Fertilizer Recommendation System For

Disease Prediction

TEAM ID: PNT2022TMID22303

TEAM MEMBERS

R.HARISHKUMAR - 113119UG03032
V.PAVITHRAN - 113119UG03068
P.VIGNESH - 113119UG03115
J.AKASH - 113119UG03005

1. INTRODUCTION

1.1 PROJECT OVERVIEW

In today's world agriculture is very important for life and helps to save the natural resources around as. Doing agriculture is the very hard in current scenario because of many natural disasters are happening every day. Most of the plants are affected by many diseases due to pollution in water, air, soil. Identifying the disease is one of the huge hurtles in agriculture. Most of the plants are affected by leaf disease and it's hard to find to correct fertilizer to cure. The main objective of this project is to identify the disease in the plants and cure it in the early stage of the infection. In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques.
An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

1.2 Purpose

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

2. LITERATURE SURVEY

2.1 Existing problem

Indumathi proposed a method for leaf disease detection and suggest fertilizers to cure leaf diseases. But the method involves less number of train and test sets which results in poor accuracy. Pandi selvi proposed a simple prediction method for soil based fertilizer recommendation system for predicted crop diseases. This method gives less accuracy and prediction. Shiva reddy proposed an IoT based system for leaf disease detection and fertilizer recommendation which is based on Machine Learning techniques yields less 80 percentage accuracies.

2.2 REFERENCES

[1]. R Indumathi.; N Saagari.; V Thejuswini.; R Swarnareka.," Leaf Disease Detectionand Fertilizer Suggestion", IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), 29-30 March 2019, DOI: 10.1109/ICSCAN.2019.8878781.
[2]. P. Pandi Selvi, P. Poornima, "Soil Based Fertilizer Recommendation System for Crop Disease Prediction System", International Journal of Engineering Trends and Applications (IJETA) – Volume 8 Issue 2, Mar-Apr 2021.
[3]. H Shiva reddy, Ganesh hedge, Prof. DR Chinnaya3, "IoT based Leaf Disease Detection and Fertilizer Recommendation", International Research Journal of Engineering and Technology (IRJET), Volume: 06 Issue: 11, Nov 2019, e-ISSN: 2395-0056.

2.3. PROBLEM STATEMENT DEFINITION:-

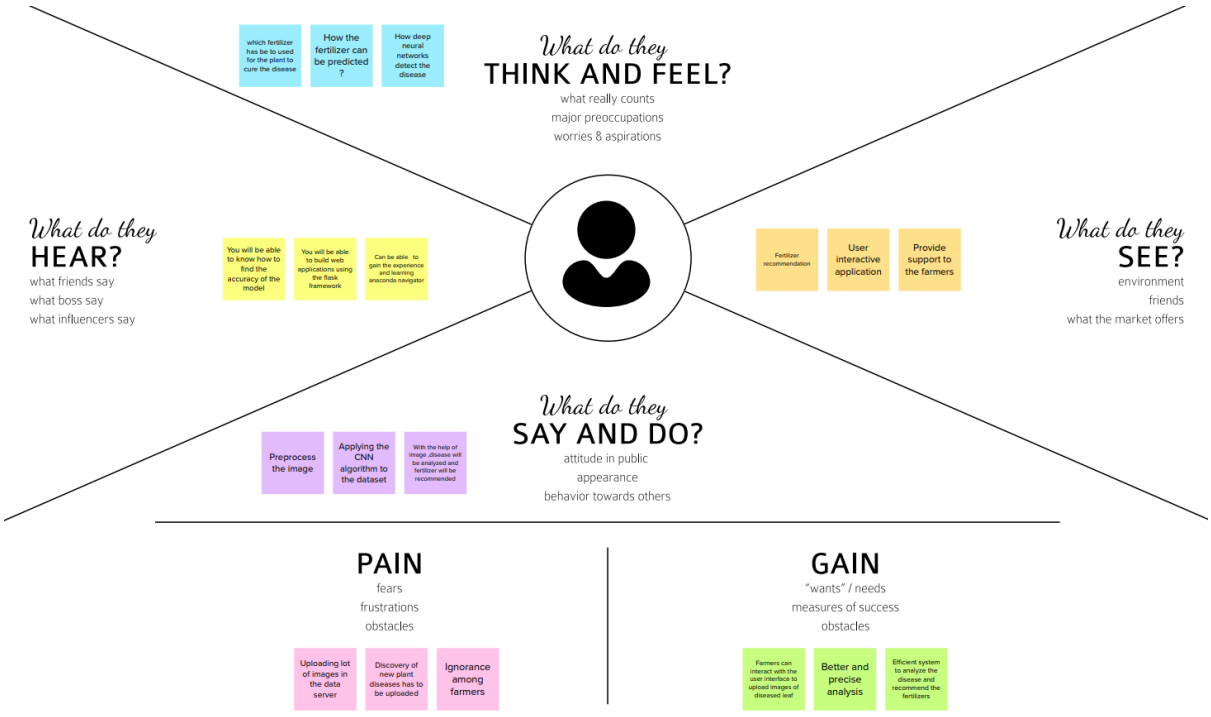
Agriculture plays an important role in economy as well as it is considered the backbone of many countries. Agriculture is the art of growing plants. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Therefore, early and accurate prediction of crop disease is necessary. Hence, a system is to be developed to identify plant disease and recommend the fertilizers.

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	Agricultural worker	To yield healthy products	Plants are affected by diseases	Fungus and bacteria	Depression
PS-2	Farmer	To get high profit in yield	Plants are affected by diseases	Fungal and bacterial diseases	Feeling blue

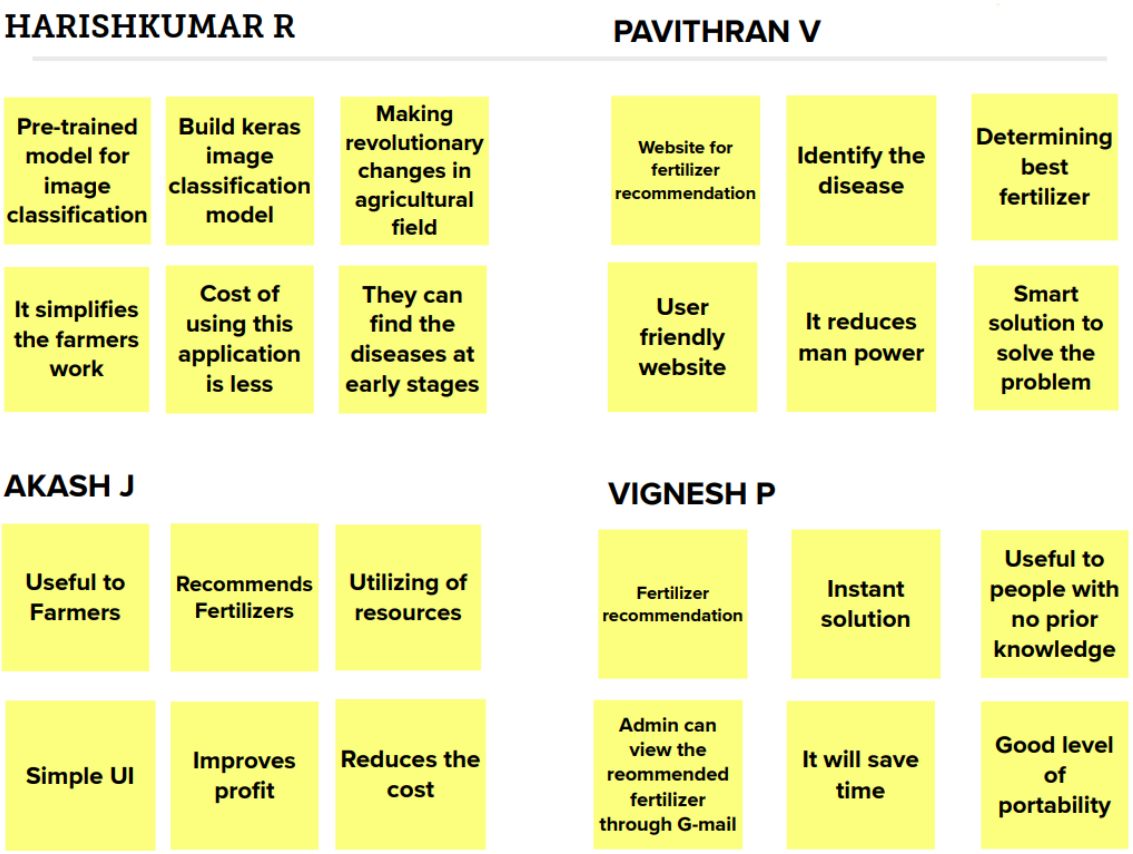
3. IDEATION &PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:

Agriculture plays a major role in economic growth and development. As the provider of food it is a corner stone of human existence. As a furnisher of industrial raw materials it is an important contributor to economic activity in other sectors of the economy. For our project, we are getting surveys from farmers to understand what they truly require and desire.



3.2 IDEATION AND BRAINSTORMING:

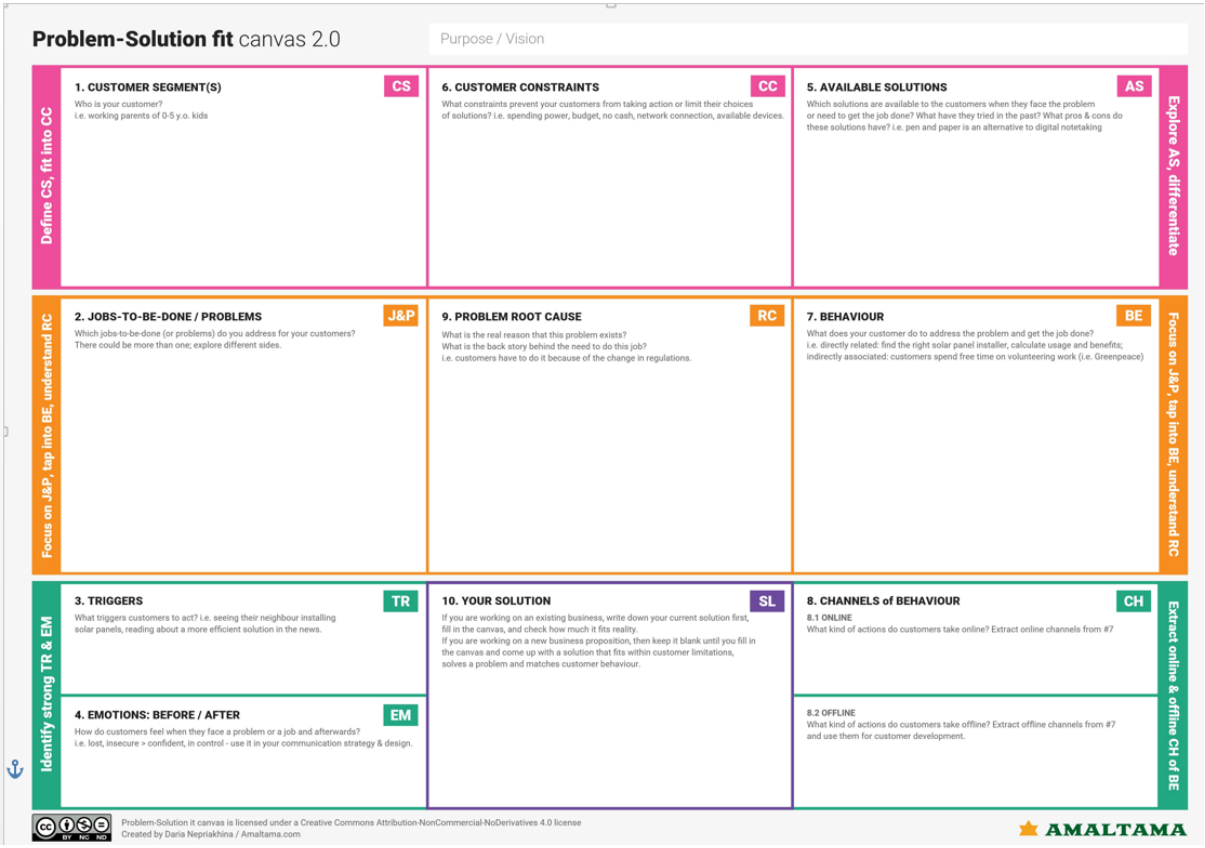


3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	•Farmers are unable to detect crop diseases due to a lack of knowledge and old practices • Growing only certain crops depletes the soil and if the crops are harmed by illnesses

2.	Idea / Solution description	•Plant disease reduces the production and quality of food, fibre and biofuel crops. It has been a major factors that influencing the farmers life as well as our life. •To overcome this problem we develop this project to predict the plant if the crop is affected with which disease, and a viable remedy is then offered to the user.
3.	Novelty / Uniqueness	• Crop diseases detection using image processing in which user get pesticides based on disease images. • To predict the accurate disease for plant and crops we add more image dataset with wider variations are trained. •Most of the farmers are uneducated so we develop the system which is easily accessible by anyone.
4.	Social Impact / Customer Satisf action	• Providing Complete irrigation data through cloud computing. • It helpful for farmers to increase productivity. Increase the usability of natural manure. • Efficient utilization of existing knowledge through artificial intelligence.
5.	Business Model (Revenue Model)	•As long as this system is beneficial to users, subscribtions will increase which gives benefits to industry.
6.	Scalability of the Solution	•Useful for those who don't know the basic about cultivation.

3.4 PROBLEM SOLUTION FIT :



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

FR-2	Image Capture	Take image of a leaf Check the leaf is captured under given parameters
FR-3	Image Processing	Upload the leaf image Click the predict button
FR-4	Updated Native Language	Languages can be changed according to the user, which he is more understandable with. (Ex: English, Hindi, Tamil)
FR-5	Leaf Prediction	Add the pesticides and fertilizers to be used for an unhealthy leaf

FR-6	Image Description	Show the prescribed fertilizer and description of the disease for curing a unhealthy leaf

FR No.	Functional Requirement	Sub Requirement
FR-1	User Registration	Registration through form Registration through Gmail Registration through LinkedIn
FR-7	Providing Datasets	Training datasets Testing datasets
FR-8	Adding Datasets	Fruit datasets for fruits Vegetable datasets for vegetables
FR-9	E-mail Notification	Farmers will be received a Email notification about the leaf and its history

4.2 NON-FUNCTIONAL REQUIREMENTS

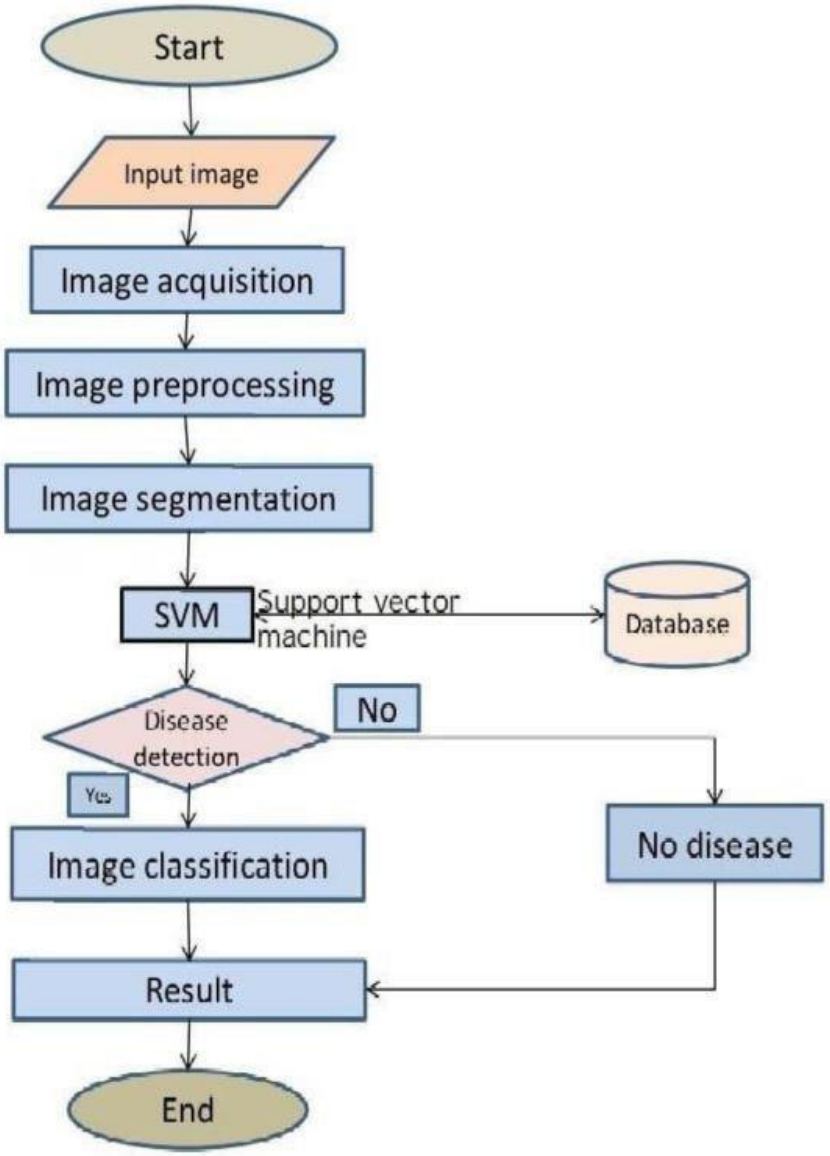
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	Leaf datasets can be used for detection of all kind of leafs Datasets can be reusable Data sets can be prepared according to the leaf
NFR-2	Security	User information and leaf data are secured The algorithms used are more secure
ZZNFR-3	Reliability	The leaf quality is more The datasets and image capturing performs consistently well
NFR-4	Performance	Leaf problem defines once the leaf is detected Performs well according to the quality of leaf provides certain cure to it.
NFR-5	Availability	Quality of leaf will be used again for detection Available and easy access of datasets provided
NFR-6	Scalability	Increase in growth of predicting the results and defining a leaf

5. PROJECT DESIGN

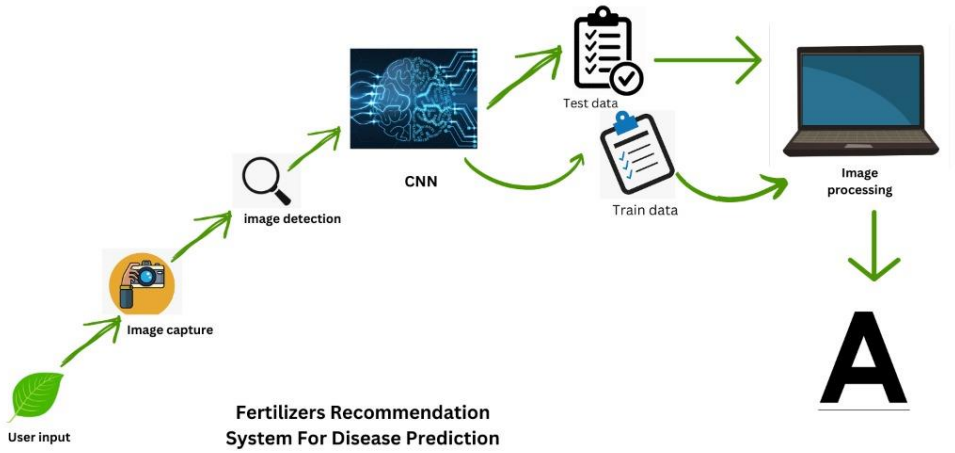
5.1 DATA FLOW DIAGRAMS



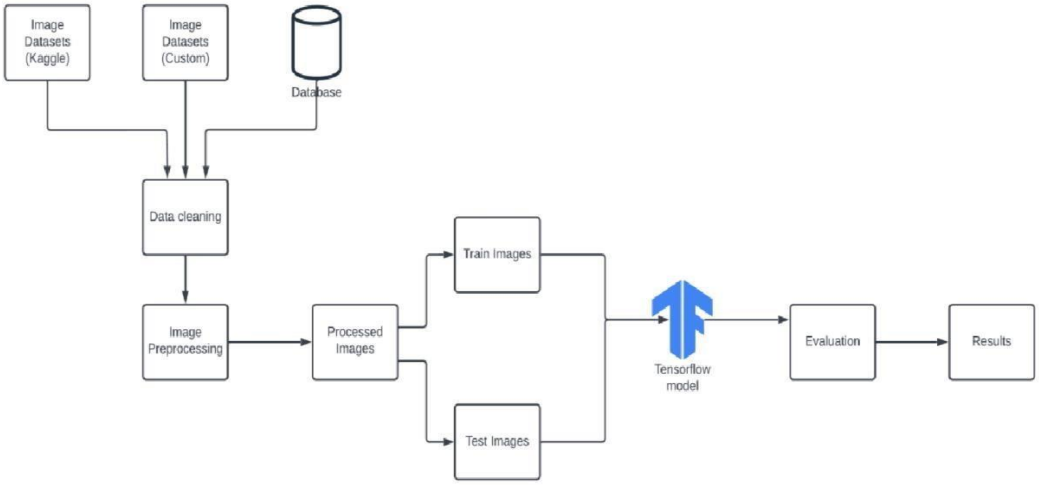
5.2 SOLUTION & TECHNICAL ARCHITECTURE:

Plant crop disease is anticipated, and appropriate fertilizer is advised for a higher yield. The diseased plant photos are acquired and pre processed in comparison to the dataset of diseased plants. The photos are processed using a Deep Learning algorithm, which is subsequently tested. A model is then created based on the evaluations, trained using a variety of inputs, and predictions are presented to the users, aiding in the fertilizer recommendation process.

PROPOSED SOLUTION ARCHITECTURE DIAGRAM



TECHNICAL ARCHITECTURE DIAGRAM:



TECHNOLOGICAL ARCHITECTURE DIAGRAM(2):

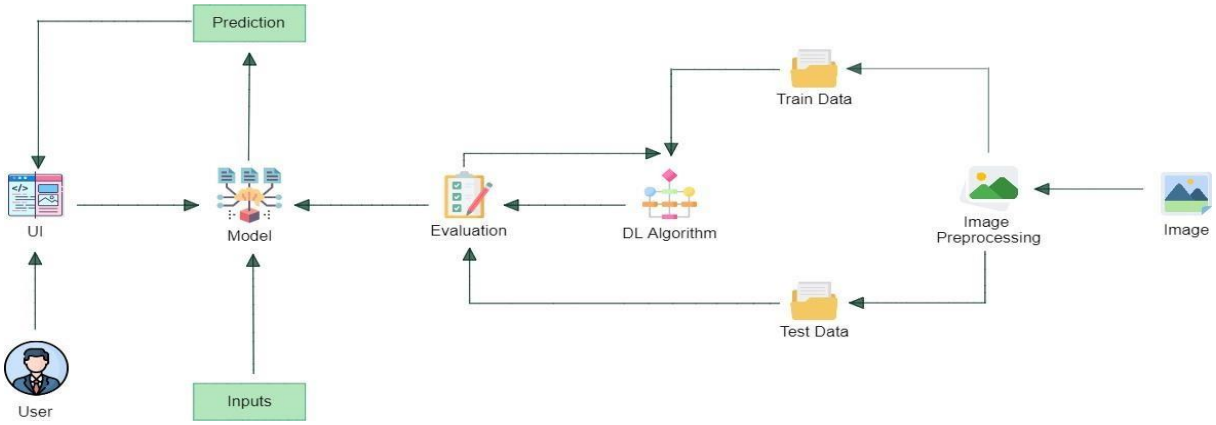


Table-1: Components & Technologies:

1.	User Interface	Web UI	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application logic-1	Image Preprocessing	Python
3.	Application logic-2	CNN Model	IBM Watson STT service
4.	Application logic-3	Web UI Application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	File storage	File storage requirements	IBM DB2, IBM Cloudant etc.
7.	External Api	Purpose of External API used in the application	IBM Block Storage or Other Storage Service or Local Filesystem
8.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
9.	Infrastructure (Server)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask	Flask Frameworks
2.	Security Implementations	CSRF Protection,Secure Flag For Cookies	Flask-WTF, Session Cookie Secure
3.	Scalable Architecture	Micro-Services	Micro Web Application Framework By Flask

5.3 USER STORIES:

A digital camera or similar devices are used to take images of different types, and then those are used to identify the affected area in leaves. Then different types of image-processing techniques are applied to them, the process those images, to get different and useful features needed for the purpose of analyzing later-Plant leaf disease identification is especially needed to predict both the quality and quantity of the First segmentation step primarily based on a mild polygonal leaf model is first achieved and later used to guide the evolution of an energetic contour. Combining global shape descriptors given by the polygonal model with local curvature.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	I can register as a user on the website with eitheran email address or a phone number and password.	I can createmy account.	High	Sprint-3
	Login	USN-2	With theprovided Login credentials, I can accessthe website as a user.	I can log in andaccess myaccount .	High	Sprint-3
	Upload image	USN-3	I can post my data as a userin formats like pdf and doc.	I can uploadmy data.	Medium	Sprint-3
Customer (Web user)	Admin Login	USN-4	. As a user, I can login to web dashboard just Like website dashboard	I can log in and analyze the user data.	High	Sprint-3
	Data collection	USN-5	As a user, I can login to myweb dashboard with the login credentials	I can collect the dataset.	Low	Sprint-1
	Create model	USN-6	As a user, I can view the web application where i can upload my images for getting the suggestion of the fertilizer	I can create andtrain the model.	High	Sprint-1
	Test the model	USN-7	As a user, the fertilizer recommended to me Is in high accurate	I can testthe model.	High	Sprint-2
Administrator	Diagnosis	USN-8	I can access the application's diagnosis results as a userand continue with treatments..	I can access my dashboard	High	Sprint-2

6. PROJECT PLANNING AND SCHEDULING:-

6.1 SPRINT PLANNING AND ESTIMATION:-

Milestone:

Modern Technology are increasing and optimizing the Performance of the Artificial Intelligences (AI) Model. Based Crop Yield Disease Prediction System, is helpful for farmers to prevent the crop from the various Disease which can identify the Disease with in a process of capturing the Image at the plant and Machine Learning Algorithm will give affected Disease Name. In this Project Milestone will be given the Best Solution for the farmer using the complete friendly and simple user interface web application to fetching the solution by own. In addition, process we are planned to add a valid Module that is Fertilizer recommendation for the Specific Disease. It can give both Artificial fertilizer and Natural Fertilizer in suggestion manner.

Activity List:

In Project Management Planning is an important task to scheduling the phrase of the project to the Team Member. In this Activity can shows the various activity are allocated and done by the Team Members! In this Project we can Split into the Four Step of Phrases are:

Phrase 1: Information Collection and Requirement Analysis.
Phrase 2: Project Planning and Developing Modules.
Phrase 3: Implementing the High Accuracy Deep Learning Algorithm to Perform.
Phrase 4: Deploying the Model on Cloud and Testing the Model and UI Performance.

6.1 SPRINT PLANNING & ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points (Total)	Priority	Team Members
Sprint-1	Model Creation and Training (Fruits)		Create a model which can classify diseased fruit plants from given images. I also need to test the model and deploy it on IBM Cloud	8	High	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J
	Model Creation and Training (Vegetables)		Create a model which can classify diseased vegetable plants from given images	2	High	Pavithran.V, Harish kumar .R Vignesh.P, Akash.J

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points (Total)	Priority	Team Members
Sprint-2	Model Creation and Training (Vegetables)		Create a model which can classify diseased vegetable plants from given images and train on IBM Cloud	6	High	Vignesh.P, Pavithran.V, Akash.J, Harish kumar.R
	Registration	USN-1	As a user, I can register by entering my email, password, and confirming my password or via OAuth API	3	Medium	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J.
	Upload page	USN-2	As a user, I will be redirected to a page where I can upload my pictures of crops	4	High	Akash.J, Harish kumar.R, Vignesh.P, Pavithran.V,
	Suggestion results	USN-3	As a user, I can view the results and then obtain the suggestions provided by the ML model	4	High	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J
	Base Flask App		A base Flask web app must be created as an interface for the ML model	2	High	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J
Sprint-3	Login	USN-4	As a user/admin/shopkeeper, I can log into the application by entering email & password	2	High	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J
	User Dashboard	USN-5	As a user, I can view the previous results and history	3	Medium	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J
	Integration		Integrate Flask, CNN model with Cloudant DB	5	Medium	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J
Sprint-4	Dashboard (Admin)	USN-6	As an admin, I can view other user details and uploads for other purposes	2	Medium	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J
	Dashboard (Shopkeeper)	USN-7	As a shopkeeper, I can enter fertilizer products and then update the details if any	2	Low	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J
	Containerization		Create and deploy Helm charts using Docker Image made before	2	Low	Harish kumar.R, Vignesh.P, Pavithran.V, Akash.J

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	10	6 Days	24 Oct 2022	29 Oct 2022	10	30 Oct 2022
Sprint-2	15	6 Days	31 Oct 2022	05 Nov 2022	15	06 Nov 2022
Sprint-3	15	6 Days	07 Nov 2022	12 Nov 2022	15	13 Nov 2022
Sprint-4	12	6 Days	14 Nov 2022	19 Nov 2022	10	20 Nov 2022

7.CODING & SOLUTIONING :

7.1 FEATURE 1 :

The application's registration page is created. User registration is carried out if the user hasn't already done so. Enough work was put into making this process seamless. If the user has registered, he can now log in directly. Email address, name, and password were required for registration. The code to link it to the backend was successful, and this data is stored in Firebase.

7.2 FEATURE 2 :

The trained machine learning model can predict the output from an image that is uploaded, and the nutrition facts are also displayed on the same page. The model's accuracy was determined to be 95%, and when it was trained on the IBM cloud, it reached 99%.

7.3 DATABASE SCHEMA :

The Firebase platform was used. A mechanism for storing and retrieving data that is modelled in ways other than the tabular relations used in relational databases is provided by the Firebase database (NoSQL).

8.TESTING :

8.1 TEST CASES :

The test cases include invalid email and unrecognizable images. For the image part, a text file or other format files were uploaded as a corner case.

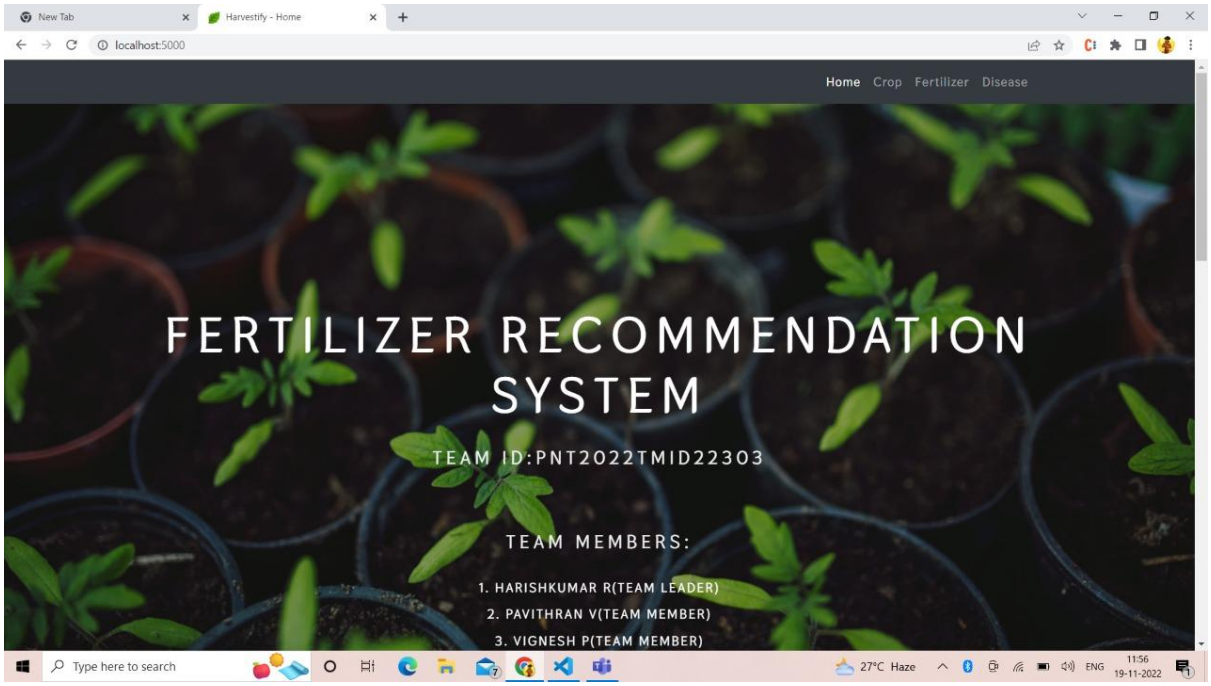
8.2 USER ACCEPTANCE TESTING :

10 users of the test application were able to discover the nutritional data for the fruit image they supplied. This combined and tested both the registration and prediction modules, which showed to provide accurate results.

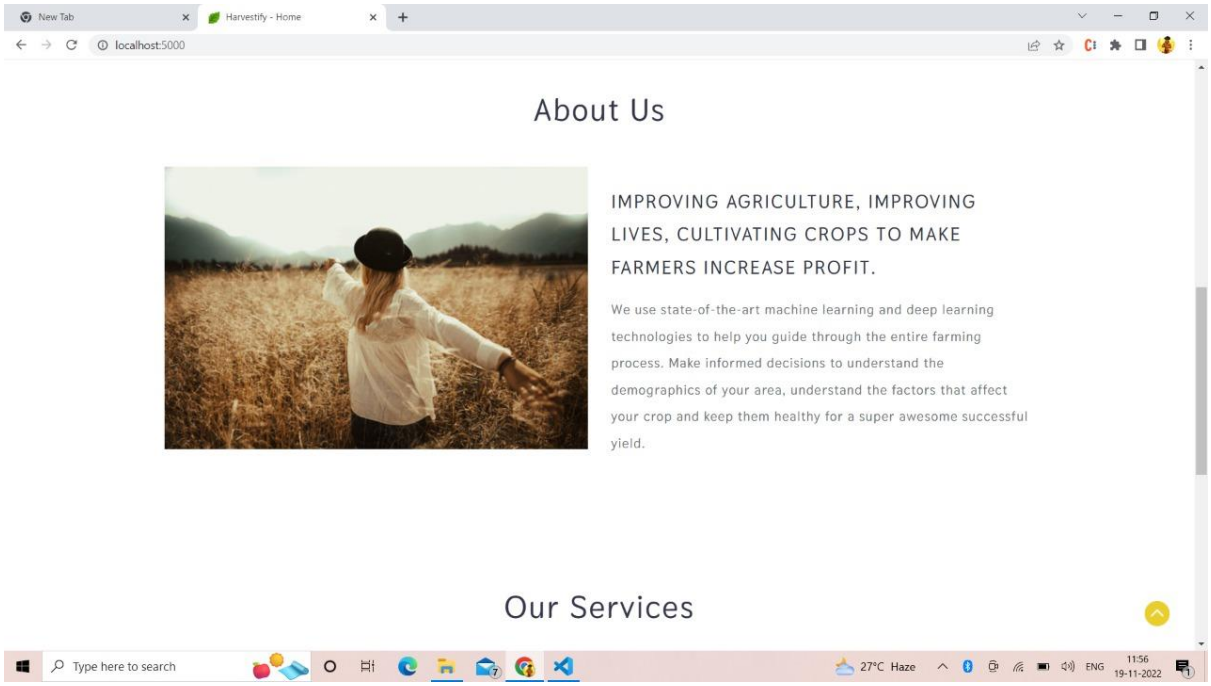
9. RESULTS

9.1 PERFORMANCE METRICS:

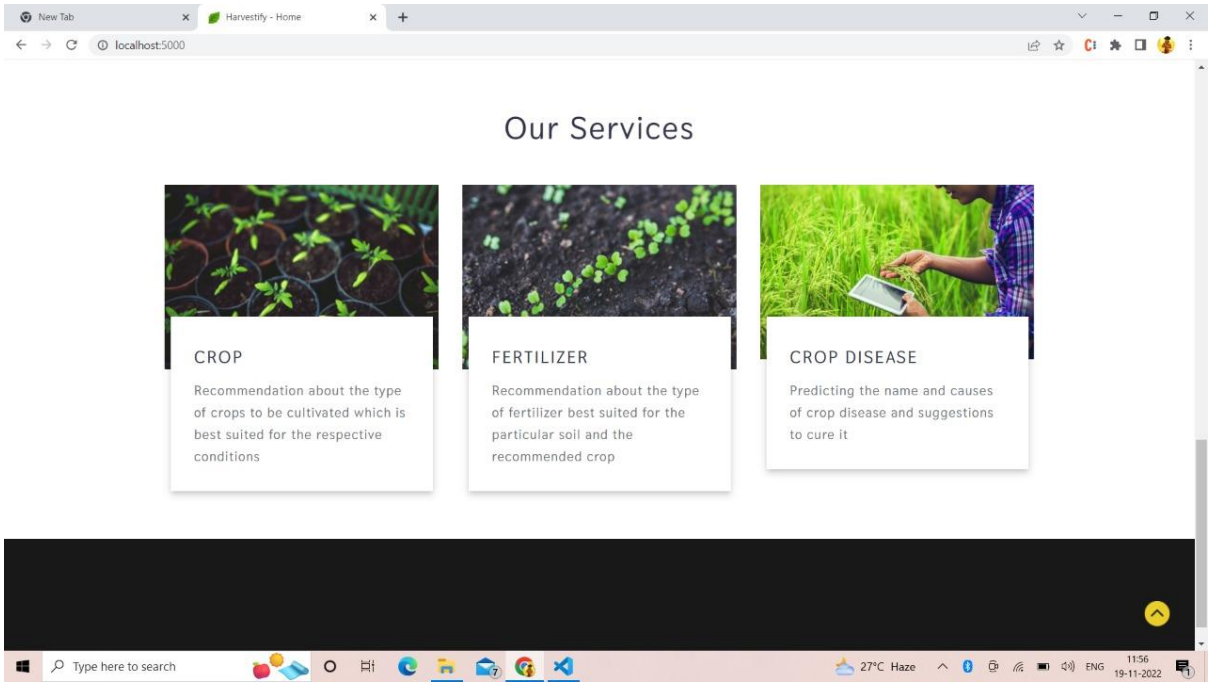
INTRODUCTION PAGE :-



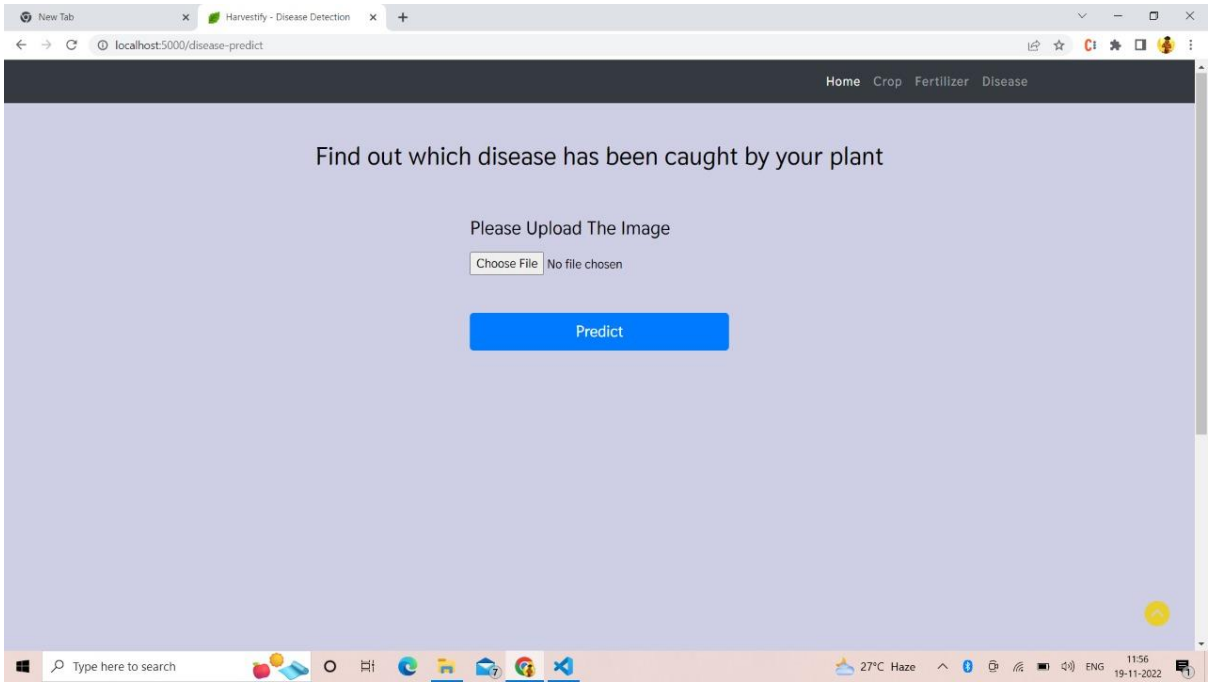
SELECTION MENU:



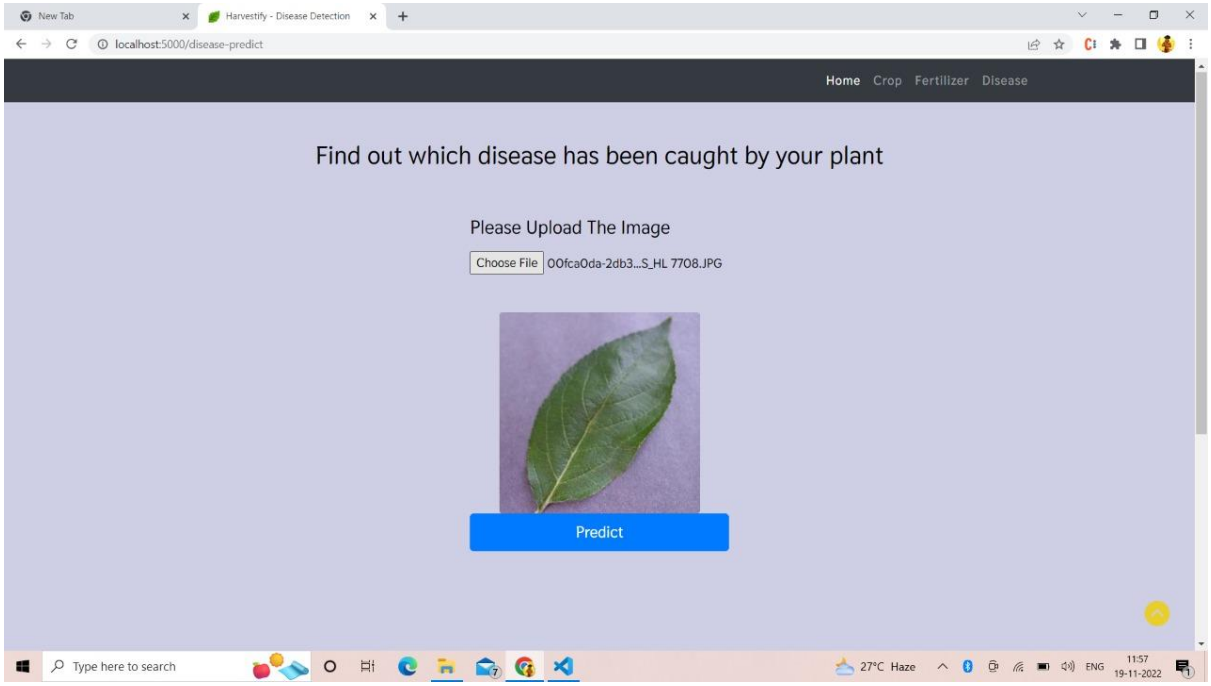
SELECTION MENU:



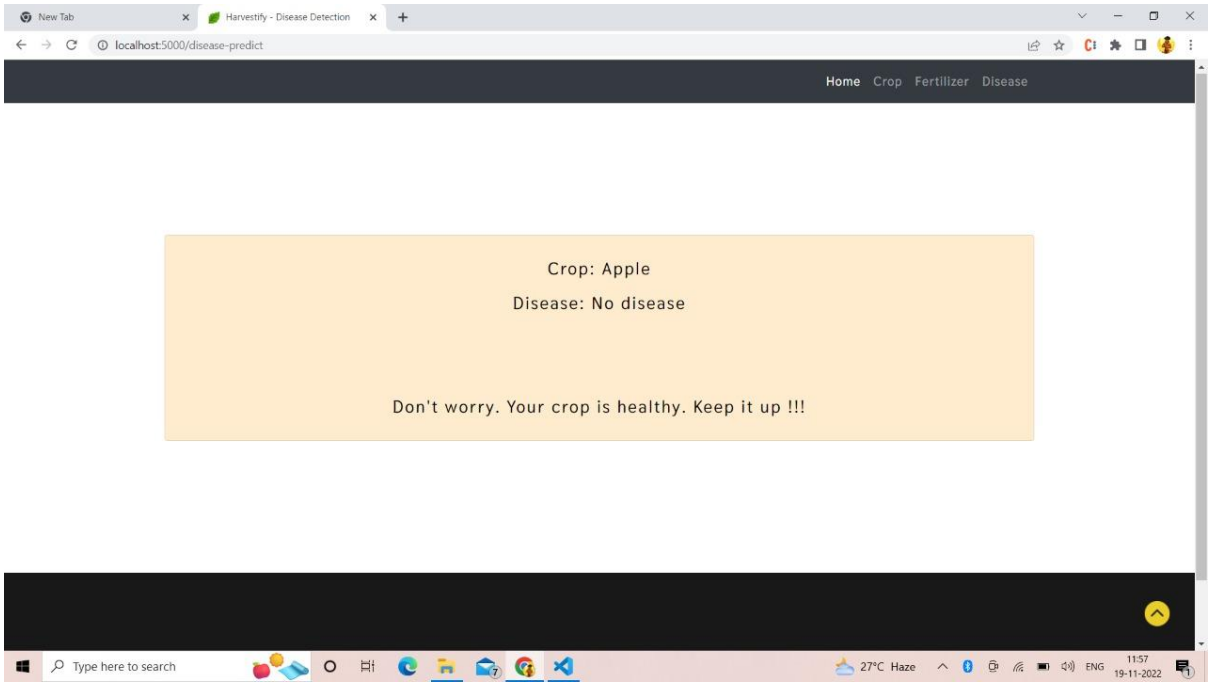
UPLOADING IMAGE:



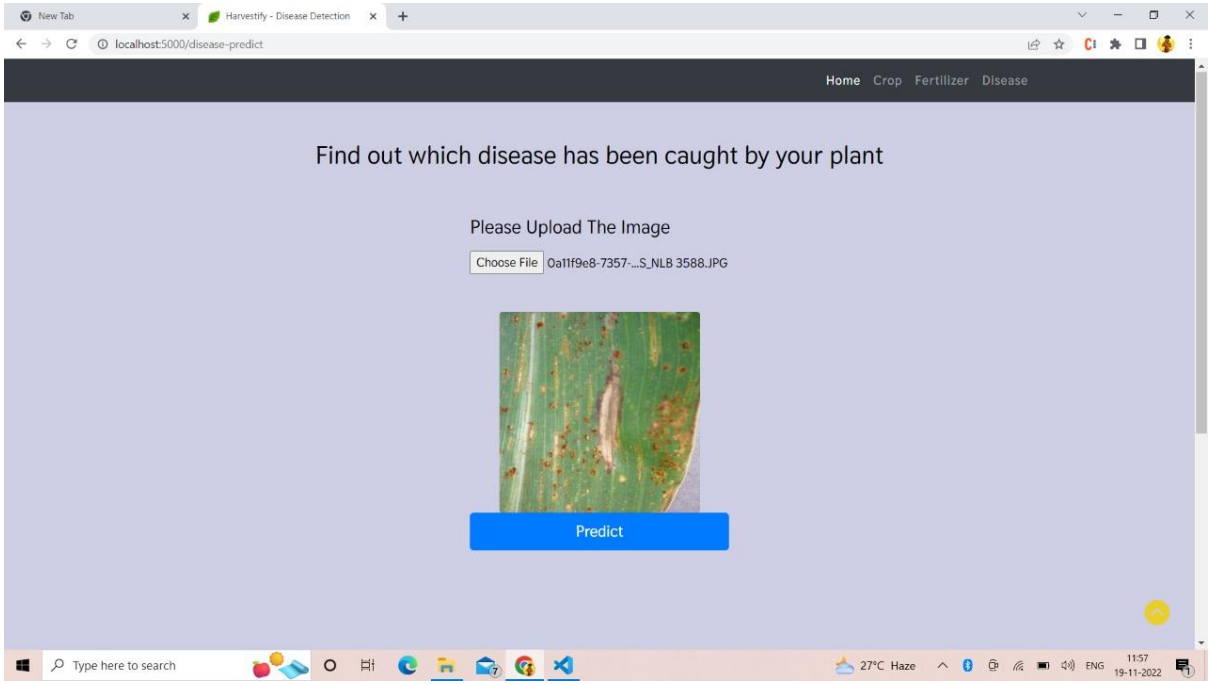
UPLOADING LEAF IMAGE:



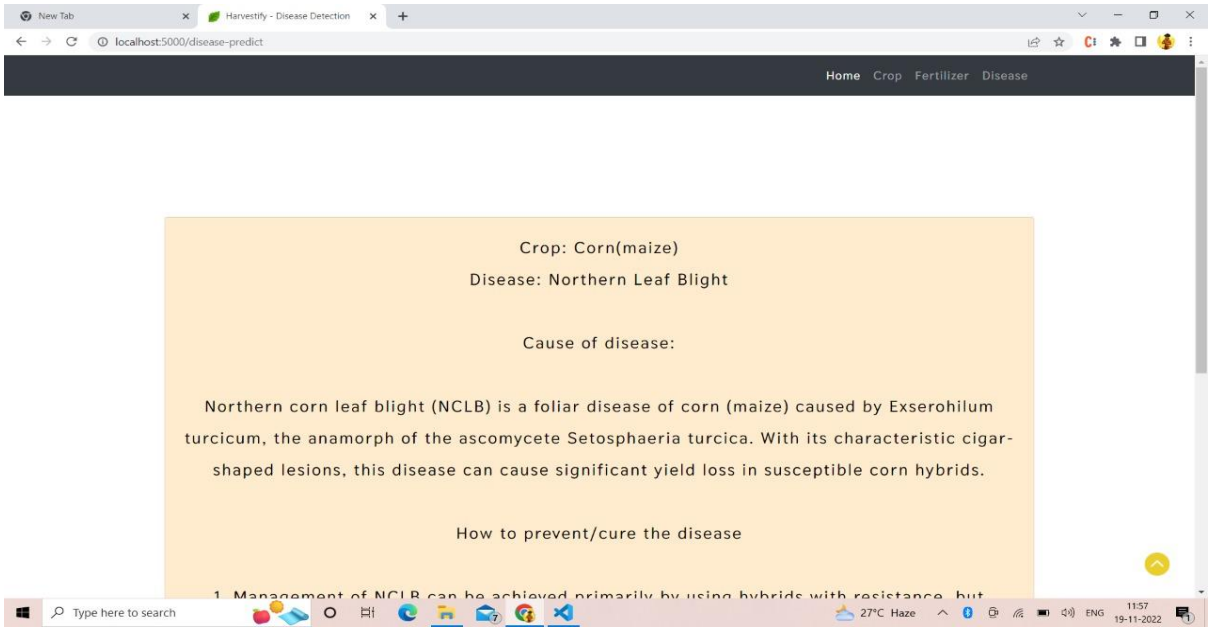
PREDICTED OUTCOME :

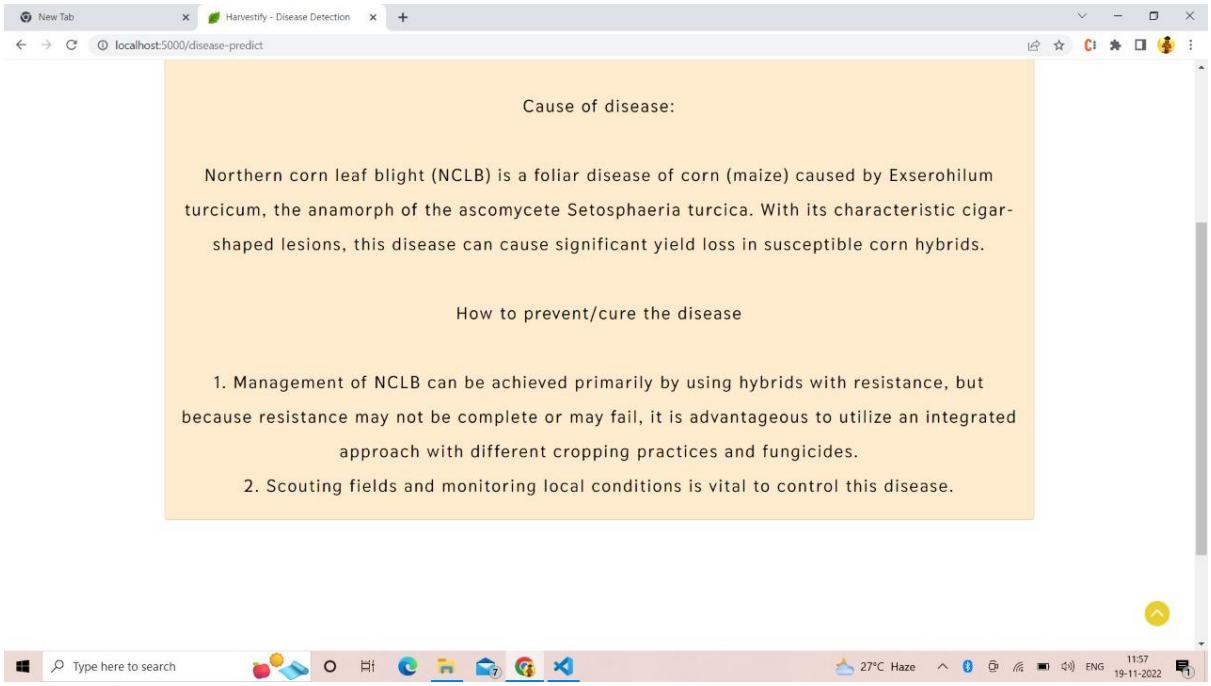


UPLOADING LEAF IMAGE:



PREDICTED OUTCOME:





10. ADVANTAGES &DISADVANTAGES:

ADVANTAGES:

- The suggested model yields extremely high classification accuracy
- It can train and test on very large datasets.
- It can resize very high-quality images within itself.

DISADVANTAGES:

- The proposed model is computationally expensive to train and test.
- The neural network architecture used in this project work is highly complex.

11. CONCLUSION:

The model here involves classifying images from datasets of fruits and vegetables. The number of epochs was increased to boost categorization accuracy. Different classification accuracies are obtained for different batch sizes. The accuracies are increased by adding more convolution layers. The accuracy of classification is also increased by adjusting the number of dense layers. The accuracies are different while varying the size of the train and test datasets.

12. FUTURE SCOPE :

The model that is being provided in this project work can be expanded to recognise images. Using python to exe software, the complete model may be turned into application software. With the aid of the OpenCV Python package, real-time image categorization, picture recognition, and video processing are all made feasible. This project's work can be expanded to include security applications including face, iris, and figure print recognition.

13. APPENDIX

13.1 SAMPLE SOURCE CODE:-

Index.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">Harvestify</h3>
                                <h4 class="text-wh mx-auto my-4"><b>Get
informed decisions about your farming strategy.</b></h4>
                                <br>
                                <h4 class="text-wh mx-auto my-4"><strong> Here
are some questions we'll answer</strong></h4>
                                <p class="text-li mx-auto mt-2">
                                    1. What crop to plant here? <br>
                                    2. What fertilizer to use? <br>
                                    3. Which disease do your crop have?<br>
                                    4. How to cure the disease?</p>
                            </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/s3.jpg') }}"
class="img-fluid" alt="" />
                <a href="{{ url_for('crop_recommend') }}">
                    <div class="blog-info">

                        <h4>Crop</h4>

                        <p class="mt-2"> Recommendation about the type of
crops to be cultivated which is best suited
                            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">
                        <h4>Fertilizer</h4>
                        <p class="mt-2">Recommendation about the 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">
                        <h4>Crop Disease</h4>
                        <p class="mt-2">Predicting the name and causes of crop
disease and suggestions to cure it</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 %}
```

Crop.html:

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

<style>

    html body {
      background-color: rgb(206, 206, 228);


  }
```

```
</style>
<!--Form Section-->
<br /><br />
<h2 style="text-align: center; margin: 0px; color: black">
  <b>Find out the most suitable crop to grow in your farm</b>
</h2>
<br />

<div
  style="
    width: 350px;
    height: 50rem;
    margin: 0px auto;
    color: black;
    border-radius: 25px;
    padding: 10px 10px;
  "
>
  <form method="POST" action="{{ url_for('crop_prediction') }}">
    <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="number"
        step="0.01"
        class="form-control"
        id="ph"
        name="ph"
        placeholder="Enter the value"
        style="font-weight: bold"
        required
      />
    </div>
    <div class="form-group">
      <label for="Rainfall" style="font-size: 17px"><b>Rainfall (in
mm)</b></label>
      <input
        type="number"
        step="0.01"
        class="form-control"
        id="Rainfall"
        name="rainfall"
        placeholder="Enter the value"
        style="font-weight: bold"
        required
      />
    </div>
    <div class="form-group">
      <label for="State" style="font-size: 17px "><b>State</b></label>
      <select
        onchange="print_city('state', this.selectedIndex);"
        id="sts"
        name="stt"
        class="form-control"
        style="font-weight: bold; color: black;"
```

```
        required
      ></select>
      <br />
      <label for="City" style="font-size: 17px"><b>City</b></label>
      <select
        id="state"
        class="form-control"
        name="city"
        style="font-weight: bold; color: black;"
        required
      ></select>
      <script language="javascript">
        print_state("sts");
      </script>
    </div>

    <div class="d-flex justify-content-center">
      <button
        type="submit"
        class="btn btn-info"
        style="color: black; font-weight: bold; width: 130px; height:50px;
border-radius:12px; font-size: 21px;"
      >
        Predict
      </button>
    </div>
  </form>
</div>



<!-- Form section -->

{% endblock %}
```

Crop-Results.html:

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

<style>
  .mt-0 {
  margin-top: 50 !important;
}
</style>


<div class="container py-2 mx-auto my-50 h-10 " style="margin: 12rem;">
  <div class="row">
```

```
      <div class="col-sm py-2 py-md-3">
          <div class="card card-body" style="justify-content: center;">
               <h1 class="text-center"><b>You should grow <i>{{ prediction }}
</i>in your farm</b></h1>
          </div>
      </div>
  </div>
</div>
{% endblock %}
```

Disease.html:

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


<style>
  html body {
    background-color: rgb(206, 206, 228);
  }
</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 %}
```

Disease-result.html:

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

Fertilizer.html:

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

<style>
  html body {
    background-color: rgb(206, 206, 228);
  }
</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="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: black;
          font-weight: bold;
          width: 130px;
          height: 50px;
          border-radius: 12px;
          font-size: 21px;
        "
      >
        Predict
      </button>
    </div>
  </form>
</div>
{% endblock %}
```

Fertilizer-result.html:

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

Layout.html:

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

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

    <!-- for-mobile-apps -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="utf-8">
```

```
    <meta name="keywords" content="Agro Harvest Responsive web template,
Bootstrap Web Templates, Flat Web Templates, Android Compatible web template,
Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG,
SonyEricsson, Motorola web design" />

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

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

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

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

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

            html {
                font-size: 1rem;
            }

            h1 {
                font-size: 1.2rem;
            }

            h2 {
                font-size: 1.1rem;
            }

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

```
            }

            h1 {
                font-size: 1.3rem;
            }

            h2 {
                font-size: 1.2rem;
            }
        }

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

            h1 {
                font-size: 1.5rem;
            }

            h2 {
                font-size: 1.4rem;
            }
        }

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

            h1 {
                font-size: 1.7rem;
            }

            h2 {
                font-size: 1.6rem;
            }
        }

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

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

```
        }

    </script>

    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
        crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js
"
        integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
        crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
        integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
        crossorigin="anonymous"></script>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
        crossorigin="anonymous"></script>
    </body>
    <!-- css files -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css
"
        integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <link href="{{ url_for('static', filename='css/bootstrap.css') }}"
rel='stylesheet' type='text/css' />
    <!-- bootstrap css -->
    <link href="{{ url_for('static', filename='css/style.css') }}"
rel='stylesheet' type='text/css' />
    <!-- custom css -->
    <link href="{{ url_for('static', filename='css/font-awesome.min.css') }}"
rel="stylesheet"><!-- fontawesome css -->
    <!-- //css files -->
    <!-- <link rel="icon" type="image/png" href="{{ url_for('static',
filename='images/favicon.png?') }}"> -->
```

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


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

    <style>
        header {
            background-color: rgba(30, 30, 30, 1);
            margin-top: 0rem;
            display: block;

        }
    </style>
</head>

<body>

    <!-- Navigation -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark static-top"
style="background-color: #1C00ff00;">
        <div class="container">
            <a class="navbar-brand" href="{{ url_for('home') }}">
                <img src="{{ url_for('static', filename='images/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') }}">
                    Harvestify
                </a>
            </h2>
            <!-- //logo -->
            <!-- address -->
            <div class="contact-left-footer mt-4">




                <!-- <a href="community.html">Community</a> -->
                </p>
            </div>
            <div class="w3l-copy text-center">
                <p class="text-da">An Environmental Intelligence Startup<br>
</p>
            </div>
            <p class="homelogo">
            <p>Made with ❤ by Atharva</p>
            <p>&copy; Copyright 2021 Atharva Ingle</p>

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

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

</html>
```

Try again.html:

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

<div class="container py-2 mx-auto my-50 h-10 text-center" 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:blanchedalmond">
        <h1 class="text-center" style="color: black; font-size:
20px;"><b>Sorry we couldn't process your request
            currently. <br> Please try again</b></h1>


        <a href="{{ url_for('home') }}">
          <button type="submit" class="btn btn-info text-center" style="
        color: black;
        font-weight: bold;
        margin: 1rem;">
            Try again
          </button>
        </a>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

Fertilizer.csv:

	A	B	C	D	E	F	G	H
1		Crop	N	P	K	pH	soil_moisture	
2	0	rice	80	40	40	5.5	30	
3	3	maize	80	40	20	5.5	50	
4	5	chickpea	40	60	80	5.5	60	
5	12	kidneybeans	20	60	20	5.5	45	
6	13	pigeonpeas	20	60	20	5.5	45	
7	14	mothbeans	20	40	20	5.5	30	
8	15	mungbean	20	40	20	5.5	80	
9	18	blackgram	40	60	20	5	60	
10	24	lentil	20	60	20	5.5	90	
11	60	pomegranate	20	10	40	5.5	30	
12	61	banana	100	75	50	6.5	40	
13	62	mango	20	20	30	5	15	
14	63	grapes	20	125	200	4	60	
15	66	watermelon	100	10	50	5.5	70	
16	67	muskmelon	100	10	50	5.5	30	
17	69	apple	20	125	200	6.5	50	
18	74	orange	20	10	10	4	60	
19	75	papaya	50	50	50	6	20	
20	88	coconut	20	10	30	5	45	
21	93	cotton	120	40	20	5.5	70	
22	94	jute	80	40	40	5.5	20	
23	95	coffee	100	20	30	5.5	20	
24								

App.py:

```
# 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
```

```python
#
================================================================================
================

# 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('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':
        file = request.files.get('file')

        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)
```

Requirement.txt:

```
numpy
pandas
Flask
scikit-learn
https://download.pytorch.org/whl/cpu/torch-1.7.0%2Bcpu-cp36-cp36m-
linux_x86_64.whl
https://download.pytorch.org/whl/cpu/torchvision-0.8.1%2Bcpu-cp36-cp36m-
linux_x86_64.whl
requests
Pillow
gunicorn == 20.0.4
```

13.2 GITHUB LINK :

https://github.com/IBM-EPBL/IBM-Project-21738-1659789794.git

13.3 PROJECT DEMO LINK:

Video Link:
https://drive.google.com/file/d/1h7CRTlk4RtGqcYpvB51hMjM4SjMmjlg1/view?usp=share_link