FERTILIZER RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

A PROJECT REPORT

Submitted by

ISHWARYA K (917719IT035)

JAYASREE G (917719IT038)

JUHI PADMAJA P (917719IT042)

RESHMA A (917719IT077)

In the course of,

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP, 19ES790

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI - 15

NOVEMBER 2022

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

In today's society, agriculture is the most significant industry. An extensive range of bacterial and fungal diseases harm the majority of plants. Plant diseases severely limit productivity and pose 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. An automated technique is now available to recognise many plant diseases by examining the symptoms seen on the plant's leaves. Deep learning algorithms are used to diagnose diseases and provide preventative measures that

1.2 Purpose

Detection and recognition of plant diseases using machine learning are very efficient in providing symptoms of identifying diseases at its earliest. Plant pathologists can analyze the digital images using digital image processing for diagnosis of plant diseases. Identifying the plant disease in prior and recommending the most appropriate fertilizers to be used helps the farmers to increase the crop yield.

2. LITERATURE SURVEY

2.1 Existing problem

S. No	Application Details	Company	About the Application
1	NGT - Agricultural Pest Prediction and Advisory (APPA)	Techuva solutions Pvt Ltd	Next to weather variability, the occurrence of pests and diseases in crops is one of the main reasons for yield loss. Climate change is the main culprit of agricultural pest progress and outbreaks.

			Better controlling the outbreaks of pests and diseases can therefore greatly contribute to the food security and livelihood of Indian farmers. Here it is the integrated solution to increase the yield and profit by timely effective management practices with reducing the cost of input and safe environment.
	Plantix - Your crop doctor	Plantix	Plantix turns your Android phone into a mobile crop doctor with which you can accurately detect pests and diseases on crops within seconds. Plantix serves as a complete solution for crop production and management. What Plantix Offers, ● Heal Your Crop ● Disease Alerts ● Farmer Community ● Cultivation Tips ● Agri Weather Forecast ● Fertilizer Calculator ● Diagnose and Treat Crop Issues

2.2 References

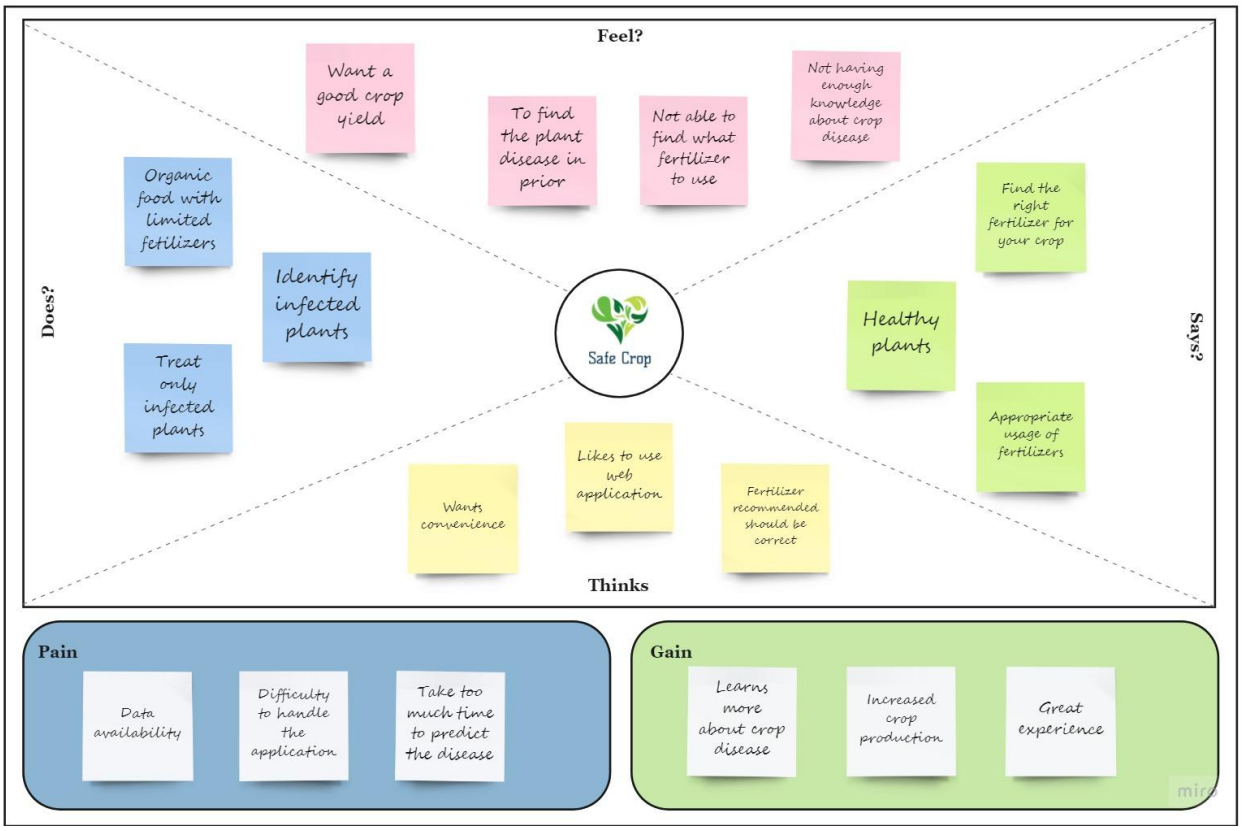
- https://play.google.com/store/apps/details?id=com.peat.GartenBank&hl=en_IN&gl=US
- https://play.google.com/store/apps/details?id=com.techuva.iot.ngt&hl=en_CA&gl=US&pli=1

2.3 Problem Statement Definition

The major problems that the farmers of our country are currently facing include Crop Failure, Lack of adequate knowledge, Crop damage due to ignorance/carelessness, Lack of professional assistance, Inaccessibility to agro-tech solutions. The website that we are going to develop will help the farmers to deal with these problems by providing the following aids: Fertilizer suggestion system, Crop Disease Detection System. We will develop a website that will detect crop diseases by scanning the leaves of the crops and recommending the fertilizer that has to be used to get rid of the disease.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

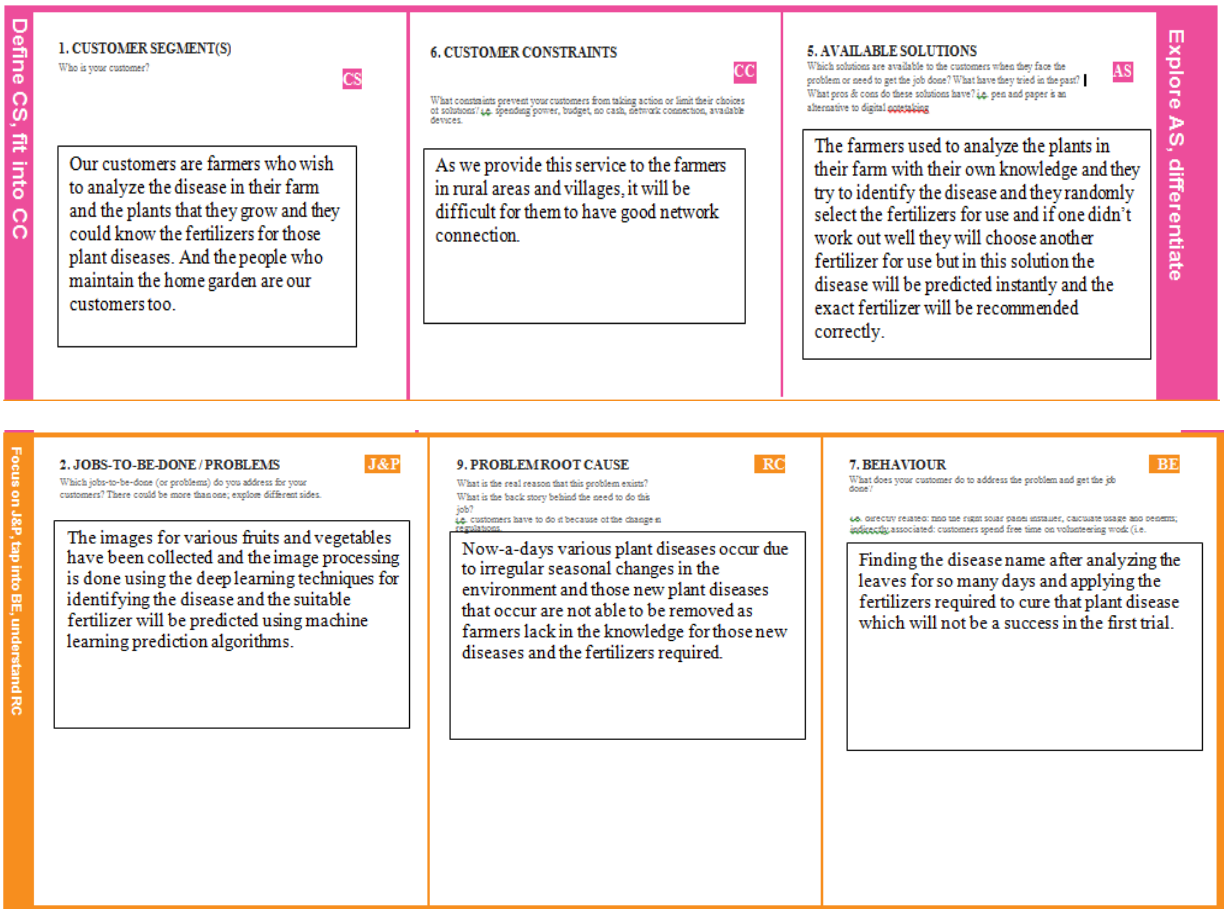
3.3 Proposed Solution

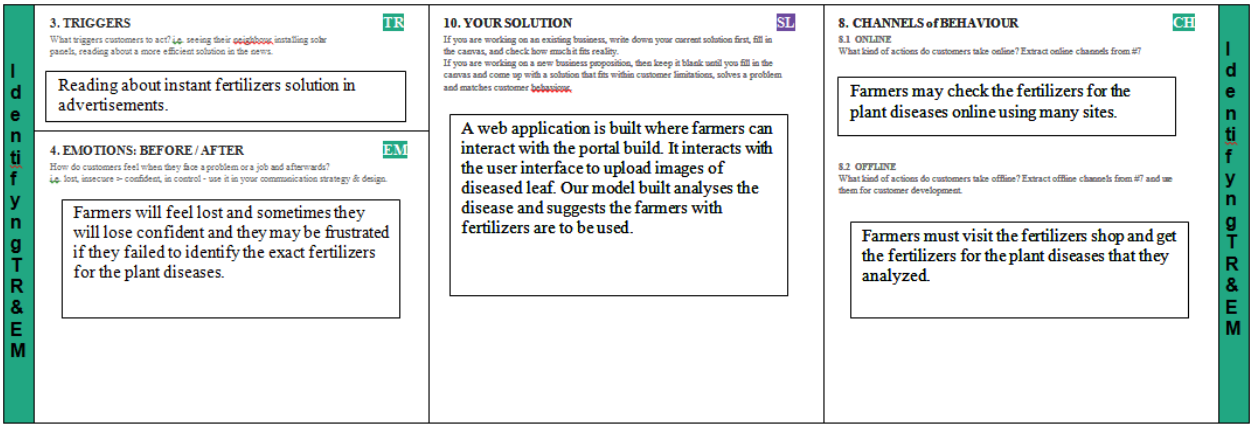
S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Not all farmers are expert in nuances of agriculture, few may be novice. Novice cannot accurately predict whether a plant is affected by any disease or not and the amount of fertilizers to be sprayed to get rid of the disease. So we have decided to develop a model for disease prediction and fertilizer recommendation.

2.	Idea/ Solution Description	In our project we have planned to develop a web application which will be helpful for the farmers for predicting the crop infestation. It can be predicted by emerging technologies like AI, ML and DL algorithms. Along with prediction, we have planned to recommend the proper pesticide and quantity of the pesticide that should be used in order to recover the crop from deterioration. First the train and test image dataset is preprocessed and CNN algorithm is applied to build a neural network for predicting the crop disease. A web application using Flask is created as an interface for the farmers to use.
3.	Uniqueness/ Novelty	· To Check whether the crop in the field is affected by any pest or not. · Recommending the fertilizer to the farmer if the crop is affected by any disease. · It also recommends the amount of fertilizer to use.
4.	Social Impact/ Customer Satisfaction	By letting the farmers know about their crops' condition might be helpful for them to make the right decision at the right time and it also helps them to increase the yield by protecting the plants from deterioration.

5.	Business model (Revenue model)	· Provide the farmers the most relevant and expected result they are looking for. · Additionally, we must bear in mind the concept of personalization according to the user's needs.
6.	Scalability of the solution	· Functional quality of the web application will never get compromised; it will be available at every time. · The time it takes for the request and response is very less.

3.4 Problem Solution fit





4. REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No	Functional Requirements
FR 1	Uploading the image of the plant leaf which is affected by the disease.
FR 2	Proper disease prediction.
FR 3	Timely fertilizer recommendation.

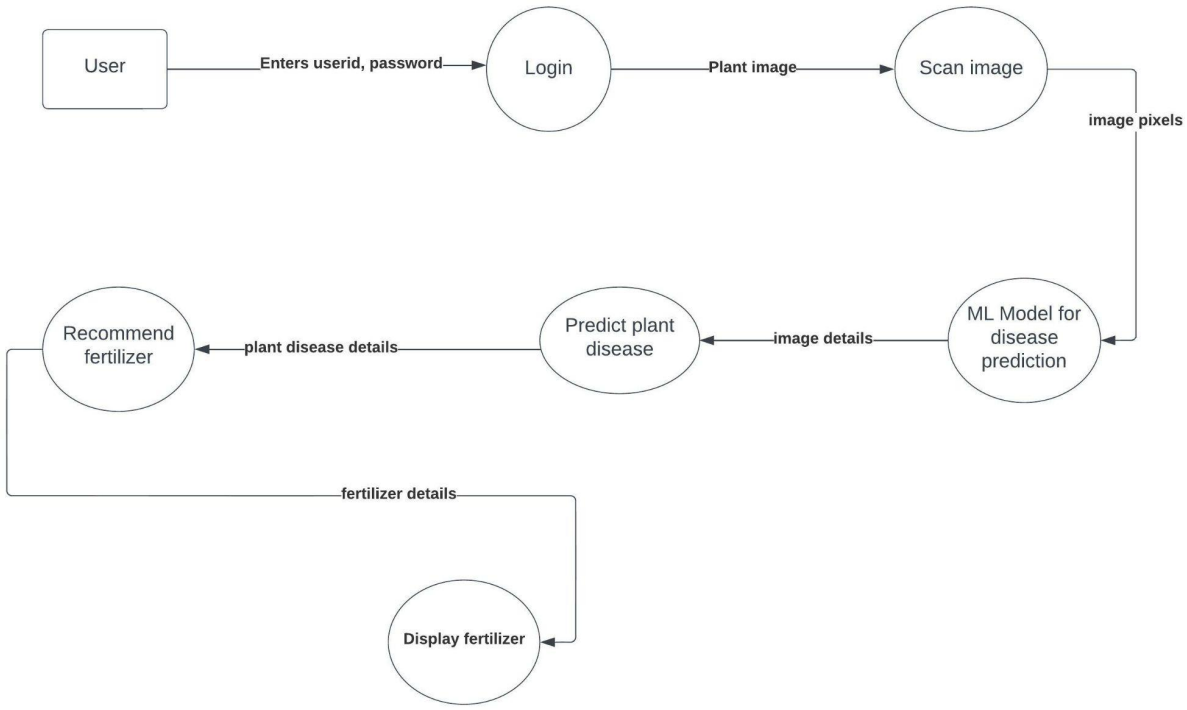
4.2 Non-Functional requirements

NFR No	Non - Functional Requirements
NFR 1	The performance of the application should be high.
NFR 2	The responsiveness of the application should be user friendly.
NFR 3	The application should be safe and secure to use.

NFR 4	Serviceability of the application should be high and proper.

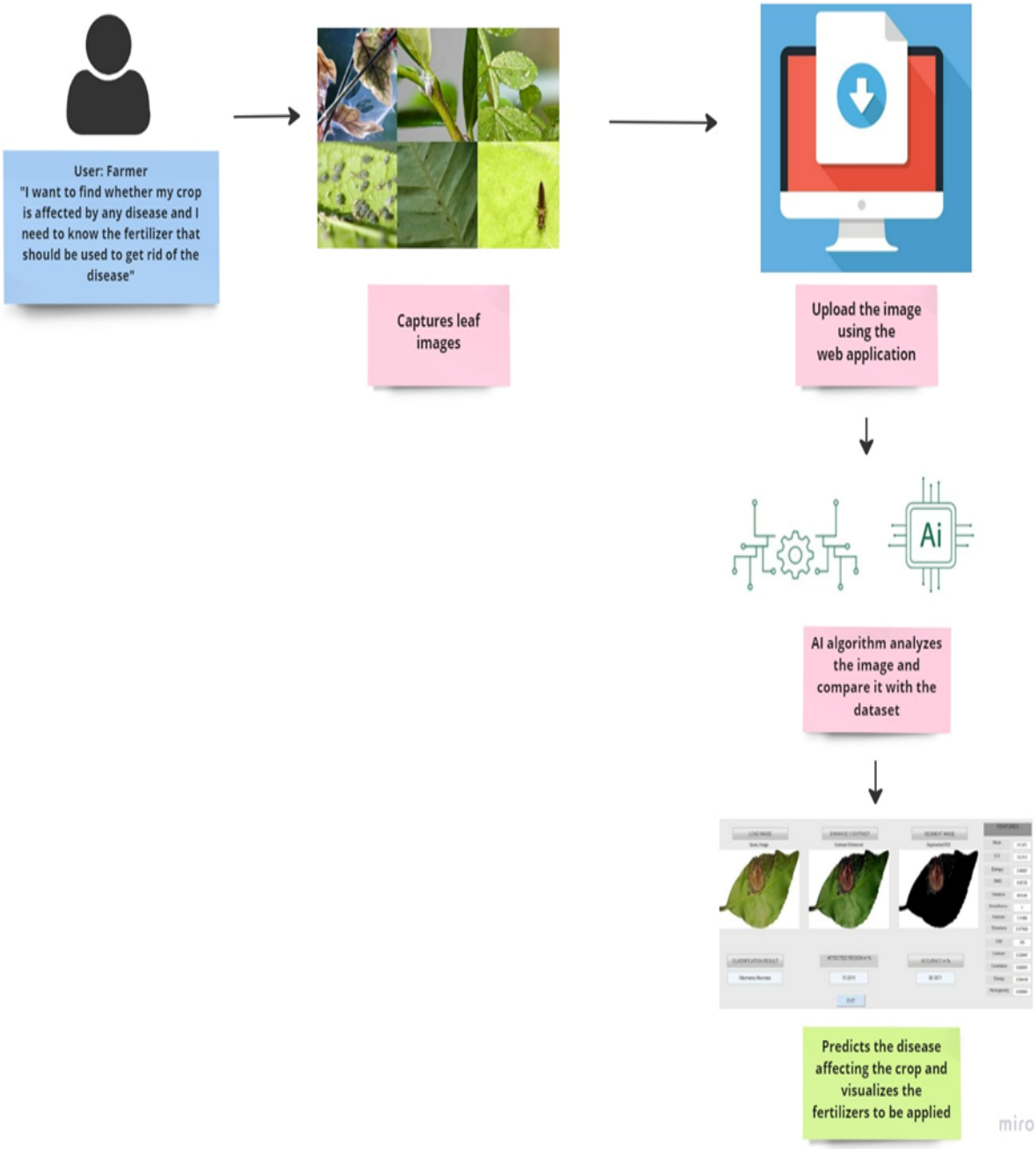
5. PROJECT DESIGN

5.1 Data Flow Diagrams

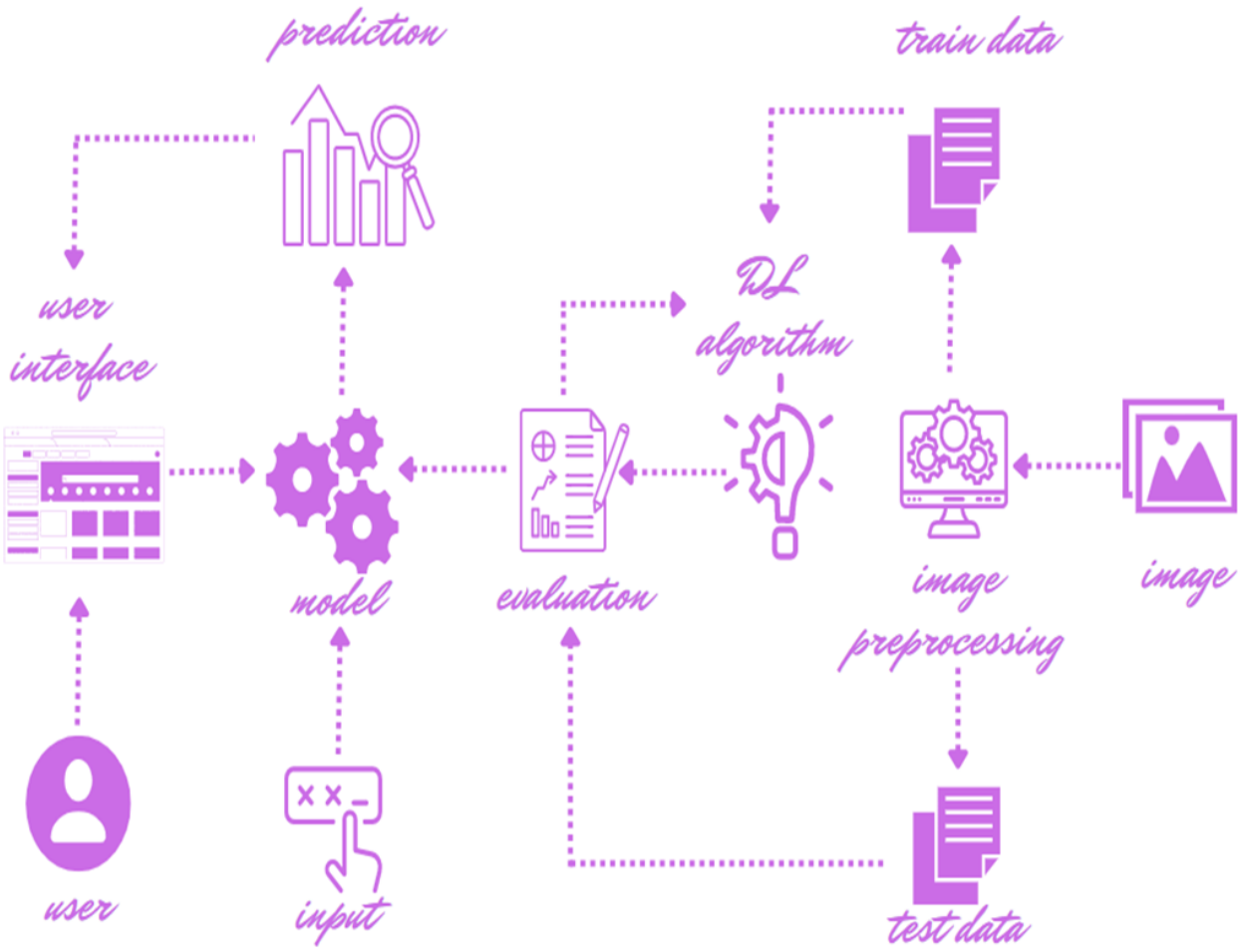


5.2 Solution & Technical Architecture

5.2.1 Solution Architecture



5.2.2 Technical Architecture



5.3 User Stories

As a user, I could able to upload the crop image.	As a user, I can view the plant disease information.	As a user, I can view fertilizer suitable for the crop.	As a user, I could able to access the application in offline.

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022		2 Nov 2022
Sprint-2	15	6 Days	31 Oct 2022	05 Nov 2022		06 Nov 2022
Sprint-3	15	6 Days	07 Nov 2022	12 Nov 2022		13 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022		20 Nov 2022

6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Dataset analysis and preprocessing of data (Fruits dataset)	1	Download the dataset and examine the dataset and preprocess the dataset given on fruits dataset	5	High	Ishwarya,Juhi Padmaja, Jayasree, Reshma
	Dataset analysis and preprocessing of data (Vegetable dataset)	2	Download the dataset and examine the dataset and preprocess the dataset given on vegetable dataset	5	High	Ishwarya, Juhi Padmaja, Jayasree, Reshma
Sprint-2	Model creation which can classify diseased fruit plants from	3	Create a model which can classify diseased fruit plants from given	7.5	High	Ishwarya,Juhi Padmaja, Jayasree

	given images.(Fruits dataset)		images.(Fruits dataset)			, Reshma
	Model creation which can classify diseased fruit plants from given images.(Vegetables dataset)	4	Create a model which can classify diseased fruit plants from given images.(Fruits dataset)	7.5	High	Ishwarya,Juhi Padmaja, Jayasree, Reshma

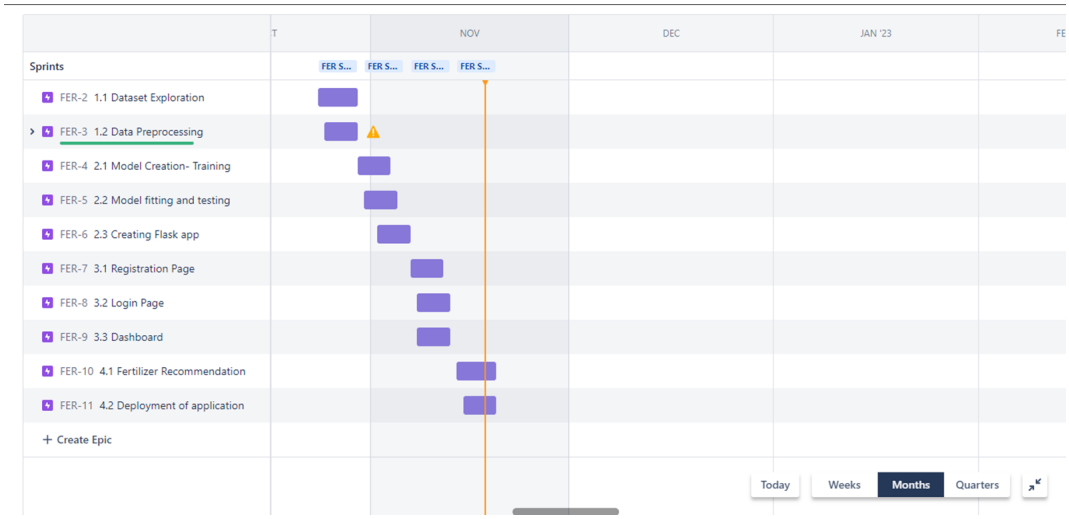
Sprint-3	Train Vegetable model on IBM Watson Studio	5	Login to IBM watson and download the ipynb file and upload and train it	2.5	High	Ishwarya,Juhi Padmaja, Jayasree, Reshma
	Train Vegetable model on IBM	6	Login to IBM watson and download the ipynb file and upload and train it	2.5	High	Ishwarya,Juhi Padmaja,

	Watson Studio					Jayasree, Reshma
	Upload a photo and check if the predicted value is true	7	As a user, I can access the application and upload the images of crops and get my fertilizer recommended	5	High	Ishwarya, Juhi Padmaja, Jayasree, Reshma
	Flask app creation	8	A base Flask web app must be created as an interface for the ML model	5	High	Ishwarya,Juhi Padmaja, Jayasree, Reshma
Sprint-4	Recommended Fertilizer	9	As per the disease predicted the application should display the recommended fertilizer	5	High	Ishwarya,Juhi Padmaja, Jayasree, Reshma

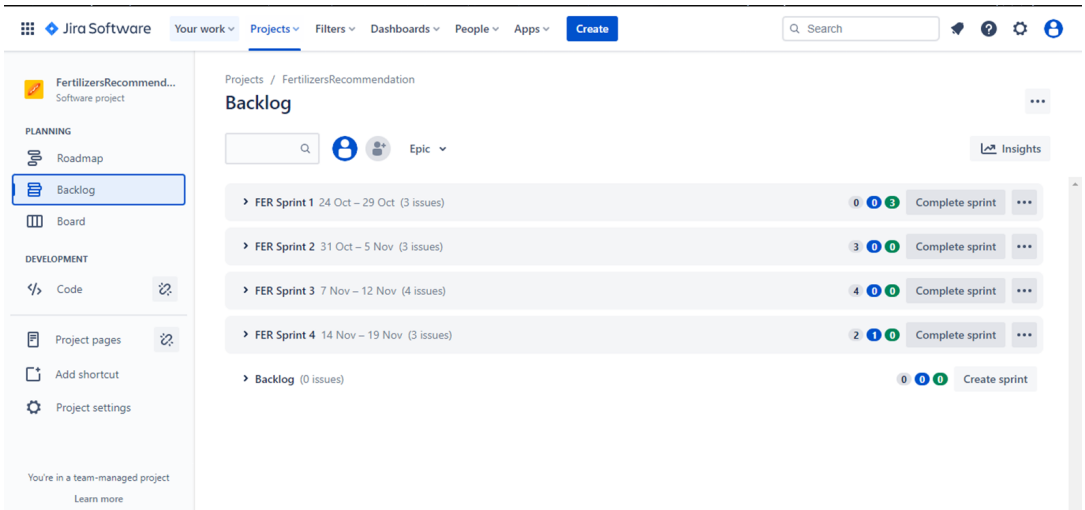
	Containerization / Hosting of application	10	Containerize the application and create a docker image or host the application on IBM Cloud	5	Medium	Ishwarya, Juhi Padmaja, Jayasree, Reshma

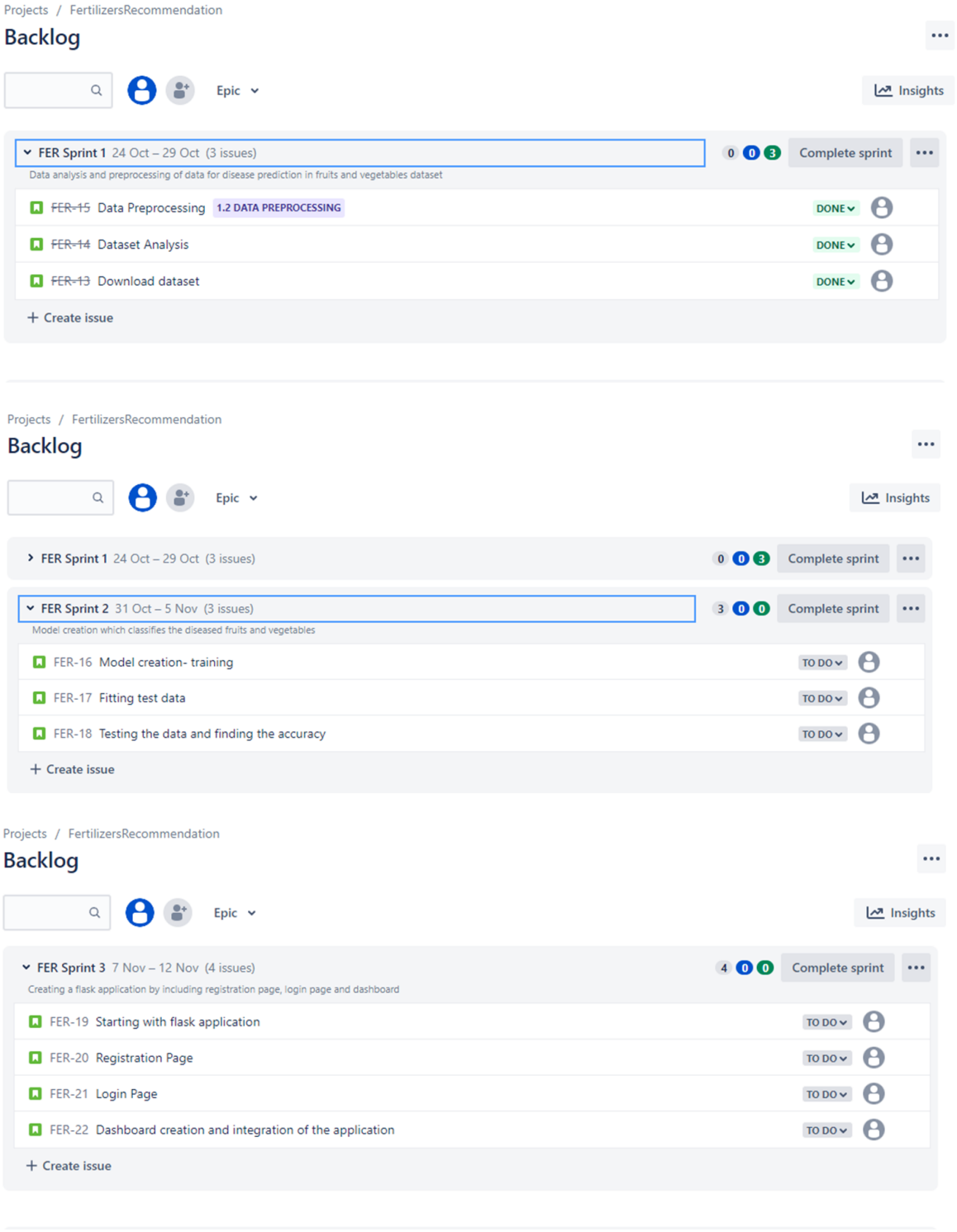
6.3 Reports from JIRA

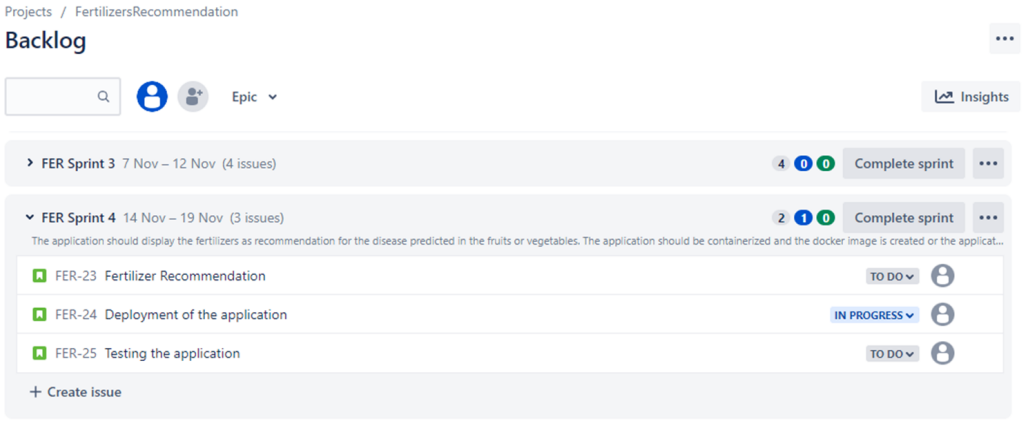
Roadmap (Sprints)

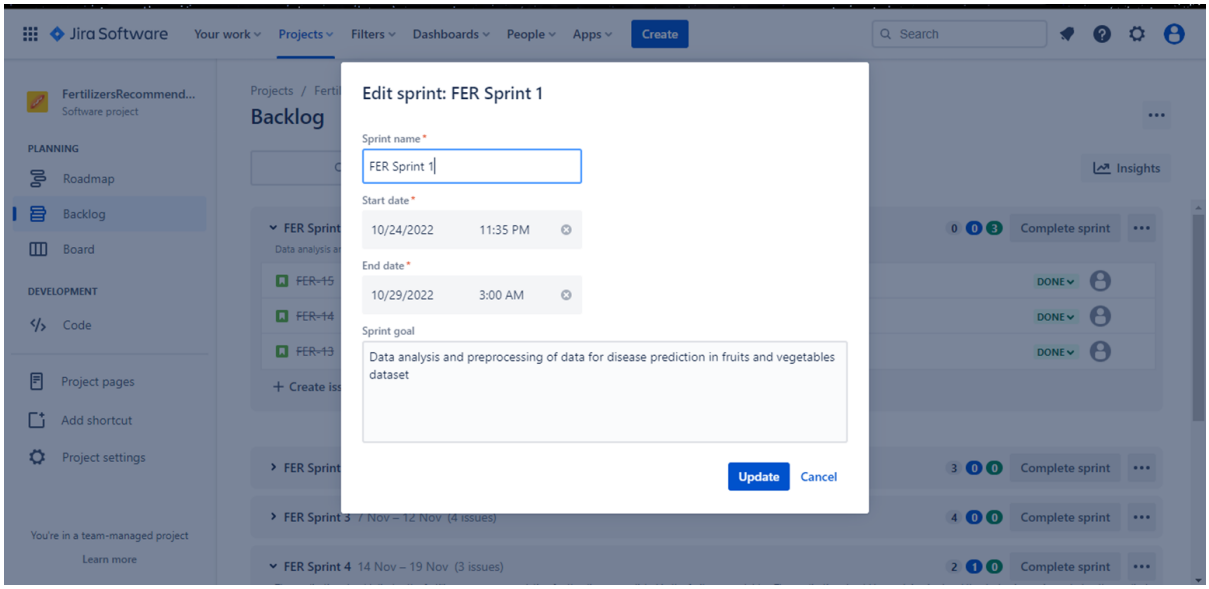


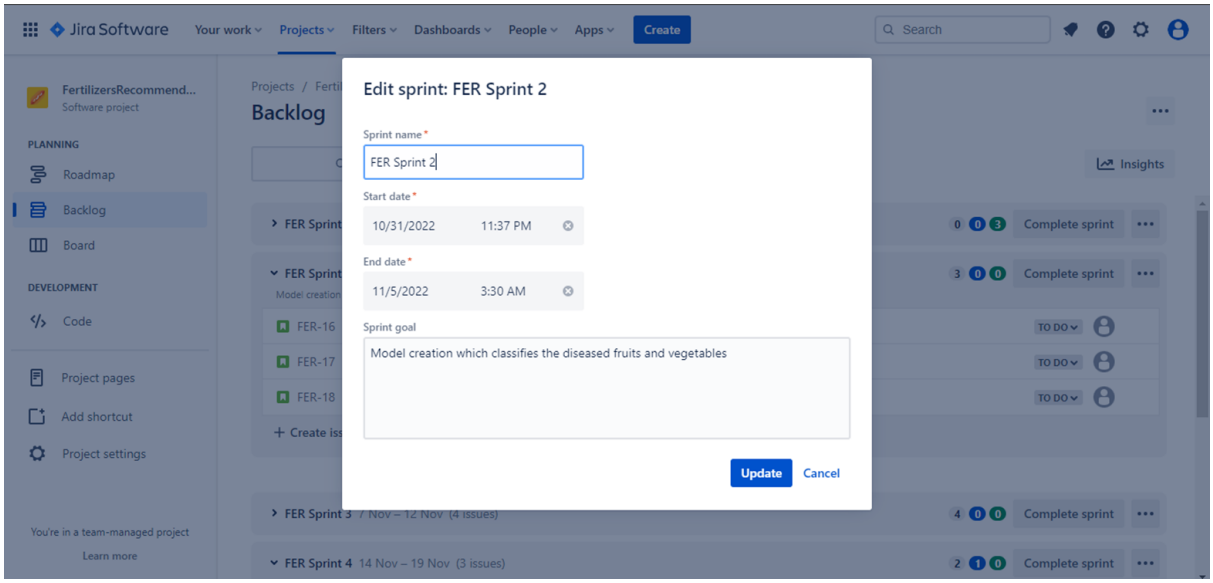
Backlog

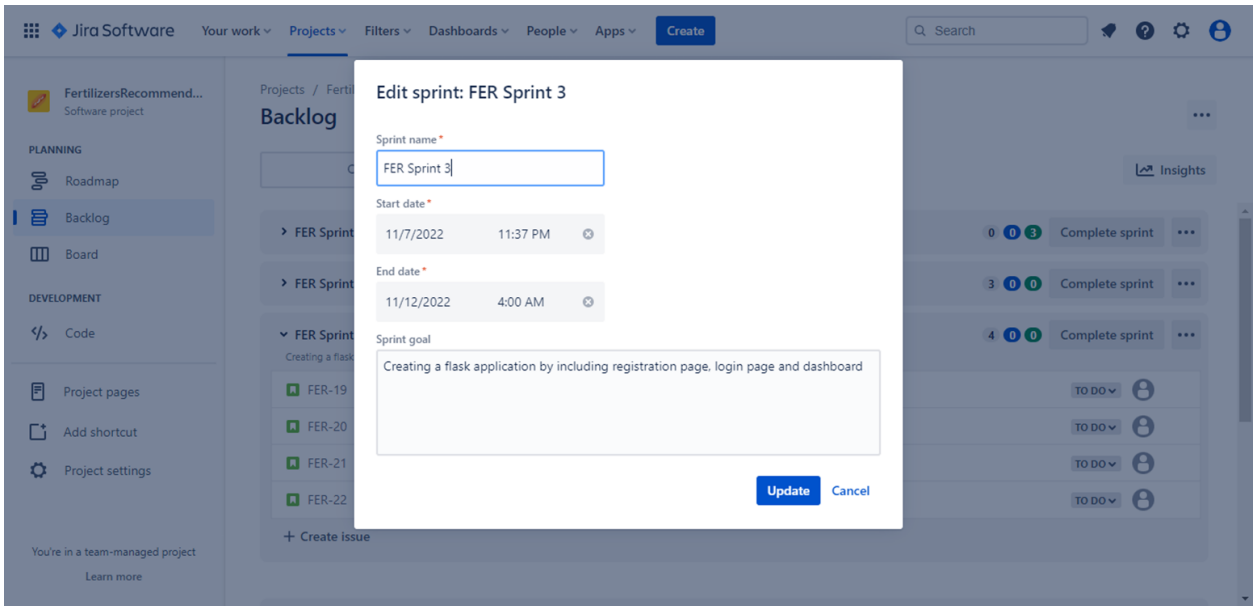


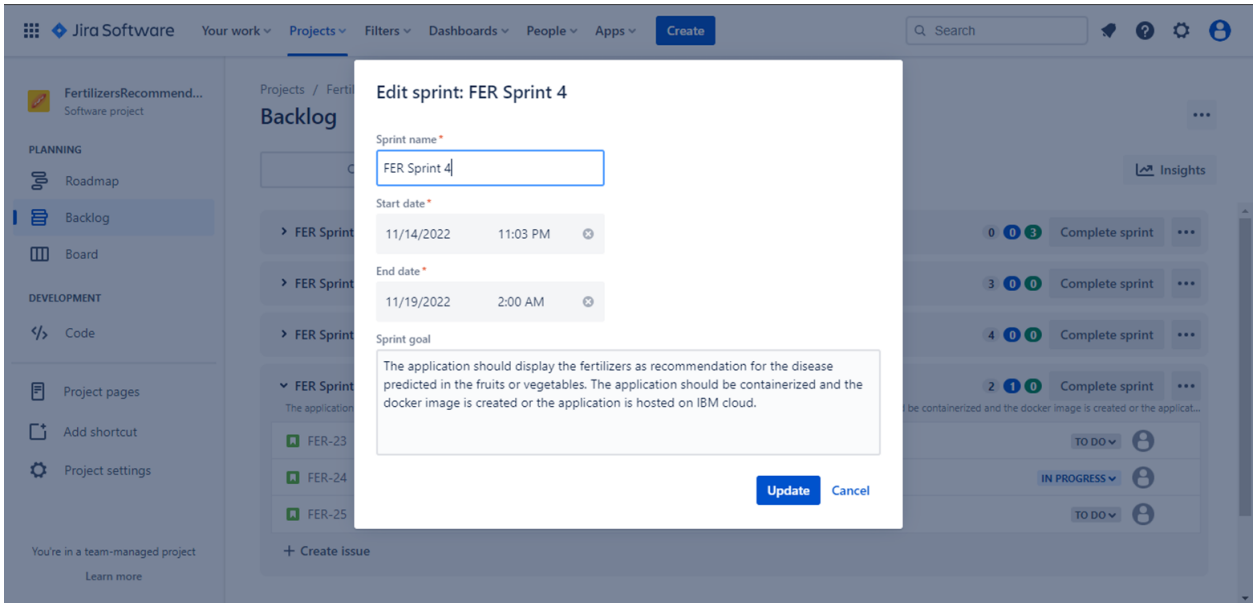

Projects / FertilizersRecommendation
Backlog
Epic
Insights
FER Sprint 1 24 Oct – 29 Oct (3 issues)
0 0 3
Complete sprint
Data analysis and preprocessing of data for disease prediction in fruits and vegetables dataset
FER-15 Data Preprocessing
1.2 DATA PREPROCESSING
DONE
FER-14 Dataset Analysis
DONE
FER-13 Download dataset
DONE
Create issue
Projects / FertilizersRecommendation
Backlog
Epic
Insights
FER Sprint 1 24 Oct – 29 Oct (3 issues)
0 0 3
Complete sprint
FER Sprint 2 31 Oct – 5 Nov (3 issues)
3 0 0
Complete sprint
Model creation which classifies the diseased fruits and vegetables
FER-16 Model creation- training
TO DO
FER-17 Fitting test data
TO DO
FER-18 Testing the data and finding the accuracy
TO DO
Create issue
Projects / FertilizersRecommendation
Backlog
Epic
Insights
FER Sprint 3 7 Nov – 12 Nov (4 issues)
4 0 0
Complete sprint
Creating a flask application by including registration page, login page and dashboard
FER-19 Starting with flask application
TO DO
FER-20 Registration Page
TO DO
FER-21 Login Page
TO DO
FER-22 Dashboard creation and integration of the application
TO DO
Create issue


Projects / FertilizersRecommendation
Backlog
Epic
Insights
FER Sprint 3 7 Nov – 12 Nov (4 issues)
Complete sprint
FER Sprint 4 14 Nov – 19 Nov (3 issues)
Complete sprint
The application should display the fertilizers as recommendation for the disease predicted in the fruits or vegetables. The application should be containerized and the docker image is created or the applicat...
FER-23 Fertilizer Recommendation
TO DO
FER-24 Deployment of the application
IN PROGRESS
FER-25 Testing the application
TO DO
Create issue

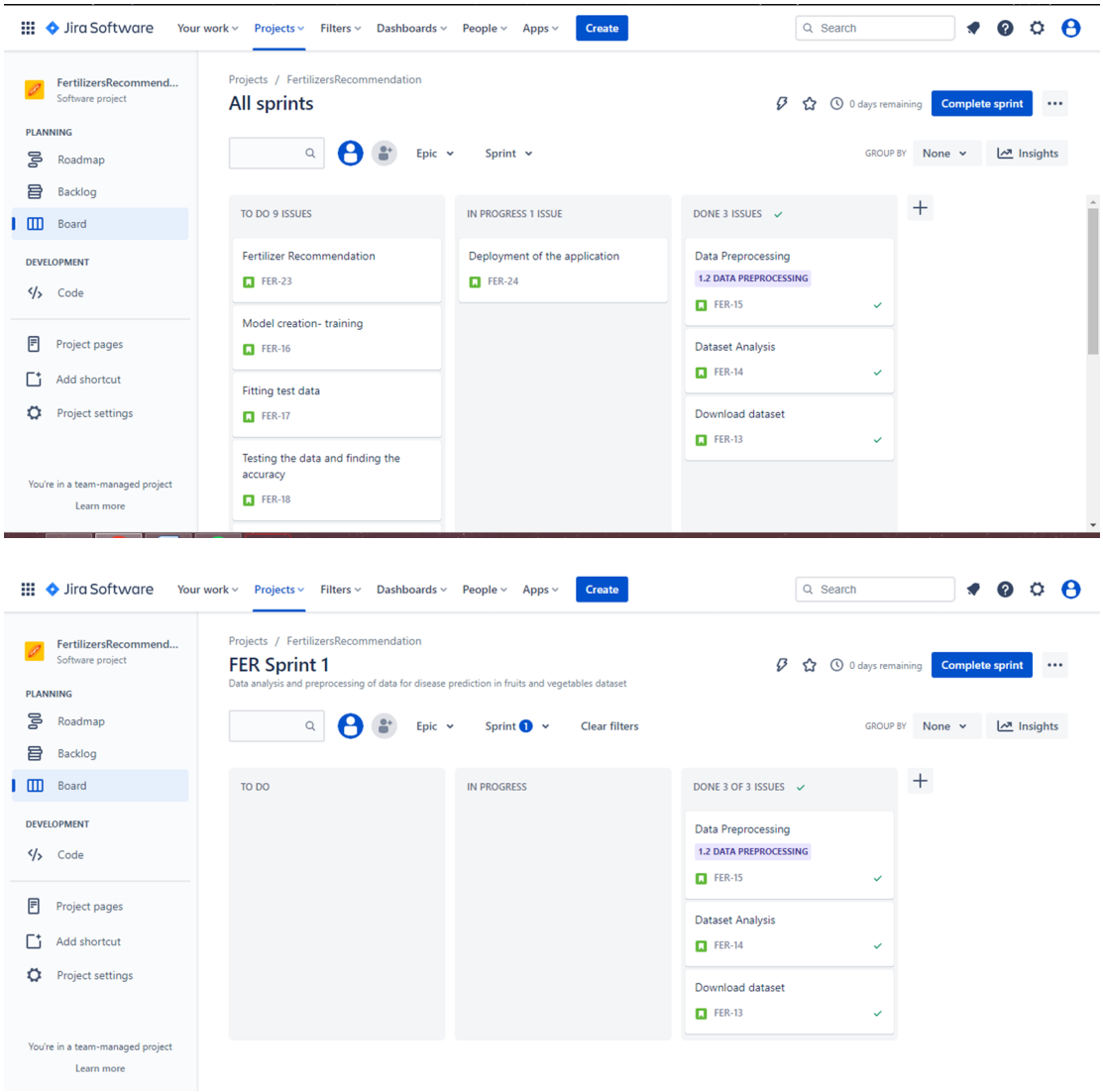

Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
FertilizersRecommend...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Backlog
Insights
Edit sprint: FER Sprint 1
Sprint name *
FER Sprint 1
Start date *
10/24/2022
11:35 PM
End date *
10/29/2022
3:00 AM
Sprint goal
Data analysis and preprocessing of data for disease prediction in fruits and vegetables dataset
Update
Cancel
FER Sprint 4 14 Nov – 19 Nov (3 issues)
Complete sprint


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
FertilizersRecommend...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Backlog
Insights
Edit sprint: FER Sprint 2
Sprint name *
FER Sprint 2
Start date *
10/31/2022
11:37 PM
End date *
11/5/2022
3:30 AM
Sprint goal
Model creation which classifies the diseased fruits and vegetables
Update
Cancel
FER Sprint 4 14 Nov – 19 Nov (3 issues)
Complete sprint


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
FertilizersRecommend...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Ferti
Backlog
Insights
Edit sprint: FER Sprint 3
Sprint name *
FER Sprint 3
Start date *
11/7/2022
11:37 PM
End date *
11/12/2022
4:00 AM
Sprint goal
Creating a flask application by including registration page, login page and dashboard
Update
Cancel
FER-19
FER-20
FER-21
FER-22
Create issue
Complete sprint
TO DO


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
FertilizersRecommend...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Ferti
Backlog
Insights
Edit sprint: FER Sprint 4
Sprint name *
FER Sprint 4
Start date *
11/14/2022
11:03 PM
End date *
11/19/2022
2:00 AM
Sprint goal
The application should display the fertilizers as recommendation for the disease predicted in the fruits or vegetables. The application should be containerized and the docker image is created or the application is hosted on IBM cloud.
Update
Cancel
FER-23
FER-24
FER-25
Create issue
Complete sprint
TO DO
IN PROGRESS

Board



7. CODING & SOLUTIONING

7.1 Feature 1: *Uploading the Leaf Image*

Using this feature a picture of a leaf to check if it is diseased or not can be uploaded.

```
<form action="/predict" method="post"  enctype="multipart/form-data" onsubmit="showloading()" name='plant'>
    <label for="cars">Choose plant type :</label>
      <select name="plant" id="plant">
        <option value="Fruit">Fruit</option>
        <option value="Vegetable">Vegetable</option>

      </select>
    <br><br><br>
    <input type="file" name="image" id='image' class="upload">
    <br>
    <input type="submit" value="Predict">
```

7.2 Feature 2: *Disease Prediction*

Using this feature we can detect whether the uploaded leaf is affected by any disease or not.

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

7.3 Feature 3: *Fertilizer Recommendation*

Using the feature we can recommend the appropriate fertilizers that have to be used to get rid of the pest.

```
if(plant=='Vegetable'):
    preds=model.predict(x)
    print(preds)
    df=pd.read_excel('precautions-veg.xlsx')
    print(df.iloc[preds[0]]['caution'])
else:
    preds=model1.predict(x)
    print("name=",preds[0])
    df=pd.read_excel('precautions-fruits.xlsx')
    #print(df.iloc[preds[0]]['caution'])
```

8. TESTING

8.1 Test Cases

Test Scenario	Component	Expected Result	Actual Result	Status
Verify if the user could view the Predict page	Predict page	Predict page contents should display	Working as expected	Pass
To verify if the user could choose the leaf type	Predict page	Type of leaf is successfully chosen	Working as expected	Pass
Verify if the user could upload image in the form	Predict page	Image is successfully uploaded	Working as expected	Pass

Navigation to the predicted page on submitting the form on predict page	Predict page	Form is successfully submitted and navigated to predicted page	Working as expected	Pass
Verify if the user could view the Predictedpage	Predicted page	Contents of predicted page is loaded successfully	Working as expected	Pass
Verify if the model predicts the disease in the uploaded leaf image	Predicted page	Predicted page is successfully predicted the disease in the leaf image	Working as expected	Pass

8.2 User Acceptance Testing

8.2.1 Defect Analysis

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Sub Total
By design	1	0	0	0	1
Duplicate	0	0	0	0	0
External	0	0	0	0	0
Fixed	2	1	0	0	3
Not Reproduced	0	0	0	0	0

Skipped	0	0	0	0	0
Won't fix	0	0	0	0	0
Total	3	1	0	0	4

8.2.2 Test Case Analysis

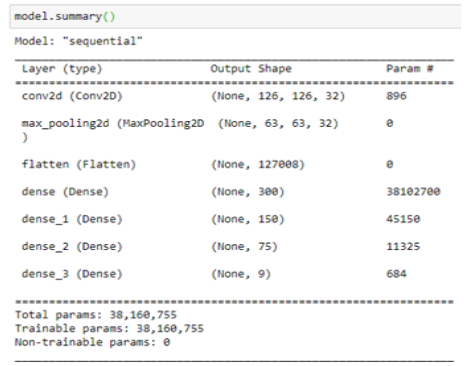
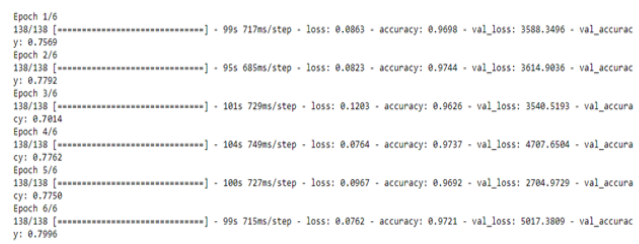
Page	**Total testcases**	**Pass**	**Fail**	**Not tested**
Home page	1	1	0	0
Prediction page	4	4	0	0
Predicted page	2	2	0	0

9. RESULTS

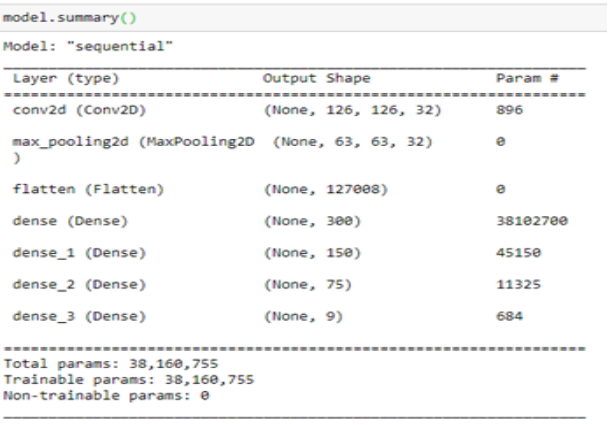
9.1 Performance Metrics

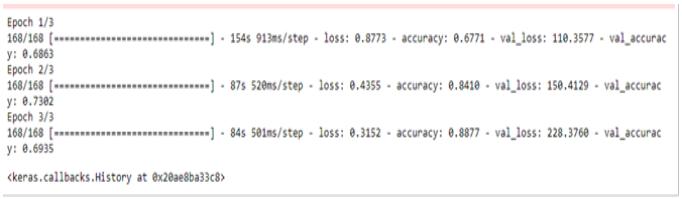
9.1.1 Model Performance Testing - Vegetable Model

S. No	**Parameter**	**Values**	**Screenshot**

1	Vegetable model summary	Total params: 38,160,755 Trainable params: 38,160,755 Non-trainable params: 0	
2	Accuracy	Training accuracy-0.9744 Validation accuracy-0.7996	

9.1.2 Model Performance Testing - Fruit Model

S. No	Parameter	Values	Screenshot
1	Fruit model summary	Total params: 5,082,202 Trainable params: 5,082,202 Non-trainable params: 0	

2	Accuracy	Training accuracy-0.8877 Validation accuracy-0.7302	

10. ADVANTAGES & DISADVANTAGES

Advantages:

These are all the benefits that will be achieved while implementing the project,

- Increases the crop yield.
- Helps to predict whether a plant is affected by pests at an early stage.
- Helps in recommending suitable pests that have to be used.
- Acts as a helpful guide for farmers.
- Possibility of financial loss to farmers will become less.

Disadvantages:

These are all the constraints that might be caused while implementing the project,

- Prediction might go wrong sometime, which leads to wrong decisions.
- Can't access in offline mode.
- Might be complex to handle for the stakeholders.
- Working with wrong data might cause any wrong prediction.
- Might consume time and resources.

11. CONCLUSION

In this project, various approaches and models of deep learning techniques were studied and used to correctly detect and classify plant diseases through image processing of plant leaves. The procedure begins by collecting images used for training, testing, and validation, goes through image preprocessing and enhancement, and compares the accuracy of different pre-trained models. Finally, our model recognizes and distinguishes between healthy plants and various diseases and provides appropriate treatments to cure the diseases. In this paper, we proposed and developed a system that uses plant leaf images to detect different types of diseases in tomato crops and also provides suitable fertilization suggestions.

12. FUTURE SCOPE

The system can also correctly interpret various diseases and provide fertilizer suggestions for specific diseases. Additionally, the system can be made more robust by integrating image datasets with greater variation, such as multiple leaves in a single image. You can also develop an app for this project. This makes the farmer's work easier. Upload your images directly into the app to instantly see diseases and treatments. This reduces the time and effort required. The project is currently limited to specific crop species, but more crop and flower datasets may be added in the future, making it useful for any agricultural need. You can also add new models and try them out over time. This improves accuracy and makes the model even faster.

13. APPENDIX

Source Code

App.py

```
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('vegetable.h5')
model1=load_model('fruit.h5')
#home page
@app.route('/', methods=['GET', 'POST'])
def home():
    return render_template('home.html')
```

```
@app.route('/prediction')
def prediction():
    return render_template('predict.html')
@app.route('/predict',methods=['POST','GET'])
def predict():
    if(request.method=='POST'):
            f=request.files['image']
            basepath=os.path.dirname(__file__)
            file_path=os.path.join(basepath,'',secure_filename(f.filename))
            #print(file_path)
            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=model.predict(x)
                    print(preds)
                    df=pd.read_excel('precautions-veg.xlsx')
                    print(df.iloc[preds[0]]['caution'])
            else:
                    preds=model1.predict(x)
                    print("name=",preds[0])
                    df=pd.read_excel('precautions-fruits.xlsx')
                    print(df.iloc[preds[0]]['caution'])
            return
    render_template("predicted.html",disease=str(df.iloc[np.where(preds[0]==1)[0][0]]['disea
    se_name']),data=str(df.iloc[np.where(preds[0]==1)[0][0]]['caution']))

if(__name__=="__main__"):
```

```
    app.run(debug=True)
```

Home.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Plant disease prediction and Fertilizer recommendation</title>
<meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
   <!-- Bootstrap CSS -->
                                                    <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg
    6uqKI2xXr2" crossorigin="anonymous">
   <link href="style.css" rel="stylesheet" type="text/css" />
<style>
body {
   font-family: avenirnextltpro-regular;
   color: #4a4a4a;
   font-weight: 400;
    background-image: linear-gradient(to right, #B2D2A4 0%, #B2D2A4  51%, #B2D2A4
    100%);
   margin: 0;
 font-family: Arial, Helvetica, sans-serif;
}
.topnav {
 overflow: hidden;
 background-color: black;
}
.topnav a {
```

```
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}
.topnav a:hover {
  background-color: silver;
  color: black;
}

.topnav a.active {
  background-color: black;
  color: white;
}
.topnav-right {
  float: right;
}
img {
  border-radius: 50%;
}
.margin-text{
  margin: 70px;
  text-align: justify;
}
</style>

</head>
<body>
<div class="topnav">
```

```
    <a style = "font-size:25px;font-family: 'Trebuchet MS', sans-serif;" class="active"
    href=#>Plant disease prediction</a>
  <div class="topnav-right">
    <a style = "font-size:20px;font-family: 'Trebuchet MS', sans-serif;" class="active"
    href=/>Home</a>
        <a style = "font-size:20px;font-family: 'Trebuchet MS', sans-serif;"
    href=prediction>Prediction</a>
  </div>
</div>
<br><br>


<div align=center>
<table style="width:90%; ">
<tr style="width:60%;font-family: 'Trebuchet MS', sans-serif; ">
<td style="width:65%;">
<center><b><h1 style = "font-size:60px; color: black;">Detect if your plant is
    infected!!</h1></b></center>
<h5 style="font-family: 'Trebuchet MS', sans-serif; text-align: justify;line-height: 1.6; color:
    black;">
Agriculture is one of the major sectors world 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
    laboratories. This application helps farmers in detecting the diseases by observing the
    spots on the leaves, which inturn saves effort and labour costs.</h5>
</td>
<td style="width:60%;">
<center>
<img
    src="https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcTyo6fM_V5XTdWNIN2
```

```
    ebEglHRFzm-fhqeoqSUPDEMEG3gjvavsg" alt="crop_img"   width="450" height="400"
    >
</center>
</td>
</tr>
</table>
</div>


<div class="margin-text">
</div>
</body>
</html>
```

Predict.html

```
<html>
<head>
<style>
.topnav {
  overflow: hidden;
  background-color: black;
}
.topnav a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}
.topnav a:hover {
```

```
  background-color: silver;
  color: black;
}
.topnav a.active {
  background-color: black;
  color: white;
}
.topnav-right {
  float: right;
}

.margin-text{
  margin: 70px;
  text-align: justify;
}
* {
            margin: 0px;
            padding: 0px;
            box-sizing: border-box;
        }
        form {
            display: block;;
            height: 85vh;
            justify-content: left;
            align-items: left;
            margin-top: 120px;
            width: 60%;
            text-align: left;
      }
 .details h2 {
            position: relative;
```

```
        top: 100px;
        margin: auto;
        color: rgb(18, 231, 231);
        font-size: 3rem;
    }
 .details h2 {
        /* margin-bottom: 300px; */
        position: relative;
        top: 100px;
        margin: auto;
        color: rgb(18, 231, 231);
        font-size: 3rem;

    }
.details h1 {
        color: white;
        padding: 20px;
        border-radius: 15px;
        background-color: rgb(8, 8, 8);
    }.upload {
  font-size: 15px;
        background-color: rgb(255, 252, 252);
        border-radius: 20px;
        outline: none;
        width: 500px;
        color: rgb(0, 0, 0);
        border: 3px solid rgb(45, 47, 49);
 }
    ::-webkit-file-upload-button {
        color: rgb(255, 252, 252);
        padding: 20px;
```

```
        border: 2px solid rgb(201, 6, 6);
        background-color: rgb(201, 6, 6);
        border-radius: 15px;
    }
    ::-webkit-file-upload-button:hover {
        border-radius: 20px;
        border: 2px solid rgb(177, 174, 174);
}



    input[type="submit"] {
        margin-top: 80px;
        margin-right: 30px;
        padding: 15px 35px;
        border-radius: 15px;
        color: black;
        font-size: 1.5rem;
            }
</style>
</head>
<body background="{{url_for('static',filename = 'images/Background.jpg')}}">
<div class="topnav">
    <a class="active" style = "font-size:25px;font-family: 'Trebuchet MS', sans-serif;"
    href=#>Plant disease prediction</a>
  <div class="topnav-right">
    <a class="active" style = "font-size:20px;font-family: 'Trebuchet MS', sans-serif;"
    href=/>Home</a>
        <a style = "font-size:20px;font-family: 'Trebuchet MS', sans-serif;"
    href=prediction>Prediction</a>
  </div>
```

```
</div>
<center>
    <table style="width:80%;">
    <tr style="width:60%;  border:">
    <td style="width:50%; ">
    <center>
    <img src="\static\crop.png" width="450" height="300"></center>
    </td>
    <td style="width:80%;  ">
    <form action="/predict" method="post" enctype="multipart/form-data" onsubmit="showloading()" name='plant'>
                                    <label for="cars" style="font-family: 'Trebuchet MS', sans-serif;font-size:20px;">Choose plant type to get the prediction</label><br><br>
                        <select name="plant" style="border-radius:8px;width:400px;height:30px; background-color: WhiteSmoke;font-family: 'Trebuchet MS', sans-serif;font-size:1vw;" id="plant">
              <option value="Fruit">Fruit</option>
              <option value="Vegetable">Vegetable</option>

            </select>
          <br><br><br><br>
                            <label for="cars" style="font-family: 'Trebuchet MS', sans-serif;font-size:20px;">Drop in the image to get the prediction:</label><br>
          <br>
                            <input type="file" name="image" id='image' class="upload">

                            <br>
              <input type="submit" style = "background-color:black;color:white;font-family: 'Trebuchet MS', sans-serif;" value="Predict   ">
        </form>
    </td>
```

```
    </tr>
    </table></center>
  </section>
</body>
</html>
```

Predicted.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Plant disease prediction and Fertilizer recommendation</title>
<meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <!-- Bootstrap CSS -->
                                                    <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg
    6uqKI2xXr2" crossorigin="anonymous">
  <link href="style.css" rel="stylesheet" type="text/css" />
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<style>body {
  font-family: avenirnextltpro-regular;
  color: #4a4a4a;
  font-weight: 400;
    background-image: linear-gradient(to right, #B2D2A4 0%, #B2D2A4  51%, #B2D2A4
    100%);
  margin: 0;
 font-family: Arial, Helvetica, sans-serif;
}
```

```
.topnav {
  overflow: hidden;
  background-color: black;
}.topnav a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

.topnav a:hover {
  background-color: silver;
  color: black;
}.topnav a.active {
  background-color: black;
  color: white;
}
.topnav-right {
  float: right;
}
img {
  border-radius: 100%;
}
.margin-text{
  margin: 70px;
  text-align: justify;
}
</style></head>
<body>
```

```
<div class="topnav">
    <a style = "font-size:25px;font-family: 'Trebuchet MS', sans-serif;" class="active"
    href=#>Plant disease prediction</a>
  <div class="topnav-right">
    <a style = "font-size:20px;font-family: 'Trebuchet MS', sans-serif;" class="active"
    href=/>Home</a>
        <a style = "font-size:20px;font-family: 'Trebuchet MS', sans-serif;"
    href=prediction>Prediction</a>
  </div>
</div>
<br><br>


<div align=center>
<table style="width:90%; ">
<tr style="width:60%;font-family: 'Trebuchet MS', sans-serif; ">
<td style="width:65%;">
<center><b><h1 style = "font-size:60px; color: black;">Detected your
    disease!!</h1></b></center>
<h5 style="font-family: 'Trebuchet MS', sans-serif; text-align: justify;line-height: 1.6; color:
    black;">
  <center><b><h5 style = "font-size:60px; color: black;">
    <div class="w3-container">
    <div class="w3-panel w3-round-xlarge w3-teal">
      <b><h3>{{disease}}</h3></b>
          <h3><p> 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 laboratories. This application helps
    farmers in detecting the diseases by observing the spots on the leaves, which inturn saves
    effort and labour costs.</p></h3>
```

```
    </div>
   </div>
  </h5></b></center>
</td>
</tr>
<tr>
  <td style="width:65%;">
            <center><b><h1 style = "font-size:60px; color: black;">Symptoms & Precautions</h1></b></center>
     <h5 style="font-family: 'Trebuchet MS', sans-serif; text-align: justify;line-height: 1.6; color: black;">
     <center><b><h5 style = "font-size:60px; color: black;">
      <div class="w3-container">
       <div class="w3-panel w3-round-xlarge w3-teal">
          <h3><p>{{data}}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 laboratories. This application helps farmers in detecting the diseases by observing the spots on the leaves, which inturn saves effort and labour costs.</p></h3>
       </div>
      </div>
      </h5></b></center>
   </td>
</tr>
</table>
</div><div class="margin-text">
</div>
</body>
</html>
```

Fruit_training.ipynb

```
import os
import cv2
import numpy as np
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator (rescale = 1./255, shear_range= 0.2,zoom_range= 0.2,
    horizontal_flip = True)
test_datagen =ImageDataGenerator (rescale = 1)
x_train = train_datagen.flow_from_directory(r'D:\IBM Project\Dataset Plant
    Disease\fruit-dataset\fruit-dataset\train',target_size = (128,128), batch_size = 32,
    class_mode = 'categorical')
x_test = test_datagen.flow_from_directory(r"D:\IBM Project\Dataset Plant
    Disease\fruit-dataset\fruit-dataset\test",target_size = (128,128), batch_size = 32,
    class_mode = 'categorical')
from keras.models  import Sequential
from keras. layers import Dense
from keras. layers import Convolution2D
from keras. layers import MaxPooling2D
from keras. layers import Flatten
model = Sequential()
model.add(Convolution2D(32,(3,3),input_shape = (128,128,3),activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(units = 40 ,kernel_initializer = 'uniform',activation = 'relu'))
model.add(Dense(units = 20 ,kernel_initializer = 'random_uniform',activation = 'relu'))
model.add(Dense(units = 6,activation = 'softmax',kernel_initializer ='random_uniform'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit_generator(x_train,steps_per_epoch=168,epochs=3,validation_data=x_test,validatio
    n_steps=52)
model.summary()
```

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

Vegetable_training.ipynb

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator (rescale = 1./255, shear_range= 0.2,zoom_range= 0.2,
    horizontal_flip = True)
test_datagen =ImageDataGenerator (rescale = 1)
x_train = train_datagen.flow_from_directory(r'D:\IBM Project\Dataset Plant
    Disease\Veg-dataset\Veg-dataset\train_set',target_size = (128,128), batch_size = 32,
    class_mode = 'categorical')
x_test = test_datagen.flow_from_directory(r"D:\IBM Project\Dataset Plant
    Disease\Veg-dataset\Veg-dataset\test_set",target_size = (128,128), batch_size = 32,
    class_mode = 'categorical')
from keras.models  import Sequential
from keras. layers import Dense
from keras. layers import Convolution2D
from keras. layers import MaxPooling2D
from keras. layers import Flatten
model = Sequential()
model.add(Convolution2D(32,(3,3),input_shape = (128,128,3),activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(units = 300 ,kernel_initializer = 'uniform',activation = 'relu'))
model.add(Dense(units = 150 ,kernel_initializer = 'uniform',activation = 'relu'))
model.add(Dense(units = 75 ,kernel_initializer = 'uniform',activation = 'relu'))
model.add(Dense(units=9,activation = 'softmax',kernel_initializer='uniform'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit_generator(x_train,steps_per_epoch=77,epochs=18,validation_data=x_test,validatio
    n_steps=52)
model.summary()
model.save('vegetable.h5')
```

IBM_Fruit_Training.ipynb

```
!pip install watson-machine-learning-client
from ibm_watson_machine_learning import APIClient
wml_credentials={
                "url":"https://us-south.ml.cloud.ibm.com",
                "apikey":"QXdS9eYZqw3SK9Cvz9QV34qVW2O-V7DEBJ9Hpe1B39gO"
            }
client=APIClient(wml_credentials)
client=APIClient(wml_credentials)
def guid_from_space_name(client,space_name):
   space=client.spaces.get_details()
                        return(next(item for item in space['resources'] if
    item['entity']['name']==space_name)['metadata']['id'])
space_uid=guid_from_space_name(client,'fruit_tarining')
print("Space UID = "+space_uid)
client.set.default_space(space_uid)
client.repository.download('c9ec4900-a382-4fab-addd-b534376fc98d',"my_model.tar.gz")
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np
model=load_model('fruit1.h5')
img=image.load_img('apple1.jpg',target_size=(128,128))
img
import numpy as np
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
x.shape
pred=model.predict(x)
pred[0]
```

```
disease=["Apple Black Rot","Apple Healthy","Corn Healthy","Corn Northern Leaf
    Blight","Peach Bacterial Spot","Peach Healthy"]
#print(disease[pred[0]])
disease[np.where(pred[0]==1)[0][0]]
```

IBM_Vegetable_Training.ipynb

```
!pip install watson-machine-learning-client
from ibm_watson_machine_learning import APIClient
wml_credentials={
                "url":"https://us-south.ml.cloud.ibm.com",
                "apikey":"QXdS9eYZqw3SK9Cvz9QV34qVW2O-V7DEBJ9Hpe1B39gO"
            }
client=APIClient(wml_credentials)
client=APIClient(wml_credentials)
def guid_from_space_name(client,space_name):
   space=client.spaces.get_details()
                        return(next(item for item in space['resources'] if
    item['entity']['name']==space_name)['metadata']['id'])
space_uid=guid_from_space_name(client,'fruit_tarining')
print("Space UID = "+space_uid)
client.set.default_space(space_uid)
client.repository.download('0b361923-1c54-4719-821f-5702e0dd668d',"my_model.tar.gz")
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np
model=load_model('vegetable1.h5')
img=image.load_img('tomato.jpg',target_size=(128,128))
img
import numpy as np
x=image.img_to_array(img)
```

```
x=np.expand_dims(x,axis=0)
x.shape
pred=model.predict(x)
pred[0]
disease=["Pepper bell bacterial spot","Pepper bell healthy","Potato early blight","Potato healthy","Potato late blight","Tomato Bacterial Spot","Tomato late blight","Tomato leaf model","Tomato septoria leaf spot"]
#print(disease[pred[0]])
disease[np.where(pred[0]==1)[0][0]]
```

Plant_disease_testing.ipynb

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np
model=load_model('fruit.h5')
img=image.load_img(r'D:\IBM Project\Dataset Plant Disease\fruit-dataset\fruit-dataset\test\Apple___healthy\00fca0da-2db3-481b-b98a-9b67bb7b105c___RS_HL 7708.JPG',target_size=(128,128))
img
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=model.predict(x)
pred
y=np.argmax(model.predict(x),axis=1)
index=['Apple___Black_rot', 'Apple___healthy', 'Corn_(maize)___healthy', 'Corn_(maize)___Northern_Leaf_Blight', 'Peach___Bacterial_spot', 'Peach___healthy']
index[y[0]]
```

Fruit_Preprocessing.ipynb

```
import matplotlib.pyplot as plt
```

```
import numpy as np
import pandas as pd
import os
import PIL
import random
import cv2
from tensorflow.keras import layers, models
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
import pickle
import zipfile
tf.__version__
TRAIN_PATH="D:\IBM Project\Dataset Plant Disease\fruit-dataset\fruit-dataset\train"
FRUIT_TRAIN_CLASS_NAMES=np.array(os.listdir(r"D:\IBM Project\Dataset Plant
    Disease\fruit-dataset\fruit-dataset\train"))
print("FRUIT_TRAIN_CLASS_NAMES")
FRUIT_TRAIN_CLASS_NAMES
TEST_PATH="D:\IBM Project\Dataset Plant Disease\fruit-dataset\fruit-dataset\test"
FRUIT_TEST_CLASS_NAMES=np.array(os.listdir(r"D:\IBM Project\Dataset Plant
    Disease\fruit-dataset\fruit-dataset\test"))
print("FRUIT_TEST_CLASS_NAMES")
FRUIT_TEST_CLASS_NAMES
files_train_count = []
for i,f in enumerate(FRUIT_TRAIN_CLASS_NAMES):
  #print(i,f)
                folder_path = os.path.join(r"D:\IBM Project\Dataset Plant
    Disease\fruit-dataset\fruit-dataset\train", str(f))
  for path in os.listdir(os.path.join(folder_path)):
    files_train_count.append(['{}/{}'.format(folder_path,path), f, i])
```

```
fruits_train_df = pd.DataFrame(files_train_count, columns=['filepath', 'class_name', 'label'])
print("FRUIT TRAIN")
fruits_train_df.head()
files_test_count = []
for i,f in enumerate(FRUIT_TEST_CLASS_NAMES):
   #print(i,f)
                    folder_path = os.path.join(r"D:\IBM Project\Dataset Plant
    Disease\fruit-dataset\fruit-dataset\test", str(f))
   for path in os.listdir(os.path.join(folder_path)):
      files_test_count.append(['{}/{}'.format(folder_path,path), f, i])
fruits_test_df = pd.DataFrame(files_test_count, columns=['filepath', 'class_name', 'label'])
print("FRUIT TEST")
fruits_test_df.head()
#fruits train class count
fruits_train_df.class_name.value_counts()
#fruits test class count
fruits_test_df.class_name.value_counts()
#fruits train
quantidade_por_class = 310
fruits_train_df = pd.concat([fruits_train_df[fruits_train_df['class_name']==
    i][:quantidade_por_class] for i in FRUIT_TRAIN_CLASS_NAMES])
#fruits test
quantidade_por_class = 310
fruits_test_df = pd.concat([fruits_test_df[fruits_test_df['class_name']==
    i][:quantidade_por_class] for i in FRUIT_TEST_CLASS_NAMES])
#fruits train
fruits_train_df.class_name.value_counts()
#fruits test
fruits_test_df.class_name.value_counts()
#fruits train
print("FRUIT TRAIN IMAGE")
```

```
train_image = cv2.imread(fruits_train_df.filepath[100])
imgplot = plt.imshow(train_image)
plt.show()
#fruits test
print("FRUIT TEST IMAGE")
test_image = cv2.imread(fruits_test_df.filepath[104])
imgplot = plt.imshow(test_image)
plt.show()
x_train = fruits_train_df['filepath']
y_train = fruits_train_df['label']
x_train_tensor = tf.convert_to_tensor(x_train.values, dtype=tf.string)
y_train_tensor = tf.convert_to_tensor(y_train.values)
x_test = fruits_test_df['filepath']
y_test = fruits_test_df['label']
x_test_tensor = tf.convert_to_tensor(x_test.values, dtype=tf.string)
y_test_tensor = tf.convert_to_tensor(y_test.values)
train_data = tf.data.Dataset.from_tensor_slices((x_train_tensor, y_train_tensor))
test_data = tf.data.Dataset.from_tensor_slices((x_test_tensor, y_test_tensor))
def map_fn(path, label):
   image = tf.image.decode_jpeg(tf.io.read_file(path))
   return image, label
#apply the function
train_data_img = train_data.map(map_fn)
test_data_img = test_data.map(map_fn)
fig, ax = plt.subplots(1,2, figsize = (15,5))
for i,l in train_data_img.take(1):
   ax[0].set_title('Image from train dataset');
   ax[0].imshow(i);
for i,l in test_data_img.take(1):
   ax[1].set_title('Image from test dataset');
   ax[1].imshow(i);
```

```
hist = cv2.calcHist([train_image],[0],None,[50],[0,256])

# different methods for displaying a histogram
plt.bar(range(50), hist.ravel())
plt.title('Histogram of the fruit train image')
plt.xlabel('Gray values')
plt.ylabel('Frequency')
hist = cv2.calcHist([test_image],[0],None,[50],[0,256])

# different methods for displaying a histogram
plt.bar(range(50), hist.ravel(), color='Red')
plt.title('Histogram of the fruit test image')
plt.xlabel('Gray values')
plt.ylabel('Frequency')
print("Train image size = ",train_image.shape)
print("Test image size = ",test_image.shape)
```

Vegetable_Preprocessing.ipynb

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import PIL
import random
import cv2
from tensorflow.keras import layers, models
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
import pickle
```

```
import zipfile
tf.__version__
TRAIN_PATH="D:\IBM Project\Dataset Plant Disease\veg-dataset\veg-dataset\train_set"
VEG_TRAIN_CLASS_NAMES=np.array(os.listdir(r"D:\IBM Project\Dataset Plant
    Disease\veg-dataset\veg-dataset\train_set"))
print("VEG_TRAIN_CLASS_NAMES")
VEG_TRAIN_CLASS_NAMES
TEST_PATH="D:\IBM Project\Dataset Plant Disease\veg-dataset\veg-dataset\test_set"
VEG_TEST_CLASS_NAMES=np.array(os.listdir(r"D:\IBM Project\Dataset Plant
    Disease\veg-dataset\veg-dataset\test_set"))
print("FRUIT_TEST_CLASS_NAMES")
VEG_TEST_CLASS_NAMES
files_train_count = []
for i,f in enumerate(VEG_TRAIN_CLASS_NAMES):
  #print(i,f)
                folder_path = os.path.join(r"D:\IBM Project\Dataset Plant
    Disease\veg-dataset\veg-dataset\train_set", str(f))
  for path in os.listdir(os.path.join(folder_path)):
    files_train_count.append(['{}/{}'.format(folder_path,path), f, i])
veg_train_df = pd.DataFrame(files_train_count, columns=['filepath', 'class_name', 'label'])
print("VEG TRAIN")
veg_train_df.head()
files_test_count = []
for i,f in enumerate(VEG_TEST_CLASS_NAMES):
  #print(i,f)
                folder_path = os.path.join(r"D:\IBM Project\Dataset Plant
    Disease\veg-dataset\veg-dataset\test_set", str(f))
  for path in os.listdir(os.path.join(folder_path)):
    files_test_count.append(['{}/{}'.format(folder_path,path), f, i])
veg_test_df = pd.DataFrame(files_test_count, columns=['filepath', 'class_name', 'label'])
print("VEG TEST")
```

```
veg_test_df.head()
#veg train class count
veg_train_df.class_name.value_counts()
#veg test class count
veg_test_df.class_name.value_counts()
#veg train
quantidade_por_class = 310
veg_train_df = pd.concat([veg_train_df[veg_train_df['class_name']==
    i][:quantidade_por_class] for i in VEG_TRAIN_CLASS_NAMES])
#veg test
quantidade_por_class = 310
veg_test_df = pd.concat([veg_test_df[veg_test_df['class_name']== i][:quantidade_por_class]
    for i in VEG_TEST_CLASS_NAMES])
#veg train
veg_train_df.class_name.value_counts()
#veg test
veg_test_df.class_name.value_counts()
#veg train
print("VEG TRAIN IMAGE")
train_image = cv2.imread(veg_train_df.filepath[40])
imgplot = plt.imshow(train_image)
plt.show()
#veg test
print("VEG TEST IMAGE")
test_image = cv2.imread(veg_test_df.filepath[40])
imgplot = plt.imshow(test_image)
plt.show()
x_train = veg_train_df['filepath']
y_train = veg_train_df['label']
x_train_tensor = tf.convert_to_tensor(x_train.values, dtype=tf.string)
y_train_tensor = tf.convert_to_tensor(y_train.values)
```

[{"metadata":{"trusted":true},"cell_type":"code","source":"x_test =
veg_test_df['filepath']\ny_test = veg_test_df['label']","execution_count":18,"outputs":[]}]

```
x_test = veg_test_df['filepath']
y_test = veg_test_df['label']
x_test_tensor = tf.convert_to_tensor(x_test.values, dtype=tf.string)
y_test_tensor = tf.convert_to_tensor(y_test.values)
train_data = tf.data.Dataset.from_tensor_slices((x_train_tensor, y_train_tensor))
test_data = tf.data.Dataset.from_tensor_slices((x_test_tensor, y_test_tensor))
def map_fn(path, label):
    image = tf.image.decode_jpeg(tf.io.read_file(path))
    return image, label
#apply the function
train_data_img = train_data.map(map_fn)
test_data_img = test_data.map(map_fn)
fig, ax = plt.subplots(1,2, figsize = (15,5))
for i,l in train_data_img.take(1):
    ax[0].set_title('Image from train dataset');
    ax[0].imshow(i);
for i,l in test_data_img.take(1):
    ax[1].set_title('Image from test dataset');
    ax[1].imshow(i);
hist = cv2.calcHist([train_image],[0],None,[50],[0,256])

# different methods for displaying a histogram
plt.bar(range(50), hist.ravel())
plt.title('Histogram of the fruit train image')
plt.xlabel('Gray values')
plt.ylabel('Frequency')
hist = cv2.calcHist([test_image],[0],None,[50],[0,256])

# different methods for displaying a histogram
```

```
plt.bar(range(50), hist.ravel(), color='Red')
plt.title('Histogram of the fruit test image')
plt.xlabel('Gray values')
plt.ylabel('Frequency')
print("Train image size = ",train_image.shape)
print("Test image size = ",test_image.shape)
```

GitHub & Project Demo Link:

- **Github Link: https://github.com/IBM-EPBL/IBM-Project-34619-1660240015**
- **Project Demo Link: https://youtu.be/Gt2cb455X08**