FERTILIZERS RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

USING ARTIFICIAL INTELLIGENCE

A PROJECT REPORT

Submitted by

Team ID: PNT2022TMID26773

Nandhini R	310519104078
Sowmiya G	310519104134
Senthil kumar M	310519104117
Velu N	310519104303

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

DHANALAKSHMI SRINIVASAN COLLEGE

OF ENGINEERING AND TECHNOLOGY

ANNA UNIVERSITY: CHENNAI-600025

November 2022

CONTENT

1. INTRODUCTION

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.

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.

PURPOSE

This project is used to test the fruits and vegetables samples and identify the different diseases. Also ,this project recommends fertilizers for predicted diseases Agriculture serves as a means of supplying food to a population that is always expanding, as well as a significant source of energy and a means of combating globalwarming.

Plant diseases are very important because they can have a negative impact on the quality and quantity of crops produced in agriculture. Early detection of plant diseases is crucial for their treatment and management. Typically, illnesses are identified using the naked eye technique. Experts who can recognise variations in leaf colour are involved in this process. This method requires a lot of work, takes a while, and is not appropriate for fields with a lot of space.. They forecast plant disease and recommend fertiliser for the damaged plants. This frequently involves a range of methods for assessing the qualities of the herbs that largely influence the plants. These complex systems that contain a large amount of datasets are forecasted using the neural network. By using artificial intelligence, complex manual systems' working models can be made simpler and more precise

2. LITERATURE SURVEY

EXISTING PROBLEM

Proposed a method for leaf disease detection and suggest fertilizers to cure leaf diseases. But the method involves less number of train and test sets which results in poor accuracy. Also proposed a simple prediction method for soil based fertilizer recommend ratio system for predicted crop diseases.
This method gives less accuracy and prediction. proposed an IoT based system for leaf disease detection and fertilizer recommendation which is based on Machine Learning techniques yields less 80 percentage accuracies.

REFERENCES

[1]. R Indumathi.; N Saagari.; V Thejuswini.; R Swarnareka.," Leaf Disease Detectionand Fertilizer Suggestion", IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), 29-30 March 2019, DOI: 10.1109/ICSCAN.2019.8878781.

[2]. P. Pandi Selvi, P. Poornima, "Soil Based Fertilizer Recommendation System for Crop Disease Prediction System", International Journal of Engineering Trends and Applications (IJETA) – Volume 8 Issue 2, Mar-Apr 2021 .

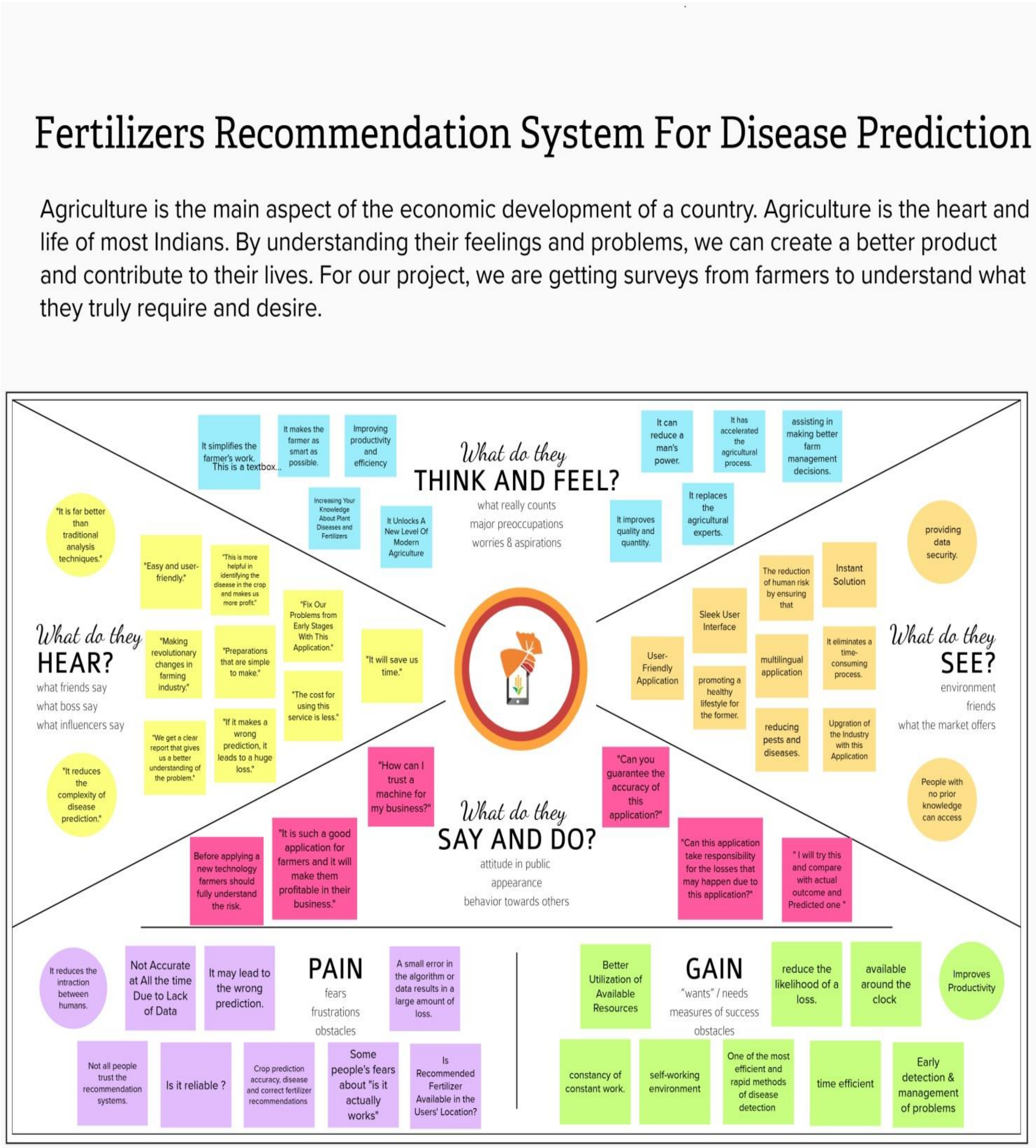
[3]. H Shiva reddy, Ganesh hedge, Prof. DR Chinnaya3, "IoT based Leaf Disease Detection and Fertilizer Recommendation", International Research Journal of Engineering andTechnology (IRJET), Volume: 06 Issue: 11, Nov 2019, e-ISSN: 2395-0056.

PROBLEM STATEMENT DEFINITION

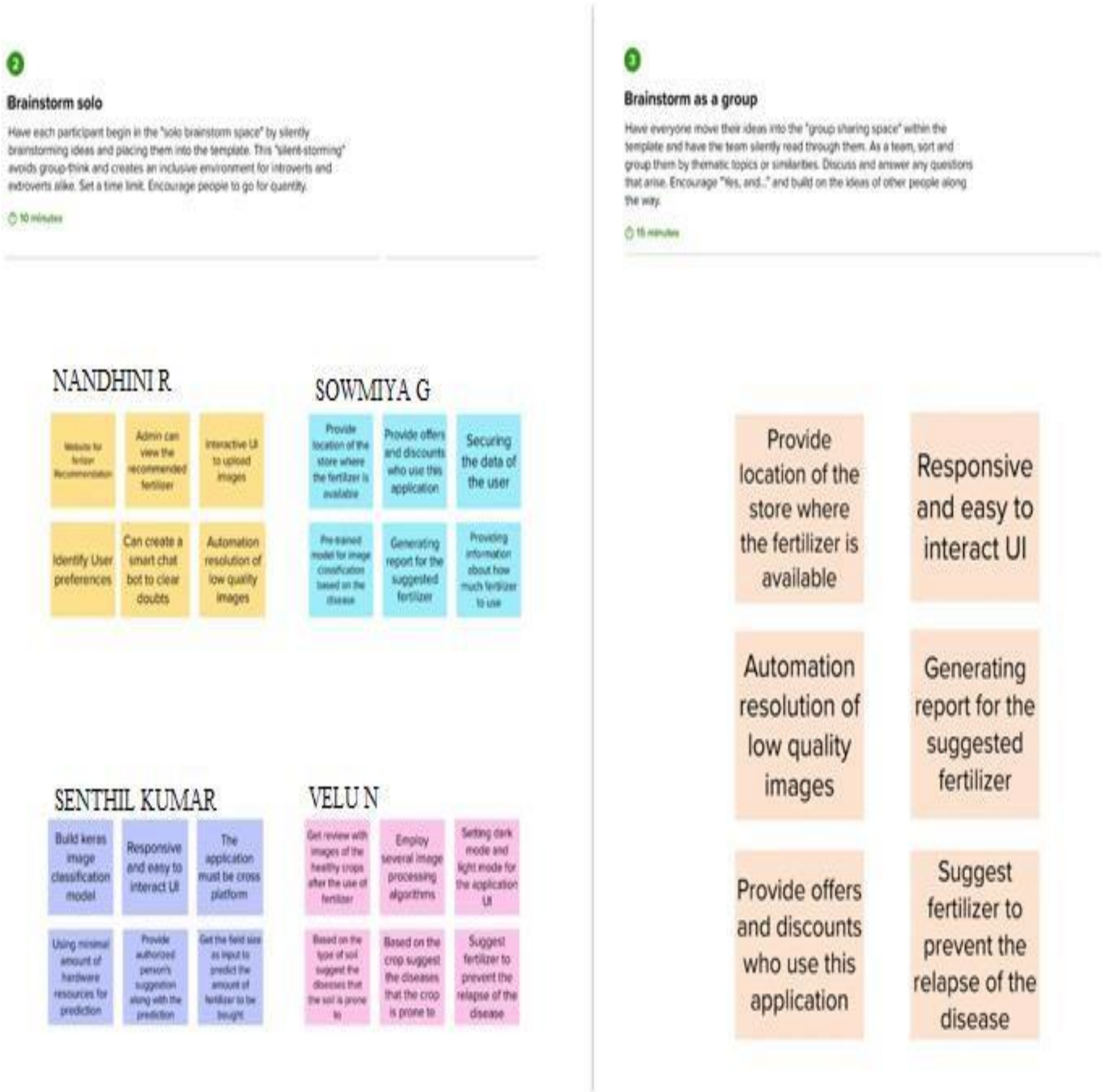
- Preprocess the images.
- Applying the CNN algorithm to the dataset.
- How deep neural networks detect the disease.
- You will be able to know how to find the accuracy of the model.
- You will be able to build web applications using the Flask framework.

3. IDEATION &PROPOSED SOLUTION

EMPATHY MAP CANVAS



IDEATION & BRAINSTROMING



PROBLEM STATEMENTS

Problem Statement (PS)	I am	I’m trying to	But	Because	Which makes mefeel
PS-1	a farmer	grow and harvest healthy crops	my crops are affected by some disease	of not using the needed fertilizers	sad and disappointed to seethe diseased crops
PS-2	a gardener	grow some vegetable plants in the home	there are spots appearing on the leaves	of not maintaining and providingthe needed nutrients	not productive and not useful
PS-3	a nursery manager	grow and sell the plant saplings	the baby plants are not growing after some stage	not doing the needed tasks in order to maintain the growth	unprofitable in the future

PROPOSED SOLUTION

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal 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. So, based on above situation, how to help the farmers to choose the right fertilizer based upon the data given by them to promote productivity as well as eradicate the crop disease? What kind of model is suitable for this task? These are the main problems which are needed to be considered Here in this scenario.
2.	Idea/Solution description	The proposed idea is based on the usage of CNN primarily. The images with respective class names are made to train using CNN to classify the disease present, then based upon The results, the required fertilizer will be suggested to the user via the web interface.

3.	Novelty/Uniqueness	The novelty of the project lies on the parameters used on the CNN model. Anotheraspect is the usage of both sample and Real time data during model training.
4.	Social Impact/ Customer Satisfaction	The proposed idea will impact the farmers a lot such that the right kind of fertilizer will be suggested to eradicate the diseases presentin crops as well as increase the production By promoting their growth. The farmer can have insights about the fertilizers as well.
5.	Business Model(Revenue Model)	The proposed model can be deployed both inweb as well as containerized model (docker,k8s)for enterprise usage. The Web app will be made for free with certain limitations to all users. The enterprise app could be providedto businesses as standalone app which they could run on their own servers and then provide services to consumers for specific amount.
6.	Scalability of the Solution	At present, the model will be trained with certain crop diseases and provide suggestion son the fertilizer to use on the crops. By time, the model will be trained withmore images To classify more crop diseases to predict andsuggest fertilizers in efficient way.

4. REQUIRMENT ANALYSIS

FUNCTIONAL REQUIREMENTS

Functional Requirements:

Following are the functional requirements of the proposed solution.

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	User Login	Login with user name Login with password
FR-4	Profile update	Update the user credentials Update the Contact details
FR-5	Uploading Images	Upload the image of the affected Crop
FR-6	Image processing	Upload image will be processed to predict the disease in leaf
FR-7	Recommendation	User will request the fertilizer Get the fertilizer recommendations as per the disease
FR-8	Ratings and Reviews	Share their Experiences Give the Feedback

NON-FUNCTIONAL REQUIREMENTS

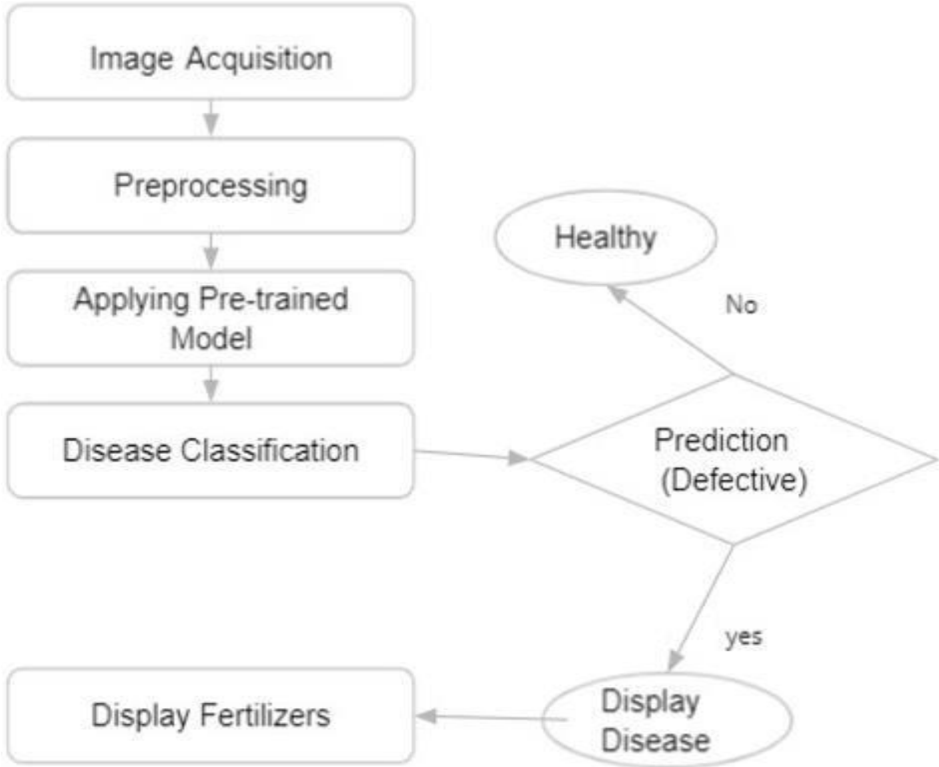
Non-functional Requirements:

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

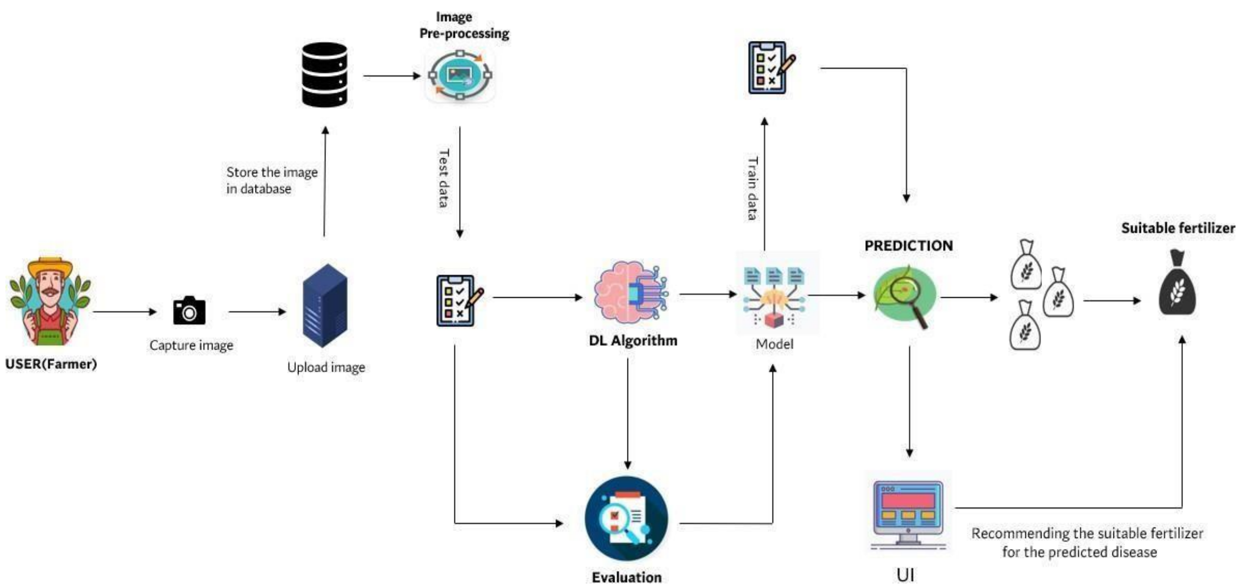
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	Usability	It can be used on both website and mobile browsers so that the usability of this application is very efficient.
NFR-2	Security	The information belongs to user and about leaf is highly secured.
NFR-3	Reliability	The leaf quality is necessary to predict the disease in the leaf.
NFR-4	Performance	The performance of application is very fast and efficient, based on the prediction of the disease.
NFR-5	Availability	It will predict any type of new disease by learning from the available data and predict the disease accurately.
NFR-6	Scalability	It can be accessed by a greater number of users at the same time without any performance issues.

5. PROJECT DESIGN

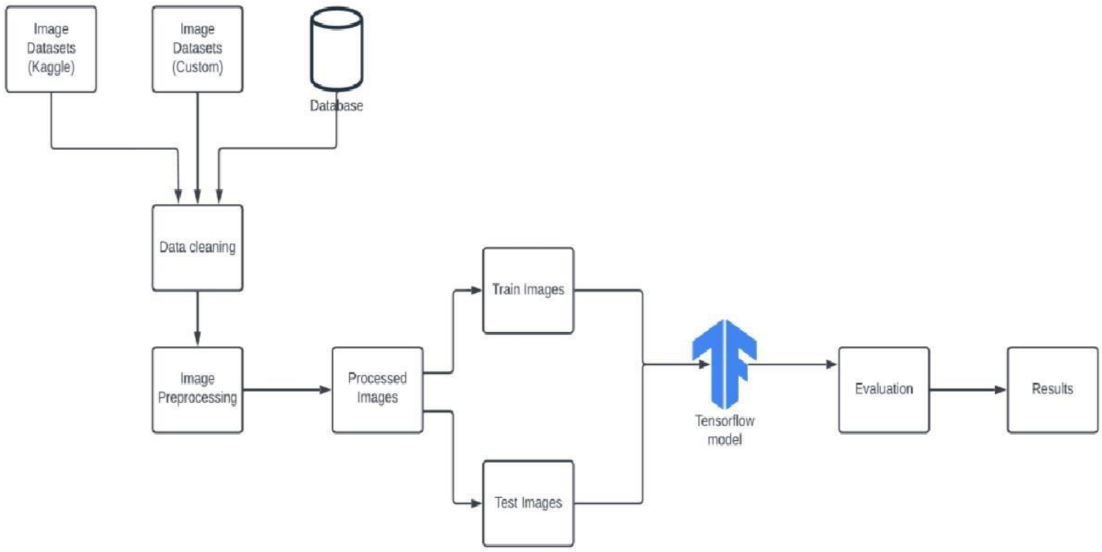
DATA FLOW DIAGRAMS



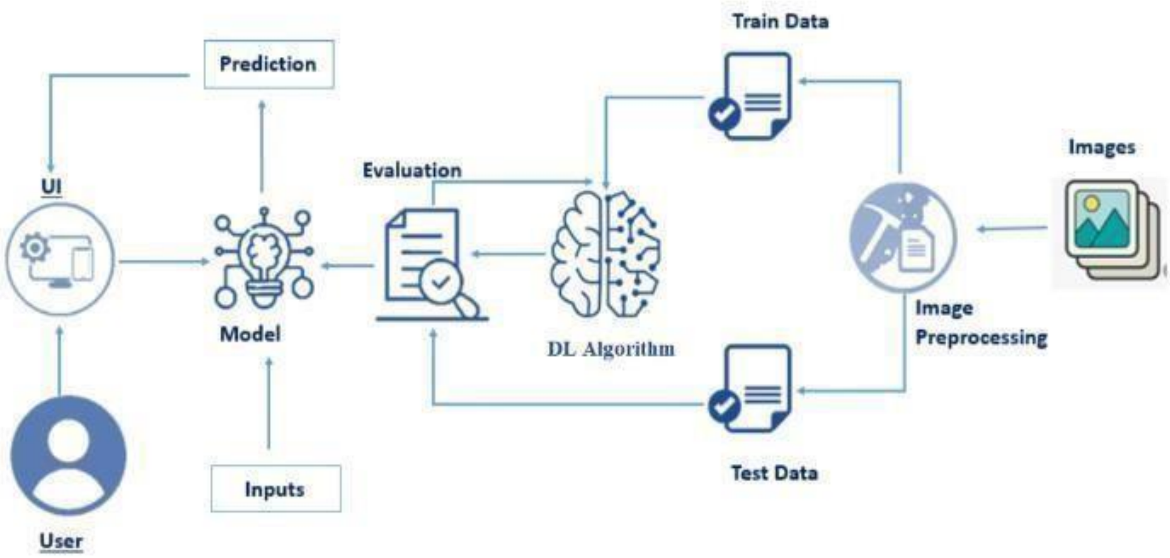
SOLUTION ARCHITECTURE



Plant crop disease is anticipated, and appropriate fertilizer is advised for a higher yield. The diseased plant photos are acquired and preprocessed in comparison to the dataset of diseased plants. The photos are processed using a Deep Learning algorithm, which is subsequently tested. A model is then created based on the evaluations, trained using a variety of inputs, and predictions are presented to the users, aiding in the fertilizer recommendation process. The inclusion of the Convolutional layers in the classification and processing of the images further aids in



TECHNICAL ARCHITECTURE



USER STORIES

A digital camera or similar devices are used to take images of different types, and then those are used to identify the affected area in leaves. Then different types of image-processing techniques are applied to them, the process those images, to get different and useful features needed for the purpose of analyzing later-Plant leaf disease identification is especially needed to predict both the quality and quantity of the First segmentation step primarily based on a mild polygonal leaf model is first achieved and later used toguide the evolution of an energetic contour. Combining global shape descriptors given by the polygonal model with local curvature

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
Customer (Mobile user)	Download the database	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
	Register	USN-2	As a user, I can register for the application by entering my email,password, and confirming my password.	I can receive confirmati on email & click confirm	High
	Login	USN-3	As a user, I will receive confirmation email once I have registered for the application	I can register & access the dashboard With Login	Low
	Upload the image	USN-4	As a user, I must upload the imageto identify the disease		Medium
Customer (Webuser)	The functional requirements Are same asmobile user	USN-5	Same as mobile user can fill thedetail asked here	Same as mobile user can register the details	High(when compare dto mobile users)
Administrat or		USN-6	As the administrator I predict theoutput	Show the result	High

6. PROJECT PLANNING & SCHEDULING

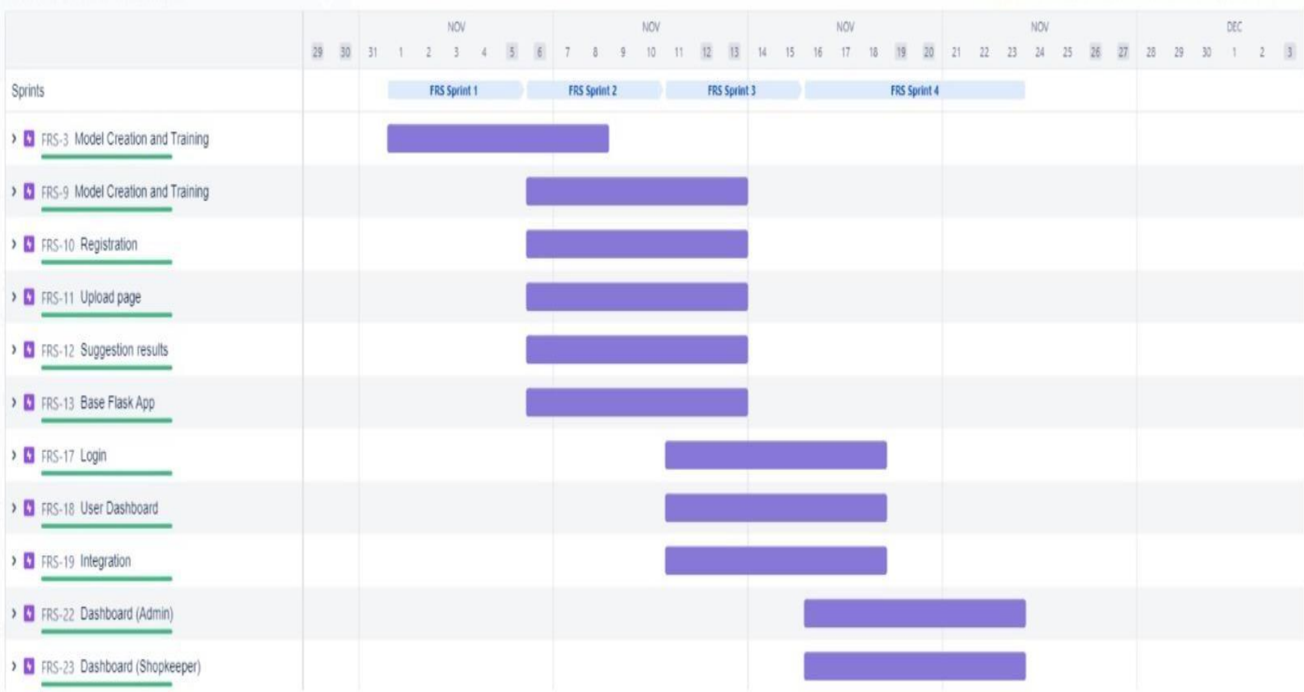
SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	Collecting plant disease dataset	2	High	Nandhini R
Sprint-1		USN-2	Labeling the dataset according to class	1	High	Sowmiya G
Sprint-2	Testing training and creating a model	USN-3	Start initiating the model	2	Low	Senthilkumar M
Sprint-1		USN-4	Adding different layers of CNN(convolution, pooling dense, flat ten)	2	Medium	Velu N
Sprint-1		USN-5	Training the data	1	High	Sowmiya G, Nandhini R
Sprint-3			Testing the data	1	Medium	Senthilkumar M, Velu N
Sprint-3	Flask and framework design		Creating backend framework with flask	2	High	Nandhini R, SowmiyaG, Senthilkumar M, Velu N
Sprint-4			Predicting disease and recommend fertilizer	2	High	Nandhini R, Sowmiya G, Senthilkumar M, Velu N

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	6Days	24Oct2022	29Oct2022	10	05 NOV 2022
Sprint-2	20	6Days	31Oct2022	05Nov2022	15	10 Nov 2022
Sprint-3	20	6Days	07Nov2022	12Nov2022	15	15 Nov 2022
Sprint-4	20	6Days	14Nov2022	19Nov2022	10	20 Nov 2022

REPORTS FROM JIRA



6. CODING & SOLUTIONING

FEATURE 1

The application's registration page is created. User registration is carried out if the user hasn't already done so. Enough work was put into making this process seamless. If the user has registered, he can now log in directly. Email address, name, and password were required for registration.

MODEL BUILDING

1. Import The Libraries

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

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

2. Initializing The Model

Keras has 2 ways to define a neural network:

- Sequential
- Function API

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

```
model=Sequential()
```

3. ADD CNN Layers

We will be adding three layers for CNN

- Convolution layer
- Pooling layer
- Flattening layer

Add Convolution Layer

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution2D class is used. It takes a number of feature detectors, feature detector size, expected input shape of the image, and activation function as arguments. This **1 9** layer applies feature detectors on the input image and returns a feature map (features from the image).
Activation Function: These are the functions that help us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.

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

Add the pooling layer

Max Pooling selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map. After the convolution layer, a pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. Efficient size of the pooling matrix is (2,2). It returns the pooled feature maps. (Note: Any number of convolution layers, pooling and dropout layers can be added)

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

Add the flatten layer

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

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

4. Add Dense Layers

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

Dense layers

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

Adding Hidden layers

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

- ● init is the weight initialization; function which sets all the weights and biases of a network to values suitable as a starting point for training.
- ● units/ output_dim, which denote is the number of neurons in the hidden layer.
- ● The activation function basically decides to deactivate neurons or activate them to get the desired output. It also performs a nonlinear transformation on the input to get better results on a complex neural network.
- ● You can add many hidden layers, in our project we are added two hidden layers. The 1st hidden layer with 40 neurons and 2nd hidden layer with 20neurons.

Adding the output layer

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

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

5. Train And Save The Model

Compile the model

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

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

Fit and save the model

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

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

Accuracy, Loss: Loss value implies how poorly or well a model behaves after each iteration of optimization. An accuracy metric is used to measure the algorithm's performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage. The weights are to be saved for future use. The weights are saved in as .h5 file using save().

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

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

6. Test The Model

The model is to be tested with different images to know if it is working correctly.

Import the packages and load the saved model

Import the required libraries

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

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

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

Load the test image, pre-process it and predict Pre-processing the image includes converting the image to array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.

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

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

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

```
pred
```

```
[1]
```

The predicted class is 1.

FEATURE 2

Python Code

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

To Build a flask application do the following:

Step 1: Load the required packages
Step 2: Initialize the flask app and load the model
Step 3: Configure the home page
Step 4: Pre-process the frame and run

The trained machine learning model can predict the output from an image that is uploaded, and the nutrition facts are also displayed on the same page. The model's accuracy was determined to be 95%, and when it was trained on the IBM cloud, it reached 100%.

7. TESTING

TEST CASES

The test cases include invalid email and unrecognizable images. For the image part, a text fileor other format files were uploaded as a corner case.

USER ACCEPTANCE TESTING

Acceptance testing is performed on a collection of business functions in a Production environment and after the completion of functional testing. This is the final Stage in the testing process before the system is accepted for operational use. This testing should be done with original data and with the presence of the users. This test confirms the systemready for production.

Purpose of 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).

Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
Leaf spots	10	4	2	3	19
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	119

8. RESULTS

PERFORMANCE METRICS

Model Performance Testing:

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

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Totalparams:896 Trainableparams:896 Non-trainableparams:0	model.summary() Model: "sequential" Layer (type) Output Shape Param # conv2d (Conv2D) (None, 126, 126, 32) 896 max_pooling2d (MaxPooling2D (None, 63, 63, 32) 0) flatten (Flatten) (None, 127008) 0 Total params: 896 Trainable params: 896 Non-trainable params: 0
2.	Accuracy	Training Accuracy– 96.55 ValidationAccuracy-97.45	Epoch 1/10 225/225 [==============================] - 96s 425ms/step - loss: 1.1095 - accuracy: 0.7829 - val_loss: 0.3157 - val_accuracy: 0.8861 Epoch 2/10 225/225 [==============================] - 88s 393ms/step - loss: 0.2825 - accuracy: 0.9042 - val_loss: 0.3015 - val_accuracy: 0.9075 Epoch 3/10 225/225 [==============================] - 85s 375ms/step - loss: 0.2032 - accuracy: 0.9303 - val_loss: 0.2203 - val_accuracy: 0.9288 Epoch 4/10 225/225 [==============================] - 84s 374ms/step - loss: 0.1576 - accuracy: 0.9463 - val_loss: 0.2424 - val_accuracy: 0.9164 Epoch 5/10 225/225 [==============================] - 84s 372ms/step - loss: 0.1719 - accuracy: 0.9389 - val_loss: 0.1330 - val_accuracy: 0.9632 Epoch 6/10 225/225 [==============================] - 85s 376ms/step - loss: 0.1240 - accuracy: 0.9580 - val_loss: 0.1340 - val_accuracy: 0.9573 Epoch 7/10 225/225 [==============================] - 87s 388ms/step - loss: 0.1235 - accuracy: 0.9591 - val_loss: 0.1638 - val_accuracy: 0.9478 Epoch 8/10 225/225 [==============================] - 83s 371ms/step - loss: 0.1012 - accuracy: 0.9643 - val_loss: 0.1468 - val_accuracy: 0.9561 Epoch 9/10 225/225 [==============================] - 83s 367ms/step - loss: 0.0967 - accuracy: 0.9655 - val_loss: 0.1412 - val_accuracy: 0.9531 Epoch 10/10 225/225 [==============================] - 83s 369ms/step - loss: 0.0954 - accuracy: 0.9655 - val_loss: 0.0905 - val_accuracy: 0.9745

Model Summary

```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 flatten (Flatten)           (None, 127008)            0

=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

Accuracy

```
l.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)

sers\Sree Ram\AppData\Local\Temp\ipykernel_13228\1582812018.py:1: UserWarning: `Model.fit_generator` is deprecated and will
emoved in a future version. Please use `Model.fit`, which supports generators.
del.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)

1 1/10
225 [==============================] - 96s 425ms/step - loss: 1.1095 - accuracy: 0.7829 - val_loss: 0.3157 - val_accuracy:
51
1 2/10
225 [==============================] - 88s 393ms/step - loss: 0.2825 - accuracy: 0.9042 - val_loss: 0.3015 - val_accuracy:
75
1 3/10
225 [==============================] - 85s 375ms/step - loss: 0.2032 - accuracy: 0.9303 - val_loss: 0.2203 - val_accuracy:
38
1 4/10
225 [==============================] - 84s 374ms/step - loss: 0.1576 - accuracy: 0.9463 - val_loss: 0.2424 - val_accuracy:
54
1 5/10
225 [==============================] - 84s 372ms/step - loss: 0.1719 - accuracy: 0.9389 - val_loss: 0.1330 - val_accuracy:
32
1 6/10
225 [==============================] - 85s 376ms/step - loss: 0.1240 - accuracy: 0.9580 - val_loss: 0.1340 - val_accuracy:
73
1 7/10
225 [==============================] - 87s 388ms/step - loss: 0.1235 - accuracy: 0.9591 - val_loss: 0.1638 - val_accuracy:
78
1 8/10
225 [==============================] - 83s 371ms/step - loss: 0.1012 - accuracy: 0.9643 - val_loss: 0.1468 - val_accuracy:
51
1 9/10
225 [==============================] - 83s 367ms/step - loss: 0.0967 - accuracy: 0.9655 - val_loss: 0.1412 - val_accuracy:
31
1 10/10
225 [==============================] - 83s 369ms/step - loss: 0.0954 - accuracy: 0.9655 - val_loss: 0.0905 - val_accuracy:
45
```

9. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- The suggested model yields extremely high classification accuracy
- It can train and test on very large datasets.
- It can resize very high-quality images within itself.

DISADVANTAGES:

- Prediction is limited to few plants as we havent trained all the plants.
- The proposed model is computationally expensive to train and test.
- The neural network architecture used in this project work is highly complex.

10. CONCLUSION

The model here involves classifying images from datasets of fruits and vegetables. The number of epochs was increased to boost categorization accuracy. Different classification accuracies are obtained for different batch sizes. The accuracies are increased by adding more convolution layers. The accuracy of classification is also increased by adjusting the number of dense layers. The accuracies are different while varying the size of the train and test datasets.

11. FUTURE SCOPE

- The model that is being provided in this project work can be expanded to recognise images. Using python to exe software, the complete model may be turned into application software.

- As of now we have just built the web application which apparently takes the input as an image and then predict the out in the near future we can develop an application which computer vision and AI techniques to predict the infection once you keep the camera near the plant or leaf this could make our project even more usable

- 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 vegetables and fruits.

12. APPENDIX

SAMPLE SOURCE CODE:-

```
import tensorflow as tf
import  time import numpy as np import os

start = time.time()
#try:
# Total iterations
final_iter = 1000

# Assign the batch value
batch_size = 20

# 20% of the data will automatically be used for validation
validation_size = 0.2
img_size = 128
num_channels = 3 train_path = r'data\Train'

# Prepare input data
if not os.path.exists(train_path):
print("No such directory") raise Exception
classes = os.listdir(train_path) num_classes = len(classes)

# We shall load all the training and validation images and labels into
memory using openCV and use that during training
data = dataset.read_train_sets(train_path, img_size, classes,
validation_size=validation_size)

# Display the stats
print("Complete reading input data. Will Now print a snippet of it")
print("Number of files in Training-
set:\t\t{}".format(len(data.train.labels))) print("Number of files in
Validation-set:\t{}".format(len(data.valid.labels))) session =
tf.compat.v1.Session()
x = tf.compat.v1.placeholder(tf.float32, shape=[None, img_size, img_size,
num_channels], name='x')

## labels
y_true = tf.compat.v1.placeholder(tf.float32, shape=[None, num_classes],
name='y_true')
y_true_cls = tf.argmax(y_true, dimension=1)

##Network graph params
     filter_size_conv1
     num_filters_conv1

     filter_size_conv2
     num_filters_conv2
```

```
filter_size_conv3
num_filters_conv3


fc_layer_size = 128

def create_weights(shape):
return tf.Variable(tf.random.truncated_normal(shape, stddev=0.05))


def create_biases(size):
return tf.Variable(tf.constant(0.05, shape=[size]))

def make_generator_model(input,num_input_channels,conv_filter_size,
num_filters):
## We shall define the weights that will be trained using create_weights
function. weights = create_weights(shape=[conv_filter_size,
conv_filter_size, num_input_channels, num_filters])
## We create biases using the create_biases function. These are also
trained.
biases = create_biases(num_filters)

## Creating the convolutional layer layer = tf.nn.conv2d(input=input,
filter=weights,
strides=[1, 1, 1, 1], padding='SAME')

layer += biases

## We shall be using max-pooling. layer = tf.nn.max_pool(value=layer,
ksize=[1, 2, 2, 1],
strides=[1, 2, 2, 1], padding='SAME')
## Output of pooling is fed to Relu which is the activation function for
us.
layer = tf.nn.relu(layer)

return layer


# Function to create a Flatten Layer
def create_flatten_layer(layer):
# We know that the shape of the layer will be [batch_size img_size
img_size num_channels]
# But let's get it from the previous layer.
layer_shape = layer.get_shape()

## Number of features will be img_height * img_width* num_channels. But
we shall calculate it in place of hard-coding it.
num_features = layer_shape[1:4].num_elements()

## Now, we Flatten the layer so we shall have to reshape to num_features
layer = tf.reshape(layer, [-1, num_features])
```

```
return layer


# Function to create a Fully - Connected Layer
def create_fc_layer(input,num_inputs, num_outputs, use_relu=True):
# Let's define trainable weights and biases.
weights = create_weights(shape=[num_inputs, num_outputs]) biases =
create_biases(num_outputs)

# Fully connected layer takes input x and produces wx+b.Since, these are
matrices, we use matmul function in Tensorflow
layer = tf.matmul(input, weights) + biases
if use_relu:
layer = tf.nn.relu(layer)

return layer


# Create all the layers
layer_conv1 = make_generator_model(input=x,
num_input_channels=num_channels, conv_filter_size=filter_size_conv1,
num_filters=num_filters_conv1)
layer_conv2 = make_generator_model(input=layer_conv1,
num_input_channels=num_filters_conv1, conv_filter_size=filter_size_conv2,
num_filters=num_filters_conv2)

layer_conv3 = make_generator_model(input=layer_conv2,
num_input_channels=num_filters_conv2, conv_filter_size=filter_size_conv3,
num_filters=num_filters_conv3)

layer_flat = create_flatten_layer(layer_conv3)

layer_fc1 = create_fc_layer(input=layer_flat,
num_inputs=layer_flat.get_shape()[1:4].num_elements(),
num_outputs=fc_layer_size,
use_relu=True)

layer_fc2 = create_fc_layer(input=layer_fc1, num_inputs=fc_layer_size,
num_outputs=num_classes,
use_relu=False)

y_pred = tf.nn.softmax(layer_fc2, name='y_pred')

y_pred_cls = tf.argmax(y_pred, dimension=1)
session.run(tf.compat.v1.global_variables_initializer())
cross_entropy =
tf.nn.softmax_cross_entropy_with_logits_v2(logits=layer_fc2,
labels=y_true)
cost = tf.reduce_mean(cross_entropy)
```

```
optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=1e-
4).minimize(cost) correct_prediction = tf.equal(y_pred_cls, y_true_cls)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

session.run(tf.compat.v1.global_variables_initializer())


# Display all stats for every epoch
def show_progress(epoch, feed_dict_train, feed_dict_validate,
val_losstotal_epochs):
acc = session.run(accuracy, feed_dict=feed_dict_train) val_acc =
session.run(accuracy, feed_dict=feed_dict_validate)
msg = "Training Epoch {0}/{4} --- Training Accuracy: {1:>6.1%},
ValidationAccuracy: {2:>6.1%},  Validation Loss: {3:.3f}"
print(msg.format(epoch + 1, acc, val_acc, val_loss, total_epochs))


total_iterations = 0

saver = tf.compat.v1.train.Saver() print("")


# Training Function
def train(num_iteration):
global total_iterations

for i in range(total_iterations,
total_iterations + num_iteration):

x_batch, y_true_batch, _, cls_batch = data.train.next_batch(batch_size)
x_valid_batch, y_valid_batch, _, valid_cls_batch =
data.valid.next_batch(batch_size)

feed_dict_tr = {x: x_batch,
y_true: y_true_batch} feed_dict_val = {x: x_valid_batch,
y_true: y_valid_batch} session.run(optimizer, feed_dict=feed_dict_tr)

if i % int(data.train.num_examples / batch_size) == 0:
val_loss = session.run(cost, feed_dict=feed_dict_val) epoch = int(i /
int(data.train.num_examples / batch_size))
# print(data.train.num_examples)
# print(batch_size)
# print(int(data.train.num_examples/batch_size)) # print(i)

total_epochs = int(num_iteration / int(data.train.num_examples /
batch_size)) + 1 show_progress(epoch, feed_dict_tr, feed_dict_val,
val_loss, total_epochs)
saver.save(session, 'trained_model') total_iterations += num_iteration
```

```
train(num_iteration=final_iter)


#except Exception as e: #print("Exception:",e)

# Calculate execution time
end = time.time()dur = end-start print("")
if dur<60:
print("Execution Time:",dur,"seconds")elif dur>60 and dur<3600:
dur=dur/60
print("Execution Time:",dur,"minutes")else:
dur=dur/(60*60)
print("Execution Time:",dur,"hours")
from flask import Flask, render_template, flash, request,
session,send_file
from flask import render_template, redirect, url_for, request
import warningsimport datetimeimport cv2
import tensorflow as tf
import numpy as np

from tkinter import *
import os


app = Flask(_name_)app.config['DEBUG']
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

@app.route("/")
def homepage():

return render_template('index.html')



@app.route("/Test")
def Test():
return render_template('Test.html')


@app.route("/train", methods=['GET', 'POST'])def train():
if request.method == 'POST':
import model as model

return render_template('Tranning.html')




@app.route("/testimage", methods=['GET', 'POST'])def testimage():
if request.method == 'POST':


file = request.files['fileupload'] file.save('data/alien_test/Test.jpg')
```

```
img = cv2.imread('data/alien_test/Test.jpg')

train_path = r'data\train'if not os.path.exists(train_path):print("No
such directory")
raise Exception
# Path of testing images dir_path  =  r'data\alien_test' if not
os.path.exists(dir_path):print("No  such  directory") raise Exception

# Walk though all testing images one by onefor root, dirs, files in
os.walk(dir_path):for name in files:

print("")


print(filename)

num_channels = 3images = []
image_path = name
filename = dir_path + '\\' + image_pathimage_size =128


if os.path.exists(filename):

# Reading the image using OpenCV
image1 = cv2.imread(filename)

import_file_path = filename

image = cv2.imread(import_file_path)
fnm = os.path.basename(import_file_path)filename = 'Test.jpg'
cv2.imwrite(filename, image)
# print("After saving image:")

print("\n*******************\nImage : " + fnm +
"\n*******************")img = cv2.imread(import_file_path)
if img is None: print('no data')


print(img.shape)
img1 = cv2.imread(import_file_path)

img = cv2.resize(img, ((int)(img.shape[1] / 5),
(int)(img.shape[0] / 5)))
original = img.copy() neworiginal = img.copy() cv2.imshow('original',
img1)
gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

cv2.imshow('Original image', img1)orimage = 'static/Out/Test.jpg'
cv2.imwrite(orimage, img1)
```

```
cv2.imshow('Gray image', gray)gry = 'static/Out/gry.jpg'

cv2.imwrite(gry, gray)


p = 0
for i in range(img.shape[0]):

for j in range(img.shape[1]):
B = img[i][j][0]
G = img[i][j][1]
R = img[i][j][2]
if (B >110 and G >110 and R >110):
p += 1

totalpixels = img.shape[0] * img.shape[1]
per_white = 100 * p / totalpixels
if per_white >10:


# Guassian blur


img[i][j] = [500, 300, 200]
cv2.imshow('color change', img)
blur1 = cv2.GaussianBlur(img, (3, 3), 1)
# mean-shift algo
newimg = np.zeros((img.shape[0], img.shape[1], 3), np.uint8)
criteria = (cv2.TERM_CRITERIA_EPS +cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
img = cv2.pyrMeanShiftFiltering(blur1, 20, 30, newimg, 0,criteria)

cv2.imshow('means shift image', img)noise = 'static/Out/noise.jpg'

cv2.imwrite(noise, img)




# Guassian blur
blur = cv2.GaussianBlur(img, (11, 11), 1)

blur = cv2.GaussianBlur(img, (11, 11), 1)
# Canny-edge detection
canny = cv2.Canny(blur, 160, 290)
canny = cv2.cvtColor(canny, cv2.COLOR_GRAY2BGR)
# contour to find leafs
bordered = cv2.cvtColor(canny, cv2.COLOR_BGR2GRAY)
contours, hierarchy = cv2.findContours(bordered,cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
maxC = 0 for x in range(len(contours)):if len(contours[x]) > maxC:
maxC = len(contours[x])maxid = x
perimeter = cv2.arcLength(contours[maxid], True)
```

```
# print perimeter
Tarea = cv2.contourArea(contours[maxid])
cv2.drawContours(neworiginal, contours[maxid], -1, (0, 0,
255))
cv2.imshow('Contour', neworiginal)
# cv2.imwrite('Contour complete leaf.jpg',neworiginal)# Creating
rectangular roi around contour
height, width, _ = canny.shape
min_x, min_y = width, height max_x = max_y = 0
frame = canny.copy()
# computes the bounding box for the contour, and draws it on the frame,
for contour, hier in zip(contours, hierarchy):
(x, y, w, h) = cv2.boundingRect(contours[maxid]) min_x, max_x = min(x,
min_x), max(x + w, max_x) min_y, max_y = min(y, min_y), max(y + h, max_y)
if w >80 and h >80:
# cv2.rectangle(frame, (x,y), (x+w,y+h), (255, 0, 0), 2)  #we do not draw
therectangle as it interferes with contour later on
roi = img[y:y + h, x:x + w]
originalroi = original[y:y + h, x:x + w]
if (max_x - min_x >0 and max_y - min_y >0):
roi = img[min_y:max_y, min_x:max_x]
originalroi = original[min_y:max_y, min_x:max_x] cv2.rectangle(frame,
(min_x, min_y), (max_x, max_y), (255,
0, 0),
2)  # we do not draw the rectangle as it interferes with contour
cv2.imshow('ROI', frame)

roi12 = 'static/Out/roi.jpg'

cv2.imwrite(roi12, frame)



cv2.imshow('rectangle ROI', roi)
img = roi
# Changing colour-space
# imghsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
imghls = cv2.cvtColor(roi, cv2.COLOR_BGR2HLS)
cv2.imshow('HLS', imghls)
imghls[np.where((imghls == [30, 200, 2]).all(axis=2))] = [0,
200, 0]

# Only hue channel

cv2.imshow('new HLS', imghls)
huehls = imghls[:, :, 0]
cv2.imshow('img_hue hls', huehls)
# ret, huehls = cv2.threshold(huehls,2,255,cv2.THRESH_BINARY)
huehls[np.where(huehls == [0])] = [35]
cv2.imshow('img_hue with my mask', huehls)
# Thresholding on hue image
ret, thresh = cv2.threshold(huehls, 28, 255, cv2.THRESH_BINARY_INV)
cv2.imshow('thresh', thresh)
```

```
# Masking thresholded image from original image
mask = cv2.bitwise_and(originalroi, originalroi, mask=thresh)
cv2.imshow('masked out img', mask)

# Resizing the image to our desired size and preprocessing will be done
exactly as done during training
image = cv2.resize(image1, (image_size, image_size), 0, 0,
cv2.INTER_LINEAR) images.append(image)
images = np.array(images, dtype=np.uint8) images =
images.astype('float32')
images = np.multiply(images, 1.0 / 255.0)
# The input to the network is of shape [None image_size image_size
num_channels]. Hence we reshape.
x_batch = images.reshape(1, image_size, image_size, num_channels)

# Let us restore the saved model
sess = tf.compat.v1.Session()
# Step-1: Recreate the network graph. At this step only graph is created.
saver = tf.compat.v1.train.import_meta_graph('models/trained_model.meta')
# Step-2: Now let's load the weights saved using the restore method.
saver.restore(sess, tf.train.latest_checkpoint('./models/'))

# Accessing the default graph which we have restored
graph = tf.compat.v1.get_default_graph()

# Now, let's get hold of the op that we can be processed to get the
output.
# In the original network y_pred is the tensor that is the prediction of
the network
y_pred = graph.get_tensor_by_name("y_pred:0")

## Let's feed the images to the input placeholders
x = graph.get_tensor_by_name("x:0")
y_true = graph.get_tensor_by_name("y_true:0") y_test_images =
np.zeros((1, len(os.listdir(train_path))))

# Creating the feed_dict that is required to be fed to calculate y_pred
feed_dict_testing = {x: x_batch, y_true: y_test_images}
result = sess.run(y_pred, feed_dict=feed_dict_testing)
# Result is of this format [[probabiliy_of_classA probability_of_classB
..................................................... ]]
print(result)

# Convert np.array to list
a = result[0].tolist()
r = 0

# Finding the maximum of all outputs
max1 = max(a)
index1 = a.index(max1) predicted_class = None
```

```
# Walk through directory to find the label of the predicted output
count = 0
for root, dirs, files in os.walk(train_path):
for name in dirs:
if count == index1:
predicted_class = namecount += 1

# If the maximum confidence output is largest of all by a big margin then
# print the class or else print a warning
for i in a:
if i != max1:
if max1 - i < i:r = 1

out = ''

pre = "" if r == 0:
print(predicted_class)
if (predicted_class == "Black spot"):
out = predicted_class

pre = 'Griffin  Fertilizer  reducing the fungus'elif (predicted_class ==

"canker"):
out = predicted_class
pre = 'sprayed with Bordeaux mixture 1.0 per cent.'


elif (predicted_class == "greening"):
out =  predicted_class
pre =  'Mn-Zn-Fe-B micronutrient fertilizer'

elif (predicted_class == "healthy"):
out =  predicted_class
# messagebox.showinfo("Uses", '')
elif (predicted_class == "Melanose"):
out = predicted_class
pre =  'strobilurin fungicide'





else:

out = 'Could not classify with definite confidence'



else:
print("File does not exist")

org = 'static/Out/Test.jpg'
gry ='static/Out/gry.jpg' noise = 'static/Out/noise.jpg'roi12 =
```

```
'static/Out/roi.jpg'

return
render_template('Test.html',result=out,org=org,gry=gry,inv=noise,noi=roi1
2,fer=pre)
def sendmsg(targetno,message):
import requests

requests.post("http://smsserver9.creativepoint.in/api.php?username=fantasy
&password=596692&to=" + targetno + "&from=FSSMSS&message=Dear user  your
msg is " +
message + " Sent By FSMSG
FSSMSS&PEID=1501563800000030506&templateid=1507162882948811640")




if__name__== '__main__': app.run(debug=True, use_reloader=True)
import cv2 import os import glob
from sklearn.utils import shuffle
import numpy as np


def load_train(train_path, image_size, classes):images = []
labels = [] img_names = []cls = []

print('Going to read training images')for fields in classes:
index = classes.index(fields)
print('Now going to read {} files (Index: {})'.format(fields, index))path
= os.path.join(train_path, fields, '*g')
files = glob.glob(path)
for fl in files:
image = cv2.imread(fl)
image = cv2.resize(image, (image_size, image_size),0,0,cv2.INTER_LINEAR)
image = image.astype(np.float32)
image = np.multiply(image, 1.0 / 255.0)images.append(image)
label = np.zeros(len(classes))label[index] = 1.0
labels.append(label)
flbase = os.path.basename(fl)img_names.append(flbase) cls.append(fields)
images = np.array(images) labels = np.array(labels) img_names =
np.array(img_names)cls = np.array(cls)

return images, labels, img_names, cls


class DataSet(object):

def_init_(self, images, labels, img_names, cls):self._num_examples =
images.shape[0]
self._images = images self._labels = labels self._img_names = img_names
```

```
self._cls = cls self._epochs_done = 0
self._index_in_epoch = 0

@property
def images(self):
return self._images

@property
def labels(self):
return self._labels

@property
def img_names(self):
return self._img_names

@property
def cls(self):
return self._cls

@property
def num_examples(self):
return self._num_examples

@property
def epochs_done(self):
return self._epochs_done

def next_batch(self, batch_size):
"""Return the next `batch_size` examples from this data set."""
start = self._index_in_epoch self._index_in_epoch += batch_size

if self._index_in_epoch >self._num_examples:
# After each epoch we update this
self._epochs_done += 1
start = 0
self._index_in_epoch = batch_size
assert batch_size <= self._num_examples end = self._index_in_epoch

return self._images[start:end], self._labels[start:end],
self._img_names[start:end], self._cls[start:end]


def read_train_sets(train_path, image_size, classes, validation_size):
class DataSets(object):
pass
data_sets = DataSets()

images, labels, img_names, cls = load_train(train_path, image_size,
classes) images, labels, img_names, cls = shuffle(images, labels,
img_names, cls)

if isinstance(validation_size, float):
```

```
validation_size = int(validation_size * images.shape[0])
validation_images = images[:validation_size] validation_labels =
labels[:validation_size] validation_img_names =
img_names[:validation_size]validation_cls = cls[:validation_size]

train_images = images[validation_size:] train_labels =
labels[validation_size:] train_img_names = img_names[validation_size:]
train_cls = cls[validation_size:]

data_sets.train = DataSet(train_images, train_labels, train_img_names,
train_cls)
data_sets.valid = DataSet(validation_images, validation_labels,
validation_img_names, validation_cls)

return  data_sets import tensorflow as tfimport numpy as np

from tkinter import *
import os
from tkinter import filedialog
import cv2
import time
from matplotlib import pyplot as plt
from tkinter import messagebox




def endprogram():
print ("\nProgram terminated!")sys.exit()



def training():

import Training as tr



def imgtraining():
import_file_path = filedialog.askopenfilename()

image = cv2.imread(import_file_path)filename = 'Test.jpg'
cv2.imwrite(filename, image)print("After saving image:")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow('Original image', image)cv2.imshow('Gray image', gray)
# import_file_path = filedialog.askopenfilename()
```

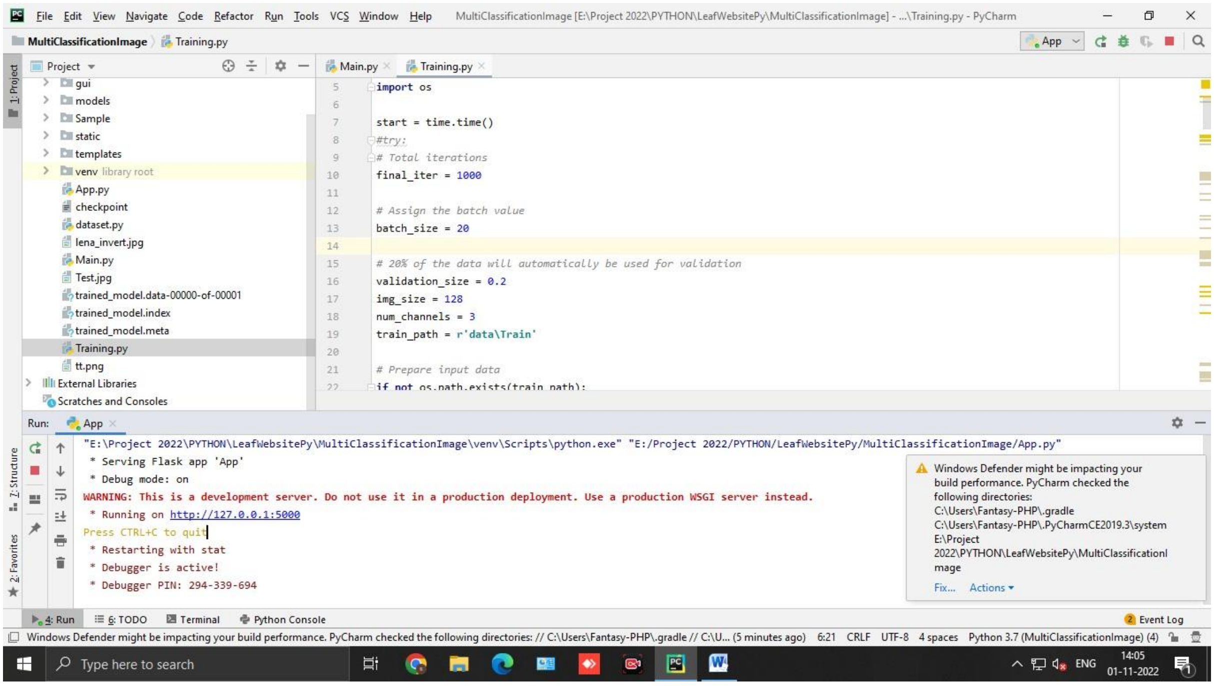
```
print(import_file_path)
fnm = os.path.basename(import_file_path)
print(os.path.basename(import_file_path))

from PIL import Image, ImageOps

im = Image.open(import_file_path) im_invert = ImageOps.invert(im)
im_invert.save('lena_invert.jpg', quality=95)
im = Image.open(import_file_path).convert('RGB') im_invert =
ImageOps.invert(im) im_invert.save('tt.png')
image2 = cv2.imread('tt.png') cv2.imshow("Invert", image2)

""""-----------------------------------------------"""

img = image

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) cv2.imshow('Original image',
img)
#cv2.imshow('Gray image', gray)
dst = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)
cv2.imshow("Nosie Removal", dst)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)



print("\n*******************\nImage : " + fnm +
"\n*******************") img = cv2.imread(import_file_path)
if img is None:
print('no data')

img1 = cv2.imread(import_file_path) print(img.shape)
img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))
original = img.copy()
neworiginal = img.copy() cv2.imshow('original', img1)
gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

cv2.imshow('Original image', img1)
# cv2.imshow('Gray image', gray)
p = 0
for i in range(img.shape[0]):

for j in range(img.shape[1]):
B = img[i][j][0]
G = img[i][j][1]
R = img[i][j][2]
if (B >110 and G >110 and R >110):
p += 1

totalpixels = img.shape[0] * img.shape[1] per_white = 100 * p /
totalpixels
```

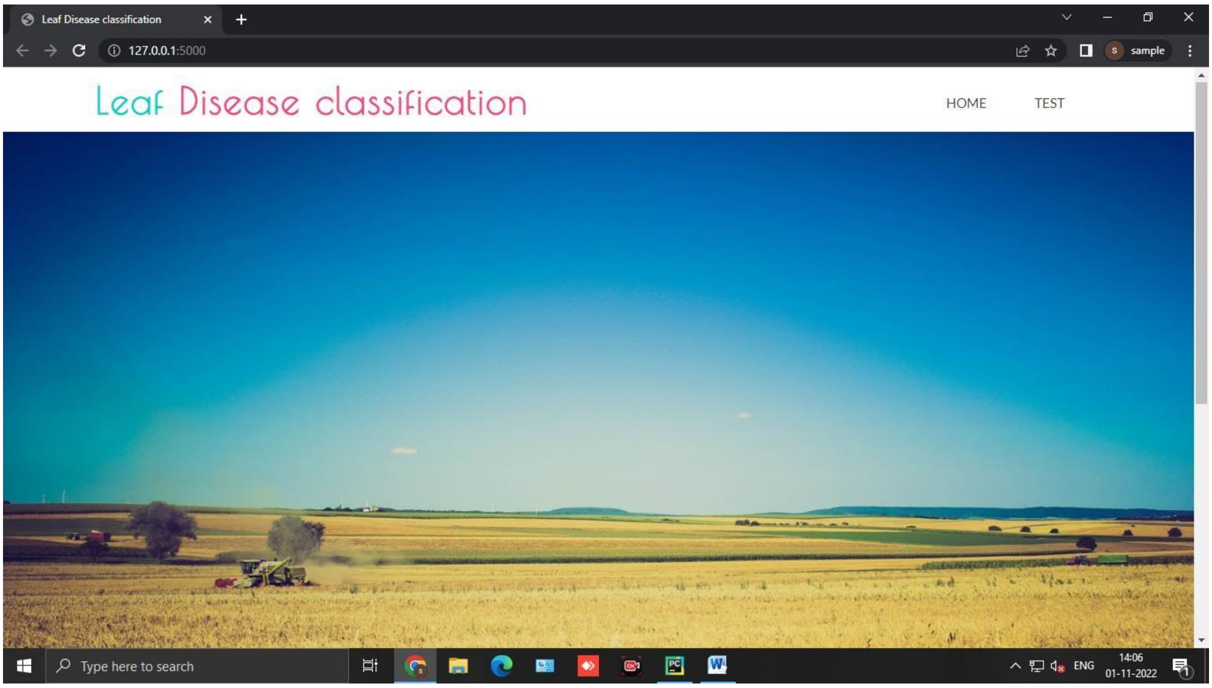
```
if per_white >10:
img[i][j] = [500, 300, 200]
cv2.imshow('color change', img)
# Guassian blur
blur1 = cv2.GaussianBlur(img, (3, 3), 1)
# mean-shift algo
newimg = np.zeros((img.shape[0], img.shape[1], 3), np.uint8)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
img = cv2.pyrMeanShiftFiltering(blur1, 20, 30, newimg, 0, criteria)
cv2.imshow('means shift image', img)
# Guassian blur
blur = cv2.GaussianBlur(img, (11, 11), 1)cv2.imshow('Noise Remove', blur)
corners = cv2.goodFeaturesToTrack(gray, 27, 0.01, 10)corners =
np.int0(corners)

# we iterate through each corner,
# making a circle at each point that we think is a corner.
for i in corners:
x, y = i.ravel()
cv2.circle(image, (x, y), 3, 255, -1)plt.imshow(image), plt.show()




def testing():
global testing_screen
testing_screen = Toplevel(main_screen)testing_screen.title("Testing")
# login_screen.geometry("400x300")
testing_screen.geometry("600x450+650+150")testing_screen.minsize(120, 1)
testing_screen.maxsize(1604, 881)
testing_screen.resizable(1, 1)
# login_screen.title("New Toplevel")

Label(testing_screen, text='''Upload Image''', background="#d9d9d9",
disabledforeground="#a3a3a3",
foreground="#000000", bg="turquoise", width="300", height="2",
font=("Calibri",16)).pack()
Label(testing_screen, text="").pack() Label(testing_screen,
text="").pack() Label(testing_screen, text="").pack()
Button(testing_screen, text='''Upload Image''', font=(
'Verdana', 15), height="2", width="30", command=imgtest).pack()

def imgtest():
import_file_path = filedialog.askopenfilename()

image = cv2.imread(import_file_path)print(import_file_path)
filename = 'data/alien_test/Test.jpg'cv2.imwrite(filename,  image)
print("After saving image:")
```

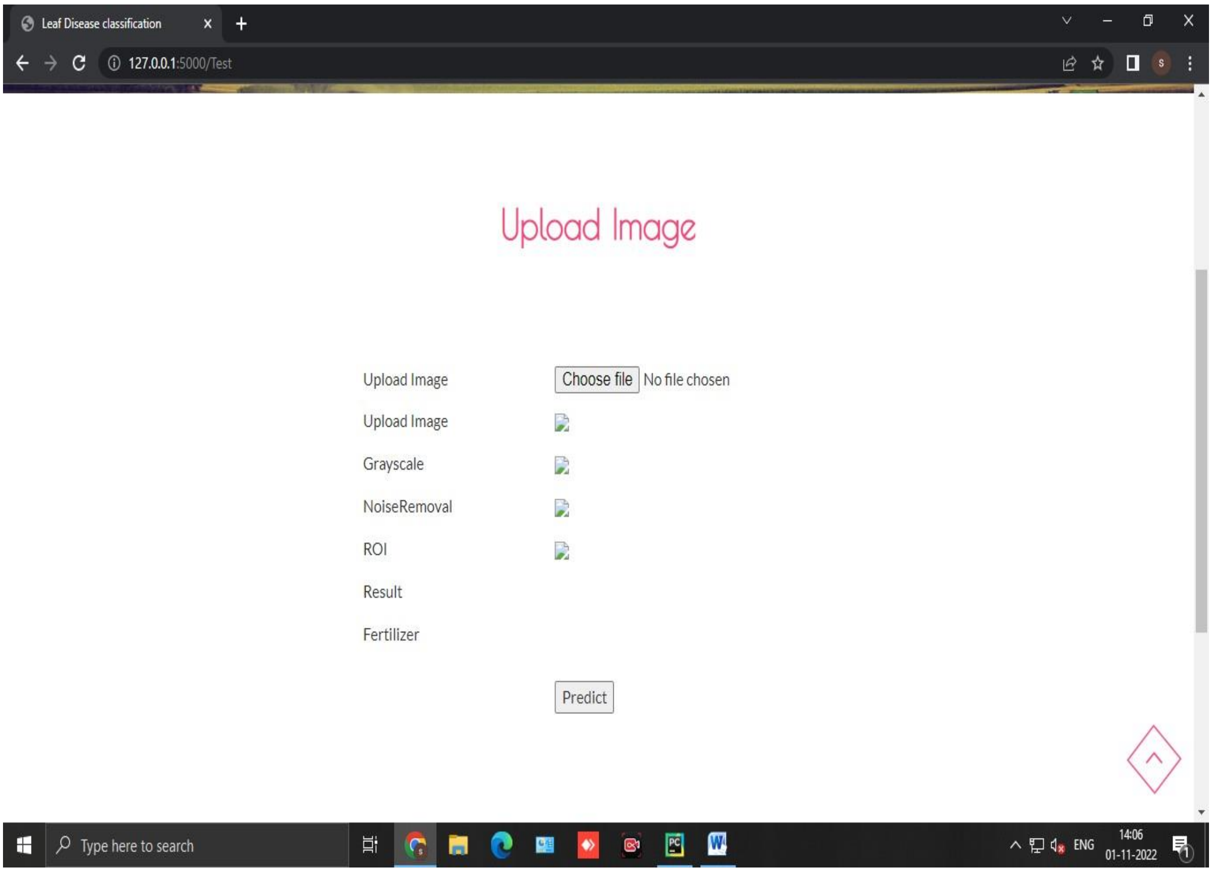
```
def main_account_screen():
from PIL import Image, ImageTk
global main_screen main_screen = Tk()width = 600
height = 600
screen_width = main_screen.winfo_screenwidth() screen_height =
main_screen.winfo_screenheight()x = (screen_width / 2) - (width / 2)
y = (screen_height / 2) - (height / 2) main_screen.geometry("%dx%d+%d+%d"
% (width, height, x, y))main_screen.resizable(0, 0)
# main_screen.geometry("300x250")
main_screen.title("Leaf Disease classification")

Label(text="Leaf Disease classification", bg="turquoise", width="300",
height="5", font=("Calibri", 16)).pack()
Label(text="").pack()
Label(text="").pack()

image = ImageTk.PhotoImage(Image.open('gui/12344.jpg'))

Label(main_screen, text='Hello', image=image, compound='left',
height="100",width="200",).pack()

Button(text="Training", font=(
'Verdana', 15), height="2", width="30", command=training,
highlightcolor="black").pack(side=TOP)
Label(text="").pack() Button(text="Testing", font=(
'Verdana', 15), height="2", width="30", command=testing).pack(side=TOP)

Label(text="").pack()

main_screen.mainloop()


main_account_screen()
```

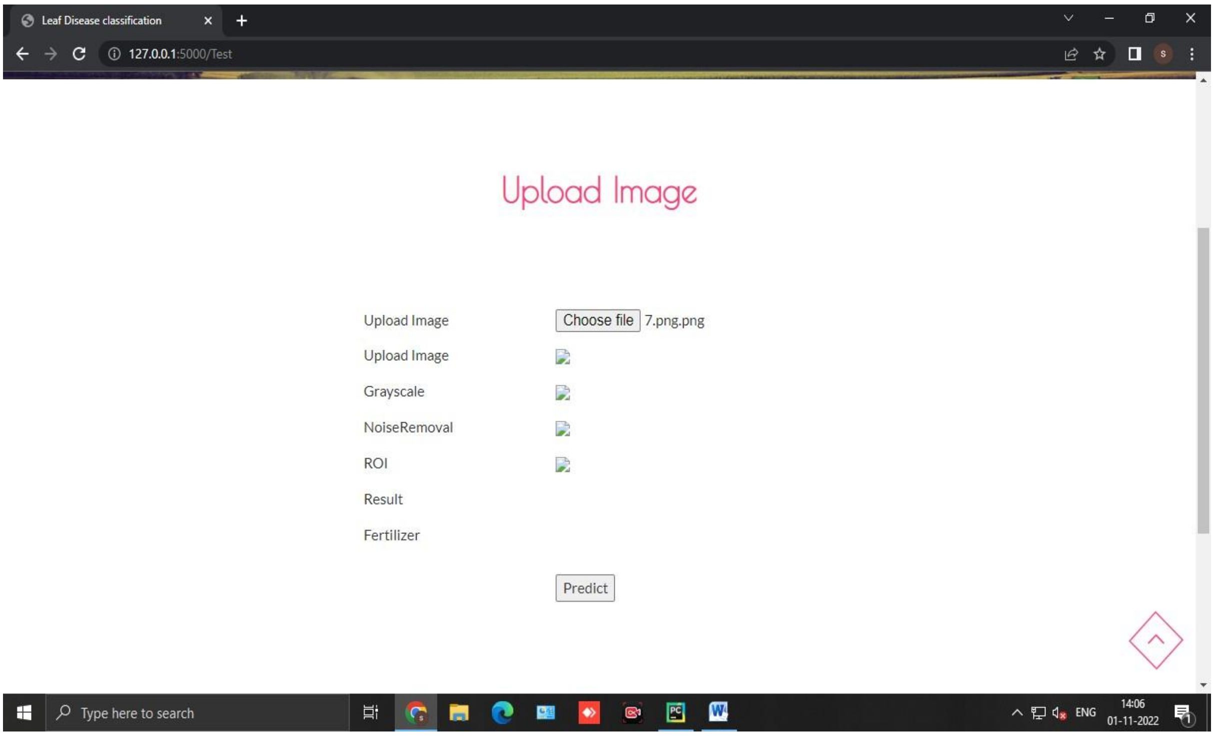
APPENDIX -2

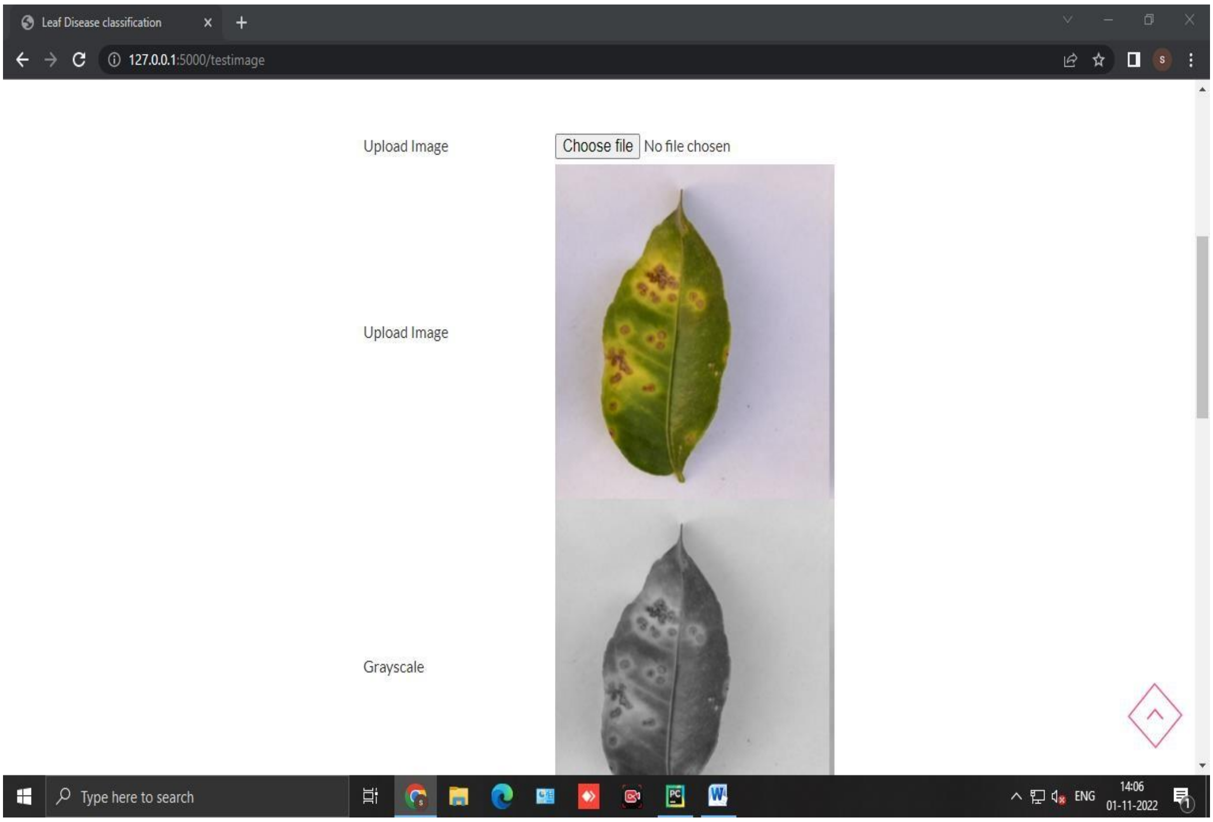
OUTPUT SCREENSHOTS:

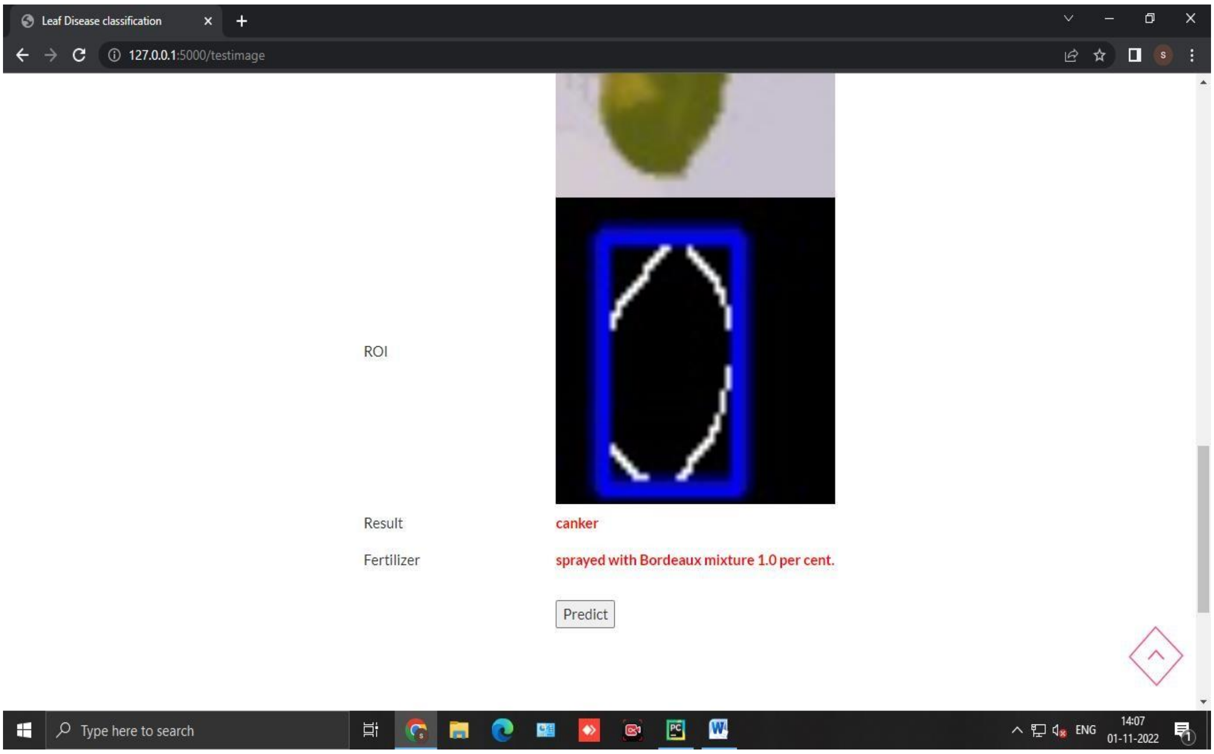





Leaf Disease classification
127.0.0.1:5000/Test
Upload Image
Upload Image
Choose file
No file chosen
Upload Image
Grayscale
NoiseRemoval
ROI
Result
Fertilizer
Predict
Type here to search
ENG
14:06
01-11-2022


Leaf Disease classification
127.0.0.1:5000/Test
Upload Image
Upload Image
Choose file
7.png.png
Upload Image
Grayscale
NoiseRemoval
ROI
Result
Fertilizer
Predict
Type here to search
ENG
14:06
01-11-2022


Leaf Disease classification
127.0.0.1:5000/testimage
Upload Image
Choose file No file chosen
Upload Image
Grayscale
Type here to search
ENG
14:06
01-11-2022


Leaf Disease classification
127.0.0.1:5000/testimage
ROI
Result
canker
Fertilizer
sprayed with Bordeaux mixture 1.0 per cent.
Predict
Type here to search
ENG
14:07
01-11-2022

GITHUB:

https://github.com/IBM-EPBL/IBM-Project-28925-1660119014

DEMO VIDEO LINK:

https://drive.google.com/file/d/15g577gIuyUb0dPvop-337Y7pB64MEMUC/view?usp=share_link