

# **FERTILIZER RECOMMENDATION SYSTEM FOR DISEASE PREDICTION**

**NALAIYA THIRAN PROJECT REPORT**

**IBM-Project-15673-1659602957**

**TEAM ID : PNT2022TMID08291**

**Submitted by**

GARIGAPATI YAMINI  
BINENA KEERTHANA  
MAHESWARI M  
MADALA SUPRIYA

(810419104025)  
(810419104013)  
(810419104302)  
(810419104058)

*in partial fulfilment for the award of the degree*

**of**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE**

**(AUTONOMOUS)**

**PERAMBALUR-621212**

## **TABLE OF CONTENTS**

1.	<b>INTRODUCTION</b>	01
	1.1 Project Overview	
	1.2 Purpose	
2.	<b>LITERATURE SURVEY</b>	02
	2.1 Existing Problem	
	2.2 References	
	2.3 Problem Statement Definition	
3.	<b>IDEATION &amp; PROPOSED SOLUTION</b>	05
	3.1 Empathy Map Canvas	
	3.2 Ideation & Brainstorming	
	3.3 Proposed Solution	
	3.4 Problem Solution fit	
4.	<b>REQUIREMENT ANALYSIS</b>	08
	4.1 Functional requirements	
	4.2 Non-Functional requirements	
5.	<b>PROJECT DESIGN</b>	09
	5.1 Data Flow Diagrams	
	5.2 Solution & Technical Architecture	
	5.3 User Stories	
6.	<b>PROJECT PLANNING &amp; SCHEDULING</b>	11
	6.1 Sprint Planning & Estimation	
	6.2 Sprint Delivery Schedule	
	6.3 Reports from JIRA	
7.	<b>CODING &amp; SOLUTIONING</b>	13
	7.1 Home Page	
	7.2 Prediction Page	
	7.3 Predicting Disease & Giving Precaution	
8.	<b>TESTING</b>	14
	8.1 Test Cases	
	8.2 User Acceptance Testing	
9.	<b>RESULTS</b>	16
	9.1 Performance Metrics	
10.	<b>ADVANTAGES &amp; DISADVANTAGES</b>	17
11.	<b>CONCLUSION</b>	18
12.	<b>FUTURE SCOPE</b>	19
13.	<b>APPENDIX</b>	20
	Source Code	
	GitHub & Project Demo Link	

# **INTRODUCTION**

## **1.1 PROJECT OVERVIEW:**

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.

## **1.2 PURPOSE:**

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.

## **2. LITERATURE SURVEY**

### **2.1 EXISTING PROBLEM:**

- 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.
- System only able to detect the 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 pre-processed and then does the further work. K-Means Clustering used for image segmentation and then system extract the GLCM features from disease detected images. The disease classification done through the SVM classifier.
- The system uses leaf images taken from an online dataset, so cannot implement in real time. This paper focuses on the detecting and classifying the leaf disease of soybean plant. Using SVM the proposed system classifies the leaf disease in three 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.
- 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 system, they use soil moisture and temperature sensor, and sensor data send to the cloud. The user can also detect the plant leaf disease. K-means clustering used for feature extraction.: 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.
- Due to the changing climatic conditions, accurate results cannot be predicted by this
- 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 realworld field circumstances.
- 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.
- To provide fine-grained segmentations of the diseased portion of the dataset. this is not possible due to lack of such data. However, in our application, we can integrate a segmentation annotation tool where the users might be able to help us with the lack. Also, we can use some unsupervised algorithms to pinpoint the diseased areas in the image. We intend to add these features and fix these gaps in our upcoming work.

## 2.2 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-Plant-Leaves-Disease-detection-using-Image-Processing\\_Techniques.pdf](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-Plant-Leaves-Disease-detection-using-Image-Processing_Techniques.pdf)
- [4] Duan Yan-e, Design of Intelligent Agriculture Management Information System Based on IOT I, 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-For-DiseasePredictionIn-Tree-Leave.pdf>.
- [6] Swapnil Jori<sup>1</sup>, Rutuja Bhalshankar<sup>2</sup>, Dipali Dhamale<sup>3</sup>, 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.

## 2.3 PROBLEM STATEMENT:

Mr. Narasimma Rao is a 65-year-old man. He had an own farming land and do Agriculture for past 30 Years, in this 30 Years he Faced a problem in Choosing Fertilizers and Controlling of Plant Disease.

- Narasimma Rao wants to know the better recommendation for fertilizers for plants with the disease.
- He has faced huge losses for a long time.
- This problem is usually faced by most farmers.
- Mr. Narasimma Rao needs to know the result immediately.

Who does the problem affect?	Persons who do Agriculture
What are the boundaries of the problem?	People who Grow Crops and facing Issues of Plant Disease

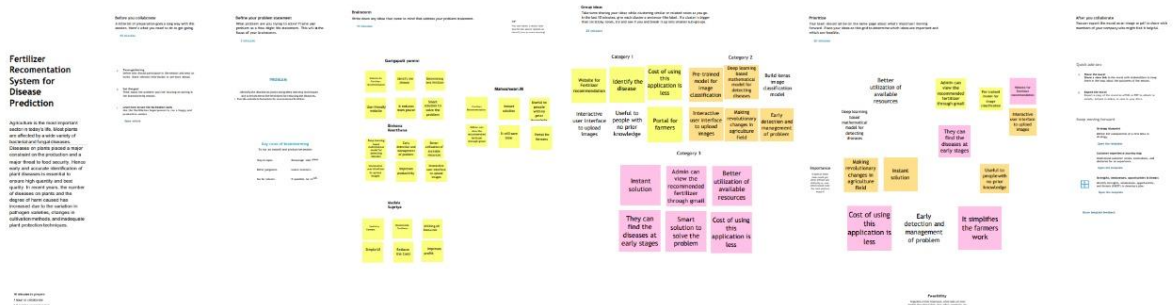
What is the issue?	In agricultural aspects, if the plant is affected by leaf disease, then it reduces the growth and productiveness. The plant diseases are caused by the abnormal physiological functionalities of plants.
When does the issue occur?	During the development of the crops as they will be affected by various diseases.
Where does the issue occur?	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 maximise 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 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 & PROPOSED SOLUTION

#### 3.1 EMPATHY MAP CANVAS:



#### 3.2 IDEATION & BRAINSTORMING:

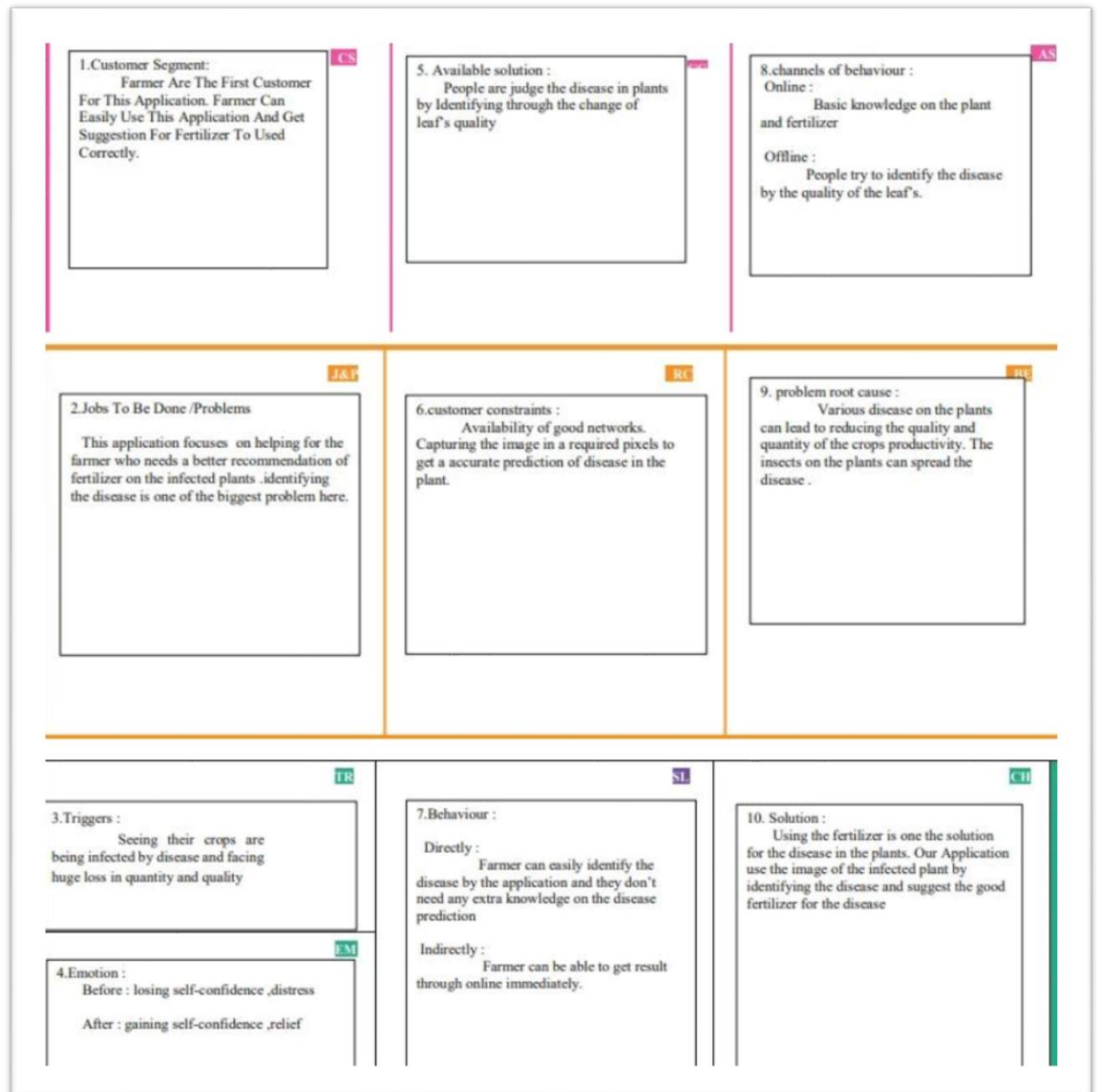


### 3.3 PROPOSED SOLUTION:

S. No	Parameter	Description
1.	Problem statement (problem to be solved)	Disease in plants reduced the quantity and quality of the plant's productivity. Identifying the disease in plant is hard to find.
2.	Idea/solution description	One of the solutions of the problem is to identify the disease in early stage and using the correct fertilizer.
3.	Novelty / uniqueness	This application can suggest good fertilizer for the disease in the plant by recognizing the images.
4.	Social impact/customer satisfaction	It helps the farmer by identifying the disease in the early stage and increase the quality and quantity of crops in efficient way
5.	Business model (revenue model)	The application is recommending to farmer in subscription basis.
6.	Scalability of the solution	This application can be improved by introducing online purchases of crops, fertilizer easily



### 3.4 PROBLEM SOLUTION FIT:



## **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 leaf.
Fr-6	Image description	Suggesting the best fertilizer for the disease.

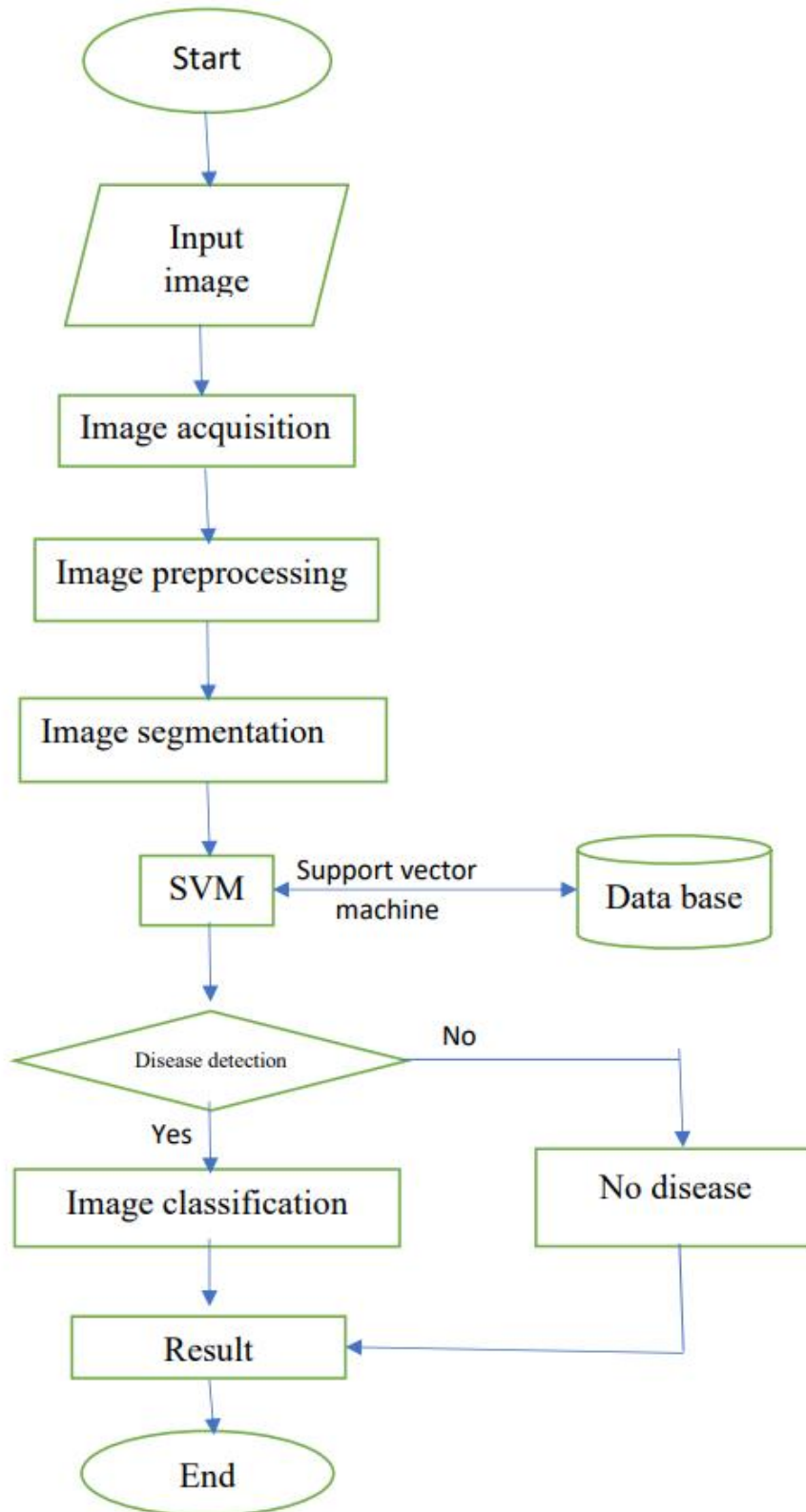
### **4.2 NON-FUNCTIONAL REQUIREMENTS:**

Following is the non-functional requirement of the proposed solution.

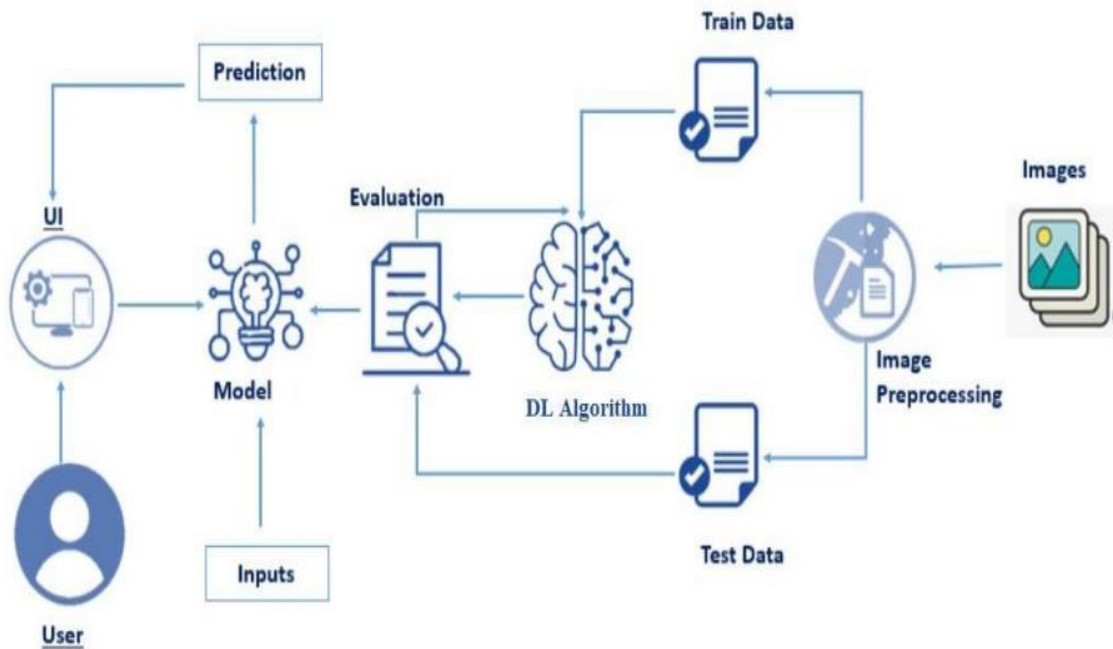
NFr.no	Non-functional requirement	Description
Nfr-1	Usability	Datasets of all the leaf is used to detecting the disease that present in the leaf.
Nfr-2	Security	The information belongs to the user and leaf are secured highly.
Nfr-3	Reliability	The leaf quality is important for the predicting the disease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

## 5. PROJECT DESIGN

### 5.1 DATA FLOW DIAGRAMS:



## 5.2 SOLUTION & TECHNICAL ARCHITECTURE:



## 5.2 USER STORIES

Sprint.	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Image Processing.	USN-1	As a user, I can retrieve useful information about the images.	1	Low	Garigapati yamini Madala supriya Binkena keerthana Maheshwari M
Sprint-2	Model Building for Fruit Disease Prediction	USN-2	As a user, I can be able to predict fruit disease using this model.	1	Medium	Garigapati yamini Madala supriya Binkena keerthana Maheshwari M
Sprint-2	Model Building for Vegetable Disease Prediction.	USN-3	As a user, I can be able to predict vegetable disease using this model.	2	Medium	Garigapati yamini Madala supriya Binkena keerthana Maheshwari M
Sprint-3	Application Building.	USN-4	As a user, I can see a web page for Fertilizers Recommendation System for Disease Prediction	2	High	Garigapati yamini Madala supriya Binkena keerthana maheshwari
Sprint-4	Train The Model on IBM Cloud.	USN-5	As a user, I can save the information about Fertilizers and crops on IBM cloud	2	High	Garigapati yamini Madala supriya Binkena keerthana Maheshwari M

## **6. PROJECT PLANNING & SCHEDULING**

### **6.1 SPRINT PLANNING & ESTIMATION**

<b>Sprint</b>	<b>Total Story Points</b>	<b>Duration</b>	<b>Sprint Start Date</b>	<b>Sprint End Date (Planned)</b>	<b>Story Points Completed (as on Planned End Date)</b>	<b>Sprint Release Date (Actual)</b>
Sprint-1	20	6 Days	27 Oct 2022	01 Nov 2022	20	26 Oct 2022
Sprint-2	20	6 Days	02 Nov 2022	07 Nov 2022	20	30 Oct 2022
Sprint-3	20	6 Days	08 Nov 2022	13 Nov 2022	20	05 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	10 Nov 2022

### **6.2 SPRINT DELIVERY SCHEDULE**

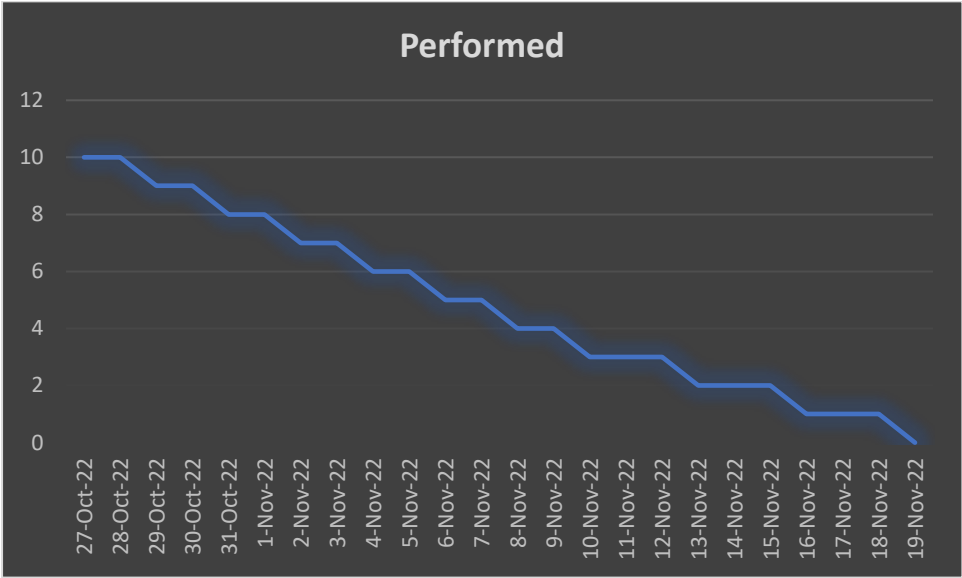
<b>Sprint</b>	<b>Duration</b>	<b>Sprint Start Date</b>	<b>Sprint End Date (Planned)</b>	<b>Sprint Release Date (Actual)</b>
Sprint-1	6 Days	27 Oct 2022	01 Nov 2022	26 Oct 2022
Sprint-2	6 Days	02 Nov 2022	07 Nov 2022	30 Oct 2022
Sprint-3	6 Days	08 Nov 2022	13 Nov 2022	05 Nov 2022
Sprint-4	6 Days	14 Nov 2022	19 Nov 2022	10 Nov 2022

### **6.3 REPORTS FROM JIRA**

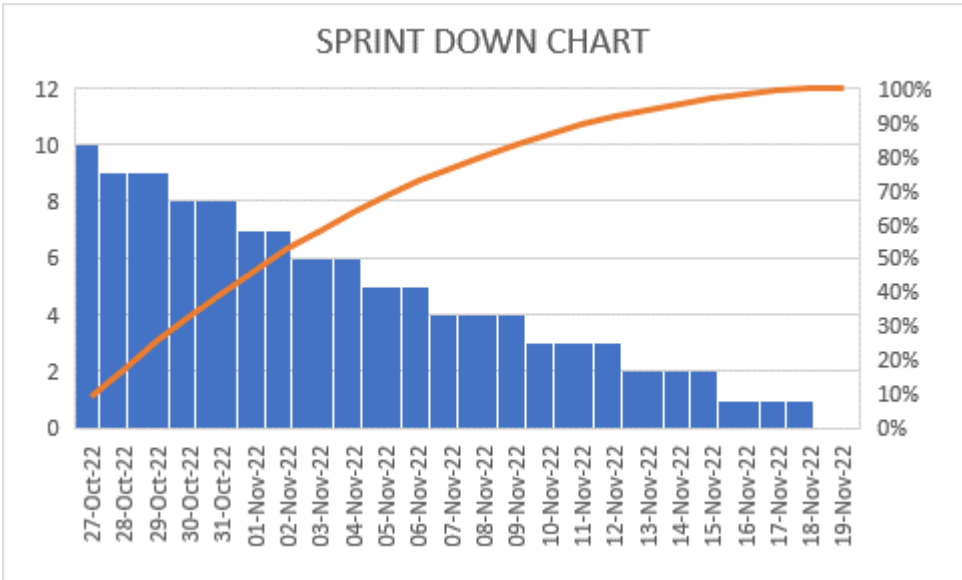
**Velocity:** Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

**BURN DOWN CHART**



**SPRINT DOWN CHART**



## 7. CODING & SOLUTIONING

### 7.1 HOME PAGE

```
#home page
@app.route('/')
def home():
    return render_template('home.html')
```

### 7.2 PREDICTION PAGE

```
#prediction page
@app.route('/predict')
def predict():
    return render_template('predict.html')
```

### 7.3 PREDICTING DISEASE & GIVING PRECAUTIONS

```
@app.route('/predict1',methods=['POST'])
def predict1():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(128, 128))

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

        plant=request.form['plant']
        print(plant)
        if(plant=="vegetable"):
            preds = model.predict1(x)
            preds=np.argmax(preds)
            print(preds)
            df=pd.read_excel('precautions-veg.xlsx')
            print(df.iloc[preds[0]]['caution'])
        else:
            preds = model1.predict1(x)
            preds=np.argmax(preds)
            print(preds)
            df=pd.read_excel('precautions-fruits.xlsx')
            print(df.iloc[preds]['caution'])
```

## **8. TESTING**

### **8.1 TEST CASES**

*A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behaviour of the system is satisfied or not.*

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

S.NO	Scenario	Input	Expected Output	Actual Output
1	Home Page	Predict	Introduction to the predict page	Predict page
2	Predict Page	Selection	Selecting the Leaf images	Leaf images selected successfully
3	Predict Page	Predicting	Predicting leaf healthy and giving precautions	Healthy of a Leaf predicted and given precautions successfully

### **8.2 USER ACCEPTANCE TESTING**

#### **8.2.1 Purpose Of The Document:**

The purpose of this document is to briefly explain the test coverage and open issues of the [Fertilizer Recommendation system for plant disease prediction] project at the time of the release to User Acceptance Testing (UAT).

#### **8.2.2 Defect Analysis:**

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
Leaf spots	109	4	2	3	1
Mosaic leaf pattern	9	6	3	6	24



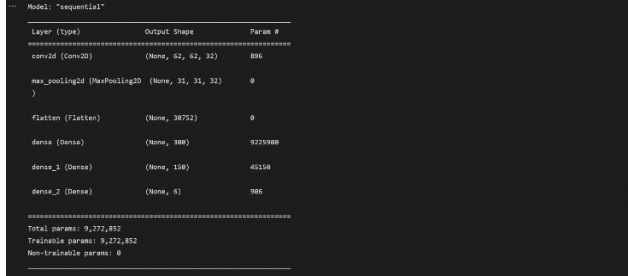
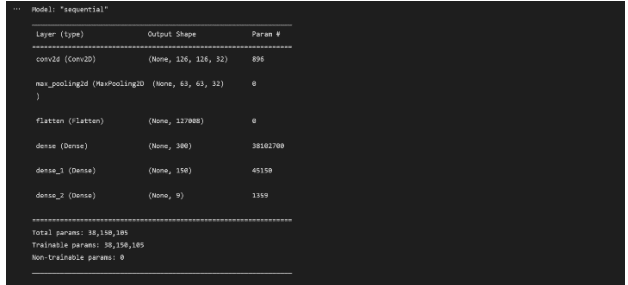
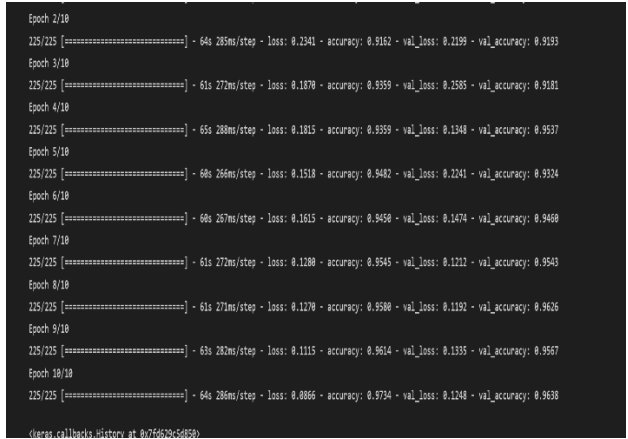
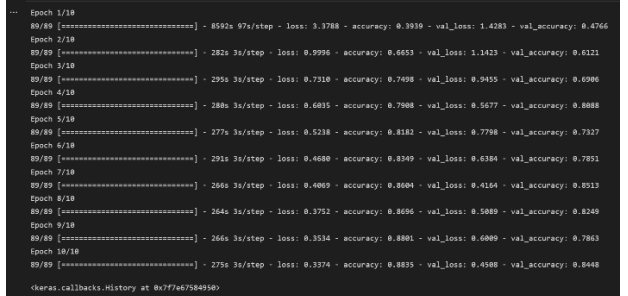
Misshapen leaves	2	7	0	1	10
Yellow leaves	11	4	3	20	38
Fruit rots	3	2	1	0	6
Fruit spots	5	3	1	1	10
Blights	4	5	2	1	12
Totals	44	31	13	32	11

### 8.2.3 Test Case Analysis:

Selection	Test Cases	Not Tested	Fail	Pass
Leaf spots	17	0	0	17
Mosaic leaf pattern	51	0	0	51
Misshapen leaves	20	0	0	20
Yellow leaves	7	0	0	7
Fruit rots	9	0	0	9
Fruit spots	4	0	0	4
Blights	2	0	0	2

## 9. RESULTS

### 9.1 PERFORMANCE METRICS

S. No.	Parameter	Values	Screenshot
1.	Model Summary of Fruit	Training the dataset of Vegetable images by using the CNN models to predict the disease of the given leaves.	 <pre> Model: "sequential" Layer (type) Output Shape Param # ----- conv1d (Conv2D) (None, 62, 62, 32) 896 max_pooling1d (MaxPooling2D) (None, 31, 31, 32) 0 Flatten (Flatten) (None, 30752) 0 dense_1 (Dense) (None, 100) 3075300 dense_2 (Dense) (None, 10) 100 Total params: 9,272,852 Trainable params: 9,272,852 Non-trainable params: 0 </pre>
2.	Model Summary for Vegetable	Training the dataset of Vegetable images by using the CNN models to predict the disease of the given leaves.	 <pre> Model: "sequential" Layer (type) Output Shape Param # ----- conv1d (Conv2D) (None, 126, 126, 32) 896 max_pooling1d (MaxPooling2D) (None, 63, 63, 32) 0 Flatten (Flatten) (None, 127808) 0 dense_1 (Dense) (None, 100) 12780900 dense_2 (Dense) (None, 10) 100 Total params: 13,156,145 Trainable params: 13,156,145 Non-trainable params: 0 </pre>
2.	Accuracy for Fruit	Training Accuracy - 0.9734 Validation Accuracy - 0.9638	 <pre> Epoch 2/20 225/225 [=====] - 64s 285ms/step - loss: 0.2341 - accuracy: 0.9162 - val_loss: 0.2399 - val_accuracy: 0.9133 Epoch 3/20 225/225 [=====] - 61s 272ms/step - loss: 0.1870 - accuracy: 0.9359 - val_loss: 0.2385 - val_accuracy: 0.9181 Epoch 4/20 225/225 [=====] - 65s 288ms/step - loss: 0.1815 - accuracy: 0.9359 - val_loss: 0.1348 - val_accuracy: 0.9537 Epoch 5/20 225/225 [=====] - 68s 266ms/step - loss: 0.1518 - accuracy: 0.9482 - val_loss: 0.1241 - val_accuracy: 0.9324 Epoch 6/20 225/225 [=====] - 68s 267ms/step - loss: 0.1615 - accuracy: 0.9458 - val_loss: 0.1474 - val_accuracy: 0.9460 Epoch 7/20 225/225 [=====] - 61s 272ms/step - loss: 0.1208 - accuracy: 0.9545 - val_loss: 0.1212 - val_accuracy: 0.9543 Epoch 8/20 225/225 [=====] - 61s 271ms/step - loss: 0.1270 - accuracy: 0.9580 - val_loss: 0.1192 - val_accuracy: 0.9626 Epoch 9/20 225/225 [=====] - 63s 282ms/step - loss: 0.1115 - accuracy: 0.9614 - val_loss: 0.1335 - val_accuracy: 0.9557 Epoch 10/20 225/225 [=====] - 64s 286ms/step - loss: 0.0866 - accuracy: 0.9734 - val_loss: 0.1248 - val_accuracy: 0.9638 (keras.callbacks.History at 0x7f65265d8590) </pre>
3.	Accuracy for Vegetable	Training Accuracy - 0.8835 Validation Accuracy - 0.8448	 <pre> Epoch 1/20 89/89 [=====] - 85s2s 97s/step - loss: 3.3788 - accuracy: 0.3939 - val_loss: 1.4283 - val_accuracy: 0.4766 Epoch 2/20 89/89 [=====] - 282s 3s/step - loss: 0.9996 - accuracy: 0.6653 - val_loss: 1.1423 - val_accuracy: 0.6121 Epoch 3/20 89/89 [=====] - 295s 3s/step - loss: 0.7318 - accuracy: 0.7498 - val_loss: 0.9435 - val_accuracy: 0.6086 Epoch 4/20 89/89 [=====] - 288s 3s/step - loss: 0.6835 - accuracy: 0.7888 - val_loss: 0.5677 - val_accuracy: 0.8888 Epoch 5/20 89/89 [=====] - 277s 3s/step - loss: 0.5238 - accuracy: 0.8182 - val_loss: 0.7798 - val_accuracy: 0.7327 Epoch 6/20 89/89 [=====] - 291s 3s/step - loss: 0.4688 - accuracy: 0.8349 - val_loss: 0.6304 - val_accuracy: 0.7851 Epoch 7/20 89/89 [=====] - 268s 3s/step - loss: 0.4809 - accuracy: 0.8008 - val_loss: 0.4184 - val_accuracy: 0.8513 Epoch 8/20 89/89 [=====] - 264s 3s/step - loss: 0.3752 - accuracy: 0.8696 - val_loss: 0.5989 - val_accuracy: 0.8249 Epoch 9/20 89/89 [=====] - 264s 3s/step - loss: 0.3534 - accuracy: 0.8881 - val_loss: 0.6009 - val_accuracy: 0.7863 Epoch 10/20 89/89 [=====] - 275s 3s/step - loss: 0.3374 - accuracy: 0.8835 - val_loss: 0.4588 - val_accuracy: 0.8448 (keras.callbacks.History at 0x7f7e67584950) </pre>

## **10. ADVANTAGES AND DISADVANTAGES**

### **ADVANTAGES:**

- Prediction of disease of a plant using leaf images in early stages.
- Giving the precautions to the plants according to the plant condition.
- Recommends the fertilizers for the particular disease.
- It doesn't required any chemical to test the disease.
- Simple way to predict the disease by taking Images.

### **DISADVANTAGES:**

- It predicts the disease for the plants which are given in the Dataset only.
- It cannot predict the disease for the plants which are not present in the Dataset.
- Sometimes it may predict the wrong fertilizers which leads to loss

## **11. CONCLUSION**

Different approaches and models of Deep Learning methods were explored and used in this project so that it can detect and classify plant diseases correctly through image processing of leaves of the plants. The procedure starts from collecting the images used for training, testing and validation to image pre-processing and augmentation and finally comparison of different pretrained models over their accuracy. Finally, at the end , our model detects and distinguishes between a healthy plant and different diseases and provides suitable remedies so as to cure the disease. This paper proposed and developed a system which uses plant leaf images to detect different types of disease in tomato crops, and also provides appropriate fertilizer suggestions.

## **12. FUTURE SCOPE**

The system successfully interprets various Diseases and is also capable of providing fertilizers suggestion for the respective disease. Furthermore, this system can be made more robust by incorporating more image dataset with wider variations like more than one leaf in a single image. An App could also be developed for the project which could make the work of the farmers easier. They could directly upload image on the app and it would tell the disease and the cure then and there. This would reduce the time and efforts. This project is limited to just one crop for now but in the future more crops and even flowers dataset can be added so that it is helpful for every agricultural need. Newer models can also be added and tried with time which may result in better accuracy and would make the model even faster.

## 13.APPENDIX

### SOURCE CODE:

#### App.py

```
import requests
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
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__)

#load both the vegetable and fruit models
model = load_model("vegetable.h5")
model1=load_model("fruit.h5")

#home page
@app.route('/')
def home():
    return render_template('home.html')

#prediction page
@app.route('/predict')
def predict():
    return render_template('predict.html')

@app.route('/predict1',methods=['POST'])
def predict1():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(128, 128))

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

```

    plant=request.form['plant']
    print(plant)
    if(plant=="vegetable"):
        preds = model.predict1(x)
        preds=np.argmax(preds)
        print(preds)
        df=pd.read_excel('precautions-veg.xlsx')
        print(df.iloc[preds[0]]['caution'])
    else:
        preds = model1.predict1(x)
        preds=np.argmax(preds)
        print(preds)
        df=pd.read_excel('precautions-fruits.xlsx')
        print(df.iloc[preds]['caution'])

    return df.iloc[preds]['caution']

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

```

## home.html

```

<!DOCTYPE html>
<html >

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title> Plant Disease Prediction</title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{url_for('static',filename='css/final.css')}}">
<link href='https://fonts.googleapis.com/css?family=Merriweather'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin+Sans'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>

```

```

<style>
.header {
    top:0;
    margin:0px;
    left: 0px;
    right: 0px;
    position: fixed;
    background-color: #28272c;
    color: white;
    box-shadow: 0px 8px 4px grey;
    overflow: hidden;
    padding-left:20px;
    font-family: 'Josefin Sans';
    font-size: 2vw;
    width: 100%;
    height:8%;
    text-align: center;
}
.topnav {
    overflow: hidden;
    background-color: #333;
}

.topnav-right a {
    float: left;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 18px;
}

.topnav-right a:hover {
    background-color: #ddd;
    color: black;
}

.topnav-right a.active {
    background-color: #565961;
    color: white;
}

.topnav-right {
    float: right;
    padding-right:100px;
}

body {

```



```

background-color:#ffffff;
background-repeat: no-repeat;
background-size:cover;
background-position: 0px 0px;
}
.button {
background-color: #28272c;
border: none;
color: white;
padding: 15px 32px;
text-align: center;
text-decoration: none;
display: inline-block;
font-size: 16px;
border-radius: 12px;
}
.button:hover {
box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

input[type=text], input[type=password] {
width: 100%;
padding: 12px 20px;
display: inline-block;
margin-bottom:18px;
border: 1px solid #ccc;
box-sizing: border-box;
}

button {
background-color: #28272c;
color: white;
padding: 14px 20px;
margin-bottom:8px;
border: none;
cursor: pointer;
width: 15%;
border-radius:4px;
}

button:hover {
opacity: 0.8;
}

.cancelbtn {
width: auto;

```

```

padding: 10px 18px;
background-color: #f44336;
}

.imgcontainer {
text-align: center;
margin: 24px 0 12px 0;
}

img.avatar {
width: 30%;
border-radius: 50%;
}

.container {
padding: 16px;
}

span.psw {
float: right;
padding-top: 16px;
}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
span.psw {
display: block;
float: none;
}
.cancelbtn {
width: 100%;
}
}

.home{
margin:80px;

width: 84%;
height: 500px;
padding-top:10px;
padding-left: 30px;
}

.login{
margin:80px;
box-sizing: content-box;
width: 84%;
height: 420px;

```

```

padding: 30px;
border: 10px solid blue;
}
.left,.right{
box-sizing: content-box;
height: 400px;
margin:20px;
border: 10px solid blue;
}

.mySlides {display: none;}
img {vertical-align: middle;}

/* Slideshow container */
.slideshow-container {
max-width: 1000px;
position: relative;
margin: auto;
}

/* Caption text */
.text {
color: #f2f2f2;
font-size: 15px;
padding: 8px 12px;
position: absolute;
bottom: 8px;
width: 100%;
text-align: center;
}

/* The dots/bullets/indicators */
.dot {
height: 15px;
width: 15px;
margin: 0 2px;
background-color: #bbb;
border-radius: 50%;
display: inline-block;
transition: background-color 0.6s ease;
}

.active {
background-color: #717171;
}

/* Fading animation */
.fade {
-webkit-animation-name: fade;

```

```

    -webkit-animation-duration: 1.5s;
    animation-name: fade;
    animation-duration: 1.5s;
}

@-webkit-keyframes fade {
    from {opacity: .4}
    to {opacity: 1}
}

@keyframes fade {
    from {opacity: .4}
    to {opacity: 1}
}

/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
    .text {font-size: 11px}
}
</style>
</head>

<body style="font-family:'Times New Roman', Times, serif;background-
color:#C2C5A8;">

<div class="header">
    <div style="width:50%;float:left;font-size:2vw;text-align:left;color:white;
padding-top:1%">Plant Disease Prediction</div>
    <div class="topnav-right"style="padding-top:0.5%;">

        <a class="active" href="{{url_for ('home')}}">Home</a>
        <a href="{{url_for ('predict')}}">Predict</a>
    </div>
</div>

<div style="background-color:#ffffff;">
<div style="width:60%;float:left;">
<div style="font-size:50px;font-family:Montserrat;padding-left:20px;text-
align:center;padding-top:10%;">
<b>Detect if your plant<br> is infected!!</b></div><br>
<div style="font-size:20px;font-family:Montserrat;padding-left:70px;padding-
right:30px;text-align:justify;">Agriculture is one of the major sectors worlds
wide. Over the years it has developed and the use of new technologies and
equipment replaced almost all the traditional methods of farming. The plant
diseases effect the production. Identification of diseases and taking
necessary precautions is all done through naked eye, which requires labour and
laboratries. This application helps farmers in detecting the diseases by

```

```

observing the spots on the leaves, which inturn saves effort and labor
costs.</div><br><br>
</div>
</div>
<div style="width:40%;float:right;"><br><br>


</div>
</div>

<div class="home">

<br>

</div>

<script>
var slideIndex = 0;
showSlides();

function showSlides() {
    var i;
    var slides = document.getElementsByClassName("mySlides");
    var dots = document.getElementsByClassName("dot");
    for (i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }
    slideIndex++;
    if (slideIndex > slides.length) {slideIndex = 1}
    for (i = 0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex-1].style.display = "block";
    dots[slideIndex-1].className += " active";
    setTimeout(showSlides, 2000); // Change image every 2 seconds
}
</script>
</body>
</html>

```

## Predict.html

```

<!DOCTYPE html>
<html >

<head>
    <meta charset="UTF-8">

```

```

    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title> Plant Disease Prediction</title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet">
    <script
src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script
src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Merriweather'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>
<link href="{{url_for('static',filename='css/final.css')}}" rel="stylesheet">
<style>
.header {
    top:0;
    margin:0px;
    left: 0px;
    right: 0px;
    position: fixed;
    background-color: #28272c;
    color: white;
    box-shadow: 0px 8px 4px grey;
    overflow: hidden;
    padding-left:20px;
    font-family: 'Josefin Sans';
    font-size: 2vw;
    width: 100%;
    height:8%;
    text-align: center;
}
    .topnav {
overflow: hidden;
background-color: #333;
}

.topnav-right a {

```

```

float: left;
color: #f2f2f2;
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 18px;
}

.topnav-right a:hover {
  background-color: #ddd;
  color: black;
}

.topnav-right a.active {
  background-color: #565961;
  color: white;
}

.topnav-right {
  float: right;
  padding-right: 100px;
}

.login{
margin-top: -70px;
}
body {

  background-color: #ffffff;
  background-repeat: no-repeat;
  background-size: cover;
  background-position: 0px 0px;
}
.login{
  margin-top: 100px;
}

.container {
  margin-top: 40px;
  padding: 16px;
}
select {
  width: 100%;
  margin-bottom: 10px;
  background: rgba(255,255,255,255);
  border: none;
  outline: none;
  padding: 10px;

```

```

font-size: 13px;
color: #000000;
text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
border: 1px solid rgba(0,0,0,0.3);
border-radius: 4px;
box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px
rgba(255,255,255,0.2);
-webkit-transition: box-shadow .5s ease;
-moz-transition: box-shadow .5s ease;
-o-transition: box-shadow .5s ease;
-ms-transition: box-shadow .5s ease;
transition: box-shadow .5s ease;
}

</style>
</head>

<body style="font-family:Montserrat;overflow:scroll;">

<div class="header">
  <div style="width:50%;float:left;font-size:2vw;text-align:left;color:white;
padding-top:1%">Plant Disease Prediction</div>
  <div class="topnav-right" style="padding-top:0.5%;">

    </div>
  </div>
<div class="container">
  <div id="content" style="margin-top:2em">
    <div class="container">
      <div class="row">
        <div class="col-sm-6 bd" >

          <br>
          
        </div>
        <div class="col-sm-6">
          <div>
            <h4>Drop in the image to get the prediction </h4>
            <form action = "" id="upload-file" method="post"
enctype="multipart/form-data">
              <select name="plant">

                <option value="select" selected>Select plant
type</option>

                <option value="fruit">Fruit</option>

```



```

        <option value="vegetable">Vegetable</option>
    </select><br>
    <label for="imageUpload" class="upload-label"
style="background: black;">
        Choose...
    </label>
    <input type="file" name="image" id="imageUpload" accept=".png,
.jpg, .jpeg">
</form>

    <div class="image-section" style="display:none;">
        <div class="img-preview">
            <div id="imagePreview">
            </div>
        </div>
        <div>
            <button type="button" class="btn btn-info btn-lg "
id="btn-predict" style="background: #7cd0d3;">Predict!</button>
        </div>
    </div>

    <div class="loader" style="display:none;"></div>

    <h3>
        <span id="result" style="font-size:17px; "> </span>
    </h3>

</div>
</div>

    </div>
</div>
</div>
</div>
</body>

<footer>
    <script src="{{url_for('static',filename='js/main.js')}}"
type="text/javascript"></script>
</footer>
</html>

```

## Final.css

```

.img-preview {
    width: 256px;

```

```
height: 256px;
position: relative;
border: 5px solid #F8F8F8;
box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
margin-top: 1em;
margin-bottom: 1em;
}
```

```
.img-preview>div {
width: 100%;
height: 100%;
background-size: 256px 256px;
background-repeat: no-repeat;
background-position: center;
}
```

```
input[type="file"] {
display: none;
}
```

```
.upload-label{
display: inline-block;
padding: 12px 30px;
background: #28272c;
color: #fff;
font-size: 1em;
transition: all .4s;
cursor: pointer;
}
```

```

.upload-label:hover{
    background: #C2C5A8;
    color: #39D2B4;
}

.loader {
    border: 8px solid #f3f3f3; /* Light grey */
    border-top: 8px solid #28272c; /* Blue */
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}

```

## Main.js

```

$(document).ready(function () {
    // Init
    $('.image-section').hide();
    $('.loader').hide();
    $('#result').hide();

    // Upload Preview
    function readURL(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();
            reader.onload = function (e) {
                $('#imagePreview').css('background-image', 'url(' +
e.target.result + ')');
                $('#imagePreview').hide();
                $('#imagePreview').fadeIn(650);
            }
        }
    }
}

```

```

        reader.readAsDataURL(input.files[0]);
    }
}
$("#imageUpload").change(function () {
    $('.image-section').show();
    $('#btn-predict').show();
    $('#result').text('');
    $('#result').hide();
    readURL(this);
});

// Predict
$('#btn-predict').click(function () {
    var form_data = new FormData($('#upload-file')[0]);

    // Show loading animation
    $(this).hide();
    $('.loader').show();

    // Make prediction by calling api /predict
    $.ajax({
        type: 'POST',
        url: '/predict',
        data: form_data,
        contentType: false,
        cache: false,
        processData: false,
        async: true,
        success: function (data) {
            // Get and display the result
            $('.loader').hide();
            $('#result').fadeIn(600);
            $('#result').text('Prediction: '+data);
            console.log('Success!');
        },
    });
});
});
});

```

**GITHUB LINK:**

<https://github.com/IBM-EPBL/IBM-Project-15673-1659602957>

**DEMO VIDEO LINK:**

<https://youtu.be/CrwiGGWlf5Y>