FERTILIZERS RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

NALAIYA TIRAN PROJECT BASED LEARNING ON PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTERPRENEURSHIP

A PROJECT REPORT

SUBASHINI SL 611819106052

KAVILAIYA V 611819106017

YAMINI R 611819106064

KOMATHI M 611819106019

TEAM ID : PNT2022TMID40966

INDUSRTY MENTORS NAME : DURGA PRASAD

FACULTY MENTORS NAME : L. PRAKASAM

EVALUATOR NAME : S.SHANMUGAM

P.S.V. COLLEGE OF ENGINEERING AND TECHNOLOGY

(An ISO 9001:2015 Certified Institution)

(Accredited by NAAC with 'A' Grade)

KRISHNAGIRI-635108

NOVEMBER 2022

ANNA UNIVERSITY: CHENNAI 600 025

PROJECT REPORT FORMAT

1. INTRODUCTION

a. Project Overview

b. Purpose

2. LITERATURE SURVEY

a. Existing problem

b. References

c. Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas

b. Ideation & Brainstorming

c. Proposed Solution

d. Problem Solution fit

4. REQUIREMENT ANALYSIS

a. Functional requirement

b. Non-Functional requirements

5. PROJECT DESIGN

a. Data Flow Diagrams

b. Solution & Technical Architecture

c. User Stories

6. PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation

b. Sprint Delivery Schedule

c. Reports from JIRA

7. CODING & SOLUTIONING

(Explain the features added in the project along with code)

a. Feature 1

b. Feature 2

c. Database Schema (if Applicable)

8. TESTING

a. Test Cases

b. User Acceptance Testing

9. RESULTS

a. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

a. Source Code

b. GitHub & Project Demo Link

1.INTRODUCTION:

Agriculture is the most important sector in today's life. 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. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality. 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.

Project overview:

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 changes in cultivation method and inadequate plant protection techniques and suggest all the precautions that can be taken for those diseases.

Purpose:

To Detect and recognize the plant diseases and to recommend fertilizer, it is necessary to identify the diseases and to recommend to get different and useful features needed for the purpose of analyzing later.

To provide symptoms in identifying the disease at its earliest. Hence the authors proposed and implemented new fertilizers Recommendation System for Crop Disease Prediction.

LITREATURE SURVEY

Literature Review:

[1] 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. A**dvantages :** The prediction and diagnosing of leaf diseases are depending on the segmenta- tion such as segmenting the healthy tissues from diseased tissues of leaves.
Disadvantages : This further research is implementing the proposed algorithm with the ex- isting public datasets. Also, various segmentation algorithms can be implemented to improve accuracy. The proposed algorithm can be modified further to identify the disease that affects the various plant organs such as stems and fruits.
[2] Detection of Leaf Diseases and Classification using Digital Image Processing International .

[3] The current work examines and describes image processing strategies for identifying plant diseases in numerous plant species. BPNN, SVM, K-means clustering, and SGDM are the most common approaches used to identify plant diseases.
Disadvantages : Some of the issues in these approaches include the impact of background data on the final picture, optimization of the methodology for a specific plant leaf disease, and automation of the technique for continuous automated monitoring of plant leaf diseases in real-world field circumstances.

Existing Problem

● Adequate mineral nutrition is central to crop production. However, it can also exert considerable Influence on disease development. Fertilizer application can increase or decrease development of diseases caused by different pathogens,

and the mechanisms responsible are complex, including effects of nutrients on plant growth, plant resistance mechanisms and direct effects on the

pathogen. The effects of mineral nutrition on plant disease and the mechanisms responsible for those effects have been dealt with comprehensively elsewhere. In India, around 40% of land is kept and grown using reliable irrigation technologies, while the rest relies on the monsoon environment for water. Irrigation decreases reliance on the monsoon, increases food security, and boosts agricultural production.

● Most research articles use humidity, moisture, and temperature sensors near the plant's root, with an external device handling all of the data provided by the sensors and transmitting it directly to an Android application. It was created to measure the approximate values of temperature, humidity and moisture sensors that were programmed into a microcontroller to manage the amount of water.

Reference

[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.

Problem Statement Definition :

Mr.Narasimma Rao is a 65 years old man. He had a own farming land and do Agriculture for past 30 Years , In this 30 Years he Faced a problem in Choosing Fertilizers and Controlling of Plant Disease.

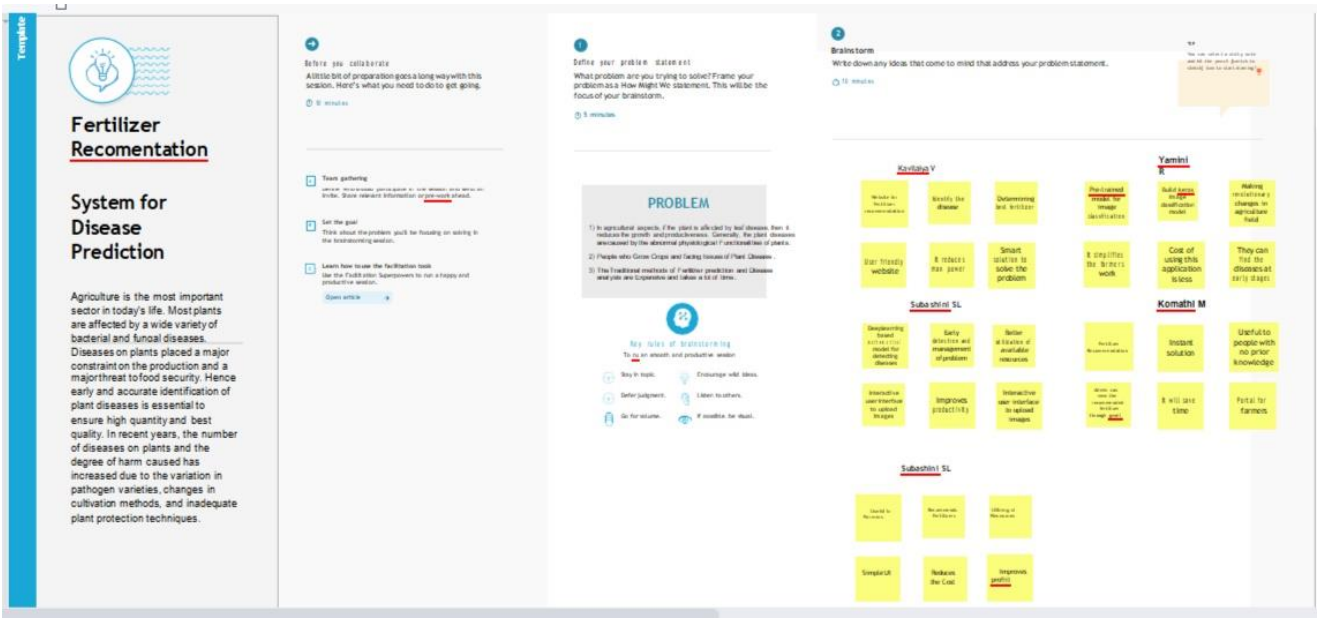
 Narasimma Rao wants to know the better recommendation for fertilizers for plants with the disease.
 He has faced huge losses for a long time.
 This problem is usually faced by most farmers.
 Mr. Narasimma Rao needs to know the result immediately.

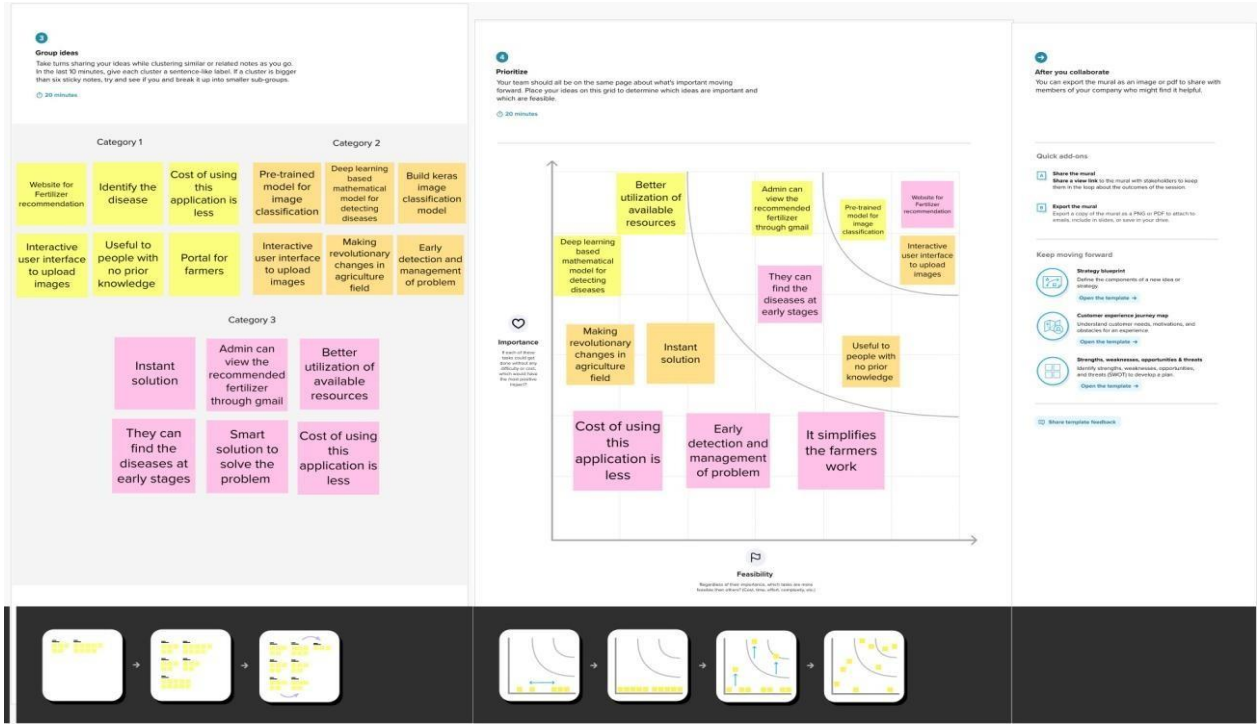
Who does the problem affect?	Persons who do Agriculture
What are the boundaries of the problem?	People who Grow Crops and facing Issues of Plant Disease
What is the issue?	In agricultural aspects, if the plant is affected by leaf disease, then it reduces the growth and productiveness. Generally, the plant diseases are caused by the abnormal physiological functionalities of plants.

When does the issue occur?	During the development of the crops as they will be affected by various diseases.
Where does the issue occur?	The issue occurs in agriculture practicing areas, particularly in rural regions.
Why is it important that we fix the problem?	It is required for the growth of better quality food products. It is important to maximise the crop yield.
What solution to solve this issue?	An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant.
What methodology used to solve the issue?	Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

IDEATION & PROPOSED SOLUTION

Ideation & Brainstorming :





Empathy map canvas:

Fertilizers Recommendation System For Disease Prediction

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.

Proposed Solution:

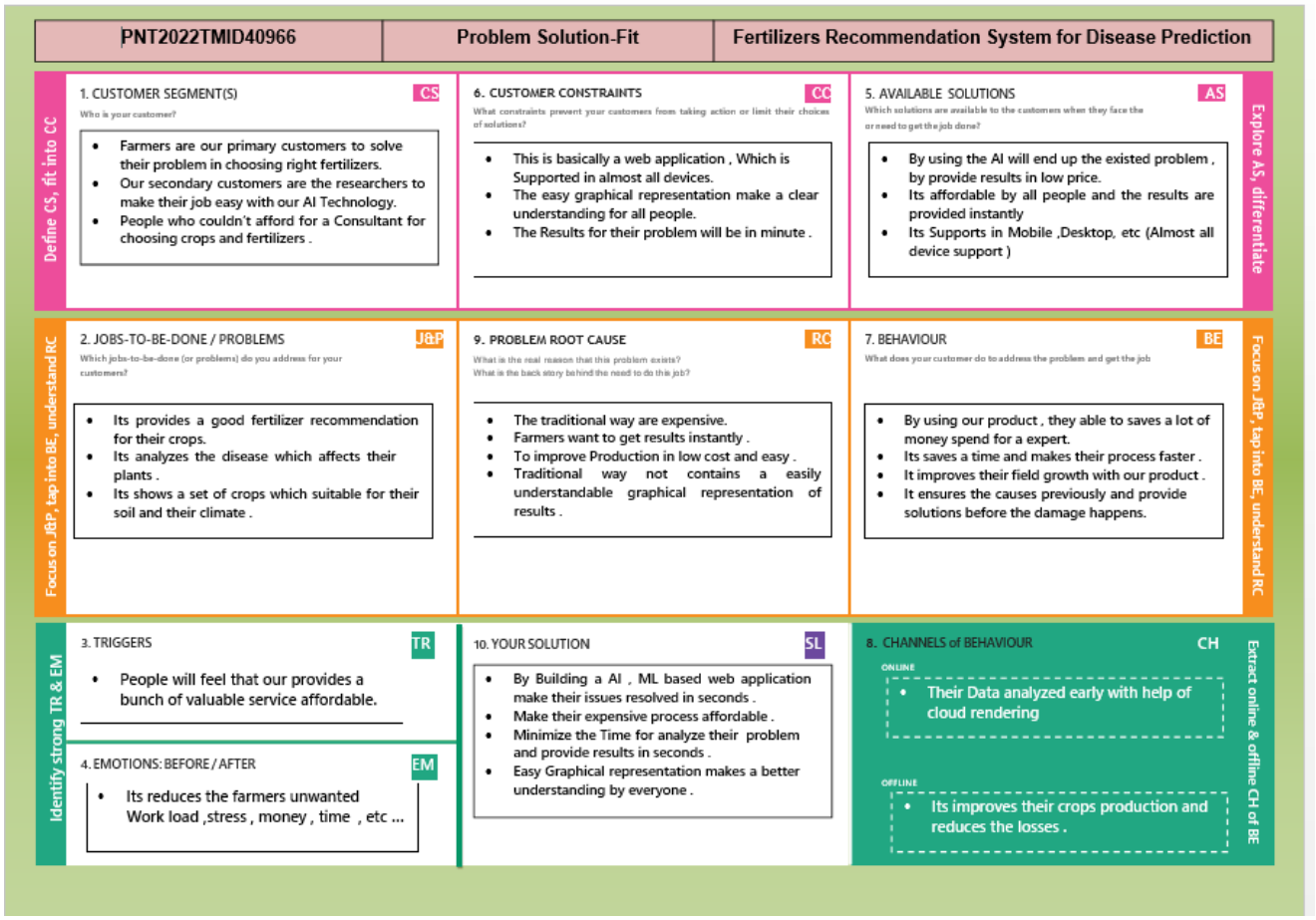
- The idea of the proposed solution uses Deep learning and Machine algorithm to classify leaves and identify the diseases and siggest the fertilizers. The deep learning process includes the MobileNetV2 and VGG19 training Models.

- Based on the leaf disease detected , the model recommendation for fertilizers for the prevention. The farmers and researchers are the endusers get benefied by the system.

- More accurate in others. The system is more robust corporating more image data sets with wider variations. This system also estimates the probability of infected plant.

- Plant growth can be enhanced. Ensure plants are getting supplied with every nutrient they need also and multiple cross in grow in every yields for every season. It also helps people's nutritional needs.

Problem Solution Fit



REQUIREMENT ANALYSIS :

Functional Requirements

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Specific characteristics	It identifies the diseases especially rice bran diseases
FR-4	Functions	The proposed methods uses the SVM to classify tree leaves, identify the diseases and suggest the fertilizer.
FR-5	Fault tolerance	This study enables a possible prediction of crop yield from the historic data collected and offers a suggestion to farmers.
FR-6	Analyze	It helps us to classify the data based on the diseases, and data extracted from the classifier is used to predict soil and crop.

Non Functional Requirements

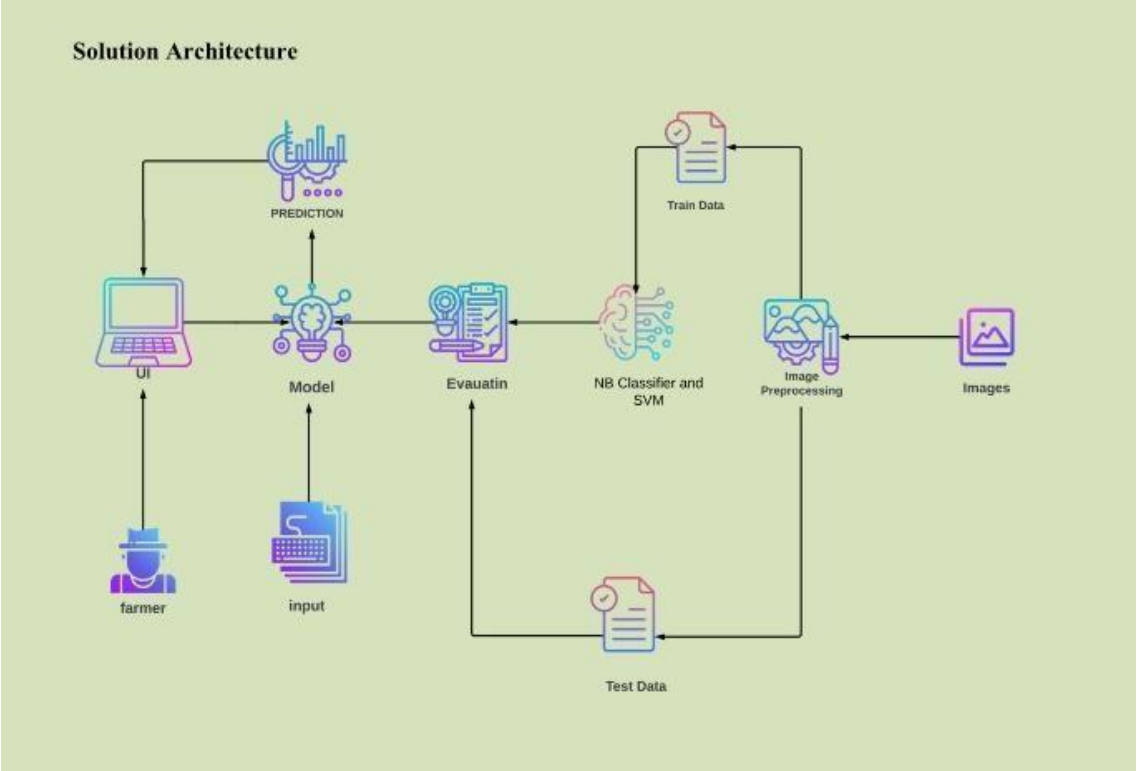
Non-functional Requirements:

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

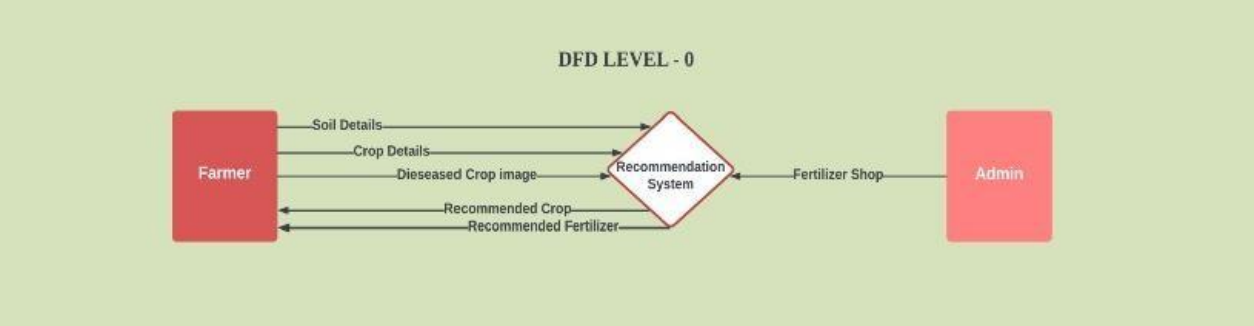
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Crop and fertilizer recommendation system help the farmer to identify the diseases.
NFR-2	**Security**	The proposed method combines two major aspects in farming , pest identification and insecticide recommendation.
NFR-3	**Reliability**	It is easy use so that health issues can be avoided.
NFR-4	**Performance**	Precision fertilizer and precision crops is mostly used. They used to predict the crop in artificial intelligence.
NFR-5	**Availability**	reduces the losses as ammonia , nitrate leaching, apply the right rate, apply accurately.
NFR-6	**Scalability**	If the soil is not replenised with nutrients through fertilizing ,crop yields will deteriorate over time.

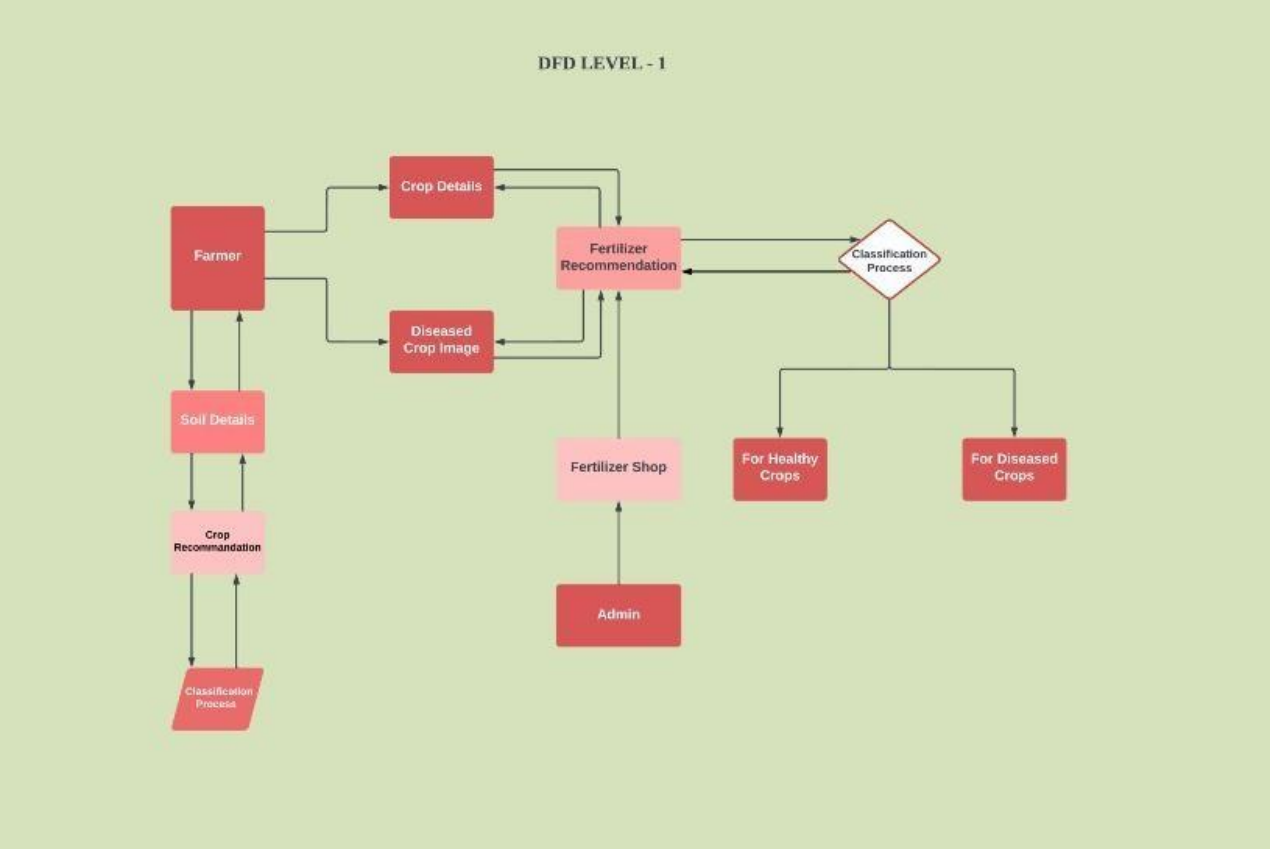
PROJECT DESIGN :

Solution & Technical Architecture



Data Flow Diagrams





User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile ssuser)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	**Sprint-1**
	Login	USN-2	As a user, I can log into the application by entering email & password	I can login using my E-mail ID accounts or user credentials	High	**Sprint-1**
	Dashboard	USN-3	As a user, I can view the page of the application where i can upload my images and the fertilizer should be recommended	I can access my account/ dashboard	High	**Sprint-2**
Customer (Webuser)	Registration	USN-4	As a user, I can login to web dashboard just Like website dashboard	I can register using my username and password	High	**Sprint-3**
	Login	USN-5	As a user, I can login to my web dashboard with the login credentials	I can login using my User credentials	High	**Sprint-3**
	Dashboard	USN-6	As a user, I can view the web application where i can upload my images and thefertilizer should be recommended	I can access my account/ dashboard	High	**Sprint-4**
		USN-7	As a user, the fertilizer recommended to me should be of higher accuracy	I can access my accou nt/ dashb oard	High	**Sprint-4**
Administrator	Login	USN-8	As a admin, I can login to the website using my login credentials	I can login to the website using my login credentials	High	**Sprint-5**
	Dashboard	**USN-9**	**As a admin, I can view the dashboard of the application**	**I can access my dashboard**	**High**	**Sprint-5**

PROJECT PLANNING & SCHEDULING:

Sprint Planning and Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

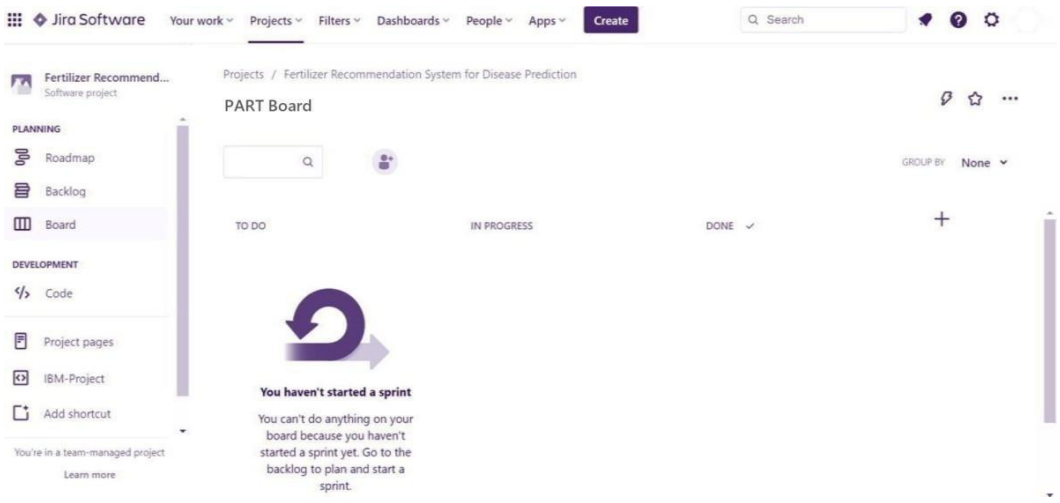
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points (Total)	Priority	Team Members
Sprint-1	Model Creation and Training (Fruits)		Create a model which can classify diseased fruit plants from given images. I also need to test the model and deploy it on IBM Cloud	8	High	Subashini SL
	Model Creation and Training (Vegetables)		Create a model which can classify diseased vegetable plants from given images	2	High	Subashini SL
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	Subashini SL
	Registration	USN-1	As a user, I can register by entering my email, password, and confirming my password or via OAuth API	3	Medium	Kavilaiya V
	Upload page	USN-2	As a user, I will be redirected to a page where I can upload my pictures of crops	4	High	Kavilaiya 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	Kavilaiya V
	Base Flask App		A base Flask web app must be created as an interface for the ML model	2	High	Komathi Kavilaiya
Sprint-3	Login	USN-4	As a user/admin/shopkeeper, I can log into the application by entering email & password	2	High	Komathi Subashini
	User Dashboard	USN-5	As a user, I can view the previous results and history	3	Medium	Komathi Kavilaiya
	Integration		Integrate Flask, CNN model with Cloudant DB	5	Medium	Yamini R
	Containerization		Containerize Flask app using Docker	2	Low	Yamini R

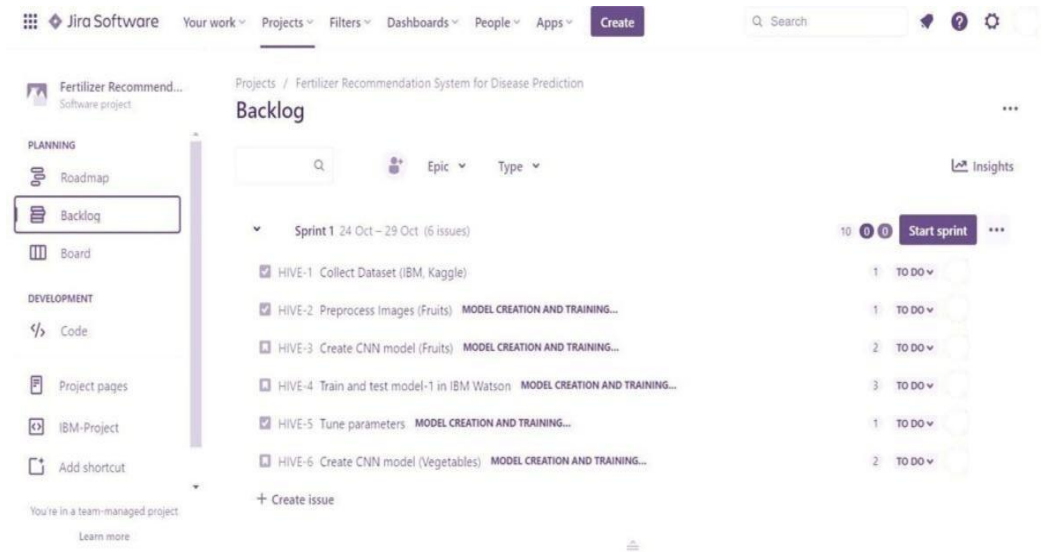
Sprint-4	Dashboard (Admin)	USN-6	As an admin, I can view other user details and uploads for other purposes	2	Medium	Yamini R
	Dashboard (Shopkeeper)	USN-7	As a shopkeeper, I can enter fertilizer products and then update the details if any	2	Low	Yamini R
	Containerization		Create and deploy Helm charts using Docker Image made before	2	Low	Komathi M

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

Reports from JIRA





CODING & SOLUTIONING

Feature 1[Model Building]:

1. Import The Libraries

Import the libraries that are required to initialize the neural network layer, and create and add different layers to the neural network model.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
```

2. Initializing The Model

Keras has 2 ways to define a neural network.

Sequential

- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method.

Now, will initialize our model.Initialize the neural network layer by creating a reference/object to the Sequential class.

```
model=Sequential()
```

3. ADD CNNLayers

We will be adding three layers for CNN

- Convolution layer
- Pooling layer
- Flattening layer

Add Convolution Layer

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution2D class is used. It takes a number of feature detectors, feature detector size, expected input shape of the image, and activation function as arguments. This larger applies feature detectors on the input image and returns a feature map (features from the image).

Activation Function: These are the functions that help us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.

```
model.add(Convolution2D(32,(3,3),input_shape = (128,128,3),activation = 'relu'))
```

Add the pooling layer

Max Pooling selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

After the convolution layer, a pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. Efficient size of the pooling matrix is (2,2). It returns the

pooled feature maps. (Note: Any number of convolution layers, pooling and dropout layers can beadded)

```
model.add(MaxPooling2D(pool_size = (2,2)))
```

Add the flatten layer

The flatten layer is used to convert n-dimensional arrays to 1-dimensional arrays. This 1D array will be given as input to ANN layers.

```
model.add(Flatten())
```

1. Add Dense Layers

Now, let's add Dense Layers to know more about dense layers click below

Dense layers

The name suggests that layers are fully connected (dense) by the neurons in a network layer. Each neuron in a layer receives input from all the neurons present in the previous layer. Dense is used to add thelayers.

Adding Hidden layers

This step is to add a dense layer (hidden layer). We flatten the feature map and convert it into a vector or single dimensional array in the Flatten layer. This vector array is fed it as an input to the neural network and applies an activation function, such as sigmoid or other, and returns the output.

- init is the weight initialization; function which sets all the weights and biases of a network to values suitable as a starting point for training.
- units/ output_dim, which denote is the number of neurons in the hidden layer.

- The activation function basically decides to deactivate neurons or activate them to get the desired output. It also performs a nonlinear transformation on the input to get better results on a complex neural network.
- You can add many hidden layers, in our project we are added two hidden layers. The 1st hidden layer with 40 neurons and 2nd hidden layer with 20neurons.

Adding the output layer

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function, and weight initializer as the arguments. We use the add () method to add dense layers. the output dimensions here is 6

```
model.add(Dense(output_dim = 40 ,init = 'uniform',activation = 'relu'))
model.add(Dense(output_dim = 20 ,init = 'random_uniform',activation = 'relu'))
model.add(Dense(output_dim = 6,activation = 'softmax',init ='random_uniform'))
```

1. Train And Save

The Model

Compile the model

After adding all the required layers, the model is to be compiled. For this step, loss function, optimizer and metrics for evaluation can be passed as arguments.

```
model.compile(loss = 'categorical_crossentropy',optimizer = "adam",metrics = ["accuracy"])
```

Fit and save the model

Fit the neural network model with the train and test set, number of epochs and validation steps. Steps per epoch is determined by number of training images/ batch size, for validation steps number of validation images/ batch size.

```
model.fit_generator(x_train, steps_per_epoch = 168,epochs = 3,validation_data = x_test,validation_steps = 52)
```

Accuracy, Loss: Loss value implies how poorly or well a model behaves after each iteration of optimization. An accuracy metric is used to measure the algorithm's performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage.

The weights are to be saved for future use. The weights are saved in as .h5 file using save().

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

model.summary() can be used to see all parameters and shapes in each layer in our models.

2.Test The Model

The model is to be tested with different images to know if it is working correctly.

Import the packages and load the saved model

Import the required libraries

```
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
```

Initially, we will be loading the fruit model. You can test it with the vegetable model in a similar way.

```
model = load_model("fruit.h5")
```

Load the test image, pre-process it and predict

Pre-processing the image includes converting the image to array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.

```
img = image.load_img('apple_healthy.JPG',target_size = (128,128))
```

```
x  = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
```

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

```
pred
```

```
[1]
```

The predicted class is 1.

Feature 2[Python code]

Build Python Code:

After the model is built, we will be integrating it into a web application so that normal users can also use it. The user needs to browse the images to detect the disease.

Activity 1: Build a flask application

Step 1: Load the required packages

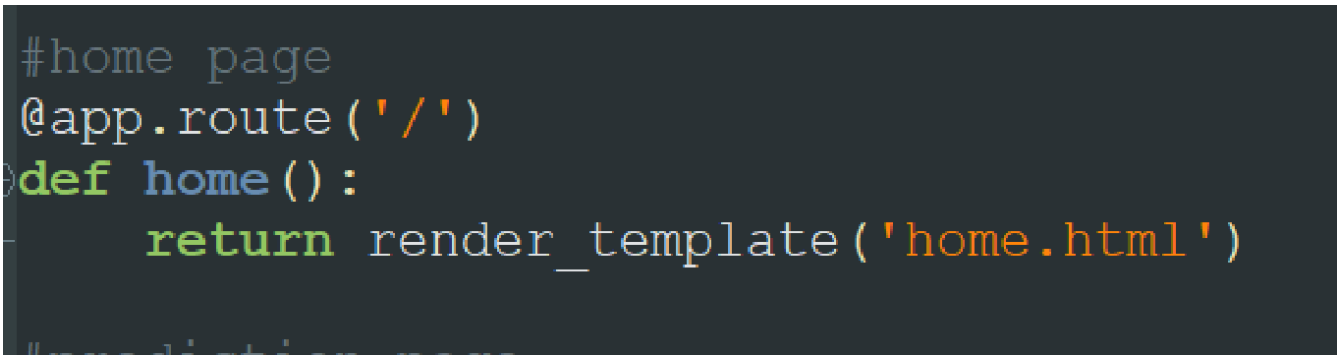
```
import requests
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import numpy as np
import pandas as pd
import tensorflow as tf
from flask import Flask, request, render_template, redirect, url_for
import os
from werkzeug.utils import secure_filename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

Step 2: Initialize the flask app and load the model

An instance of Flask is created and the model is loaded using load_model from Keras.

```
app = Flask(__name__)

#load both the vegetable and fruit models
model = load_model("vegetable.h5")
modell=load_model("fruit.h5")
```

Step 3: Configure the home page



Step 4: Pre-process the frame and run

Pre-process the captured frame and give it to the model for prediction. Based on the prediction the output text is generated and sent to the HTML to display. We will be loading the precautions for fruits and vegetables excel file to get the precautions based on the output and return it to the HTML Page.

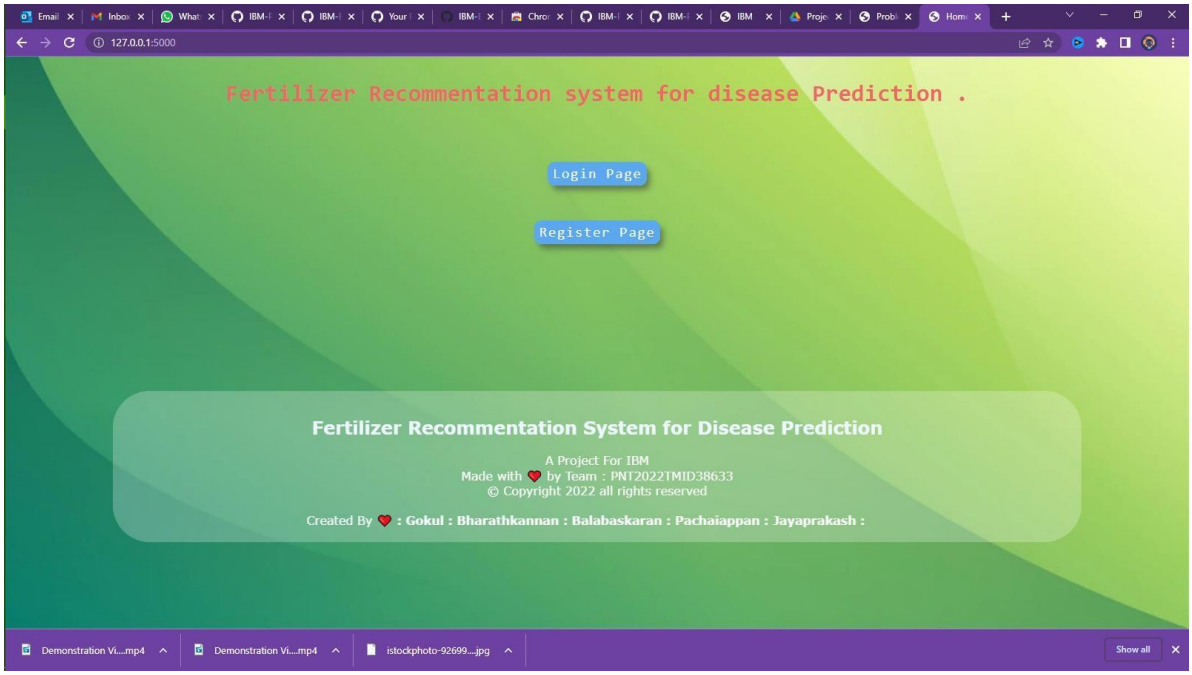
```python
#prediction page
@app.route('/prediction')
def prediction():
    return render_template('predict.html')

@app.route('/predict',methods=['POST'])
def predict():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']
        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(128, 128))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        plant=request.form['plant']
        print(plant)
        if(plant=="vegetable"):
            preds = model.predict_classes(x)
            print(preds)
            df=pd.read_excel('precautions - veg.xlsx')
            print(df.iloc[preds[0]]['caution'])
        else:
            preds = model1.predict_classes(x)

            df=pd.read_excel('precautions - fruits.xlsx')
            print(df.iloc[preds[0]]['caution'])
        return df.iloc[preds[0]]['caution']
```

Run the flask application using the run method. By default, the flask runs on 5000 port. If the port is to be changed, an argument can be passed and the port can be modified.

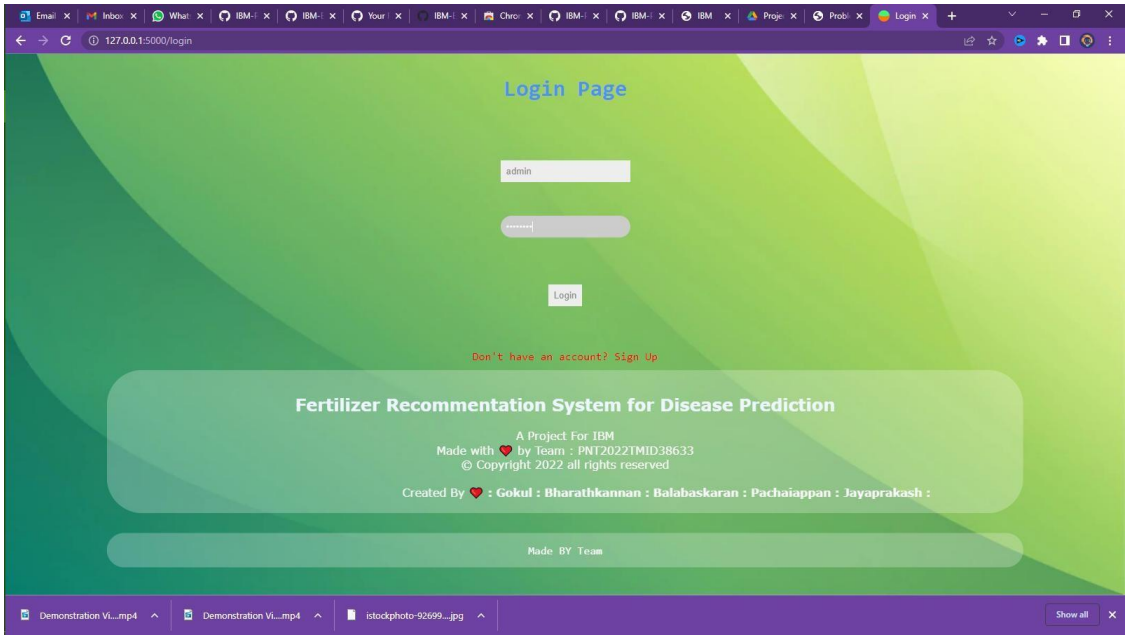
```python
if __name__ == "__main__":
    app.run(debug=False)
```

TESTING

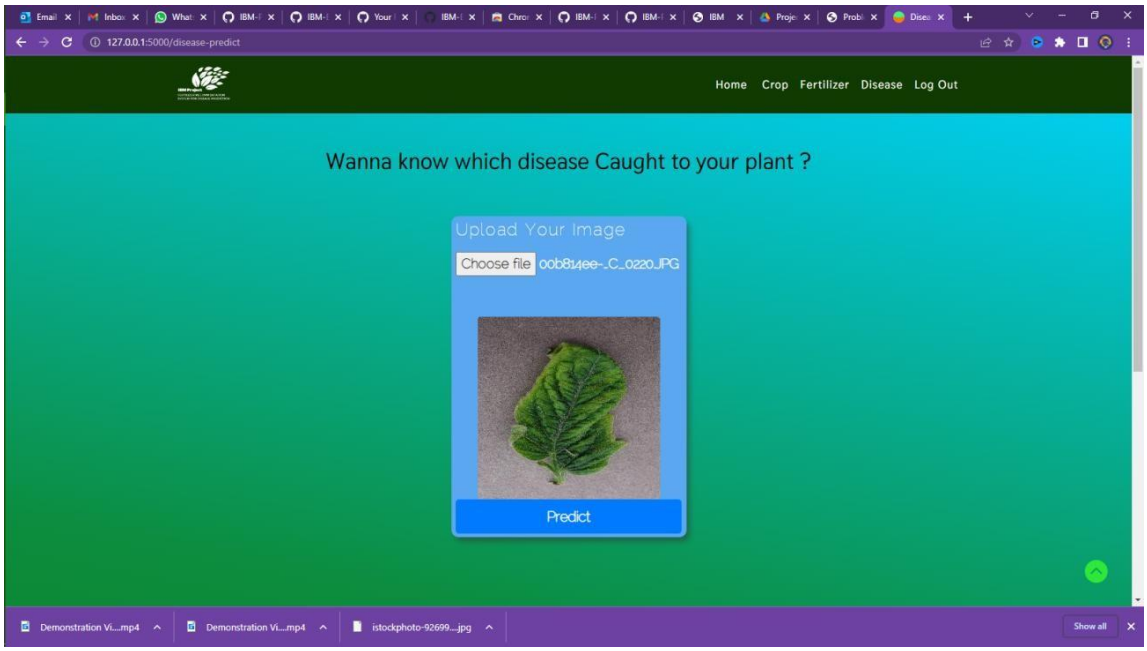
TEST CASES 1



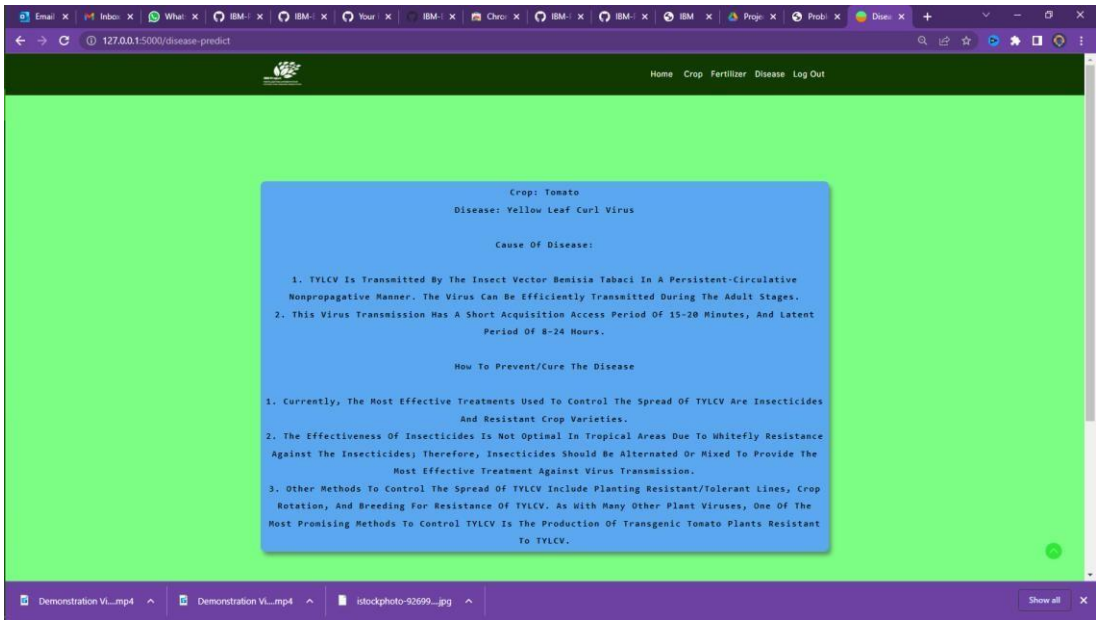
TEST CASES 2



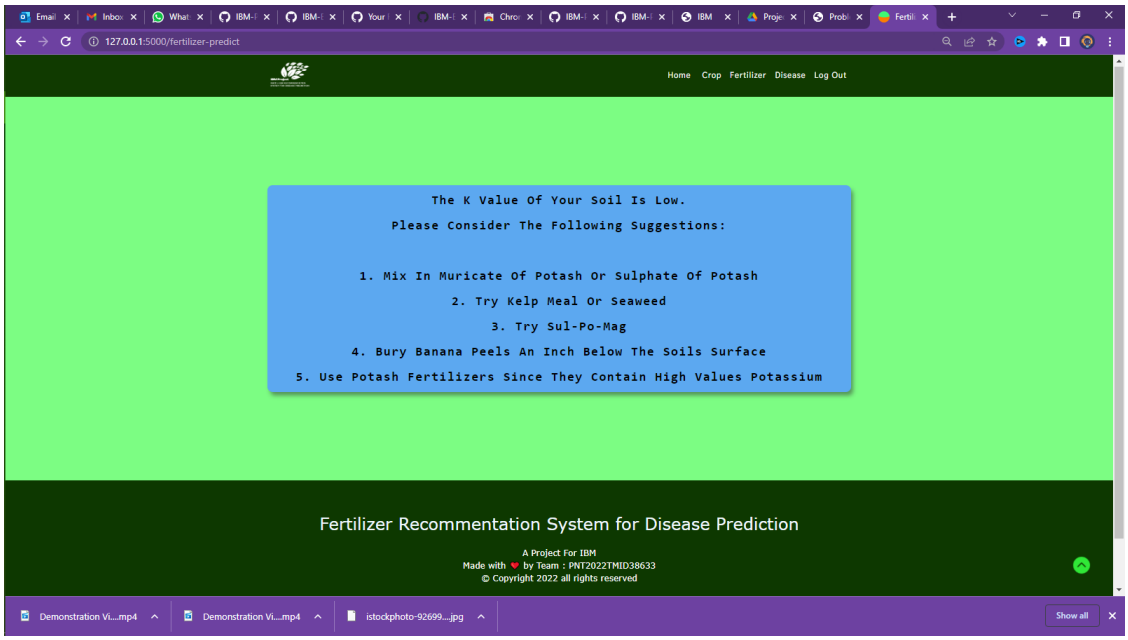
TEST CASES 3



TEST CASES 4



TEST CASES 5


127.0.0.1:5000/fertilizer-predict
Home Crop Fertilizer Disease Log Out
The K Value Of Your Soil Is Low.
Please Consider The Following Suggestions:
1. Mix In Muricate Of Potash Or Sulphate Of Potash
2. Try Kelp Meal Or Seaweed
3. Try Sul-Po-Mag
4. Bury Banana Peels An Inch Below The Soils Surface
5. Use Potash Fertilizers Since They Contain High Values Potassium
Fertilizer Recommentation System for Disease Prediction
A Project For IBM
Made with by Team : PNT2022TMID38633
© Copyright 2022 all rights reserved
Demonstration Vi....mp4
Demonstration Vi....mp4
istockphoto-92699....jpg
Show all

User Acceptance Testing:

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Fertilizers Recommendation System for Disease Prediction project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	1	0	1
Duplicate	1	3	2	2	8
External	2	3	0	0	5
Fixed	4	4	4	4	16
Not Reproduced	0	0	0	1	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	7	10	7	7	31

3. Test Case Analysis

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

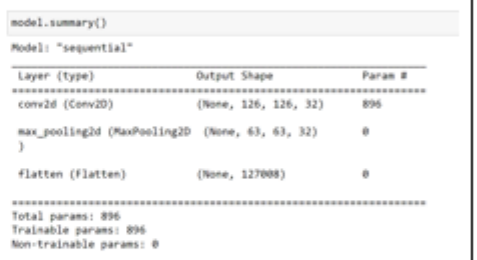
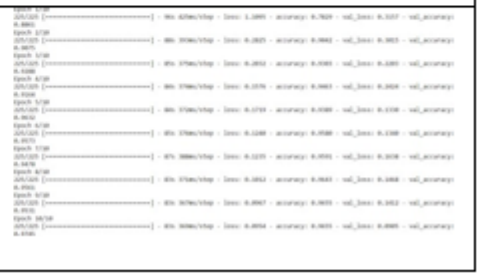
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	1	0	0	1
Client Application	1	0	0	1
Security	1	0	0	1
Outsource Shipping	1	0	0	1
Exception Reporting	1	0	0	1
Final Report Output	1	0	0	1
Version Control	1	0	0	1

RESULTS:

Performance Metrics:

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Total params: 896 Trainable params: 896 Non-trainable params: 0	
2.	Accuracy	Training Accuracy – 96.55 Validation Accuracy – 97.45	

Model Summary:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 flatten (Flatten)           (None, 127008)            0

=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

Accuracy:

```
Epoch 1/10
225/225 [==============================] - 96s 425ms/step - loss: 1.1095 - accuracy: 0.7829 - val_loss: 0.3157 - val_accuracy:
0.8861
Epoch 2/10
225/225 [==============================] - 88s 393ms/step - loss: 0.2825 - accuracy: 0.9042 - val_loss: 0.3015 - val_accuracy:
0.9075
Epoch 3/10
225/225 [==============================] - 85s 375ms/step - loss: 0.2032 - accuracy: 0.9303 - val_loss: 0.2203 - val_accuracy:
0.9288
Epoch 4/10
225/225 [==============================] - 84s 374ms/step - loss: 0.1576 - accuracy: 0.9463 - val_loss: 0.2424 - val_accuracy:
0.9164
Epoch 5/10
225/225 [==============================] - 84s 372ms/step - loss: 0.1719 - accuracy: 0.9389 - val_loss: 0.1330 - val_accuracy:
0.9632
Epoch 6/10
225/225 [==============================] - 85s 376ms/step - loss: 0.1240 - accuracy: 0.9580 - val_loss: 0.1340 - val_accuracy:
0.9573
Epoch 7/10
225/225 [==============================] - 87s 388ms/step - loss: 0.1235 - accuracy: 0.9591 - val_loss: 0.1638 - val_accuracy:
0.9478
Epoch 8/10
225/225 [==============================] - 83s 371ms/step - loss: 0.1012 - accuracy: 0.9643 - val_loss: 0.1468 - val_accuracy:
0.9561
Epoch 9/10
225/225 [==============================] - 83s 367ms/step - loss: 0.0967 - accuracy: 0.9655 - val_loss: 0.1412 - val_accuracy:
0.9531
Epoch 10/10
225/225 [==============================] - 83s 369ms/step - loss: 0.0954 - accuracy: 0.9655 - val_loss: 0.0905 - val_accuracy:
0.9745
```

ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

1.Fertilizers provide crops with nutrients like potassium, phosphorus, and nitrogen, which allow crops to grow bigger, faster, and to produce more food. Nitrogen in particular is an essential nutrient for the growth of every organism on Earth. Nitrogen is all around us and makes up about 78% of the air you breathe.

2.Sometimes plants need a quick fix to survive, in this type of cases fertilizers play a vital role to improve plants' health. plants need nutrients that can be absorbed quickly which is fulfilled by fertilizers. They are easily soluble and fastly absorbed by plants and as soon as possible it helps to regain and boost plant health.

DISADVANTAGES:

1.Fertilizers are man-made so they need production in factories which makes them costlier than naturally made manure. But it is important for plant nutrients so it is in demand and thus it has high value.

2.Fertilizers are used in moderate quantities if we use excessive fertilizers it surely damages the roots of plants and their tissues and thus plants can die. Fertilizers are used according to the need of the plant. Unnecessary use of them can affect the plant's health specially if plants have good fertile soil.

Conclusion :

- The core strategy of this project is to predict the crop based on the soil nutrient content and the location where the crop is growing. This system will help he farmers to choose the right crop for their land and to give the suitable amount of fertilizer to produce the maximum yield. The Support Vector Machine algorithm helps to predict the crop the precisely based on the pre-processed

crop data. This system will also help the new comers to choose the crop which will grow in their area and produce them a good profit. A decent amount of profit will attract morepeopletowardstheagriculture.

Future Scope :

- As of now we have just built the web application which apparently takes the input as an image and then predict the out in the near future we can develop an application which computer vision and AI techniques to predict the infection once you keep the camera near the plant or leaf this could make our project even more usable.
- This further research is implementingthe proposed algorithm with the existing public datasets. Also, various segmentation algorithms can be implemented to improve accuracy. The proposed algorithm can be modified further to identify the disease that affects the various plant organs such as vegetables and fruits.

Appendix :

Source code:

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

app = Flask(__name__)

#load both the vegetable and fruit models
model = load_model("vegetable.h5")
model1=load_model("fruit.h5")
#home page
@app.route('/')
def home():

    return render_template('home.html')

#prediction page
@app.route('/prediction')
def prediction():
return render_template('predict.html')

@app.route('/predict',methods=['POST'])
def predict():
if request.method == 'POST':

    # Get the file from post request
    f = request.files['image']


# Save the file to ./uploads
```

```
basepath = os.path.dirname(__file__)
file_path = os.path.join(
        basepath, 'uploads', secure_filename(f.filename))
f.save(file_path)
img = image.load_img(file_path, target_size=(128, 128))


x = image.img_to_array(img)
      x = np.expand_dims(x, axis=0)

plant=request.form['plant']
print(plant)

if(plant=="vegetable"):
        preds = model.predict(x)
        preds=np.argmax(preds)
        print(preds)
        df=pd.read_excel(r'precautions - veg.xlsx',engine='openpyxl')
        print(df.iloc[preds]['caution'])
else:

        preds = model1.predict(x)
        preds=np.argmax(preds)
        print(preds)
        df=pd.read_excel(r'precautions - fruits.xlsx',engine='openpyxl')
        print(df.iloc[preds]['caution'])
        return df.iloc[preds]['caution']


return df.iloc[preds]['caution']

if __name__ == "__main__":

      app.run(debug=False)
```

Github link: https://github.com/IBM-EPBL/IBM-Project-45109-1660728315

Project demo link : https://drive.google.com/file/d/1cPVgobptSjDr-BDhNZQzxvQi1UOvSTPe/view?usp=drivesdk