Fertilizers Recommendation System For Disease Prediction

PROJECT REPORT

Team ID : PNT2022TMID18626

Team Size : 4

1. **Team Leader :** Abbas Kashim S
2. **Team member :** Bharath Kumar R
3. **Team member :** Devi Shruthi S
4. **Team member :** Jeneni S

Project Report Format

1. INTRODUCTION

1.1Project Overview:

i. Today, agriculture is the most significant industry. An extensive range of bacterial and fungal diseases harm most plants. Plant diseases severely limited productivity and posed a serious threat to food security. To achieve maximum quantity and optimum quality, early and accurate identification of plant diseases is crucial. The variety of pathogen strains, adjustments to production practices, and insufficient plant protection systems have all contributed to an increase in the number of plant diseases in recent years, as well as the severity of the damage they inflict.

ii. An automated technique is now available to recognize many plant diseases by examining the symptoms seen on the plant's leaves. To identify diseases and recommend preventative measures, deep learning techniques are used.

1.1 Purpose:

Many plant diseases can now be identified automatically by looking at the symptoms on the plant's leaves. Deep learning techniques are used to pinpoint ailments and suggest prevention measures.

2.2 LITERATURE SURVEY

2.1 Existing Problem:

1. Crop production relies heavily on sufficient mineral nutrition. It can, however, have a significant impact on the emergence of diseases. The effects of nutrients on plant growth, plant defence systems, and direct impacts on the pathogen are some of the complicated methods by which fertiliser application can promote or decrease the development of diseases brought on by various pathogens. Comprehensive coverage of the impacts of mineral nutrition on plant disease and the mechanisms underlying such effects can be found elsewhere. In India, about 40% of the land is maintained and used for farming using dependable irrigation techniques, while the remaining 60% is watered by the monsoon season. The use of irrigation raises agricultural production, improves food security, and lessens dependency on the monsoon.

2. Most studies articles use humidity, moisture, and temperature sensors close to the plant's root, with an outside tool managing all the information furnished via way of means of the sensors and transmitting it immediately to an outside show or an Android application. The application became created to degree the approximate values of temperature, humidity and moisture sensors that have been programmed right into a microcontroller to manipulate the quantity of water.

3. Detection and recognition of plant diseases using machine learning are very efficient in providing symptoms of identifying diseases at its earliest. Plant pathologists can analyse the digital images using digital image processing for diagnosis of plant diseases. Application of computer vision and image processing strategies simply assist farmers in all of the regions of agriculture. Generally, the plant diseases are caused by the abnormal physiological functionalities of plants. Therefore, the characteristic

symptoms are generated based on the differentiation between normal physiological functionalities and abnormal physiological functionalities of the plants.

2.2 References:

- http://www.ijetajournal.org/volume-8/issue-2/IJETA-V8I2P1.pdf
- https://ieeexplore.ieee.org/document/8878781
- https://ieeexplore.ieee.org/document/8921213
- https://ieeexplore.ieee.org/abstract/document/9334934
- https://www.sciencedirect.com/science/article/pii/S0168169921004245

2.3 Problem Statement Definition:

The most significant sector in modern life is agriculture. A wide range of bacterial and fungal diseases affect most plants. Plant diseases hampered production significantly and posed a significant threat to food security. As a result, in order to guarantee maximum quantity and quality, plant diseases must be identified promptly and accurately. The variety of pathogen varieties, shifts in cultivation practices, and inadequate plant protection methods have all contributed to an increase in the number and severity of plant diseases in recent years. By examining the symptoms that are displayed on the leaves of the plant, an automated system is introduced to identify various plant diseases. The diseases are identified and the measures that can be taken to prevent them are suggested using deep learning methods**.**

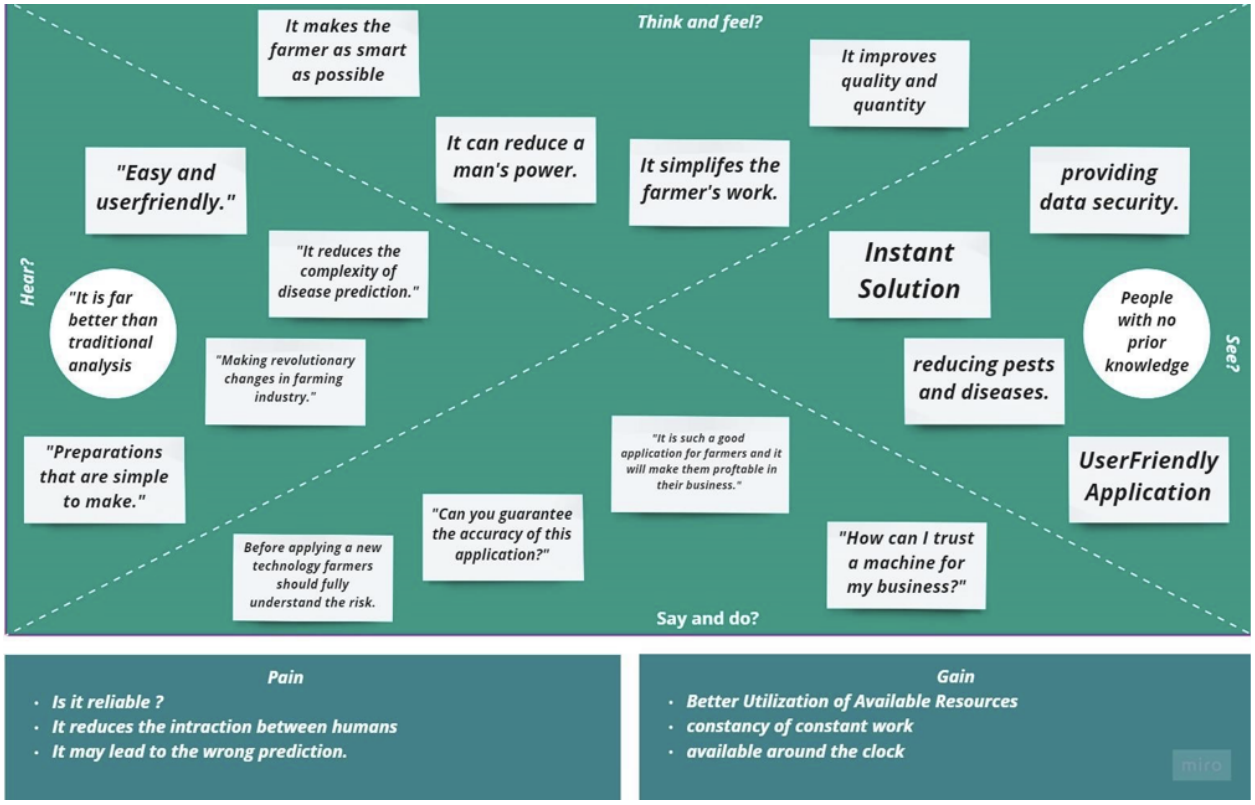
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

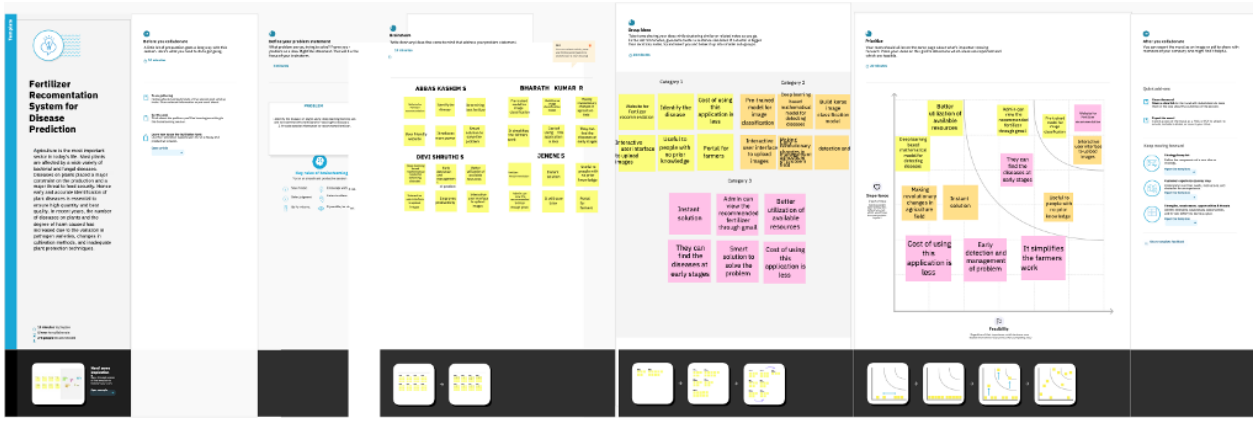
Empathy Map for Fertilizer Recommendation System:

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community .

Empathy Map:



3.2.Ideation & Brainstorming:

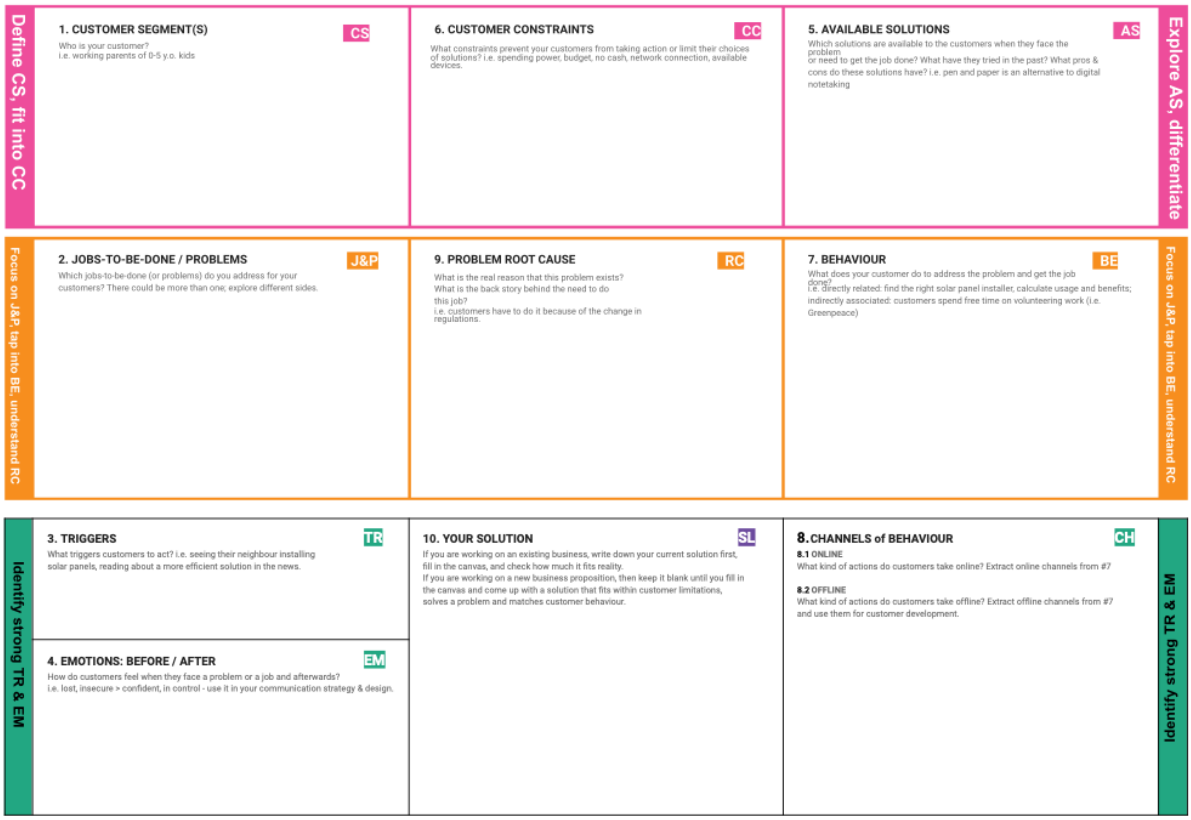


3.3. Proposed Solution :

A deep learning-based neural network is used to train and test the collected datasets in this project.CNN, a neural network based on deep learning, achieves classification accuracy of more than 90%.The accuracy rate can be increased to 95% to 98% by increasing the number of dense layers and modifying hyper parameters like batch size and number of epoch.

3.4.Problem Solution fit:

Project Title: **Project Design Phase-I - Solution Fit Template** **Team ID:** PNT2022TMIDxxxxxx



4. REQUIREMENT ANALYSIS

4.1.Functional requirement :

FR No	Functional Requirements	Sub Tasks
FR 1	Uploading the image of the plant leaf which is affected by the disease	Updates the leaf image of the plant affected by the disease instantly
FR 2	Proper disease prediction	The plant disease is predicted with the image
FR 3	Timely fertilizer recommendation	Timely recommendation of the fertilizer for the plant disease

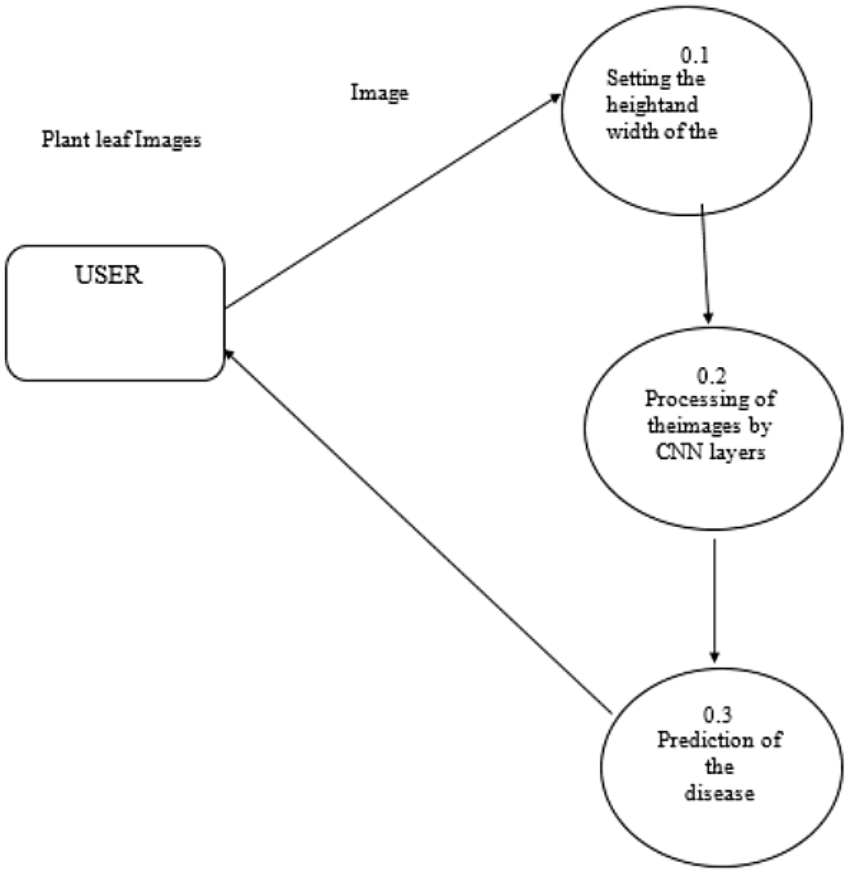
4.2. Non-Functional requirements :

NFR-4	Availability	Should be available from anywhere Maintenance should be done such that the down time is not more than 2hours
NFR-5	Scalability	Should be able to scale millions of users and their data

5. PROJECT DESIGN

5.1. Data Flow Diagrams

It can be used to plan or record the specific/necessary detail about the system's functioning.



5.2. Solution & Technical Architecture:

5.3. User Stories

Open software that can be used to predict the disease of a fruit or vegetable plant by uploading a picture of the leaf. This will give you a prediction of the disease as well as the steps that should be taken to treat it.

6. PROJECT PLANNING & SCHEDULING

6.1. Sprint Planning & Estimation

Sprint	Functional Requirement	User story Number	User Story/ Task	Story Points	Prior ty	Team Members
Sprint 1	Image processing		Process the images for model training	2	High	1. Abbas kashim S 2. Bharath kumar R 3. Devi Shruthi S 4. Jeneni S
	Data Collection		Gather the data set from kaggle	2	High	1. Abbas kashim S 2. Bharath kumar R 3. Devi Shruthi S 4. Jeneni S

Sprint	Functional Requirement	User story Number	User Story/ Task	Story Points	Prior ty	Team Members
Sprint-2	model building and training(fruit)		Create a model which can classify diseased fruit plants from given images	8	High	5. Abbas kashim S 6. Bharath kumar R 7. Devi Shruthi S 8. Jeneni S
	Model building and training(Vege table)		Create a model which can classify diseased vegetable plants from given images.	8	High	1. Abbas kashim S 2. Bharath kumar R 3. Devi Shruthi S 4. Jeneni S

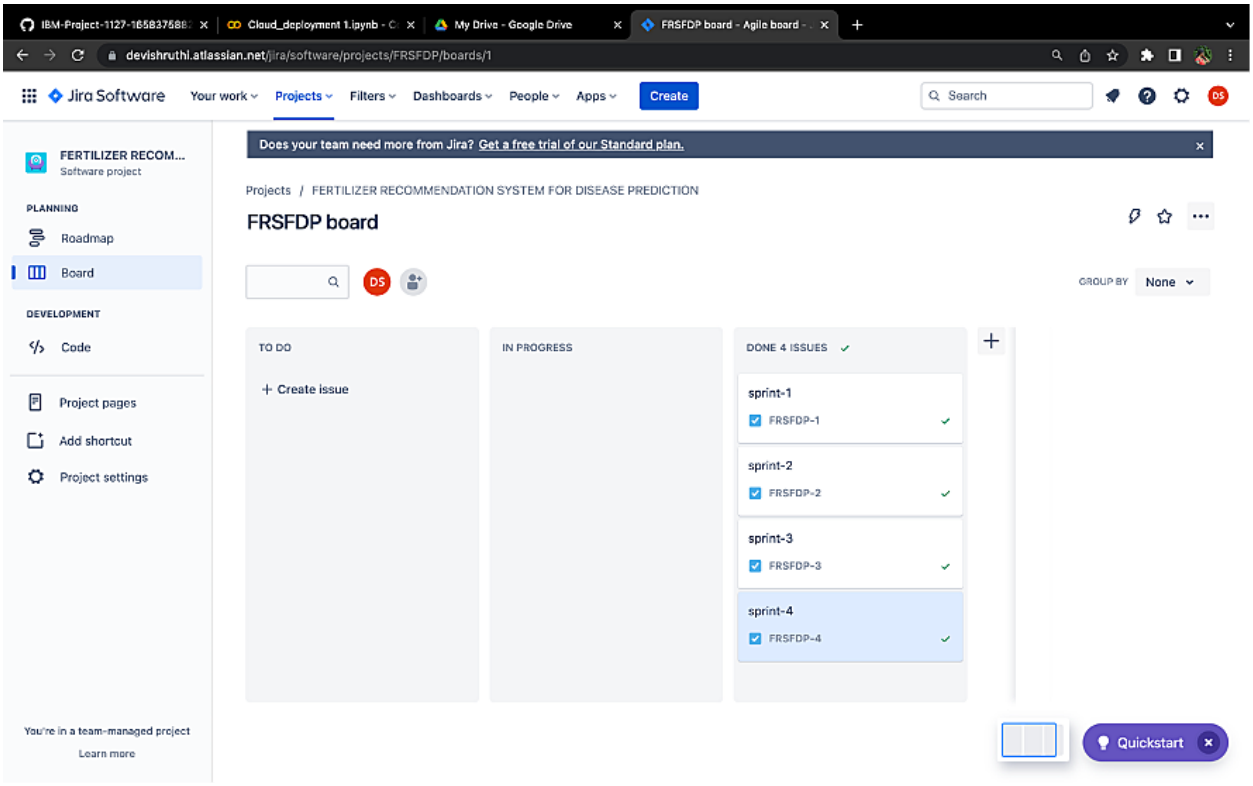
Sprint	Functional Requirement	User story Number	User Story/ Task	Story Points	Priority	Team Members
sprint-3	Testing the model (Fruit)		Test the Fruit model built and train it on IBM Cloud	8	High	9. Abbas kashim S 10. Bharath kumar R 11. Devi Shruthi S 12. Jeneni S
	Testing the model (Vegetable)		Test the Vegetable model built and Train it on IBM Cloud	8	High	1. Abbas kashim S 2. Bharath kumar R 3. Devi Shruthi S 4. Jeneni S

Sprint	Functional Requirement	User story Number	User Story/ Task	Story Points	Priorty	Team Members
Sprint -4	Application Building		Develop the Web pages. Home.html and predict.html	8	HIgh	1. Abbas kashim S 2. Bharath kumar R 3. Devi Shruthi S 4. Jeneni S
	Flask Application		Integrate the HTML pages with Flask and build web application	8	High	5. Abbas kashim S 6. Bharath kumar R 7. Devi Shruthi S 8. Jeneni S

6.2. Sprint Delivery Schedule:

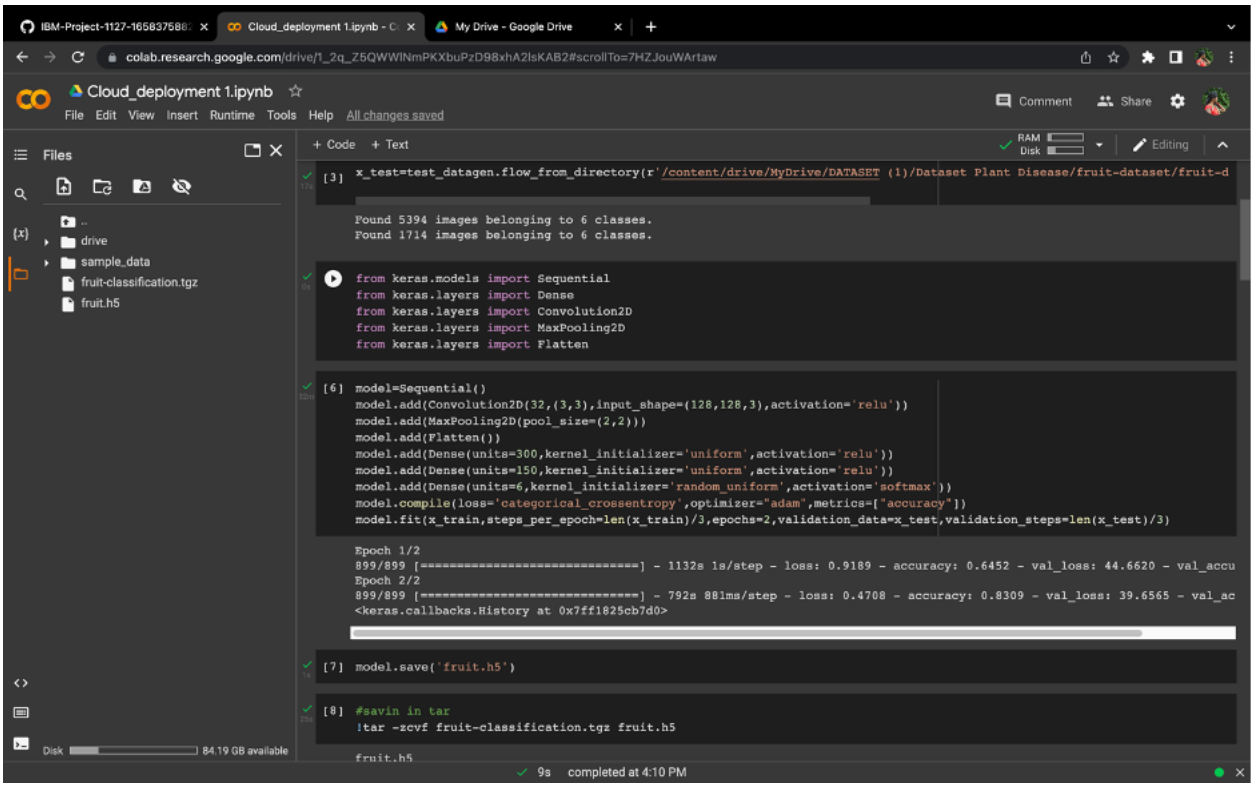
Sprint	Functional Requirement	Priority	Estimated Delivery Date
Sprint-1	Image preprocessing	High	29th October 2022
Spint-2	Model Training	High	5th November 2022
Sprint-3	Model Testing & Cloud Deployment	High	12th November 2022
Sprint-4	Application building	High	19th November 2022

6.3. Reports from JIRA:



7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1:



The CNN algorithm is used to predict diseases for the fruit and vegetable disease prediction. To improve accuracy, the model is tested on 1686 images across six classes and trained on 5384 images. To improve accuracy, we added a single convolution layer and max pooling of size 2, but additional convolution layers and larger max pooling can be added. The model is then trained over a period of two epochs, which can be extended for higher accuracy rates.

7.2 Feature 2:

```
app = Flask(__name__)
model=load_model("fruit.h5")
model1=load_model("veg.h5")

@app.route('/')
def home():
    return render_template('home.html')
@app.route('/prediction')
def prediction():
    return render_template('predict.html')
@app.route('/predict',methods=['POST'])
def predict():
    if( request.method=='POST'):
        f = request.files['image']
        basepath=os.path.dirname(__file__)
        file_path=os.path.join(basepath,'uploads',secure_filename(f.filename))
        f.save(file_path)
        img=image.load_img(file_path,target_size=(128,128))
        x=image.img_to_array(img)
        x=np.expand_dims(x,axis=0)
        plant=request.form['plant']
        print(plant)
        if(plant=='vegetable'):
            preds =model1.predict(x)
            classes=np.argmax(preds,axis=1)
            print(classes)
            df=pd.read_excel('precautions - veg.xlsx')
            print(df.iloc[classes[0]]['caution'])
        else:
            preds =model.predict(x)
            classes=np.argmax(preds,axis=1)
            df=pd.read_excel('precautions - fruits.xlsx')
            print(df.iloc[classes[0]]['caution'])
        return df.iloc[classes[0]]['caution']
if __name__=="__main__":
    app.run(debug=False)
```

We have developed a flask application, for smooth user experience while getting the predictions. The home page displays the information regarding the importance of Agriculture and disease prediction. The prediction lets the user to choose between the vegetable or fruit and then upload the image that need to be scanned for disease prediction.

8. TESTING

8.1. Test Cases:

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

failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	6	0	0	6
Client Application	20	0	2	18
Security	2	0	0	2
Outsource Shipping	0	0	0	0
Exception Reporting	10	0	0	10
Final Report Output	1	0	0	1
Version Control	1	0	0	1

8.2 User Acceptance Testing

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

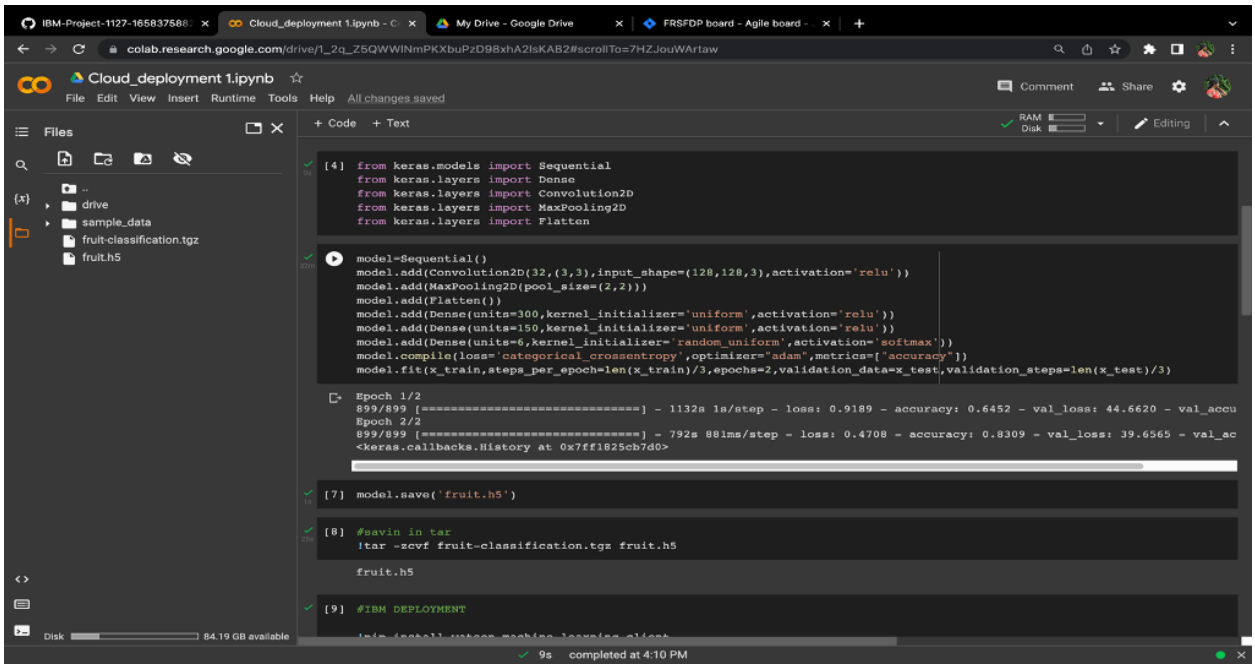
failed,and untested

Resolution	Severity 1	Severity 2	Severity 3	Severity 4
By Design	9	4	2	3
Duplicate	1	0	3	0
External	0	0	2	3
Fixed	15	2	4	10
Not Reproduced	0	0	0	1
Skipped	0	0	1	1
Won't Fix	2	0	0	0

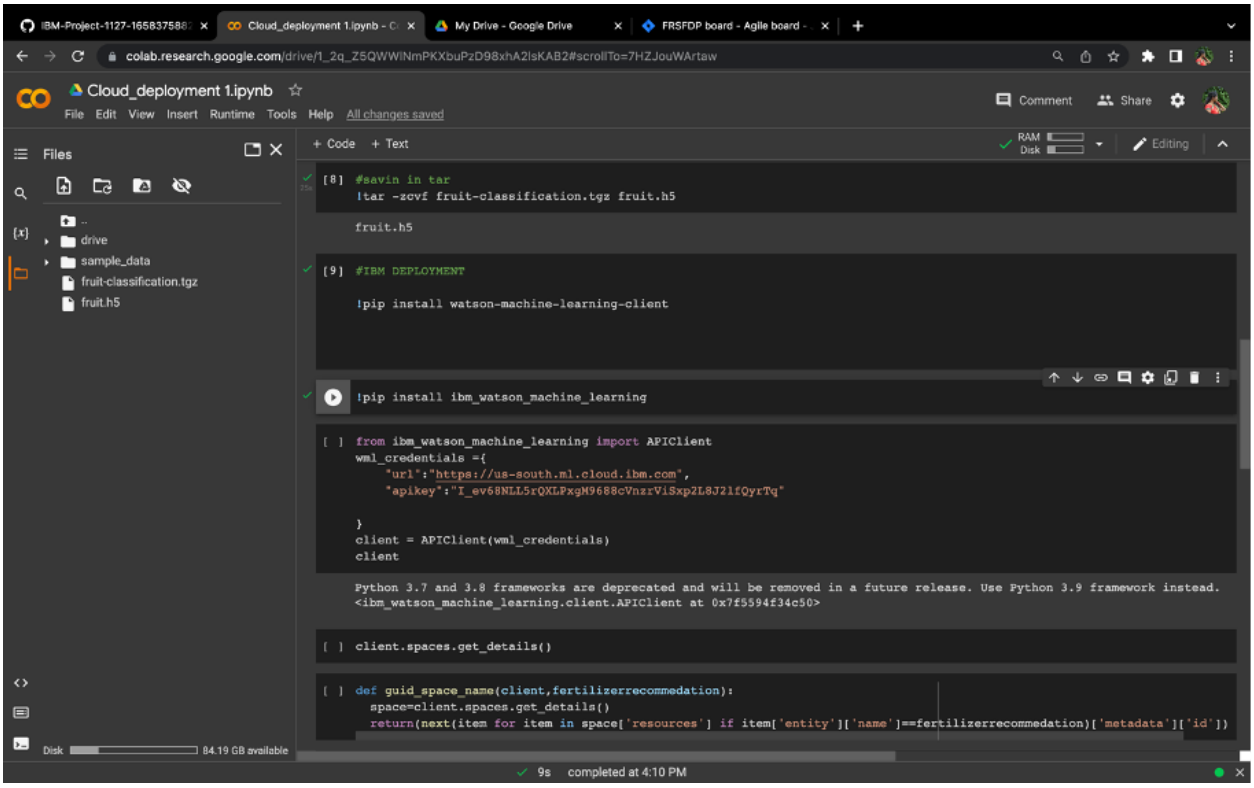
9. RESULTS

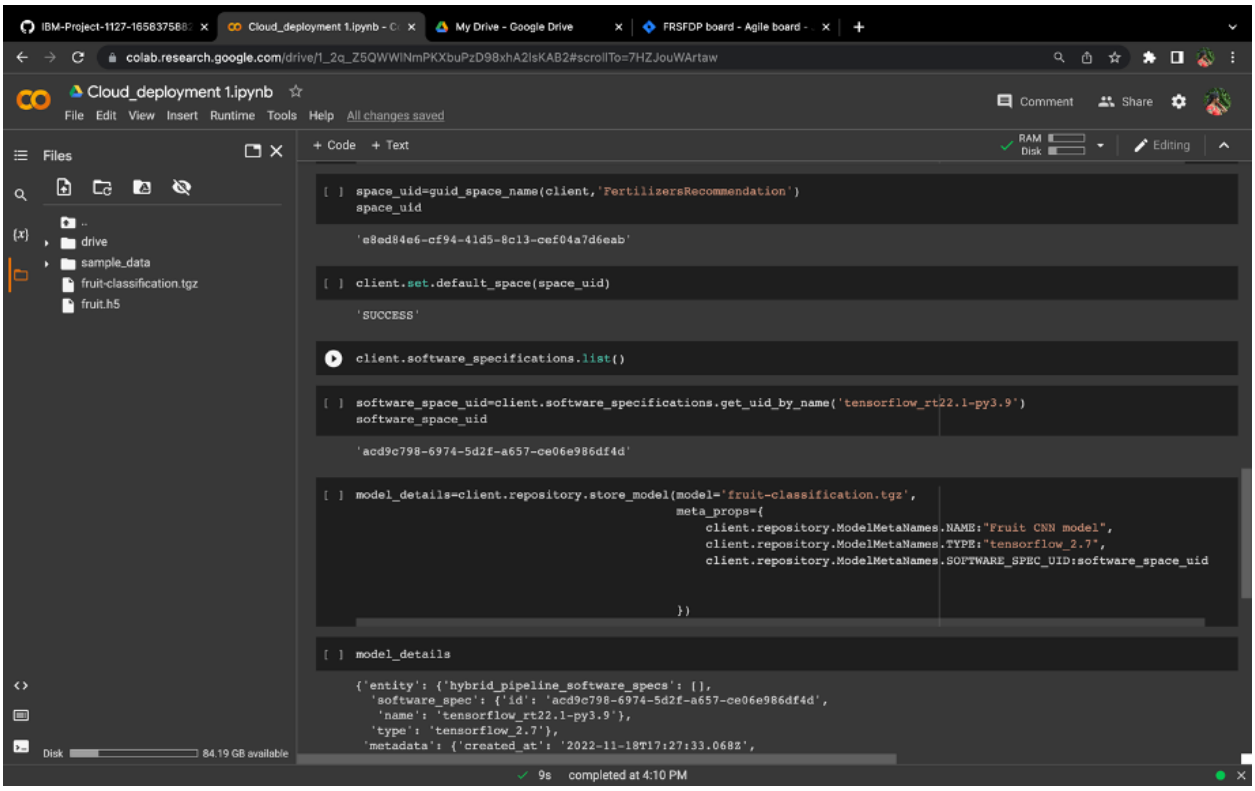
9.1 Performance Metrics:

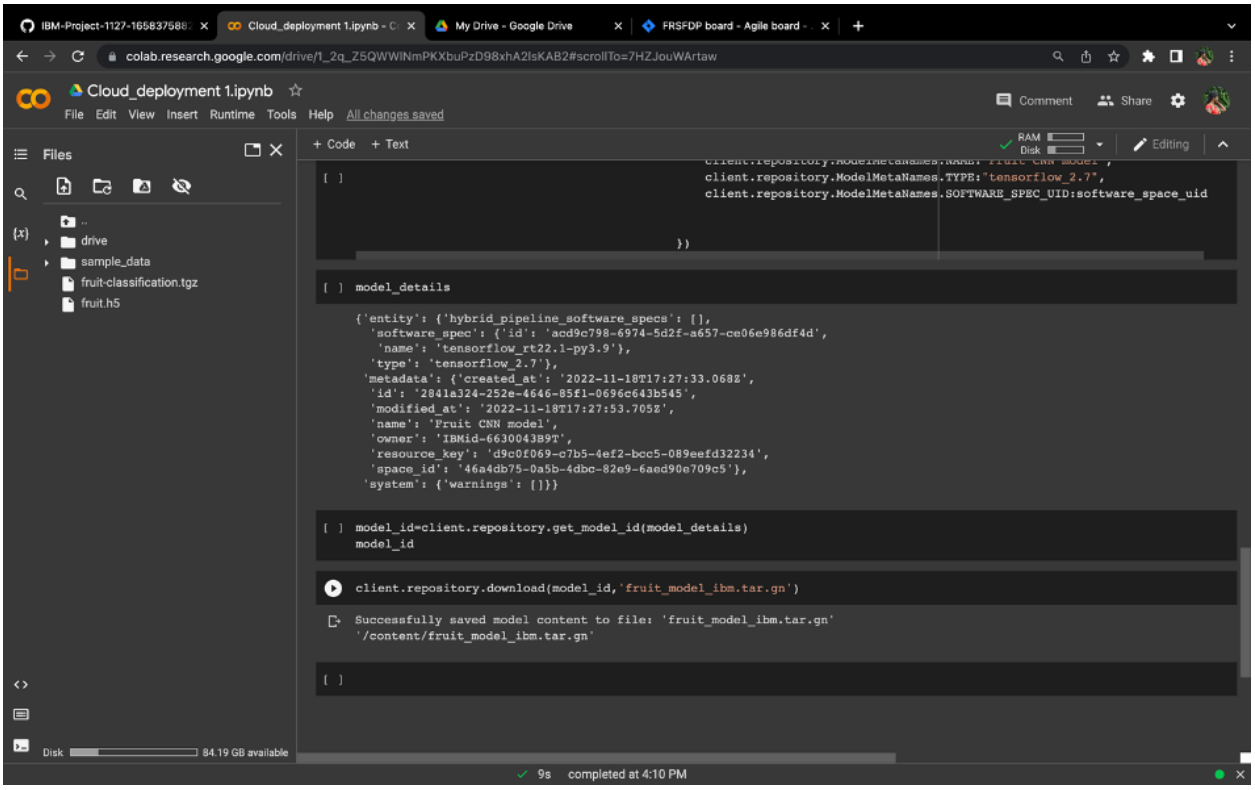
Final findings(output) of the project given below in the form of screenshot:
Training and Testing of Fruit dataset



- The above images shows the code and output for model building of fruit and vegetable.

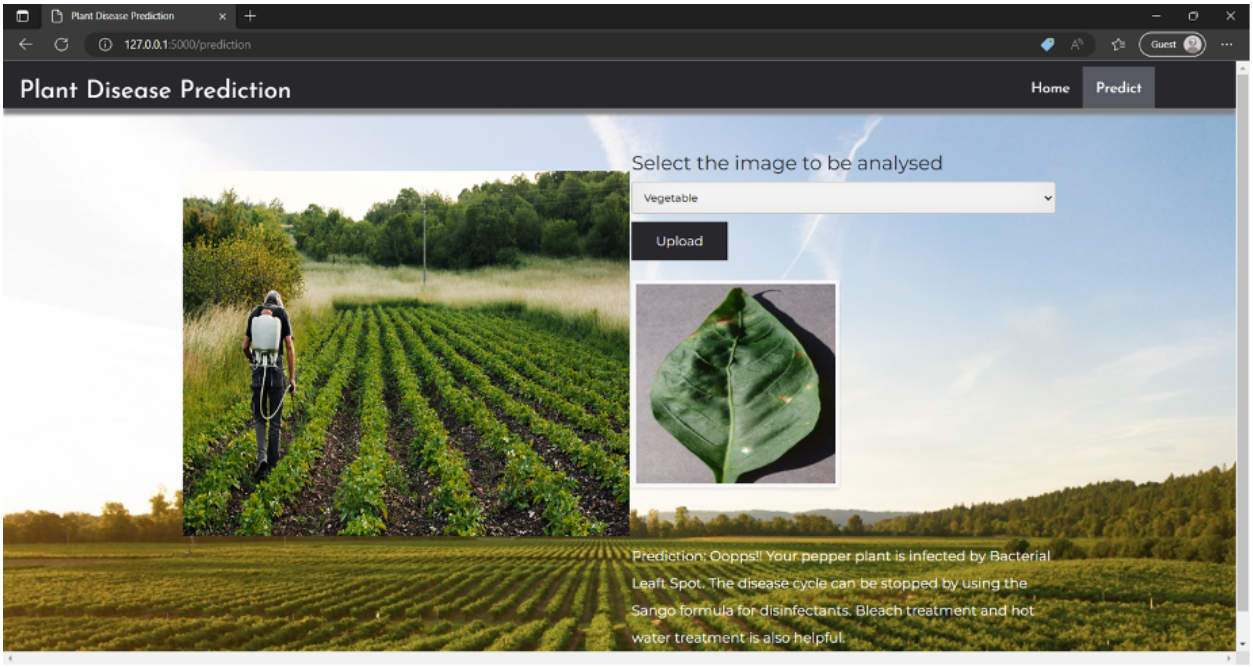






- Deploying the fruit and vegetable models in IBM Cloud.

- The home page rendered using the Flask framework.



- Prediction based on the image uploaded by the user in the flask application and the results are shown.

10. ADVANTAGES & DISADVANTAGES

List of Advantages:

1. Classification accuracy is extremely high with this proposed model.
2. Additionally, numerous datasets can be trained and tested.
3. Within the proposal itself, images can be resized to huge sizes.

List of Disadvantages :

1. For preparing and testing, the proposed model calls for high as can be computational investment.
2. This project's neural network architecture is extremely complex.

11. CONCLUSION:

The proposed model focuses on vegetable and fruit dataset image classification. The following observations are made during the model's testing and training:

1. The classification accuracy improved as the number of epochs increased.
2. For various batch sizes, various classification accuracies are obtained.
3. The accuracy is improved by the increased number of convolution layers.
4. Classification precision was also improved by variable dense layers.
5. Various accuracies can be achieved by altering the size of the kernel used in the convolution layer's output. The accuracy varies when the train and test datasets are different sizes.

12. FUTURE SCOPE

So far, we've only built web applications that take input as images and seem to predict the results we might develop soon. Applying computer vision and AI techniques used to predict infection as soon as a camera is brought close to a plant or leaf could help the project even more.

APPENDIX

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-27129-1660047253

Project Demo:

https://drive.google.com/file/d/1p6eGid7IkjM1kKOpQmEMmUfg7T2jvbHH/view?usp=share_link

Python code:

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

```
from werkzeug.utils import secure_filename
from tensorflow.python.keras.backend import set_session

app = Flask(__name__)
model=load_model("fruit.h5")
model1=load_model("veg.h5")

@app.route('/')
def home():
    return render_template('home.html')
@app.route('/prediction')
def prediction():
    return render_template('predict.html')
@app.route('/predict',methods=['POST'])
def predict():
    if( request.method=='POST'):
        f = request.files['image']
        basepath=os.path.dirname(__file__)

file_path=os.path.join(basepath,'uploads',secure_filename(f.filena
me))
        f.save(file_path)
```

```
            img=image.load_img(file_path,target_size=(128,128))
            x=image.img_to_array(img)
            x=np.expand_dims(x,axis=0)
            plant=request.form['plant']
            print(plant)
            if(plant=='vegetable'):
                preds =model1.predict(x)
                classes=np.argmax(preds,axis=1)
                print(classes)
                df=pd.read_excel('precautions - veg.xlsx')
                print(df.iloc[classes[0]]['caution'])
            else:
                preds =model.predict(x)
                classes=np.argmax(preds,axis=1)
                df=pd.read_excel('precautions - fruits.xlsx')
                print(df.iloc[classes[0]]['caution'])
            return df.iloc[classes[0]]['caution']
        if __name__=="__main__":
            app.run(debug=False)
```

[HTML Code]:

Home Page:

```
<!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-image: url('static/images/130.jpeg');
  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: 50px;
}
.button:hover {
  box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0
rgba(0,0,0,0.19);
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-
right:400px;}

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

button {
  background-color: #28272c;
  color: white;
```

```
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
  cursor: pointer;
  width: 15%;
  border-radius:4px;
}

button:hover {
  opacity: 0.8;
}

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

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

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

```
.container {
  padding: 16px;
}

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

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

.home{
  margin:80px;

  width: 84%;
  height: 500px;
```

```
  padding-top:10px;
  padding-left: 30px;

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

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

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

```
}



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

.active {
  background-color: #717171;
```

```
}

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

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

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

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

```
<body style="font-family:'Times New Roman', Times, serif; background-image: url(home.jpeg);">

<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('prediction')}}">Predict</a>
  </div>
</div>

<div style="background-color:#ffffff;">
<div style="width:60%;float:left;">
<div style="font-size:50px;font-family:Montserrat;padding-left:20px;text-align:center;padding-top:10%;font-weight: bold;">
<b>Detect if your plant is infected!!</div><br>
<div style="font-size:20px;font-family:Montserrat;padding-left:70px;padding-right:30px;text-align:justify;">The key development in the rise of human civilisation was agriculture, which allowed humans to live in cities due to an abundance of food provided by farming acclimatised species.
  In at least 11 different parts of the world, plants had developed separately. Even though 2 billion people still relied on agriculture to maintain themselves,
```

industrial agriculture based on extensive monocropping in the twentieth century began to have an impact on agricultural productivity. The productivity is impacted by plant diseases.

All disease detection and preventative measures are done by the naked eye, which involves labour and laboratries. By observing the blotches on the leaves, this programme assists farmers in identifying diseases, saving them time and money on labour costs.

<img src="{{url_for('static',filename='images/127.jpeg')}}" style="height:450px;width:550px"class="img-rounded" alt="Gesture">

<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>
```

Prediction Page:

```
<!DOCTYPE html>
<html >

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

```
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title> Plant Disease Prediction</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
```

```
<link href='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>
<link href="{{ url_for('static', filename='css/final.css') }}"
rel="stylesheet">
<style>
.header {
        top:0;
        margin:0px;
        left: 0px;
        right: 0px;
        position: fixed;
        background-color: #28272c;
        color: white;
        box-shadow: 0px 8px 4px grey;
        overflow: hidden;
        padding-left:20px;
        font-family: 'Josefin Sans';
        font-size: 2vw;
        width: 100%;
        height:8%;
        text-align: center;
    }
    .topnav {
  overflow: hidden;
  background-color: #333;
}
```

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

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

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

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

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

```
body {

  background-color:#ffffff;
  background-image: url('home.jpeg');
  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('home')}}">Home</a>
           <a class = "active" href="{{
url_for('prediction')}}">Predict</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/128.jpeg')}}"
style="height:450px;width:550px"class="img-rounded"
alt="Gesture">
            </div>
            <div class="col-sm-6">
               <div>
                  <h4>Select the image to be analysed </h4>
            <form action = "" id="upload-file" method="post"
enctype="multipart/form-data">
               <select name="plant">

                  <option value="select" selected>Select plant
type</option>
                  <option value="fruit">Fruit</option>
                  <option value="vegetable">Vegetable</option>
        </select><br>
              <label for="imageUpload" class="upload-label">
                 Upload
```

```
            </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!</button>
            </div>
        </div>

        <div class="loader" style="display:none;"></div>
<br>
<br>
        <h3>
            <span id="result" style="font-size:17px; color: #ffffff; ">
</span>
        </h3>

      </div>
        </div>
```

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

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