PLANT DISEASE FERTILIZER RECOMMENDATION SYSTEM

A UG-PROJECT PHASE-1 REPORT

Submitted To

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfillment for the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By:

THALLADA SAI TEJA

H.NO: 18UK1A05B2

Under the Guidance of

MS G.ARUNA KRANTHI

Assistance Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEEERING COLLEGE

(Affiliated to JNTUH, Hyderabad)

Bollikunta, Warangal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLLEGE

BOLLIKUNTA WARANGAL,(T.S)

CERTIFICATE OF COMPLETION
UG PROJECT PHASE-1

This is to certify that the UG Project Phase-1 entitled **"Plant Disease Fertilizer Recommendation System"** is submitted by ***THALLADA SAI TEJA*** Bearing the **H.no:18UK1A05B2** in partial fulfillment of the requirement for the award of the Degree in Bachelor of Technology in **Computer Science and Engineering** to **Jawaharlal Nehru Technological University, Hyderabad** during the academic year(2021-2022), is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
MS. G. Aruna Kranthi	**Dr. R. Naveen Kumar**
(Assistant Professor)	(Professor)

External

ACKNOWLEDGEMENT

I wish to take this opportunity to express our sincere gratitude an deep sense of respect to our beloved **DR. P. PRASAD RAO, PRINCIPAL** Vaagdevi Engineering College for making us available all the required and for this support and inspiration to carry out this UG Project phase-1 in the institute.

I extend our heartfelt thanks to **DR.R.NAVEEN KUMAR,** Head, Department of CSE, Vaagdevi Engineering College for providing us the necessary infrastructure and there by giving us the freedom to carry out the UG Project phase-1.

I express heartfelt thanks to the guide, **MS.G.ARUNA KRANTHI,** Assistant Professor Department of CSE for this content support and giving necessary guidance for completion of this UG Project phase-1.

I express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

Finally, I express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experiencing throughout these.

THALLADA SAI TEJA

H.NO: 18UK1A05B2

ABSTRACT

In our country agriculture is the main occupation. Most of the people lead their life from agriculture field, they are fully relying on agricultural products. If any plant is enduring disease, then it causes reduction in both quality and quantity of agriculture crops. Hence it is necessary to detect and analysis of disease. Authentic exposure and recognition of crop disease plays an important role in adequately regulating and inhibiting disease for feasible agriculture and food preservation.

Thus detection and diagnosis of disease at the right time is essential to the farmer. This proposed paper offers a candid and computationally resourceful manner which is useful in the leaf disease detection and selection of fertilizers using artificial neural network. This proposed system involves different concepts related to image processing such as image acquisition, image pre-processing, feature extraction, artificial neural network based training, classification, diagnosis and treatment by using artificial neural network.

In this performance, database is a accumulation of different texture features of some leaves. In this paper it is possible to get the disease name and also we can get the fertilizer which is precise for that disease. But in previous attempts it was not happen so because of this we proposed this project, it is better and gives good performance compared to other processing system.

INDEX

LIST OF FIGURES

1. INTRODUCTION

1.1 Overview

The project Plant Disease Fertilizer Recommendation System is a web application developed by Deep Learning in which the user can upload the images of plant leaves which are affected by diseases and will get to know the problem and predicted solution to recover the plant diseases. Agriculture is the most important sector in today’s life. Most of the plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and major threat to food security. Hence, early and accurate identification of plant diseases are essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods and inadequate plant protection techniques.

1.2 Purpose

Farmers or Gardeners are often worried about the plant diseases and the effect of diseases on plants. The aim of the Plant Disease Fertilizer Recommendation System is to find solution for the plant disease and for suggesting ideas to avoid such type of diseases by using recommended fertilizers. 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.

Plant disease especially on leaves is the one of the major reagent of reduction in both quality and quantity of the food crops. The quality and quantity of food production become reduced only because of pest’s presence in the crops and leaves. Thus it leads to increase in difficulty, food insecurity and fatality rate. In modern years in order to identify the plant disease, so many different concepts of image process technology have been adapted.

2. LITERATURE SURVEY

2.1 Existing Problem

Farmers spend so much money on disease management, often without adequate technical support, resulting in poor disease control, pollution and harmful results. In addition, plant disease can devastate natural ecosystems, compounding environmental problems caused by habitat loss and poor land management. A symptom of plant disease is a visible effect of disease on the plant. Symptoms may include a detectable change in colour, shape or function of the plant as it responds to the pathogen. Leaf wilting is a typical symptom.

2.2 Proposed Solution

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.This system is based on Machine Vision Technology and Artificial Neural Network and it is useful for automatically detecting the leaf plant and also leaf disease and grading but from this it is not possible to detect the fertilizer name which is suitable for the disease.

Our system covers almost all possible viral diseases for cucumber and imposes less diagnostic restrictions during image acquisition. The proposed system uses Euclidean distance technique and K means clustering technique for segmentation of image to segment the leaf area, disease area and background area of the input leaf image in order to calculate the percentage infection of the disease in the leaf and to grade them into various classes.

3. THEORITICAL ANALYSIS

3.1 Block Diagram

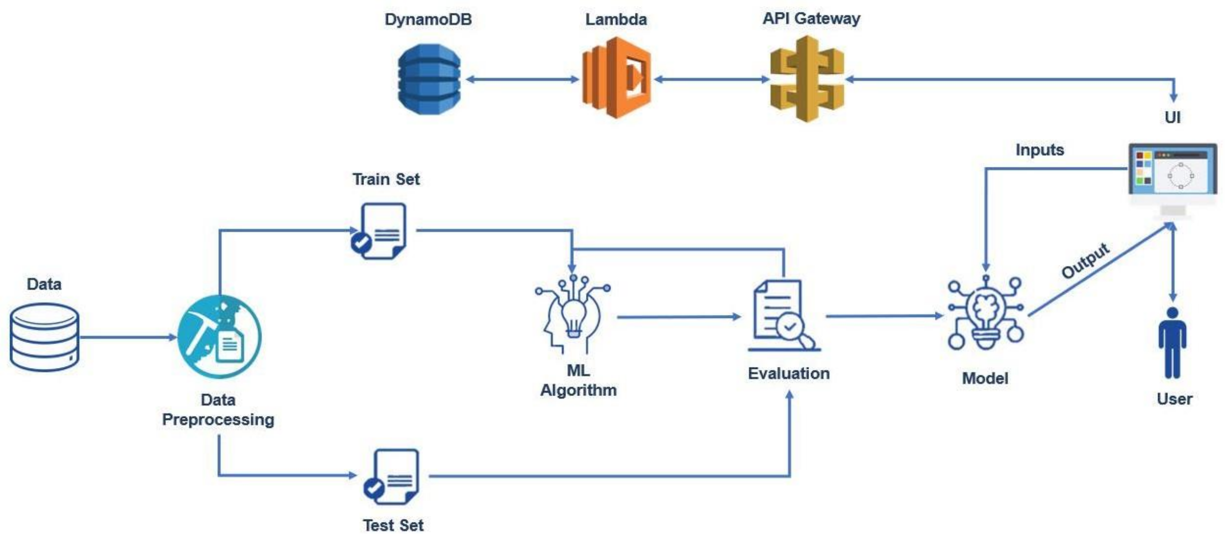


Figure: 1 Block Diagram

In the above block diagram, the data set (images of different kind of fruits and vegetables plant leaves which are affected from different kind of diseases) is classified into train and test sets. Algorithms are performed on train set and then the train and test sets are evaluated. At the end a deep learning model is built. User gives the input (plant disease image) and the model provides predicted output (the disease from which the plant is affected and fertilizer to recover it) to the user. DynamoDB on AWS cloud is used to store the user’s data, Lambda and API Gateway are used to retrieve user’s data.

3.2 Software and Hardware Designing

Software Used

- Python (version 3.6x)
- Anaconda 3
- Jupyter Notebook & Spyder
- Windows 7 or higher
- Python Flask

Hardware Required

- Processor – Dual Core or higher
- Hard Disk – 50 GB
- Memory – 4 GB RAM

4.EXPERIMENTAL INVESTIGATION

According to the International Research Journal of Engineering and Technology (IRJET), the proposal introduced in 2019 which deals with IOT based system using hardware and various image processing tools. Detection and recognition of plant diseases using Deep learning is efficient providing symptoms of identifying diseases at its earliest. This project is a Deep learning model with highest accuracy rate and the least error.

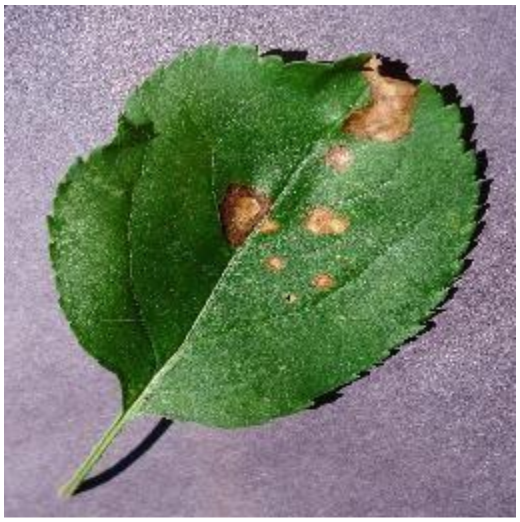


Figure 2 : Apple plant leaf effected with bacterial spot.

In first phase recognition of plant based on feature of leaf is done and in the second phase classification of disease present in the leaf and grading of disease is done on the basis of the amount of disease in the leaf. This system is based on Machine Vision Technology and Artificial Neural Network and it is useful for automatically detecting the leaf plant and also leaf disease and grading but from this it is not possible to detect the fertilizer name which is suitable for the disease. Our system covers almost all possible viral diseases for cucumber and imposes less diagnostic restrictions during image acquisition.

The proposed system uses Euclidean distance technique and K means clustering technique for segmentation of image to segment the leaf area, disease area and background area of the input leaf image in order to calculate the percentage infection of the disease in the leaf and to grade them into various classes. Ms.pooja pawar et, al presents algorithm for detecting crop disease early and exactly, this system is developed using image processing techniques and artificial neural network. It includes different concepts relevant to image processing such as image acquisition, image pre-processing, feature extraction, creating database and classification by using artificial neural network.

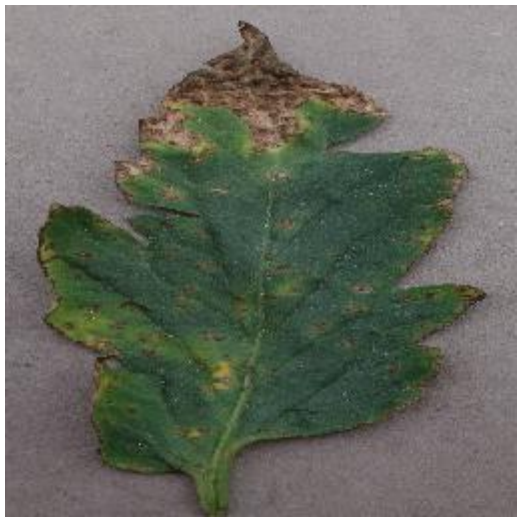


Figure 3 : Tomato plant leaf effected with bacterial spot.

In this, database is a collection of various texture features of leaves. This system involves collecting leaf samples of cucumber crop diseases. Process is executed to diagnose cucumber crop disease and to provide treatment for the detected crop disease. Two cucumber crop diseases downy mildew and powdery mildew are considered in this work. The GLCM and first order statistical moments are used to extract texture features. System provides classification accuracy of only 80.45% not 100%. This is possible to use in more than one crop of different types. But for other crop type, system has to select those features only that can classify their crop diseases accurately. Erika fujita, yusuke Kawasaki et, al introduced a system of Basic investigation on a robust and practical plant diagnostic system.

This system offers a new practical plant-disease detection system, for this it takes 7,520 cucumber leaf images comprising images of healthy leaves and those infected by different viral diseases.

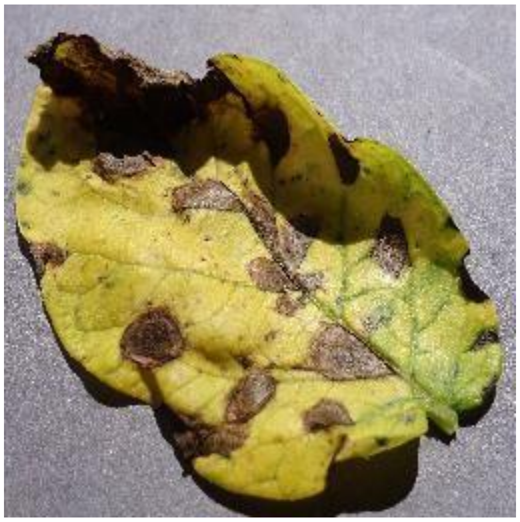


Figure 4 : Potato plant leaf effected with Bacterial spot.

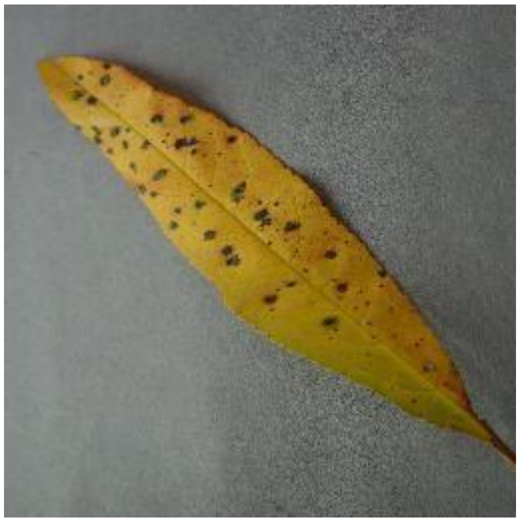


Figure 5: Peach plant leaf effected with Bacterial spot.

The leaves were photographed on site under only one requirement, that is, each image must contain a leaf roughly at its center, thus providing them with a large variety of appearances. Although half of the images used in this experiment were taken in bad conditions, In this paper classification is done on the basis of conventional neural networks attained an average of 82.3% accuracy under the 4-fold cross validation strategy.

5.FLOW CHAT

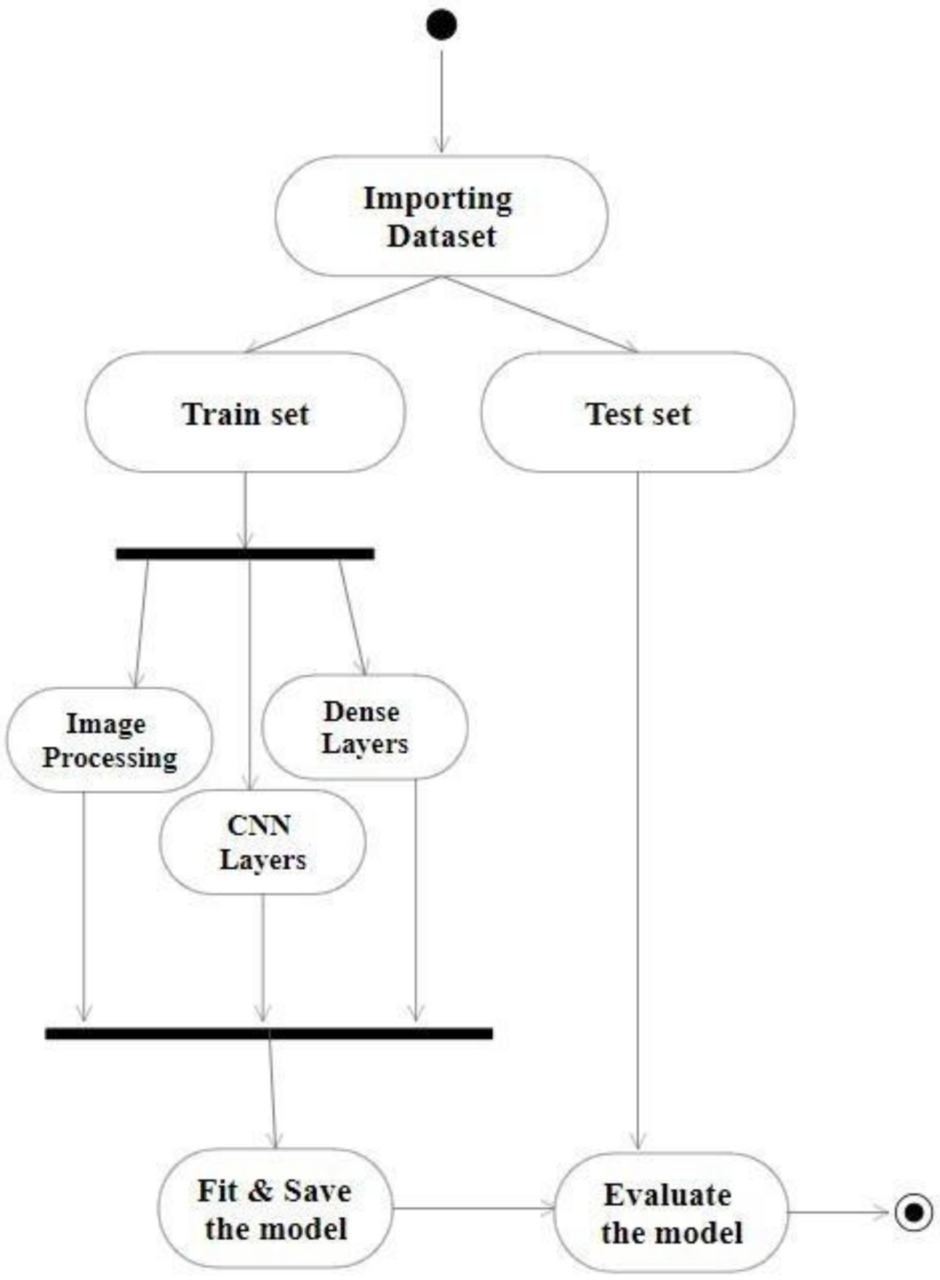


Figure 6 : Flow Chart.

6.CONCLUSION

On the basis of the fruit and vegetable plant disease data set, the proposed model provides the users disease from which the plant is affected from and recommends fertilizer to recuperate the affected area. We have loaded the data set into the model and checked its outcome. We have achieved 91% of accurate result from obtained outcome. By this we conclude that the project Plant Disease Fertilizer Recommendation System is accurate.

7.FUTURE SCOPE

UG Project Phase-2 is the extension of UG Project Phase-1. UG Project Phase-2 involves all the coding and implementation of the design which we have retrieved from UG Project Phase-1. All the implementation is done and conclusions will be retrieved in the phase. We will also work on the applications, advantages, and disadvantages of the project in this phase. Future scope of the project will be also discussed in the UG Project Phase-2.

PLANT DISEASE FERTILIZER RECOMMENDATION SYSTEM

A UG-PROJECT PHASE-2 REPORT

Submitted To

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfillment for the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By:

THALLADA SAI TEJA

H.NO: 18UK1A05B2

Under the Guidance of

MS G.ARUNA KRANTHI

Assistance Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEEERING COLLEGE

(Affiliated to JNTUH, Hyderabad)

Bollikunta, Warangal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLLEGE

BOLLIKUNTA WARANGAL,(T.S)

CERTIFICATE OF COMPLETION
UG PROJECT PHASE-2

This is to certify that the UG Project Phase-2 entitled **"Plant Disease Fertilizer Recommendation System"** is submitted by ***THALLADA SAI TEJA*** Bearing the **H.no:18UK1A05B2** in partial fulfillment of the requirement for the award of the Degree in Bachelor of Technology in **Computer Science and Engineering** to **Jawaharlal Nehru Technological University, Hyderabad** during the academic year(2021-2022), is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
MS. G. Aruna Kranthi	**Dr. R. Naveen Kumar**
(Assistant Professor)	(Professor)

External

ACKNOWLEDGEMENT

I wish to take this opportunity to express our sincere gratitude an deep sense of respect to our beloved **DR. P. PRASAD RAO, PRINCIPAL** Vaagdevi Engineering College for making us available all the required and for this support and inspiration to carry out this UG Project phase-2 in the institute.

I extend our heartfelt thanks to **DR.R.NAVEEN KUMAR,** Head, Department of CSE, Vaagdevi Engineering College for providing us the necessary infrastructure and there by giving us the freedom to carry out the UG Project phase-2.

I express heartfelt thanks to the guide, **MS.G.ARUNA KRANTHI,** Assistant Professor Department of CSE for this content support and giving necessary guidance for completion of this UG Project phase-2.

I express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-2 and for their support in completing the UG Project Phase-2.

Finally, I express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experiencing throughout these.

THALLADA SAI TEJA

H.NO: 18UK1A05B2

INDEX

LIST OF FIGURES:

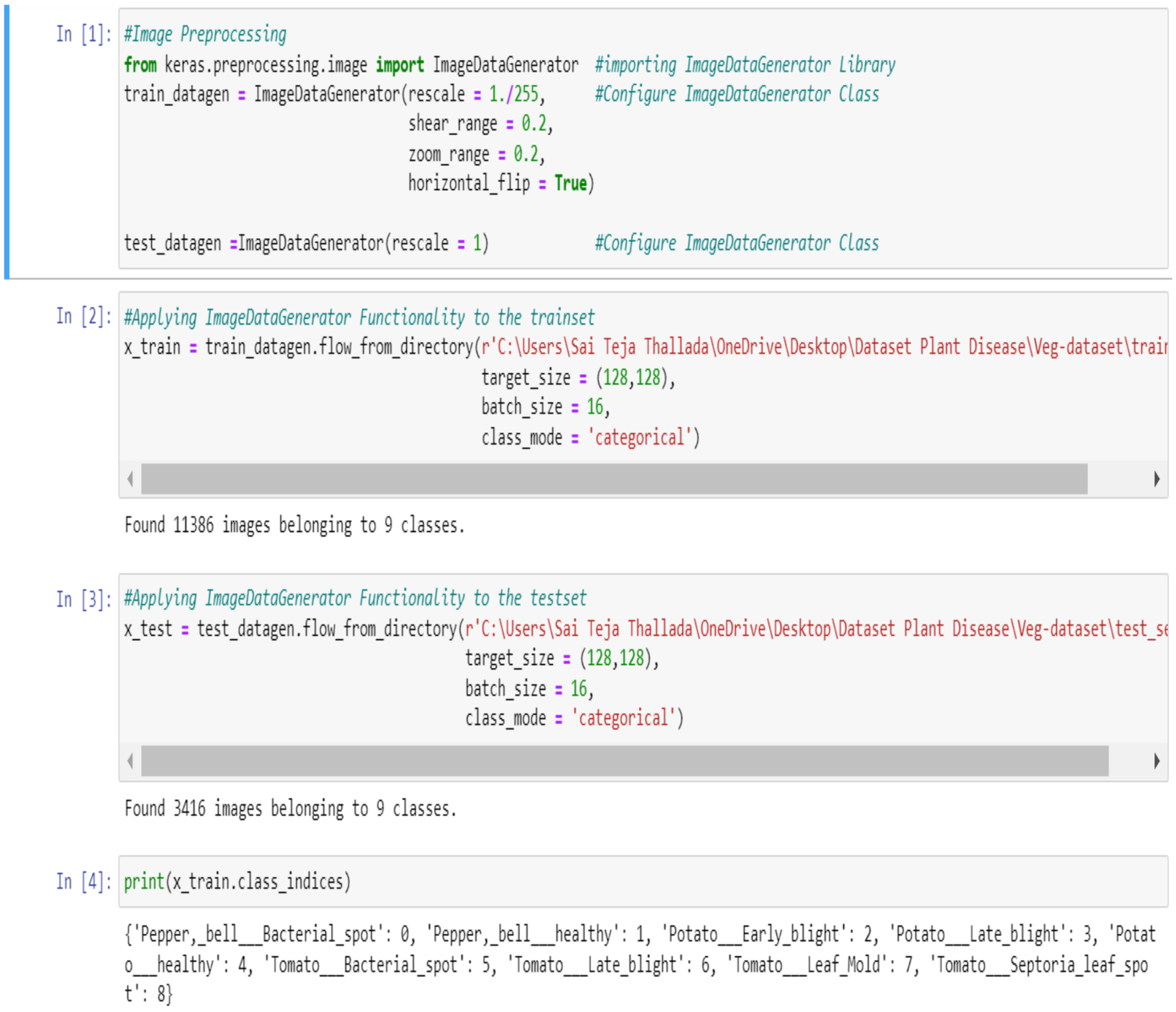
1.INTRODUCTION

The project Plant Disease Fertilizer Recommendation System is a web application developed by Deep Learning in which the user can upload the images of plant leaves which are affected by diseases and will get to know the problem and predicted solution to recover the plant diseases. Agriculture is the most important sector in today’s life. Most of the plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and major threat to food security. Hence, early and accurate identification of plant diseases are essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods and inadequate plant protection techniques

Plant disease especially on leaves is the one of the major reagent of reduction in both quality and quantity of the food crops. The quality and quantity of food production become reduced only because of pest’s presence in the crops and leaves. Thus it leads to increase in difficulty, food insecurity and fatality rate. In modern years in order to identify the plant disease, so many different concepts of image process technology have been adapted.

2.CODE SNIPPETS

2.1 Vegetable Model code:



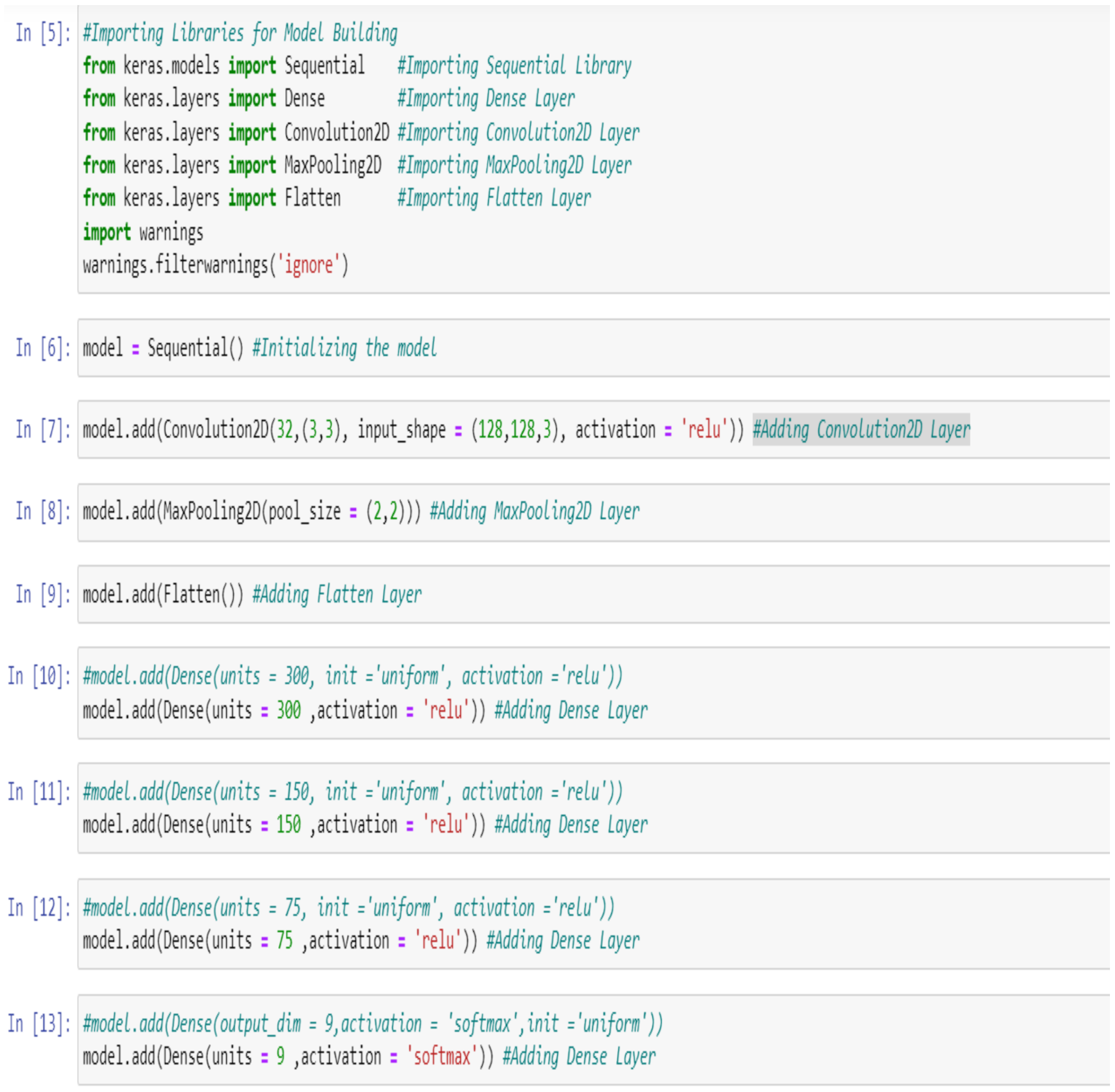
```
In [1]: #Image Preprocessing
        from keras.preprocessing.image import ImageDataGenerator  #importing ImageDataGenerator Library
        train_datagen = ImageDataGenerator(rescale = 1./255,      #Configure ImageDataGenerator Class
                                           shear_range = 0.2,
                                           zoom_range = 0.2,
                                           horizontal_flip = True)

        test_datagen =ImageDataGenerator(rescale = 1)             #Configure ImageDataGenerator Class
```

```
In [2]: #Applying ImageDataGenerator Functionality to the trainset
        x_train = train_datagen.flow_from_directory(r'C:\Users\Sai Teja Thallada\OneDrive\Desktop\Dataset Plant Disease\Veg-dataset\trair
                                                    target_size = (128,128),
                                                    batch_size = 16,
                                                    class_mode = 'categorical')

        Found 11386 images belonging to 9 classes.
```

```
In [3]: #Applying ImageDataGenerator Functionality to the testset
        x_test = test_datagen.flow_from_directory(r'C:\Users\Sai Teja Thallada\OneDrive\Desktop\Dataset Plant Disease\Veg-dataset\test_se
                                                  target_size = (128,128),
                                                  batch_size = 16,
                                                  class_mode = 'categorical')

        Found 3416 images belonging to 9 classes.
```

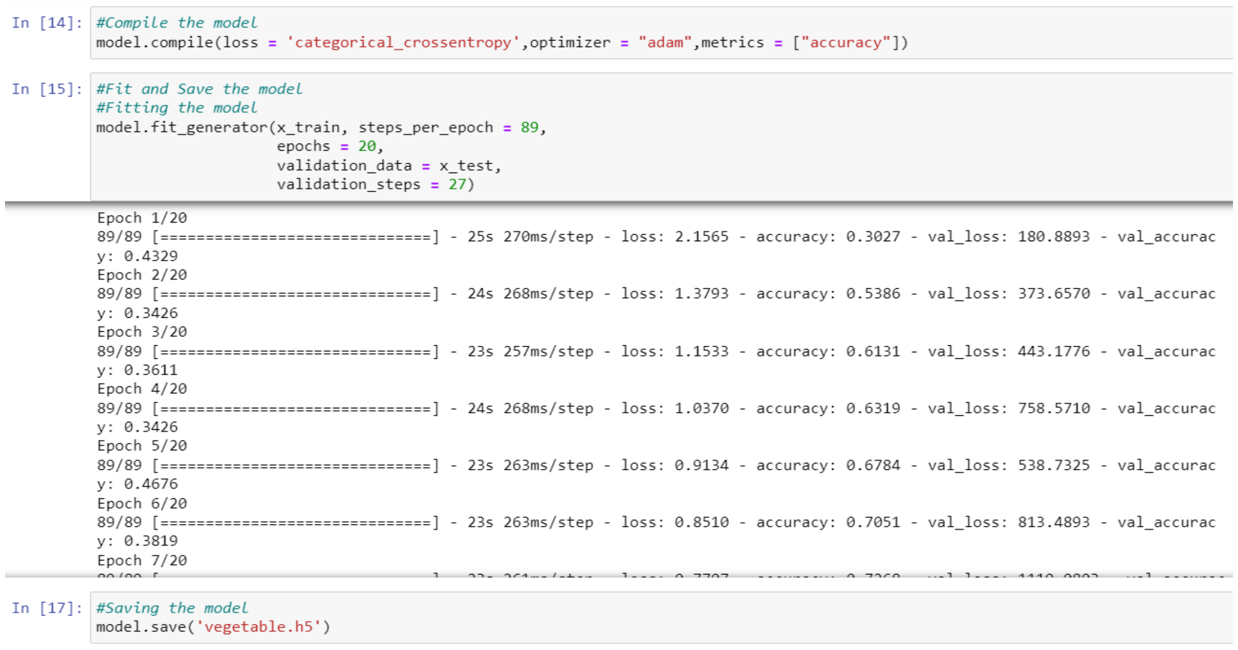
```
In [4]: print(x_train.class_indices)

        {'Pepper,_bell___Bacterial_spot': 0, 'Pepper,_bell___healthy': 1, 'Potato___Early_blight': 2, 'Potato___Late_blight': 3, 'Potat
        o___healthy': 4, 'Tomato___Bacterial_spot': 5, 'Tomato___Late_blight': 6, 'Tomato___Leaf_Mold': 7, 'Tomato___Septoria_leaf_spo
        t': 8}
```

Figure 1 : .ipynb code describing the Image Processing and applying the Image Data Generator Functionality to the Train set and Test set.



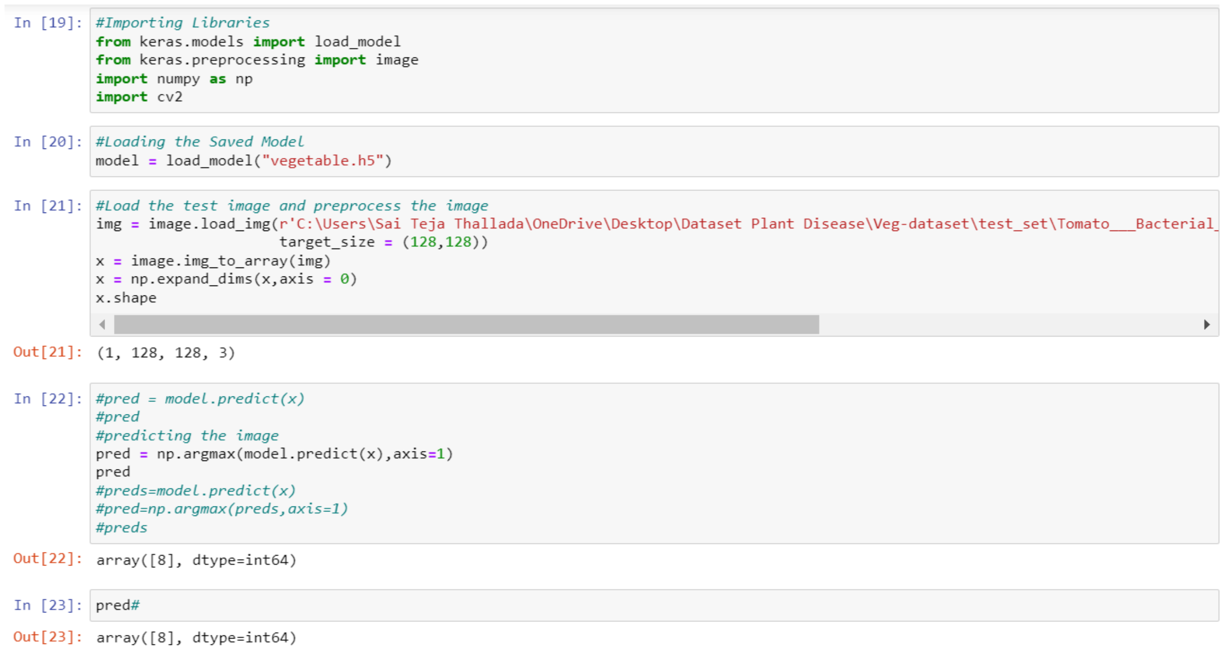
```
In [5]: #Importing Libraries for Model Building
        from keras.models import Sequential    #Importing Sequential Library
        from keras.layers import Dense         #Importing Dense Layer
        from keras.layers import Convolution2D #Importing Convolution2D Layer
        from keras.layers import MaxPooling2D  #Importing MaxPooling2D Layer
        from keras.layers import Flatten       #Importing Flatten Layer
        import warnings
        warnings.filterwarnings('ignore')

In [6]: model = Sequential() #Initializing the model

In [7]: model.add(Convolution2D(32,(3,3), input_shape = (128,128,3), activation = 'relu')) #Adding Convolution2D Layer

In [8]: model.add(MaxPooling2D(pool_size = (2,2))) #Adding MaxPooling2D Layer

In [9]: model.add(Flatten()) #Adding Flatten Layer

In [10]: #model.add(Dense(units = 300, init ='uniform', activation ='relu'))
         model.add(Dense(units = 300 ,activation = 'relu')) #Adding Dense Layer

In [11]: #model.add(Dense(units = 150, init ='uniform', activation ='relu'))
         model.add(Dense(units = 150 ,activation = 'relu')) #Adding Dense Layer

In [12]: #model.add(Dense(units = 75, init ='uniform', activation ='relu'))
         model.add(Dense(units = 75 ,activation = 'relu')) #Adding Dense Layer

In [13]: #model.add(Dense(output_dim = 9,activation = 'softmax',init ='uniform'))
         model.add(Dense(units = 9 ,activation = 'softmax')) #Adding Dense Layer
```

Figure 2: .ipynb code describing Importing Libraries , Initializing and Adding layers to the Model.



```
In [14]: #Compile the model
         model.compile(loss = 'categorical_crossentropy',optimizer = "adam",metrics = ["accuracy"])

In [15]: #Fit and Save the model
         #Fitting the model
         model.fit_generator(x_train, steps_per_epoch = 89,
                             epochs = 20,
                             validation_data = x_test,
                             validation_steps = 27)

         Epoch 1/20
         89/89 [==============================] - 25s 270ms/step - loss: 2.1565 - accuracy: 0.3027 - val_loss: 180.8893 - val_accurac
         y: 0.4329
         Epoch 2/20
         89/89 [==============================] - 24s 268ms/step - loss: 1.3793 - accuracy: 0.5386 - val_loss: 373.6570 - val_accurac
         y: 0.3426
         Epoch 3/20
         89/89 [==============================] - 23s 257ms/step - loss: 1.1533 - accuracy: 0.6131 - val_loss: 443.1776 - val_accurac
         y: 0.3611
         Epoch 4/20
         89/89 [==============================] - 24s 268ms/step - loss: 1.0370 - accuracy: 0.6319 - val_loss: 758.5710 - val_accurac
         y: 0.3426
         Epoch 5/20
         89/89 [==============================] - 23s 263ms/step - loss: 0.9134 - accuracy: 0.6784 - val_loss: 538.7325 - val_accurac
         y: 0.4676
         Epoch 6/20
         89/89 [==============================] - 23s 263ms/step - loss: 0.8510 - accuracy: 0.7051 - val_loss: 813.4893 - val_accurac
         y: 0.3819
         Epoch 7/20

In [17]: #Saving the model
         model.save('vegetable.h5')
```

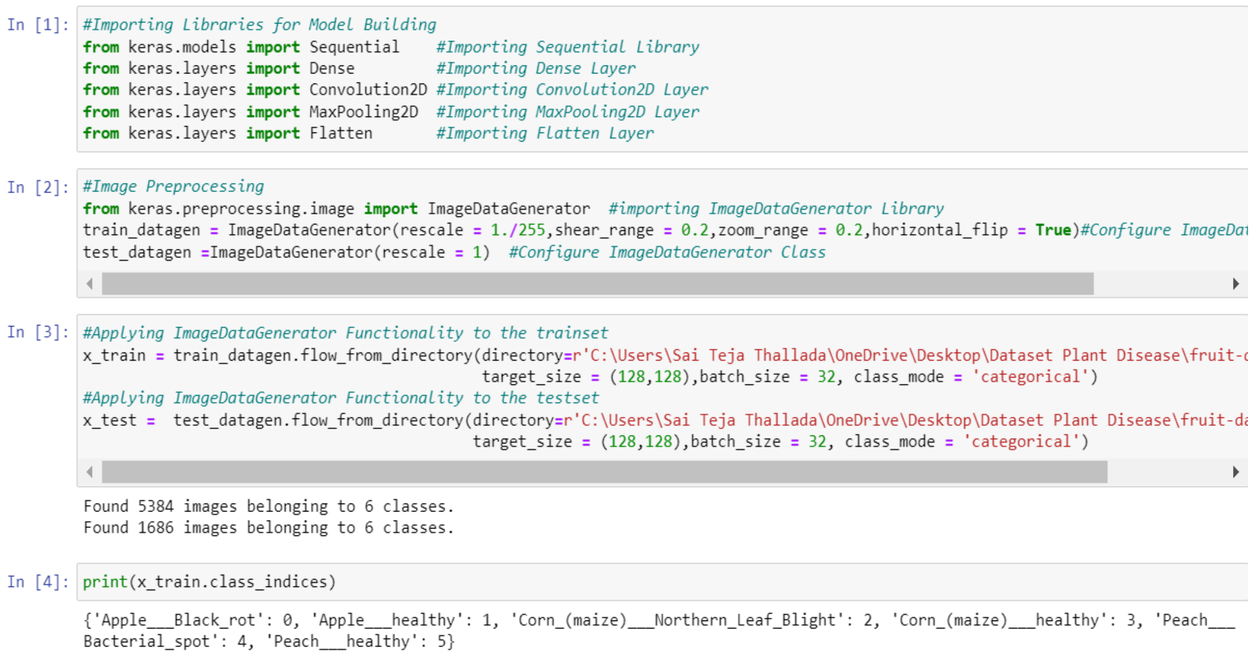
Figure 3: .ipynb code describing Compiling , Fitting and Saving the Model.



```
In [19]: #Importing Libraries
         from keras.models import load_model
         from keras.preprocessing import image
         import numpy as np
         import cv2

In [20]: #Loading the Saved Model
         model = load_model("vegetable.h5")

In [21]: #Load the test image and preprocess the image
         img = image.load_img(r'C:\Users\Sai Teja Thallada\OneDrive\Desktop\Dataset Plant Disease\Veg-dataset\test_set\Tomato___Bacterial_
                              target_size = (128,128))
         x = image.img_to_array(img)
         x = np.expand_dims(x,axis = 0)
         x.shape

Out[21]: (1, 128, 128, 3)

In [22]: #pred = model.predict(x)
         #pred
         #predicting the image
         pred = np.argmax(model.predict(x),axis=1)
         pred
         #preds=model.predict(x)
         #pred=np.argmax(preds,axis=1)
         #preds

Out[22]: array([8], dtype=int64)

In [23]: pred#

Out[23]: array([8], dtype=int64)
```

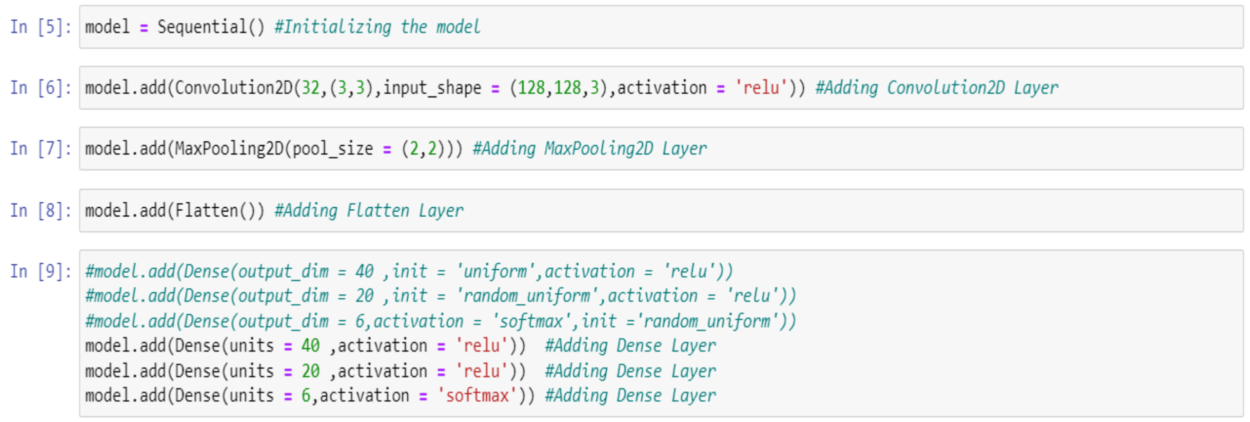
Figure 4: .ipynb code describing Testing the Saved Model.

2.2 Fruit Model code:



```
In [1]: #Importing Libraries for Model Building
        from keras.models import Sequential    #Importing Sequential Library
        from keras.layers import Dense         #Importing Dense Layer
        from keras.layers import Convolution2D #Importing Convolution2D Layer
        from keras.layers import MaxPooling2D  #Importing MaxPooling2D Layer
        from keras.layers import Flatten       #Importing Flatten Layer

In [2]: #Image Preprocessing
        from keras.preprocessing.image import ImageDataGenerator  #importing ImageDataGenerator Library
        train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,zoom_range = 0.2,horizontal_flip = True)#Configure ImageDat
        test_datagen =ImageDataGenerator(rescale = 1)  #Configure ImageDataGenerator Class

In [3]: #Applying ImageDataGenerator Functionality to the trainset
        x_train = train_datagen.flow_from_directory(directory=r'C:\Users\Sai Teja Thallada\OneDrive\Desktop\Dataset Plant Disease\fruit-c
                                                    target_size = (128,128),batch_size = 32, class_mode = 'categorical')
        #Applying ImageDataGenerator Functionality to the testset
        x_test =  test_datagen.flow_from_directory(directory=r'C:\Users\Sai Teja Thallada\OneDrive\Desktop\Dataset Plant Disease\fruit-da
                                                   target_size = (128,128),batch_size = 32, class_mode = 'categorical')

        Found 5384 images belonging to 6 classes.
        Found 1686 images belonging to 6 classes.

In [4]: print(x_train.class_indices)

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

Figure 5 : .ipynb code describing Importing Libraries, Image Processing and applying the Image Data Generator Functionality to the Train set and Test set.



```
In [5]: model = Sequential() #Initializing the model

In [6]: model.add(Convolution2D(32,(3,3),input_shape = (128,128,3),activation = 'relu')) #Adding Convolution2D Layer

In [7]: model.add(MaxPooling2D(pool_size = (2,2))) #Adding MaxPooling2D Layer

In [8]: model.add(Flatten()) #Adding Flatten Layer

In [9]: #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'))
        model.add(Dense(units = 40 ,activation = 'relu'))  #Adding Dense Layer
        model.add(Dense(units = 20 ,activation = 'relu'))  #Adding Dense Layer
        model.add(Dense(units = 6,activation = 'softmax')) #Adding Dense Layer
```

Figure 6: .ipynb code describing the Initialization and Adding layers to the Model.

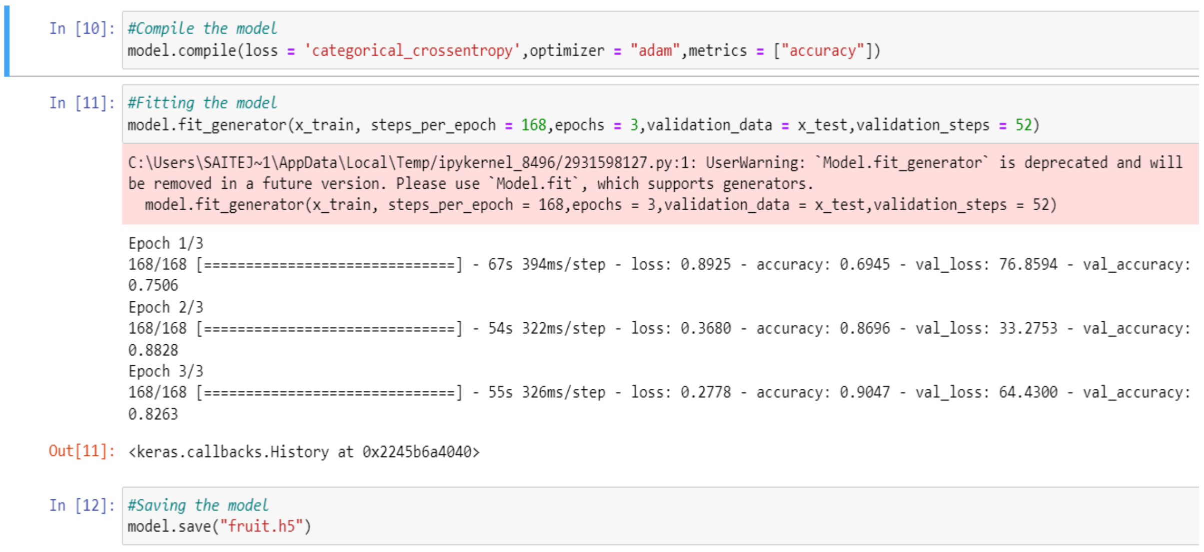

```
In [10]: #Compile the model
         model.compile(loss = 'categorical_crossentropy',optimizer = "adam",metrics = ["accuracy"])

In [11]: #Fitting the model
         model.fit_generator(x_train, steps_per_epoch = 168,epochs = 3,validation_data = x_test,validation_steps = 52)

         C:\Users\SAITEJ~1\AppData\Local\Temp/ipykernel_8496/2931598127.py:1: UserWarning: `Model.fit_generator` is deprecated and will
         be removed in a future version. Please use `Model.fit`, which supports generators.
           model.fit_generator(x_train, steps_per_epoch = 168,epochs = 3,validation_data = x_test,validation_steps = 52)

         Epoch 1/3
         168/168 [==============================] - 67s 394ms/step - loss: 0.8925 - accuracy: 0.6945 - val_loss: 76.8594 - val_accuracy:
         0.7506
         Epoch 2/3
         168/168 [==============================] - 54s 322ms/step - loss: 0.3680 - accuracy: 0.8696 - val_loss: 33.2753 - val_accuracy:
         0.8828
         Epoch 3/3
         168/168 [==============================] - 55s 326ms/step - loss: 0.2778 - accuracy: 0.9047 - val_loss: 64.4300 - val_accuracy:
         0.8263

Out[11]: <keras.callbacks.History at 0x2245b6a4040>

In [12]: #Saving the model
         model.save("fruit.h5")
```

Figure 7: .ipynb code describing Compiling , Fitting and Saving the Model.

2.3 Plant model code:

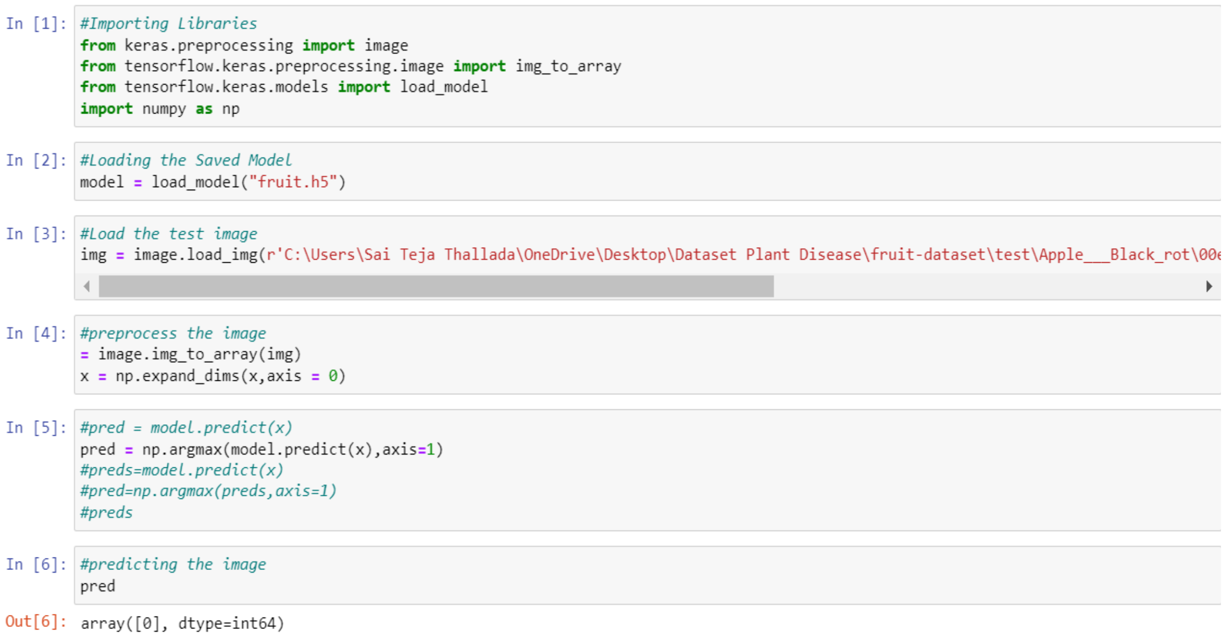
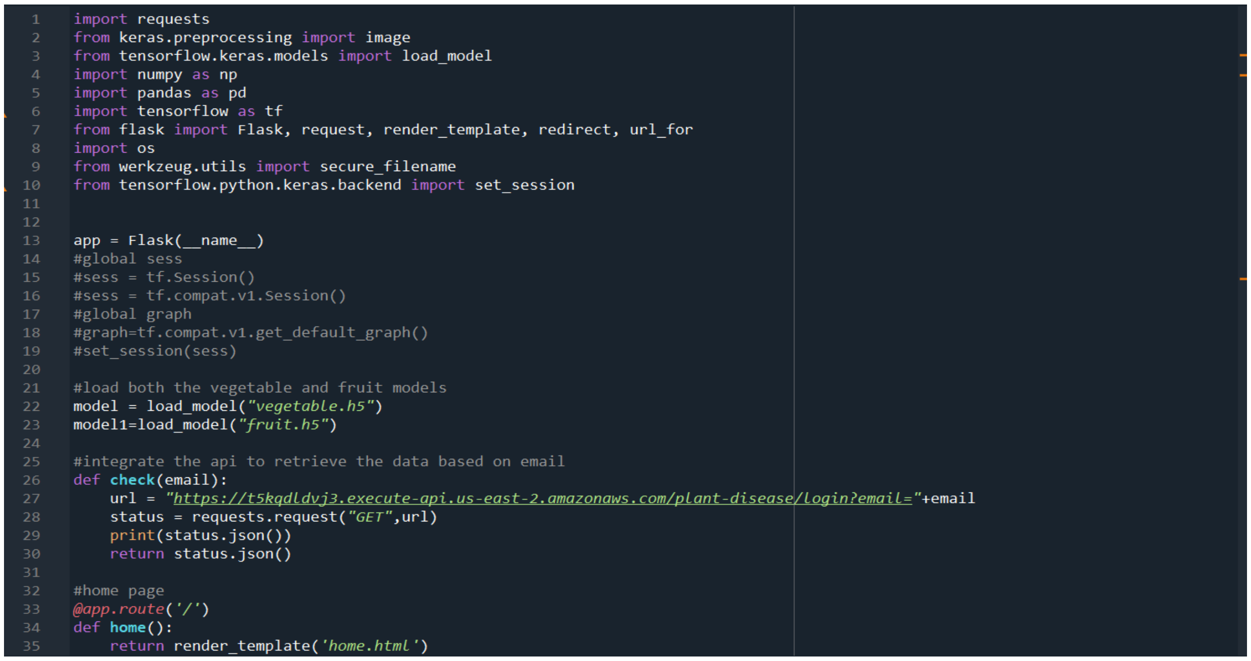

```
In [1]: #Importing Libraries
        from keras.preprocessing import image
        from tensorflow.keras.preprocessing.image import img_to_array
        from tensorflow.keras.models import load_model
        import numpy as np

In [2]: #Loading the Saved Model
        model = load_model("fruit.h5")

In [3]: #Load the test image
        img = image.load_img(r'C:\Users\Sai Teja Thallada\OneDrive\Desktop\Dataset Plant Disease\fruit-dataset\test\Apple___Black_rot\00e

In [4]: #preprocess the image
        = image.img_to_array(img)
        x = np.expand_dims(x,axis = 0)

In [5]: #pred = model.predict(x)
        pred = np.argmax(model.predict(x),axis=1)
        #preds=model.predict(x)
        #pred=np.argmax(preds,axis=1)
        #preds

In [6]: #predicting the image
        pred

Out[6]: array([0], dtype=int64)
```

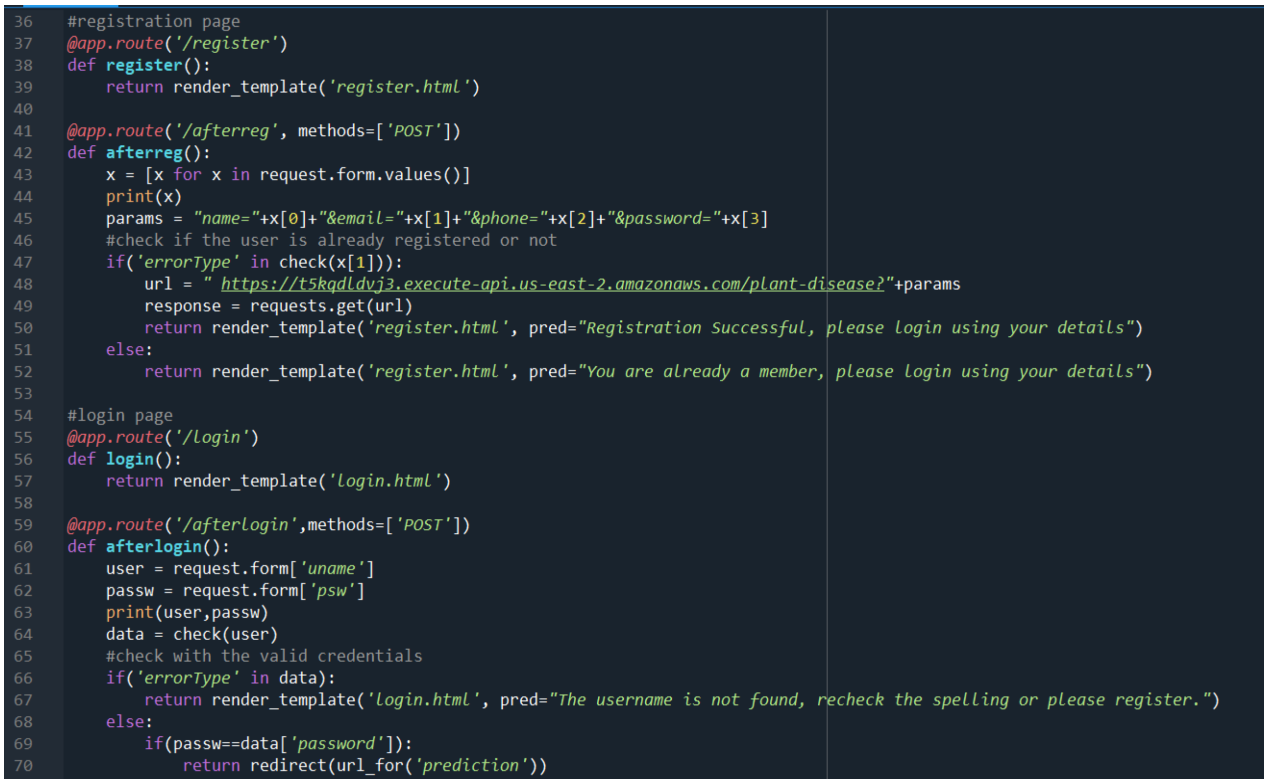
Figure 8: .ipynb code describing Testing the Saved Model.

3.HTML CODE AND PYTHON CODE:

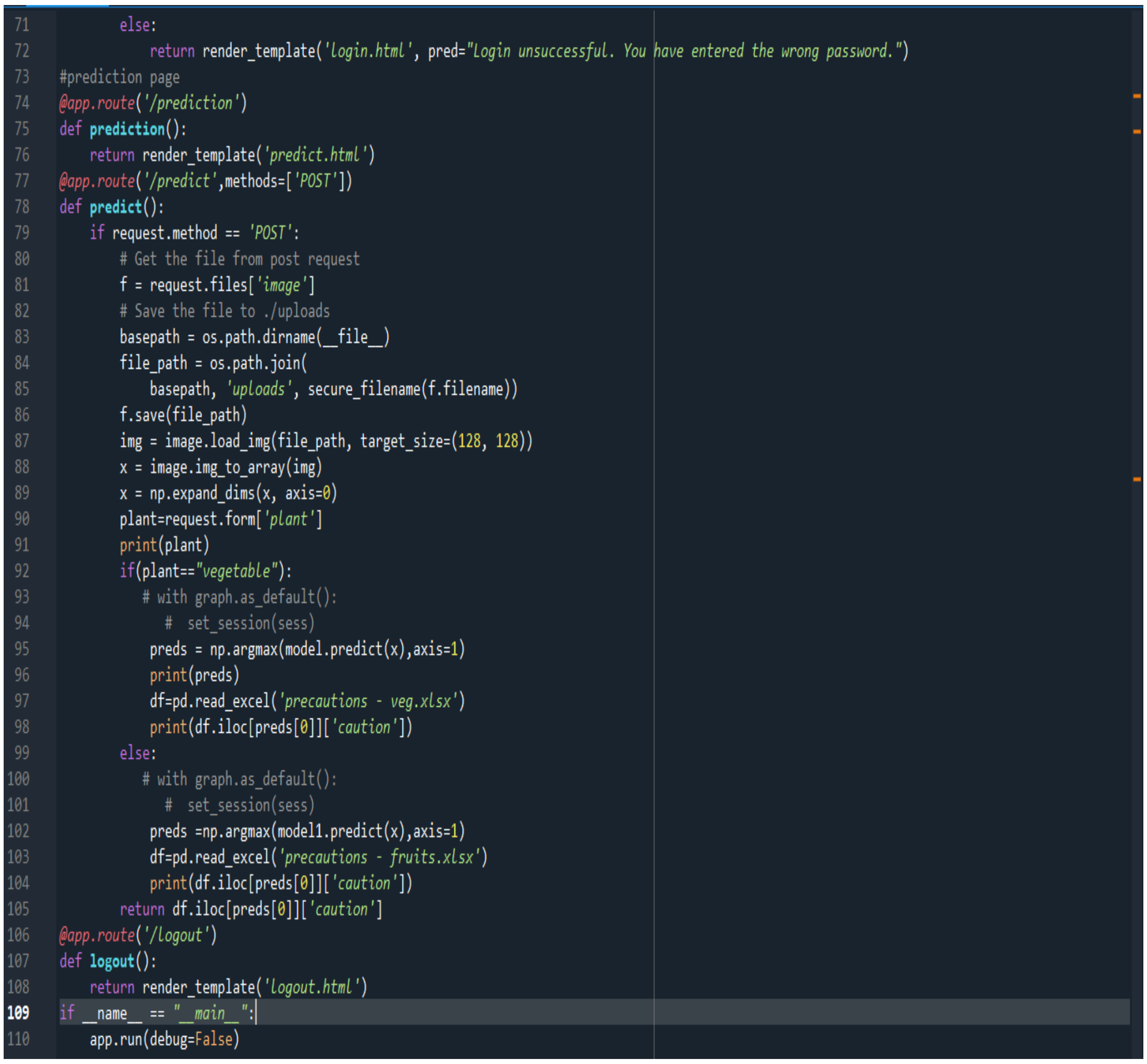
3.1 App.py code:



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


app = Flask(__name__)
#global sess
#sess = tf.Session()
#sess = tf.compat.v1.Session()
#global graph
#graph=tf.compat.v1.get_default_graph()
#set_session(sess)

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

#integrate the api to retrieve the data based on email
def check(email):
    url = "https://t5kqdldvj3.execute-api.us-east-2.amazonaws.com/plant-disease/login?email="+email
    status = requests.request("GET",url)
    print(status.json())
    return status.json()

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



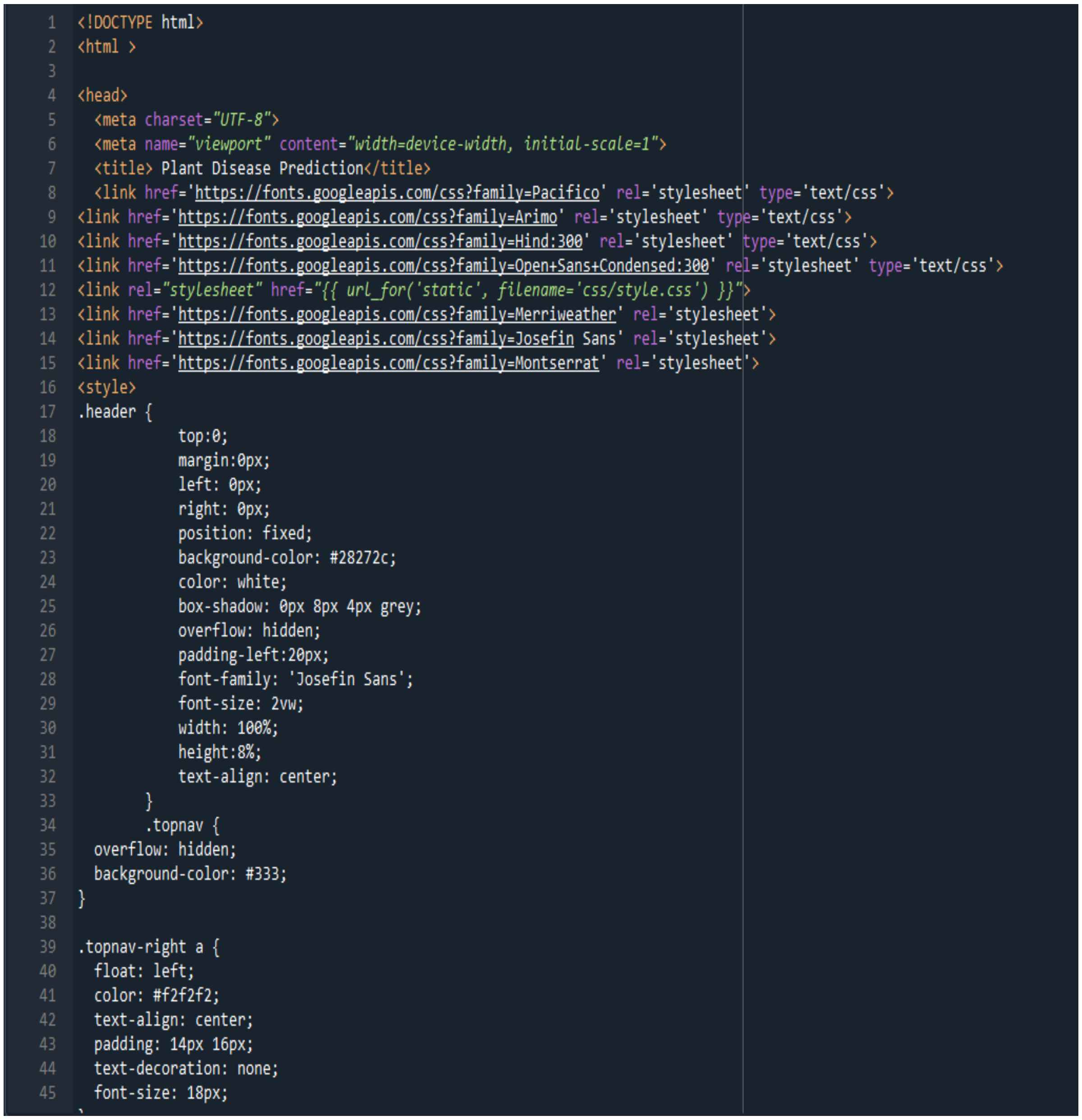
```python
#registration page
@app.route('/register')
def register():
    return render_template('register.html')

@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    params = "name="+x[0]+"&email="+x[1]+"&phone="+x[2]+"&password="+x[3]
    #check if the user is already registered or not
    if('errorType' in check(x[1])):
        url = " https://t5kqdldvj3.execute-api.us-east-2.amazonaws.com/plant-disease?"+params
        response = requests.get(url)
        return render_template('register.html', pred="Registration Successful, please login using your details")
    else:
        return render_template('register.html', pred="You are already a member, please login using your details")

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

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    user = request.form['uname']
    passw = request.form['psw']
    print(user,passw)
    data = check(user)
    #check with the valid credentials
    if('errorType' in data):
        return render_template('login.html', pred="The username is not found, recheck the spelling or please register.")
    else:
        if(passw==data['password']):
            return redirect(url_for('prediction'))
```



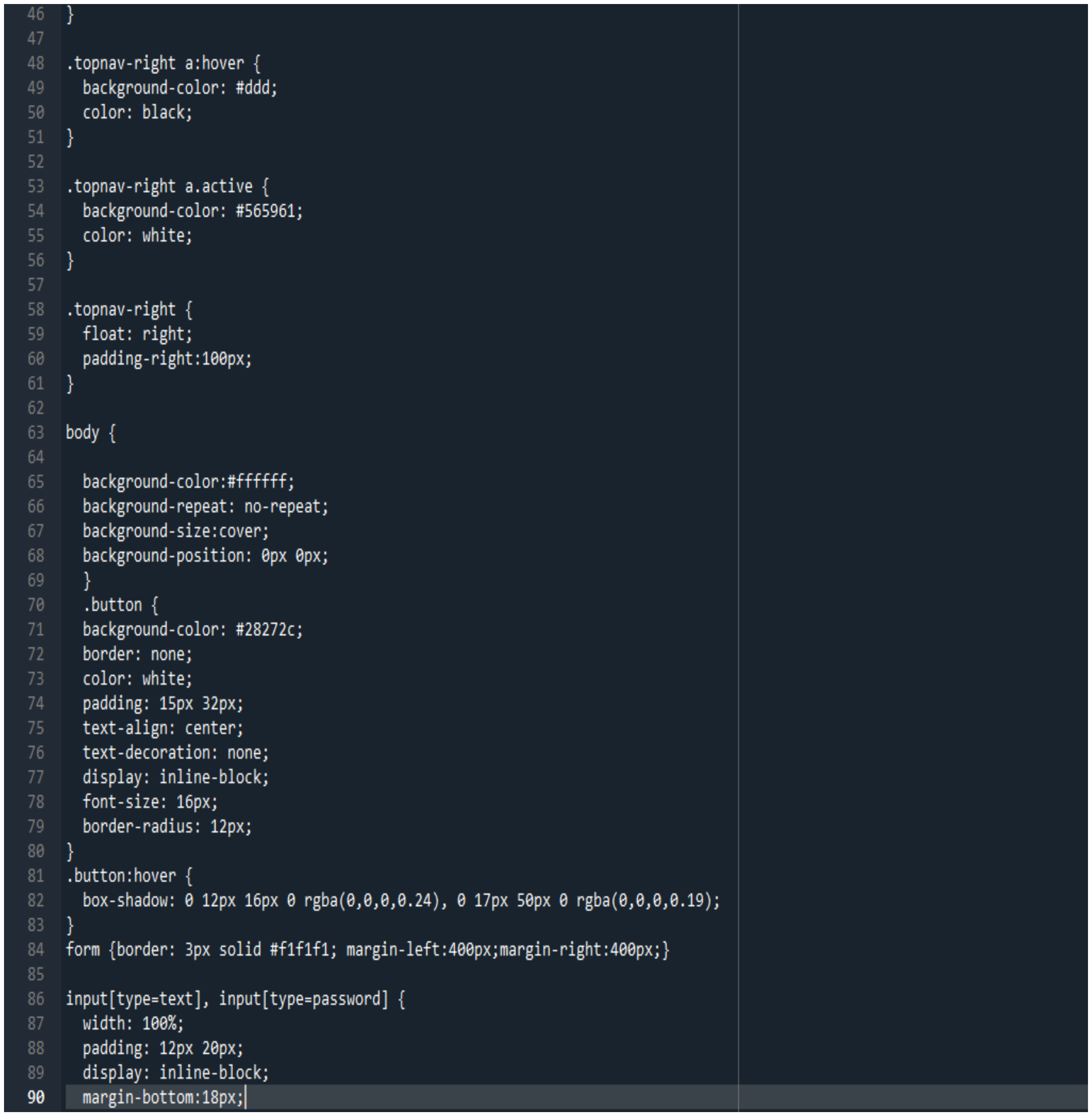
```python
        else:
            return render_template('login.html', pred="Login unsuccessful. You have entered the wrong password.")
#prediction page
@app.route('/prediction')
def prediction():
    return render_template('predict.html')
@app.route('/predict',methods=['POST'])
def predict():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']
        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(128, 128))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        plant=request.form['plant']
        print(plant)
        if(plant=="vegetable"):
          # with graph.as_default():
             #  set_session(sess)
            preds = np.argmax(model.predict(x),axis=1)
            print(preds)
            df=pd.read_excel('precautions - veg.xlsx')
            print(df.iloc[preds[0]]['caution'])
        else:
          # with graph.as_default():
             #  set_session(sess)
            preds =np.argmax(model1.predict(x),axis=1)
            df=pd.read_excel('precautions - fruits.xlsx')
            print(df.iloc[preds[0]]['caution'])
        return df.iloc[preds[0]]['caution']
@app.route('/logout')
def logout():
    return render_template('logout.html')
if __name__ == "__main__":
    app.run(debug=False)
```

Figure 9: python code used for rendering all the HTML pages.

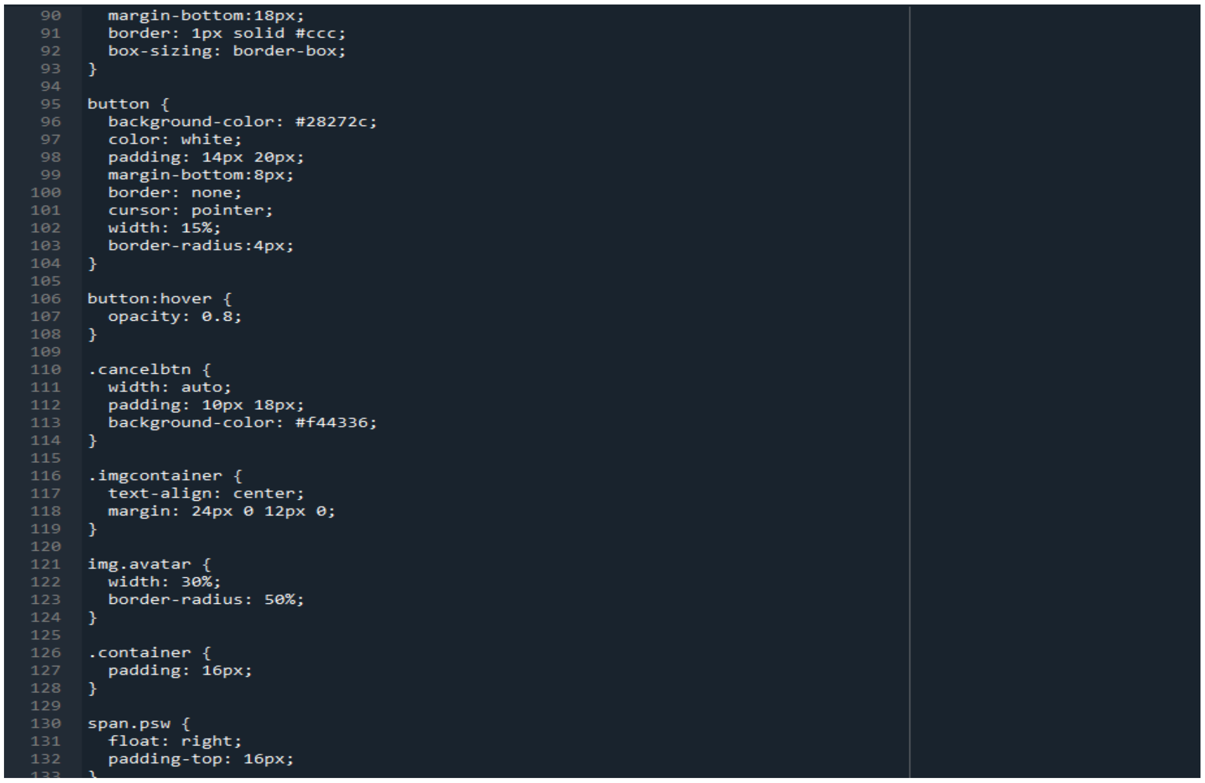
3.2 Home.html:



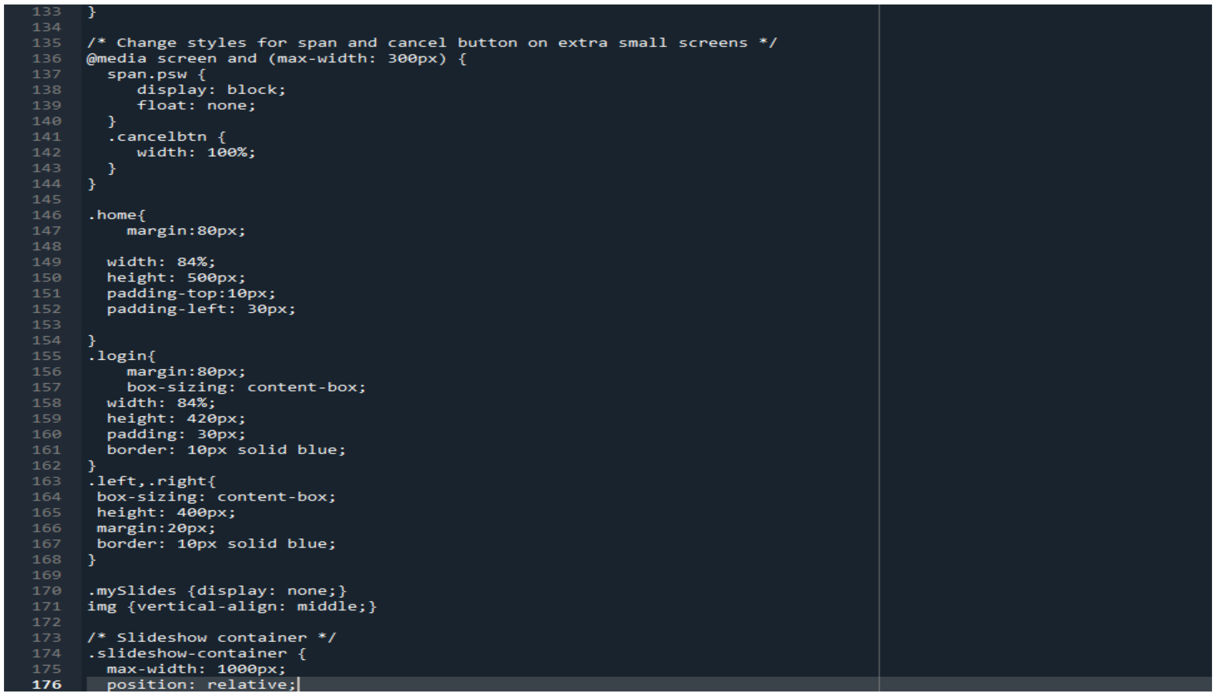
```
<!DOCTYPE html>
<html >

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

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



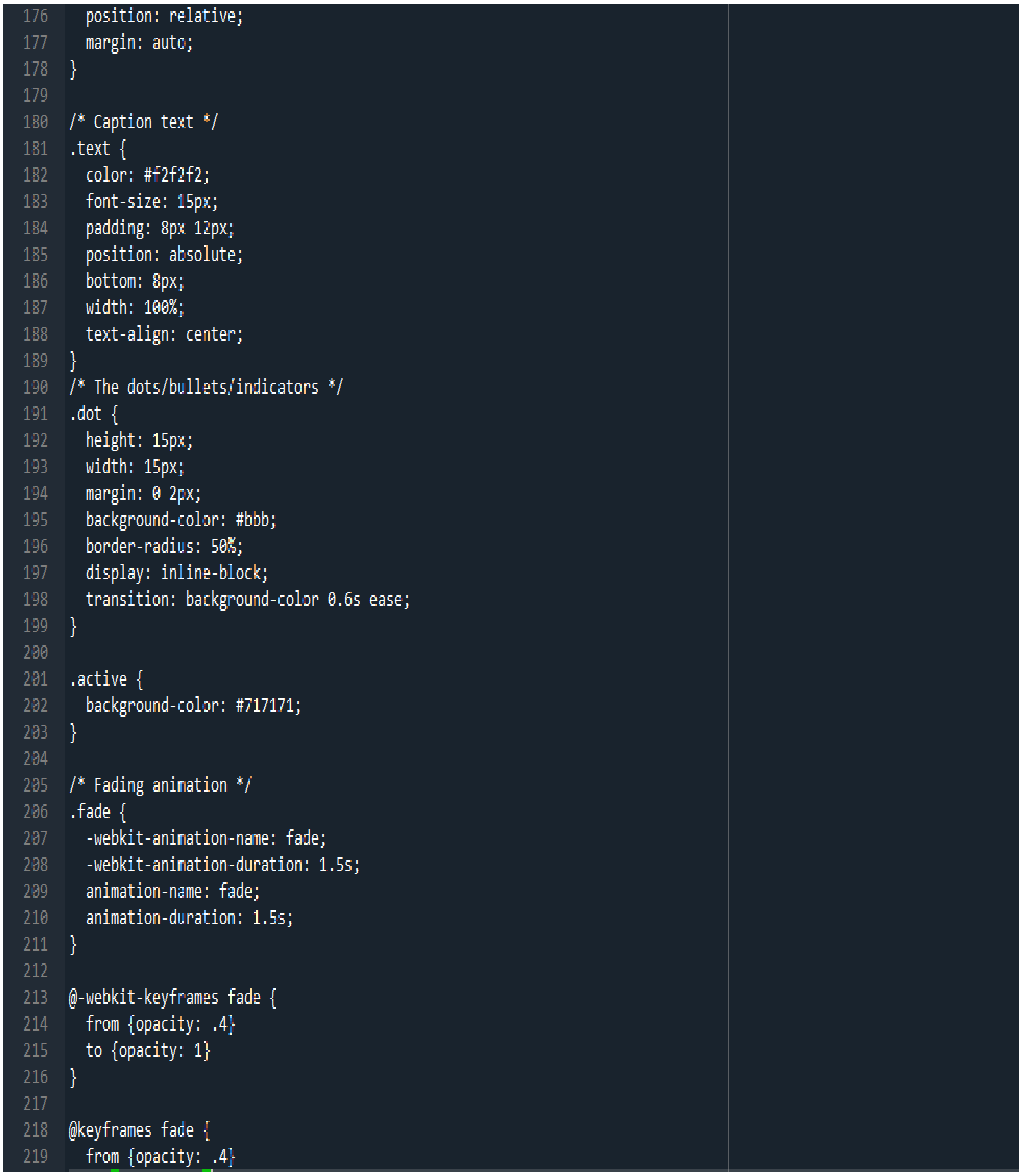
```
}

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

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

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

body {

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

input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
```



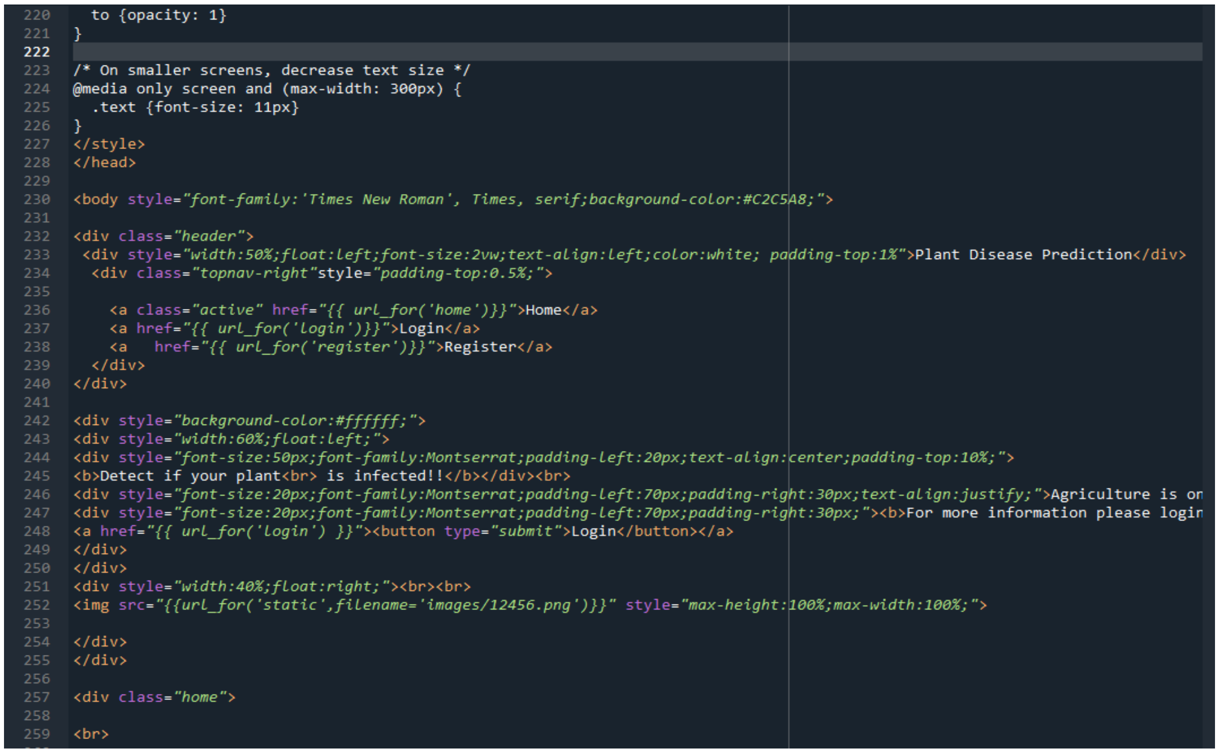
```
  margin-bottom:18px;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

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

button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

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

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

.container {
  padding: 16px;
}

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



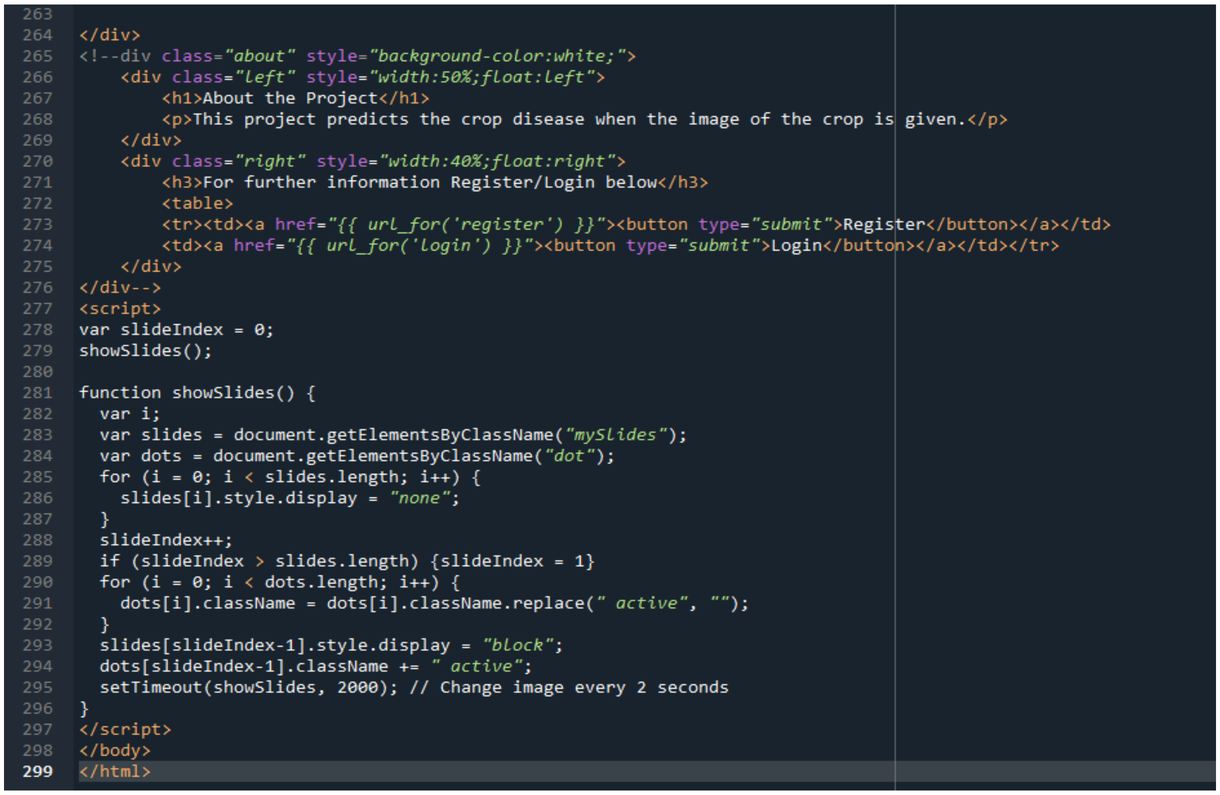
```
}

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

.home{
    margin:80px;

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

}
.login{
    margin:80px;
    box-sizing: content-box;
  width: 84%;
  height: 420px;
  padding: 30px;
  border: 10px solid blue;
}
.left,.right{
 box-sizing: content-box;
 height: 400px;
 margin:20px;
 border: 10px solid blue;
}

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

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



```
  position: relative;
  margin: auto;
}

/* Caption text */
.text {
  color: #f2f2f2;
  font-size: 15px;
  padding: 8px 12px;
  position: absolute;
  bottom: 8px;
  width: 100%;
  text-align: center;
}
/* The dots/bullets/indicators */
.dot {
  height: 15px;
  width: 15px;
  margin: 0 2px;
  background-color: #bbb;
  border-radius: 50%;
  display: inline-block;
  transition: background-color 0.6s ease;
}

.active {
  background-color: #717171;
}

/* Fading animation */
.fade {
  -webkit-animation-name: fade;
  -webkit-animation-duration: 1.5s;
  animation-name: fade;
  animation-duration: 1.5s;
}

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

@keyframes fade {
  from {opacity: .4}
```



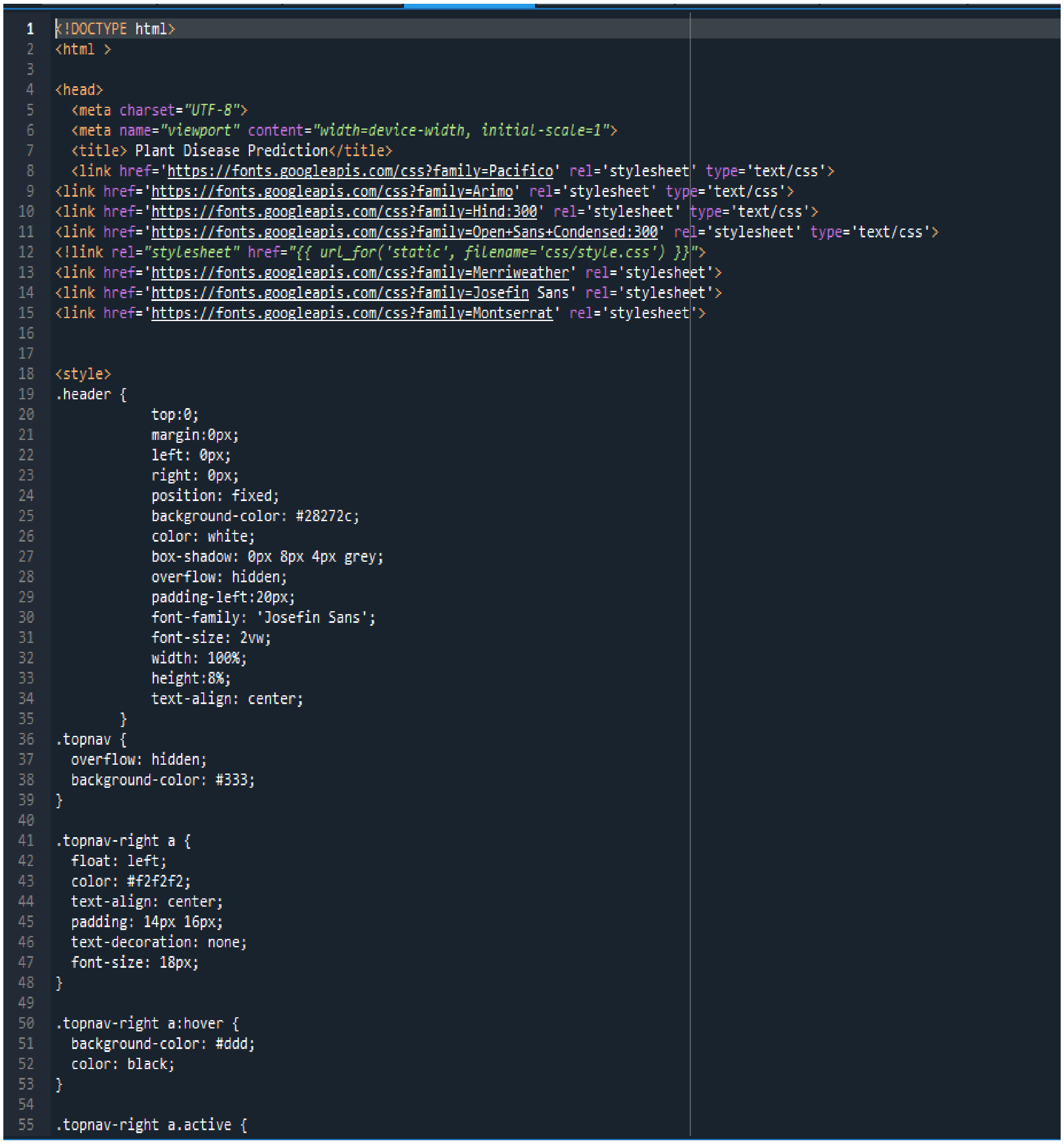
```
  to {opacity: 1}
}

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

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

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

    <a class="active" href="{{ url_for('home')}}">Home</a>
    <a href="{{ url_for('login')}}">Login</a>
    <a   href="{{ url_for('register')}}">Register</a>
  </div>
</div>

<div style="background-color:#ffffff;">
<div style="width:60%;float:left;">
<div style="font-size:50px;font-family:Montserrat;padding-left:20px;text-align:center;padding-top:10%;">
<b>Detect if your plant<br> is infected!!</b></div><br>
<div style="font-size:20px;font-family:Montserrat;padding-left:70px;padding-right:30px;text-align:justify;">Agriculture is on
<div style="font-size:20px;font-family:Montserrat;padding-left:70px;padding-right:30px;"><b>For more information please login
<a href="{{ url_for('login') }}"><button type="submit">Login</button></a>
</div>
</div>
<div style="width:40%;float:right;"><br><br>
<img src="{{url_for('static',filename='images/12456.png')}}" style="max-height:100%;max-width:100%;">

</div>
</div>

<div class="home">

<br>
```



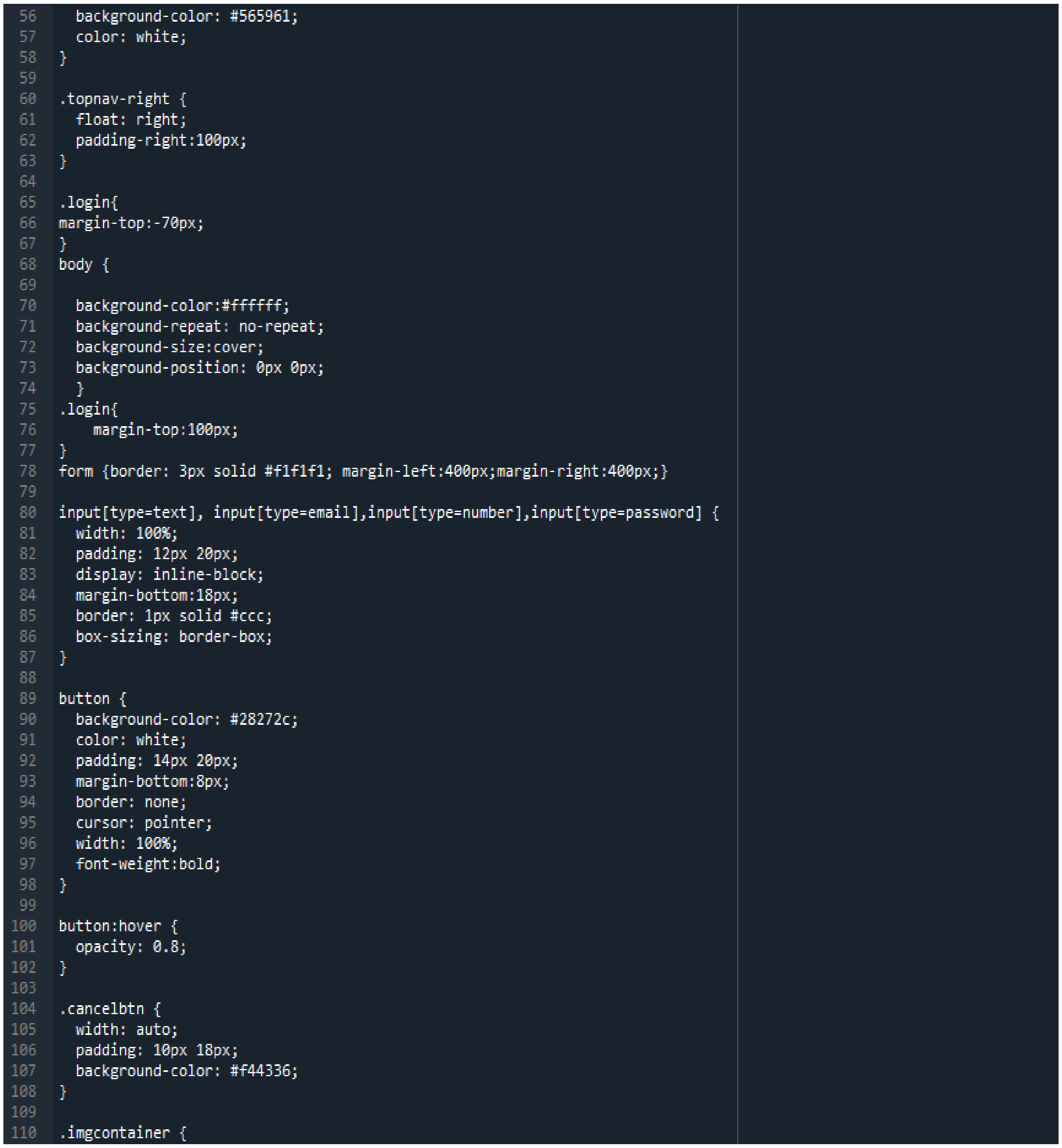
```

</div>
<!--div class="about" style="background-color:white;">
    <div class="left" style="width:50%;float:left">
        <h1>About the Project</h1>
        <p>This project predicts the crop disease when the image of the crop is given.</p>
    </div>
    <div class="right" style="width:40%;float:right">
        <h3>For further information Register/Login below</h3>
        <table>
        <tr><td><a href="{{ url_for('register') }}"><button type="submit">Register</button></a></td>
        <td><a href="{{ url_for('login') }}"><button type="submit">Login</button></a></td></tr>
    </div>
</div-->
<script>
var slideIndex = 0;
showSlides();

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

Figure 10: home.html page is the code for home page of our Web Application.

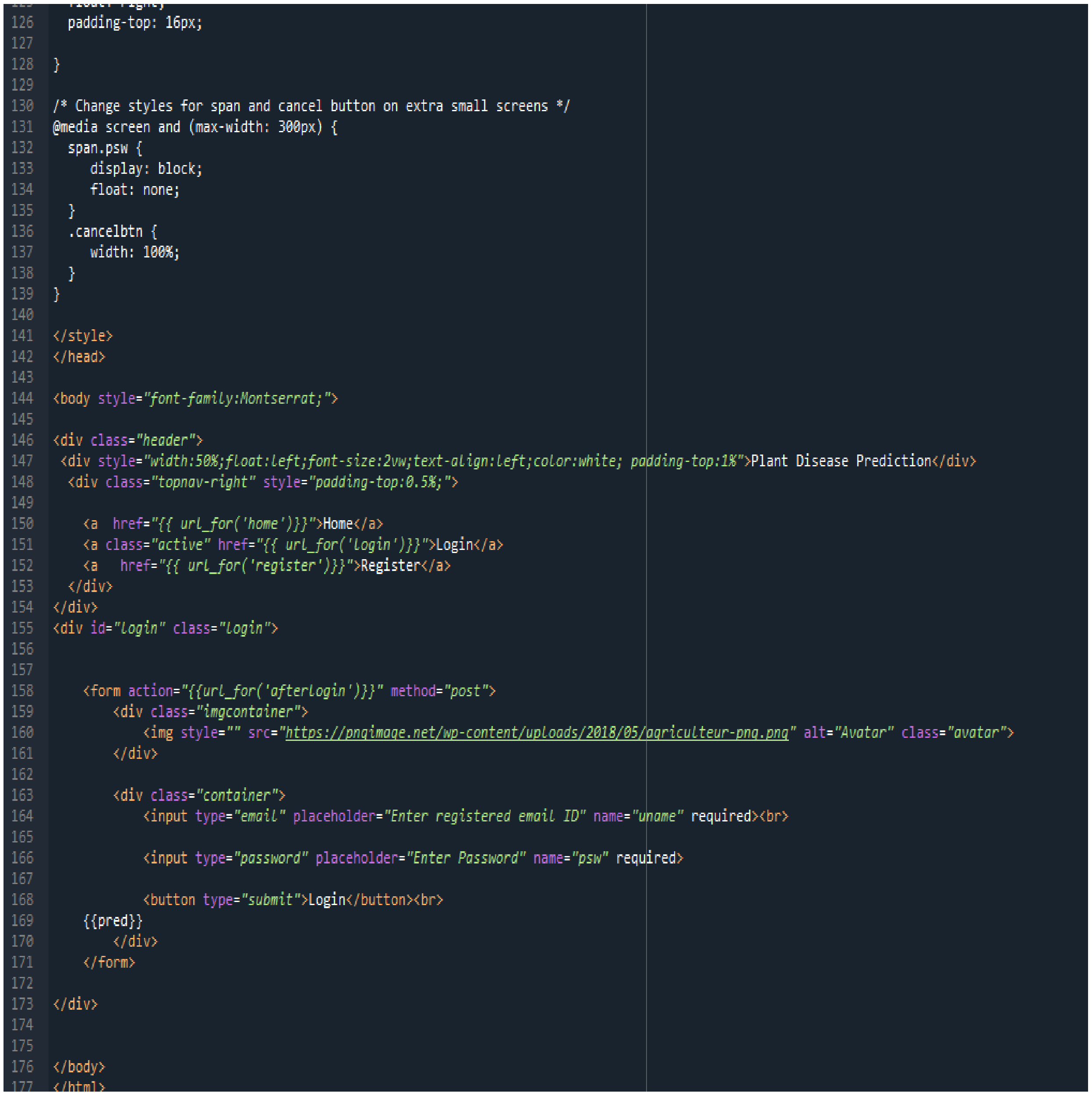
3.3. Login.html:



```
<!DOCTYPE html>
<html >

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


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

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

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

.topnav-right a.active {
```



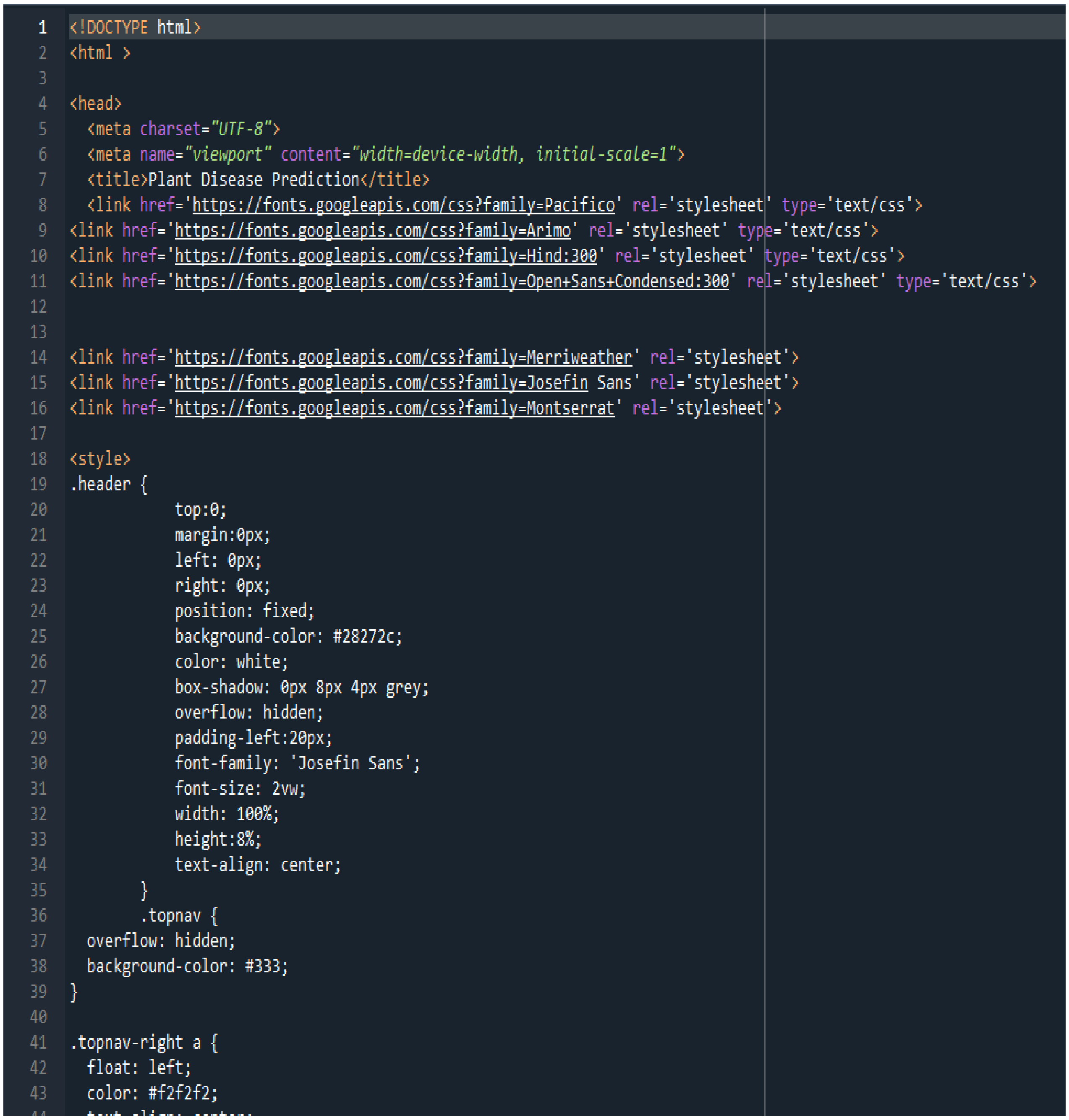
```
  background-color: #565961;
  color: white;
}

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

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

  background-color:#ffffff;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  }
.login{
    margin-top:100px;
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

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

button {
  background-color: #28272c;
  color: white;
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
  cursor: pointer;
  width: 100%;
  font-weight:bold;
}

button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

.imgcontainer {
```



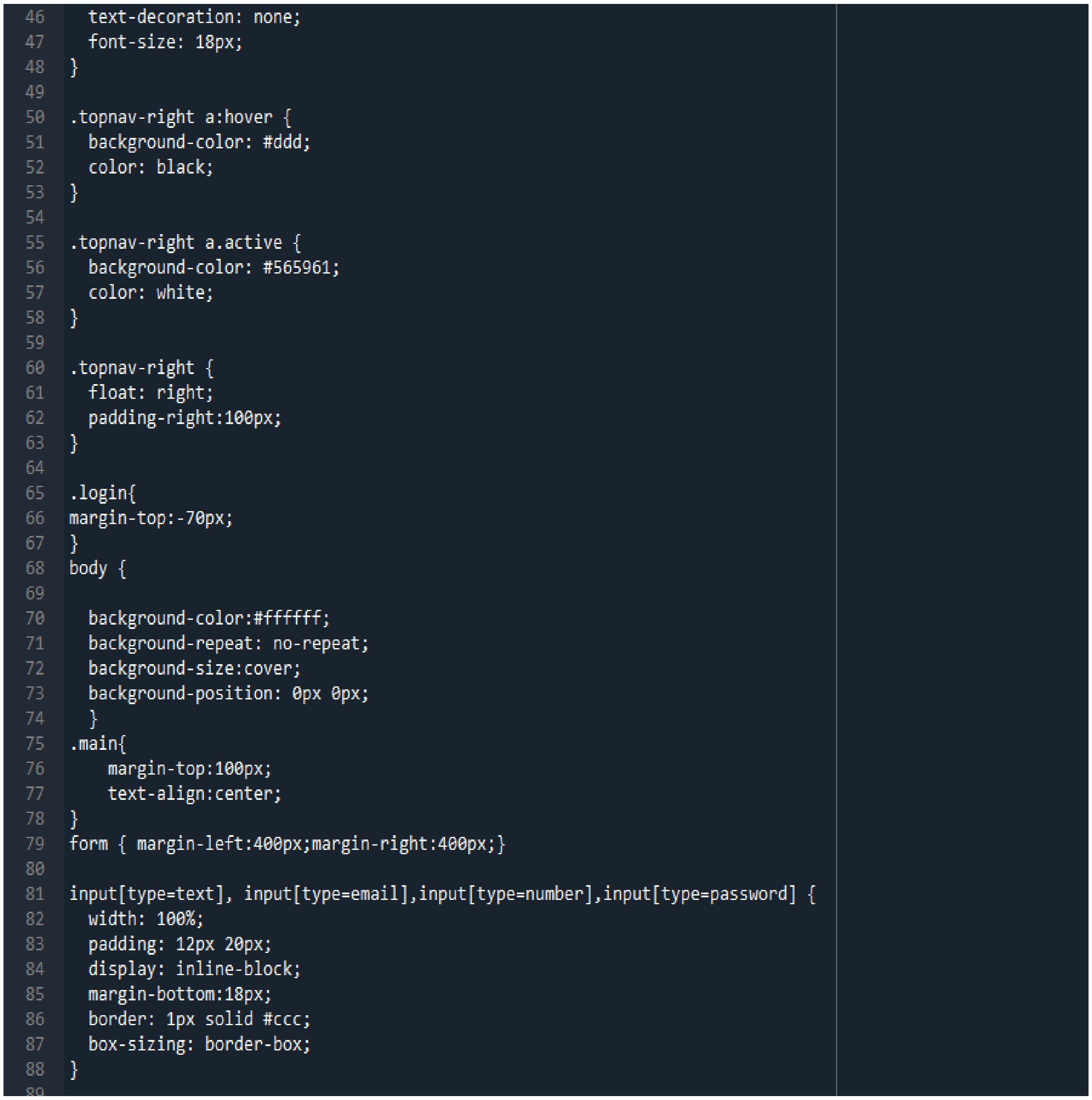
```
  float: right;
  padding-top: 16px;

}

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

</style>
</head>

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

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

    <a  href="{{ url_for('home')}}">Home</a>
    <a class="active" href="{{ url_for('login')}}">Login</a>
    <a   href="{{ url_for('register')}}">Register</a>
  </div>
</div>
<div id="login" class="login">


    <form action="{{url_for('afterlogin')}}" method="post">
        <div class="imgcontainer">
            <img style="" src="https://pngimage.net/wp-content/uploads/2018/05/agriculteur-png.png" alt="Avatar" class="avatar">
        </div>

        <div class="container">
            <input type="email" placeholder="Enter registered email ID" name="uname" required><br>

            <input type="password" placeholder="Enter Password" name="psw" required>

            <button type="submit">Login</button><br>
    {{pred}}
        </div>
    </form>

</div>


</body>
</html>
```

Figure 11: login.html page is the code for home page of our Web Application.

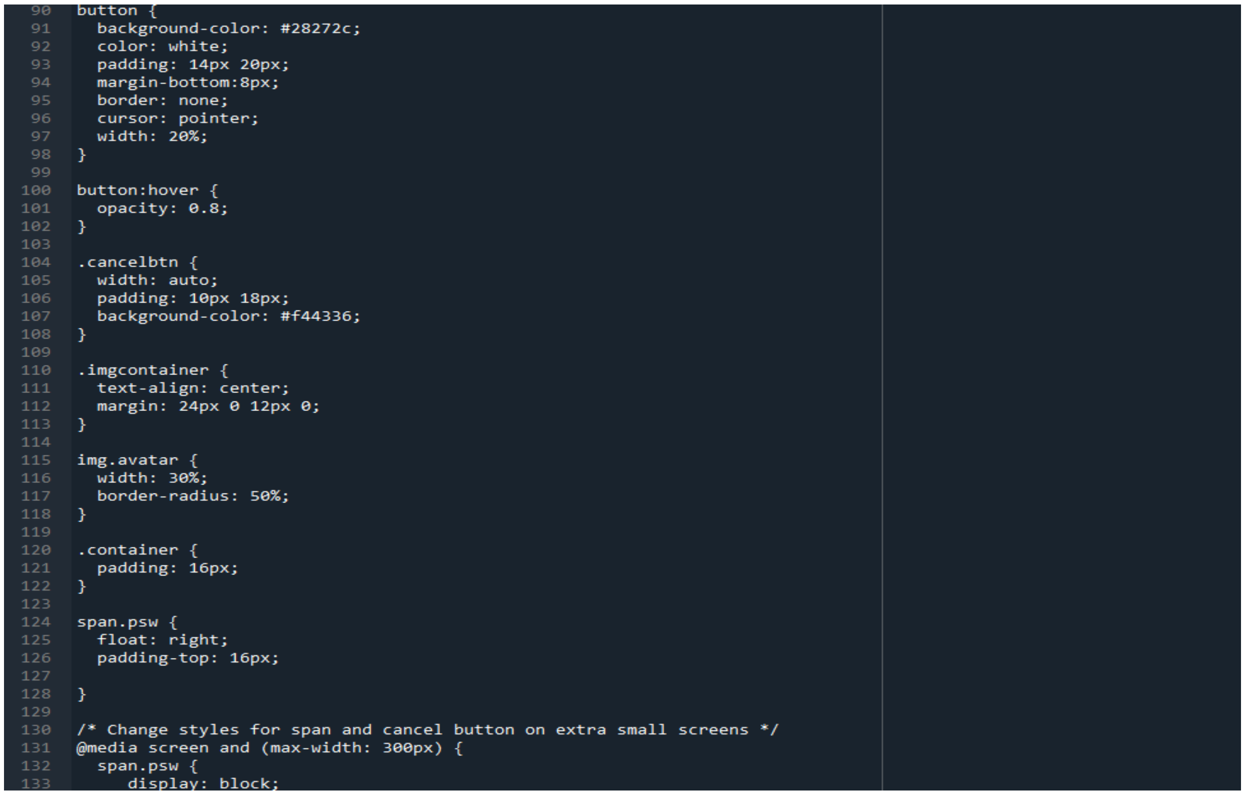
3.4. Logout.html:



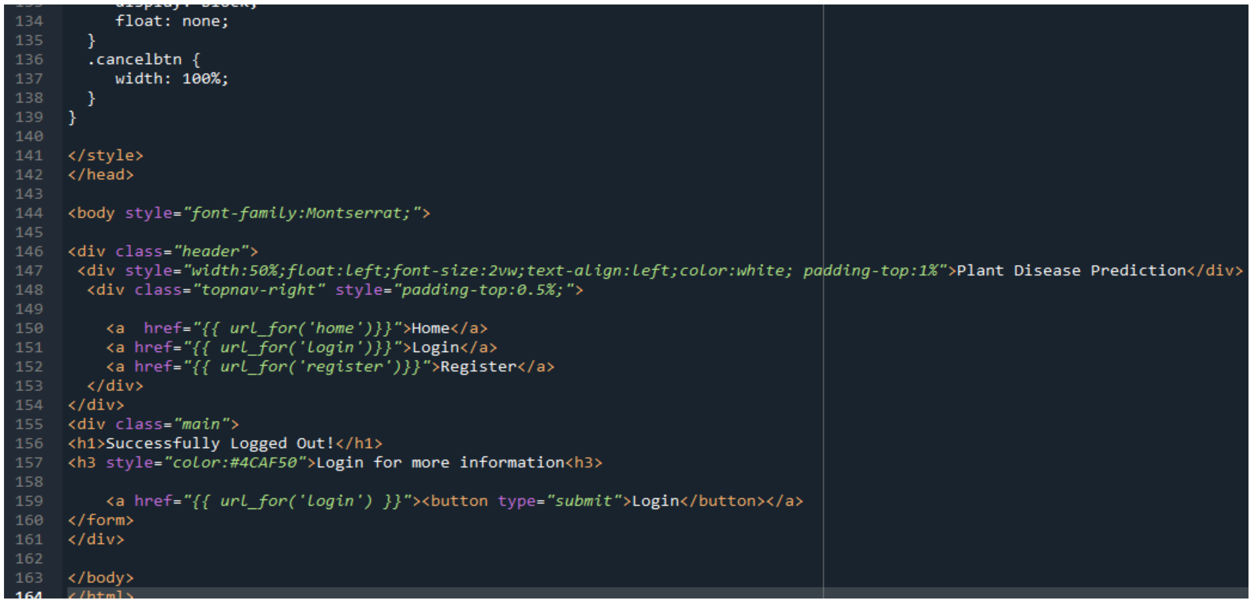
```
<!DOCTYPE html>
<html >

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


<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>

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

.topnav-right a {
  float: left;
  color: #f2f2f2;
```



```
  text-decoration: none;
  font-size: 18px;
}

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

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

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

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

  background-color:#ffffff;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  }
.main{
    margin-top:100px;
    text-align:center;
}
form { margin-left:400px;margin-right:400px;}

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



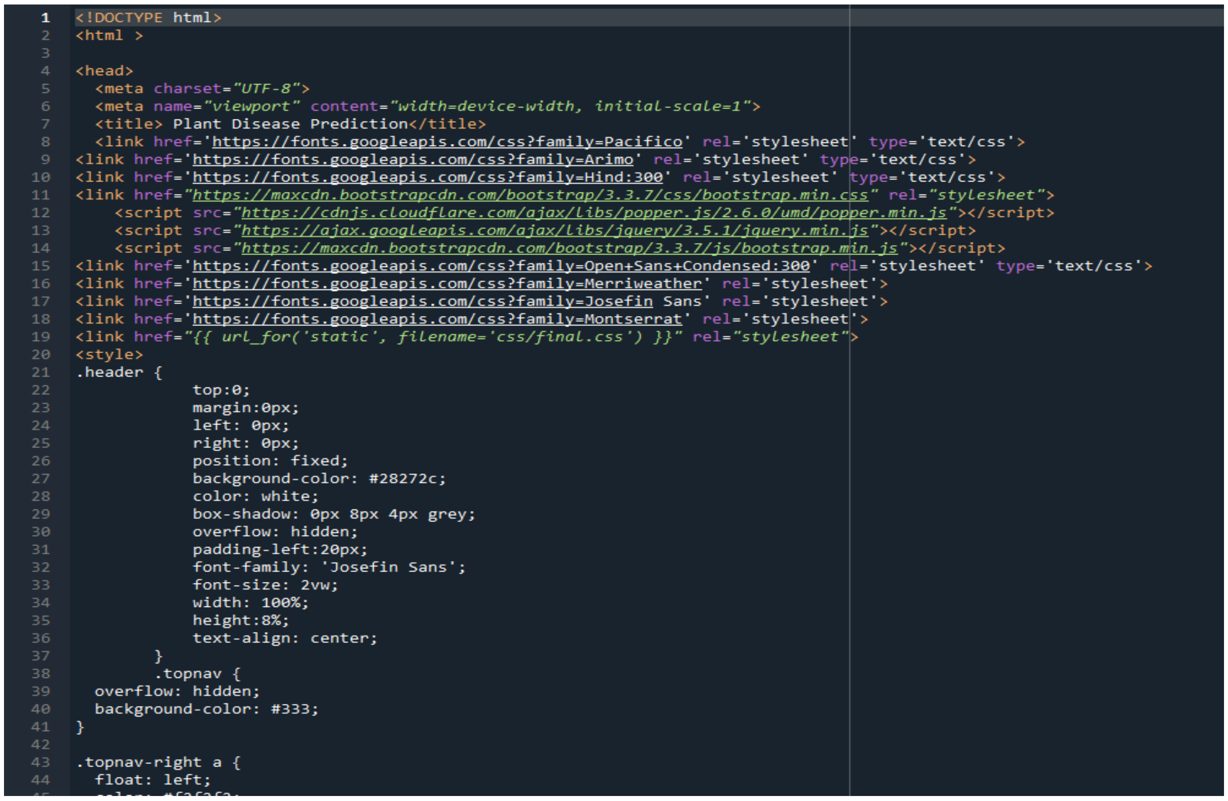
```
button {
  background-color: #28272c;
  color: white;
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
  cursor: pointer;
  width: 20%;
}

button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

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

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

.container {
  padding: 16px;
}

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

}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
  span.psw {
     display: block;
```



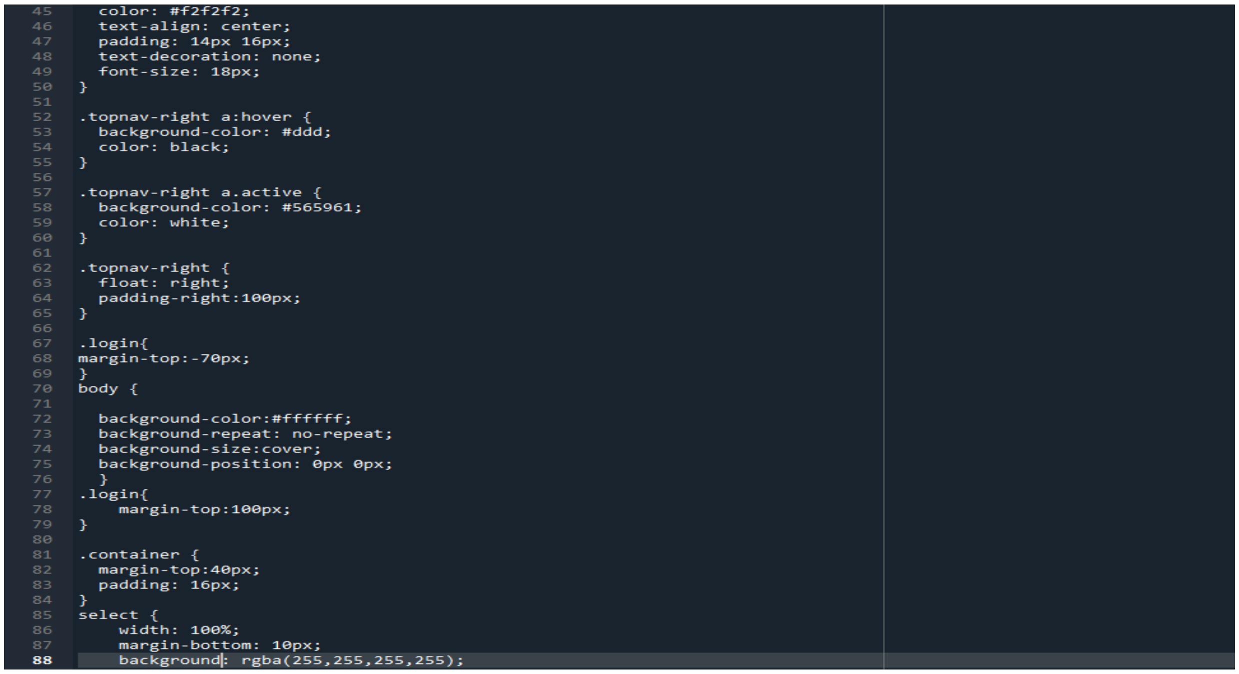
```
     float: none;
  }
  .cancelbtn {
     width: 100%;
  }
}

</style>
</head>

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

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

    <a  href="{{ url_for('home')}}">Home</a>
    <a href="{{ url_for('login')}}">Login</a>
    <a href="{{ url_for('register')}}">Register</a>
  </div>
</div>
<div class="main">
<h1>Successfully Logged Out!</h1>
<h3 style="color:#4CAF50">Login for more information<h3>

    <a href="{{ url_for('login') }}"><button type="submit">Login</button></a>
</form>
</div>

</body>
</html>
```

Figure 12: logout.html page is the code for home page of our Web Application.

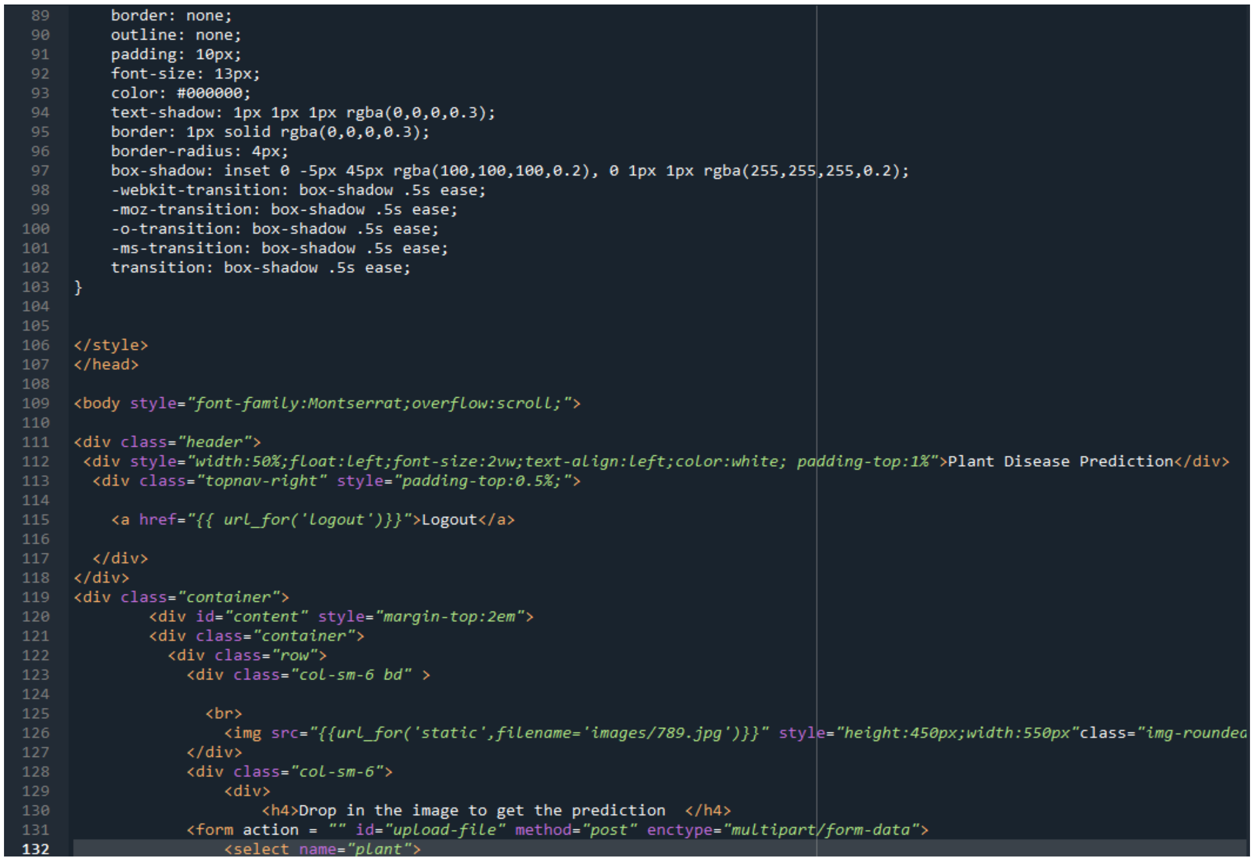
3.5. Predict.html:



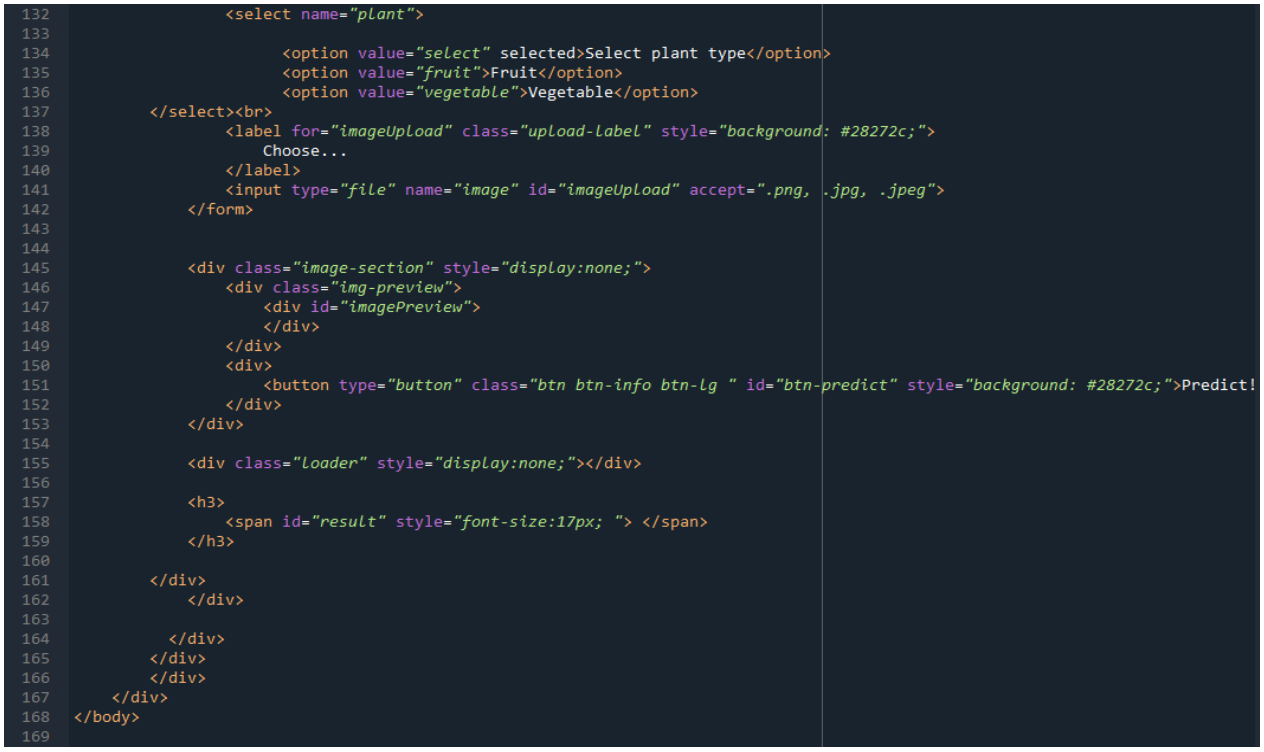
```html
<!DOCTYPE html>
<html >

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title> Plant Disease Prediction</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/2.6.0/umd/popper.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>
<link href="{{ url_for('static', filename='css/final.css') }}" rel="stylesheet">
<style>
.header {
            top:0;
            margin:0px;
            left: 0px;
            right: 0px;
            position: fixed;
            background-color: #28272c;
            color: white;
            box-shadow: 0px 8px 4px grey;
            overflow: hidden;
            padding-left:20px;
            font-family: 'Josefin Sans';
            font-size: 2vw;
            width: 100%;
            height:8%;
            text-align: center;
        }
        .topnav {
  overflow: hidden;
  background-color: #333;
}

.topnav-right a {
  float: left;
```



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

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

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

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

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

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

.container {
  margin-top:40px;
  padding: 16px;
}
select {
    width: 100%;
    margin-bottom: 10px;
    background: rgba(255,255,255,255);
```



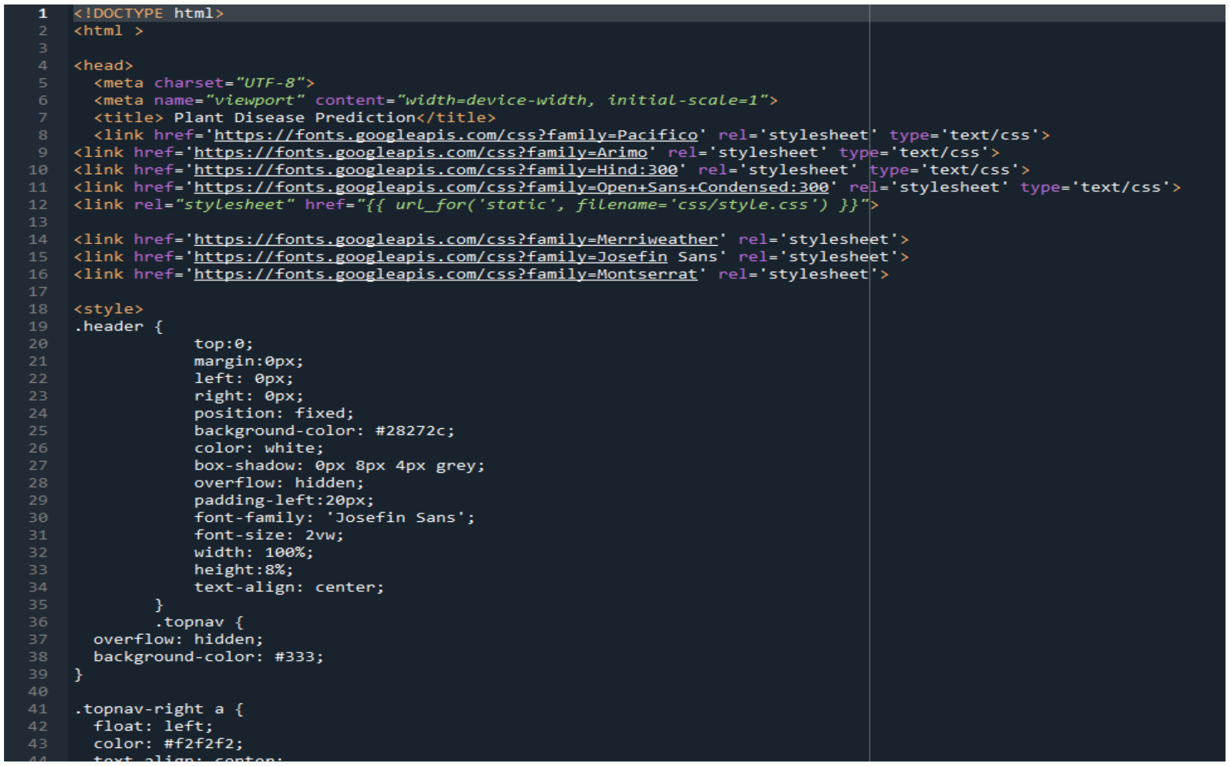
```
    border: none;
    outline: none;
    padding: 10px;
    font-size: 13px;
    color: #000000;
    text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
    border: 1px solid rgba(0,0,0,0.3);
    border-radius: 4px;
    box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
    -webkit-transition: box-shadow .5s ease;
    -moz-transition: box-shadow .5s ease;
    -o-transition: box-shadow .5s ease;
    -ms-transition: box-shadow .5s ease;
    transition: box-shadow .5s ease;
}


</style>
</head>

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

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

    <a href="{{ url_for('logout')}}">Logout</a>

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

              <br>
                <img src="{{url_for('static',filename='images/789.jpg')}}" style="height:450px;width:550px"class="img-roundea
            </div>
            <div class="col-sm-6">
                <div>
                    <h4>Drop in the image to get the prediction  </h4>
            <form action = "" id="upload-file" method="post" enctype="multipart/form-data">
                <select name="plant">
```



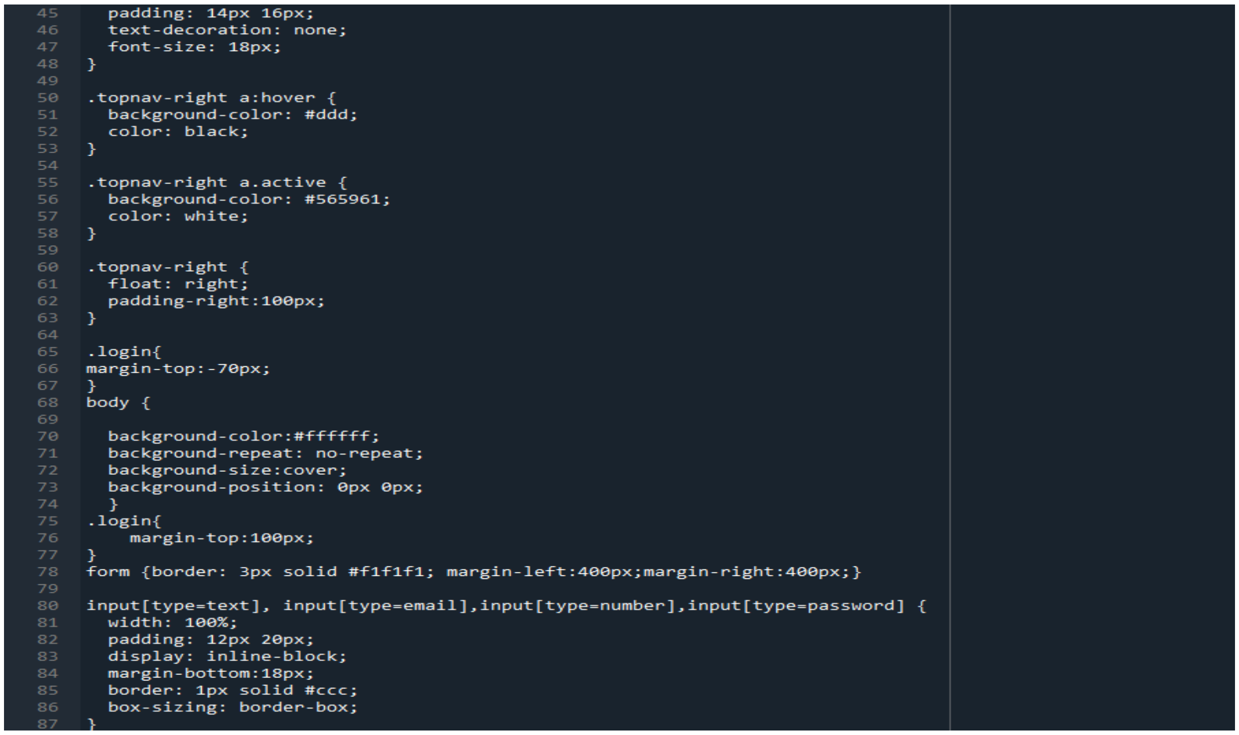
```
                <select name="plant">

                      <option value="select" selected>Select plant type</option>
                      <option value="fruit">Fruit</option>
                      <option value="vegetable">Vegetable</option>
        </select><br>
                <label for="imageUpload" class="upload-label" style="background: #28272c;">
                    Choose...
                </label>
                <input type="file" name="image" id="imageUpload" accept=".png, .jpg, .jpeg">
            </form>


            <div class="image-section" style="display:none;">
                <div class="img-preview">
                    <div id="imagePreview">
                    </div>
                </div>
                <div>
                    <button type="button" class="btn btn-info btn-lg " id="btn-predict" style="background: #28272c;">Predict!
                </div>
            </div>

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

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

        </div>
            </div>

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

Figure 13: predict.html page is the code for home page of our Web Application.

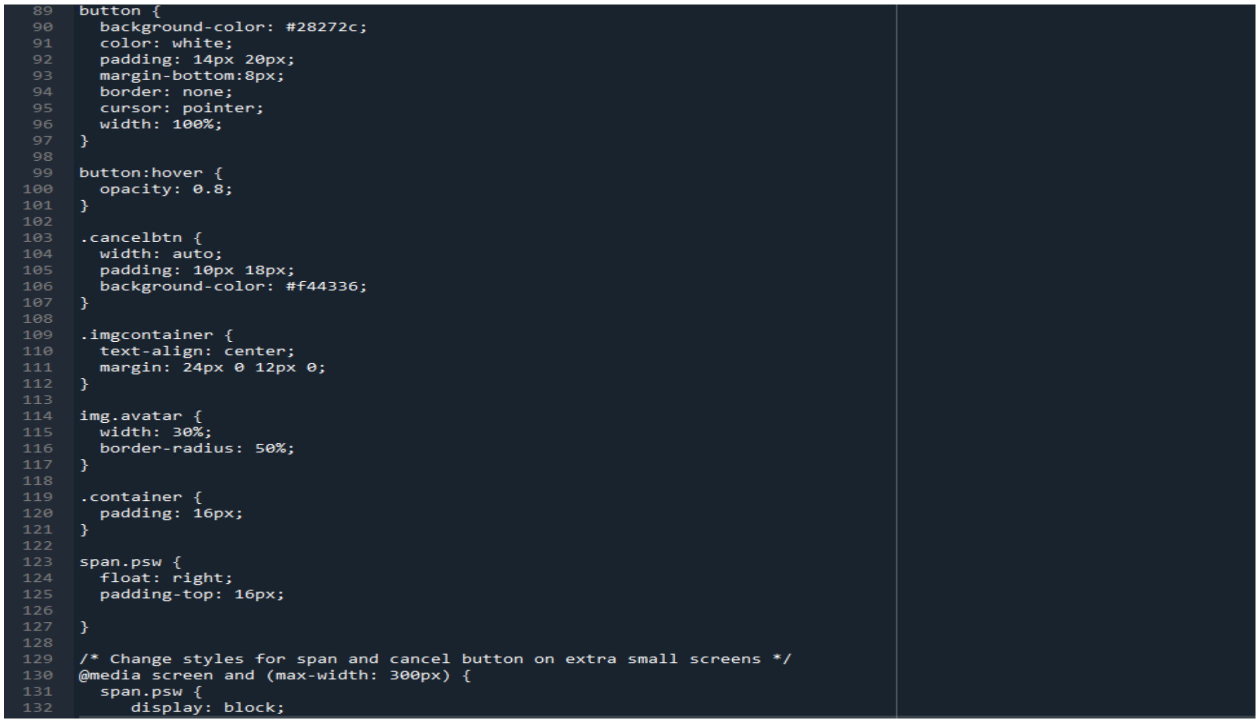
3.6. Register.html:



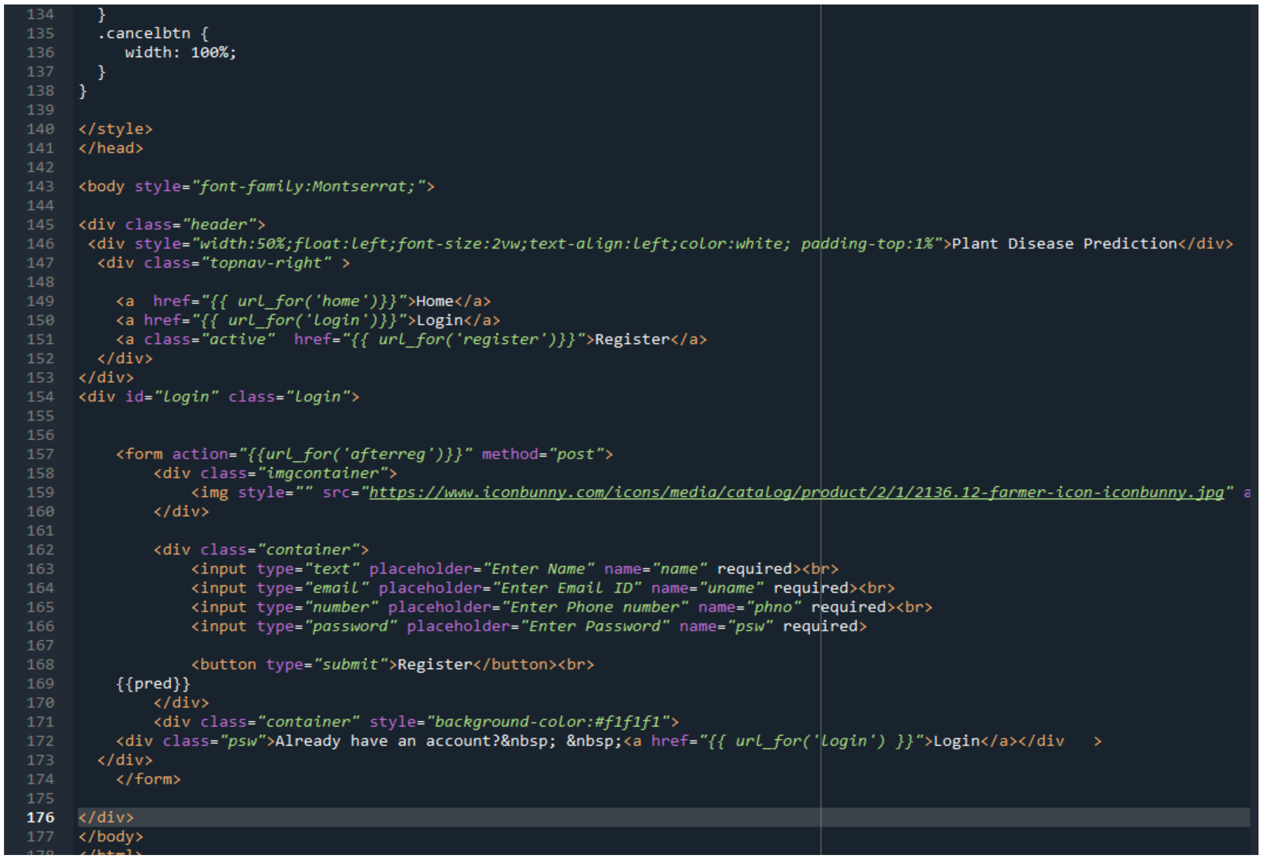
```html
<!DOCTYPE html>
<html >

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

<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>

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

.topnav-right a {
  float: left;
  color: #f2f2f2;
  text-align: center;
```



```css
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
}

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

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

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

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

  background-color:#ffffff;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  }
.login{
    margin-top:100px;
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

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



```
button {
  background-color: #28272c;
  color: white;
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
  cursor: pointer;
  width: 100%;
}

button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

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

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

.container {
  padding: 16px;
}

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

}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
  span.psw {
     display: block;
```



```
  }
  .cancelbtn {
     width: 100%;
  }
}

</style>
</head>

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

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

    <a  href="{{ url_for('home')}}">Home</a>
    <a href="{{ url_for('login')}}">Login</a>
    <a class="active"  href="{{ url_for('register')}}">Register</a>
  </div>
</div>
<div id="login" class="login">


    <form action="{{url_for('afterreg')}}" method="post">
        <div class="imgcontainer">
            <img style="" src="https://www.iconbunny.com/icons/media/catalog/product/2/1/2136.12-farmer-icon-iconbunny.jpg" a
        </div>

        <div class="container">
            <input type="text" placeholder="Enter Name" name="name" required><br>
            <input type="email" placeholder="Enter Email ID" name="uname" required><br>
            <input type="number" placeholder="Enter Phone number" name="phno" required><br>
            <input type="password" placeholder="Enter Password" name="psw" required>

            <button type="submit">Register</button><br>
    {{pred}}
        </div>
        <div class="container" style="background-color:#f1f1f1">
    <div class="psw">Already have an account?&nbsp; &nbsp;<a href="{{ url_for('login') }}">Login</a></div   >
  </div>
    </form>

</div>
</body>
```

Figure 14: register.html page is the code for home page of our Web Application.

4.OUTPUT

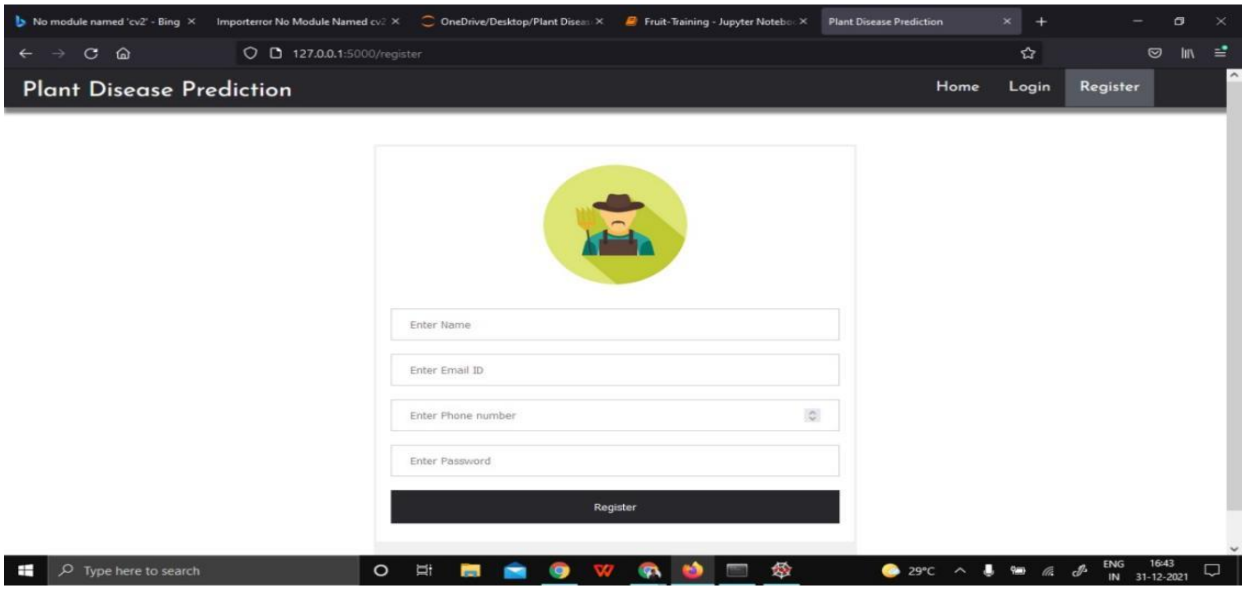


Figure 15 : Registration Page

This is the registration page the user needs to provide the necessary details for getting login into the page where you predict the plant disease.

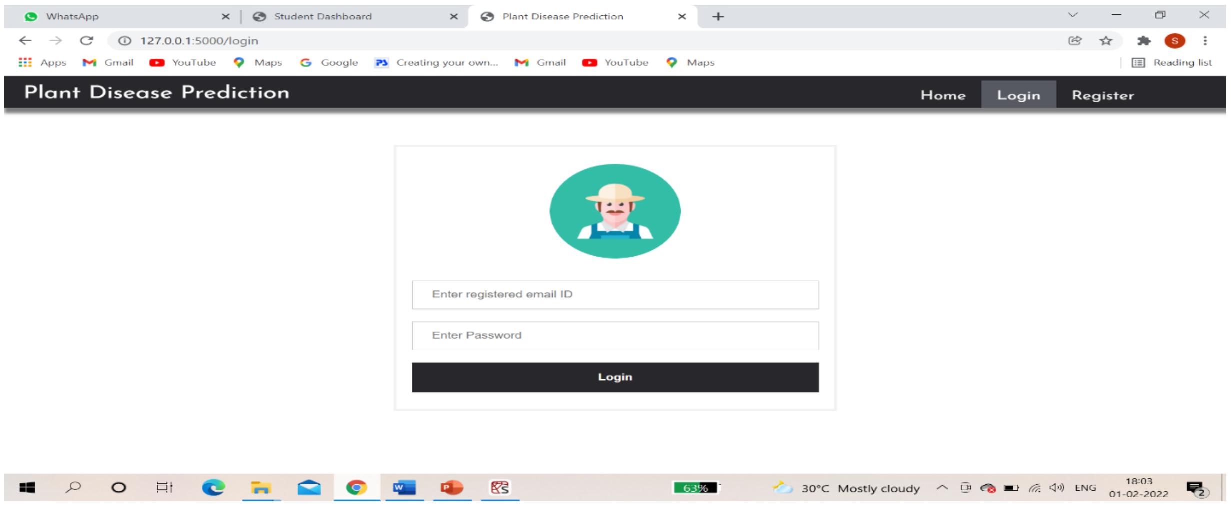
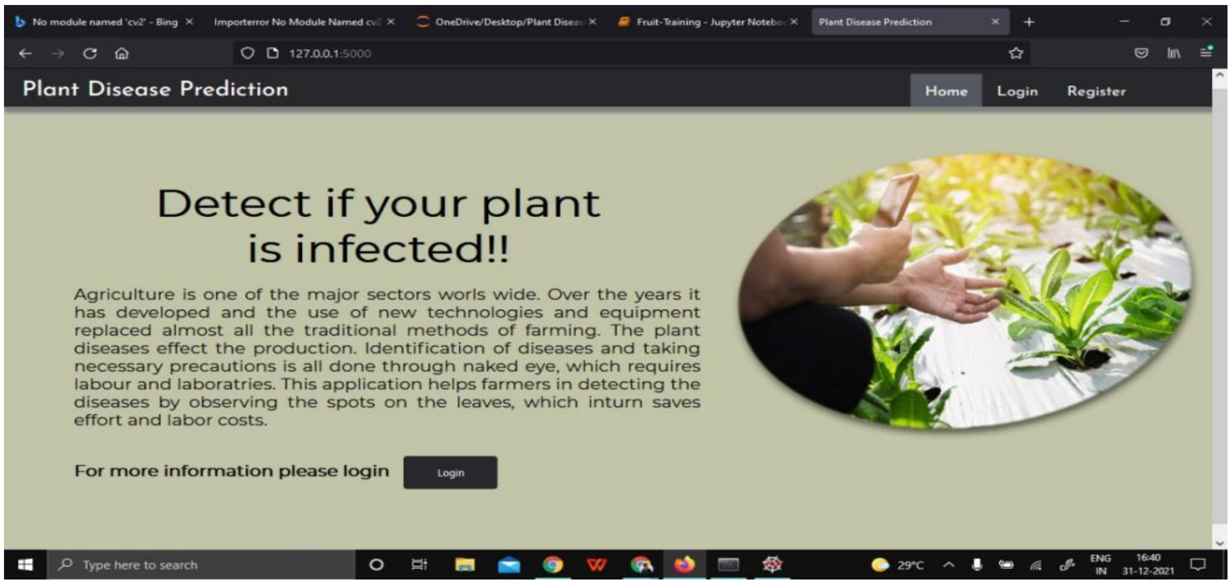


Figure 16 : Login Page

This is the login page the user needs to provide the necessary registered details for getting login into the page where you predict the plant disease.

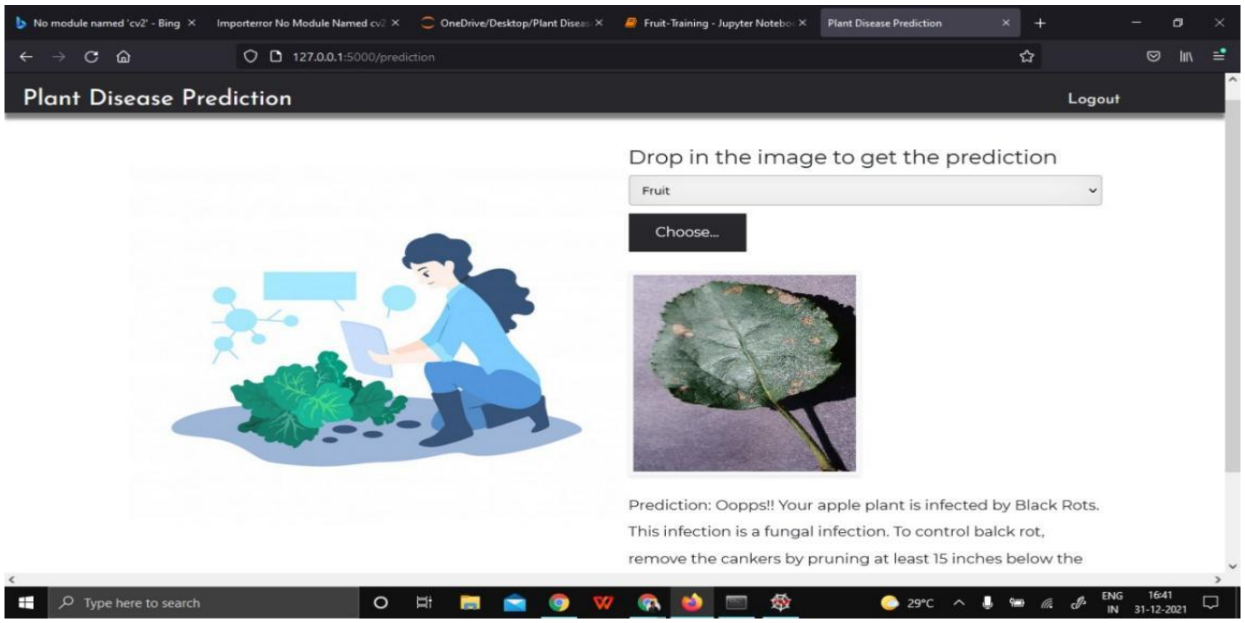


Detect if your plant is infected!!

Agriculture is one of the major sectors worls wide. Over the years it has developed and the use of new technologies and equipment replaced almost all the traditional methods of farming. The plant diseases effect the production. Identification of diseases and taking necessary precautions is all done through naked eye, which requires labour and laboratries. This application helps farmers in detecting the diseases by observing the spots on the leaves, which inturn saves effort and labor costs.

For more information please login

Figure 17 : Home page of Plant Disease Prediction ie Detect if your plant is infected!!!!


No module named 'cv2' - Bing
Importerror No Module Named cv2
OneDrive/Desktop/Plant Diseas
Fruit-Training - Jupyter Notebo
Plant Disease Prediction
127.0.0.1:5000/prediction
Plant Disease Prediction
Logout
Choose...
Type here to search
29°C
ENG IN
16:41 31-12-2021

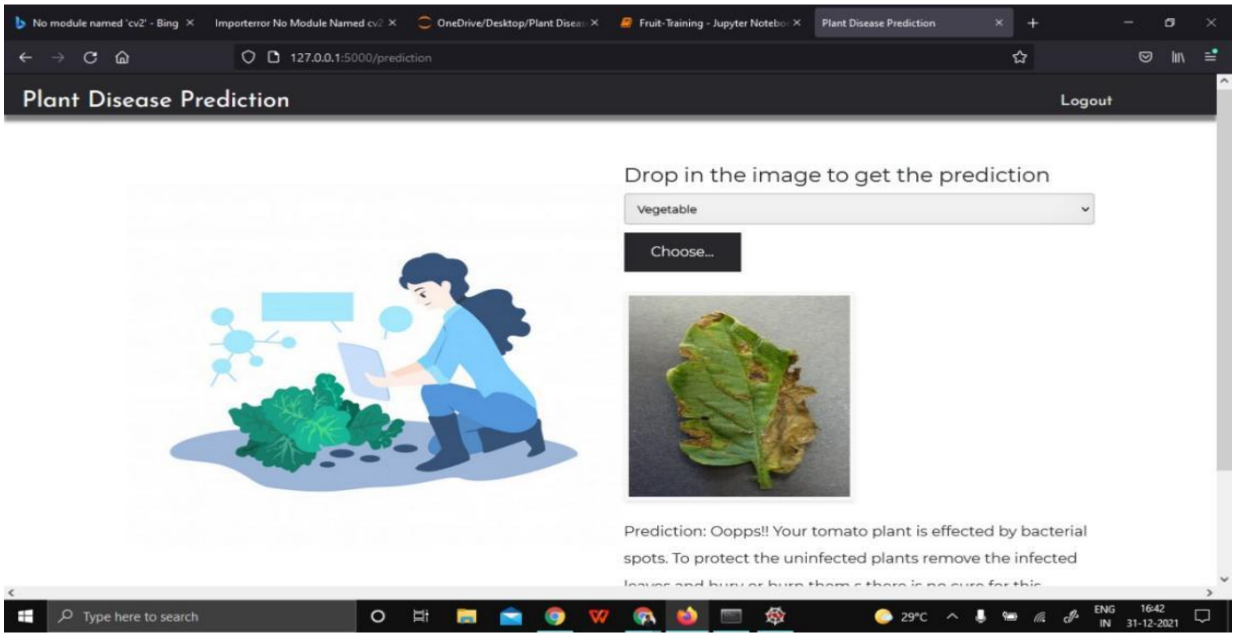
Drop in the image to get the prediction

Fruit

Prediction: Oopps!! Your apple plant is infected by Black Rots. This infection is a fungal infection. To control balck rot, remove the cankers by pruning at least 15 inches below the

Figure 18 : Prediction Page of Fruit Plant

Here we are selecting fruit plant i.e, apple plant image as input it will predict the plant condition And recommends a fertilizer to cure that plant disease.



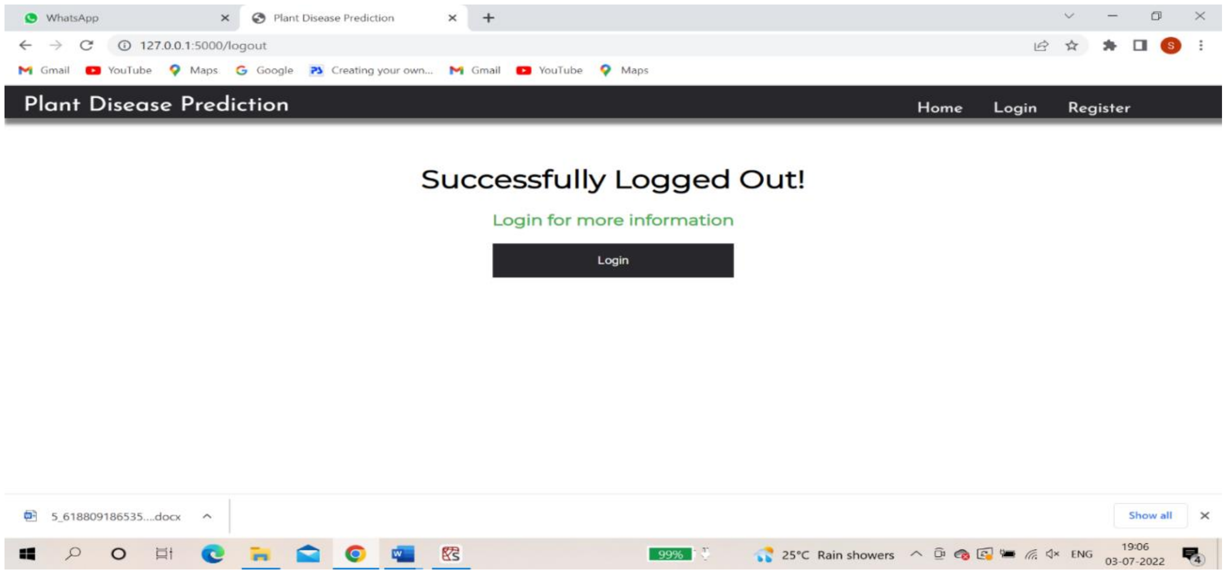
Drop in the image to get the prediction

Vegetable

Prediction: Oopps!! Your tomato plant is effected by bacterial spots. To protect the uninfected plants remove the infected leaves and bury or burn them s there is no cure for this

Figure 19 : Prediction Page of Vegetable Plant

Here we are selecting vegetable plant i.e, tomato plant image as input it will predict the plant condition and recommends a fertilizer to cure that plant disease.



Successfully Logged Out!

Login for more information

Figure 20 : Logout page

5.APPLICATION

- It is used by gardeners and farmers to predict the plant diseases.
- It is used for both fruit and vegetable plant diseases prediction and for its recovery.

6. ADVANTAGES & DISADVANTAGES

Advantages:

- It helps farmers or gardeners to know the disease from which the plant is affected from.
- It provides a recommended fertilizer to the users to recover plants from diseases.
- It is more flexible for the users to browse the images of plant disease leaves.
- It increases the crop yielding.
- The chance of occurrence of error is very less.
- It is fast efficient and reliable.
- Avoids data redundancy and inconsistency.
- It is a system user- friendly.
- Easy accessibility of data.

Disadvantages:

- It requires internet connection.
- Appropriate plant disease leaf images should be uploaded for anticipated output.

7.FUTURE SCOPE

This future 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.

8.REFERENCES

1) Aakanksha Rastogi, Ritika Arora, Shanu Sharma “advances in image processing for Leaf Disease Detection and Grading using Computer Vision Technology &Fuzzy Logic” International conference on signal processing and integrated network SPIN, pp 500-505.

2) Ms.pooja pawar, Dr.varsha turkar, Prof.pravin patil presents “algorithm for detecting crop disease early and exactly, this system is developed using image processing techniques and artificial neural network”.

3) Erika fujita, yusuke Kawasaki, Hiroyuki uga, Sathoshi kagiwada, Hitoshi lyatomi introduced a system of “Basic investigation on a robust and practical plant diagnostic system”.

4) Jayamala K. Patil1 and Raj Kumar, “Advances in image processing for detection of plant diseases”, Journal of Advanced Bio informatics Applications and Research, ISSN 0976-2604Vol 2, Issue 2, pp 135-141,June-2011.

5) Pokrajac, D. Lazarevic, A. ; Vucetic, S. ; Fiez, T. and Obradovic, Z.,“Image Processing in Preciion Agriculture”, 4th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, pp. 616-619, 2009.

9.HELP FILE

PROJECT EXECUTION:

STEP-1: Go to **Start,** search and launch **ANACONDA NAVIGATOR.**

STEP-2: After launching of **ANACONDA NAVIGATOR,** launch **JUPYTER NOTEBOOK.**

STEP-3: Open **“Major project code” IPYNB file.**

STEP-4: Then run all the cells.

STEP-5: All the **image pre-processing**, **training and testing**, **model building** of the model can be showcased.

STEP-6: And a pickle file will be generated.

STEP-7: Create a Folder named **FLASK** on the **DESKTOP.** Extract the pickle file into this Flask Folder.

STEP-8: Extract all the html files (home.html, login.html, logout.html, predict.html, register.html) into the **FLASK Folder.**

STEP-9: Then go back to **ANACONDA NAVIGATOR** and the launch the **SPYDER.**

STEP-10: After launching Spyder, give the path of **FLASK FOLDER** which you havecreated on the DESKTOP.

STEP-11: Open all the app.py and html files present in the Flask Folder.

STEP-12: After running of the app.py, open **ANACONDA PROMPT** and follow the belowsteps:

cd File Path→click enter

python app.py→click enter (We could see running of files).

STEP-13: Then open **BROWSER,** at the URL area type **–local host:5000”.**

STEP14: Home page of the project will be displayed.

STEP-15: Click on **–Go to login”.** Directly it will be navigated to login page and provide your login credentials.

STEP-16:A predict page will be displayed where the user needs to select the type of the plant(ie.fruit or vegetable plant) and select the image of the plant as input and then click on **–Predict”.** Output will be generated whether the plant is infected from disease or not.