



Fertilizer Recommendation System for Disease Prediction



Team ID: PNT2022TMID38633

Submitted by

Padmasandhiya P

Shivaraj B

Surya S

Vasanthas Prasath M

University College of Engineering Tindivanam

Anna University:Chennai-600 025

Nov 2022

Contents

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

7. CODING & SOLUTIONING (Explaining the features added in the project along with code)

- a. Feature 1
- b. Feature 2

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 Video Link

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.

LITERATURE SURVEY

[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 0.7 and 0.8 for SVM, the accuracy of identification of leaf disease of CNN is 0.6 and SVM is 0.8.

Advantages : The prediction and diagnosing of leaf diseases are depending on the segmentation such as segmenting the healthy tissues from diseased tissues of leaves.

Disadvantages : This further research is implementing the 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 stems and fruits.

[2] Detection of Leaf Diseases and Classification using Digital Image Processing International Conference on Innovations in Information, Embedded and Communication Systems(ICII ECS), IEEE, 2017.

Advantages: The system detects the diseases on citrus leaves with 90% accuracy.

Disadvantages: System only able to detect the disease from citrus leave.

The main objective of this paper is image analysis & classification techniques for detection of leaf diseases and classification. The leaf image is firstly preprocessed and then does the further work. K-Means Clustering used for image segmentation and then system extract the GLCM features from disease detected images. The disease classification done through the SVM classifier.

Algorithm used: Gray-Level Co-Occurrence Matrix (GLCM) features, SVM, K-Means Clustering .

[3] Semi-automatic leaf disease detection and classification system for soybean culture IET Image Processing, 2018

Advantages: The system helps to compute the disease severity.

Disadvantages: The system uses leaf images taken from an online dataset, so cannot implement in real time.

This paper mainly focuses on the detecting and classifying the leaf disease of soybean plant. Using SVM the proposed system classifies the leaf disease in 3 classes like i.e. downy mildew, frog eye, and septoria leaf blight etc. The proposed system gives maximum average classification accuracy reported is ~90% using a big dataset of 4775 images.

Algorithm used: SVM.

[4] Cloud Based Automated Irrigation And Plant Leaf Disease Detection System Using An Android Application. International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017.

Advantages: It is simple and cost effective system for plant leaf disease detection.

Disadvantages: Any H/w failures may affect the system performance.

The current paper proposes an android application for irrigation and plant leaf disease detection with cloud and IoT. For monitoring irrigation system they use soil moisture and temperature sensor and sensor data send to the cloud. The user can also detect the plant leaf disease. K- means clustering used for feature extraction.

Algorithm used: K-means clustering,

Other than this there are some other levels which can be used for sentimental analysis these are- document level, sentence level, entity and aspect level to study positive and negative, interrogative, sarcastic, good and bad functionality, sentiment without sentiment, conditional sentence and author and reader understanding points.

[5] The author proposes a method which helps us predict crop yield by suggesting the best crops. It also focuses on soil types in order to identify which crop should be planted in the field to increase productivity. In terms of crop yield, soil types are vital. By incorporating the weather details of the previous year into the equation, soil information can be obtained.

Advantages :It allows us to predict which crops would be appropriate for a given climate. Using the weather and disease related data sets, the crop quality can also be improved. Prediction algorithms help us to classify the data based on the disease, and data extracted from the classifier is used to predict soil and crop.

Disadvantages :Due to the changing climatic conditions, accurate results cannot be predicted by this system.

[6] 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.

[7] 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 0.7 and 0.8 for SVM, the accuracy of identification of leaf disease of CNN is 0.6 and SVM is 0.8.

Advantages : The prediction and diagnosing of leaf diseases are depending on the segmentation such as segmenting the healthy tissues from diseased tissues of leaves.

Disadvantages : This further research is implementing the 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 stems and fruits.

[8] In this paper, we propose a user-friendly web application system based on machine learning and web-scraping called the 'Farmer's Assistant'. With our system, we are successfully able to provide several features - crop recommendation using Random Forest algorithm, fertilizer recommendation using a rule based classification system, and crop disease detection using Efficient Net model on leaf images. The user can provide the input using forms on our user interface and quickly get their results. In addition, we also use the LIME interpretability

Method to explain our predictions on the disease detection image, which can potentially help understand why our model predicts what it predicts, and improve the datasets and models using this information.

Advantages : For crop recommendation and fertilizer recommendation, we can provide the availability of the same on the popular shopping websites, and possibly allow users to buy the crops and fertilizers directly from our application.

Disadvantages : To provide fine-grained segmentations of the diseased portion of the dataset. this is not possible due to lack of such data. However, in our application, we can integrate a segmentation annotation tool where the users might be able to help us with the lack. Also, we can use some unsupervised algorithms to pin-point the diseased areas in the image. We intend to add these features and fix these gaps in our upcoming work.

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.

References :

1. Food and Agriculture Organization: India at a glance. <https://www.fao.org/india/fao-in-india/india-at-a-glance/en/>. Accessed 13 July 2020
2. Mithiya, D., Bandyopadhyay, S., Mandal, K.: Measuring technical efficiency and returns to scale in Indian agriculture using panel data: a case study of West Bengal. Appl. Econ. 1–14 (2019)[CrossRef](#) [Google Scholar](#)
3. GSVA/NSVA by economic activities, Ministry of Statistics and Programmer Implementation, Govt. of India. <https://mospi.nic.in/GSVA-NSVA>. Accessed 12 July 2020
4. Dhaliwal, G.S., Jindal, V., Dhawan, A.K.: Insect pest problems and crop losses: changing trends. Indian J. Ecol. 37(1), 1–7 (2010)[Google Scholar](#)
5. Carroll, C.L., Carter, C.A., Goodhue, R.E., Lawell, C.Y.: Crop disease and agricultural productivity. National Bureau of Economic Research, Working paper 23513 (2017)[Google Scholar](#)
6. Gruhn, P., Goletti, F., Yudelman, M.: Integrated nutrient management, soil fertility, and sustainable agriculture: current issues and future challenges. Int. Food Pol. Res. Inst. (2000)[Google Scholar](#)
7. Land degradation in south Asia: Its severity, causes and effects upon the people, FAO. <https://www.fao.org/3/v4360e/V4360E05.htm>. Accessed 15 May 2020
8. Hossain, M.A., Kamiya, T., Burritt, D., Tran, L., Fujiwara, T.: Plant Macronutrient Use Efficiency: Molecular and Genomic Perspectives in Crop Plants. Academic Press, London (2017)[Google Scholar](#)
9. Sillanpää, M.: Micronutrients and the Nutrient Status of Soils: a Global Study, vol. 48. FAO, Finland (1982)[Google Scholar](#)
10. Fertilizer use by crop in India: Land and Plant Nutrition Management Service, Land and Water Development Division. FAO, Rome (2005)[Google Scholar](#)
11. Soil Nutrient Indices, Ministry of Statistics and Programmer Implementation, Govt. of India. https://www.mospi.gov.in/sites/default/files/reports_and_publication/statistical_publication/EnviStats/b14_Chapter%202.pdf. Accessed 12 July 2020
12. Ju, X.T., Kou, C.L., Christie, P., Dou, Z.X., Zhang, F.S.: Changes in the soil environment from excessive application of fertilizers and manures to two contrasting intensive cropping systems on the North China Plain. Environ. Pollut. 145(2), 497–506 (2007)[CrossRef](#) [Google Scholar](#)
13. Bannerjee, G., Sarkar, U., Das, S., Ghosh, I.: Artificial intelligence in agriculture: a literature survey. Int. J. Sci. Res. Comput. Sci. Appl. Manage. Stud. 7(3), 1–6 (2018)[Google Scholar](#)
14. Broner, I., Comstock, C.R.: Combining expert systems and neural networks for learning site-specific conditions. Comput. Electron. Agric. 19(1), 37–53 (1997)[CrossRef](#) [Google Scholar](#)
15. Moreno, R.H., Garcia, O.: Model of neural networks for fertilizer recommendation and amendments in pasture crops. In: 2018 ICAI Workshops (ICAIW), pp. 1–5. IEEE (2018)[Google Scholar](#)

Problem Statement

Definition:

Mr.Narasima 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.


- Narasima 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. Narasima 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 :



Fertilizer Recommendation System for Disease Prediction

10 minutes to prepare
1 hour to complete
2-4 people recommended

Agriculture is the most important sector in today's world. Plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants present 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 yields and food quality. In recent years, the number of diseases on plants and the degree of damage caused has increased due to the variation in pathogen strains, changes in cultivation methods, and inadequate plant protection techniques.

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

Team gathering

Define who should participate in the session and send an invite. Share relevant information or previous ideas.

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

Learn how to use the facilitation tools

Use the Facilitation Subscribers to run a happy and productive session.

[Open article](#)

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

Problem

Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.

Key rules of brainstorming

To run an effective and productive session

- Stay in topic
- Encourage wild ideas
- Defer judgement
- Listen to others
- Go for volume
- If possible, let others

2 Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Brain

You can select a sticky note and move it to the board to add it to your session.

Brainstorming #1

Problem	Solution
Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.	Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.

Brainstorming #2

Problem	Solution
Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.	Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.

Brainstorming #3

Problem	Solution
Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.	Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.

Brainstorming #4

Problem	Solution
Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.	Identify the disease or plant using deep learning based on the image. The system will recommend the fertilizer for reducing the disease. The fertilizer will be recommended based on the recommended fertilizer.

3 Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you end break it up into smaller sub-groups.

20 minutes

Category 1

Website for fertilizer recommendation	Identify the disease	Cost of using this application is less
Interactive user interface to upload images	Useful to people with no prior knowledge	It simplifies the farmers work

Category 2

Pre-trained model for image classification	Deep learning based mathematical model for detecting diseases	Build keras image classification model
They can find the diseases at early stages	Making revolutionary changes in agriculture field	Early detection and management of problem

Category 3

Instant solution	Admin can view the recommended fertilizer through gmail	Better utilization of available resources
It saves time	Smart solution to solve the problem	Cost of using this application is less

4 Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

Importance

Each of these ideas could get some attention, but which would have the most positive impact?

Better utilization of available resources	Admin can view the recommended fertilizer through gmail	Pre-trained model for image classification	Website for fertilizer recommendation
Deep learning based mathematical model for detecting diseases	They can find the diseases at early stages	Interactive UI to upload images	Useful to people with no prior knowledge
Making revolutionary changes in agriculture field	Instant solution	Cost of using this application is less	Early detection and management of problem
It simplifies work	It saves time	Smart solution to solve the problem	Cost of using this application is less

5 After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

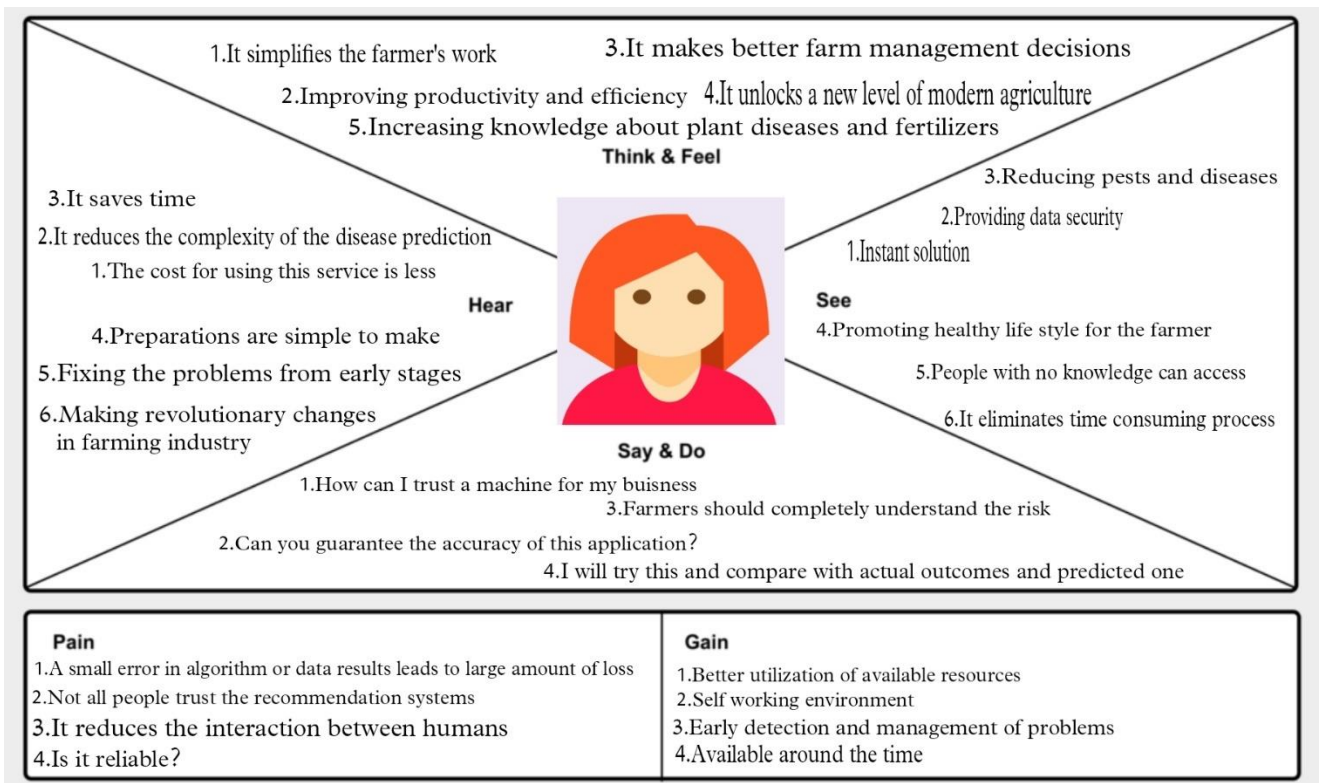
- Share the mural**
Share a new link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save to your drive.

Keep moving forward

- Strategy blueprint**
Define the components of a new idea or strategy.
[Open the template](#)
- Customer experience journey map**
Understand customer needs, motivations, and obstacles for an experience.
[Open the template](#)
- Strengths, weaknesses, opportunities & threats**
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
[Open the template](#)

[Share template feedback](#)

Empathy Map Canvas:



Proposed Solution:

S.No	Parameter	Description
1	Problem Statement (Problem to be solved)	Disease in plants reduced the quality of the plants productivity, identifying the disease in plant is hard to find.
2	Idea/Solution description	One of the solution of the problem is to identifying the disease in early stage and using the correct fertilizer.
3	Novelty/Uniqueness	This application can suggest good fertilizer for the disease in the plant by recognizing the images.
4	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.
5	Business model (revenue model)	The application recommends to farmer in subscription basis.
6	Scalability of the solution	This application can be improved by introducing online purchases of crops, fertilizers easily.

Problem Solution Fit

CUSTOMER SEGMENT: Farmers are the first customer for this application. It can be used easily and get suggestion for fertilizer to use correctly.	AVAILABLE SOLUTION: People were judge the disease in plants by identifying the leaf's quality.	CHANNELS OF BEHAVIOUR: Online: Basic knowledge on the plant and the fertilizers. Offline: People try to identify the disease by the quality of the leaf.
PROBLEM TO SOLVE: This application focuses on helping for the farmer who needs a better recommendation of fertilizer on the infected plants. Identifying the disease is one of the biggest problem here.	CUSTOMER CONSTRAINTS: Availability of good networks. Capturing the image in a required pixels to get accurate prediction of disease in the plant.	PROBLEM NOT CAUSE: Various disease on the plants leads to reducing the quality and quantity of the crops productivity. The insects on the plants can spread the disease.
TRIGGERS: Crops are being infected by disease and facing huge loss in quality and quantity.	BEHAVIOUR: Direct: Farmer can easily identify the disease by the application and they don't need any extra knowledge on the disease prediction. Indirect: Farmer can be able to get result through online immediately.	SOLUTION: Using fertilizer is one of the solution for the disease in the plants. Our application use the image of the infected leaf/plant for identifying the disease and suggest the good fertilizer for the disease.
EMOTIONS: Before: Losing self-confidence, distress After: Gaining self-confidence, relief.		

REQUIREMENT ANALYSIS :

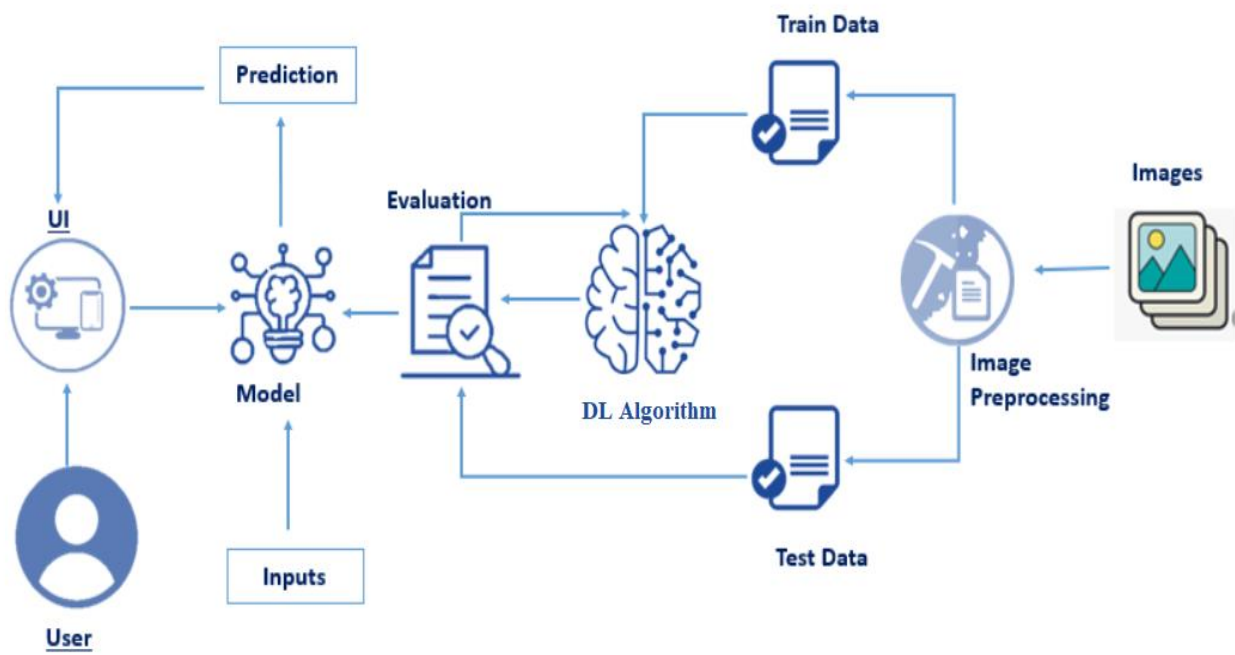
Functional requirements

S No	Functional requirements	Sub requirements(Story/subtask)
1	User registration	Registration through form Registration through Gmail
2	User confirmation	Confirmation via OTP Confirmation via Email
3	Capturing image	Capture the image of the leaf And check the parameter of the captured image
4	Image processing	Upload the image for the prediction of the disease in the leaf
5	Leaf identification	Identify the leaf and predict the disease in leaf
6	Image description	Suggesting the best fertilizer for the disease

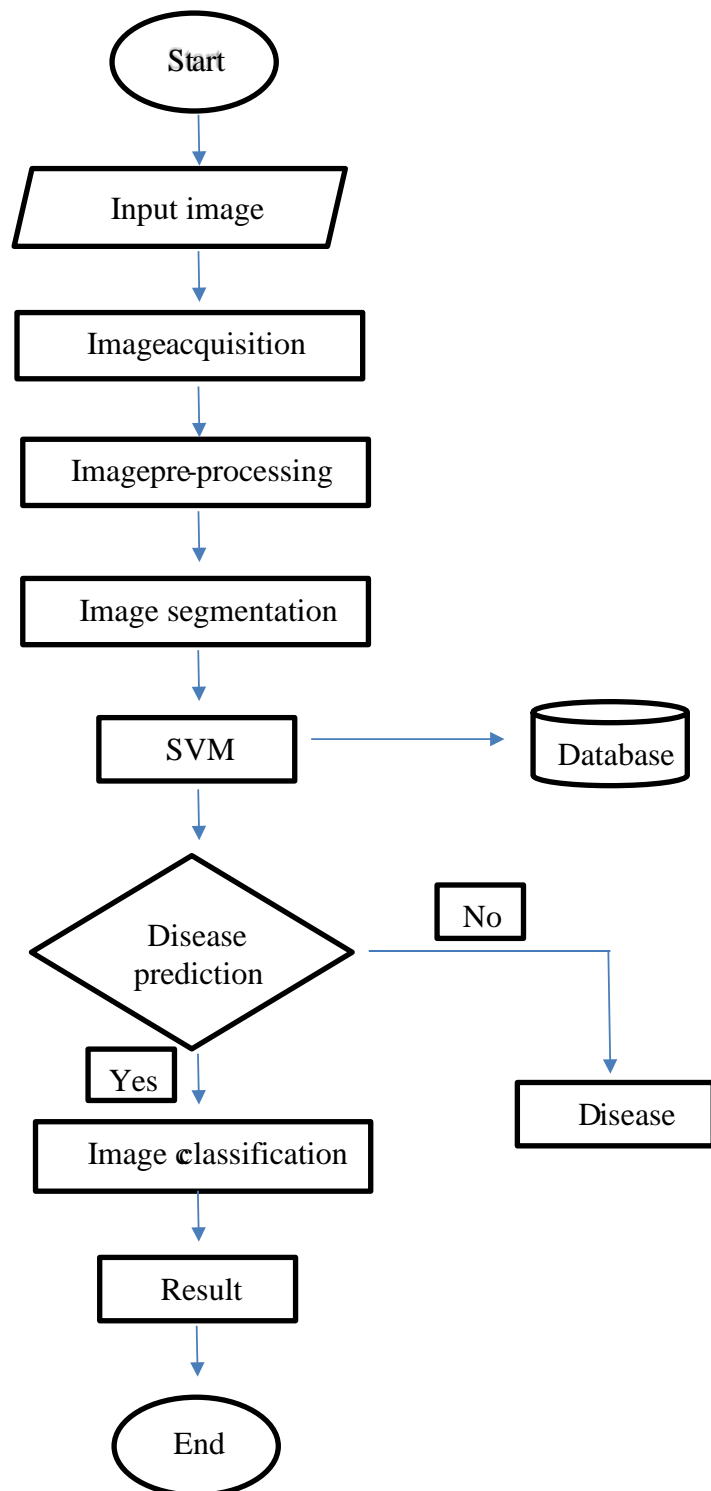
Non-functional requirements

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

Solution & Technical Architecture



DATA FLOW DIAGRAM



User Stories

User Type	Functional Requirement	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Customer (Mobile user)	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 m 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 (Web User)	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 we dashboard with the login credentials	I can login using my User credentials	High	Sprint-3
	Dashboard	USN-6	As a user, I can login I can view the web application where I can upload my images for getting the suggestion of the fertilizer	I can access my account/dashboard	High	Sprint-4
		USN-7	As a user, the fertilizer recommended to me is in high accurate	I can access my account/dashboard	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 to my dashboard	High	Sprint-5

PROJECT PLANNING & SCHEDULING:

Sprint Planning and Estimation

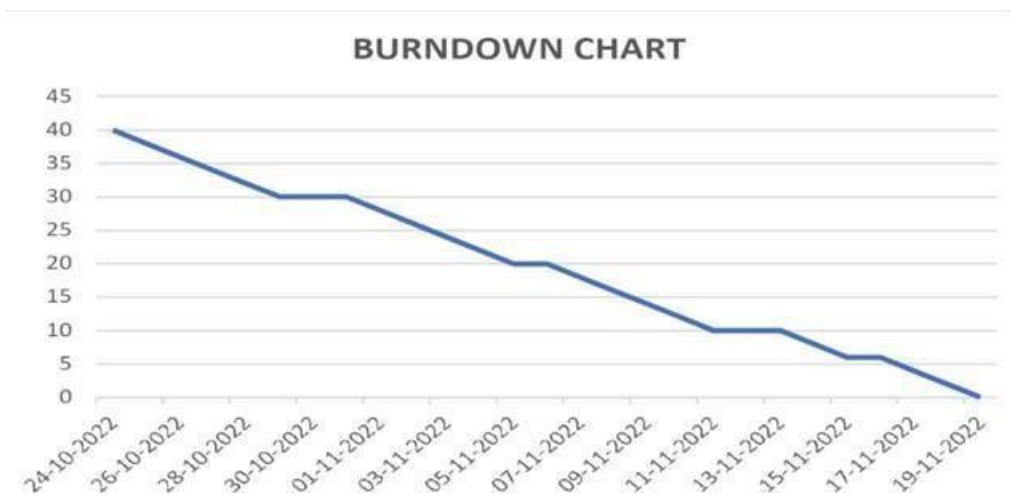
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Planning Phase	USN- 1	As a customer, I can understand the farmer's problems. Because country farmers face numerous challenges, such as detecting the actual disease.	3	Medium	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint 1	Planning Phase	USN- 2	Data collection-include gathering photos of diseased leaves from various types.	2	Medium	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint 1	Planning Phase	USN- 3	Image Preprocessing - Preprocess the diseaseaffected photos by doing things like rotating them to grayscale and calling them.	3	Low	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint 1	Planning Phase	USN- 4	Train and test the gathered dataset, as	4	Medium	Padmasandhiya P Shivaraj B Vasanthaprasath M
			well as assess its accuracy.			Surya S

Sprint2	Development Phase	USN- 5	Model building - Creating a CNN model for image segmentation	5	Low	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint2	Development Phase	USN- 6	Cnn model evaluation - Checking the accuracy and precision of the cnn model.	3	High	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint2	Development Phase	USN- 7	SVM algorithm - The SVM algorithm is used to classify images and provides 95% accuracy	5	High	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint2	Development Phase	USN- 8	Create a database for each dataset class.	3	Medium	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint3		USN- 9	Creation of User Database for the user details	2	Low	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint3		USN- 10	Description Page - The description page offers information on the predicting criteria as well as user guides.	3	Medium	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint3		USN- 11	Login Page - Login with the user's email address.	2	Low	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S

Sprint3		USN- 12	Access via password.	3	Medium	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint4	Deployment Phase	USN- 13	Dashboard and Input page creation - User profiles and prediction accuracy are included. We can feed the input images into the input page..	2	Low	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint4	Deployment Phase	USN- 14	Prediction page - Display the prediction depending on user input.	4	Low	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint4	Deployment Phase	USN- 15	Model Load – creation of API using flask	5	High	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint4	Deployment Phase	USN- 16	Using IBM cloud to deploy the application.	5	High	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint4	Deployment Phase	USN- 17	User interface and backend API calls are connected	5	High	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint4	Testing Phase	USN- 18	Test that the application function works with good accuracy and low latency.	5	High	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S
Sprint4	Testing Phase	USN- 19	Testing the application as a user ensures that all user interfaces are operational and that the prediction accuracy is correct.	5	High	Padmasandhiya P Shivaraj B Vasanthaprasath M Surya S

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	24 Oct 2022	08 Nov 2022	20	08 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	09 Nov 2022	20	09 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint -4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022



Velocity:

Sprint 1 average velocity: $20 / 2 = 10$

Sprint 2 average velocity: $20 / 2 = 10$

Sprint 3 average velocity: $20 / 1 = 20$

Sprint 4 average Velocity = $20 / 2 = 10$

CODING AND SOLUTIONING:

a. 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

layer 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 be added)

```
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())
```

4. 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 the layers.

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'))
```

5. 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.

6. 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")
```

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")
model1=load_model("fruit.h5")
```

Step 3: Configure the home page

```

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

```

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.

```

#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']

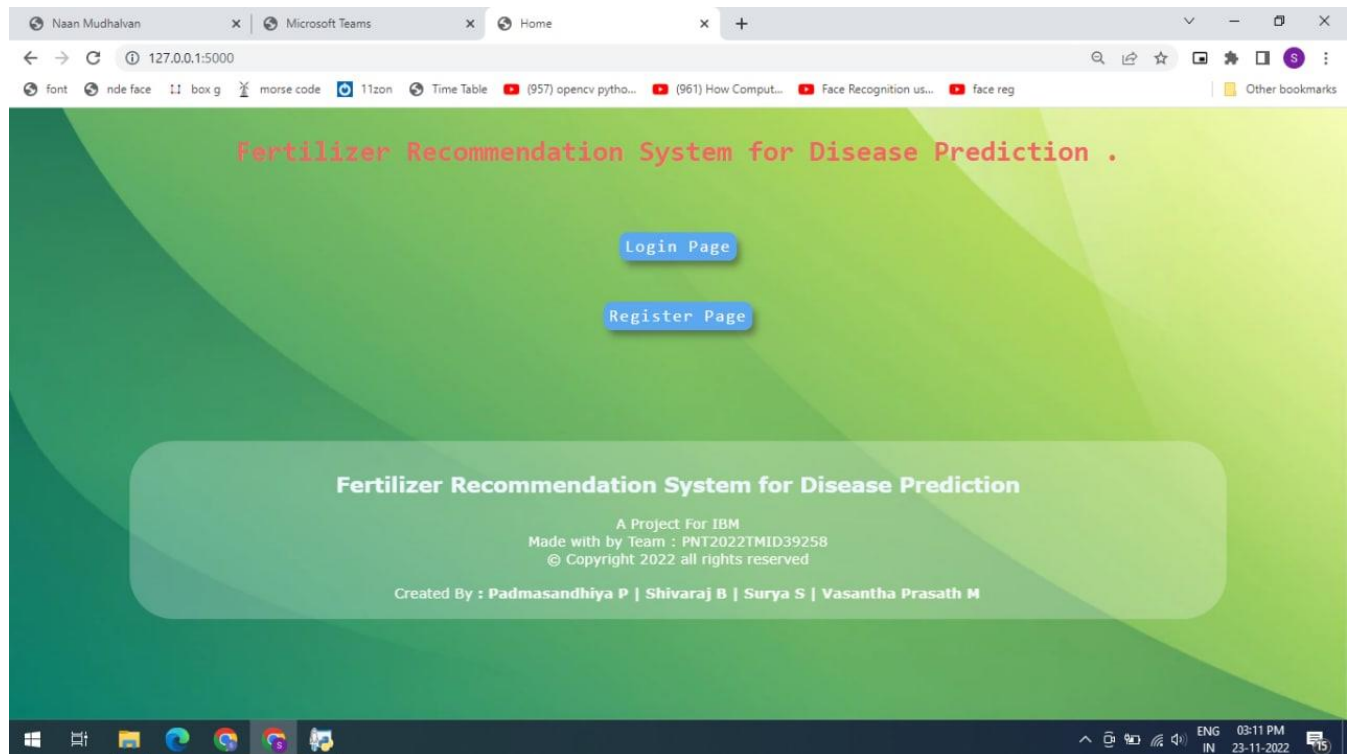
```

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.

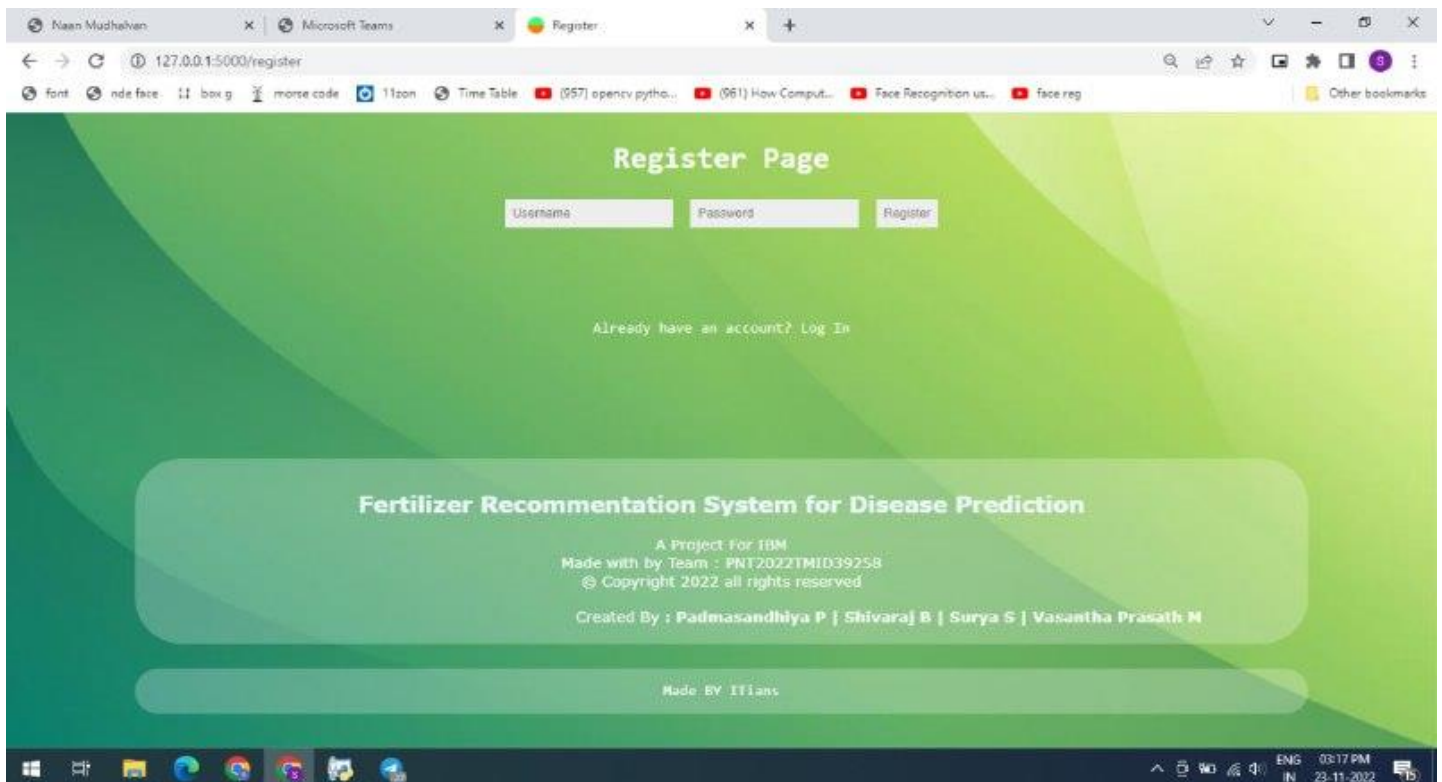
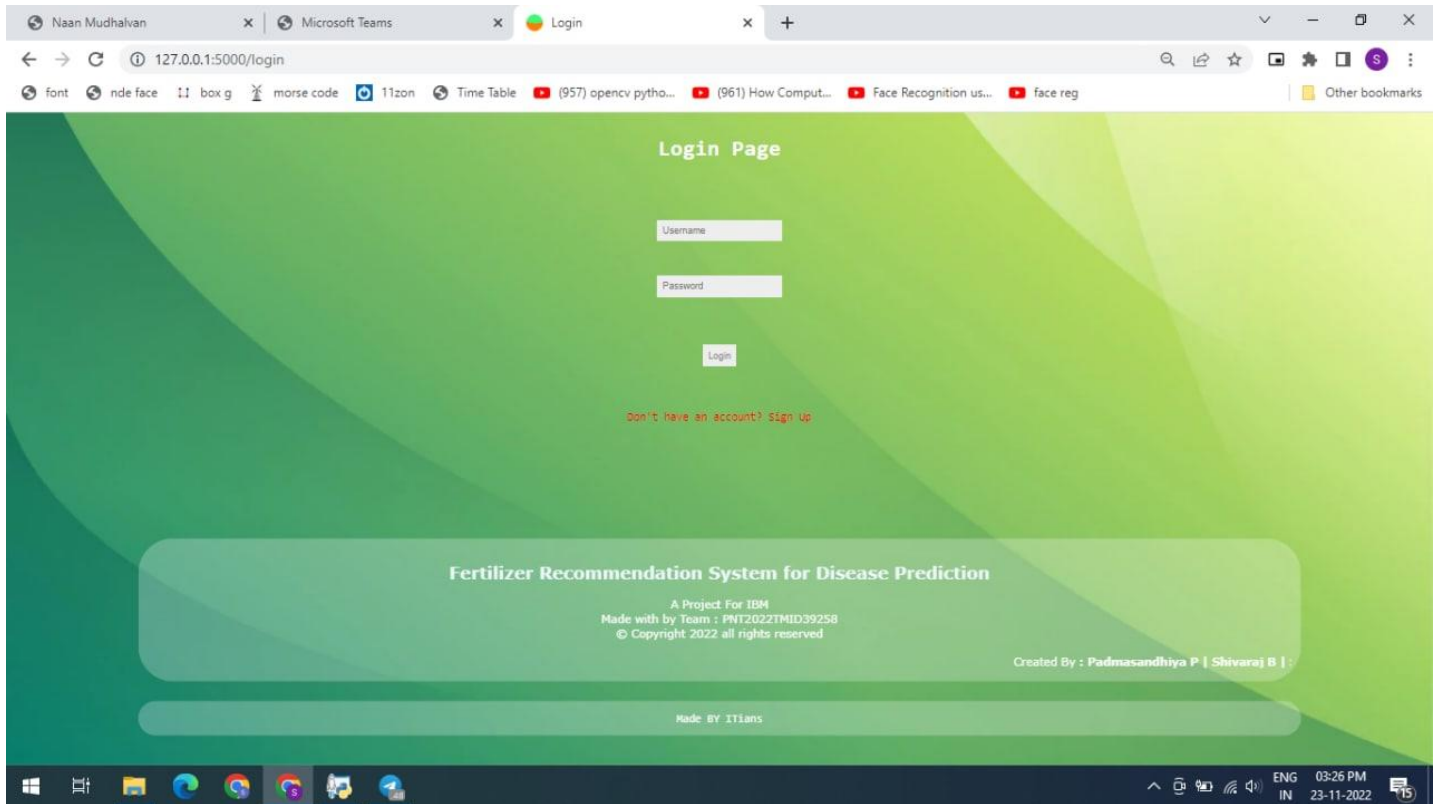
```
if __name__ == "__main__":  
    app.run(debug=False)
```

TESTING

TEST CASE 1:



TEST CASE 2:



TEST CASE 3:

Naan Mudhalvan x Microsoft Teams x Crop Recommendation x +

127.0.0.1:5000/crop-recommend

font nde face box g morse code 11zon Time Table (957) opencv pytho... (961) How Comput... Face Recognition us... face reg Other bookmarks

Wanna know which crop is suitable for your soil ?

Nitrogen
23

Phosphorous
34

Potassium
21

Ph Level
7.2

Rainfall (in Mm)
1

State
Tamil Nadu

City
Sivaganga

Predict

Fertilizer Recommendation System for Disease Prediction

A Project For IBM
Made with by Team : PNT2022TMID39258
© Copyright 2022 all rights reserved

asanthha Prasath M

Windows taskbar: File Explorer, Edge, Chrome, VS Code, Teams, etc. System tray: Network, Sound, ENG IN, 03:25 PM, 23-11-2022, 15 notifications.

Naan Mudhalvan x Microsoft Teams x Crop Recommendation x +

127.0.0.1:5000/crop-predict

font nde face box g morse code 11zon Time Table (957) opencv pytho... (961) How Comput... Face Recognition us... face reg Other bookmarks

Home Crop Fertilizer Disease Log Out

You Should Grow *Mothbeans* In Your Farm

Fertilizer Recommendation System for Disease Prediction

A Project For IBM
Made with by Team : PNT2022TMID39258
© Copyright 2022 all rights reserved

Created By : Padmasandhiya P | Shivaraj B | Surya S | Vasantha Prasath P

Windows taskbar: File Explorer, Edge, Chrome, VS Code, Teams, etc. System tray: Network, Sound, ENG IN, 03:25 PM, 23-11-2022, 15 notifications.

TEST CASE 4:

The screenshot shows a web browser window with the URL `127.0.0.1:5000/fertilizer`. The page has a dark green header with navigation links: Home, Crop, Fertilizer, Disease, and Log Out. The main content area has a teal background with the heading "Want Expert advise on fertilizer based on your Soil?". Below this, there are four input fields: "Nitrogen" with value 21, "Phosphorous" with value 33, "Potassium" with value 44, and "Crop You Want To Grow" with a dropdown menu showing "mungbean". A blue "Predict" button is at the bottom of the form. The footer of the page says "Fertilizer Recommendation System for Disease Prediction".

Want Expert advise on fertilizer based on your Soil?

Nitrogen
21

Phosphorous
33

Potassium
44

Crop You Want To Grow
mungbean

Predict

Fertilizer Recommendation System for Disease Prediction

The screenshot shows the same web browser window, but now the URL is `127.0.0.1:5000/fertilizer-predict`. The page has a light green background. A blue box contains the following text:

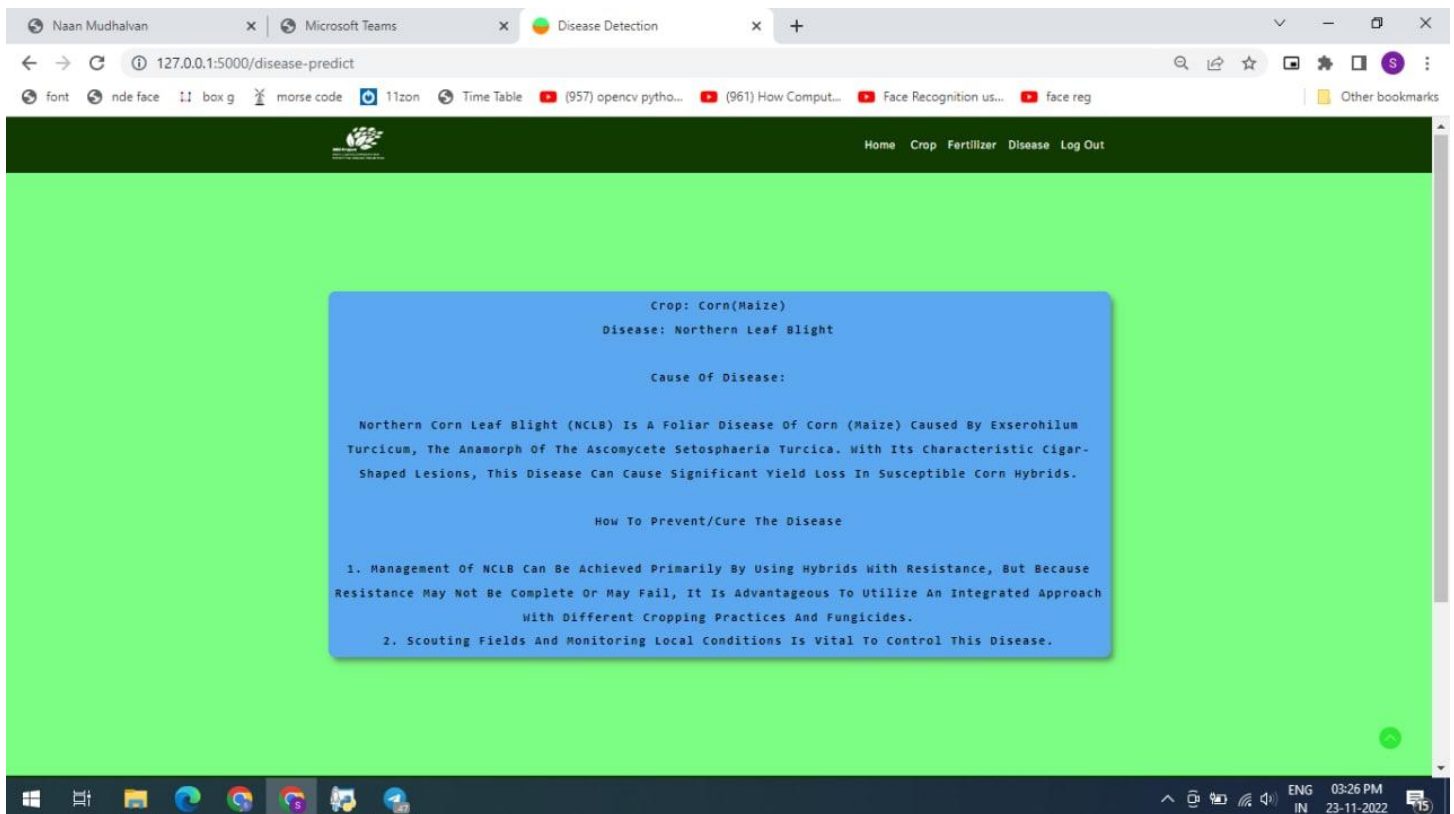
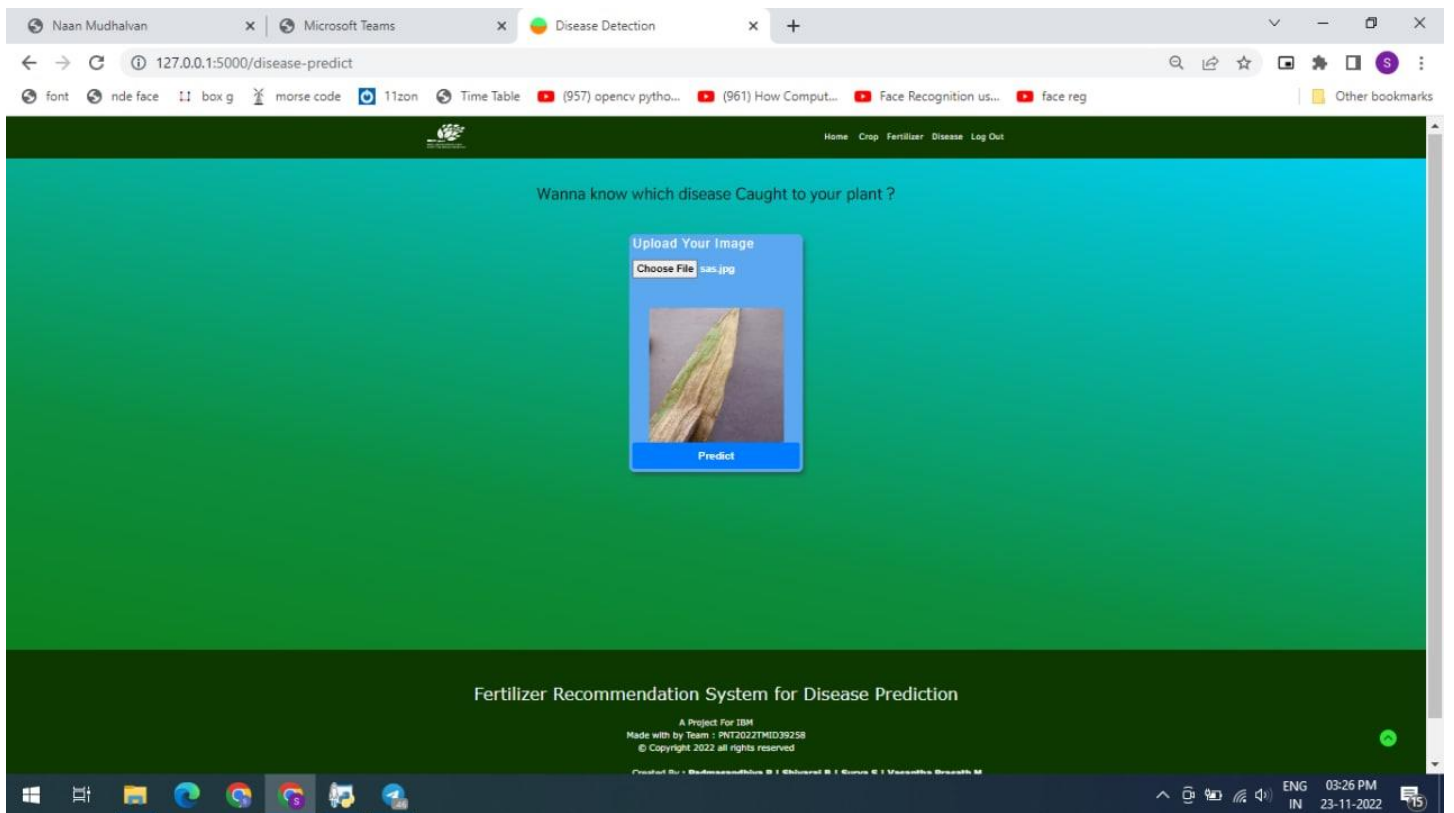
The K Value Of Your Soil Is High.
Please Consider The Following Suggestions:

1. Loosen The Soil Deeply With A Shovel, And Water Thoroughly To Dissolve Water-Soluble Potassium. Allow The Soil To Fully Dry, And Repeat Digging And Watering The Soil Two Or Three More Times.
2. Sift Through The Soil, And Remove As Many Rocks As Possible, Using A Soil Sifter. Minerals Occurring In Rocks Such As Mica And Feldspar Slowly Release Potassium Into The Soil Slowly Through Weathering.
3. Stop Applying Potassium-Rich Commercial Fertilizer. Apply Only Commercial Fertilizer That Has A '0' In The Final Number Field.

Commercial Fertilizers Use A Three Number System For Measuring Levels Of Nitrogen, Phosphorous And Potassium. The Last Number Stands For Potassium. Another Option Is To Stop Using Commercial Fertilizers All Together And To Begin Using Only Organic Matter To Enrich The Soil.

4. Mix Crushed Eggshells, Crushed Seashells, Wood Ash Or Soft Rock Phosphate To The Soil To Add Calcium. Mix In Up To 10 Percent Of Organic Compost To Help Amend And Balance The Soil.
5. Use NPK Fertilizers With Low K Levels And Organic Fertilizers Since They Have Low NPK Values.
6. Grow A Cover Crop Of Legumes That Will Fix Nitrogen In The Soil. This Practice Will Meet The Soil's Needs For Nitrogen Without Increasing Phosphorus Or Potassium.

TEST CASE 5:



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

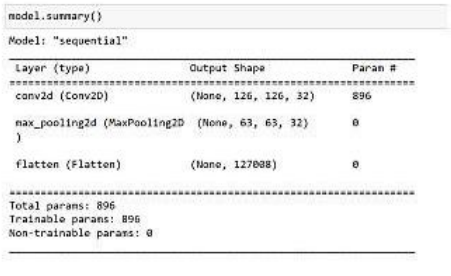
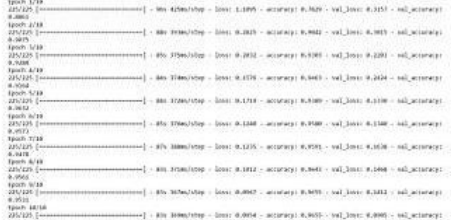


RESULTS

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
```

Locust report:

Locust Test Report

During: 11/23/2022,11:35:47 AM - 11/23/2022, 11:45:15 AM

Target Host: http://127.0.0.1:5000

Script: app.py

Request Statistics

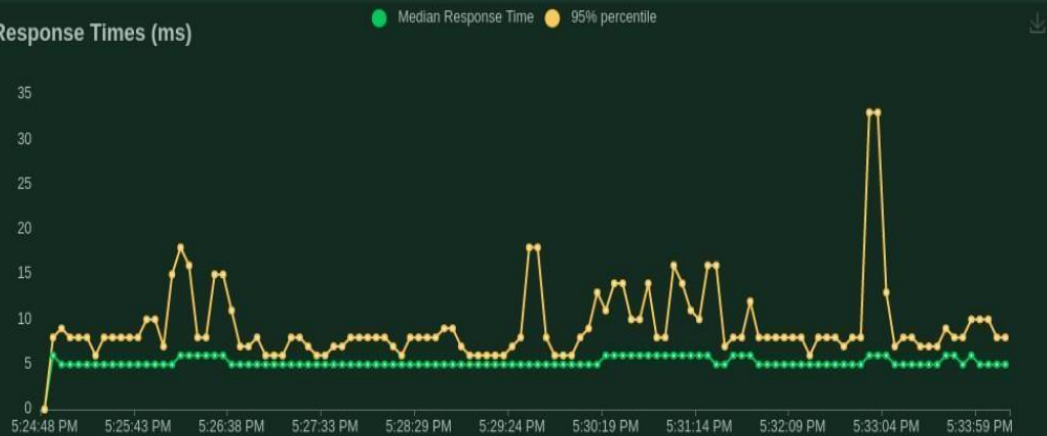
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	1890	0	5	4	41	6381	3.3	0.0
GET	/Register	1828	0	5	4	34	4484	3.2	0.0
Aggregated		3718	0	5	4	41	5448	6.5	0.0
GET	/Login	1890	0	5	4	41	6381	3.3	0.0
GET	/Dashboard	1828	0	5	4	34	4484	3.2	0.0
Aggregated		3718	0	5	4	41	5448	6.5	0.0
GET	/Crop	1890	0	5	4	41	6381	3.3	0.0
GET	/Crop-Predict	1828	0	5	4	34	4484	3.2	0.0
Aggregated		3718	0	5	4	41	5448	6.5	0.0

Charts

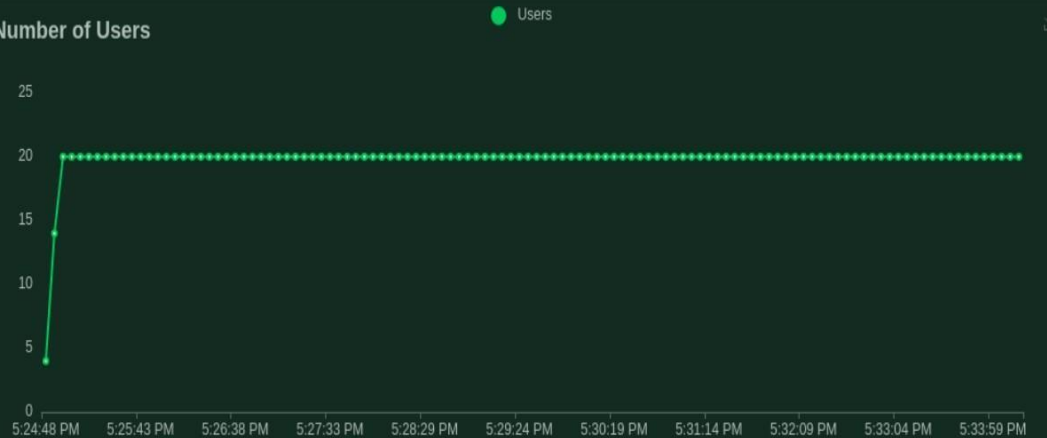
Total Requests per Second



Response Times (ms)



Number of Users



Final ratio

Ratio per User class

- 100.0% AppUser
 - 50.0% home
 - 50.0% prediction

Total ratio

- 100.0% AppUser
 - 50.0% home
 - 50.0% prediction

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- The proposed model could predict the disease just from the image of a particular plant.
- Easy to use UI.
- Model has some good accuracy in detecting the plant just by taking the input(leaf).
- These kind of web applications can be used in the agricultural sector as well as for small house hold plants as well.

DISADVANTAGES:

- Prediction is limited to few plants as we haven't trained all the plants.

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 the 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 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 more people towards the agriculture.

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 implementing the 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

Requirement.txt

numpy
pandas
Flask
scikit-
learn
torch
Pillow
gunicorn == 20.0.4
asgiref==3.5.0
bcrypt==3.2.0
cffi==1.15.0
click==8.1.2
dnspython==2.2.1
email-validator==1.1.3
Flask==2.1.1
Flask-Bcrypt==1.0.1
Flask-Login==0.6.0
Flask
SQLAlchemy==2.5.1
Flask-WTF==1.0.1
greenlet==1.1.2
idna==3.3
importlib-
metadata==4.11.3
itsdangerous==2.1.2
Jinja2==3.1.1
MarkupSafe==2.1.1
pyparser==2.21
six==1.16.0
SQLAlchemy==1.4.3.5
sqlparse==0.4.2
Werkzeug==2.1.1
WTForms==3.0.1
zipp==3.8.0
flask_sqlalchemy
flask_login flask_wtf
wtforms
wtforms.validators
flask_bcrypt

SOURCE CODE

```
# Importing essential libraries and modules

from flask import Flask, render_template, request, Markup, url_for, redirect
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
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import InputRequired, Length, ValidationError
from flask_bcrypt import Bcrypt

#
=====
=====

# -----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__)

db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'

```

```

app.config['SECRET_KEY'] = 'thisisasecretkey'

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False, unique=True)
    password = db.Column(db.String(80), nullable=False)

class RegisterForm(FlaskForm):
    username = StringField(validators=[
        InputRequired(), Length(min=4, max=20)], render_kw={"placeholder": "Username"})

    password = PasswordField(validators=[
        InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Password"})

    submit = SubmitField('Register')

    def validate_username(self, username):
        existing_user_username = User.query.filter_by(
            username=username.data).first()
        if existing_user_username:
            raise ValidationError(
                'That username already exists. Please choose a different one.')

class LoginForm(FlaskForm):
    username = StringField(validators=[
        InputRequired(), Length(min=4, max=20)], render_kw={"placeholder": "Username"})

    password = PasswordField(validators=[
        InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Password"})

    submit = SubmitField('Login')

@app.route('/')
def home():
    return render_template('home.html')

```

```

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user:
            if bcrypt.check_password_hash(user.password, form.password.data):
                login_user(user)
                return redirect(url_for('dashboard'))
    return render_template('login.html', form=form)

@app.route('/dashboard', methods=['GET', 'POST'])
@login_required
def dashboard():
    return render_template('dashboard.html')

@app.route('/logout', methods=['GET', 'POST'])
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))

@ app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegisterForm()

    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data)
        new_user = User(username=form.username.data, password=hashed_password)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))

    return render_template('register.html', form=form)

# render home page

# render crop recommendation form page

@ app.route('/crop-recommend')
def crop_recommend():
    title = 'Crop Recommendation'

```

```

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

# render fertilizer recommendation form page

@ app.route('/fertilizer')
def fertilizer_recommendation():
    title = '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 = '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 = '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 = '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)

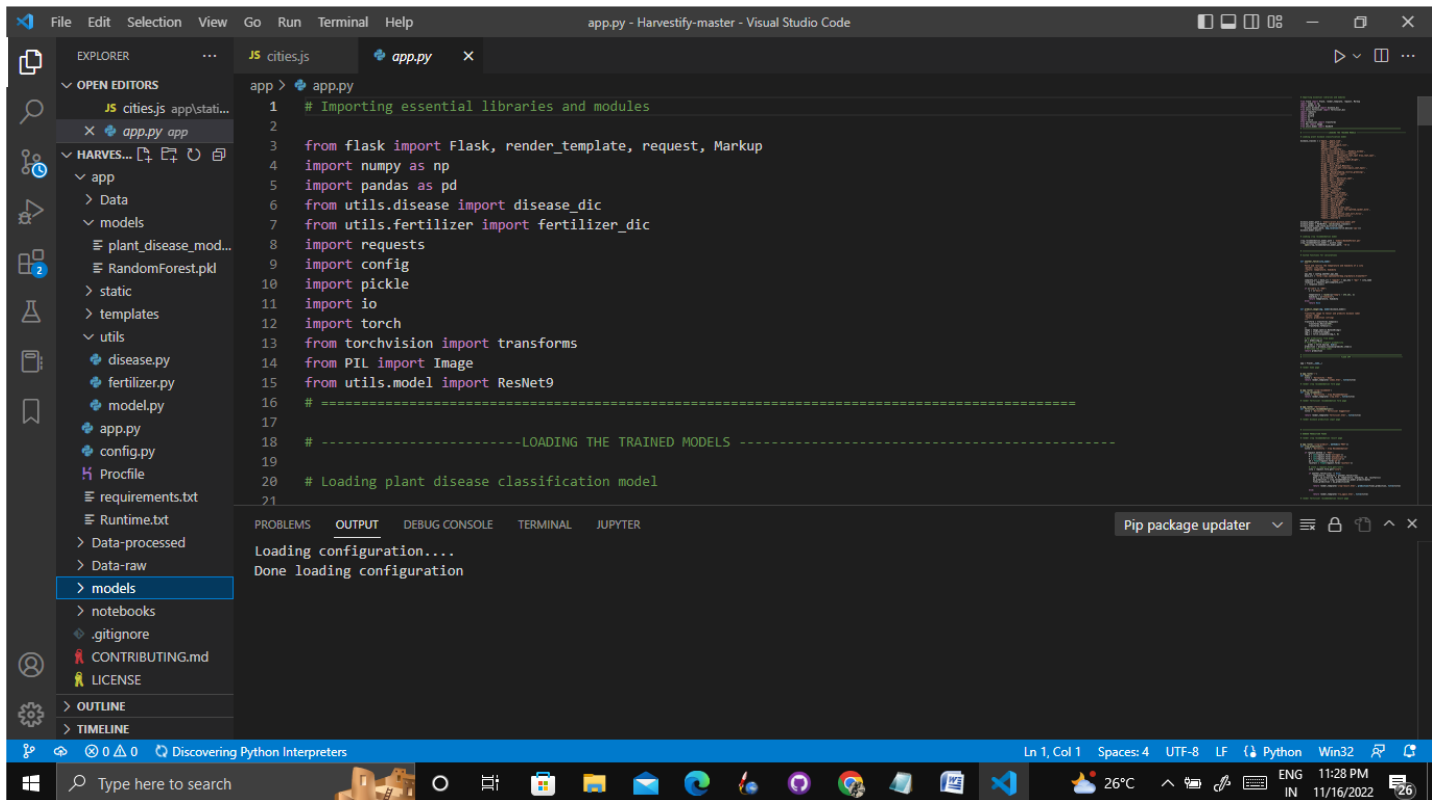
            prediction = Markup(str(disease_dic[prediction]))
            return render_template('disease-result.html', prediction=prediction, title=title)
        except:
            pass
    return render_template('disease.html', title=title)

#
=====

if __name__ == '__main__':
    app.run(debug=True)

```


Project Structure:



GITHUB LINK:

<https://github.com/IBM-EPBL/IBM-Project-18268-1659682285>

VIDEO LINK:

<https://youtu.be/c7OwMMX8Sik>

