1
	Dashboard	USN-6	As a user, I can use my dashboard in my web UI, App UI, etc.	I can access my dashboard in my application user interface.	High	Sprint-1
Customer (Web user)	Web UI	USN-7	As a user, I can register and access my dashboard in my Web UI.	I can register and access my dashboard using my Gmail id and password.	Medium	Sprint-1
Customer Care Executive	Application UI	USN-8	As a user, I clear my doubts by consulting the customer care executive.	I can interact with my customer care executive using my App UI.	High	Sprint-1
Administrator	Problem definition	USN-9	As a user, I upload the affected sample image using the application option.	I can upload the image sample using App UI & Web UI.	High	Sprint-1

6. PROJECT PLANNING AND SCHEDULING

6.1. Sprint Planning and Estimation:

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points (Total)	Priority	Team Members
Sprint-1	Dataset Collection	USN1	Create Train and Test folders with each folder having subfolders with leaf images of different plant diseases.	5	High	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K
	Image Preprocessing	USN2	To preprocess the images and then feed them on to the model for training.	5	High	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K

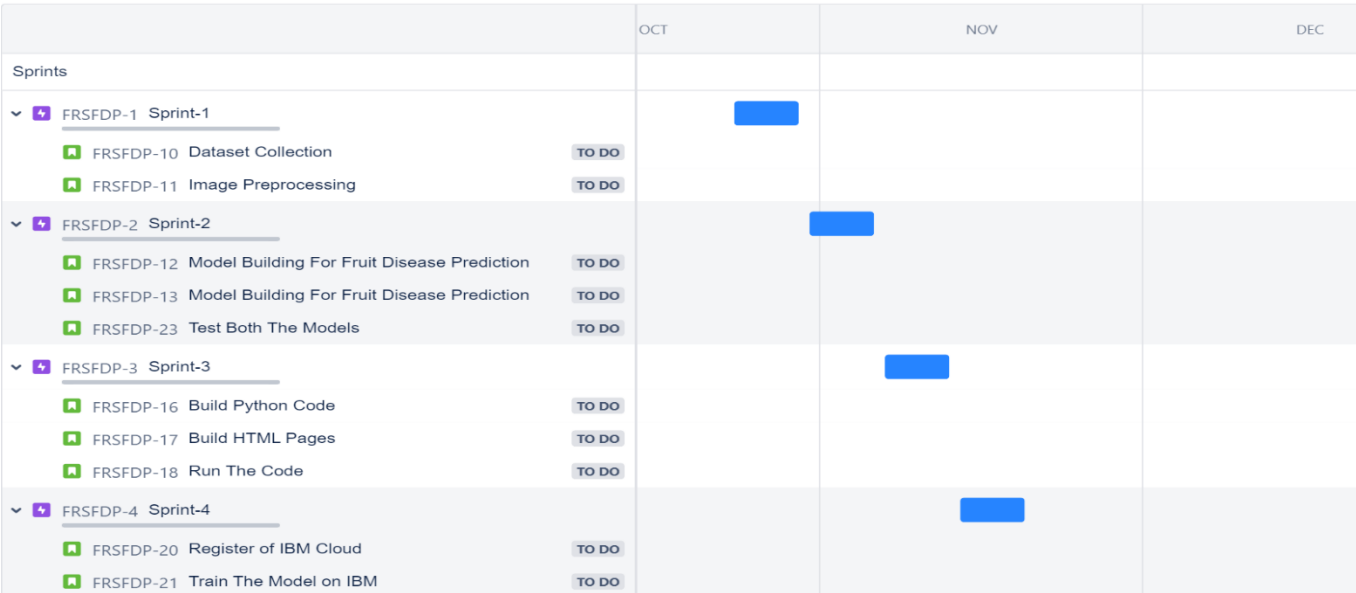
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points (Total)	Priority	Team Members
Sprint-2	Model Creation and Training (Vegetables)	USN-3	Create a model which can classify diseased vegetable plants from given images	6	High	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K
	Model Creation and Training (Fruits)	USN-4	Create a model which can classify diseased fruits plants from given images	6	High	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K
	Test the Model	USN-5	To test both the models by loading the saved models	3	Medium	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K
Sprint-3	Build the python code	USN-6	To Build a flask application	6	High	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K
	Build the HTML pages	USN-7	To Build the UI where a home page will have details about the application, a prediction page where a user is allowed to browse an image and get the predictions.	6	High	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K
	Run the Code	USN-8	To run the Application	3	Medium	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K
Sprint-4	Register for IBM Cloud	USN-6	To Register For IBM Cloud	2	Low	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K
	Train the model on IBM	USN-7	To train the model on IBM and integrate it with the flask Application.	10	High	Shakeel Mohammed G, Thanish Malai P, Jayaraj D, Saranraj A K

6.2. Sprint Delivery Schedule:

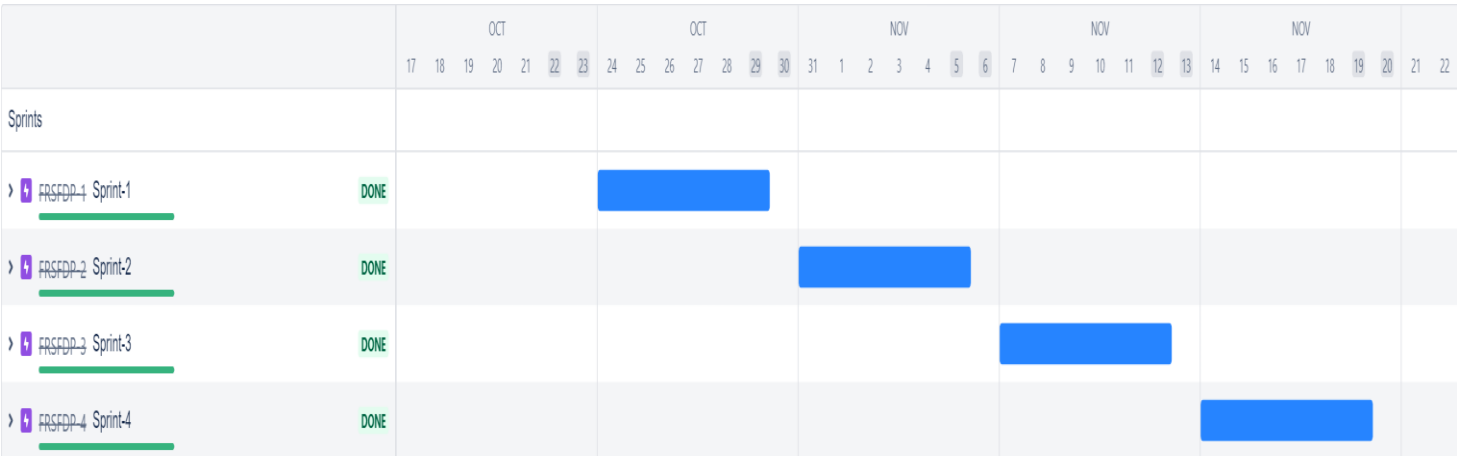
Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10	30 Oct 2022
Sprint-2	15	6 Days	31 Oct 2022	05 Nov 2022	15	06 Nov 2022
Sprint-3	15	6 Days	07 Nov 2022	12 Nov 2022	15	13 Nov 2022
Sprint-4	12	6 Days	14 Nov 2022	19 Nov 2022	10	20 Nov 2022

6.3. Reports from JIRA:



Roadmap After Sprint Delivery:



7. CODING & SOLUTIONING

7.1. Feature 1 [Model Building]

1. Import the Libraries:

Import the libraries that are required to initialize the neural network layer and create and add different layers to the neural network model.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
```

2. Initializing the Model:

Keras has 2 ways to define a neural network:

- Sequential
- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method. Now, we will initialize our model. Initialize the neural network layer by creating a reference/object to the Sequential class.

```
model=Sequential()
```

3. ADD CNN Layers:

We will be adding three layers for CNN

- Convolution layer
- Pooling layer
- Flattening layer

Add Convolution Layer:

The first layer of the neural network model, the convolution layer, will be added. To create a convolution layer, Convolution2D class is used. It takes a number of feature detectors, feature detector size, expected input shape of the image, and activation function as arguments. This

layer applies feature detectors on the input image and returns a feature map (features from the image).

Activation Function: These are the functions that help us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.

```
model.add(Convolution2D(32,(3,3),input_shape = (128,128,3),activation = 'relu'))
```

Add the pooling layer:

Max Pooling selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

After the convolution layer, a pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. The efficient size of the pooling matrix is (2,2). It returns the pooled feature maps.

```
model.add(MaxPooling2D(pool_size = (2,2)))
```

Add the flatten layer:

The flatten layer is used to convert n-dimensional arrays to 1-dimensional arrays. This 1D array will be given as input to ANN layers.

```
model.add(Flatten())
```

4. Add Dense Layers:

Now, let's add Dense Layers to know more about dense layers click below.

Dense layers: The name suggests that layers are fully connected (dense) by the neurons in a network layer. Each neuron in a layer receives input from all the neurons present in the previous layer. Dense is used to add the layers.

Adding Hidden layers:

This step is to add a dense layer (hidden layer). We flatten the feature map and convert it into a vector or single dimensional array in the Flatten layer. This vector array is fed as an input to the neural network and applies an activation function, such as sigmoid or other, and returns the output.

- init is the weight initialization; function which sets all the weights and biases of a network to values suitable as a starting point for training.
- units/ output_dim, which denote is the number of neurons in the hidden layer.

- The activation function basically decides to deactivate neurons or activate them to get the desired output. It also performs a nonlinear transformation on the input to get better results on a complex neural network.
- You can add many hidden layers, in our project we are added two hidden layers. The 1st hidden layer with 40 neurons and 2nd hidden layer with 20neurons.

Adding the output layer:

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function, and weight initializer as the arguments. We use the add () method to add dense layers. the output dimensions here is 6.

```
model.add(Dense(output_dim = 40 ,init = 'uniform',activation = 'relu'))
model.add(Dense(output_dim = 20 ,init = 'random_uniform',activation = 'relu'))
model.add(Dense(output_dim = 6,activation = 'softmax',init = 'random_uniform'))
```

5. Train and Save The Model:

Compile the model:

After adding all the required layers, the model is to be compiled. For this step, loss function, optimizer and metrics for evaluation can be passed as arguments.

```
model.compile(loss = 'categorical_crossentropy',optimizer = "adam",metrics = ["accuracy"])
```

Fit and save the model:

Fit the neural network model with the train and test set, number of epochs and validation steps. Steps per epoch is determined by number of training images/ batch size, for validation steps number of validation images/ batch size.

```
model.fit_generator(x_train, steps_per_epoch = 168,epochs = 3,validation_data = x_test,validation_steps = 52)
```

Accuracy, Loss: Loss value implies how poorly or well a model behaves after each iteration of optimization. An accuracy metric is used to measure the algorithm's performance in a 2 1 interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage.

The weights are to be saved for future use. The weights are saved in as .h5 file using save ().

```
model.save("fruit.h5")
```

model.summary() can be used to see all parameters and shapes in each layer in our models.

6. Test the Model:

The model is to be tested with different images to know if it is working correctly. Import the packages and load the saved model Import the required libraries.


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

Initially, we will be loading the fruit model. You can test it with the vegetable model in a similar way.

```
model = load_model("fruit.h5")
```

Load the test image, pre-process it and predict.

Pre-processing the image includes converting the image to array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.

```
img = image.load_img('apple_healthy.JPG',target_size = (128,128))
```

```
x = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
```

```
pred = model.predict_classes(x)
```

```
pred
```

7.2. Feature 2 [Python Code]:

Build Python Code:

After the model is built, we will be integrating it into a web application so that normal users can also use it. The user needs to browse the images to detect the disease.

Activity 1: Build a flask application

Step 1: Load the required packages

```
1  import requests
2  import os
3  from keras.preprocessing import image
4  from keras.models import load_model
5  from keras.utils import load_img,array_to_img
6  import numpy as np
7  import pandas as pd
8  import tensorflow as tf
9  from flask import Flask, request, render_template,redirect,url_for
10 import os
11 from werkzeug.utils import secure_filename
12 from tensorflow.python.keras.backend import set_session
```

Step 2:

Initialize the flask app and load the model.

An instance of Flask is created, and the model is loaded using load_model from Keras.

```
app=Flask(__name__)
model=load_model(r'C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Sprint-2\Model Building\vegdata.h5')
model1=load_model(r'C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Sprint-2\Model Building\fruitdata.h5')
```

Step 3: Configure the home page

```
@app.route('/', methods=['GET', 'POST'])
def home():
    return render_template('home.html')
```

Step 4: Pre-process the frame and run.

Pre-process the captured frame and give it to the model for prediction. Based on the prediction the output text is generated and sent to the HTML to display. We will be loading the precautions for fruits and vegetables into an excel file to get the precautions based on the output and return it to the HTML Page.

```
@app.route('/prediction')
def prediction():
    return render_template('predict.html')
@app.route('/predict',methods=['POST','GET'])

def predict():
    if(request.method=='POST'):
        f=request.files['image']
        basepath=os.path.dirname(__file__)
        file_path=os.path.join(basepath,'',secure_filename(f.filename))
        #print(file_path)
        f.save(file_path)
        img=tf.keras.utils.load_img(file_path,target_size=(128,128))
        x=tf.keras.utils.img_to_array(img)
        x=np.expand_dims(x,axis=0)
        plant=request.form['plant']
        print(plant)
        if(plant=='Vegetable'):
            preds=model.predict(x)
            print(preds)
            df=pd.read_excel(r'C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Sprint-3\precautions_veg.xlsx')
            #print(df.iloc[preds[0]]['caution'])
        else:
            preds=model1.predict(x)
            print("name=",preds[0])
            df=pd.read_excel(r'C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Sprint-3\precautions_fruits.xlsx')
            #print(df.iloc[preds[0]]['caution'])

    return render_template("predicted.html",disease=str(df.iloc[np.where(preds[0]==1)[0][0]]['disease']),data=str(df.iloc[np.where(preds[0]==1)[0][0]]['caution']))
```

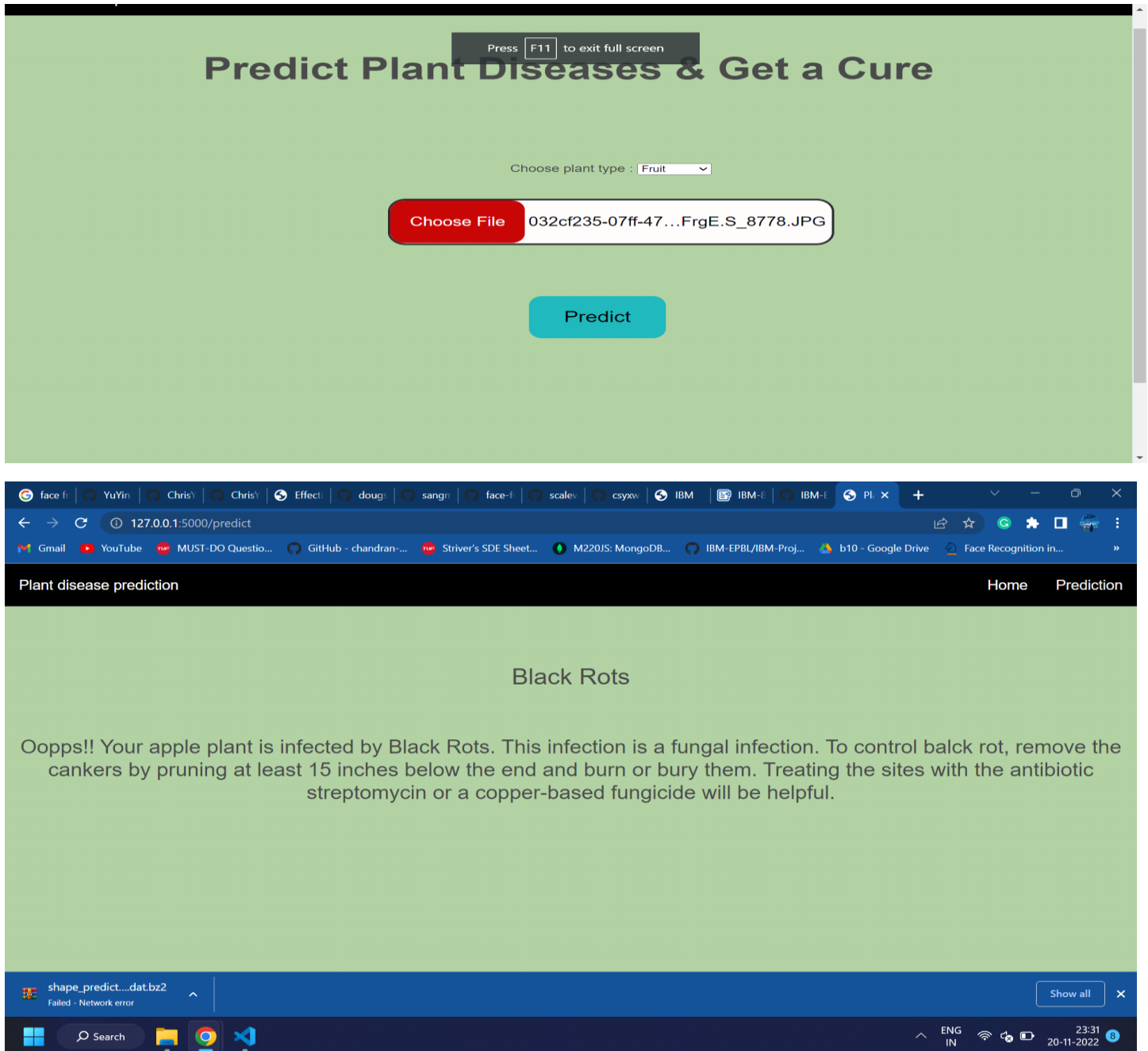
Run the flask application using the run method. By default, the flask runs on 5000 ports. If the port is to be changed, an argument can be passed, and the port can be modified.

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

8. TESTING

8.1. Test Cases:

Test Case 1:



Test Case 2:

The screenshot shows a web application interface for predicting plant diseases. The main heading is "Predict Plant Diseases & Get a Cure". Below this, there is a dropdown menu for "Choose plant type" with "Vegetable" selected. A file upload section includes a red "Choose File" button and a text box displaying the filename "00f2e69a-1e56-4...B.Spot_3132.JPG". A teal "Predict" button is positioned below the file input. The bottom of the interface features a black navigation bar with "Plant disease prediction" on the left and "Home" and "Prediction" on the right. The main content area, which has a light green background, displays the prediction result: "Septoria leaf spots". Below this, a message states: "Oops!! Your tomato plant is infected by Septoria leaf spot. Removing the infected leaves immediately will curb the spread of infection. Organic and chemical fungicides with chlorothalonil are effective in treatment."

Predict Plant Diseases & Get a Cure

Choose plant type : Vegetable ▼

Choose File 00f2e69a-1e56-4...B.Spot_3132.JPG

Predict

Plant disease prediction Home Prediction

Septoria leaf spots

Oops!! Your tomato plant is infected by Septoria leaf spot. Removing the infected leaves immediately will curb the spread of infection. Organic and chemical fungicides with chlorothalonil are effective in treatment.

8.2. User Acceptance Testing:

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Fertilizers Recommendation System for Disease Prediction project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	1	0	1
Duplicate	1	3	2	2	8
External	2	3	0	0	5
Fixed	4	4	4	4	16
Not Reproduced	0	0	0	1	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	7	10	7	7	31

3. Test Case Analysis



This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	1	0	0	1
Client Application	1	0	0	1
Security	1	0	0	1
Outsource Shipping	1	0	0	1
Exception Reporting	1	0	0	1
Final Report Output	1	0	0	1
Version Control	1	0	0	1

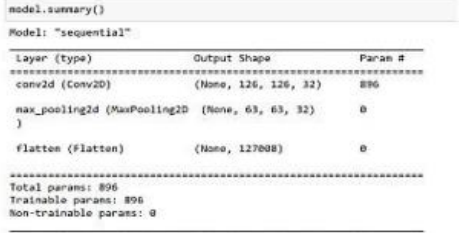



9. RESULTS

9.1. Performance Metrics:

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Total params: 896 Trainable params: 896 Non-trainable params: 0	
2.	Accuracy	Training Accuracy – 96.55 Validation Accuracy – 97.45	

Model Summary:

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
flatten (Flatten)	(None, 127008)	0
Total params: 896		
Trainable params: 896		
Non-trainable params: 0		

Accuracy:

```
Epoch 1/10
225/225 [=====] - 96s 425ms/step - loss: 1.1095 - accuracy: 0.7829 - val_loss: 0.3157 - val_accuracy: 0.8861
Epoch 2/10
225/225 [=====] - 88s 393ms/step - loss: 0.2825 - accuracy: 0.9042 - val_loss: 0.3015 - val_accuracy: 0.9075
Epoch 3/10
225/225 [=====] - 85s 375ms/step - loss: 0.2032 - accuracy: 0.9303 - val_loss: 0.2203 - val_accuracy: 0.9288
Epoch 4/10
225/225 [=====] - 84s 374ms/step - loss: 0.1576 - accuracy: 0.9463 - val_loss: 0.2424 - val_accuracy: 0.9164
Epoch 5/10
225/225 [=====] - 84s 372ms/step - loss: 0.1719 - accuracy: 0.9389 - val_loss: 0.1330 - val_accuracy: 0.9632
Epoch 6/10
225/225 [=====] - 85s 376ms/step - loss: 0.1240 - accuracy: 0.9580 - val_loss: 0.1340 - val_accuracy: 0.9573
Epoch 7/10
225/225 [=====] - 87s 388ms/step - loss: 0.1235 - accuracy: 0.9591 - val_loss: 0.1638 - val_accuracy: 0.9478
Epoch 8/10
225/225 [=====] - 83s 371ms/step - loss: 0.1012 - accuracy: 0.9643 - val_loss: 0.1468 - val_accuracy: 0.9561
Epoch 9/10
225/225 [=====] - 83s 367ms/step - loss: 0.0967 - accuracy: 0.9655 - val_loss: 0.1412 - val_accuracy: 0.9531
Epoch 10/10
225/225 [=====] - 83s 369ms/step - loss: 0.0954 - accuracy: 0.9655 - val_loss: 0.0905 - val_accuracy: 0.9745
```

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- The proposed model could predict the disease just from the image of a particular plant.
- Easy to use UI.
- The model has some good accuracy in detecting the plant just by taking the input(leaf).
- These kinds of web applications can be used in the agricultural sector as well as for small household plants as well.

Disadvantages:

Prediction is limited to a few plants as we haven't trained all the plants.

11. CONCLUSION

The core strategy of this project is to predict the crop based on the soil nutrient content and the location where the crop is growing. This system will help the farmers to choose the right crop for their land and to give a suitable amount of fertilizer to produce the maximum yield. The Support Vector Machine algorithm helps to predict the crop precisely based on the pre-processed crop data. This system will also help the newcomers to choose the crop which will grow in their area and produce a good profit. A decent amount of profit will attract more people towards agriculture.

12. FUTURE SCOPE

- As of now we have just built the web application which apparently takes the input as an image and then predicts the output. Soon we can develop an application which uses computer vision and AI techniques to predict the infection once you keep the camera near the plant or leaf. This could make our project even more usable.
- Further research is implemented by using the proposed algorithm with the existing public datasets. Also, various segmentation algorithms can be implemented to improve accuracy. The proposed algorithm can be modified further to identify the disease that affects the various plant organs such as vegetables and fruits.

13. APPENDIX

Dataset Training:

Veg Dataset:

```
In [ ]: import tensorflow
```

```
In [ ]: import os
```

```
In [ ]: from keras.preprocessing.image import ImageDataGenerator
```

```
In [ ]: train_datagen=ImageDataGenerator(rescale=1./255, zoom_range=0.2, horizontal_flip=True, vertical_flip=False)
```

```
In [ ]: test_datagen=ImageDataGenerator(rescale=1./255)
```

```
In [ ]: x_train=train_datagen.flow_from_directory(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\Veg-dataset\Veg-dataset\train_set", target_size=(224, 224), class_mode='categorical', batch_size=24)
```

Found 11386 images belonging to 9 classes.

```
In [ ]: x_test=test_datagen.flow_from_directory(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\Veg-dataset\Veg-dataset\test_set", target_size=(224, 224), class_mode='categorical', batch_size=24)
```

Found 3416 images belonging to 9 classes.

Fruit Dataset:

```
In [ ]: import tensorflow
```

```
In [ ]: import os
```

```
In [ ]: from keras.preprocessing.image import ImageDataGenerator
```

```
In [ ]: train_datagen=ImageDataGenerator(rescale=1./255, zoom_range=0.2, horizontal_flip=True, vertical_flip=False)
```

```
In [ ]: test_datagen=ImageDataGenerator(rescale=1./255)
```

```
In [ ]: x_train=train_datagen.flow_from_directory(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\fruit-dataset\fruit-dataset\train", target_size=(224, 224), class_mode='categorical', batch_size=24)
```

Found 5384 images belonging to 6 classes.

```
In [ ]: x_test=test_datagen.flow_from_directory(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\fruit-dataset\fruit-dataset\test", target_size=(224, 224), class_mode='categorical', batch_size=24)
```

Found 1686 images belonging to 6 classes.

Model Training:

Fruit Data Model:

```
import os
import tensorflow
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Convolution2D, MaxPooling2D, Flatten
from keras.models import load_model
from keras.utils import load_img, img_to_array
from keras.preprocessing import image
```

```
train_datagen=ImageDataGenerator(rescale=1./255, zoom_range=0.2, horizontal_flip=True, vertical_flip=False)
```

```
test_datagen=ImageDataGenerator(rescale=1./255)
```

```
x_train=train_datagen.flow_from_directory(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\fruit-dataset\fruit-dataset\train", target_size=(
    class_mode='categorical', batch_size=24)
```

Found 5384 images belonging to 6 classes.

```
x_test=test_datagen.flow_from_directory(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\fruit-dataset\fruit-dataset\test", target_size=(
    class_mode='categorical', batch_size=24)
```

Found 1686 images belonging to 6 classes.

```
model=Sequential()
```

```
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

```
model.add(Flatten())
```

```
model.summary()
```

```
32*(3*3*3+1)
```

```
model.add(Dense(300,activation='relu'))
```

```
model.add(Dense(150,activation='relu'))
```

```
model.add(Dense(6,activation='softmax'))
```

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

```
len(x_train)
```

225

1238/24

51.583333333333336

```
model.fit(x_train, steps_per_epoch=len(x_train), validation_data=x_test, validation_steps=len(x_test), epochs=10)
```

Epoch 1/10

225/225 [=====] - 80s 350ms/step - loss: 0.6349 - accuracy: 0.8189 - val_loss: 0.2122 - val_accuracy: 0.9259

Epoch 2/10

225/225 [=====] - 67s 297ms/step - loss: 0.2355 - accuracy: 0.9207 - val_loss: 0.1760 - val_accuracy: 0.9389

Epoch 3/10

225/225 [=====] - 66s 294ms/step - loss: 0.1924 - accuracy: 0.9352 - val_loss: 0.1518 - val_accuracy: 0.9531

Epoch 4/10

225/225 [=====] - 65s 289ms/step - loss: 0.1549 - accuracy: 0.9482 - val_loss: 0.1683 - val_accuracy: 0.9448

Epoch 5/10

225/225 [=====] - 67s 298ms/step - loss: 0.1415 - accuracy: 0.9493 - val_loss: 0.1472 - val_accuracy: 0.9543

Epoch 6/10

225/225 [=====] - 70s 311ms/step - loss: 0.1190 - accuracy: 0.9580 - val_loss: 0.1223 - val_accuracy: 0.9632

Epoch 7/10

225/225 [=====] - 71s 314ms/step - loss: 0.1052 - accuracy: 0.9629 - val_loss: 0.1258 - val_accuracy: 0.9644

Epoch 8/10

225/225 [=====] - 75s 333ms/step - loss: 0.1233 - accuracy: 0.9614 - val_loss: 0.1368 - val_accuracy: 0.9632

Epoch 9/10

225/225 [=====] - 82s 364ms/step - loss: 0.0820 - accuracy: 0.9708 - val_loss: 0.0998 - val_accuracy: 0.9703

Epoch 10/10

225/225 [=====] - 73s 326ms/step - loss: 0.0988 - accuracy: 0.9658 - val_loss: 0.1371 - val_accuracy: 0.9656

<keras.callbacks.History at 0x22039ef13f0>

```

model.save('fruitdata.h5')

model=load_model('fruitdata.h5')

img=tensorflow.keras.utils.load_img(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\fruit-dataset\fruit-dataset\test\Corn_(maize)___Nor
img

x=tensorflow.keras.utils.img_to_array(img)

x

array([[126., 139., 95.],
       [126., 142., 103.],
       [ 95., 115., 80.],
       ...,
       [ 77., 72., 78.],
       [ 78., 73., 79.],
       [ 78., 73., 80.]],

       [[145., 158., 114.],
       [128., 144., 105.],
       [ 94., 116., 80.],
       ...,
       [ 79., 74., 80.],
       [ 79., 74., 80.],
       [ 79., 74., 81.]],

       [[160., 176., 131.],

```

```
x=np.expand_dims(x,axis=0)
```

```
y=np.argmax(model.predict(x),axis=1)
```

```
1/1 [=====] - 0s 112ms/step
```

```
y
```

```
array([3], dtype=int64)
```

```
x_train.class_indices
```

```
{'Apple___Black_rot': 0,
 'Apple___healthy': 1,
 'Corn_(maize)___Northern_Leaf_Blight': 2,
 'Corn_(maize)___healthy': 3,
 'Peach___Bacterial_spot': 4,
 'Peach___healthy': 5}
```

```
index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn_(maize)___healthy','Peach___Bacterial_spot','Peach___healthy']
```

```
index[y[0]]
```

```
'Corn_(maize)___healthy'
```

Vegetable Data Model:

```

import os
import tensorflow
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten
from keras.models import load_model
from keras.utils import load_img, img_to_array

train_datagen=ImageDataGenerator(rescale=1./255, zoom_range=0.2, horizontal_flip=True, vertical_flip=False)

test_datagen=ImageDataGenerator(rescale=1./255)

x_train=train_datagen.flow_from_directory(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\Veg-dataset\Veg-dataset\train_set",target_size=(
    class_mode='categorical',batch_size=24)

Found 11386 images belonging to 9 classes.

x_test=test_datagen.flow_from_directory(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\Veg-dataset\Veg-dataset\test_set",target_size=(
    class_mode='categorical',batch_size=24)

Found 3416 images belonging to 9 classes.

model=Sequential()

model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.summary()

```

```
32*(3*3*3+1)
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
```

```
model.add(Dense(9,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
len(x_train)
```

475

1238/24

51.583333333333336

```
model.fit(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
```

```
Epoch 1/10
475/475 [=====] - 161s 337ms/step - loss: 1.2189 - accuracy: 0.6350 - val_loss: 0.5418 - val_accuracy: 0.8062
Epoch 2/10
475/475 [=====] - 151s 317ms/step - loss: 0.5382 - accuracy: 0.8120 - val_loss: 0.3462 - val_accuracy: 0.8879
Epoch 3/10
475/475 [=====] - 158s 333ms/step - loss: 0.4042 - accuracy: 0.8557 - val_loss: 0.3197 - val_accuracy: 0.8888
Epoch 4/10
475/475 [=====] - 153s 321ms/step - loss: 0.3498 - accuracy: 0.8816 - val_loss: 0.3323 - val_accuracy: 0.8826
Epoch 5/10
475/475 [=====] - 139s 292ms/step - loss: 0.3001 - accuracy: 0.8957 - val_loss: 0.3153 - val_accuracy: 0.8902
Epoch 6/10
475/475 [=====] - 134s 282ms/step - loss: 0.2564 - accuracy: 0.9126 - val_loss: 0.1950 - val_accuracy: 0.9356
Epoch 7/10
475/475 [=====] - 133s 281ms/step - loss: 0.2468 - accuracy: 0.9134 - val_loss: 0.3115 - val_accuracy: 0.8937
Epoch 8/10
475/475 [=====] - 133s 280ms/step - loss: 0.2240 - accuracy: 0.9213 - val_loss: 0.1796 - val_accuracy: 0.9374
Epoch 9/10
475/475 [=====] - 144s 303ms/step - loss: 0.1972 - accuracy: 0.9311 - val_loss: 0.1617 - val_accuracy: 0.9432
Epoch 10/10
475/475 [=====] - 140s 295ms/step - loss: 0.2049 - accuracy: 0.9287 - val_loss: 0.1365 - val_accuracy: 0.9508
<keras.callbacks.History at 0x2043e9b2d70>
```

```
model.save('vegdata.h5')
```

```
model=load_model('vegdata.h5')
```

```
img=tensorflow.keras.utils.load_img(r"C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Dataset_Plant_Disease\Veg-dataset\Veg-dataset\test_set\Potato___Early_bli
```

img



```
x=tensorflow.keras.utils.img_to_array(img)
```

x

```
array([[191., 189., 200.],
       [189., 187., 198.],
       [189., 187., 198.],
       ...,
       [178., 176., 190.],
       [183., 181., 195.],
       [174., 172., 186.]],

       [[184., 182., 193.],
       [192., 190., 201.],
       ...
       [178., 176., 190.],
       [183., 181., 195.],
       [174., 172., 186.]]])
```

```
x=np.expand_dims(x,axis=0)
```

```
y=np.argmax(model.predict(x),axis=1)
```

1/1 [=====] - 0s 125ms/step

x_train.class_indices

```
{'Pepper_bell___Bacterial_spot': 0,
 'Pepper_bell___healthy': 1,
 'Potato___Early_blight': 2,
 'Potato___Late_blight': 3,
 'Potato___healthy': 4,
 'Tomato___Bacterial_spot': 5,
 'Tomato___Late_blight': 6,
 'Tomato___Leaf_Mold': 7,
 'Tomato___Septoria_leaf_spot': 8}
```

```
index=['Pepper_bell___Bacterial_spot','Pepper_bell___healthy','Potato___Early_blight','Potato___Late_blight','Potato___healthy','Tomato___Bacterial_spot','Tomato___Late_blight','Tomato___Leaf_Mold']
```

```
index[y[0]]
```

'Potato___Early_blight'

App.py:

```
import requests
import os
from keras.preprocessing import image
from keras.models import load_model
from keras.utils import load_img, array_to_img
import numpy as np
import pandas as pd
import tensorflow as tf
from flask import Flask, request, render_template, redirect, url_for
import os
from werkzeug.utils import secure_filename
from tensorflow.python.keras.backend import set_session

app=Flask(__name__)
model=load_model(r'C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Sprint-2\Model Building\vegdata.h5')
model1=load_model(r'C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Sprint-2\Model Building\fruitdata.h5')
#home page
@app.route('/', methods=['GET', 'POST'])
def home():
    return render_template('home.html')

@app.route('/prediction')
def prediction():
    return render_template('predict.html')
@app.route('/predict', methods=['POST', 'GET'])

def predict():
    if(request.method=='POST'):
        f=request.files['image']
        basepath=os.path.dirname(__file__)
        file_path=os.path.join(basepath, '', secure_filename(f.filename))
        #print(file_path)
        f.save(file_path)
        img=tf.keras.utils.load_img(file_path, target_size=(128,128))
        x=tf.keras.utils.img_to_array(img)
        x=np.expand_dims(x, axis=0)
        plant=request.form['plant']
        print(plant)
        if(plant=='Vegetable'):
            preds=model.predict(x)
            print(preds)
            df=pd.read_excel(r'C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Sprint-3\precautions_veg.xlsx')
            #print(df.iloc[preds[0]]['caution'])
        else:
            preds=model1.predict(x)
            print("name=", preds[0])
            df=pd.read_excel(r'C:\Users\thani_k\Downloads\Fertilizers_Recommendation_System_For_Disease_Prediction\Sprint-3\precautions_fruits.xlsx')
            #print(df.iloc[preds[0]]['caution'])

        return render_template("predicted.html", disease=str(df.iloc[np.where(preds[0]==1)[0][0]]['disease']), data=str(df.iloc[np.where(preds[0]==1)[0][0]]['caution']))

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

GitHub Link: <https://github.com/IBM-EPBL/IBM-Project-870-1658327386>

Project Demonstration Video Link:

<https://drive.google.com/file/d/1x7xMVJLmT0johBtAfaNQkRK9y63Pc7Xj/view?usp=sharing>