TEAM ID: PNT2022TMID22999

Fertilizers Recommendation System for Disease Prediction

A PROJECT REPORT

Submitted By

HARSHINI T(913119104032)

PAARKAVI PRIYA S(913119104065)

KHARSHITHA BHUVANI E(913119104045)

DIVYA SRI N (913119104023)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY MADURAI

ANNA UNIVERSITY: CHENNAI 600 025

NALAIYA THIRAN : Executed by IBM

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this project report “**Fertilizers Recommendation System for Disease Prediction**” is the bonafide work of **“HARSHINI T(913119104032), PAARKAVI PRIYA S(913119104065), KHARSHITHA BHUVANI E(913119104045), DIVYA SRI N(913119104023)”** who carried out the project work under my supervision along with industrial mentors of IBM.

SIGNATURE	**SIGNATURE**
Dr. R. Deepalakshmi, B.E., M.E., Ph.D	Dr. R. Vijayalakshmi, M.E.,Ph.D.
DEAN (Planning & Development)	**SUPERVISOR**
Professor and Head,	**Associate Professor,**
CSE,	CSE,
Velammal College of Engineering and Technology	Velammal College of Engineering and Technology
Viraganoor	Viraganoor
Madurai	Madurai

TABLE OF CONTENTS

ABSTRACT

In India, The Agriculture industry is extremely vital and crucial for economic and social development and jobs. The agricultural sector provides a living for almost 48% of the population. As per the 2019-2020 economic survey, an Indian farmer's median wage in 16 states is Rupees 2500. Most of the Indian population depends on agriculture for their livelihood. Agriculture gives an opportunity of employment to the village people to develop a country like India on large scale and give a push in the economic sector. The majority of farmers face the problem of planting an inappropriate crop for their land based on a conventional or non-scientific approach. This is a challenging task for a country like India, where agriculture feeds approximately 42% of the population. And the outcomes for the farmer of choosing the wrong crop for land is moving towards metro city for livelihoods, suicide, quitting the agriculture and give land on lease to industrialist or use for the non-agriculture purpose. The outcome of wrong crop selection is less yield and less profit.

CHAPTER 1 – INTRODUCTION

1.1 PROJECT OVERVIEW

Detection and recognition of plant diseases using machine learning are very efficient in providing symptoms of identifying diseases at its earliest. Plant pathologists can analyze the digital images using digital image processing for diagnosis of plant diseases. Application of computer vision and image processing strategies simply assist farmers in all of the regions of agriculture. Generally, the plant diseases are caused by the abnormal physiological functionalities of plants. Therefore, the characteristic symptoms are generated based on the differen- tiation between normal physiological functionalities and abnormal physiological functionalities of the plants. Mostly, the plant leaf diseases are caused by Pathogens which are posi- tioned on the stems of the plants. These different symptoms and diseases of leaves are predicted by different methods in image processing. These different methods includedifferent fundamental processes like segmentation, feature extraction and classification and so on. Mostly, the prediction and diagnosis of leaf diseases are depending on the segmentation such as segmenting the healthy tissues from diseased tissues of leaves.

1.2 PURPOSE

- Detect and recognize the plant diseases and to recommend fertilizer, it is necessary to identify the diseases and to recommend to get different and useful features needed for the purpose of analyzing later.
- Plants need 17 elements for normal growth. Carbon, hydrogen, and oxygen come from the air and water. Soil is the principal source of other nutrients. Primary nutrients (nitrogen, phosphorus, and potassium) are used in relatively large amounts by plants, and often are supplemented as fertilizers.
- Predicting the fertilizers, Analyzing the disease in a tap makes the life of farmers easy with minimal subscriptions would provide an acceptable return for the organization. This action adds a lot of value to the company and the business in society.

CHAPTER 2 - LITERATURE SURVEY

2.1 EXISTING PROBLEM

In natural systems, host plant and pathogen are constantly changing with pathogens evolving new pathogenicity to overcome host defense systems and plants evolving to reduce pathogen attack. Plant disease results from complex interactions among biotic and abiotic factors including hosts, pathogens and environments, to which should be added vectors for some diseases and human activities that modify the interaction intentionally or unintentionally through agricultural practices such as cropping systems, resistance gene deployment and application of pesticides. In recent times, technology evolutions are at their peak and the public like il-literate, rural people, are aware of technologies right now. Advantages of new technologies in agriculture:

- Steadier crop yields.
- Decrease in water, fertilizer and pesticide use, in turn, keeps food prices low.
- Reduced impact on the surrounding environment.
- Prevent runoff.
- Safety improvements for workers.

Going on with the flow, if the fertilizer recommendation is done manually it might delay the process so we are in need of an application that fastens the process.

2.2 REFERENCES

1. **Establishing a scientific basis for fertilizer recommendations for wheat in China**: Yield response and agronomic efficiency; Limin Chuana, Ping Hea; Volume 140; January 2013

2. **Design and Implementation of Crop Recommendation Fertilization Decision System Based on WEBGIS at Village Scale**; Hao Zhang, Li Zhang, Yanna Ren, Juan Zhang, Xin Xu, Xinming Ma, Zhongmin Lu; vol. 345. Springer, Berlin, Heidelberg

3. **Crop Recommendation and Fertilizer Purchase System;** Mansi Shinde1, Kimaya Ekbote, Sonali Ghorpade, Sanket Pawar, Shubhada Mone; International Journal of Computer Science and Information Technologies, Vol. 7 (2)

4. http://www.smart-fertilizer.com/

5. **Design of Precision Fertilization Management Information System on GPS and GIS Technologies;** Zhimin Liu, Weidong Xiong, Xuewei Cao; Vol 2

6. **OWL 2 Web Ontology Language:** Primer (Second Edition) Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, Sebastian Rudolph, eds.;

7. **Crop Recommendation and Fertilizer Purchase System;** Mansi Shinde, Kimaya Ekbote, Sonali Ghorpade, Sanket Pawar, Shubhada Mone; International

Journal of Computer Science and Information Technologies, Vol. 7 (2)

8. **Web Based Recommendation System for Farmers;** Kiran Shinde , Jerrin Andrei , Amey Oke; International Journal on Recent and Innovation Trends in Computing and Communication; ISSN: 2321-8169 Volume: 3

9. Models Library (http://models.pps.wur.nl)

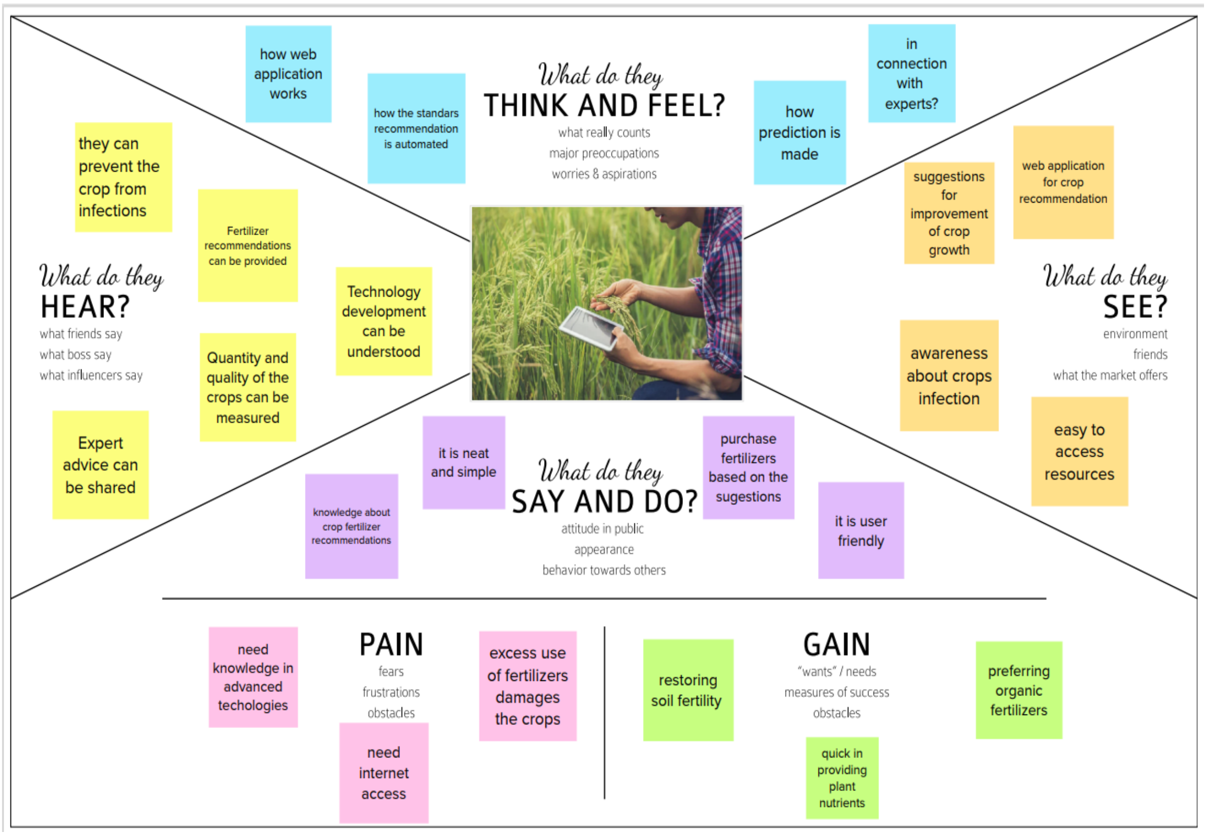
10. **Soil Test based Fertilizer Recommendation System (STFRS),** Jitendra Roy; Department of Agriculture, Government of West Bengal, 2015

2.3 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. 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. 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.

CHAPTER 3 - IDEATION & PROPOSED SOLUTION

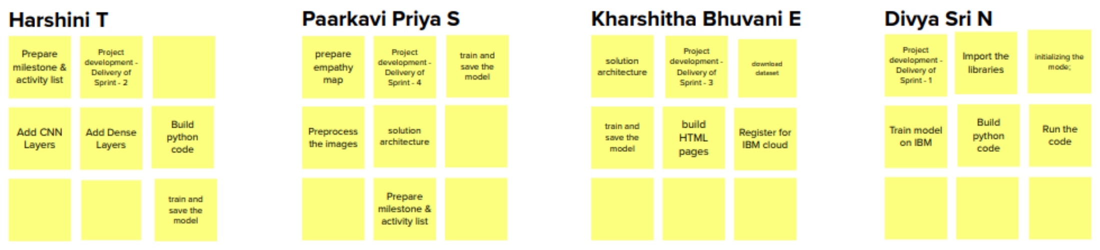
3.1 EMPATHY MAP CANVAS



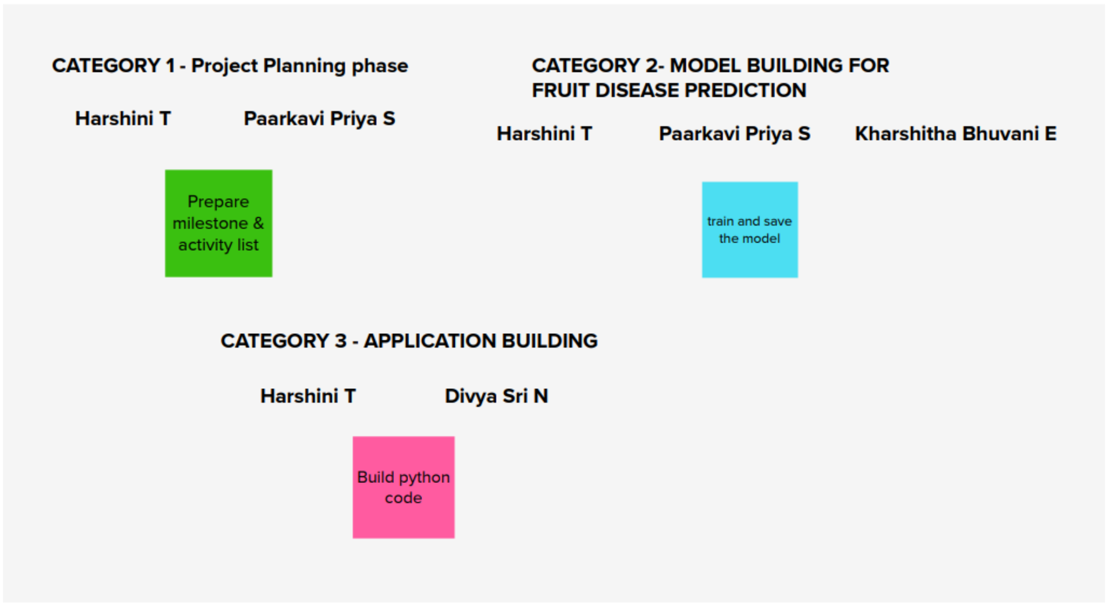
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
how web application works
how the standars recommendation is automated
how prediction is made
in connection with experts?
What do they HEAR?
what friends say
what boss say
what influencers say
they can prevent the crop from infections
Fertilizer recommendations can be provided
Technology development can be understood
Quantity and quality of the crops can be measured
Expert advice can be shared
What do they SEE?
environment
friends
what the market offers
suggestions for improvement of crop growth
web application for crop recommendation
awareness about crops infection
easy to access resources
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
it is neat and simple
knowledge about crop fertilizer recommendations
purchase fertilizers based on the sugestions
it is user friendly
PAIN
fears
frustrations
obstacles
need knowledge in advanced techologies
need internet access
excess use of fertilizers damages the crops
GAIN
"wants" / needs
measures of success
obstacles
restoring soil fertility
quick in providing plant nutrients
preferring organic fertilizers

3.2 IDEATION & BRAINSTORMING

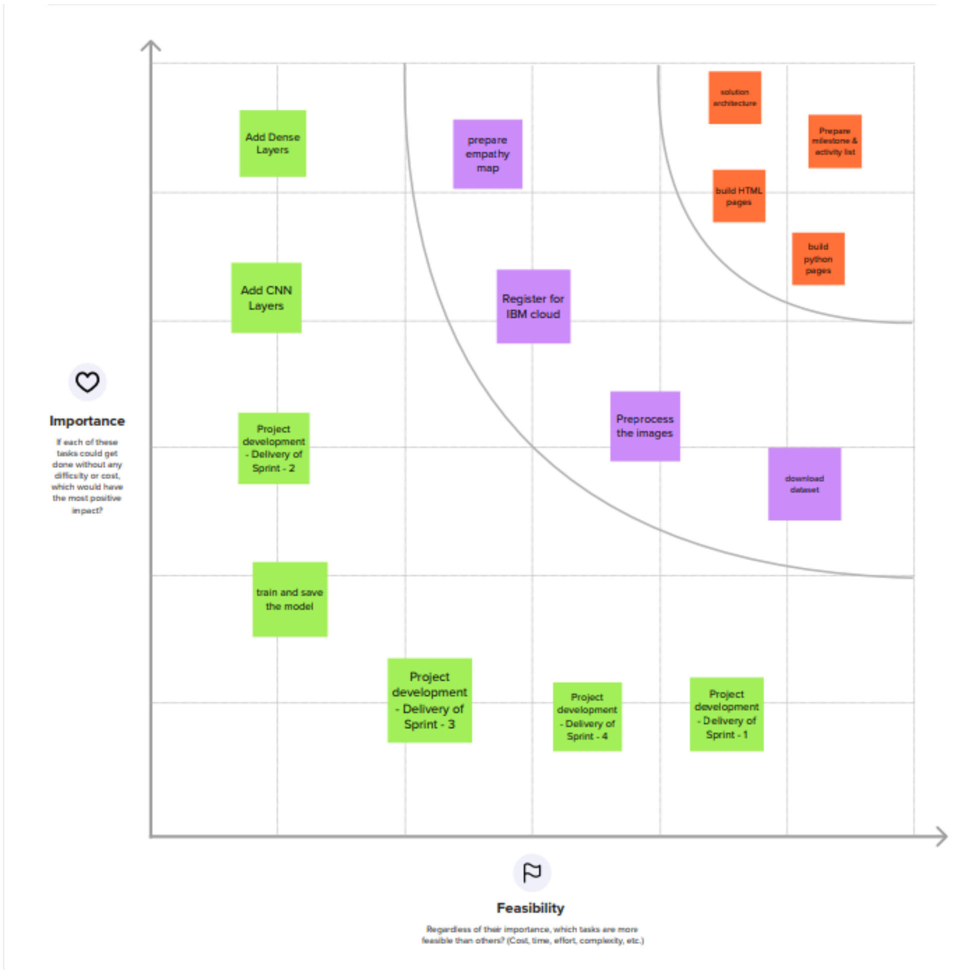
Brainstorm



Group Ideas


CATEGORY 1 - Project Planning phase
Harshini T
Paarkavi Priya S
Prepare milestone & activity list
CATEGORY 2- MODEL BUILDING FOR FRUIT DISEASE PREDICTION
Harshini T
Paarkavi Priya S
Kharshitha Bhuvani E
train and save the model
CATEGORY 3 - APPLICATION BUILDING
Harshini T
Divya Sri N
Build python code

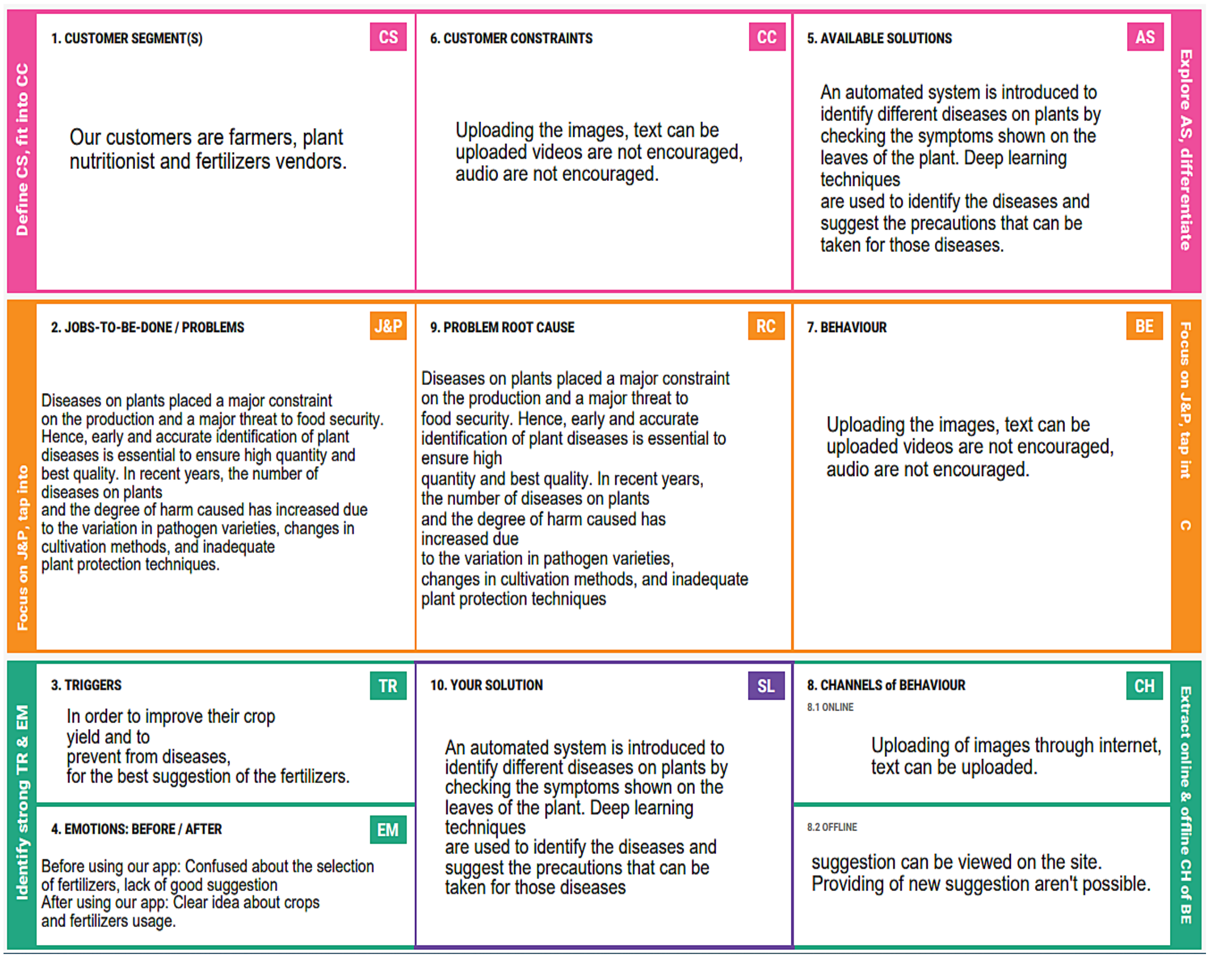
Prioritize



3.3 PROPOSED SOLUTION

The proposed system recommends the fertilizer for affected leaves based on the severity level of the crops. Fertilizers may be organic or inorganic. The user can also save the recommended fertilizers in the device's local storage and can be viewed at any time. The measurements of fertilizers (i.e.) the effectiveness of the fertilizers are suggested based on disease severity. We propose a user-friendly web application system based on machine learning. So, the user can provide the input using forms on our user interface and quickly get their results. The proposed method is also found to perform better and produce a higher number of yields. Using the proposed model, crop yield production increased and gave the super ability to decide the right combination of different types of available resources. This will help farmers and agriculture experts to adopt the method for other crops. A digital camera or similar devices can be used to capture the image of the affected leaves. Then the user uploads the image to the model. Then different image preprocessing techniques are applied to the dataset and then split into training and testing data and also to get different features needed for the purpose of analyzing leaf disease identification. Now the trained data and tested data are evaluated using the Machine Learning algorithm and then the algorithm generates an output image as a grayscale, an invert, and a smoothed one. After that, the prediction of disease is done and a suitable fertilizer is recommended to the user. Now the user can use the recommended fertilizers for the diseased plants.

3.4 PROBLEM SOLUTION FIT



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Our customers are farmers, plant nutritionist and fertilizers vendors.

6. CUSTOMER CONSTRAINTS CC

Uploading the images, text can be uploaded videos are not encouraged, audio are not encouraged.

5. AVAILABLE SOLUTIONS AS

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.

Explore AS, differentiate

Focus on J&P, tap into

2. JOBS-TO-BE-DONE / PROBLEMS J&P

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.

9. PROBLEM ROOT CAUSE RC

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

7. BEHAVIOUR BE

Uploading the images, text can be uploaded videos are not encouraged, audio are not encouraged.

Focus on J&P, tap int C

Identify strong TR & EM

3. TRIGGERS TR

In order to improve their crop yield and to prevent from diseases, for the best suggestion of the fertilizers.

4. EMOTIONS: BEFORE / AFTER EM

Before using our app: Confused about the selection of fertilizers, lack of good suggestion
After using our app: Clear idea about crops and fertilizers usage.

10. YOUR SOLUTION SL

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

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

Uploading of images through internet, text can be uploaded.

8.2 OFFLINE

suggestion can be viewed on the site. Providing of new suggestion aren't possible.

Extract online & offline CH of BE

CHAPTER 4 - REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Image Uploading	Upload from local storage
FR-4	Image Pre-processing	Evaluating using DL Algorithm
FR-5	Displaying result	Display results got from the model
FR-6	Feedback	Give feedback through forms

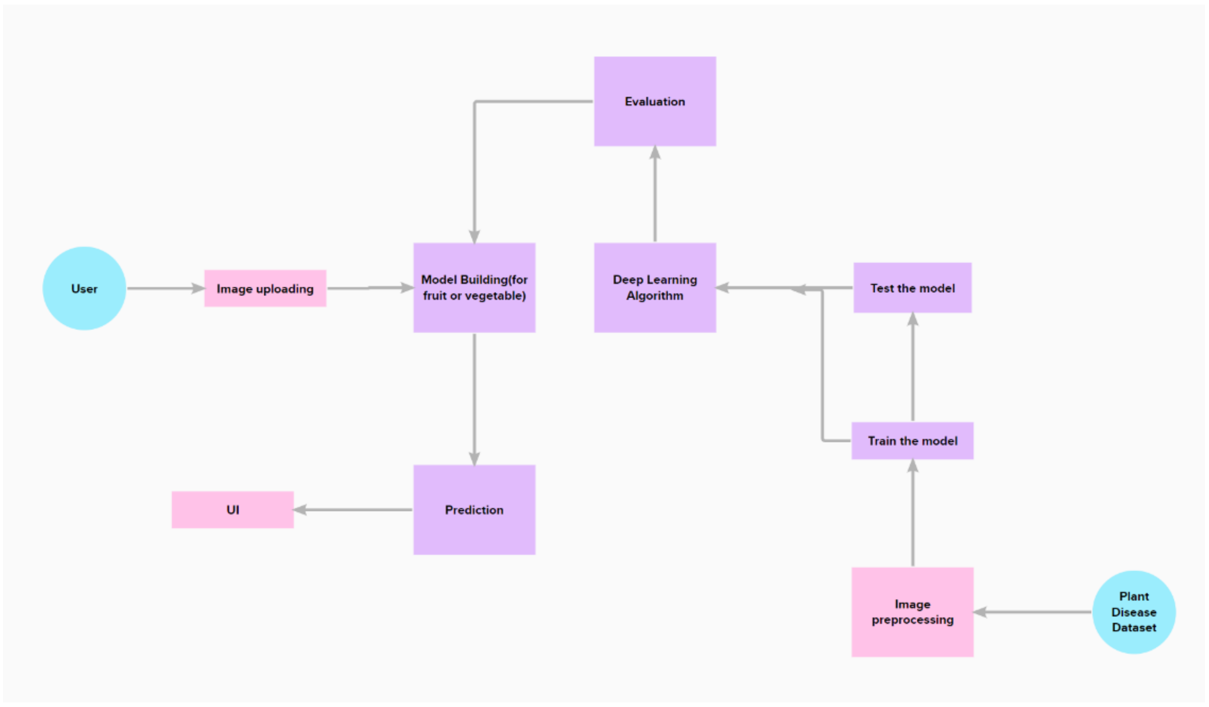
4.2 NON-FUNCTIONAL REQUIREMENT

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	We propose a user-friendly web application system based on machine learning. So, the user can provide the input using forms on our user interface and quickly get their results. The proposed method is also found to perform better and produce a higher number of yields.

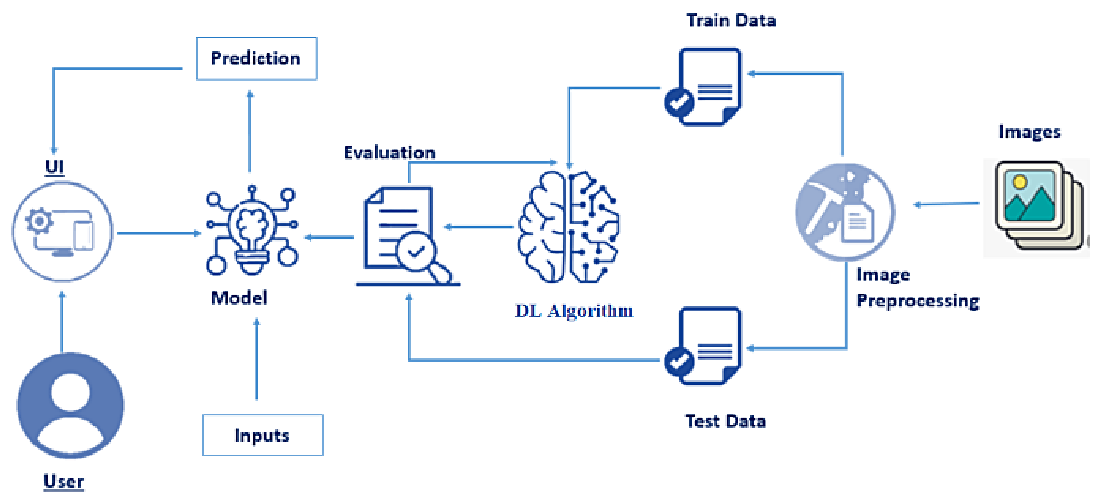
NFR-2	**Reliability**	More farmers get benefited from this system as they simply have to upload an image to get the fertilizer recommendation. Using the proposed model, crop yield production increased and gave the super ability to decide the right combination of different types of available resources. This will help farmers and agriculture experts to adopt the method for other crops.
NFR-3	**Performance**	Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases. So, it provides better performance and recommends fertilizers in a quick manner.
NFR-4	**Scalability**	More farmers get benefited from this system as they simply have to upload an image to get the fertilizer recommendation. The proposed system is also beneficial to the government in analyzing the soil condition of any region and the requirements of the farmer to maximize soil production. The fertilizer companies can use the dataset produced in the process to create customizable fertilizer depending on the need of each region

CHAPTER 5 - PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 USER STORIES

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**	**Release**
Customer (Web 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
	Login	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
	Login	USN-3	As an existing user, I can access the website with login credentials that are specific to myself.	I can register & access the dashboard with login credentials that are provided before.	Low	Sprint-2
	Dashboard	USN - 4	I am a new user, ready to explore the site via dashboard.	Registration is sufficient	High	Sprint - 1
		USN - 5	I am an existing user, pick from where I left.	Registration is enough	High	Sprint - 2
Suggestion provider	Image uploading	USN - 1	I am in need of fertilizer for the crops in my field. So I am uploading the images for the same.	Successful login and images with high clarity	High	Sprint - 3

		USN - 2	I am in need of a suggestion of my land (soil) but I don’t have a clear picture of it. So I am specifying it through the TEXT format.	Successful login and clearance in specification(data)	High	Sprint - 3
Public Recommendation For fertilizers	View of Recommends	USN - 1	I am just an explorer, the view of recommendations is adequate.	Successful login is ample	Low	Sprint - 4
Interpretation	Feedback providing	USN - 1	I finished off my journey on the website, time to provide feedback.	Successful login is enough	Low	Sprint - 4
Administrator	Registration	USN - 1	I am in search of my profile details and my exploring stuff.	Authenticated login is an adequacy.	High	Sprint - 1
	Feedback collection	USN - 2	I need to see all other peer reviewed reviews about this site.	Authenticated login is necessary.	Low	Sprint - 4

CHAPTER 6 - PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration (Customer)	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Harshini T
Sprint-1	Login	USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Divya Sri N
Sprint-1	Dashboard	USN-4	I am a new user, ready to explore the site via dashboard	2	High	Kharshitha Bhuvani E
Sprint-1	Registration (Administrator)	USN-1	I am in seek of my profile details and my exploring stuffs	2	High	Paarkavi Priya S
Sprint-2	Login	USN-3	As an existing user, I can access the website with login credentials that are specific to myself.	1	Low	Divya Sri N
Sprint-2	Dashboard	USN-5	I am an existing user, pick from where I left	1	High	Kharshitha Bhuvani E
Sprint-3	Image Uploading	USN-1	I am in need of fertilizer suggestions for the crops in my field. So I am uploading the images for the same.	1	High	Harshini T

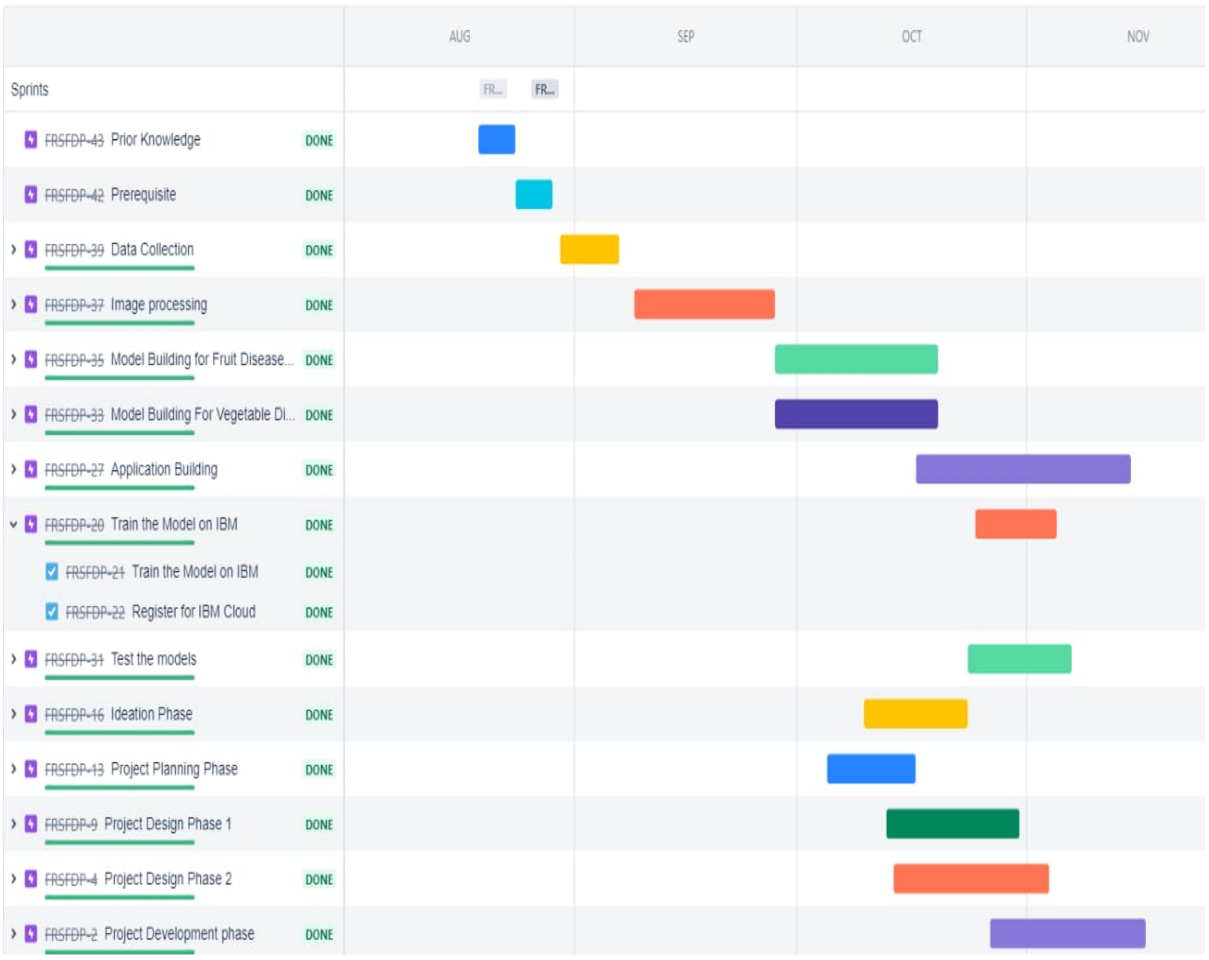
Sprint-3	Image Uploading	USN-2	I am in need of suggestions of my land(soil) but I don't have a clear picture of it. So I am specifying through text format.	2	High	Paarkavi Priya S
Sprint-4	View of recommends	USN-1	I am just an explorer, view of recommends are adequate	2	Low	Harshini T
Sprint-4	Feedback Providing	USN-1	I finished my journey on the website, time to provide feedback.	1	Low	Kharshitha Bhuvani E
Sprint -4	Feedback Collection	USN-2	I need to see all other peer members' reviews about this site.	2	Low	Divya Sri N, Paarkavi Priya S

6.2 SPRINT DELIVERY SCHEDULE

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	20	7 Days	24 Oct 2022	30 Oct 2022	20	30 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA

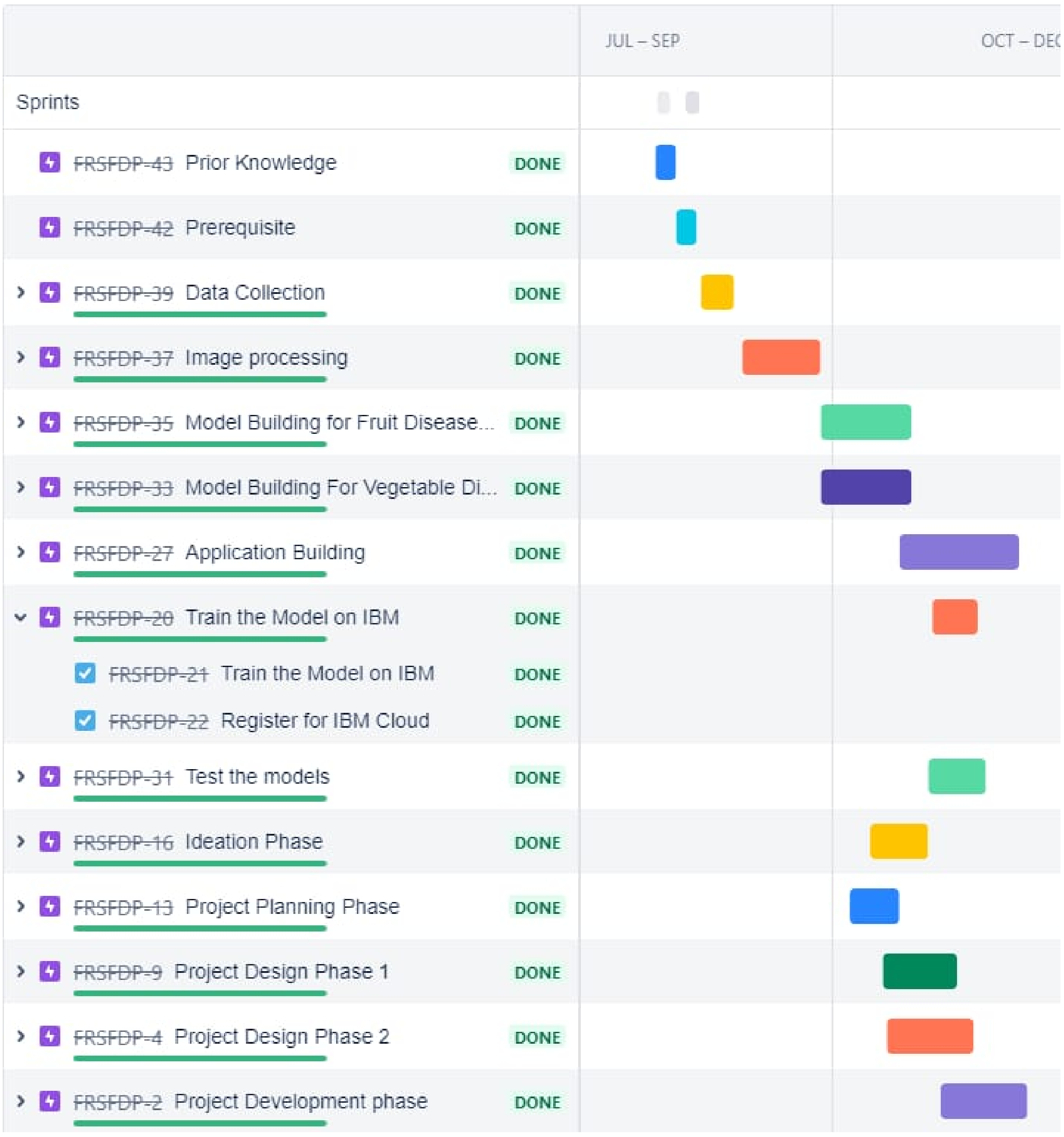
Monthly Report



Sprints

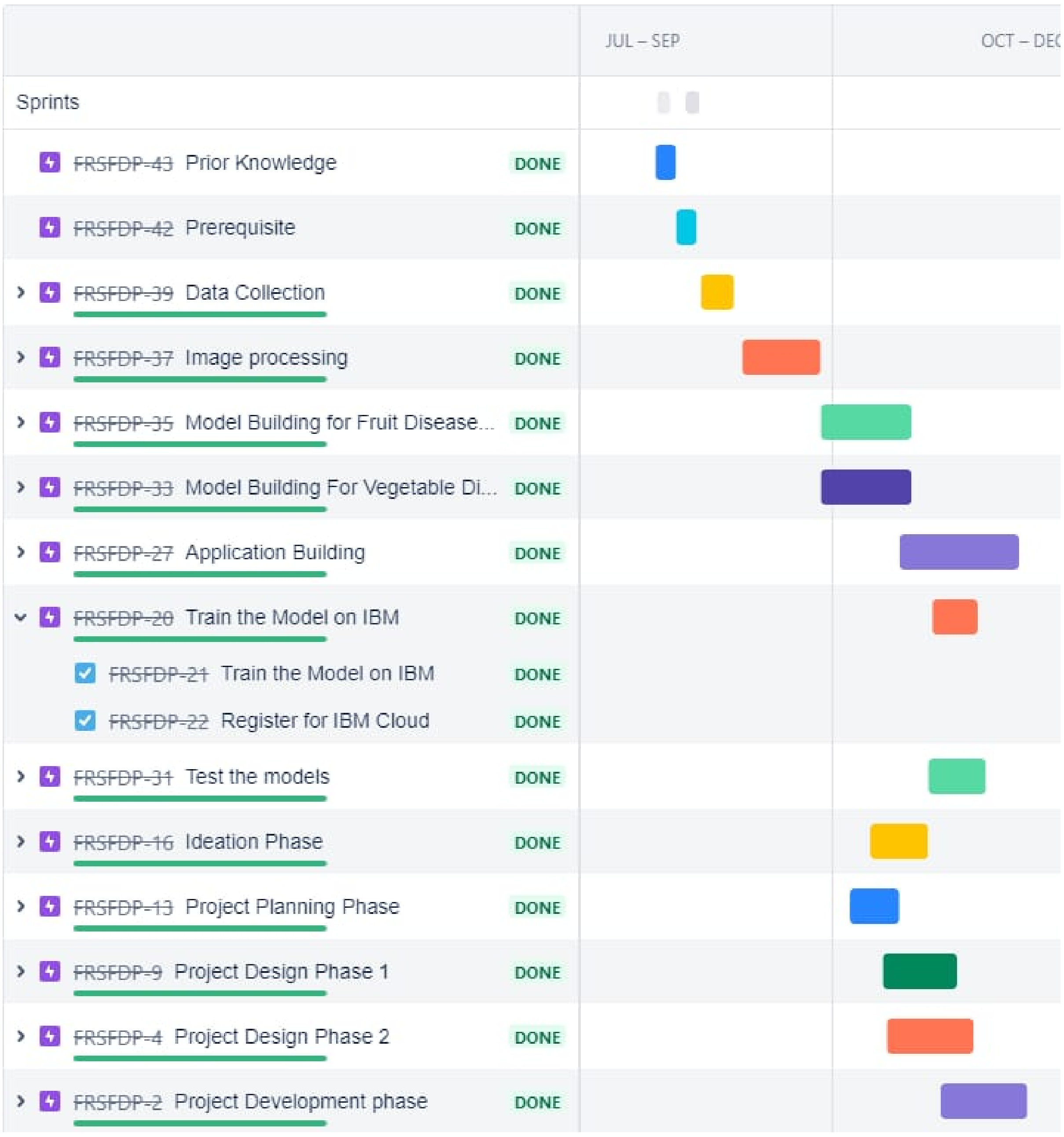
~~FRSFDP-43~~ Prior Knowledge DONE
~~FRSFDP-42~~ Prerequisite DONE
~~FRSFDP-39~~ Data Collection DONE
~~FRSFDP-37~~ Image processing DONE
~~FRSFDP-35~~ Model Building for Fruit Disease... DONE
~~FRSFDP-33~~ Model Building For Vegetable Di... DONE
~~FRSFDP-27~~ Application Building DONE
~~FRSFDP-20~~ Train the Model on IBM DONE
~~FRSFDP-21~~ Train the Model on IBM DONE
~~FRSFDP-22~~ Register for IBM Cloud DONE
~~FRSFDP-31~~ Test the models DONE
~~FRSFDP-16~~ Ideation Phase DONE
~~FRSFDP-13~~ Project Planning Phase DONE
~~FRSFDP-9~~ Project Design Phase 1 DONE
~~FRSFDP-4~~ Project Design Phase 2 DONE
~~FRSFDP-2~~ Project Development phase DONE

Weekly Report



	JUL – SEP	OCT – DEC
Sprints		
~~FRSFDP-43~~ Prior Knowledge DONE		
~~FRSFDP-42~~ Prerequisite DONE		
~~FRSFDP-39~~ Data Collection DONE		
~~FRSFDP-37~~ Image processing DONE		
~~FRSFDP-35~~ Model Building for Fruit Disease... DONE		
~~FRSFDP-33~~ Model Building For Vegetable Di... DONE		
~~FRSFDP-27~~ Application Building DONE		
~~FRSFDP-20~~ Train the Model on IBM DONE		
~~FRSFDP-21~~ Train the Model on IBM DONE		
~~FRSFDP-22~~ Register for IBM Cloud DONE		
~~FRSFDP-31~~ Test the models DONE		
~~FRSFDP-16~~ Ideation Phase DONE		
~~FRSFDP-13~~ Project Planning Phase DONE		
~~FRSFDP-9~~ Project Design Phase 1 DONE		
~~FRSFDP-4~~ Project Design Phase 2 DONE		
~~FRSFDP-2~~ Project Development phase DONE		

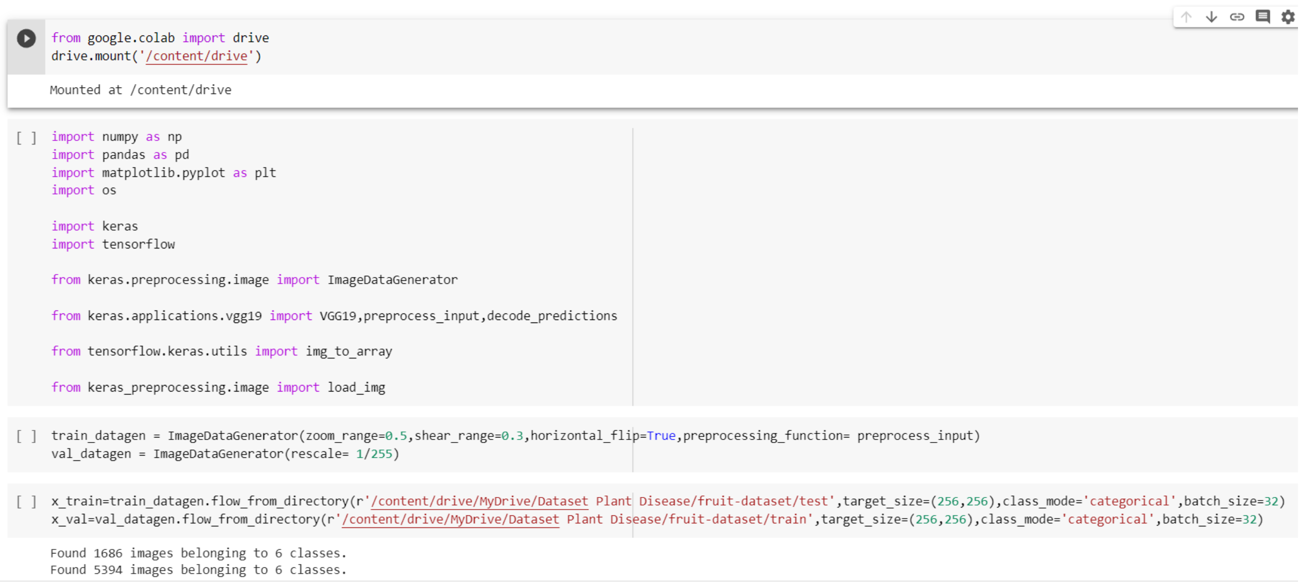
Quarter Report



	JUL – SEP	OCT – DEC
Sprints		
~~FRSFDP-43~~ Prior Knowledge DONE		
~~FRSFDP-42~~ Prerequisite DONE		
~~FRSFDP-39~~ Data Collection DONE		
~~FRSFDP-37~~ Image processing DONE		
~~FRSFDP-35~~ Model Building for Fruit Disease... DONE		
~~FRSFDP-33~~ Model Building For Vegetable Di... DONE		
~~FRSFDP-27~~ Application Building DONE		
~~FRSFDP-20~~ Train the Model on IBM DONE		
~~FRSFDP-21~~ Train the Model on IBM DONE		
~~FRSFDP-22~~ Register for IBM Cloud DONE		
~~FRSFDP-31~~ Test the models DONE		
~~FRSFDP-16~~ Ideation Phase DONE		
~~FRSFDP-13~~ Project Planning Phase DONE		
~~FRSFDP-9~~ Project Design Phase 1 DONE		
~~FRSFDP-4~~ Project Design Phase 2 DONE		
~~FRSFDP-2~~ Project Development phase DONE		

CHAPTER 7 - CODING & SOLUTIONING

7.1 Model Building for Fruit Disease Prediction

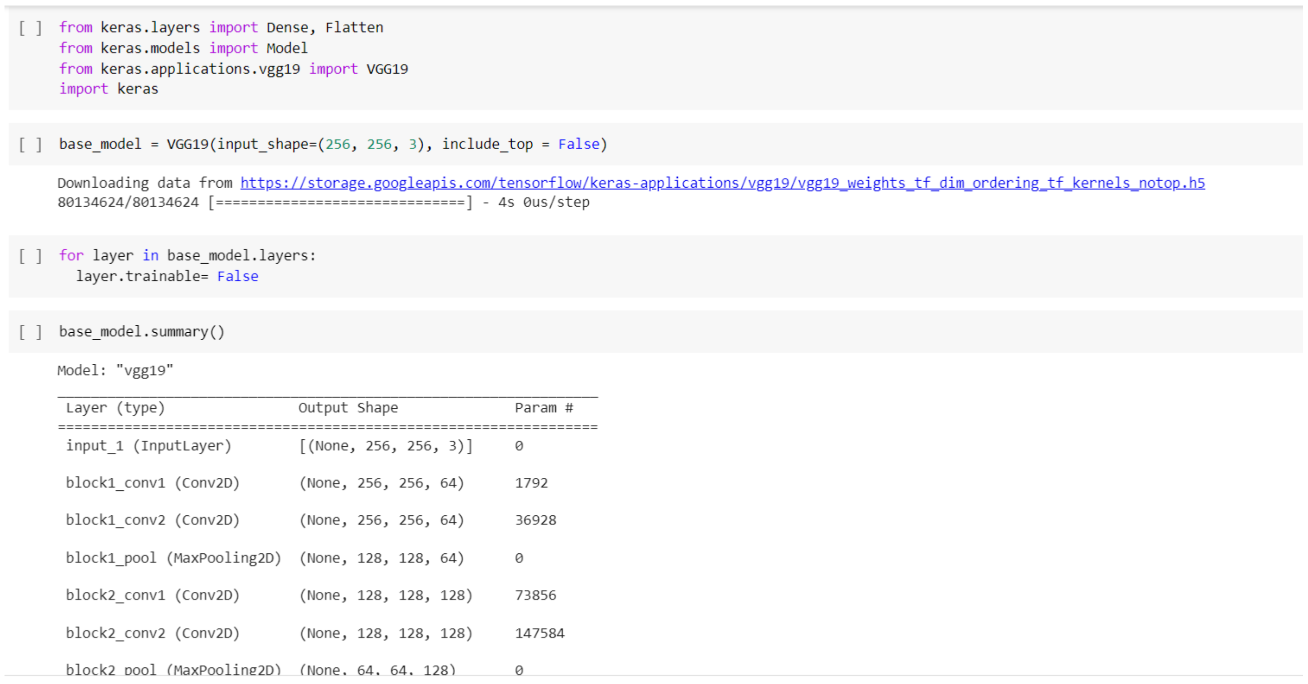


```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

import keras
import tensorflow

from keras.preprocessing.image import ImageDataGenerator

from keras.applications.vgg19 import VGG19,preprocess_input,decode_predictions

from tensorflow.keras.utils import img_to_array

from keras_preprocessing.image import load_img
```

```
train_datagen = ImageDataGenerator(zoom_range=0.5,shear_range=0.3,horizontal_flip=True,preprocessing_function= preprocess_input)
val_datagen = ImageDataGenerator(rescale= 1/255)
```

```
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset Plant Disease/fruit-dataset/test',target_size=(256,256),class_mode='categorical',batch_size=32)
x_val=val_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset Plant Disease/fruit-dataset/train',target_size=(256,256),class_mode='categorical',batch_size=32)

Found 1686 images belonging to 6 classes.
Found 5394 images belonging to 6 classes.
```

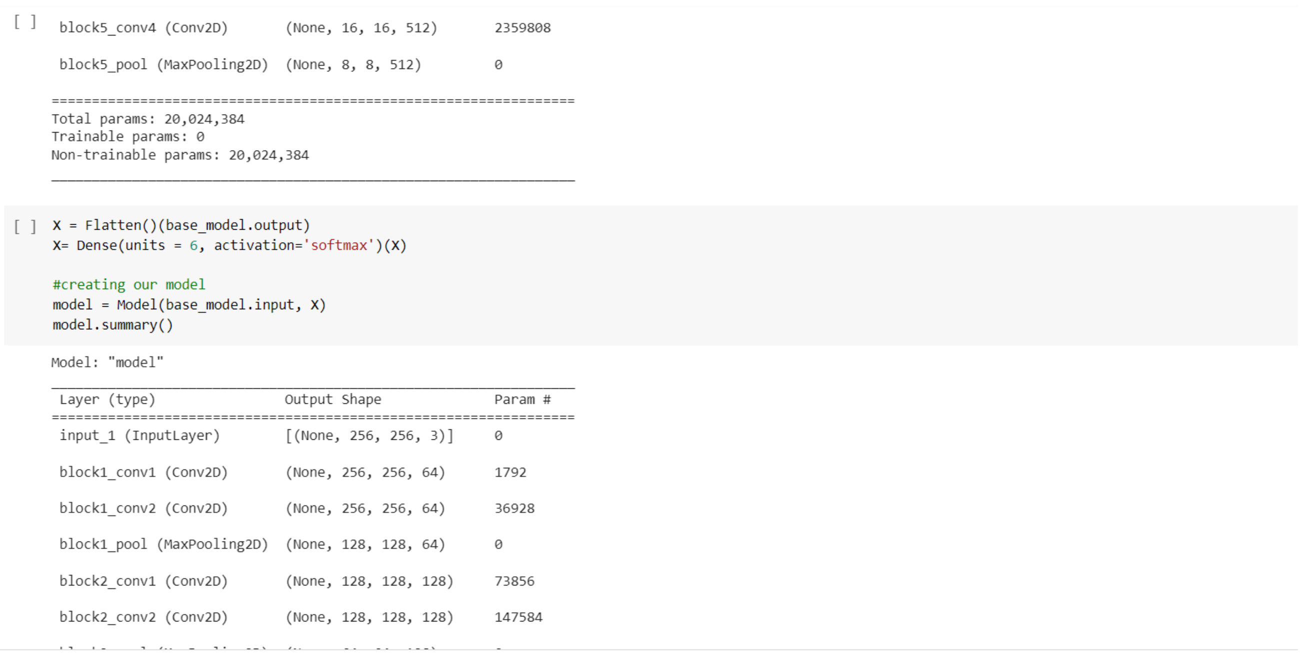


```
from keras.layers import Dense, Flatten
from keras.models import Model
from keras.applications.vgg19 import VGG19
import keras
```

```
base_model = VGG19(input_shape=(256, 256, 3), include_top = False)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 4s 0us/step
```

```
for layer in base_model.layers:
  layer.trainable= False
```

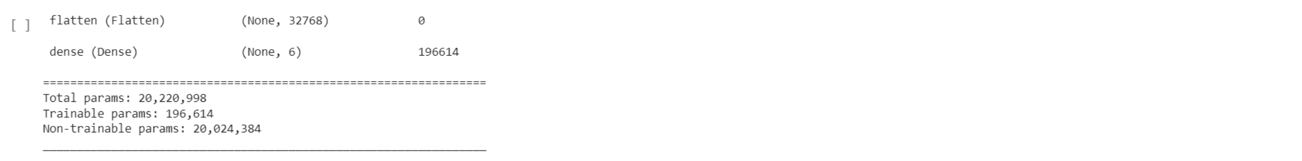
```
base_model.summary()

Model: "vgg19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0
```



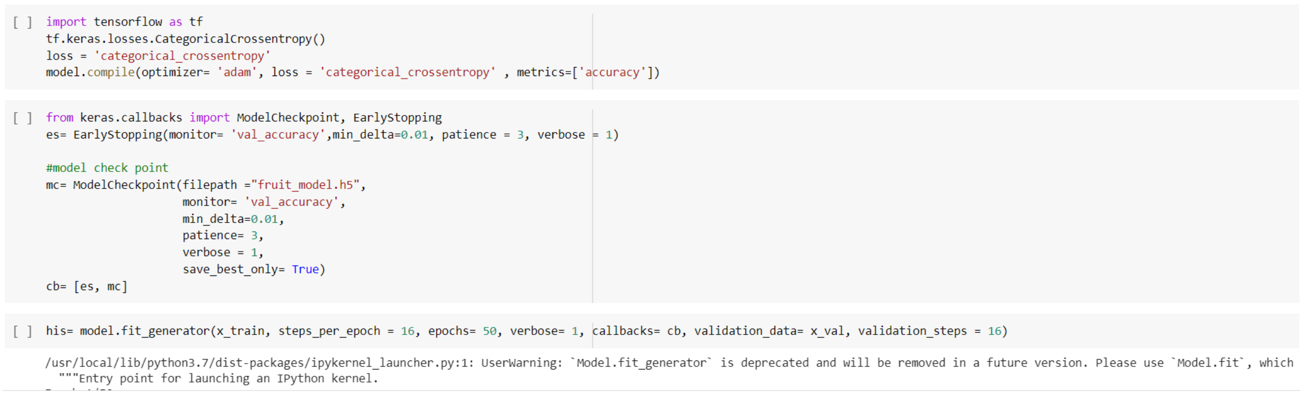
```
[ ]  block5_conv4 (Conv2D)       (None, 16, 16, 512)       2359808

     block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

    =================================================================
    Total params: 20,024,384
    Trainable params: 0
    Non-trainable params: 20,024,384
    _________________________________________________________________
```

```
[ ] X = Flatten()(base_model.output)
    X= Dense(units = 6, activation='softmax')(X)

    #creating our model
    model = Model(base_model.input, X)
    model.summary()
```

```
    Model: "model"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     input_1 (InputLayer)        [(None, 256, 256, 3)]     0

     block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

     block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

     block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

     block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

     block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584
```



```
[ ]  flatten (Flatten)           (None, 32768)             0

     dense (Dense)               (None, 6)                 196614

    =================================================================
    Total params: 20,220,998
    Trainable params: 196,614
    Non-trainable params: 20,024,384
    _________________________________________________________________
```



```
[ ] import tensorflow as tf
    tf.keras.losses.CategoricalCrossentropy()
    loss = 'categorical_crossentropy'
    model.compile(optimizer= 'adam', loss = 'categorical_crossentropy' , metrics=['accuracy'])
```

```
[ ] from keras.callbacks import ModelCheckpoint, EarlyStopping
    es= EarlyStopping(monitor= 'val_accuracy',min_delta=0.01, patience = 3, verbose = 1)

    #model check point
    mc= ModelCheckpoint(filepath ="fruit_model.h5",
                        monitor= 'val_accuracy',
                        min_delta=0.01,
                        patience= 3,
                        verbose = 1,
                        save_best_only= True)
    cb= [es, mc]
```

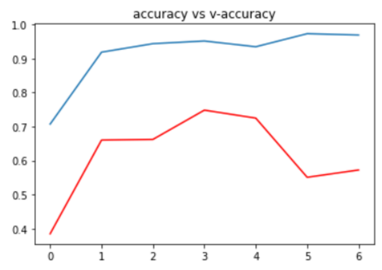
```
[ ] his= model.fit_generator(x_train, steps_per_epoch = 16, epochs= 50, verbose= 1, callbacks= cb, validation_data= x_val, validation_steps = 16)

    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which
      """Entry point for launching an IPython kernel.
```

```
16/16 [==============================] - 878s 57s/step - loss: 0.4416 - accuracy: 0.9727 - val_loss: 1.6773 - val_accuracy: 0.5508
Epoch 7/50
16/16 [==============================] - ETA: 0s - loss: 0.7685 - accuracy: 0.9688
Epoch 7: val_accuracy did not improve from 0.74805
16/16 [==============================] - 867s 56s/step - loss: 0.7685 - accuracy: 0.9688 - val_loss: 2.0626 - val_accuracy: 0.5723
Epoch 7: early stopping
```

```
h= his.history
h.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
plt.plot(h['accuracy'])
plt.plot(h['val_accuracy'], c= "red")
plt.title("accuracy vs v-accuracy")
plt.show()
```



7.2 Model Building for Vegetable Disease Prediction

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

import keras
import tensorflow

from keras.preprocessing.image import ImageDataGenerator

from keras.applications.vgg19 import VGG19,preprocess_input,decode_predictions

from tensorflow.keras.utils import img_to_array

from keras_preprocessing.image import load_img
```

```
train_datagen = ImageDataGenerator(zoom_range=0.5,shear_range=0.3,horizontal_flip=True,preprocessing_function= preprocess_input)
val_datagen = ImageDataGenerator(rescale= 1/255)
```

```
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset Plant Disease/Veg-dataset/test_set',target_size=(256,256),class_mode='categorical',batch_size=32)
x_val=val_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset Plant Disease/Veg-dataset/train_set',target_size=(256,256),class_mode='categorical',batch_size=32)
```

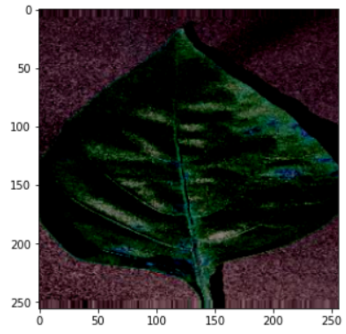
```
Found 3416 images belonging to 9 classes.
Found 11386 images belonging to 9 classes.
```

```
[ ] t_img, label = x_train.next()
```

```
[ ] def plotImage(img_arr, label):
      for im, l in zip(img_arr, label):
        plt.figure(figsize=(5,5))
        plt.imshow(im/255)
        plt.show()
```

```
[ ] plotImage(t_img[:3], label[:3])
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



```
[ ] from keras.layers import Dense, Flatten
    from keras.models import Model
    from keras.applications.vgg19 import VGG19
    import keras
```

```
[ ] base_model = VGG19(input_shape=(256, 256, 3), include_top = False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 0s 0us/step
```

```
[ ] for layer in base_model.layers:
      layer.trainable= False
```

```
base_model.summary()
```

```
Model: "vgg19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584
```

```
[ ] =================================================================
    Total params: 20,319,305
    Trainable params: 294,921
    Non-trainable params: 20,024,384
    _________________________________________________________________
```

```
[ ] import tensorflow as tf
    tf.keras.losses.CategoricalCrossentropy()
    loss = 'categorical_crossentropy'
    model.compile(optimizer= 'adam', loss = 'categorical_crossentropy' , metrics=['accuracy'])
```

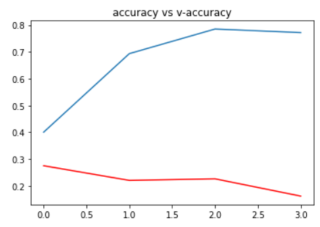
```
[ ] from keras.callbacks import ModelCheckpoint, EarlyStopping
    es= EarlyStopping(monitor= 'val_accuracy',min_delta=0.01, patience = 3, verbose = 1)

    #model check point
    mc= ModelCheckpoint(filepath ="veg_model.h5",
                        monitor= 'val_accuracy',
                        min_delta=0.01,
                        patience= 3,
                        verbose = 1,
                        save_best_only= True)
    cb= [es, mc]
```

```
[ ] his= model.fit_generator(x_train, steps_per_epoch = 16, epochs= 50, verbose= 1, callbacks= cb, validation_data= x_val, validation_steps = 16)

    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which
      """Entry point for launching an IPython kernel.
    Epoch 1/50
    16/16 [==============================] - ETA: 0s - loss: 13.5120 - accuracy: 0.4004
    Epoch 1: val_accuracy improved from -inf to 0.27539, saving model to veg_model.h5
```

```
[ ] Epoch 4/50
    16/16 [==============================] - ETA: 0s - loss: 3.9351 - accuracy: 0.7718
    Epoch 4: val_accuracy did not improve from 0.27539
    16/16 [==============================] - 875s 56s/step - loss: 3.9351 - accuracy: 0.7718 - val_loss: 6.2102 - val_accuracy: 0.1621
    Epoch 4: early stopping
```

```
[ ] h= his.history
    h.keys()

    dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
[ ] plt.plot(h['accuracy'])
    plt.plot(h['val_accuracy'], c= "red")
    plt.title("accuracy vs v-accuracy")
    plt.show()
```



```
[ ] from keras.models import load_model
    model= load_model("/content/veg_model.h5")
```

```
[ ] ref = dict(zip(list(x_train.class_indices.values()) , list(x_train.class_indices.keys())))
    ref

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

```
[ ] def prediction(path):
      img= load_img(path,target_size=(256, 256))
      i=img_to_array(img)
      im=preprocess_input(i)
      img= np.expand_dims(im, axis=0)
      pred = np.argmax(model.predict(img))
      ref = dict(zip(list(x_train.class_indices.values()) , list(x_train.class_indices.keys())))
      print(f" the image belongs to {pred}{ref[pred]}")
```

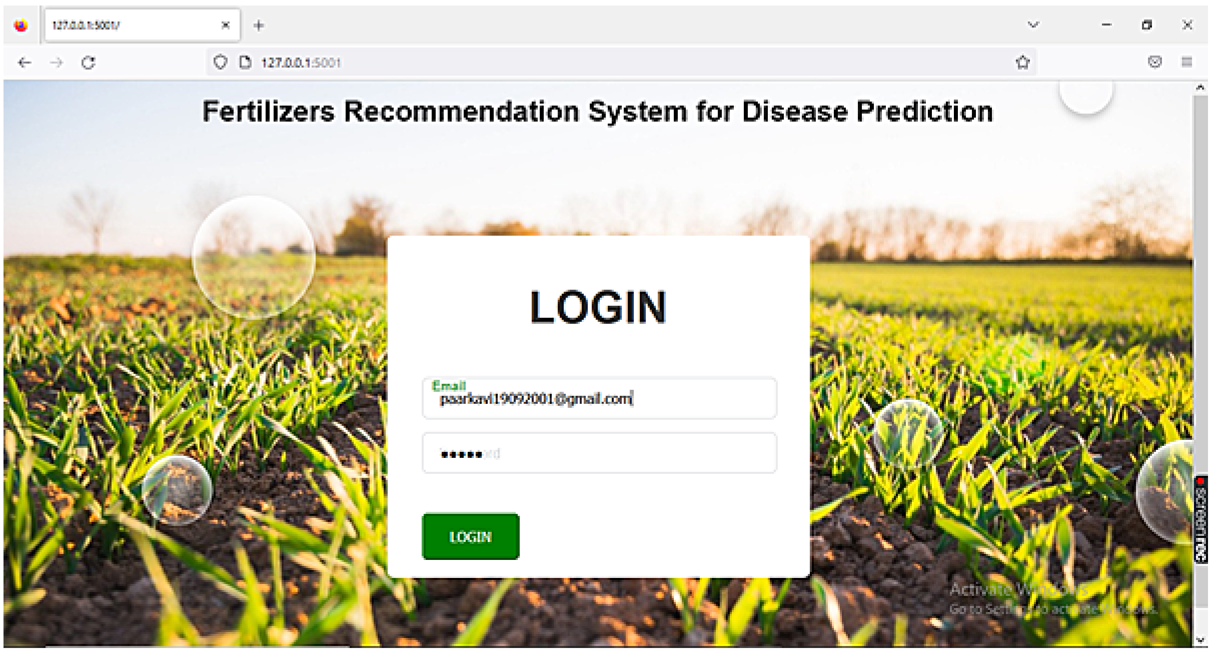
```
[ ] #prediction of images into numbers
    path = "/content/drive/MyDrive/Dataset Plant Disease/Veg-dataset/test_set/Pepper,_bell___Bacterial_spot/ad921dec-e88f-41d8-9455-0880c69063fc___NREC_B.Spot 9216.JPG"
    prediction(path)

    1/1 [==============================] - 1s 913ms/step
     the image belongs to 0Pepper,_bell___Bacterial_spot
```

7.3 Login page - signup.html

In Login page, the user enters email id and password. By using regex we will be checking whether the email id is vali

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <link rel="stylesheet" href="../static/signup_style.css">
</head>
<body>
  <div class="top">
    <h1 class="logo">Fertilizers Recommendation System for Disease
Prediction</h1>
  </div>
</div>
    <div class="signupFrm">
    <form action="/dashboard" class="form" onsubmit="return validate()"
enctype="multipart/form-data" method="post">
      <h1 class="title">LOGIN</h1>
      <div class="inputContainer">
        <input type="text" class="input" placeholder="Enter email" id="email"
required>
        <label for="email" class="label">Email</label>
      </div>
      <div class="inputContainer">
        <input type="password" class="input" placeholder="Enter password"
id="psw" required>
        <label for="psw" class="label">Password</label>
      </div>
      <button type="submit" class="signupbtn" value="Sign up" >LOGIN</button>
    </form>
  </div>
    <div id="background-wrap">
    <div class="bubble x1"></div>
    <div class="bubble x2"></div>
    <div class="bubble x3"></div>
    <div class="bubble x4"></div>
    <div class="bubble x5"></div>
    <div class="bubble x6"></div>
    <div class="bubble x7"></div>
    <div class="bubble x8"></div>
    <div class="bubble x9"></div>
    <div class="bubble x10"></div>
</div>
<script>
  function validate(){
    var email=document.getElementById("email").value;
    var pword=document.getElementById("psw").value;


      if(email.match(/^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$/)){
        if(pword.length>8 &&pword.length<12){
          window.location.replace("upload.html")
        }
        else{
          alert("password small")
        }
      }
      else{
        alert("invalid email")}}
</script>
</body>
</html>
```

OUTPUT:


Fertilizers Recommendation System for Disease Prediction

LOGIN

Email
paarkavi19092001@gmail.com

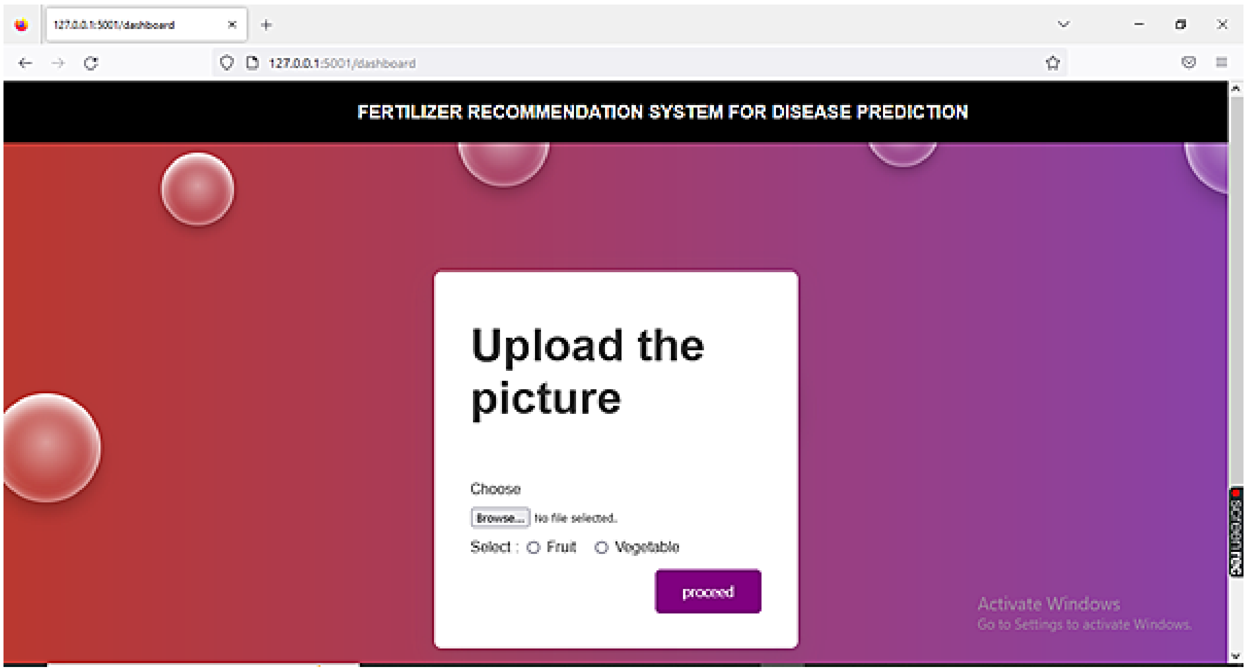
7.4 Dashboard - index.html

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
   <link rel="stylesheet" href="../static/index_style.css">
</head>
<body>
  <div class="top">
    <h1 class="logo">Fertilizers Recommendation System for Disease
Prediction</h1>
  </div>
  <div class="forms">
    <div class="form1">
      <form action="/predict" method="post" enctype="multipart/form-data"
class="form_upload" onsubmit="return validate()">
        <div class="uploadform">
          <div class="inputContainer">
            <h1 class="title">Image Uploading</h1>
            <label for="imageUpload" class="upload-label">Choose</label>
            <input type="file" name="files" id="imageUpload"
onchange="loadfile(event)" accept=".png, .jpg, .jpeg"><br>
```

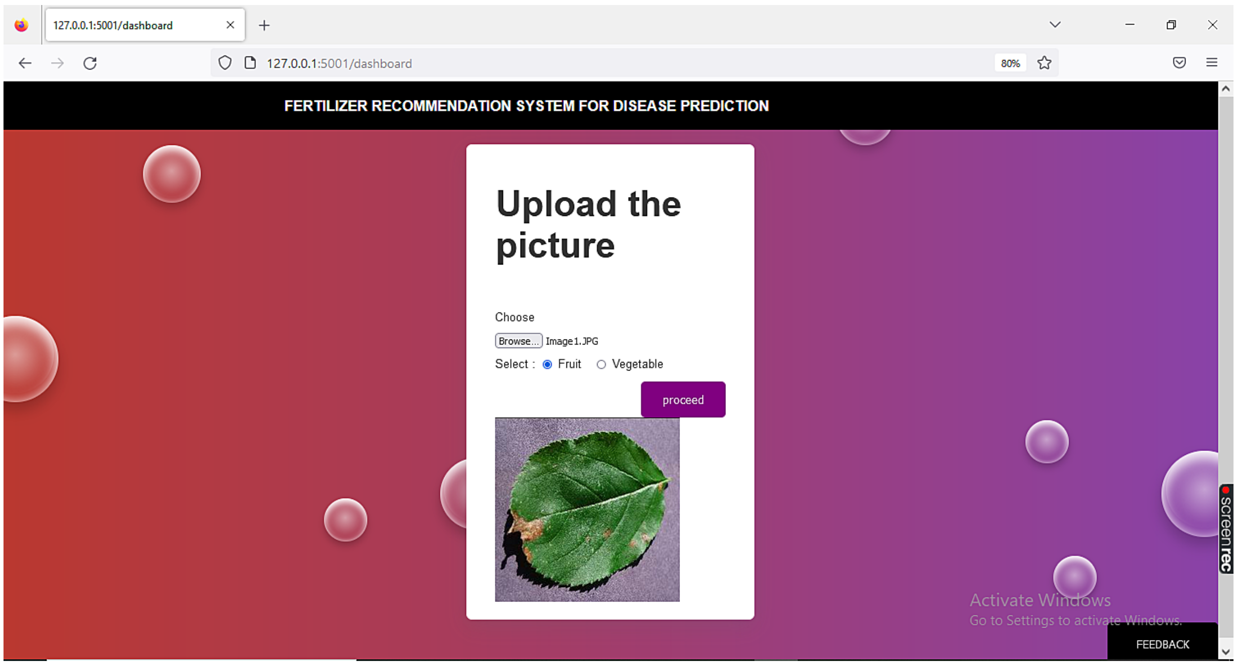
```
            Select : <input type="radio" name="select_plant" id="fruit"
value="fruit">
            <label for="fruit">Fruit</label>
            <input type="radio" name="select_plant" id="veg"
value="vegetable">
            <label for="veg">Vegetable</label> <br>
            <button type="submit" class="signupbtn" value="Sign up"
class="signupbtn" >proceed</button>
          </div><br>
          <div>
            <img id="output">
            <h2 class="result_pred">{{result}}</h2>
          </div>
        </div>
      </form>
    </div><br>

    <div class="form2">
      <form action="/feedback" method="post" class="form_feedback">
        <button type="submit" >FEEDBACK</button>
      </form>
    </div>
  </div>

      <script>
        var loadfile=function(event){
            var image=document.getElementById('output');
            image.src=URL.createObjectURL(event.target.files[0]);
        };
    </script>
    <div id="background-wrap">
      <div class="bubble x1"></div>
      <div class="bubble x2"></div>
      <div class="bubble x3"></div>
      <div class="bubble x4"></div>
      <div class="bubble x5"></div>
      <div class="bubble x6"></div>
      <div class="bubble x7"></div>
```

```
        <div class="bubble x8"></div>
        <div class="bubble x9"></div>
        <div class="bubble x10"></div>
    </div>
</body>
</html>
```

OUTPUT:



127.0.0.1:5001/dashboard
FERTILIZER RECOMMENDATION SYSTEM FOR DISEASE PREDICTION
Upload the picture
Choose
Browse... No file selected.
Select : Fruit Vegetable
proceed
Activate Windows
Go to Settings to activate Windows.



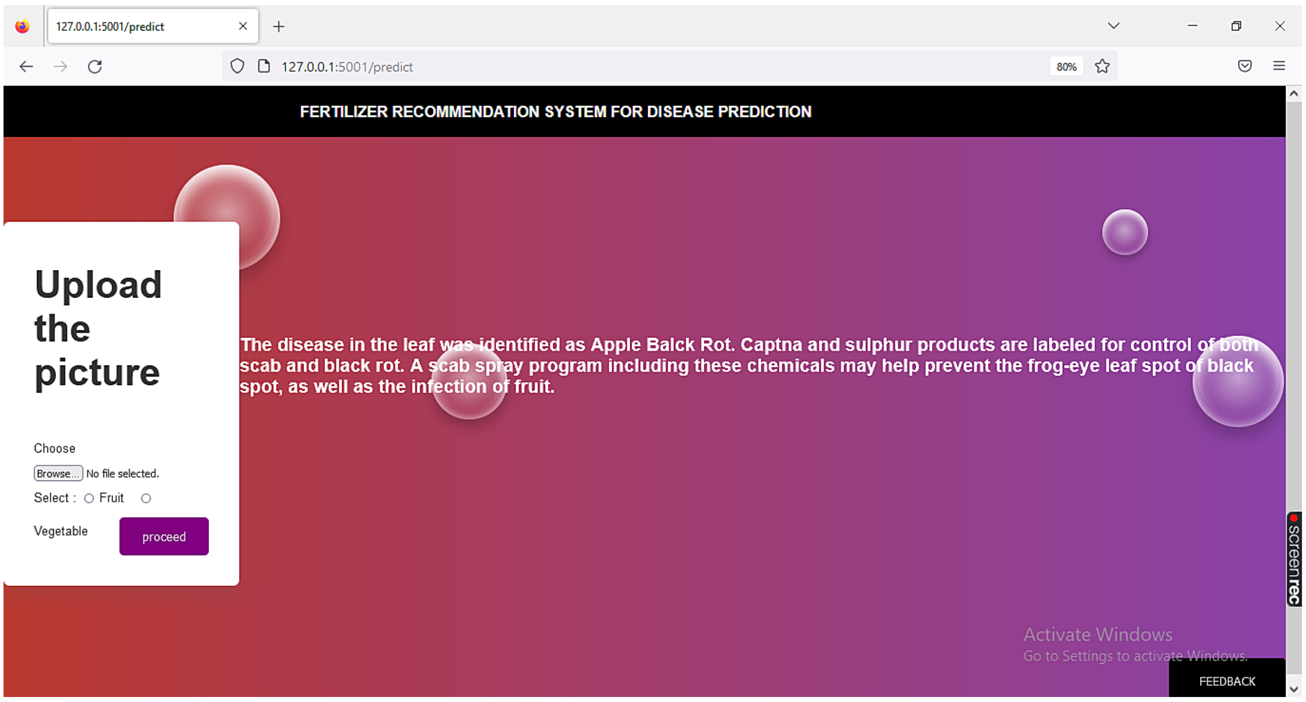
127.0.0.1:5001/dashboard
80%
FERTILIZER RECOMMENDATION SYSTEM FOR DISEASE PREDICTION
Upload the picture
Choose
Browse... Image1.JPG
Select : Fruit Vegetable
proceed
Activate Windows
Go to Settings to activate Windows.
FEEDBACK

7.5 Python Flask - app.py

```
import numpy as np
import os
import glob
from keras.applications.vgg19 import preprocess_input
from tensorflow.keras.utils import img_to_array
from tensorflow.keras.models import load_model
from keras_preprocessing.image import load_img
# Flask utils
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename

app = Flask(__name__,template_folder='template')

FRUIT_MODEL_PATH='fruit_model.h5'
model_fruit=load_model(FRUIT_MODEL_PATH)

VEG_MODEL_PATH='veg_model.h5'
model_veg=load_model(VEG_MODEL_PATH)

def model_fruitdata(img_path):
    img= load_img(img_path,target_size=(256, 256))
    i=img_to_array(img)
    im=preprocess_input(i)
    img= np.expand_dims(im, axis=0)
    prediction = np.argmax(model_fruit.predict(img))
    if prediction==0:
        prediction="The disease in the leaf was identified as Apple Black Rot.
Captna and sulphur products are labeled for control of both scab and black
rot. A scab spray program including these chemicals may help prevent the
frog-eye leaf spot of black spot, as well as the infection of fruit."
    elif prediction==1:
        prediction="The leaf is a fresh Apple Plant"
    elif prediction==2:
        prediction="The leaf is a fresh Corn(Maize) Plant"
    elif prediction==3:
        prediction="The disease in the leaf was identified as Northern Corn
Leaf Blight. In addition, the rate K60 is the one effective in control of
Northern Corn Leaf Blight in the field. However, there is a need for further
studies in the greenhouse. Thus, the availability of fertilizer at the K60 to
farmers in the endemic zones could help for sustainable management of
Northern Leaf Blight in maize. "
    elif prediction==4:
        prediction="The disease in the leaf was identified as Peach Bacterial
Spot. Compounds available for use on peach and nectarine for bacterial spot
include copper, oxytetracycline (Mycoshield and generic equivalents), and
syllit+captan; however, repeated applications are typically necessary for
even minimal disease control."
    else:
        prediction="The leaf is a fresh Peach PLant"
    return prediction

def model_vegdata(img_path):
    img= load_img(img_path,target_size=(256, 256))
    i=img_to_array(img)
    im=preprocess_input(i)
    img= np.expand_dims(im, axis=0)
    prediction = np.argmax(model_veg.predict(img))
    if prediction==0:
        prediction="The disease in the leaf was identified as Pepper Bell
Bacterial Spot.Copper sprays can be used to control bacterial leaf spot, but
they are not as effective when used alone on a continuous basis. Thus,
combining these sprays with a plant resistance inducer, such as Regalia or
Actigard, can provide good protection from the disease."
    elif prediction==1:
        prediction="The leaf is a Healthy Pepper Bell Plant"
    elif prediction==2:
        prediction="The disease in the leaf was identified as Potato Early
Blight. Mancozeb and chlorothalonil are perhaps the most frequently used
protectant fungicides for early blight management but provide insufficient
control under high disease pressure."
    elif prediction==3:
        prediction="The leaf is a Healthy Potato Plant."
    elif prediction==4:
        prediction="The disease in the leaf was identified as Potato Late
Blight. If there is some sign of blight and the potatoes are not mature, use
Dithane (mancozeb) MZ or you can also use Tattoo C or Acrobat MZ. Acrobat
used later in the season reduces late blight spores."
    elif prediction==5:
        prediction="The disease in the leaf was identified as Tomato Bacterial
Spot. A plant with bacterial spot cannot be cured. Remove symptomatic plants
from the field or greenhouse to prevent the spread of bacteria to healthy
plants. Burn, bury or hot compost the affected plants and DO NOT eat
symptomatic fruit."
    elif prediction==6:
        prediction="The disease in the leaf was identified as Tomato Late
Blight. Copper products can effectively control, or slow down, late blight
epidemics. Copper products have no activity. Therefore, they need to be
applied to all plant surfaces before infection (before symptoms are observed
in the field) and frequently so new foliage is protected as plants grow."
    elif prediction==7:
        prediction="The disease in the leaf was identified as Tomato Leaf
Mold. Applying fungicides when symptoms first appear can reduce the spread of
the leaf mold fungus significantly. Several fungicides are labeled for leaf
mold control on tomatoes and can provide good disease control if applied to
all the foliage of the plant, especially the lower surfaces of the leaves."
    else:
        prediction="The disease in the leaf was identified as Tomato Septoria
Leaf Spot. Most fungicides registered for use on tomatoes would effectively
control Septoria leaf spot. These include maneb, mancozeb, chlorothalonil,
and benomyl. Captan is not effective and zineb may be difficult to purchase."
    return prediction

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

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

@app.route('/feedback',methods=['GET','POST'])
```

```
def feedback():
    return render_template('feedback.html')

@app.route('/predict', methods=['GET','POST'])
def upload():
    if request.method=='POST':
        f=request.files['files']
        fv=request.form['select_plant']

        if fv=="fruit":
            basepath=os.path.dirname(__file__)
file_path=os.path.join(basepath,'uploads',secure_filename(f.filename))
            f.save(file_path)
            prediction=model_fruitdata(file_path)
            res=prediction
            return
render_template('index.html',result='{}'.format(res),res=res)
        else:
            basepath=os.path.dirname(__file__)
file_path=os.path.join(basepath,'uploads',secure_filename(f.filename))
            f.save(file_path)
            prediction=model_vegdata(file_path)
            res=prediction
            return
render_template('index.html',result='{}'.format(res),res=res)

if __name__ == '__main__':
    app.run(port=5001,debug=True)
```

OUTPUT :



127.0.0.1:5001/predict

FERTILIZER RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

Upload the picture

Choose

Browse... No file selected.

Select : ○ Fruit ○ Vegetable

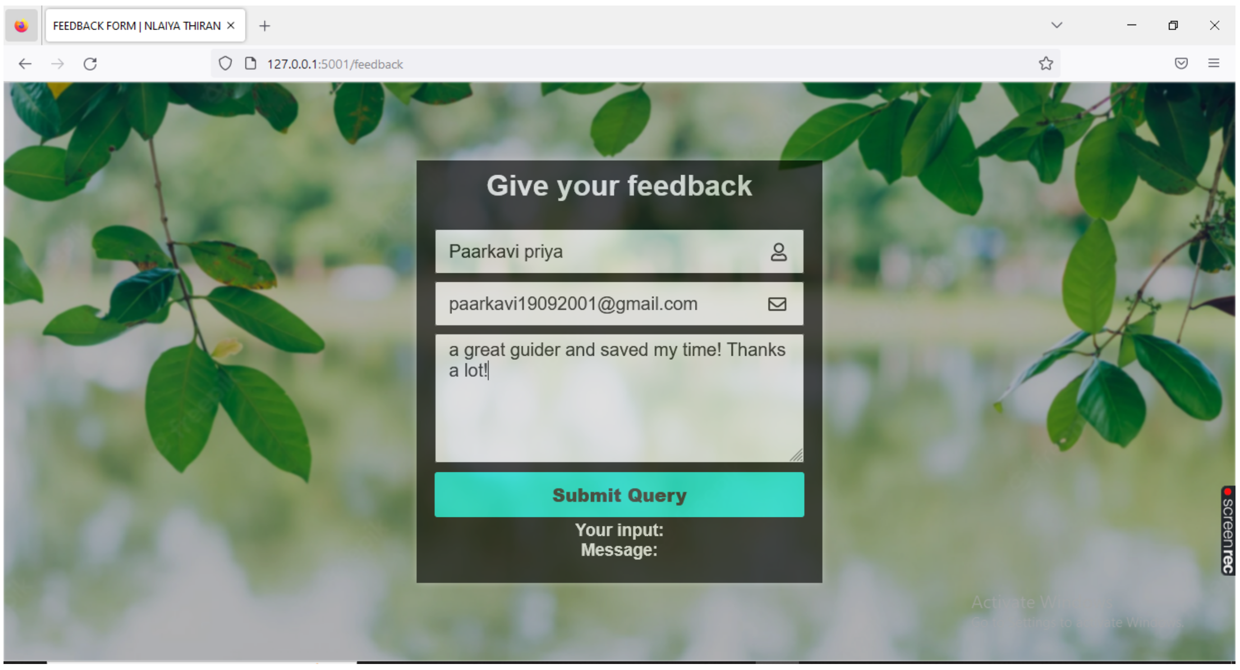
proceed

The disease in the leaf was identified as Apple Balck Rot. Captna and sulphur products are labeled for control of both scab and black rot. A scab spray program including these chemicals may help prevent the frog-eye leaf spot of black spot, as well as the infection of fruit.

7.6 Feedback - feedback.html

```
<!DOCTYPE html>
<html>
<head>
  <title>FEEDBACK FORM | NLAIYA THIRAN</title>
  <link rel="stylesheet" href="../static/feedback_style.css">
  <script src="https://kit.fontawesome.com/67c66657c7.js"></script>
</head>
<body>
  <section></section>
  <div class="container">
    <h1>Give your feedback</h1>
    <form id="details">
      <div class="id">
      <input type="text" placeholder="Full name">
        <i class="far fa-user"></i>
      </div>
      <div class="id">
        <input type="email" name=""  placeholder="Email address">
        <i class="far fa-envelope"></i>
      </div>
      <textarea cols="15" rows="5" placeholder="Enter Your opinions here.."
id="user_input"></textarea>
      <input type="submit" onclick="showInput();" class="btn"><br/>
      <label style="color: white"><h3>Your input:</h3> </label>
      <p><h3 style="color: white">Message: <span id='display' style="color:
white" ></span></h3></p>
    </form>
  </div>
<br>
<br>
  <script language="JavaScript">
    function showInput() {
        document.getElementById('display').innerHTML =
document.getElementById("user_input").value;
        alert("your response has been stored");
    }
  </script>
</body>
</html>
```

OUTPUT:



Give your feedback

Paarkavi priya

paarkavi19092001@gmail.com

a great guider and saved my time! Thanks a lot!

Your input:
Message:

CHAPTER 8 - TESTING

8.1 TEST CASES

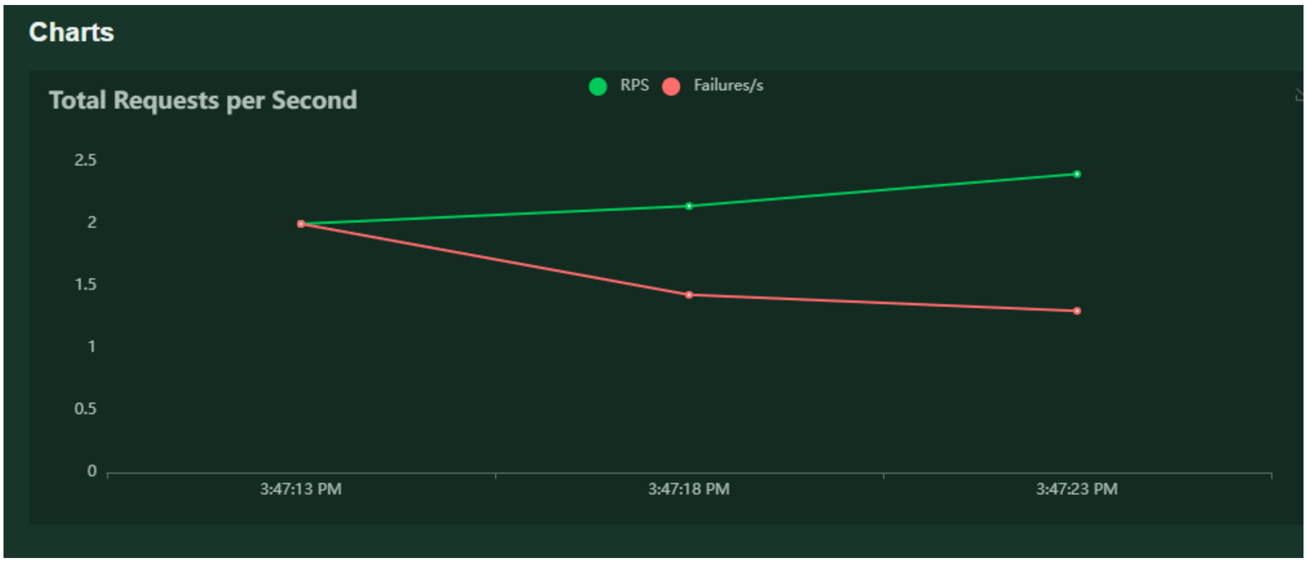
Test Case ID	Feature Type	Component	Test Scenario	Steps to execute
LoginPage_TC_OO1	Functional	Login page	Verify user is able to see the Login/Signup page when url is entered	1.Enter URL and click go 2.Verify login/Singup page displayed or not
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup page	1.Enter URL and click go 2.Verify login/Singup popup with below UI elements: a.email text box b.password text box c.Login button
LoginPage_TC_OO3	Functional	Login page	Verify user is able to log into application with Valid credentials	1.Enter URL(https://localhost:5001) and click go 2.Enter Valid email in Email text box 3.Enter valid password in password text box 4.Click on login button

LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(https://localhost:5001) and click go 2.Enter InValid email in Email text box 3.Enter valid password in password text box 4.Click on login button
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(https://localhost:5001) and click go 2.Enter Valid username/email in Email text box 3.Enter Invalid password in password text box 4.Click on login button
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(https://localhost:5001) and click go 2.Enter InValid email in Email text box 3.Enter Invalid password in password text box 4.Click on login button
Image_Upload_TC_006	UI	Image Upload Page	verify the UI elements in image upload page	1.Enter URL(https://localhost:5001) and login with credentials 2.Verify the following UI elements: a.input to upload image b.input(radio button) to select fruit or vegetable c.button to proceed to prediction d.button for feedback page
Image_Upload_TC_007	Functional	Image Upload page	verify prediction did not takes place after uploading wrong image or file and selecting one of the radio button	1.Upload wrong file stored in local device 2.Select fruit or Vegetable 3.Click proceed button

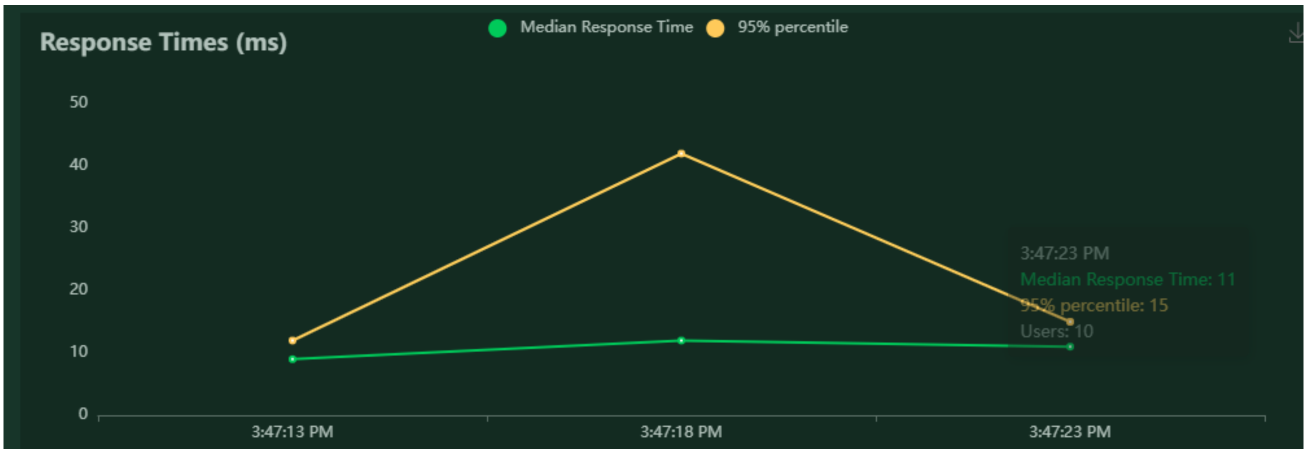
Image_Upload_TC_008	Functional	Image Upload page	verify prediction didn't not take place when image is not uploaded and selecting one of the radio button	1.Don't upload image from the dataset stored in local device 2.Select fruit or Vegetable 3.Click proceed button
Image_Upload_TC_009	Functional	Image Upload page	verify prediction takes place after uploading the image and selecting one of the radio button	1.Upload image from the dataset stored in local device 2.Select fruit or Vegetable 3.Click proceed button
Feedback_TC_010	UI	Feedback page	verify the UI elements in feedback page	1.Verify the following UI elements present in feedback page: a.input field to enter name b.input field to enter email c.input fied to enter feedback d.button to submit the feedback
Feedback_TC_011	Functional	Feedback page	verify user able to submit the feedback	1.click feedback button and navigate to feedback page 2.User enters his/her name,email and feedback 3.click on submit button after giving feedback

8.2 USER ACCEPTANCE TESTING

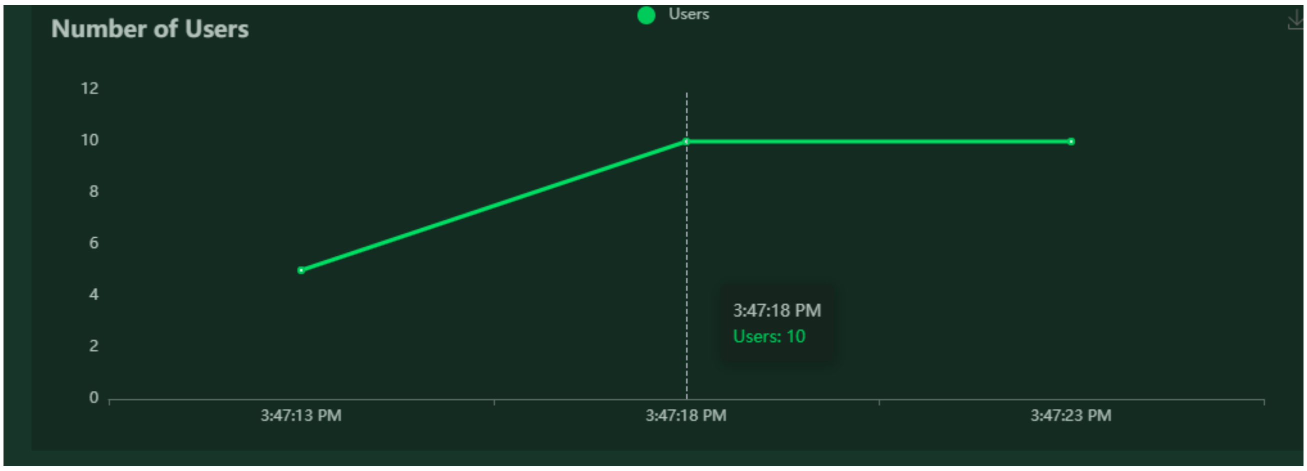
Charts:



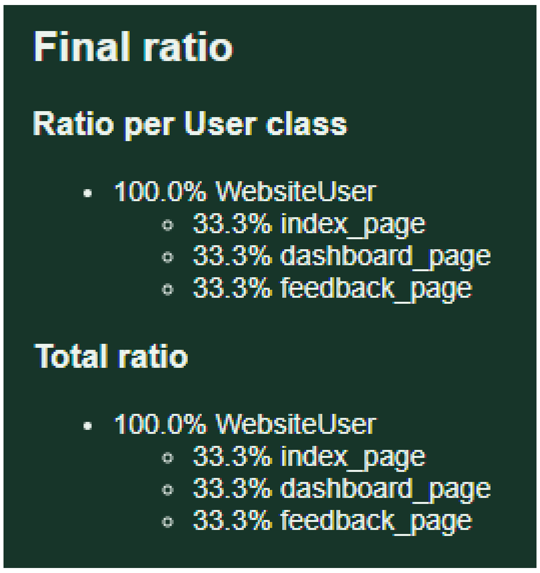
Response Times(ms)



Number of Users:



Final ratio:



Final ratio

Ratio per User class

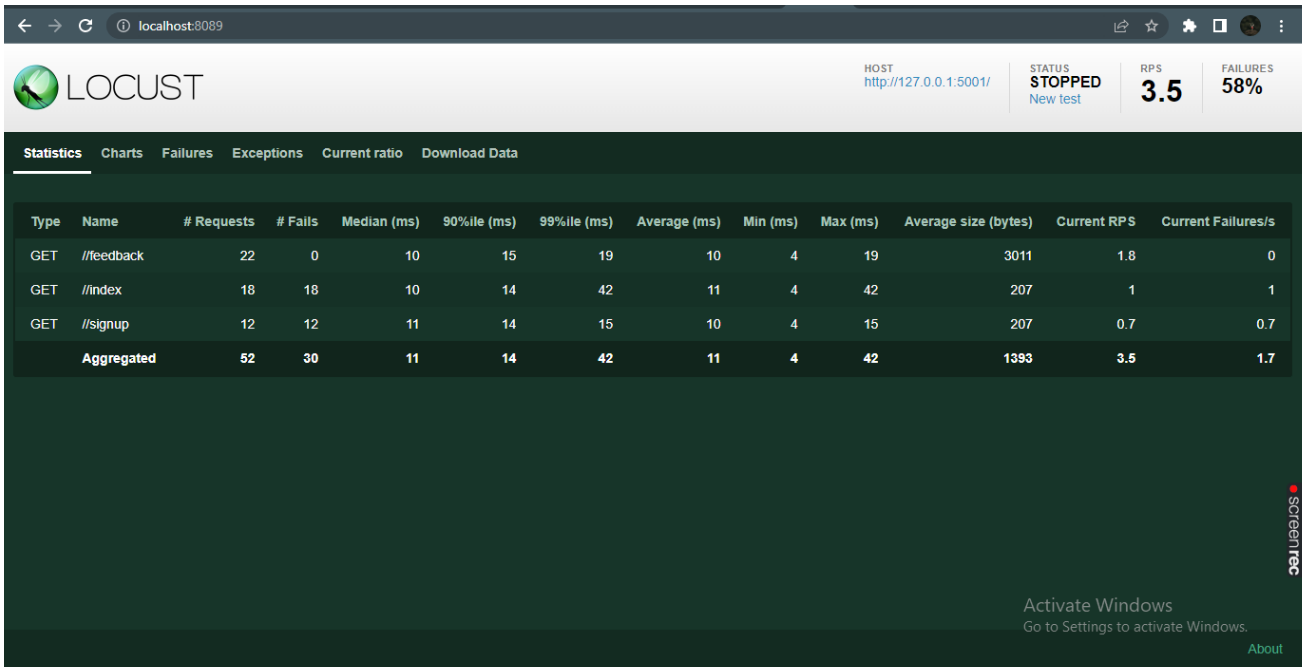
- 100.0% WebsiteUser
  - 33.3% index_page
  - 33.3% dashboard_page
  - 33.3% feedback_page

Total ratio

- 100.0% WebsiteUser
  - 33.3% index_page
  - 33.3% dashboard_page
  - 33.3% feedback_page

CHAPTER 9 - RESULTS

9.1 PERFORMANCE METRICS:



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	//feedback	22	0	10	15	19	10	4	19	3011	1.8	0
GET	//index	18	18	10	14	42	11	4	42	207	1	1
GET	//signup	12	12	11	14	15	10	4	15	207	0.7	0.7
	Aggregated	**52**	**30**	**11**	**14**	**42**	**11**	**4**	**42**	**1393**	**3.5**	**1.7**

CHAPTER 10 - ADVANTAGES & DISADVANTAGES

Advantages:

The system comes with a model to be precise and accurate in predicting crop yield and deliver the end user with proper recommendations about required fertilizer ratio based on atmospheric and soil parameters of the land which enhance to increase the crop yield and increase farmer revenue.   The prediction of crop yield based on location and proper implementation of algorithms have proved that the higher crop yield can be achieved. From above work I conclude that for soil classification Random Forest is good with accuracy 86.35% compare to Support Vector Machine. For crop yield prediction Support Vector Machine is good with accuracy 99.47% compare to Random Forest algorithm. The work can be extended further to add following functionality. Mobile application can be build to help farmers by uploading image of farms. Crop diseases detection using image processing in which user get pesticides based on disease images. Implement Smart Irrigation System for farms to get higher yield.

- Fertilizers have all nutrients required for plants growth.
- It is soluble and easily absorbed by plants.
- It enhances the metabolism of plants.
- It is easily available in the market.
- Highly needed for large production.

Disadvantages:

- Though the use of deep learning algorithms in disease prediction in agriculture crops reduces the time in determining the disease and in recommending suitable fertilizers, there are predictions of there being millions of unemployed field workers in the next decades primarily due to the impact of AI in the agriculture industry.
- Weather changes can affect the prediction process and make it slow or create some changes in it.

CHAPTER 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 he farmers to choose the right crop for their land and to give the suitable amount of fertilizer to produce the maximum yield. The Support Vector Machine algorithm helps to predict the crop the precisely based on the pre-processed crop data. This system will also help the new comers to choose the crop which will grow in their area and produce them a good profit. A decent amount of profit will attract more people towards the agriculture. Also, the crop growth is based on the climate conditions in the particular area and the seasonal monsoons happens now are unpredictable, hence it is easy for the farmers when the prediction result is also based on the climatic conditions. Live weather prediction will also help the users to predict the crop water needs and also it will help the farmers to decrease the crop damage due to the rain or drought .

The prediction of crop yield based on soil data and proper implementation of algorithms have proved that a higher crop yield can be achieved. From the above work, we conclude that for soil classification Random Forest is a suitable algorithm with an accuracy of 99.09% compare to Gaussian Naive Bayes. The work can be extended further to add the following functionality. Building a Website can be built to help farmers by uploading an image of farms. Crop diseases detection uses image processing in which users get pesticides based on disease images and Fertilizer prediction based on soil condition.

CHAPTER 12 - FUTURE SCOPE

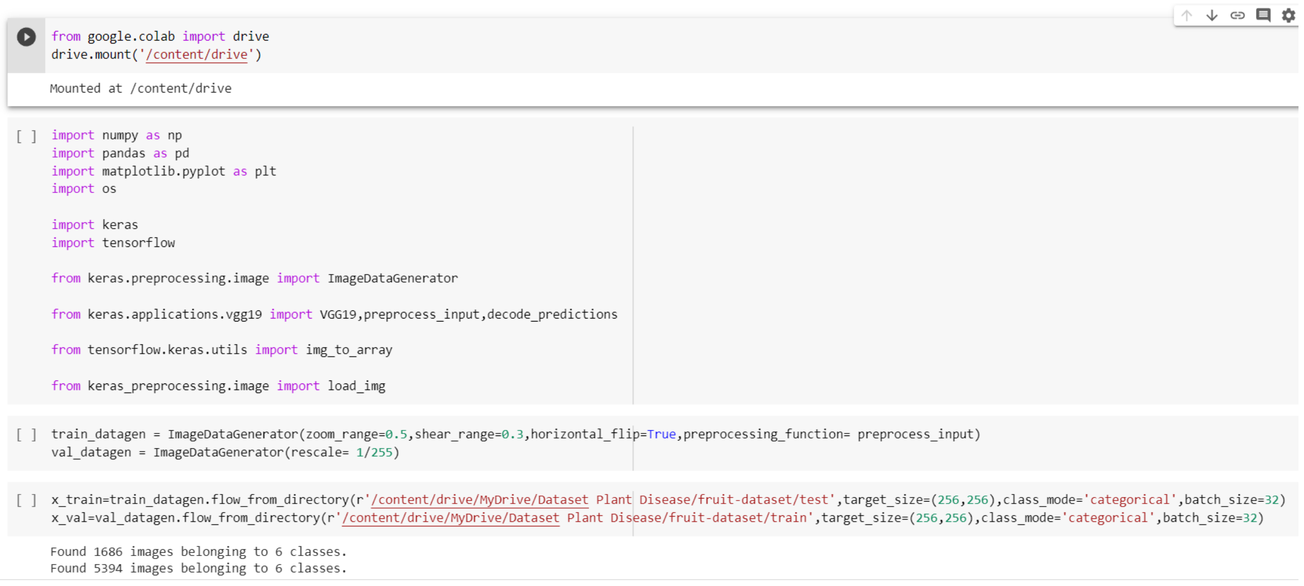
The future work is to implement Machine Learning Algorithms like Ensemble Classifiers to predict the crop yield and recommend the crop with appropriate fertilizer. In the existing system only soil characteristics were considered to provide crop recommendations. In the future work the climatic parameters will also be taken into account to provide crop recommendations. Also the method can be extended to include diverse varieties of crop to be cultivated and to analyse it's performance.

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.

CHAPTER 13 - APPENDIX

Source code:

1) Model Building for Fruit Disease Prediction

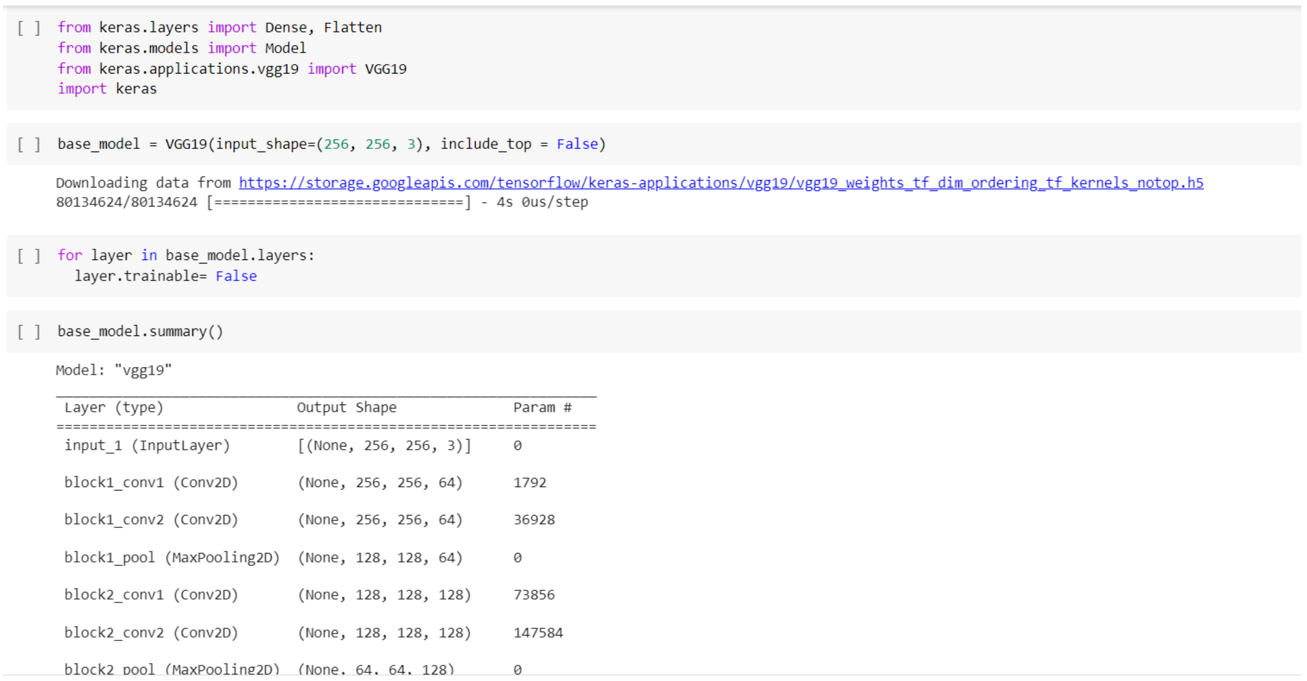


```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import os

    import keras
    import tensorflow

    from keras.preprocessing.image import ImageDataGenerator

    from keras.applications.vgg19 import VGG19,preprocess_input,decode_predictions

    from tensorflow.keras.utils import img_to_array

    from keras_preprocessing.image import load_img
```

```
[ ] train_datagen = ImageDataGenerator(zoom_range=0.5,shear_range=0.3,horizontal_flip=True,preprocessing_function= preprocess_input)
    val_datagen = ImageDataGenerator(rescale= 1/255)
```

```
[ ] x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset Plant Disease/fruit-dataset/test',target_size=(256,256),class_mode='categorical',batch_size=32)
    x_val=val_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset Plant Disease/fruit-dataset/train',target_size=(256,256),class_mode='categorical',batch_size=32)

    Found 1686 images belonging to 6 classes.
    Found 5394 images belonging to 6 classes.
```



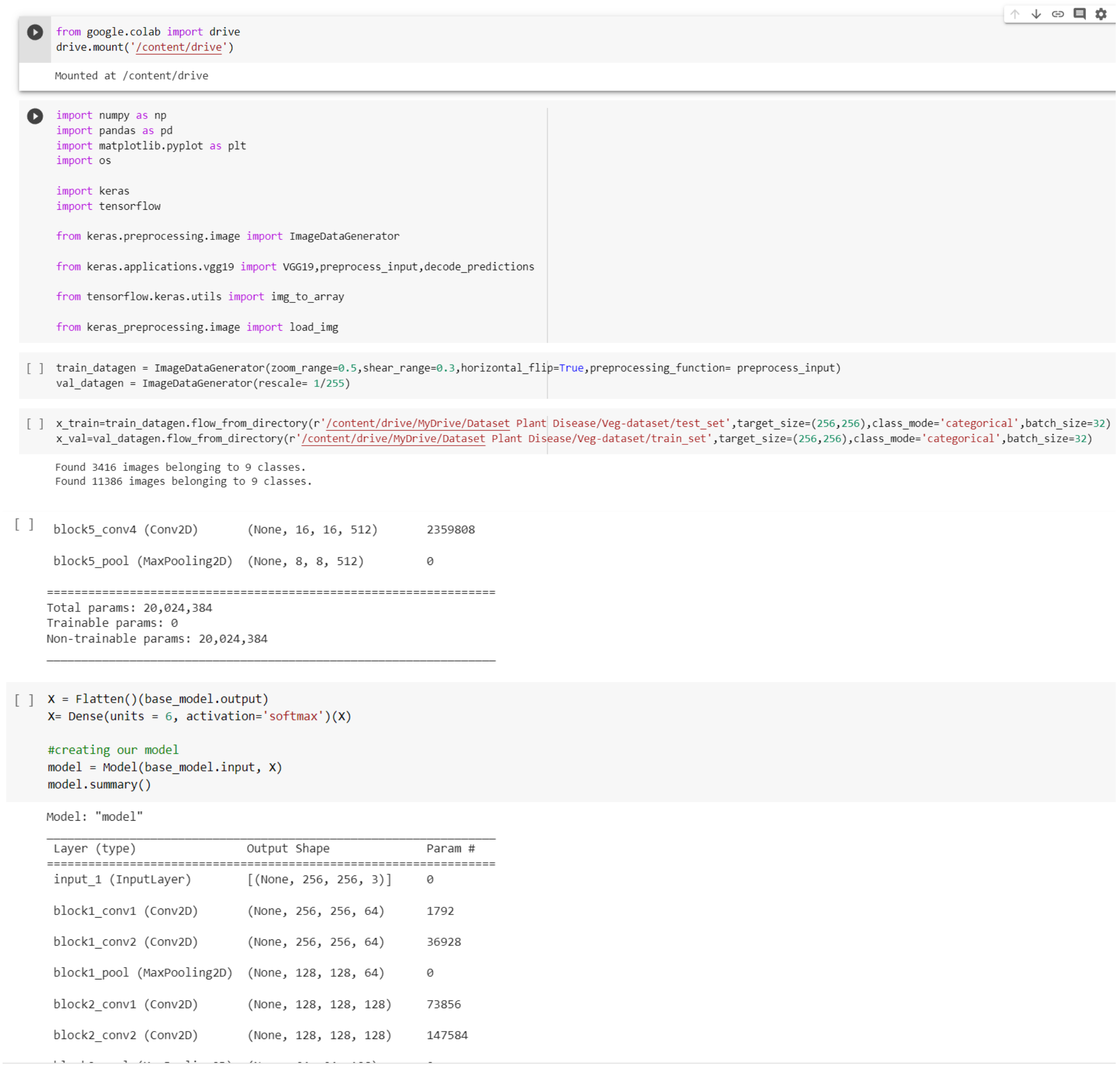
```
[ ] from keras.layers import Dense, Flatten
    from keras.models import Model
    from keras.applications.vgg19 import VGG19
    import keras
```

```
[ ] base_model = VGG19(input_shape=(256, 256, 3), include_top = False)

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
    80134624/80134624 [==============================] - 4s 0us/step
```

```
[ ] for layer in base_model.layers:
      layer.trainable= False
```

```
[ ] base_model.summary()

    Model: "vgg19"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     input_1 (InputLayer)        [(None, 256, 256, 3)]     0

     block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

     block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

     block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

     block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

     block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

     block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0
```

2) Model Building for Vegetable Disease Prediction



```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

import keras
import tensorflow

from keras.preprocessing.image import ImageDataGenerator

from keras.applications.vgg19 import VGG19,preprocess_input,decode_predictions

from tensorflow.keras.utils import img_to_array

from keras_preprocessing.image import load_img
```

```
[ ] train_datagen = ImageDataGenerator(zoom_range=0.5,shear_range=0.3,horizontal_flip=True,preprocessing_function= preprocess_input)
    val_datagen = ImageDataGenerator(rescale= 1/255)
```

```
[ ] x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset Plant Disease/Veg-dataset/test_set',target_size=(256,256),class_mode='categorical',batch_size=32)
    x_val=val_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset Plant Disease/Veg-dataset/train_set',target_size=(256,256),class_mode='categorical',batch_size=32)

    Found 3416 images belonging to 9 classes.
    Found 11386 images belonging to 9 classes.
```

```
[ ]  block5_conv4 (Conv2D)       (None, 16, 16, 512)       2359808

     block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

    =================================================================
    Total params: 20,024,384
    Trainable params: 0
    Non-trainable params: 20,024,384
    _________________________________________________________________
```

```
[ ] X = Flatten()(base_model.output)
    X= Dense(units = 6, activation='softmax')(X)

    #creating our model
    model = Model(base_model.input, X)
    model.summary()

    Model: "model"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     input_1 (InputLayer)        [(None, 256, 256, 3)]     0

     block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

     block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

     block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

     block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

     block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584
```

3) Login page - signup.html

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <link rel="stylesheet" href="../static/signup_style.css">
</head>
<body>
  <div class="top">
    <h1 class="logo">Fertilizers Recommendation System for Disease
```

```
Prediction</h1>
  </div>
</div>
    <div class="signupFrm">
    <form action="/dashboard" class="form" onsubmit="return validate()"
enctype="multipart/form-data" method="post">
      <h1 class="title">LOGIN</h1>
      <div class="inputContainer">
        <input type="text" class="input" placeholder="Enter email" id="email"
required>
        <label for="email" class="label">Email</label>
      </div>
      <div class="inputContainer">
        <input type="password" class="input" placeholder="Enter password"
id="psw" required>
        <label for="psw" class="label">Password</label>
      </div>
      <button type="submit" class="signupbtn" value="Sign up" >LOGIN</button>
    </form>
  </div>
    <div id="background-wrap">
    <div class="bubble x1"></div>
    <div class="bubble x2"></div>
    <div class="bubble x3"></div>
    <div class="bubble x4"></div>
    <div class="bubble x5"></div>
    <div class="bubble x6"></div>
    <div class="bubble x7"></div>
    <div class="bubble x8"></div>
    <div class="bubble x9"></div>
    <div class="bubble x10"></div>
</div>
<script>
  function validate(){
    var email=document.getElementById("email").value;
    var pword=document.getElementById("psw").value;
```

```
      if(email.match(/^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$/)){
        if(pword.length>8 &&pword.length<12){
          window.location.replace("upload.html")
        }
        else{
          alert("password small")
        }
      }
      else{
        alert("invalid email")}}
</script>
</body>
</html>
```

4) Dashboard - index.html

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
   <link rel="stylesheet" href="../static/index_style.css">
</head>
<body>
  <div class="top">
    <h1 class="logo">Fertilizers Recommendation System for Disease
Prediction</h1>
  </div>
  <div class="forms">
    <div class="form1">
      <form action="/predict" method="post" enctype="multipart/form-data"
class="form_upload" onsubmit="return validate()">
        <div class="uploadform">

          <div class="inputContainer">
            <h1 class="title">Image Uploading</h1>
            <label for="imageUpload" class="upload-label">Choose</label>
            <input type="file" name="files" id="imageUpload"
onchange="loadfile(event)" accept=".png, .jpg, .jpeg"><br>
            Select : <input type="radio" name="select_plant" id="fruit"
value="fruit">
```

```
            <label for="fruit">Fruit</label>
            <input type="radio" name="select_plant" id="veg"
value="vegetable">
            <label for="veg">Vegetable</label> <br>
            <button type="submit" class="signupbtn" value="Sign up"
class="signupbtn" >proceed</button>
          </div><br>
          <div>
            <img id="output">
            <h2 class="result_pred">{{result}}</h2>
          </div>
        </div>
      </form>
    </div><br>

    <div class="form2">
      <form action="/feedback" method="post" class="form_feedback">
        <button type="submit" >FEEDBACK</button>
      </form>
    </div>
  </div>

      <script>
        var loadfile=function(event){
            var image=document.getElementById('output');
            image.src=URL.createObjectURL(event.target.files[0]);
        };
    </script>
    <div id="background-wrap">
      <div class="bubble x1"></div>
      <div class="bubble x2"></div>
      <div class="bubble x3"></div>
      <div class="bubble x4"></div>
      <div class="bubble x5"></div>
      <div class="bubble x6"></div>
      <div class="bubble x7"></div>
      <div class="bubble x8"></div>
      <div class="bubble x9"></div>
```

```
        <div class="bubble x10"></div>
    </div>
</body>
</html>
```

5) Python Flask - app.py

```
import numpy as np
import os
import glob
from keras.applications.vgg19 import preprocess_input
from tensorflow.keras.utils import img_to_array
from tensorflow.keras.models import load_model
from keras_preprocessing.image import load_img
# Flask utils
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename

app = Flask(__name__,template_folder='template')

FRUIT_MODEL_PATH='fruit_model.h5'
model_fruit=load_model(FRUIT_MODEL_PATH)

VEG_MODEL_PATH='veg_model.h5'
model_veg=load_model(VEG_MODEL_PATH)

def model_fruitdata(img_path):
    img= load_img(img_path,target_size=(256, 256))
    i=img_to_array(img)
    im=preprocess_input(i)
    img= np.expand_dims(im, axis=0)
    prediction = np.argmax(model_fruit.predict(img))
    if prediction==0:
        prediction="The disease in the leaf was identified as Apple Black Rot.
Captna and sulphur products are labeled for control of both scab and black
rot. A scab spray program including these chemicals may help prevent the
frog-eye leaf spot of black spot, as well as the infection of fruit."
    elif prediction==1:
        prediction="The leaf is a fresh Apple Plant"
    elif prediction==2:
        prediction="The leaf is a fresh Corn(Maize) Plant"
    elif prediction==3:
        prediction="The disease in the leaf was identified as Northern Corn
Leaf Blight. In addition, the rate K60 is the one effective in control of
Northern Corn Leaf Blight in the field. However, there is a need for further
studies in the greenhouse. Thus, the availability of fertilizer at the K60 to
farmers in the endemic zones could help for sustainable management of
Northern Leaf Blight in maize. "
    elif prediction==4:
        prediction="The disease in the leaf was identified as Peach Bacterial
Spot. Compounds available for use on peach and nectarine for bacterial spot
include copper, oxytetracycline (Mycoshield and generic equivalents), and
syllit+captan; however, repeated applications are typically necessary for
even minimal disease control."
    else:
        prediction="The leaf is a fresh Peach PLant"
    return prediction

def model_vegdata(img_path):
    img= load_img(img_path,target_size=(256, 256))
    i=img_to_array(img)
    im=preprocess_input(i)
    img= np.expand_dims(im, axis=0)
    prediction = np.argmax(model_veg.predict(img))
    if prediction==0:
        prediction="The disease in the leaf was identified as Pepper Bell
Bacterial Spot.Copper sprays can be used to control bacterial leaf spot, but
they are not as effective when used alone on a continuous basis. Thus,
combining these sprays with a plant resistance inducer, such as Regalia or
Actigard, can provide good protection from the disease."
    elif prediction==1:
        prediction="The leaf is a Healthy Pepper Bell Plant"
    elif prediction==2:
        prediction="The disease in the leaf was identified as Potato Early
Blight. Mancozeb and chlorothalonil are perhaps the most frequently used
protectant fungicides for early blight management but provide insufficient
control under high disease pressure."
    elif prediction==3:
        prediction="The leaf is a Healthy Potato Plant."
    elif prediction==4:
        prediction="The disease in the leaf was identified as Potato Late
Blight. If there is some sign of blight and the potatoes are not mature, use
Dithane (mancozeb) MZ or you can also use Tattoo C or Acrobat MZ. Acrobat
used later in the season reduces late blight spores."
    elif prediction==5:
        prediction="The disease in the leaf was identified as Tomato Bacterial
Spot. A plant with bacterial spot cannot be cured. Remove symptomatic plants
from the field or greenhouse to prevent the spread of bacteria to healthy
plants. Burn, bury or hot compost the affected plants and DO NOT eat
symptomatic fruit."
    elif prediction==6:
        prediction="The disease in the leaf was identified as Tomato Late
Blight. Copper products can effectively control, or slow down, late blight
epidemics. Copper products have no activity. Therefore, they need to be
applied to all plant surfaces before infection (before symptoms are observed
in the field) and frequently so new foliage is protected as plants grow."
    elif prediction==7:
        prediction="The disease in the leaf was identified as Tomato Leaf
Mold. Applying fungicides when symptoms first appear can reduce the spread of
the leaf mold fungus significantly. Several fungicides are labeled for leaf
mold control on tomatoes and can provide good disease control if applied to
all the foliage of the plant, especially the lower surfaces of the leaves."
    else:
        prediction="The disease in the leaf was identified as Tomato Septoria
Leaf Spot. Most fungicides registered for use on tomatoes would effectively
control Septoria leaf spot. These include maneb, mancozeb, chlorothalonil,
and benomyl. Captan is not effective and zineb may be difficult to purchase."
    return prediction

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

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

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

@app.route('/predict', methods=['GET','POST'])
def upload():
    if request.method=='POST':
        f=request.files['files']
        fv=request.form['select_plant']

        if fv=="fruit":
            basepath=os.path.dirname(__file__)
file_path=os.path.join(basepath,'uploads',secure_filename(f.filename))
            f.save(file_path)
            prediction=model_fruitdata(file_path)
            res=prediction
            return
render_template('index.html',result='{}'.format(res),res=res)
        else:
            basepath=os.path.dirname(__file__)
file_path=os.path.join(basepath,'uploads',secure_filename(f.filename))
            f.save(file_path)
            prediction=model_vegdata(file_path)
            res=prediction
            return
render_template('index.html',result='{}'.format(res),res=res)

if __name__ == '__main__':
    app.run(port=5001,debug=True)
```

6) Feedback from the customer - feedback.html

```
<!DOCTYPE html>
<html>
<head>
  <title>FEEDBACK FORM | NLAIYA THIRAN</title>
  <link rel="stylesheet" href="../static/feedback_style.css">
  <script src="https://kit.fontawesome.com/67c66657c7.js"></script>
</head>
<body>
  <section></section>
  <div class="container">
    <h1>Give your feedback</h1>
    <form id="details">
      <div class="id">
      <input type="text" placeholder="Full name">
        <i class="far fa-user"></i>
      </div>
      <div class="id">
        <input type="email" name=""  placeholder="Email address">
        <i class="far fa-envelope"></i>
      </div>
      <textarea cols="15" rows="5" placeholder="Enter Your opinions here.."
id="user_input"></textarea>
      <input type="submit" onclick="showInput();" class="btn"><br/>
      <label style="color: white"><h3>Your input:</h3> </label>
      <p><h3 style="color: white">Message: <span id='display' style="color:
white" ></span></h3></p>
    </form>
  </div>
<br>
<br>
  <script language="JavaScript">
    function showInput() {
        document.getElementById('display').innerHTML =
document.getElementById("user_input").value;
        alert("your response has been stored");
    }
  </script>
</body>
</html>
```

Github & project demo link

GITHUB LINK: Team ID: PNT2022TMID22999

PROJECT DEMO LINK: Project Demo Link - PNT2022TMID22999