Fertilizers Recommendation System For Disease Prediction

(TEAM ID : PNT2022TMID36381)

PROJECT REPORT

Submitted by

SANJAY R
JAYA KUMAR S
ROHITH N
SENTHILKUMARAN V
JAMESRAJ A

in partial fulfilment for the award of degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

JAYA ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

ACKNOWLEDGEMENT

We acknowledge our sincere thanksto Head of the Department **Mr. KUMARAN M** for giving us valuable suggestion and help towardsus throughout this Project.

We are highly grateful to thank our Project coordinator **MANIMARAN A** and our Project Evaluator **IRANIYA PANDIAN** **JAYA ENGINEERING COLLEGE** for the coordinating us throughout this Project.

We are very much indebted to thank all the faculty members of Department of Computerscience and Engineering in our Institute, for their excellentmoral support and suggestions to complete our Project work successfully.

Finally, our acknowledgment does our parents, sisters, and friends those who had extended their excellent support and ideas to make our Project a pledge one.

SANJAY R
JAYA KUMAR S
ROHITH N
SENTHIL KUMARAN V
JAMESRAJ A

ABSTRACT

Agriculture is the main aspect of country development. Many people lead their life from agriculture field, which gives fully related to agricultural products. Plant disease, especially on leaves, is one of the major factors of reductions in both quality and quantity of the food crops. In agricultural aspects, if the plant is affected by leaf disease then it reduces the growth of the agricultural level. Finding the leaf disease is an important role of agriculture preservation. After pre-processing using a median filter, segmentation is done by Guided Active Contour method and finally, the leaf disease is identified by using Support Vector Machine. The disease-based similarity measure is used for fertilizer recommendation. Fertilizer Recommentation system for disease Prediction is a simple ML and DL based web Application which identify different diseases on plants by checking the symptoms shown on the leaves of the plant.

INDEX

c. User Stories

6. PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation

b. Sprint Delivery Schedule

c. Reports from JIRA

7.CODING & SOLUTIONING (Explain thefeatures added in theproject along with code)

a. Feature1

b. Feature2

c. Database Schema (if Applicable)

8.TESTING

a. User Acceptance Testing

b. Test Cases

9.RESULTS

a. Performance Metrics

10.ADVANTAGES & DISADVANTAGES

11.CONCLUSION

12.FUTURE SCOPE

13.APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

1.1 Overview In this project, two datasets name fruit dataset and vegetable dataset are collected. The collected datasets are trained and tested with deep learning neural network named Convolutional Neural Networks (CNN). First, the fruit dataset is trained and then tested with CNN. It has 6 classes and all the classes are trained and tested. Second, the vegetable dataset is trained and tested. The software used for training and testing of datasets is Python. All the Python codes are first written in Jupyter notebook supplied along with Anaconda Python and then the codes are tested in IBM cloud. Finally, a webbased framework is designed with help Flask a Python library. There are 2 html files are created in templates folder along with their associated files in static folder. The Python program 'app.py' used to interface with these two webpages is written in Spyder-Anaconda python and tested.

1.2 Purpose This project is used to test the fruits and vegetables samples and identify the different diseases. Also, this project recommends fertilizers for predicted diseases.

2.LITERATURE SURVEY

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

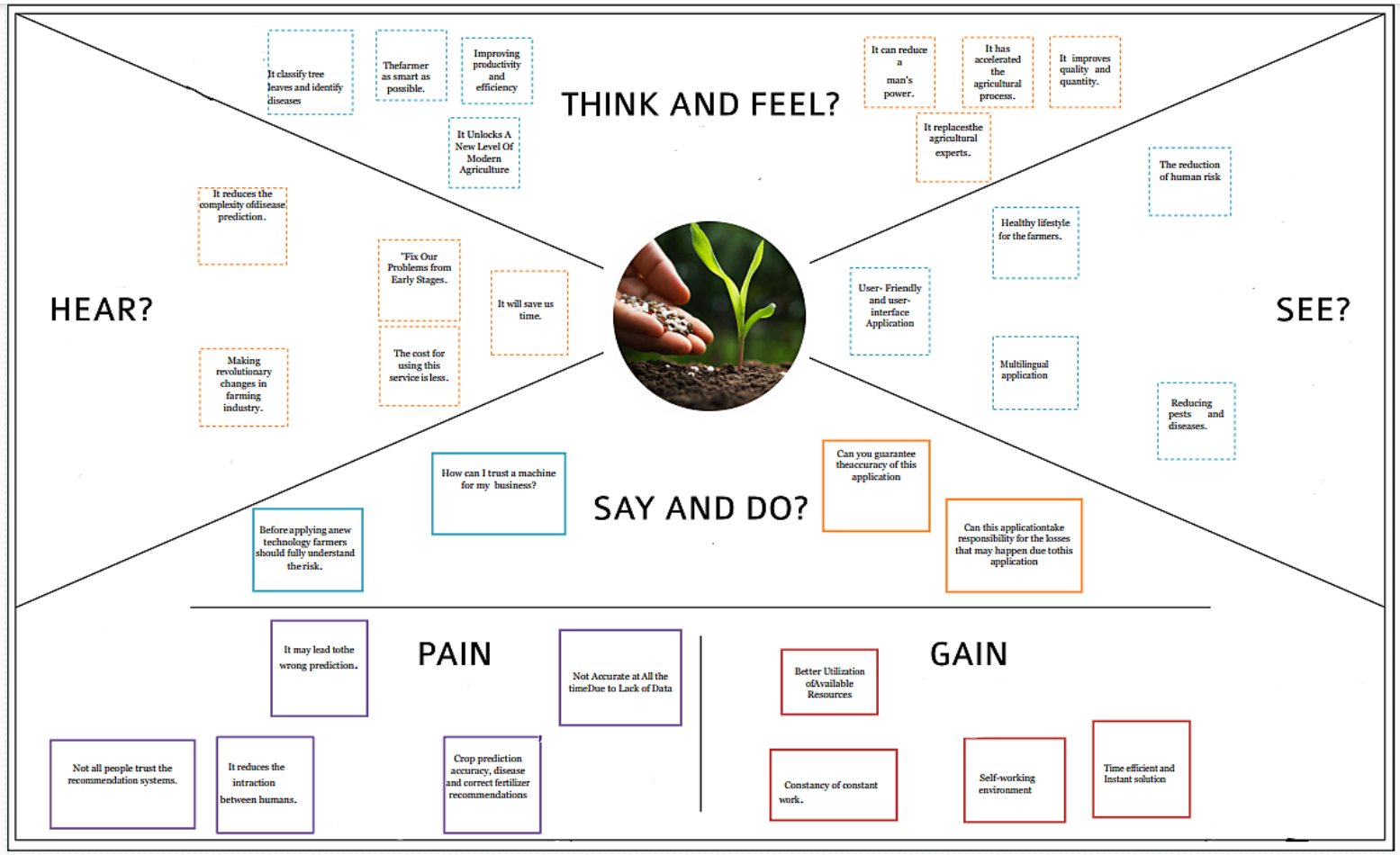
2.2 Neural Network Based Fertilizers Recommendation _System For Disorder Classification And Prediction In Petal Images .This methodology requires experts who can recognize varieties in leaf shading. Ordinarily a similar malady is characterized by a few specialists as a different sickness. This arrangement is exorbitant, in light of the fact that it requires nonstop expert management

2.3 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. In agricultural aspects, if the plant is affected by leaf disease then it reduces the growth and productiveness. Generally, the plant diseases are caused by

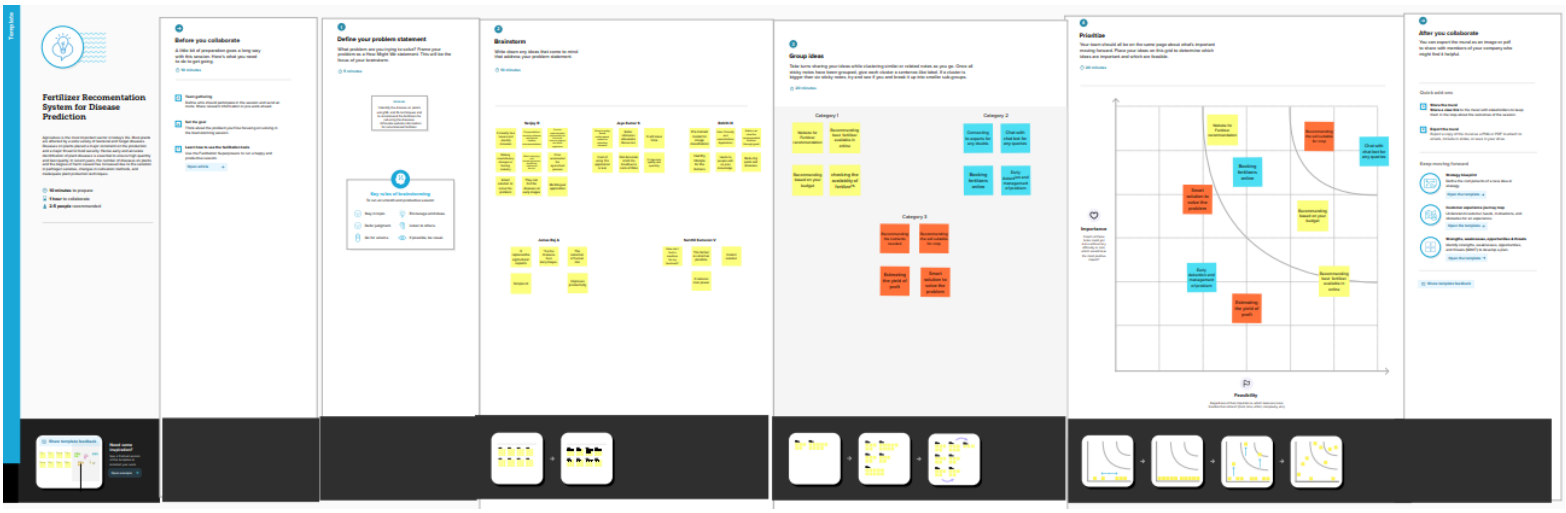
the abnormal physiological functionalities of plants

3.IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas



b. Ideation & Brainstorming

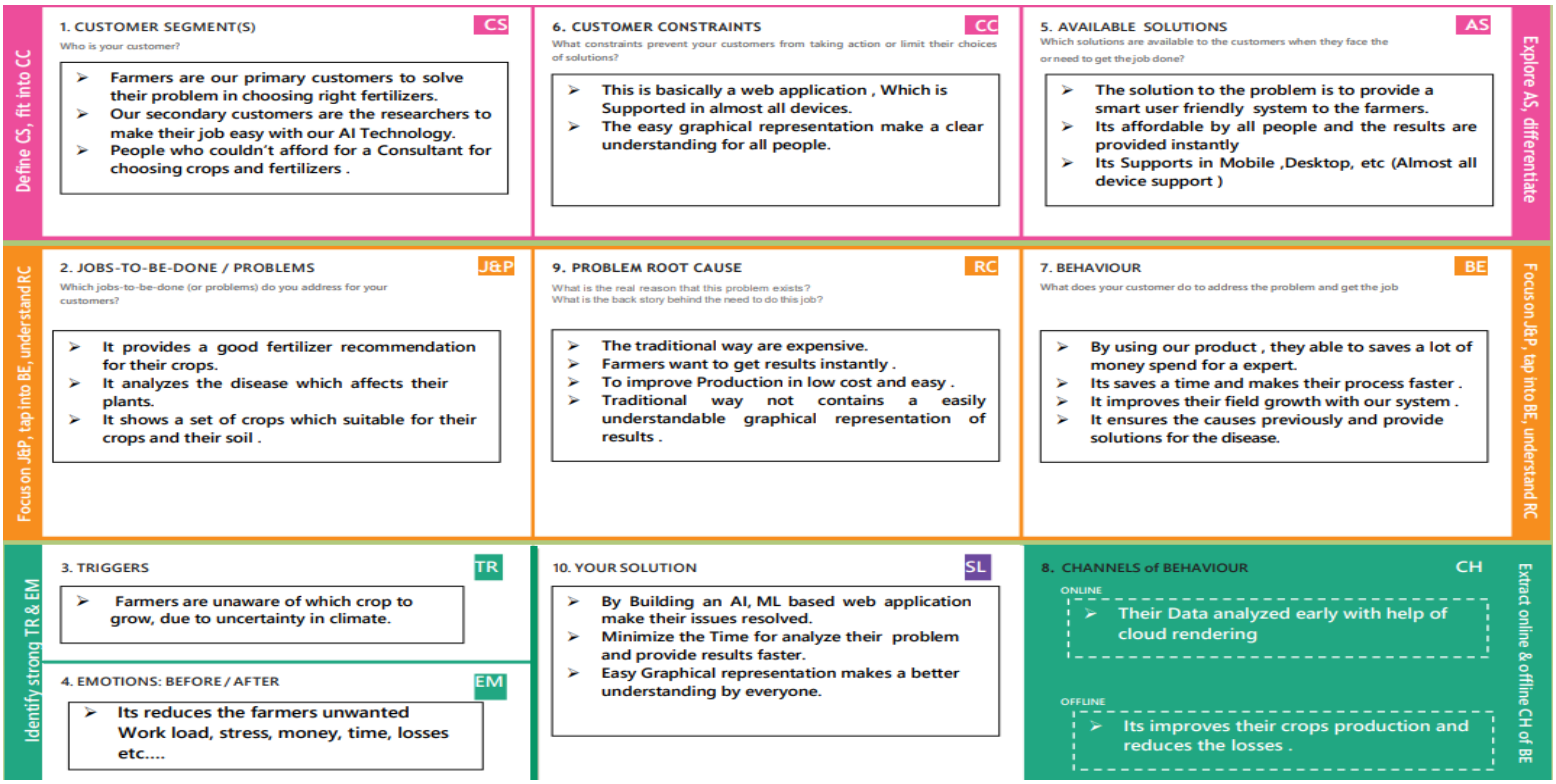


c. Proposed Solution

S.NO	Parameter	Description
1	Problem Statement (Problem to be solved)	An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.
2	Idea / Solution description	Develop an website for an farmers
3	Novelty / Uniqueness	The major agricultural products in India are rice, wheat, pulses, and spices. As our population is increasing rapidly the demand for agriculture products also increasing alarmingly. A huge amount of data are incremented from various field of agriculture
4	Social Impact / Customer Satisfaction	The project helps us to know about what type of disease were attacked in our plants how to recover using fertilizers are the main concept.
5	Business Model (Revenue Model)	Increasing demand for agricultural production owing to the growing population. Widening information management systems as well as new advanced technologies adoption for improving

		crop productivity
6	Scalability of the Solution	These tools are improving information about soils and vegetation that forms the basis for investments in management actions, provides early warning of pest and disease outbreaks, and facilitates the selection of sustainable cropland management practices.

d. Problem Solution fit



4.REQUIREMENT ANALYSIS

a.FUNCTIONAL REQUIREMENT

FR. NO	Functional requirement	Sub requirement (story/subtask)
FR-1	User registration	Registration through form Registration through Gmail
FR-2	User confirmation	Confirmation via OTP Confirmation via Email
FR-3	User Country	User should select the country to which they belong.
FR-4	Crop details	User can interact through providing details like image of the crop and soil or providing details of the soil like nitrogen, phosphorous, potassium, pH level.
FR-5	Prediction	The system will predict the issue from User details through train set and test data.
FR-6	Suggestion and Prevention	The system will suggest the solution to the issue through image or description.

b. NON FUNCTIONAL REQUIREMENT

FR No.	Non-Functional Requirement	Description
FR-1	Usability	The system is highly user friendly as the User can provide details and get suggestions from wherever they are. User can easily provide the details of their crop issue and get prevention methods and detects if the crop is affected by diseases.
FR-2	Security	The system provides good security. The system wants the User to provide only their Country and their crop details to detect the disease, suggest the solution and measures to prevent it.
FR-3	Reliability	The system will operate in all kind of environments with proper User and cropdetails.

		Farmers are unaware of which crop to grow, and what is the right time and place to start due to uncertainty in climatic conditions. So this application can be more useful for smart farming.
FR-4	Performance	This application provide most accurate prediction of diseases in crop and suggest the required methods to cure it and provide information like what kind of fertilizers need to be used. This application will be a great support to farmers.
FR-5	Availability	The system can be used by any farmers across the list of Countries mentioned in the application. Wherever the farmer is, they can use this application and get benefit from accurate identification of plant diseases which is essential to ensure high quantity and best quality of crops.
FR-6	Scalability	Provide nutrients not available in the soil. Replace nutrients removed at harvest. Balance nutrients for better produce quality and higher yield using artificial intelligence. Scalability is quick and high and also very simple to do.

5.PROJECT DESIGN

a.Data Flow Diagrams

b.Solution & Technical Architecture

S.NO	COMPONENT	DESCRIPTION	TECHNOLOGY
1.	User interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc	HTML, CSS, JavaScript , React Js etc.
2.	Prediction	Prediction of process in the Application	collaborativebased filtering technique, content-based technique, and hybrid algorithm.
3.	Evaluation	Logic of process in the Application	Java/Python
4.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc
5.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
6.	Image Preprocessing	the steps taken to format images before they are used by model training and inference	Python, Pytorch, OpenCV, Keras, Tensorflow, and Pillow.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	Train Data	To train an algorithm or machine learning	Machine learning algorithms

		model to predict the outcome of design	
9.	Machine Learning Model	Model Purpose of Machine Learning	Object Recognition Model, etc.

c.User Stories

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance**	**criteria Priority**	**Release**
Customer (Mobile user)	Application Download	USN-1	The farmer must download the appropriate application in their mobile .	The farmer must have enough storage space in his mobile	High	Sprint-1
	Registration	USN-2	As a user, the farmer can register for the application by entering email, password , and confirming the password.	Farmer can access his account / dashboard	High	Sprint-1
		USN-3	The Farmer will receive confirmation email once he had registered for the application	He can receive confirmation email & click confirm	High	Sprint-1
		USN-4	He can also register for the application through Google	He can register & access the dashboard with Google Login	low	Sprint-2
	Login	USN-5	As a user, The farmer can log into the application by entering email & password	He can login only when the user mail & password.	High	Sprint-1
	Dashboard	USN-6	The farmer has to go through the	He can can see the new	Medium	Sprint-1

			application to check out the available features	Available feature in the application		
		USN-7	The farmer can use the required features for their issues in his field.	He can solve the issues using the various fields	High	Sprint-1
Customer (Web user)	Registration	USN-8	As a user, the farmer can register for the application by entering email, password, and confirming the password	Farmer can access his account / dashboard	High	Sprint-1
		USN-9	The Farmer will receive confirmation email once he had registered for the application	He can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-10	As a user, The farmer can log in by entering email & password	He can login only when the user mail & password	High	Sprint-1
Customer Care Executive	Login issues	USN-11	Issues related to login can be reported there.	He can report the login related issue's	Medium	Sprint-1
	Queries related to prediction.	USN-12	Doubts regarding the predicted output can be rectified.	He can clear the doubts Using the predicted	High	Sprint-1
Administrator	Registration Confirmation	USN-13	Responding to farmer's registration with a confirmation mail.	He must have valid email and strong password	High	Sprint-1
	Database Management	USN-14	Farmer's inputs and predicted output are saved and managed.	Here the inputs and output are saved and managed	High	Sprint-1

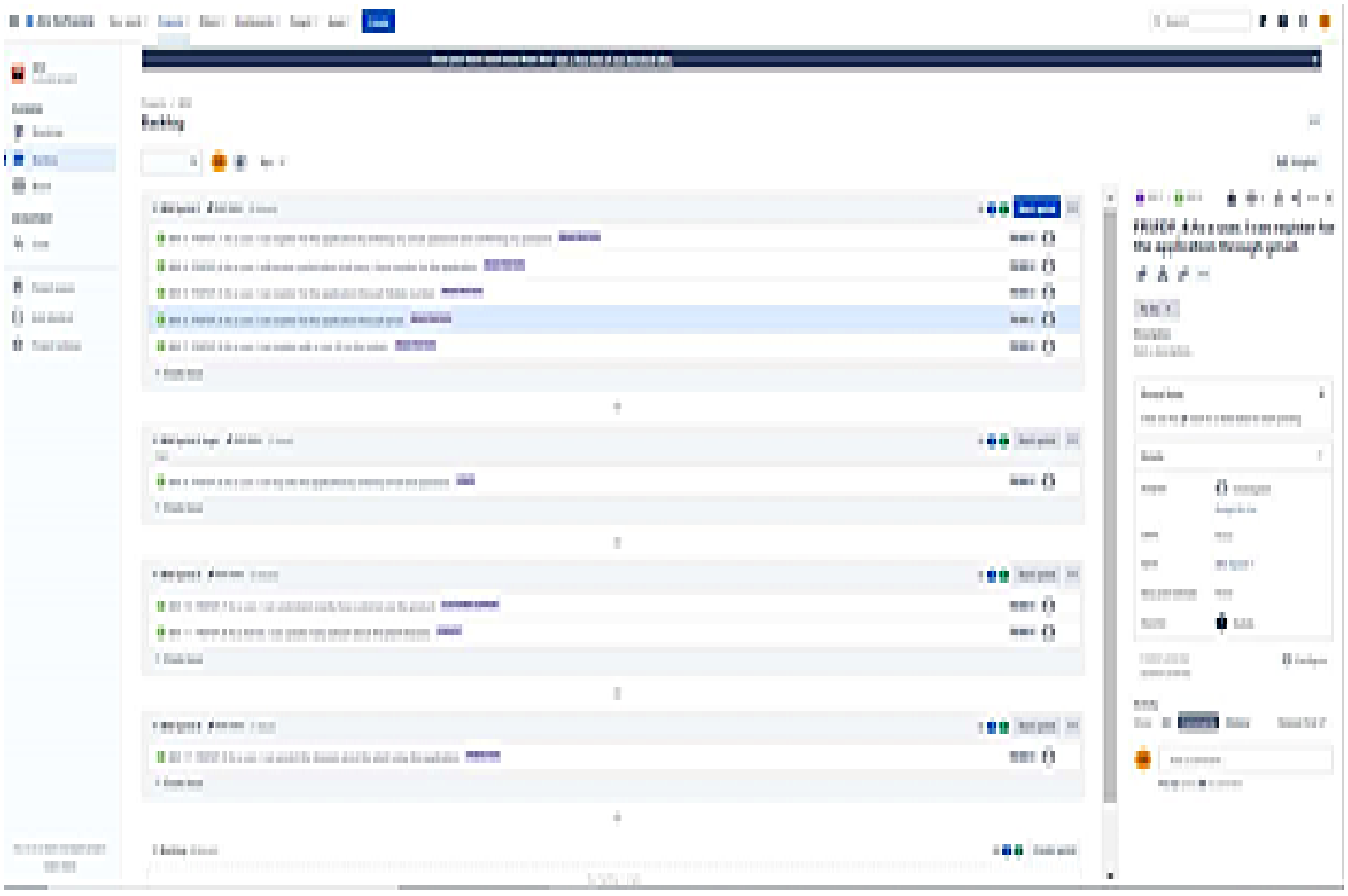
6.PROJECT PLANNING & SCHEDULING

a.Sprint Planning & Estimation

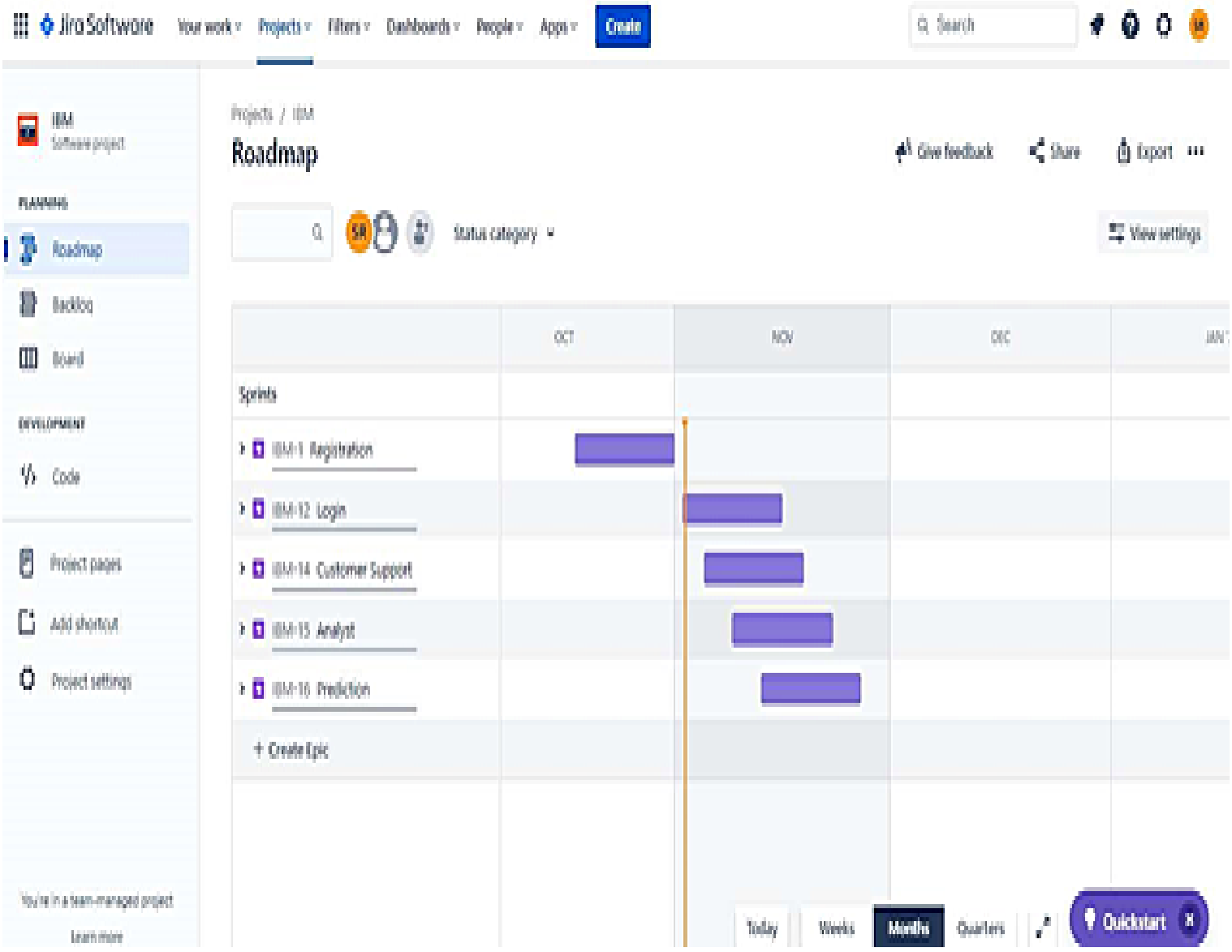
Sprint	Functional Requirement (Epic)	User StoryNumber	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Sanjay.R Jaya Kumar.S Rohith.N
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	JamesRaj.A Senthil Kumaran.V
Sprint-1		USN-3	As a user, I can register for the application through mobile number	1	Low	Sanjay.R
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Jaya Kumar.S Rohith.N
Sprint-1		USN-5	As a Web user, I can register with a User ID on the System	2	High	JamesRaj.A Senthil Kumaran.V
Sprint-2	Login	USN-6	As a user, I can log into the application by entering email & password	2	High	Sanjay.R Jaya Kumar.S Rohith.N
Sprint-3	Customer support	USN-7	As a Supporter, I can Understand exactly how customer use the product	1	Low	Sanjay.R
Sprint	Functional Requirement (Epic)	User StoryNumber	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Analyst	USN-8	As a Admin, I can Update many dataset about the Plant Diseases	2	High	Sanjay.R

B.Sprint delivery plan

Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022



Reports From Jira



7. **CODING & SOLUTIONING (Explain the Features added in the Project along with Code)**

7.1 Feature 1

```
<!DOCTYPE html>
<html lang="en">

<head>
	<title>{{ title }}</title>
	<link rel="shortcut icon" href="{{ url_for('static', filename='images/favicon.ico') }}"/>

	<!-- for-mobile-apps -->
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<meta charset="utf-8">
	<meta name="keywords" content="Agro Harvest Responsive web template, Bootstrap Web Templates, Flat Web Templates, Android Compatible web template,
Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyEricsson, Motorola web design" />

	<style>
		html {
			font-size: 1rem;
		}

		@media (min-width: 576px) {
			html {
				font-size: 1.25rem;
			}
		}

		@media (min-width: 768px) {
			html {
				font-size: 1.5rem;
```

```
        }
}

@media (min-width: 992px) {
        html {
                font-size: 1.75rem;
        }
}

@media (min-width: 1200px) {
        html {
                font-size: 2rem;
        }

        html {
                font-size: 1rem;
        }

        h1 {
                font-size: 1.2rem;
        }

        h2 {
                font-size: 1.1rem;
        }

        @media (min-width: 768px) {
                html {
                        font-size: 1.1rem;
                }

                h1 {
                        font-size: 1.3rem;
```

```
        }

        h2 {
                font-size: 1.2rem;
        }
}

@media (min-width: 991px) {
        html {
                font-size: 1.2rem;
        }

        h1 {
                font-size: 1.5rem;
        }

        h2 {
                font-size: 1.4rem;
        }
}

@media (min-width: 1200px) {
        html {
                font-size: 1.2rem;
        }

        h1 {
                font-size: 1.7rem;
        }

        h2 {
                font-size: 1.6rem;
        }
```

```
                }

            }
        </style>
        <script>
            addEventListener("load", function () {
                setTimeout(hideURLbar, 0);
            }, false);

            function hideURLbar() {
                window.scrollTo(0, 1);
            }

        </script>

        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
            integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
            crossorigin="anonymous"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
            integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
            crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
            integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
            crossorigin="anonymous"></script>
        <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
            integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
            crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
            integrity="sha384-
```

```
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
		crossorigin="anonymous"></script>
	</body>
	<!-- css files -->
	<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
		integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
	<link href="{{ url_for('static', filename='css/bootstrap.css') }}" rel='stylesheet' type='text/css' />
	<!-- bootstrap css -->
	<link href="{{ url_for('static', filename='css/style.css') }}" rel='stylesheet' type='text/css' />
	<!-- custom css -->
	<link href="{{ url_for('static', filename='css/font-awesome.min.css') }}" rel="stylesheet"><!--
fontawesome css -->
	<!-- //css files -->
	<!-- <link rel="icon" type="image/png" href="{{ url_for('static', filename='images/favicon.png?')
}}"> -->


	<script type="text/JavaScript" src="{{ url_for('static', filename='scripts/cities.js') }}"></script>


	<!-- google fonts -->
	<link
href="//fonts.googleapis.com/css?family=Thasadith:400,400i,700,700i&amp;subset=latin-
ext,thai,vietnamese"
		rel="stylesheet">
	<!-- //google fonts -->

	<style>
		header {
			background-color: rgba(30, 30, 30, 1);
```

```
                margin-top: 0rem;
                display: block;

            }
        </style>
</head>

<body>

        <!-- Navigation -->
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark static-top" style="background-color: #1C00ff00;">
            <div class="container">
                <a class="navbar-brand" href="{{ url_for('home') }}">
                    <img src="{{ url_for('static', filename='images/shutterstock-1263201358.jpg.jpg') }}" style="width: 90px; height: 55px" alt="">
                </a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive"
                    aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarResponsive">
                    <ul class="navbar-nav ml-auto">
                        <li class="nav-item active">
                            <a class="nav-link" href="{{ url_for('home') }}">Home
                                <span class="sr-only">(current)</span>
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('crop_recommend') }}">Crop</a>
```

```
                                </li>
                                <li class="nav-item">
                                        <a class="nav-link" href="{{ url_for('fertilizer_recommendation')
}}">prevention</a>
                                </li>
                                <li class="nav-item">
                                        <a class="nav-link" href="{{ url_for('disease_prediction')
}}">Disease</a>
                                </li>
                        </ul>
                </div>
        </div>
    </nav>



    {% block body %} {% endblock %}

    <!-- footer -->
    <footer class="text-center py-5">
        <div class="container py-md-3">
            <!-- logo -->
            <h2 class="logo2 text-center">
                <a href="{{ url_for('home') }}">
                    Leaf Life
                </a>
            </h2>
            <!-- //logo -->
            <!-- address -->
            <div class="contact-left-footer mt-4">
```

```
                        <!-- <a href="community.html">Community</a> -->
                        </p>
                    </div>
                    <div class="w3l-copy text-center">
                        <p class="text-da">An Environmental Intelligence for Crop Prevention<br>
</p>
                    </div>
                    <p class="homelogo">
                    <p>Made by CSE Boys</p>

                </div>
        </footer>
        <!-- //footer -->

        <!-- move top icon -->
        <a href="#home" class="move-top text-center"></a>
        <!-- //move top icon -->
</body>

</html>
```

7.2 Feature 2

```
# Importing essential libraries and modules

from flask import Flask, render_template, request, Markup
import numpy as np
import pandas as pd
from utils.disease import disease_dic
```

```
from utils.fertilizer import fertilizer_dic
import requests
import config
import pickle
import io
import torch
from torchvision import transforms
from PIL import Image
from utils.model import ResNet9
#
==============================================================================
==============

# ----------------------LOADING THE TRAINED MODELS -----------------------------------------

# Loading plant disease classification model

disease_classes = ['Apple___Apple_scab',
                   'Apple___Black_rot',
                   'Apple___Cedar_apple_rust',
                   'Apple___healthy',
                   'Blueberry___healthy',
                   'Cherry_(including_sour)___Powdery_mildew',
                   'Cherry_(including_sour)___healthy',
                   'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
                   'Corn_(maize)___Common_rust_',
                   'Corn_(maize)___Northern_Leaf_Blight',
                   'Corn_(maize)___healthy',
                   'Grape___Black_rot',
                   'Grape___Esca_(Black_Measles)',
                   'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
                   'Grape___healthy',
                   'Orange___Haunglongbing_(Citrus_greening)',
```

```
            'Peach___Bacterial_spot',
            'Peach___healthy',
            'Pepper,_bell___Bacterial_spot',
            'Pepper,_bell___healthy',
            'Potato___Early_blight',
            'Potato___Late_blight',
            'Potato___healthy',
            'Raspberry___healthy',
            'Soybean___healthy',
            'Squash___Powdery_mildew',
            'Strawberry___Leaf_scorch',
            'Strawberry___healthy',
            'Tomato___Bacterial_spot',
            'Tomato___Early_blight',
            'Tomato___Late_blight',
            'Tomato___Leaf_Mold',
            'Tomato___Septoria_leaf_spot',
            'Tomato___Spider_mites Two-spotted_spider_mite',
            'Tomato___Target_Spot',
            'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
            'Tomato___Tomato_mosaic_virus',
            'Tomato___healthy']

disease_model_path = 'models/plant_disease_model.pth'
disease_model = ResNet9(3, len(disease_classes))
disease_model.load_state_dict(torch.load(
    disease_model_path, map_location=torch.device('cpu')))
disease_model.eval()


# Loading crop recommendation model

crop_recommendation_model_path = 'models/RandomForest.pkl'
```

```
crop_recommendation_model = pickle.load(
    open(crop_recommendation_model_path, 'rb'))


#
==========================================================================
=========

# Custom functions for calculations


def weather_fetch(city_name):
    """
    Fetch and returns the temperature and humidity of a city
    :params: city_name
    :return: temperature, humidity
    """
    api_key = config.weather_api_key
    base_url = "http://api.openweathermap.org/data/2.5/weather?"

    complete_url = base_url + "appid=" + api_key + "&q=" + city_name
    response = requests.get(complete_url)
    x = response.json()

    if x["cod"] != "404":
        y = x["main"]

        temperature = round((y["temp"] - 273.15), 2)
        humidity = y["humidity"]
        return temperature, humidity
    else:
        return None
```

```
def predict_image(img, model=disease_model):
    """
    Transforms image to tensor and predicts disease label
    :params: image
    :return: prediction (string)
    """
    transform = transforms.Compose([
        transforms.Resize(256),
        transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
    img_t = transform(image)
    img_u = torch.unsqueeze(img_t, 0)

    # Get predictions from model
    yb = model(img_u)
    # Pick index with highest probability
    _, preds = torch.max(yb, dim=1)
    prediction = disease_classes[preds[0].item()]
    # Retrieve the class label
    return prediction

#
================================================================================
===============
# ---------------------------------- FLASK APP -----------------------------------------------


app = Flask(__name__)

# render home page
```

```
@ app.route('/')
def home():
    title = 'Leaf Life- Home'
    return render_template('index.html', title=title)

# render crop recommendation form page


@ app.route('/crop-recommend')
def crop_recommend():
    title = 'Leaf Life - Crop Recommendation'
    return render_template('crop.html', title=title)

# render fertilizer recommendation form page


@ app.route('/fertilizer')
def fertilizer_recommendation():
    title = 'Leaf Life - Fertilizer Suggestion'

    return render_template('fertilizer.html', title=title)

# render disease prediction input page




#
==========================================================================
==============

# RENDER PREDICTION PAGES
```

```
# render crop recommendation result page


@ app.route('/crop-predict', methods=['POST'])
def crop_prediction():
    title = 'Leaf Life - Crop Recommendation'

    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
        K = int(request.form['pottasium'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])

        # state = request.form.get("stt")
        city = request.form.get("city")

        if weather_fetch(city) != None:
            temperature, humidity = weather_fetch(city)
            data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
            my_prediction = crop_recommendation_model.predict(data)
            final_prediction = my_prediction[0]

            return render_template('crop-result.html', prediction=final_prediction, title=title)

        else:

            return render_template('try_again.html', title=title)

# render fertilizer recommendation result page
@ app.route('/fertilizer-predict', methods=['POST'])
def fert_recommend():
```

```
title = 'Leaf Life - Fertilizer Suggestion'
crop_name = str(request.form['cropname'])
  N = int(request.form['nitrogen'])
  P = int(request.form['phosphorous'])
  K = int(request.form['pottasium'])
  # ph = float(request.form['ph'])

  df = pd.read_csv('Data/fertilizer.csv')

  nr = df[df['Crop'] == crop_name]['N'].iloc[0]
  pr = df[df['Crop'] == crop_name]['P'].iloc[0]
  kr = df[df['Crop'] == crop_name]['K'].iloc[0]

  n = nr - N
  p = pr - P
  k = kr - K
  temp = {abs(n): "N", abs(p): "P", abs(k): "K"}
  max_value = temp[max(temp.keys())]
  if max_value == "N":
    if n < 0:
      key = 'NHigh'
    else:
      key = "Nlow"
  elif max_value == "P":
    if p < 0:
      key = 'PHigh'
    else:
      key = "Plow"
  else:
    if k < 0:
      key = 'KHigh'
    else:
      key = "Klow"
```

```
    response = Markup(str(fertilizer_dic[key]))

    return render_template('fertilizer-result.html', recommendation=response, title=title)

# render disease prediction result page


@app.route('/disease-predict', methods=['GET', 'POST'])
def disease_prediction():
    title = 'Leaf Life- Disease Detection'

    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files.get('file')
        if not file:
            return render_template('disease.html', title=title)
        try:
            img = file.read()

            prediction = predict_image(img)

            prediction = Markup(str(disease_dic[prediction]))
            return render_template('disease-result.html', prediction=prediction, title=title)
        except:
            pass
    return render_template('disease.html', title=title)
#
================================================================================
==============
if __name__ == '__main__':
    app.run(debug=False)
```

8.TESTING

8.1 Test Case

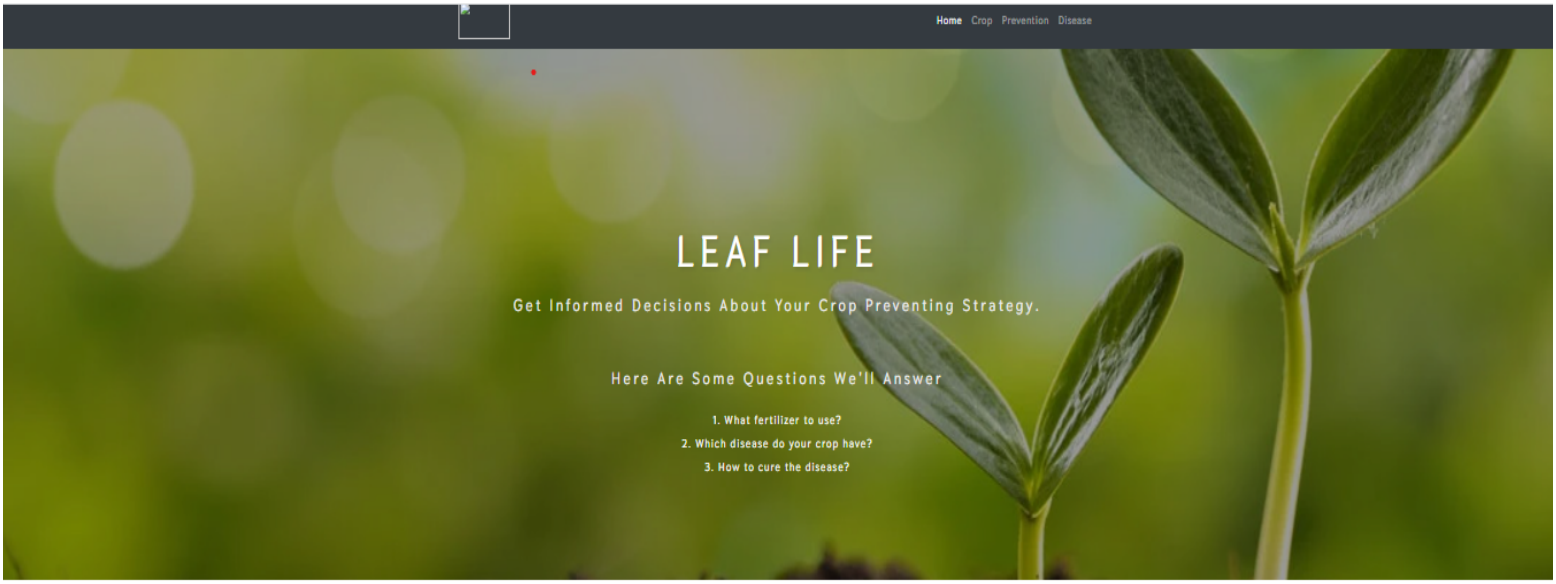
SECTION	TOTAL CASES	NOT TESTED	FAIL	PASS
Leaf spots	17	0	0	17
Mosaic Leaf Pattern	51	0	0	51
Misshapen Leaves	20	0	0	20
Yellow Leaves	7	0	0	7
Fruit Rots	9	0	0	9
Fruit Spots	4	0	0	4
Blights	2	0	0	2

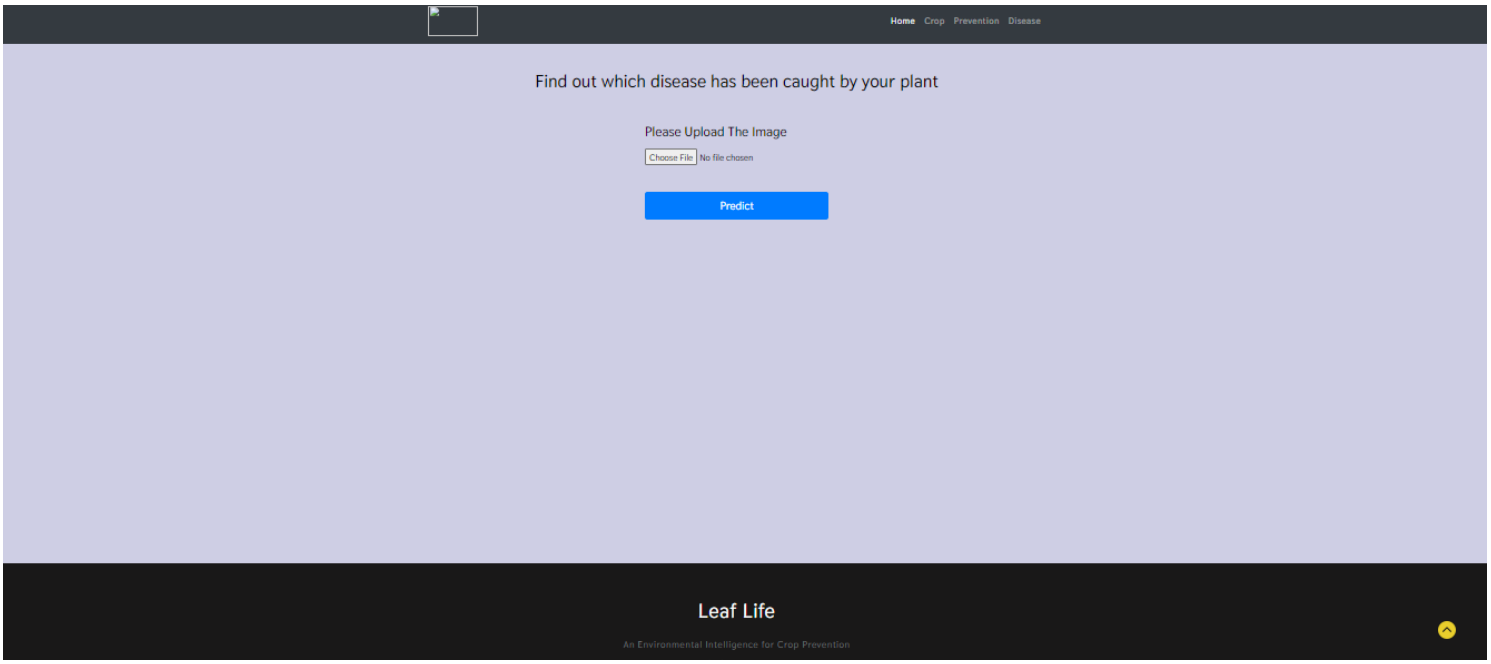
8.2 User Acceptance Testing

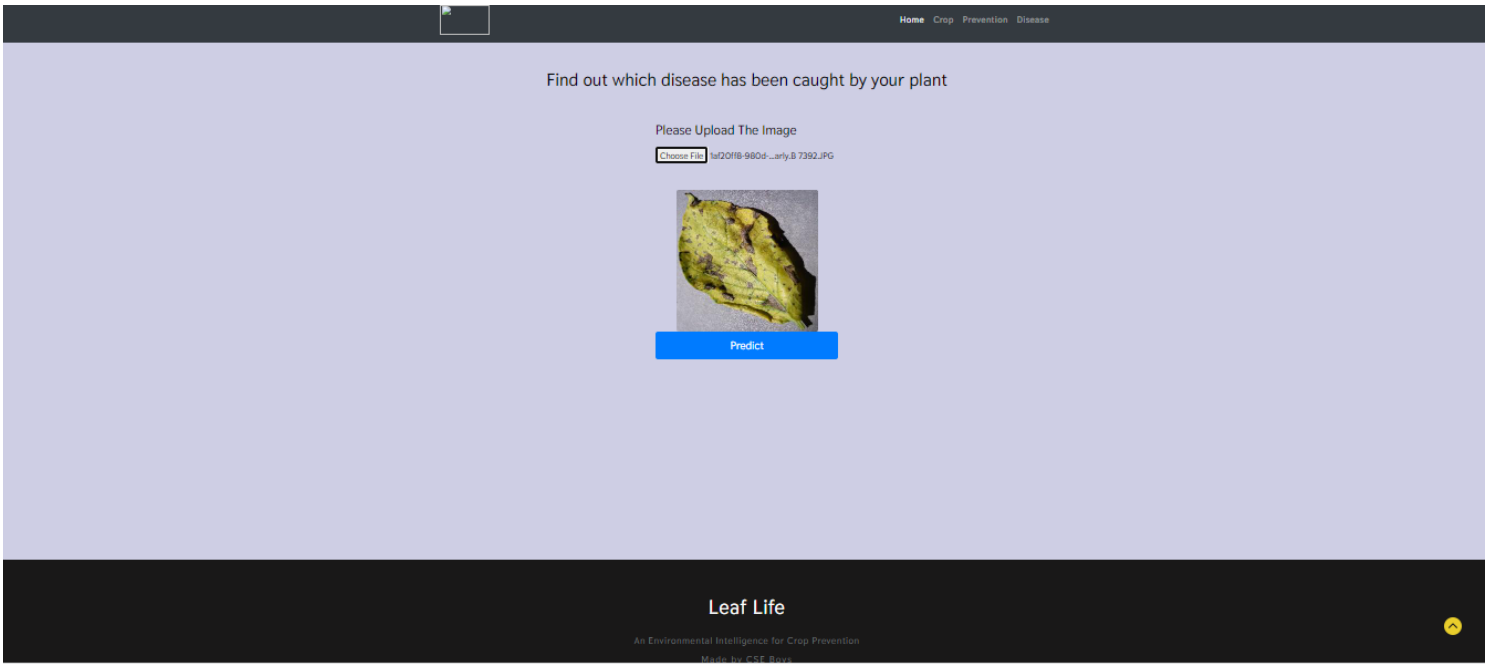
RESOLUTION	SEVERITY 1	SEVERITY 2	SEVERITY 3	SEVERITY 4	SUBTOTAL
Leaf spots	10	4	2	3	19
Mosaic Leaf Pattern	9	6	3	6	24
Misshapen Leaves	2	7	0	1	10
Yellow Leaves	11	4	3	20	38
Fruit Rots	3	2	1	0	6
Fruit Spots	5	3	1	1	10
Blights	4	5	2	1	12
Totals	44	31	13	32	119

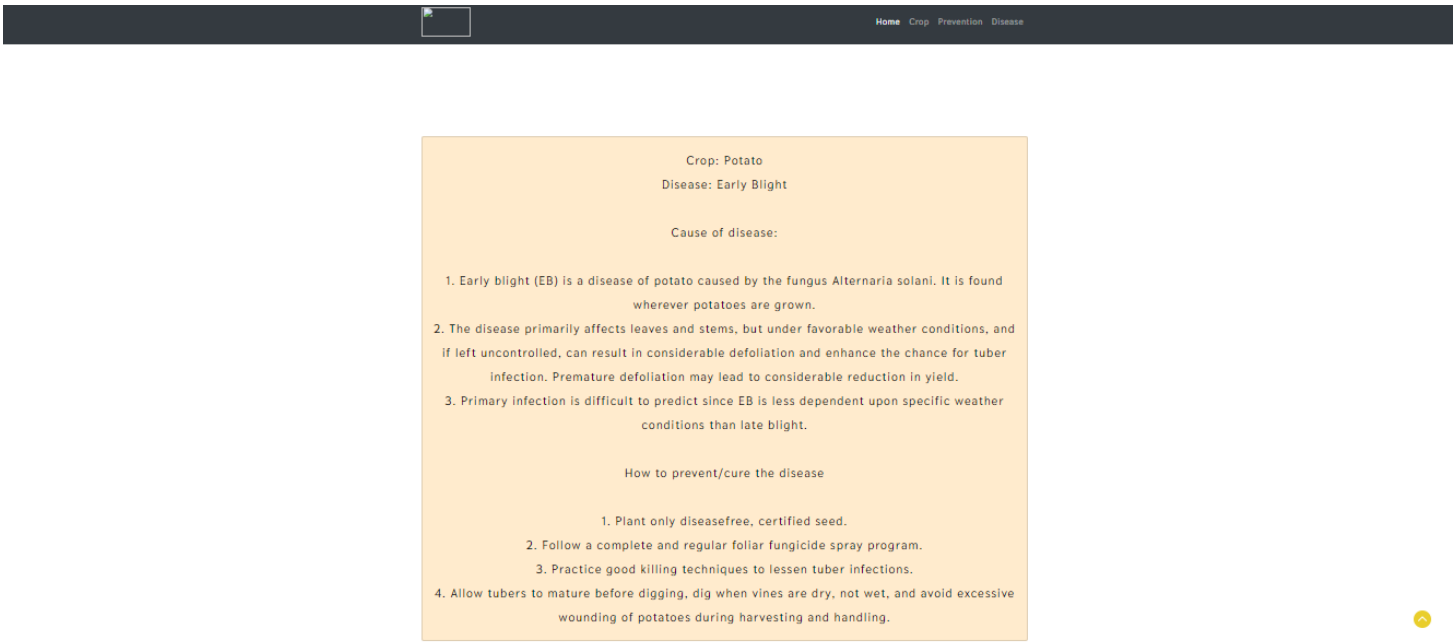
9.RESULTS

a.Performance Metrics




Home Crop Prevention Disease
Find out which disease has been caught by your plant
Please Upload The Image
Choose File No file chosen
Predict
Leaf Life
An Environmental Intelligence for Crop Prevention


Home Crop Prevention Disease
Find out which disease has been caught by your plant
Please Upload The Image
Choose File 1af20ff8-980d-...arly.B 7392.JPG
Predict
Leaf Life
An Environmental Intelligence for Crop Prevention
Made by CSE Boys



10. ADVANTAGES & DISADVANTAGES

List of advantages

- The proposed model here produces very high accuracy of classification.
- Very large datasets can also be trained and tested.
- Images of very high can be resized within the proposed itself.

List of disadvantages

- For training and testing, the proposed model requires very high computational time.
- The neural network architecture used in this project work has high complexity.

11. CONCLUSION

The model proposed here involves image classification of fruit datasets and vegetable datasets. The following points are observed during model testing and training:

- The accuracy of classification increased by increasing the number of epochs.
- For different batch sizes, different classification accuracies are obtained.

- The accuracies are increased by increasing more convolution layers.
- The accuracy of classification also increased by varying dense layers.
- Different accuracies are obtained by varying the size of kernel used in the convolution layer output.
- Accuracies are different while varying the size of the train and test datasets.

12. FUTURE SCOPE

The proposed model in this project work can be extended to image recognition. The entire model can be converted to application software using python to exe software. The real time image classification, image recognition and video processing are possible with help OpenCV python library. This project work can be extended for security applications such as figure print recognition, iris recognition and face recognition

13. APPENDIX

Source Code

```
# Importing essential libraries and modules

from flask import Flask, render_template, request, Markup
import numpy as np
import pandas as pd
from utils.disease import disease_dic
from utils.fertilizer import fertilizer_dic
import requests
import config
import pickle
import io
import torch
from torchvision import transforms
from PIL import Image
from utils.model import ResNet9
# ===============================================================================================

# ------------------------LOADING THE TRAINED MODELS -----------------------------------------------

# Loading plant disease classification model

disease_classes = ['Apple___Apple_scab',
                   'Apple___Black_rot',
                   'Apple___Cedar_apple_rust',
                   'Apple___healthy',
```

```
            'Blueberry___healthy',
            'Cherry_(including_sour)___Powdery_mildew',
            'Cherry_(including_sour)___healthy',
            'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
            'Corn_(maize)___Common_rust_',
            'Corn_(maize)___Northern_Leaf_Blight',
            'Corn_(maize)___healthy',
            'Grape___Black_rot',
            'Grape___Esca_(Black_Measles)',
            'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
            'Grape___healthy',
            'Orange___Haunglongbing_(Citrus_greening)',
            'Peach___Bacterial_spot',
            'Peach___healthy',
            'Pepper,_bell___Bacterial_spot',
            'Pepper,_bell___healthy',
            'Potato___Early_blight',
            'Potato___Late_blight',
            'Potato___healthy',
            'Raspberry___healthy',
            'Soybean___healthy',
            'Squash___Powdery_mildew',
            'Strawberry___Leaf_scorch',
            'Strawberry___healthy',
            'Tomato___Bacterial_spot',
            'Tomato___Early_blight',
            'Tomato___Late_blight',
            'Tomato___Leaf_Mold',
            'Tomato___Septoria_leaf_spot',
            'Tomato___Spider_mites Two-spotted_spider_mite',
            'Tomato___Target_Spot',
            'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
            'Tomato___Tomato_mosaic_virus',
            'Tomato___healthy']

disease_model_path = 'models/plant_disease_model.pth'
disease_model = ResNet9(3, len(disease_classes))
disease_model.load_state_dict(torch.load(
    disease_model_path, map_location=torch.device('cpu')))
disease_model.eval()
```

```
# Loading crop recommendation model

crop_recommendation_model_path = 'models/RandomForest.pkl'
crop_recommendation_model = pickle.load(
    open(crop_recommendation_model_path, 'rb'))


# ===============================================================================================

# Custom functions for calculations


def weather_fetch(city_name):
    """
    Fetch and returns the temperature and humidity of a city
    :params: city_name
    :return: temperature, humidity
    """
    api_key = config.weather_api_key
    base_url = "http://api.openweathermap.org/data/2.5/weather?"

    complete_url = base_url + "appid=" + api_key + "&q=" + city_name
    response = requests.get(complete_url)
    x = response.json()

    if x["cod"] != "404":
        y = x["main"]

        temperature = round((y["temp"] - 273.15), 2)
        humidity = y["humidity"]
        return temperature, humidity
    else:
        return None


def predict_image(img, model=disease_model):
    """
    Transforms image to tensor and predicts disease label
    :params: image
    :return: prediction (string)
```

```
    transform = transforms.Compose([
        transforms.Resize(256),
        transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
    img_t = transform(image)
    img_u = torch.unsqueeze(img_t, 0)

    # Get predictions from model
    yb = model(img_u)
    # Pick index with highest probability
    _, preds = torch.max(yb, dim=1)
    prediction = disease_classes[preds[0].item()]
    # Retrieve the class label
    return prediction

# ============================================================================================================
# ---------------------------------- FLASK APP -------------------------------------------------


app = Flask(__name__)

# render home page


@ app.route('/')
def home():
    title = 'Leaf Life- Home'
    return render_template('index.html', title=title)

# render crop recommendation form page


@ app.route('/crop-recommend')
def crop_recommend():
    title = 'Leaf Life - Crop Recommendation'
    return render_template('crop.html', title=title)

# render fertilizer recommendation form page
@ app.route('/fertilizer')
def fertilizer_recommendation():
```

```
    title = 'Leaf Life - Fertilizer Suggestion'

    return render_template('fertilizer.html', title=title)

# render disease prediction input page
# ===============================================================================================

# RENDER PREDICTION PAGES

# render crop recommendation result page


@ app.route('/crop-predict', methods=['POST'])
def crop_prediction():
    title = 'Leaf Life - Crop Recommendation'

    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
        K = int(request.form['pottasium'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])

        # state = request.form.get("stt")
        city = request.form.get("city")

        if weather_fetch(city) != None:
            temperature, humidity = weather_fetch(city)
            data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
            my_prediction = crop_recommendation_model.predict(data)
            final_prediction = my_prediction[0]

            return render_template('crop-result.html', prediction=final_prediction, title=title)

        else:

            return render_template('try_again.html', title=title)
# render fertilizer recommendation result pag
@ app.route('/fertilizer-predict', methods=['POST'])
def fert_recommend():
    title = 'Leaf Life - Fertilizer Suggestion'
```

```
    crop_name = str(request.form['cropname'])
    N = int(request.form['nitrogen'])
    P = int(request.form['phosphorous'])
    K = int(request.form['pottasium'])
    # ph = float(request.form['ph'])

    df = pd.read_csv('Data/fertilizer.csv')

    nr = df[df['Crop'] == crop_name]['N'].iloc[0]
    pr = df[df['Crop'] == crop_name]['P'].iloc[0]
    kr = df[df['Crop'] == crop_name]['K'].iloc[0]

    n = nr - N
    p = pr - P
    k = kr - K
    temp = {abs(n): "N", abs(p): "P", abs(k): "K"}
    max_value = temp[max(temp.keys())]
    if max_value == "N":
        if n < 0:
            key = 'NHigh'
        else:
            key = "Nlow"
    elif max_value == "P":
        if p < 0:
            key = 'PHigh'
        else:
            key = "Plow"
    else:
        if k < 0:
            key = 'KHigh'
        else:
            key = "Klow"

    response = Markup(str(fertilizer_dic[key]))

    return render_template('fertilizer-result.html', recommendation=response, title=title)

# render disease prediction result page

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

```
def disease_prediction():
    title = 'Leaf Life- Disease Detection'

    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files.get('file')
        if not file:
            return render_template('disease.html', title=title)
        try:
            img = file.read()

            prediction = predict_image(img)

            prediction = Markup(str(disease_dic[prediction]))
            return render_template('disease-result.html', prediction=prediction, title=title)
        except:
            pass
    return render_template('disease.html', title=title)

# ===============================================================================================
if __name__ == '__main__':
    app.run(debug=False)
```

GitHub & Project Demo Link

https://github.com/IBM-EPBL/IBM-Project-45088-1660728208

Project Demo Link

https://youtu.be/bY22--HX0Po