**Fertilizers Recommendation System for Disease**

**Predictor**

**IBM**

**PROJECT REPORT**

*Submitted by*

**TEAM ID :PNT2022TMID19532**

**BARATH N (722819106012)**

**DHARUN PANDIAN R(722819106017)**

**DINESHKUMAR M(722819106019)**

**KANMANI R (722819106040)**

*In partial fulfilment for the award of the degree Of*

**BACHELOR OF ENGINEERING**

In

**ELECTRONICS AND COMMUNICATION ENGINEERING**



# INDEX

1. **INTRODUCTION**
   1. Project Overview
   2. Purpose

1. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition

1. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit

1. **REQUIREMENT ANALYSIS** 
   1. Functional requirement
   2. Non-Functional requirements

1. **PROJECT DESIGN** 
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
2. **PROJECT PLANNING & SCHEDULING** 
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
3. **CODING & SOLUTIONING (Explain the features added in the project along with code)** 
   1. Feature 1
   2. Feature 2
   3. Database Schema (if Applicable)

1. **TESTING** 
   1. Test Cases
   2. User Acceptance Testing
2. **RESULTS** 
   1. Performance Metrics

1. **ADVANTAGES & DISADVANTAGES**
2. **CONCLUSION**
3. **FUTURE SCOPE**

**13.APPENDIX**

**13.1** Source Code

**13.2** GitHub & Project Demo Link

1.INTRODUCTION :

* 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. 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 :

Agriculture is considered as the main and the foremost culture practiced in India. Ancient people cultivate the crops in their own land and so they have been accommodated to their needs. Since the invention of new innovative technologies and techniques in the agriculture field is slowly degrading. Due to these, abundant invention people are been concentrated on cultivating artificial products that is hybrid products where there leads to an unhealthy life. Nowadays, modern people don’t have awareness about the cultivation of the crops in a right time and at a right place. Because of these cultivating techniques the seasonal climatic conditions are also being changed against the fundamental assets like soil, water and air which lead to insecurity of food. The machine learning learns the algorithm based on the supervised, unsupervised, and Reinforcement learning each has their importance and limitations. Supervised learning the algorithm builds a mathematical model from a set of data that contains both the inputs and the desired outputs. Unsupervised learning-the algorithm builds a mathematical model from a set of data which contains only inputs and no desired output labels. Semi-supervised learning- algorithms develop mathematical models from incomplete training data, where a portion of the sample input doesn't have labels.

Literature Survey:

[1] The aim of proposed system is to help farmers to cultivate crop for better yield. The crops selected in this work are based on important crops from selected location. The selected crops are Rice, Jowar, Wheat, Soyabean, and Sunflower, Cotton, Sugarcane, Tobacco, Onion, Dry Chili etc. The dataset of crop yield is collected from last 5 years from different sources.

[2]. Recommendation of fertilizers is based on Nitrogen, Phosphorous and Potassium measurements from soil. Nitrogen in the soil is responsible for color of leaves. If low quantity of nitrogen is found in the soil, then plants will have slight yellowish leaves and if quantity is moderate or high, it will have greener leaves. The phosphorous content in the soil is responsible for the reproductive system of the plant. Its value will predict the growth of fruits and flowers of the plants. The potassium content of soil is responsible for its overall growth. Its value will predict how stronger the plant roots will be and will also determine the overall growth process of the plant.

[3].The proposed system aims to estimate the nutrient content and recommend the suitable fertilizer to be used for higher productivity. Under application of fertilizer results in low yield due to insufficient nutrients present in the soil for the crop.Over usage of fertilizer results in soil pollution. The food products from the polluted soil will be food poisoning and health issues for the consumers. The system consists of four modules as follows:1)Soil analysis,2)Interfacing with java API,3)Data Anaytics,4)Recommendation.

[4]. The proposed system recommends the best suitable crop for particular land by considering parameters as annual rainfall, temperature, humidity and soil pH. Among these parameters annual rainfall is predicted by system itself by using previous year data with SVM algorithm and other parameters are have to be entered by the user. In the output section the system displays a suitable crop, required seeds/acre, market price and approximate yield of the recommended crop and also the system takes NPK values in the input section to display the required NPK for the recommended crop.

[5].The proposed method uses SVM to classify tree leaves, identify the disease and suggest the fertilizer. The proposed method is compared with the existing CNN based leaf disease prediction. The proposed SVM technique gives a better result when compared to existing CNN. For the same set of images, F-Measure for CNN is 0.7and 0.8 for SVM, the accuracy of identification of leaf disease of CNN is 0.6 and SVM is 0.8.

[6].The plant-based SSNM approach is a knowledge-intensive technology in which optimum fertiliser management for a crop field is tailored to specific local conditions for crop yield, growth duration of the variety, crop residue management, past fertiliser use, and input of nutrients from external sources. Such knowledge requirements have slowed the wide-scale promotion and adoption by the farmers. The need for more rapid uptake of the technology by farmers led to the consolidation of research conducted over the last 15 years across Asia into simple delivery systems enabling farmers to rapidly implement SSNM. The delivery system, Nutrient Expert®, is an easy-to-use, interactive computerbased decision tool that focuses on rapidly providing fertiliser recommendation to farmers while minimizing production risks and increasing the likelihood of profit. The tool acquires the necessary information required for decision making on nutrient management through a series of easy-to answer questions, which essentially mask the rigors of the SSNM principles from the end users while maintaining the robustness of the process.

3. IDEATION & PROPOSED SOLUTION:

3.1 Empathy Map :

Agriculture is the main aspect of the economic development of a country.

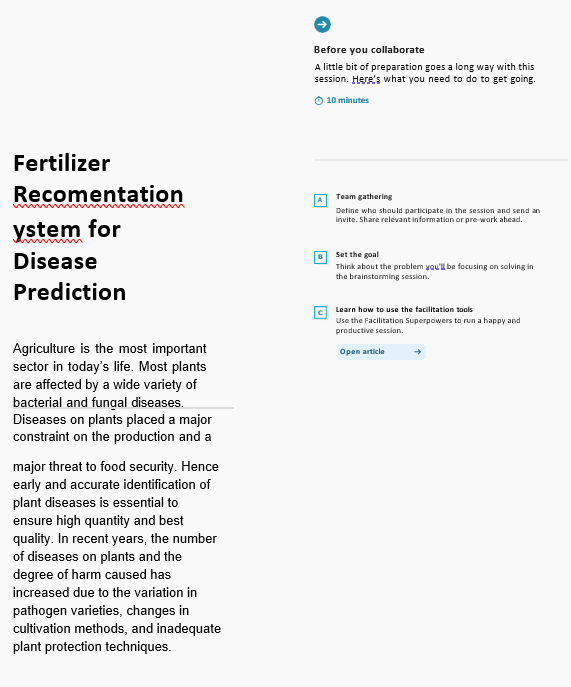
Agriculture is the heart and life of most Indians. By understanding their

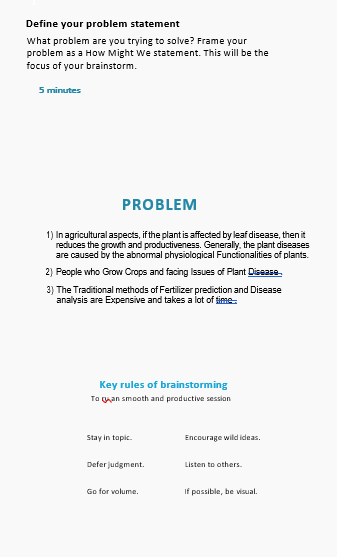
feelings and problems, we can create a better product and contribute to their

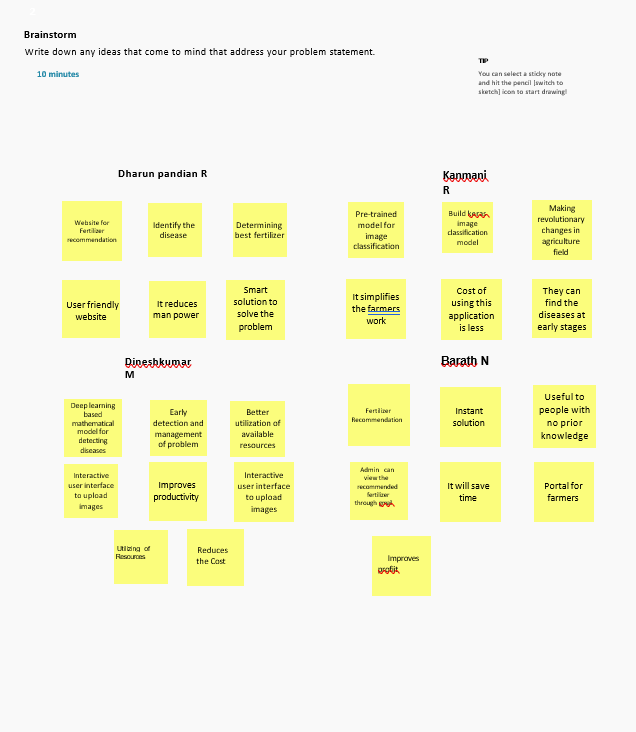
lives. For our project, we are getting surveys from farmers to understand

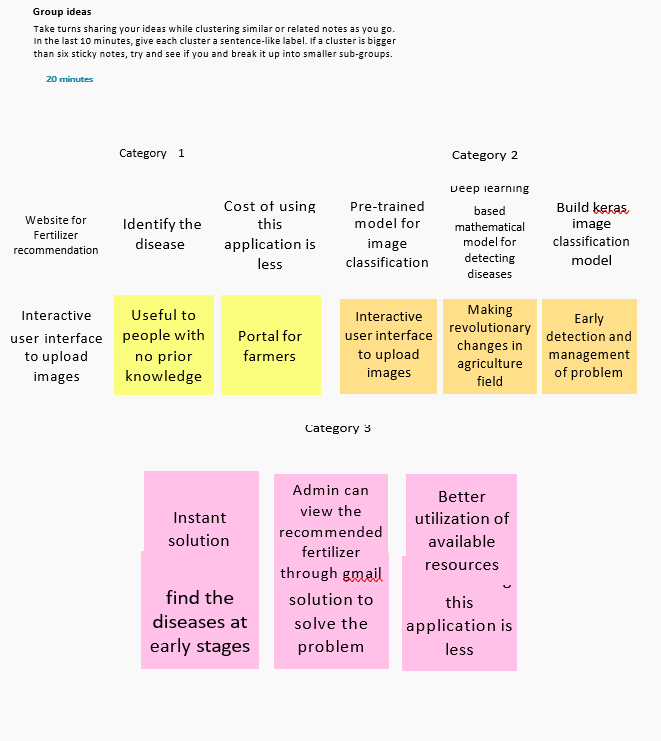
what they truly require and desire.

3.2 Ideation & Brainstorming :

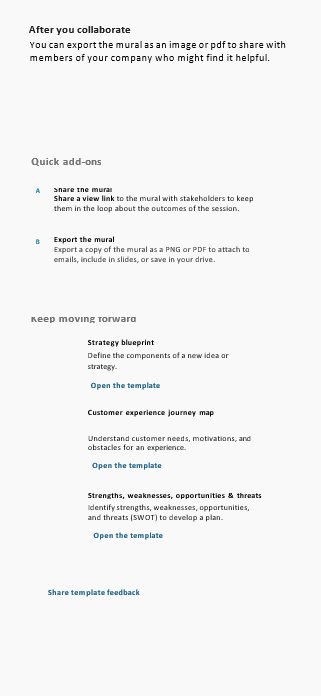










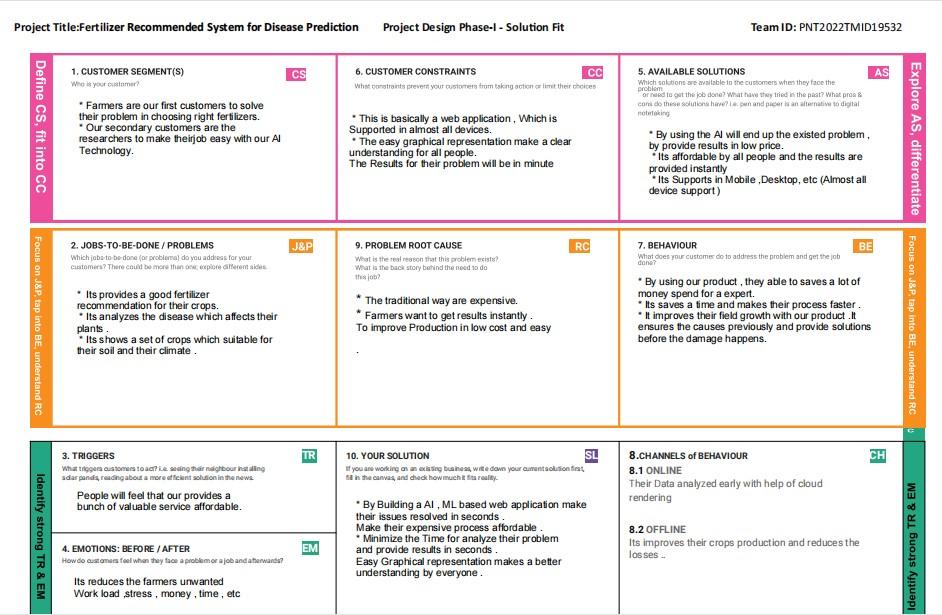


**3.3** Proposed Solution :

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

| **S.No.** | **Parameter** | **Description** |
| --- | --- | --- |
|  | Problem Statement (Problem to be solved) | Disease in plants reduced the quantity and  quality of the plants productivity. Identifying  the disease in plant is hard to find |
|  | Idea / Solution description | One of the solution of the problem is to  identifying the disease in early stage and using  the correct fertilizer. |
|  | Novelty / Uniqueness | This application can suggest good fertilizer for  the disease in the plant by recognizing the  images. |
|  | Social Impact / Customer Satisfaction | It helps the farmer by identifying the disease in  the early stage and increase the quality and  quantity of crops in efficient way. |
|  | Business Model (Revenue Model) | The application is recommends to farmer in  subscription basis. |

* 1. Problem Solution fit:



4.REQUIREMENT ANALYSIS :

4.1 Functional requirement :

|  | **FR.NO** | **Functional requirement** | **Sub requirement (story/subtask)** |
| --- | --- | --- | --- |

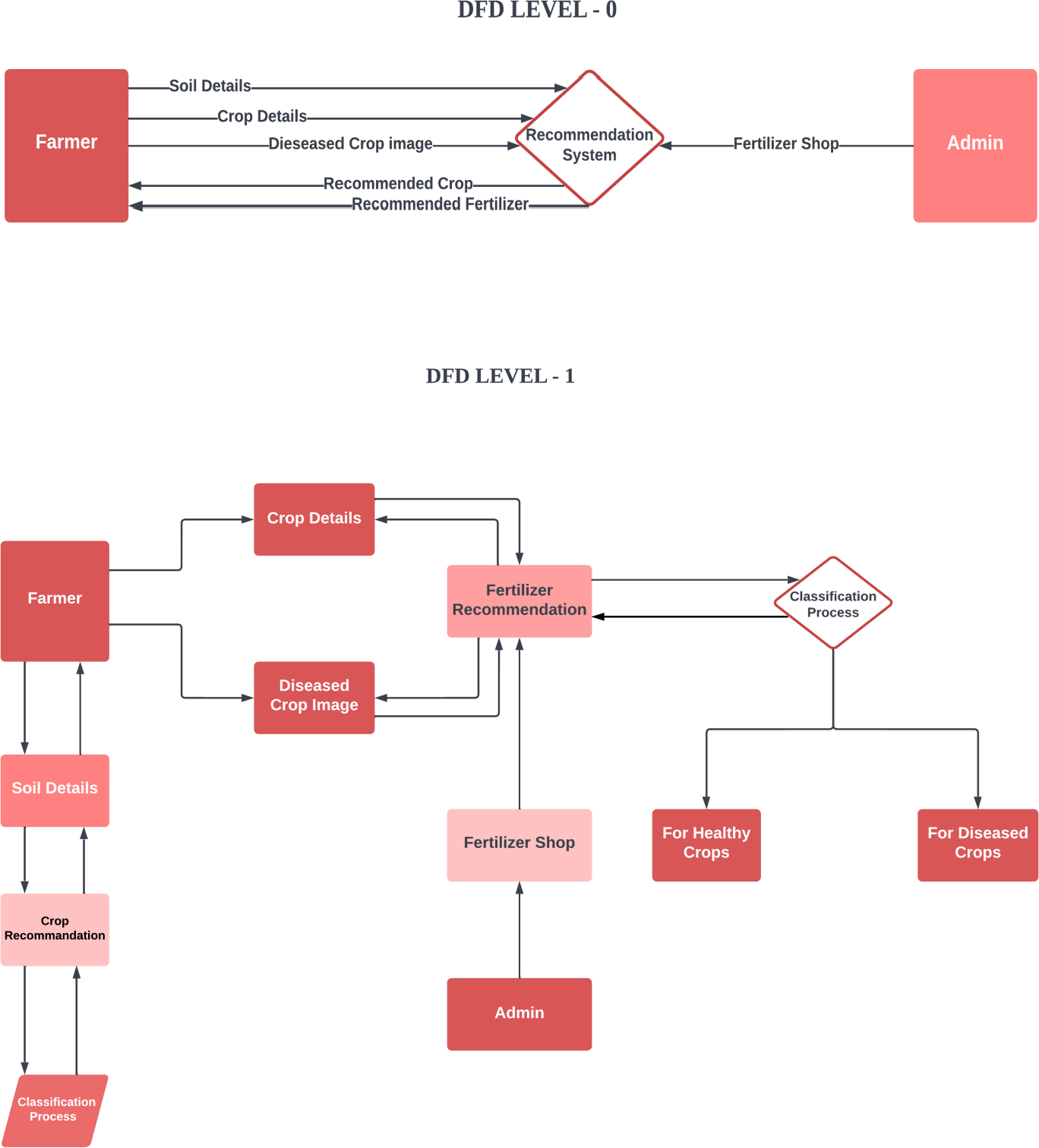
| FR-1 | Capturing image | Capture the image of the leaf And check the parameter of the captured image . |
| --- | --- | --- |
| FR-2 | Image processing | Upload the image for the prediction of the disease in the leaf. |
| FR-3 | Leaf identification | Identify the leaf and predict the disease in leaf. |
| FR-4 | Image description | Suggesting the best fertilizer for the disease. |

**4.2** Non Functional Requirements :

| **NFR.NO** | **Non-functional requirement** | **Description** |
| --- | --- | --- |
| NFR-1 | Usability | Datasets of all the leaf is used to detecting the disease that present in the leaf. |
| NFR-2 | Security | The information belongs to the user and leaf are secured highly. |
| NFR-3 | Reliability | The leaf quality is important for the predicting the disease in leaf. |
| NFR-4 | Performance | The performance is based on the quality of the leaf used for disease prediction |
| NFR-5 | Availability | It is available for all user to predict the disease in the plant |
| NFR-6 | Scalability | Increasing the prediction of the disease in the leaf |

5. PROJECT DESIGN :

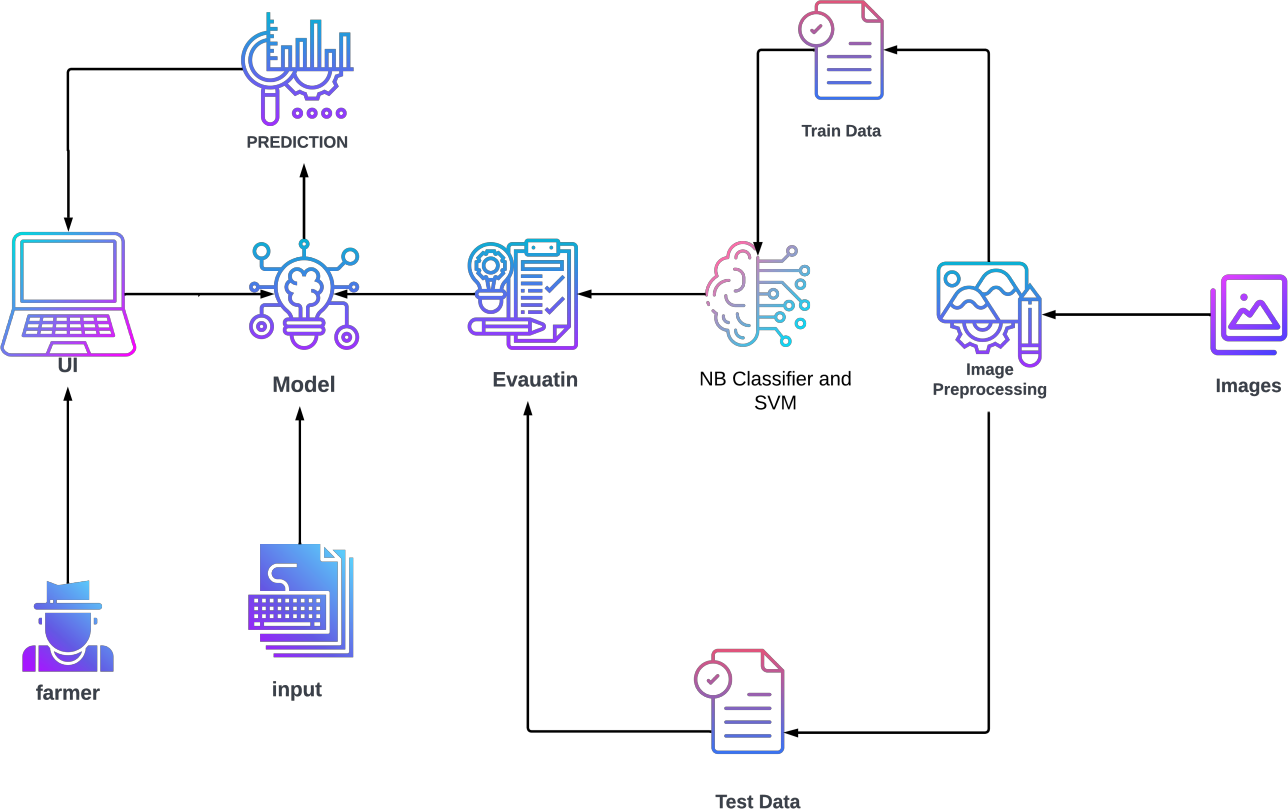
* 1. **Data Flow Diagrams :**



* 1. Solution & Technical Architecture :

Crop disease in plants is predicted and suitable fertilizer is recommended for better yield.The images of the diseased plants are obtained and it is preprocessed against the dataset of diseased plants.

Deep Learning Algorithm is used to process the images and then it is evaluated. Then a model is built on the evaluations, it is then trained using no. of. inputs and predictions are given to the users which subsequently helps in recommending the fertilizers. The Convolutional layers are used to classify and process the images and further helps in recommending the fertilizers.



**5.3** User Stories :

The image can be processed and data should be held by it. If you predict the images can be taken an photo and predict button should be made. Then the prediction will appear. If the leaf is healthy it shows healthy. If the leaf is not healthy it shoes the prediction for fertilizer.

6. PROJECT PLANNING & SCHEDULING :

* 1. Sprint Planning & Estimation :

| **Sprint** | **Functional**  **Requirement (Epic)** | **User Story**  **Number** | **User Story / Task** | **Story Points** | **Priority** | **Team**  **Members** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Data collection and preprocessing | USN-1 | Collecting plant disease dataset | 2 | Low | Barath N |
| Sprint-1 |  | USN-2 | Labelling the dataset according to class | 2 | Medium | Kanmani R |
| Sprint-1 |  | USN-3 | 38 types of plant diseases is labeled accordingly | 3 | Medium | Dinesh kumar M |
| Sprint-1 |  | USN-4 | Data set Will contain both healthy and diseased data | 1 | Low | Dharun pandian R |
| Sprint-1 | Preprocessing | USN-5 | To prepare raw data in a format that the network can accept | 2 | High | Kanmani R |

| Sprint-1 |  | USN-6 | Scaling is used for making data points generalized | 1 | Low | Barath N |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 |  | USN-7 | Shear range image will be distorted along an axis, mostly to create or rectify the perception angle | 3 | High | Dharun pandian R Dinesh kumar M |
| Sprint-1 |  | USN-8 | Zoom Augmentation will randomly zoom the image and adds new pixels for the image | 3 | High | Kanmani R Barath N |
| Sprint-1 |  | USN-9 | Flipping the entire pixels of an image horizontally | 3 | High | Dinesh kumar M  Dharun pandian R |
| Sprint-2 | Training , Testing and Creating a model | USN-10 | Start initiating the model | 3 | Medium | Kanmani R  Dharun pandian R |
| Sprint-2 |  | USN-11 | Adding different layers of cnn( convolution, pooling dense , flatten ) | 2 | Medium | Barath N |
| Sprint-2 |  | USN-12 | Creating/compiling with adam optimizer | 1 | Medium | Dinesh kumar M |
| Sprint-2 |  | USN-13 | Keras - Categorical Cross Entropy Loss Function for multi-class classification | 2 | Medium | Dharun pandian R |
| Sprint-2 |  | USN-14 | creating metrics | 2 | Medium | Dinesh kumar M |
| Sprint-2 |  | USN-15 | train the data with 20 epoch | 3 | High | Barath N  Kanmani R |

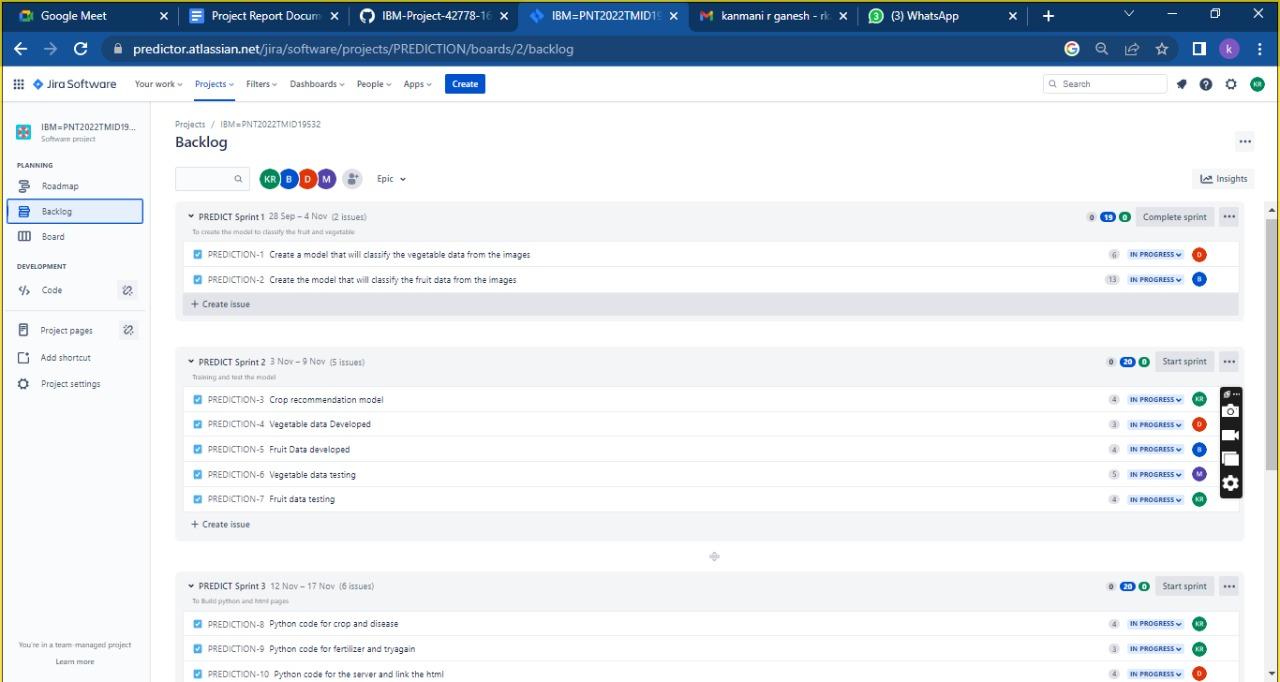
| Sprint-2 |  | USN-16 | testing the model | 5 | High | Dharun pandian R  Dinesh kumar M  Kanmani R |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-2 |  | USN-17 | save the model | 2 | Medium | Barath N |
| Sprint-3 | Flask and Frame workdesign | USN-18 | Creating backend framework with flask | 8 | High | Dinesh kumar M  Kanmani R  Barath N |
| Sprint-3 |  | USN-19 | importing the model file | 3 | Medium | Dharun pandian R |
| Sprint-3 |  | USN-20 | Create route to link htmlRoutes and View Functions in Flask Framework index file | 5 | High | Kanmani R  Barath N |
| Sprint-3 |  | USN-21 | Server Startup, requests and services in a loop | 4 | Medium | Dinesh kumar M  Kanmani R |
| Sprint-4 | Front end web application development | USN-22 | creating a html template with css file | 8 | High | Dinesh kumar M  Kanmani R  Barath N  Dharun pandian R |
| Sprint-4 |  | USN-23 | user can import diseased plant leaf in web page | 2 | Medium | Dinesh kumar M  Kanmani R  Barath N  Dharun pandian R |
| Sprint-4 |  | USN-24 | predicting what is the type of disease occurred for the given input | 2 | Medium | Kanmani R  Barath N |
| Sprint-4 |  | USN-25 | User can classify as healthy or diseased | 2 | Medium | Dharun pandian R  Dinesh kumar M |
| Sprint-4 |  | USN-26 | if plant has disease then suggest fertilizer and pesticides | 3 | Medium | Kanmani R  Barath N |
| Sprint-4 |  | USN-27 | alert the admin about the prediction with the gmail | 3 | Medium | Dinesh kumar M  Kanmani R |

**6.2** Sprint Delivery Schedule :

| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points**  **Completed (as on**  **Planned End Date)** | **Sprint Release Date (Actual)** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | 20 | 6 Days | 28 Oct 2022 | 03 Nov 2022 | 20 | 27 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 04 Nov 2022 | 09 Nov 2022 | 20 | 03 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 10 Nov 2022 | 16 Nov 2022 | 20 | 10 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 17 Nov 2022 | 23 Nov 2022 | 20 | 17 Nov 2022 |

**6.3** Reports from JIRA :

**ACTIVITY LIST**

**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: rgb(235, 30, 132);

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/logo.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') }}">Fertilizer</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') }}">

Predictor

</a>

</h2>

<div class="contact-left-footer mt-4">

</p>

</div>

<div class="w3l-copy text-center">

<p class="text-da">Based on Artificial Intelligence<br> </p>

</div>

<p class="homelogo">

<p>Made in TamilNadu,India</p>

<p>&copy; Copyright 2022 </p>

</div>

</footer>

<!-- //footer -->

<!-- move top icon -->

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

<!-- //move top icon -->

</body>

</html>

{% extends 'layout.html' %}

{% block body %}

<!-- banner -->

<section class="banner\_w3lspvt" id="home">

<div class="csslider infinity" id="slider1">

<div class="banner-top">

<div class="overlay">

<div class="container">

<div class="w3layouts-banner-info text-center">

<h3 class="text-wh">Predictor</h3>

<h4 class="text-wh mx-auto my-4"><b>Get informed decisions about your farming strategy.</b></h4>

</div>

</div>

</div>

</div>

</div>

</section>

<!-- //banner -->

<!-- core values -->

<section class="core-value py-5">

<div class="container py-md-4">

<h3 class="heading mb-sm-5 mb-4 text-center"> About Us</h3>

<div class="row core-grids">

<div class="col-lg-6 core-left">

<img src="{{ url\_for('static', filename='images/core.jpg') }}" class="img-fluid" alt="" />

</div>

<div class="col-lg-6 core-right">

<h3 class="mt-4">Improving Agriculture, Improving Lives, Cultivating Crops To Make Farmers Increase

Profit.</h3>

<p class="mt-3">We use state-of-the-art machine learning and deep learning technologies to help you

guide through

the entire farming process. Make informed decisions to understand the demographics of your area,

understand the

factors that affect your crop and keep them healthy for a super awesome successful yield.</p>

</div>

</div>

</div>

</section>

<!-- //core values -->

<!-- Products & Services -->

<section class="blog py-5">

<div class="container py-md-5">

<h3 class="heading mb-sm-5 mb-4 text-center"> Our Services</h3>

<div class="row blog-grids">

<div class="col-lg-4 col-md-6 blog-left mb-lg-0 mb-sm-5 pb-lg-0 pb-5">

<img src="{{ url\_for('static', filename='images/crop\_background.jpg') }}" class="img-fluid" alt="" />

<a href="{{ url\_for('crop\_recommend') }}">

<div class="blog-info">

<h3>Crop</h3>

<p class="mt-2"> Provide the type of crops to be cultivated which is best adapted

for the respective conditions</p>

</div>

</a>

</div>

<div class="col-lg-4 col-md-6 blog-middle mb-lg-0 mb-sm-5 pb-lg-0 pb-md-5">

<img src="{{ url\_for('static', filename='images/s2.jpg') }}" class="img-fluid" alt="" />

<a href="{{ url\_for('fertilizer\_recommendation') }}">

<div class="blog-info">

<h3>Fertilizer</h3>

<p class="mt-2">Provide type of fertilizer best suited for the particular soil

and the recommended crop</p>

</div>

</a>

</div>

<div class="col-lg-4 col-md-6 blog-right mt-lg-0 mt-5 pt-lg-0 pt-md-5">

<img src="{{ url\_for('static', filename='images/s4.jpg') }}" class="img-fluid" alt="">

<!-- <img src="images/s4.jpg" class="img-fliud" alt="" /> -->

<a href="{{ url\_for('disease\_prediction') }}">

<div class="blog-info">

<h3>Disease</h3>

<p class="mt-2">Predict the name and cause of crop disease and suggestion to rectify and grow better</p>

</div>

</a>

</div>

</div>

</div>

</section>

<!-- //Products & Services -->

<!-- Creating custom grid and hover effect

<section>

<div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">

<div class="hovereffect">

<img class="img-responsive" src="images/s2.jpg" alt="">

<div class="overlay">

<h2>Hover effect 1</h2>

<a class="info" href="#">link here</a>

</div>

</div>

</div> -->

</html>

{% endblock %}

{% extends 'layout.html' %} {% block body %}

<div class="container py-2 mx-auto my-50 h-10 " style="margin: 9rem;">

<div class="row">

<div class="col-sm py-2 py-md-3">

<div class="card card-body" style="justify-content: center; background-color:rgb(103, 255, 235)">

<p class="text-center" style="color: black; font-size: 22px;">{{ prediction }}</p>

</div>

</div>

</div>

</div>

{% endblock %}

{% extends 'layout.html' %} {% block body %}

<style></style>

html body {

background-color: rgb(166, 241, 95);

}

</style>

<br /><br />

<h2 style="text-align: center; margin: 0px; color: black">

<b>Find out which disease has been caught by your plant</b>

</h2>

<br />

<br>

<div style="

width: 350px;

height: 50rem;

margin: 0px auto;

color: black;

border-radius: 25px;

padding: 10px 10px;

font-weight: bold;

">

<form class="form-signin" method=post enctype=multipart/form-data>

<h2 class="h4 mb-3 font-weight-normal"><b>Please Upload The Image</b></h2>

<input type="file" name="file" class="form-control-file" id="inputfile" onchange="preview\_image(event)" style="font-weight: bold;">

<br>

<br>

<img id="output-image" class="rounded mx-auto d-block" />

<button class="btn btn-lg btn-primary btn-block" type="submit" style="font-weight: bold;">Predict</button>

</form>

</div>

<script type="text/javascript">

function preview\_image(event) {

var reader = new FileReader();

reader.onload = function () {

var output = document.getElementById('output-image')

output.src = reader.result;

}

reader.readAsDataURL(event.target.files[0]);

}

</script>

</div>

{% endblock %}{% extends 'layout.html' %} {% block body %}

<div class="container py-2 mx-auto my-50 h-10 " style="margin: 9rem;">

<div class="row">

<div class="col-sm py-2 py-md-3">

<div class="card card-body" style="justify-content: center; background-color:rgb(104, 241, 241)">

<p class="text-center" style="color: black; font-size: 20px;">{{ recommendation }}</p>

</div>

</div>

</div>

</div>

{% endblock %}

{% extends 'layout.html' %} {% block body %}

<style>

html body {

background-color: #5ae789;

}

</style>

<!--Form Section-->

<br /><br />

<h2 style="text-align: center; margin: 0px; color: black">

<b>Get informed advice on fertilizer based on soil</b>

</h2>

<br />

<div

style="

width: 350px;

height: 40rem;

margin: 0px auto;

color: black;

border-radius: 25px;

padding: 10px 10px;

"

>

<form method="POST" action="{{ url\_for('fert\_recommend') }}">

<div class="form-group">

<label for="Nitrogen" style="font-size: 17px"><b>Nitrogen</b></label>

<input

type="number"

class="form-control"

id="Nitrogen"

name="nitrogen"

placeholder="Enter the value (example:50)"

style="font-weight: bold"

required

/>

</div>

<div class="form-group">

<label for="Phosphorous" style="font-size: 17px"

><b>Phosphorous</b></label

>

<input

type="number"

class="form-control"

id="Phosphorous"

name="phosphorous"

placeholder="Enter the value (example:50)"

style="font-weight: bold"

required

/>

</div>

<div class="form-group">

<label for="Pottasium" style="font-size: 17px"><b>Pottasium</b></label>

<input

type="number"

class="form-control"

id="Pottasium"

name="pottasium"

placeholder="Enter the value (example:50)"

style="font-weight: bold"

required

/>

</div>

<!-- <div class="form-group">

<label for="ph" style="font-size: 17px"><b>ph level</b></label>

<input

type="text"

class="form-control"

id="ph"

name="ph"

placeholder="Enter the value"

style="font-weight: bold"

required

/>

</div> -->

<div class="form-group">

<label for="crop" style="font-size: 17px"

><b>Crop you want to grow</b></label

>

<select

name="cropname"

class="form-control"

id="crop"

placeholder="Select a crop"

style="font-weight: bold"

required

>

<option selected>Select crop</option>

<option>rice</option>

<option>maize</option>

<option>chickpea</option>

<option>kidneybeans</option>

<option>pigeonpeas</option>

<option>mothbeans</option>

<option>mungbean</option>

<option>blackgram</option>

<option>lentil</option>

<option>pomegranate</option>

<option>banana</option>

<option>mango</option>

<option>grapes</option>

<option>watermelon</option>

<option>muskmelon</option>

<option>apple</option>

<option>orange</option>

<option>papaya</option>

<option>coconut</option>

<option>cotton</option>

<option>jute</option>

<option>coffee</option>

</select>

</div>

<div class="d-flex justify-content-center">

<button

type="submit"

class="btn btn-info"

style="

color: rgb(3, 8, 29);

font-weight: bold;

width: 140px;

height: 55px;

border-radius: 12px;

font-size: 21px;

"

>

Predict

</button>

</div>

</form>

</div>

{% endblock %}

**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 = 'Harvestify - Home'

return render\_template('index.html', title=title)

# render crop recommendation form page

@ app.route('/crop-recommend')

def crop\_recommend():

title = 'Harvestify - Crop Recommendation'

return render\_template('crop.html', title=title)

# render fertilizer recommendation form page

@ app.route('/fertilizer')

def fertilizer\_recommendation():

title = 'Harvestify - 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 = 'Harvestify - 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 = 'Harvestify - 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('C:/projects/NalaiyaThiran/ProjectDevelopmentPhase/project/predictor/app/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 = 'Harvestify - 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)

import torch

import torch.nn as nn

import torch.nn.functional as F

def ConvBlock(in\_channels, out\_channels, pool=False):

layers = [nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, padding=1),

nn.BatchNorm2d(out\_channels),

nn.ReLU(inplace=True)]

if pool:

layers.append(nn.MaxPool2d(4))

return nn.Sequential(\*layers)

# Model Architecture

class ResNet9(nn.Module):

def \_init\_(self, in\_channels, num\_diseases):

super().\_init\_()

self.conv1 = ConvBlock(in\_channels, 64)

self.conv2 = ConvBlock(64, 128, pool=True) # out\_dim : 128 x 64 x 64

self.res1 = nn.Sequential(ConvBlock(128, 128), ConvBlock(128, 128))

self.conv3 = ConvBlock(128, 256, pool=True) # out\_dim : 256 x 16 x 16

self.conv4 = ConvBlock(256, 512, pool=True) # out\_dim : 512 x 4 x 44

self.res2 = nn.Sequential(ConvBlock(512, 512), ConvBlock(512, 512))

self.classifier = nn.Sequential(nn.MaxPool2d(4),

nn.Flatten(),

nn.Linear(512, num\_diseases))

def forward(self, xb): # xb is the loaded batch

out = self.conv1(xb)

out = self.conv2(out)

out = self.res1(out) + out

out = self.conv3(out)

out = self.conv4(out)

out = self.res2(out) + out

out = self.classifier(out)

return out

8.TESTING

* 1. Test Cases

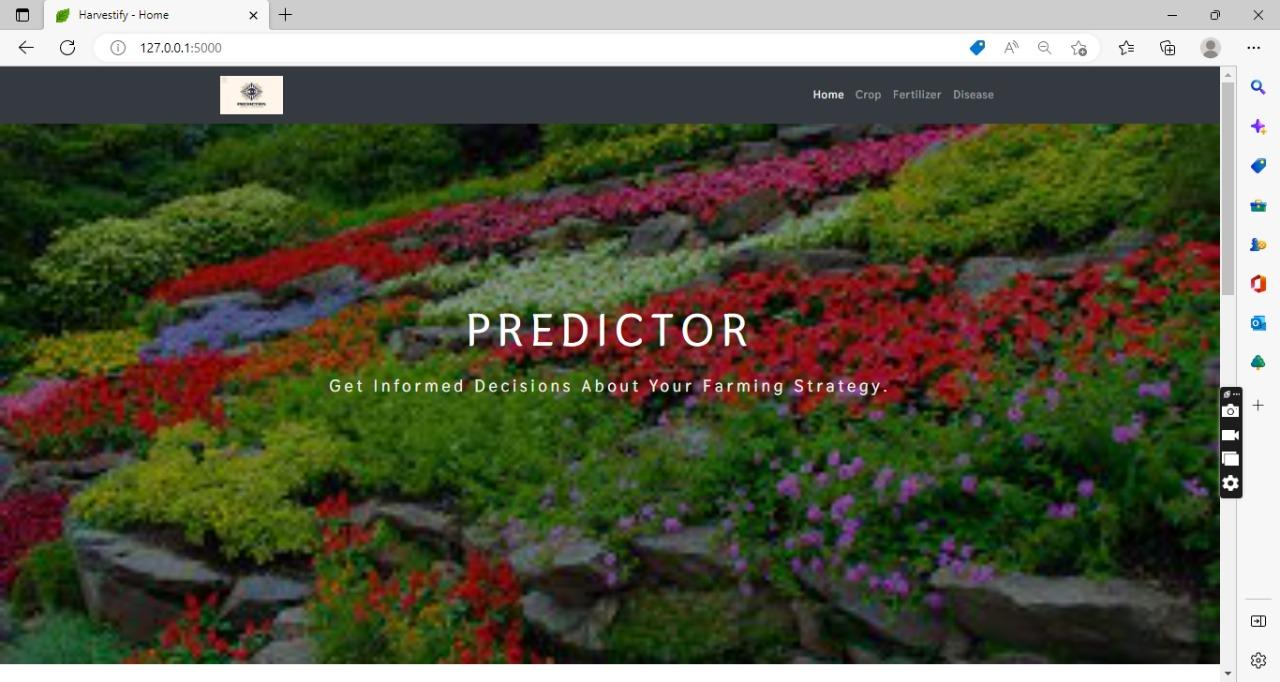
| **SECTION** | **TOTAL CASES** | **NOT TESTED** | **FAIL** | **PASS** |
| --- | --- | --- | --- | --- |
| Leaf spots | 17 | 0 | 0 | 17 |
| Mosaic Leaf Pattern | 51 | 0 | 0 | 51 |
| Misshapen Leaves | 20 | 0 | 0 | 20 |
| Yellow Leaves | 7 | 0 | 0 | 7 |
| Fruit Rots | 9 | 0 | 0 | 9 |
| Fruit Spots | 4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

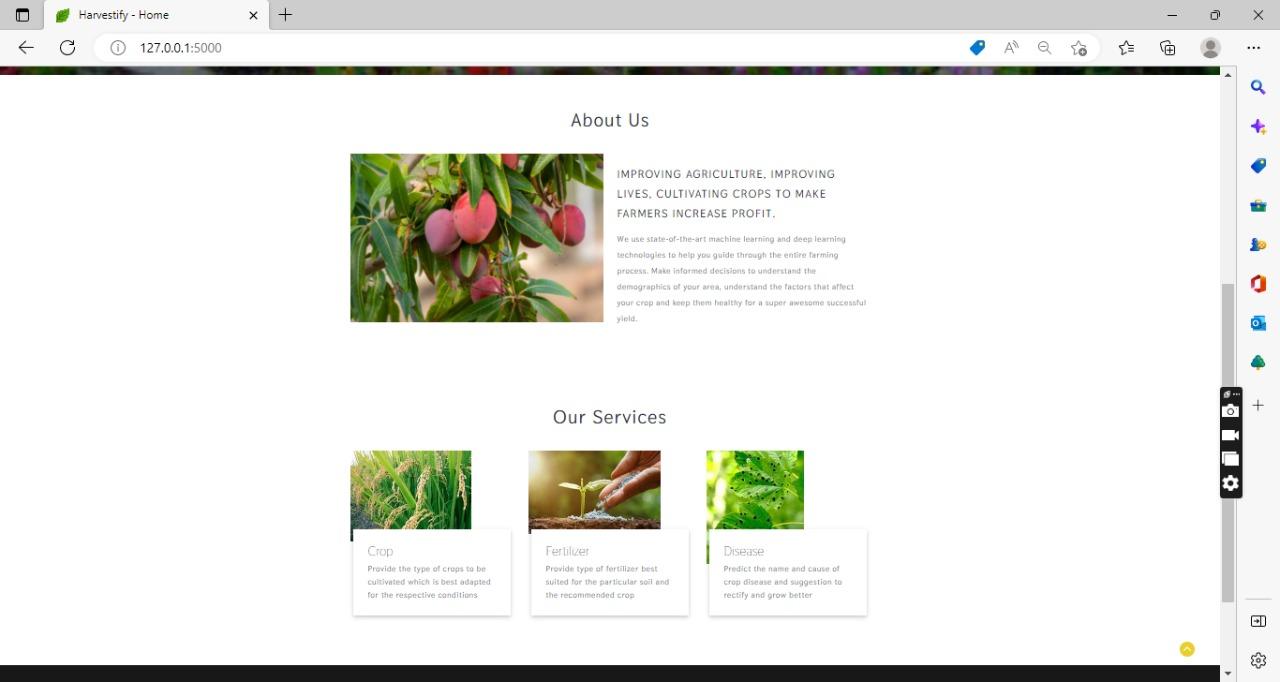
**8.2** User Acceptance Testing

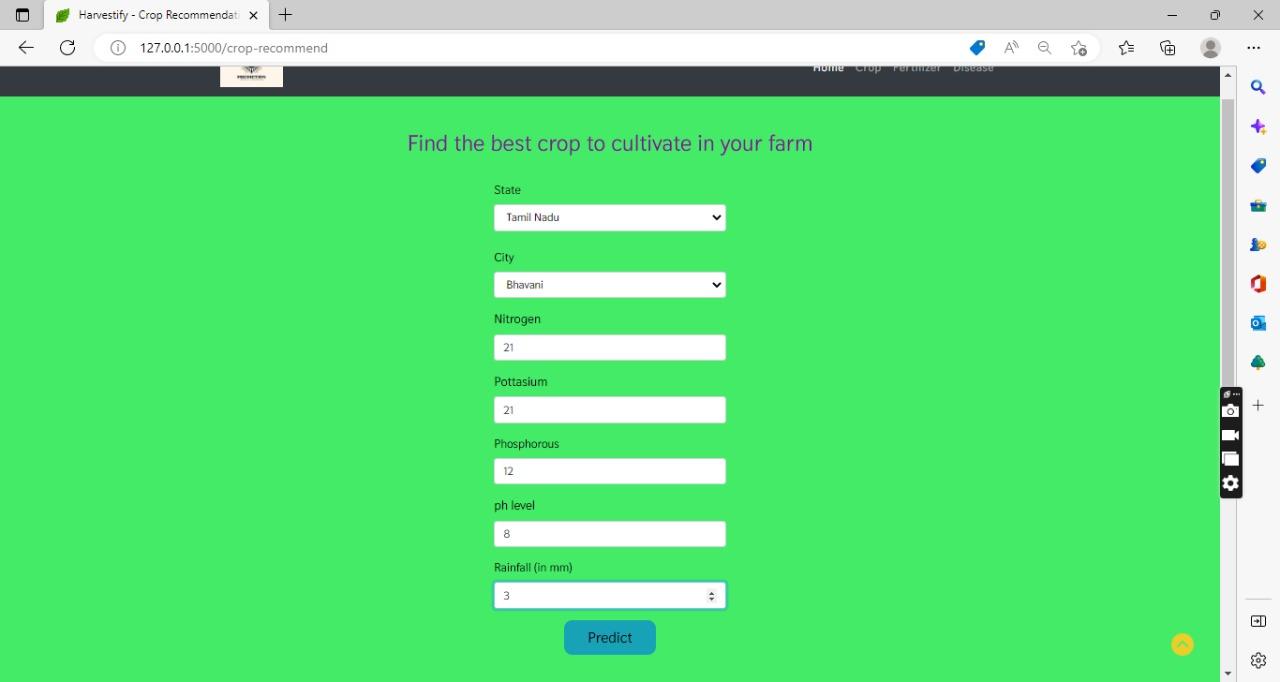
| **RESOLUTION** | **SEVERITY**  **1** | **SEVERITY**  **2** | **SEVERITY**  **3** | **SEVERITY**  **4** | **SUBTOTAL** |
| --- | --- | --- | --- | --- | --- |
| Leaf spots | 10 | 4 | 2 | 3 | 19 |
| Mosaic Leaf Pattern | 9 | 6 | 3 | 6 | 24 |
| Misshapen Leaves | 2 | 7 | 0 | 1 | 10 |
| Yellow Leaves | 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 |

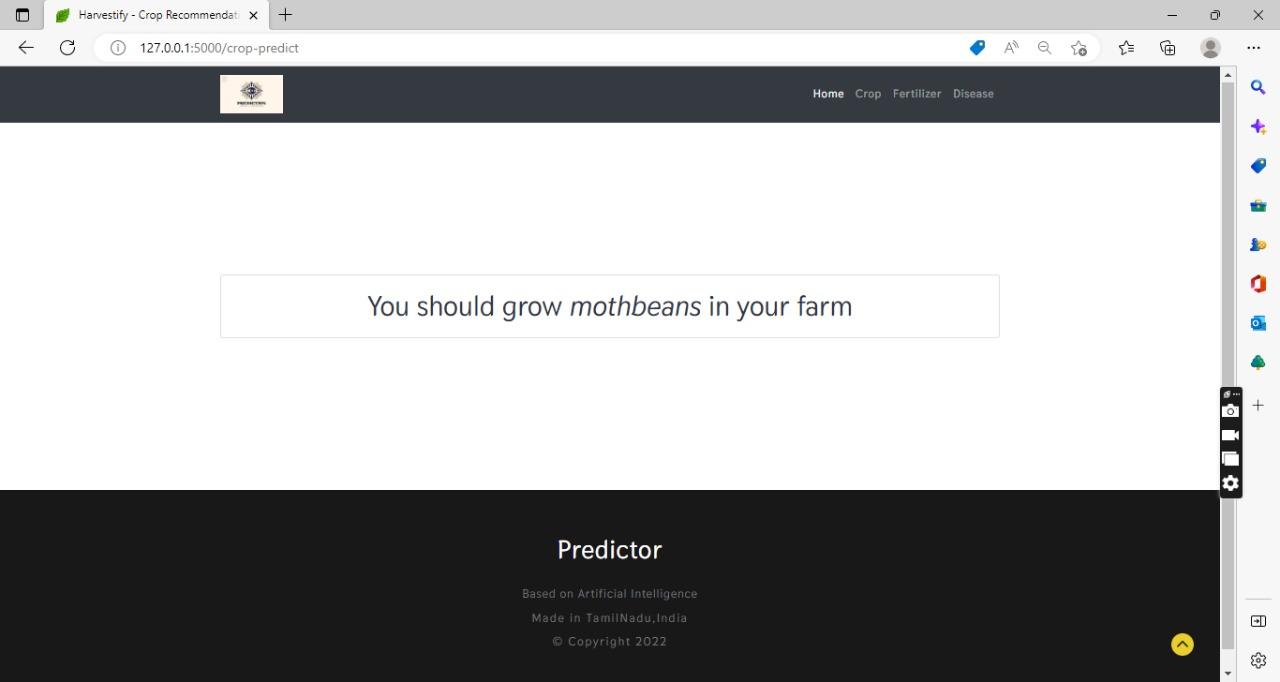
9.RESULTS :

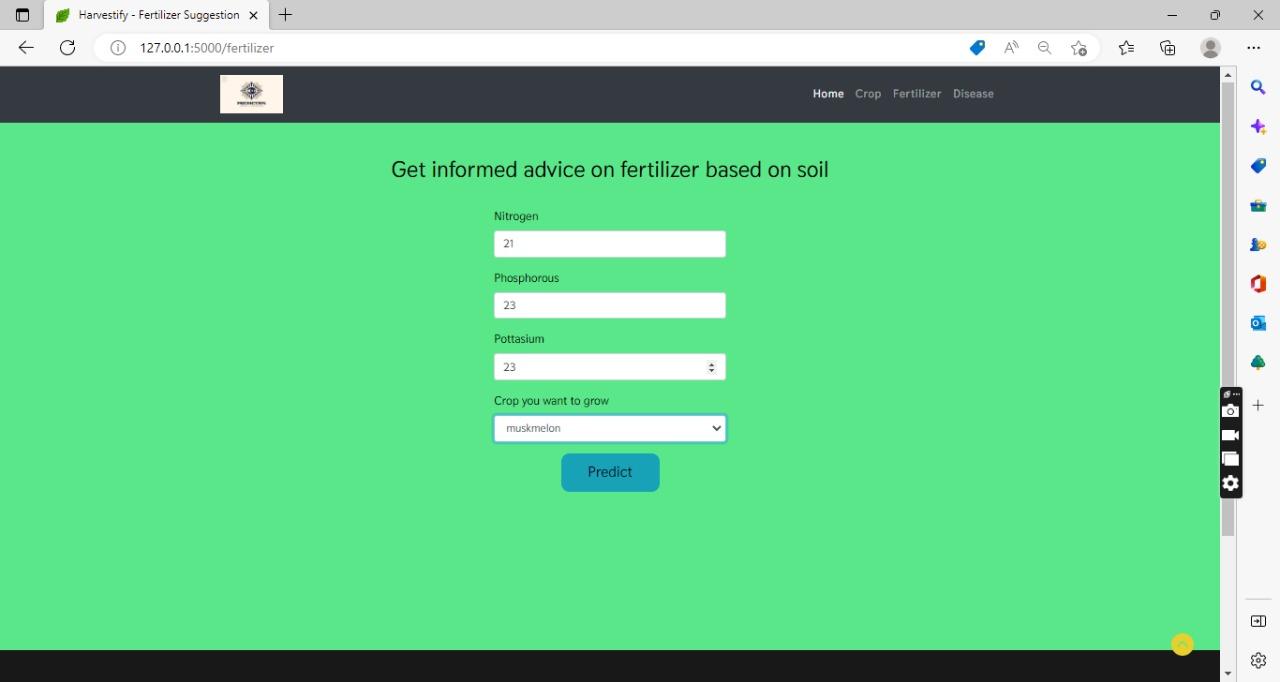
* 1. Performance Metrics :

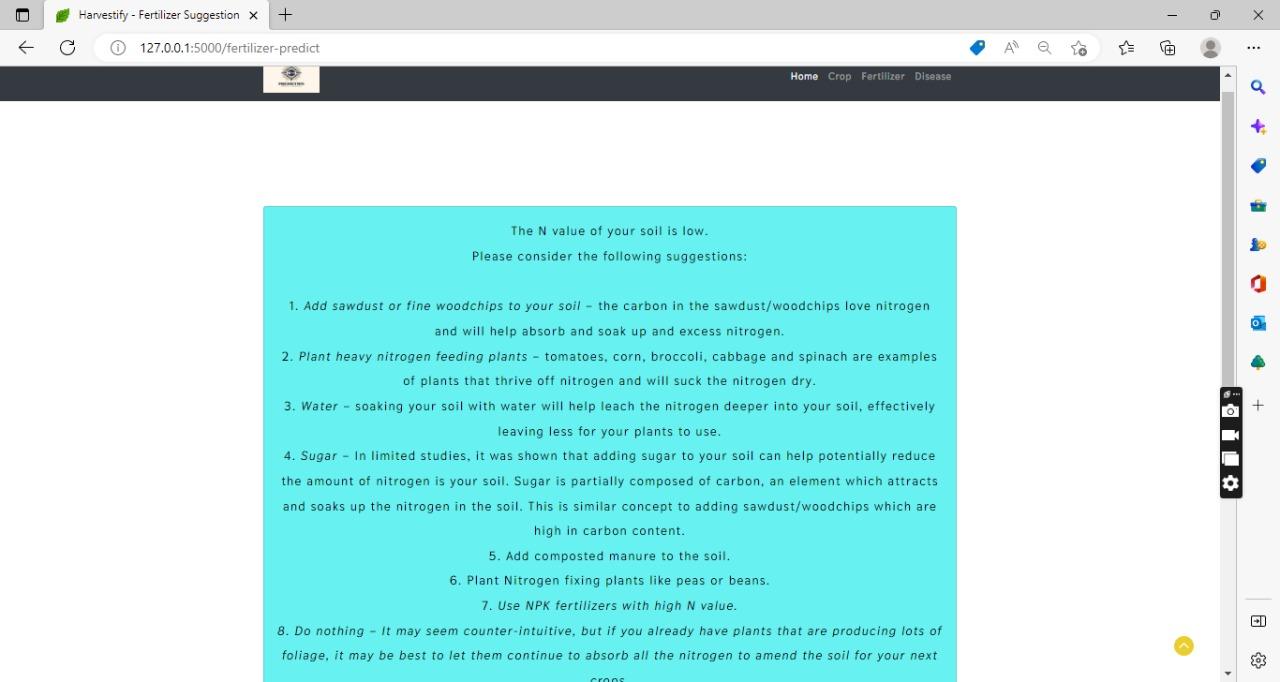




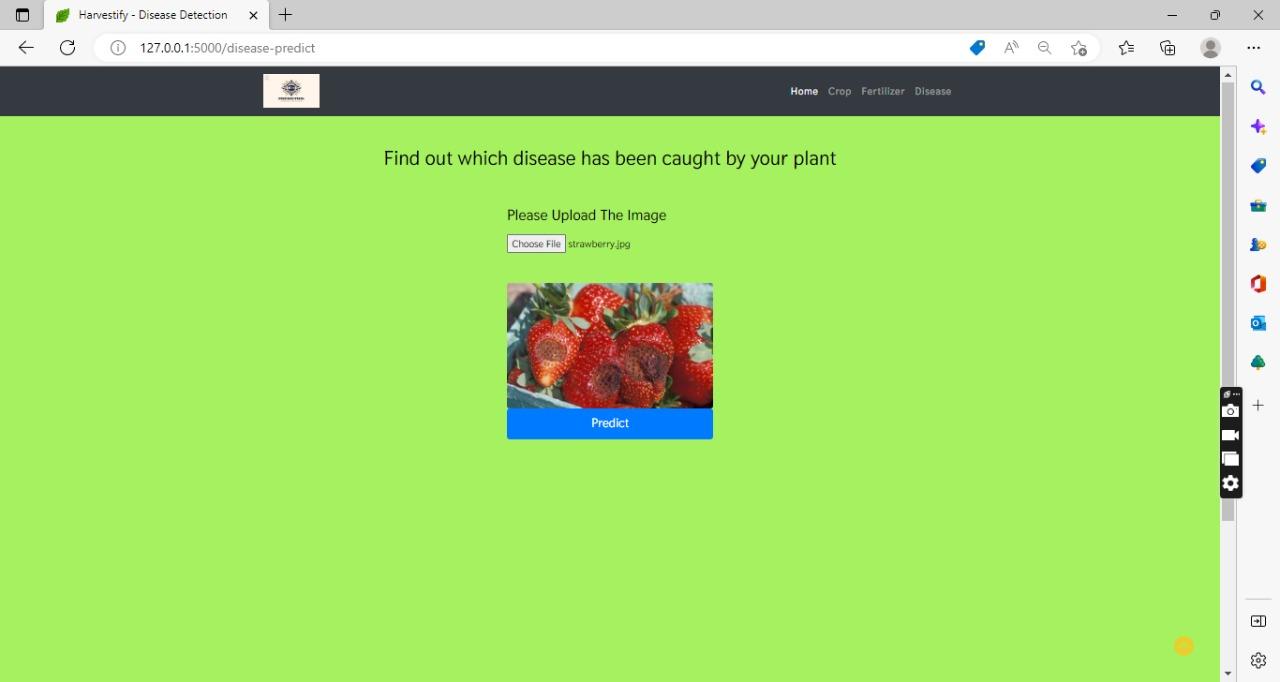


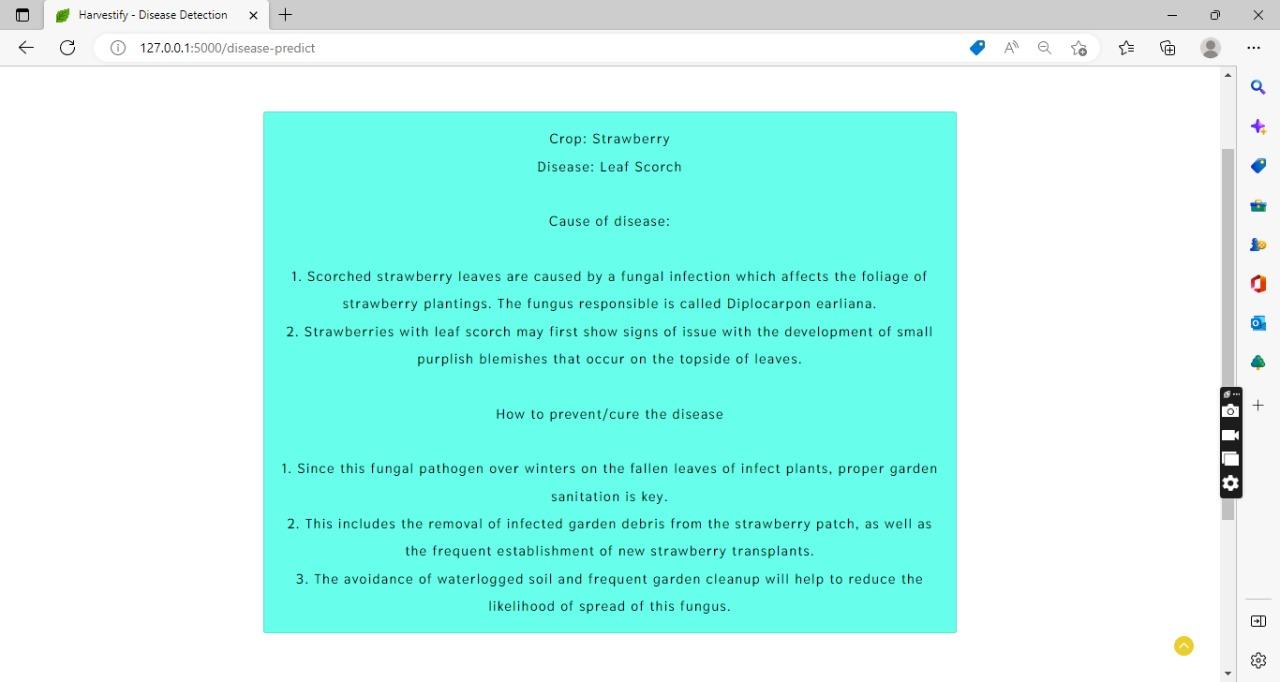












10.ADVANTAGES & DISADVANTAGES

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.

**APPENDIX**

**13.1** 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 = 'Harvestify - Home'

return render\_template('index.html', title=title)

# render crop recommendation form page

@ app.route('/crop-recommend')

def crop\_recommend():

title = 'Harvestify - Crop Recommendation'

return render\_template('crop.html', title=title)

# render fertilizer recommendation form page

@ app.route('/fertilizer')

def fertilizer\_recommendation():

title = 'Harvestify - 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 = 'Harvestify - 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 = 'Harvestify - 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('C:/projects/NalaiyaThiran/ProjectDevelopmentPhase/project/predictor/app/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 = 'Harvestify - 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)

import torch

import torch.nn as nn

import torch.nn.functional as F

def ConvBlock(in\_channels, out\_channels, pool=False):

layers = [nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, padding=1),

nn.BatchNorm2d(out\_channels),

nn.ReLU(inplace=True)]

if pool:

layers.append(nn.MaxPool2d(4))

return nn.Sequential(\*layers)

# Model Architecture

class ResNet9(nn.Module):

def \_init\_(self, in\_channels, num\_diseases):

super().\_init\_()

self.conv1 = ConvBlock(in\_channels, 64)

self.conv2 = ConvBlock(64, 128, pool=True) # out\_dim : 128 x 64 x 64

self.res1 = nn.Sequential(ConvBlock(128, 128), ConvBlock(128, 128))

self.conv3 = ConvBlock(128, 256, pool=True) # out\_dim : 256 x 16 x 16

self.conv4 = ConvBlock(256, 512, pool=True) # out\_dim : 512 x 4 x 44

self.res2 = nn.Sequential(ConvBlock(512, 512), ConvBlock(512, 512))

self.classifier = nn.Sequential(nn.MaxPool2d(4),

nn.Flatten(),

nn.Linear(512, num\_diseases))

def forward(self, xb): # xb is the loaded batch

out = self.conv1(xb)

out = self.conv2(out)

out = self.res1(out) + out

out = self.conv3(out)

out = self.conv4(out)

out = self.res2(out) + out

out = self.classifier(out)

return out

13.2 GitHub & Project Demo Link:

<https://github.com/IBM-EPBL/IBM-Project-4528-1658733869>

https://www.youtube.com/watch?v=9IrvLzId1-A