Fertilizers Recommendation System for Disease Prediction

PROJECT REPORT

Submitted by

Team ID: PNT2022TMID27540

Seshathri N (311119106046)

Jerina A (311119106018)

Emima S (311119106013)

Dino J (311119106301)

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION ENGINEERING

LOYOLA ICAM COLLEGE OF ENGINEERING AND TECHNOLOGY

CHENNAI

1. INTRODUCTION:

1.1. Project Overview

In this project, two datasets named fruit dataset and vegetable dataset are collected. The collected datasets are trained and tested with a Deep Learning Neural Network named Convolutional Neural Networks (CNN). First, the fruit dataset is trained and then tested with CNN. It has 6 classes and all the classes are trained and tested. Next, the vegetable dataset is trained and tested. The software used for training and testing of datasets is Python. All the Python codes are first written in the Jupyter notebook supplied along with Anaconda Python and then the codes are tested in IBM cloud. Finally, a web-based framework is designed with the help of Flask, a Python library. Two html files are created in the templates folder along with their associated files in the static folder. The Python program 'app.py' used to interface with these two web pages is written in Spyder-Anaconda python and tested.

1.2. Purpose

The purpose of this project is used to train and test the fruits and vegetables samples and identify the different diseases caused in fruits and vegetables and recommend suitable fertilizers to predict the diseases.

2. LITERATURE SURVEY

2.1. Existing problem

Narasimma Rao proposed a method for leaf disease detection and suggested fertilizers to cure leaf diseases. But the method involves less
number of train and test sets which results in poor accuracy. Suresh proposed a simple prediction method for soil-based fertilizer recommendation system for predicted crop diseases. This method gives less accuracy and prediction.
Shiva proposed an IoT based system for leaf disease detection and fertilizer

recommendation which is based on Machine Learning techniques yields less 80 percentage accuracy.

2.2. References

References:

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

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

[3] Ms. Kiran R. Gavhale, Ujwalla Gawande, Plant Leaves Disease detection using Image Processing Techniques, January 2014.

https://www.researchgate.net/profile/UjwallaGawande/publication/314436486_An_Overview_of_the
_Research_on_Plant_Leaves_Disease_detection_using_Image_Processing_Techniques/links/5d37106
64585153e591a3d20/An-Overviewof-the-Research-on-Plant-Leaves-Diseae detection-using-Image-ProcessingTechniques.pdf

[4] Duan Yan-e, Design of Intelligent Agriculture Management Information System Based on

IOTǁ, IEEE,4th, Fourth International reference on Intelligent Computation Technology and Automation, 2011

https://ieeexplore.ieee.org/document/5750779

[5] R. Neela, P. Fertilizers Recommendation System For Disease Prediction In Tree Leave International journal of scientific & technology research volume 8, issue 11, november 2019

http://www.ijstr.org/final-print/nov2019/Fertilizers-Recommendation-System-For-Disease-PredictionIn-Tree-Leave.pdf .

[6] Swapnil Jori1, Rutuja Bhalshankar2, Dipali Dhamale3, Sulochana Sonkamble , Healthy Farm: Leaf Disease Estimation and Fertilizer Recommendation System using Machine Learning, International Journal of All Research Education and Scientific Methods (IJARESM), ISSN:
2455-6211

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

[8] Shloka Gupta ,Nishit Jain ,Akshay Chopade, Farmer's Assistant: A Machine Learning Based Application for Agricultural Solutions.

2.3. Problem Statement Definition

Mr.Narasimma Rao is a 65 years old man. He had his own farming land and did agriculture for the past 30 years. In these 30 years, he faced problems in choosing Fertilizers and controlling of Plant Diseases.

• Narasimma Rao wants to know the best recommendation for fertilizers for plants with the disease.

• He has faced huge losses for a long time.

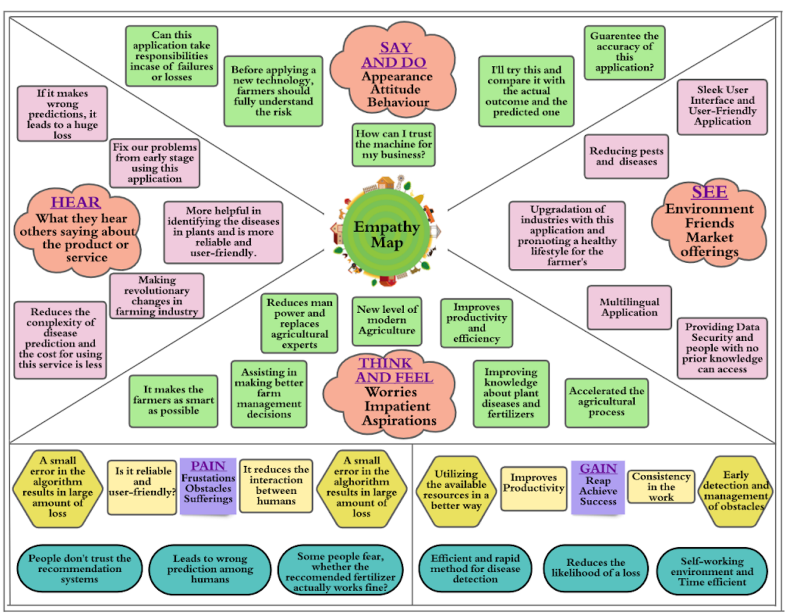
• This problem is usually faced by most farmers.

• Mr. Narasimma Rao needs to know the result immediately.

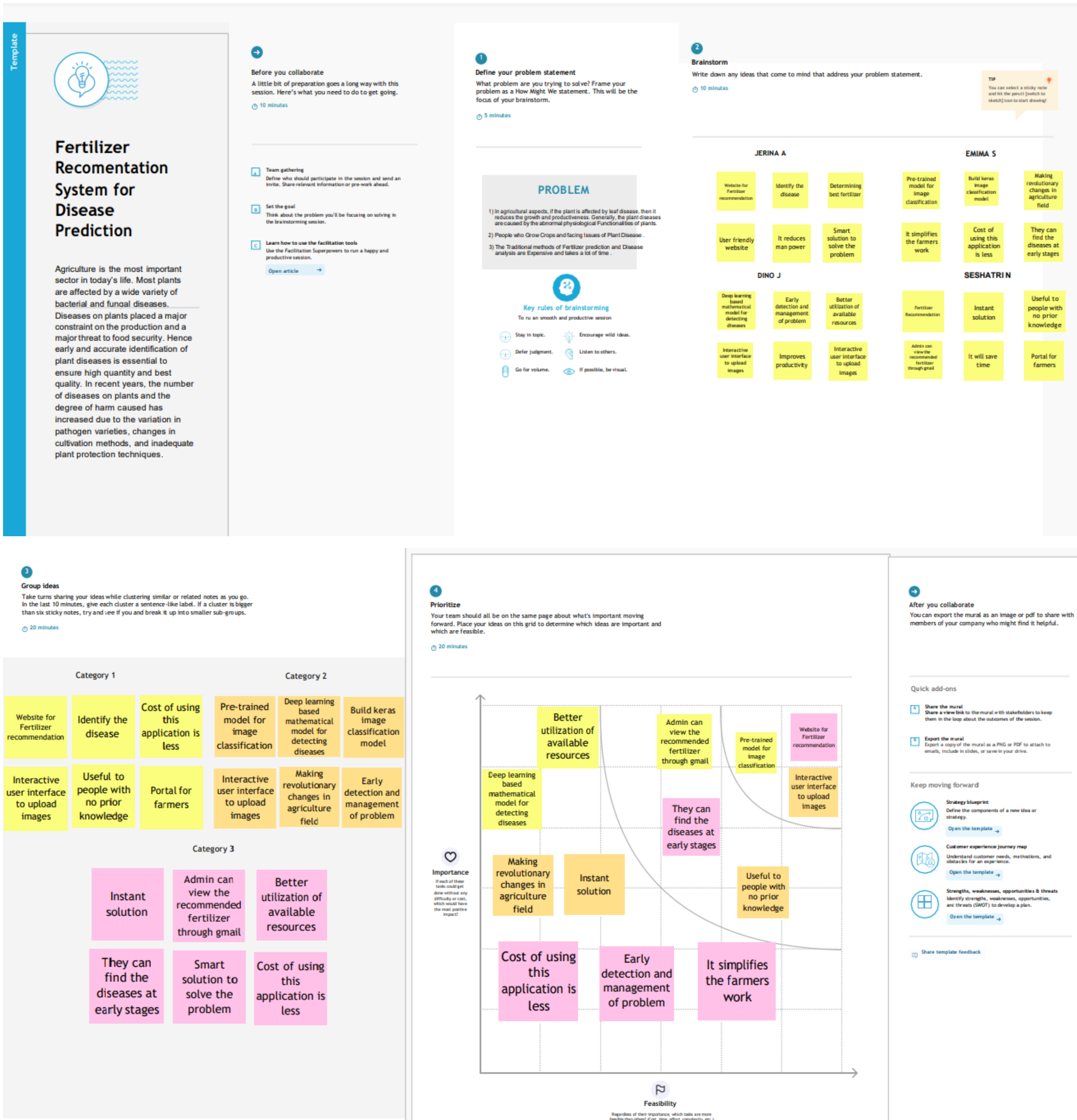
3. IDEATION & PROPOSED SOLUTION

3.1. Empathy Map Canvas

Fertilizers Recommendation System For Disease Prediction



3.2. Ideation & Brainstorming



3.3. Proposed Solution

S. No	Parameter	Description
1.	**Problem Statement (Problem to be solved)**	**Fertilizers Recommendation System for Disease Prediction** In India, the agricultural sector provides a living for almost 48% of the population. Most of the Indian population depends on agriculture for their livelihood. Majority of the farmers face the problem of planting an inappropriate crop for their land based on non-scientific approach and the outcomes for the farmer for choosing the wrong crop for the land is moving towards quitting agriculture, ending their lives and giving land on lease to industrialist or use it for non-agricultural purposes. The outcome of wrong crop selection is less yield and less profit.

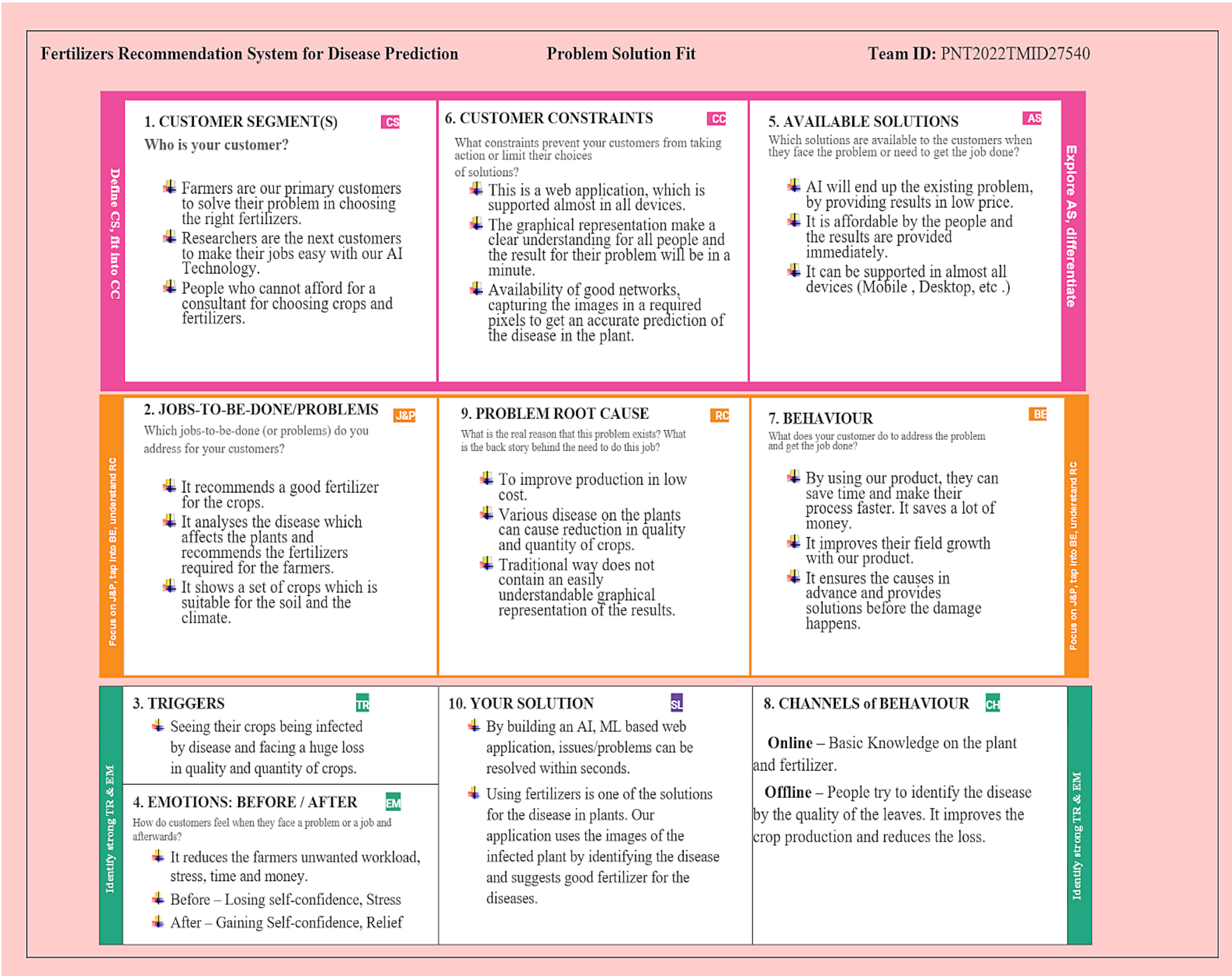
2.	**Idea / Solution description**	The solution to the problem is Machine Learning (ML) which is one of the applications of Artificial Intelligence (AI), which is used to implement the proposed system. Crop recommendation is going to recommend the best crop one can grow in their land as per soil nutrition value and along with the climate in the region. The challenging task is to recommend the best fertilizer for every crop. An important issue is when a plant gets caught by heterogeneous diseases that affect agricultural production and quality. To overcome these issues this recommendation system has been proposed. The technique is used to build a recommendation model that combines the prediction of multiple ML. Models to recommend the right crop based on soil value and the best fertilizer to use.

3.	**Novelty / Uniqueness**	The system comes with a model to be precise and accurate in predicting crop yield and deliver the end user with the proper recommendations about required fertilizer ratio based on atmospheric and soil parameters of the land which enhance to increase the crop yield and increase farmers revenue. Thus, the proposed system takes the data regarding the quality of soil and the weather-related information as an input. The quality of the soil such as Nitrogen, Phosphorous, Potassium and Ph value. Weather related information like Rainfall, Temperature and Humidity to predict the better crop

4.	**Social Impact / Customer Satisfaction**	In India, the majority of the population is dependent on agriculture for their livelihood. Many new technologies like Machine Learning (ML) and Deep Learning (DL) are being implemented into agriculture so it is easier for farmers to grow and maximize their yield crops. The beneficial users are Farmers, Seller, Buyer, Employees, Industrial people, Common people. In the crop recommending application, the user can provide soil data and the application will predict what are the crops that can be grown by the user. In the fertilizer recommending application, the soil nutrient analysis uses a soil NPK sensor with the recommendation of fertilizers according to the obtained nutrient value, the user can input the soil data and the type of crop they are growing, the application will predict what is lacking or being excess in the soil and will recommend improvements. The last application is the plant disease prediction application where the user can input an image of a diseased plant leaf, and the application will predict the disease caused and will give suggestions to cure it.

5.	**Business Model (Revenue Model)**	Predicting the fertilizers, analyzing the diseases in a tap makes the life of farmers easy with minimal subscriptions and would provide an acceptable return for the organization. This action adds a lot of value to the company and the business in society.
6.	**Scalability of the Solution**	On-spot results are obtained, and the time required for fertilizer recommendation is within the 80s. Successful identification of crops that can be grown and the necessary fertilizer is recommended with more than 90% accuracy. The proposed approach is also compared with the other intelligent approaches, such as Artificial Neural Network (ANN), K-Nearest Neighbour (KNN), and Support Vector Machine (SVM), and it is observed that the proposed CNN approach gives higher accuracy in the shortest time.

3.4. Problem Solution fit



Fertilizers Recommendation System for Disease Prediction **Problem Solution Fit** **Team ID:** PNT2022TMID27540

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

- Farmers are our primary customers to solve their problem in choosing the right fertilizers.
- Researchers are the next customers to make their jobs easy with our AI Technology.
- People who cannot afford for a consultant for choosing crops and fertilizers.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions?

- This is a web application, which is supported almost in all devices.
- The graphical representation make a clear understanding for all people and the result for their problem will be in a minute.
- Availability of good networks, capturing the images in a required pixels to get an accurate prediction of the disease in the plant.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done?

- AI will end up the existing problem, by providing results in low price.
- It is affordable by the people and the results are provided immediately.
- It can be supported in almost all devices (Mobile , Desktop, etc .)

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE/PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers?

- It recommends a good fertilizer for the crops.
- It analyses the disease which affects the plants and recommends the fertilizers required for the farmers.
- It shows a set of crops which is suitable for the soil and the climate.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?

- To improve production in low cost.
- Various disease on the plants can cause reduction in quality and quantity of crops.
- Traditional way does not contain an easily understandable graphical representation of the results.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?

- By using our product, they can save time and make their process faster. It saves a lot of money.
- It improves their field growth with our product.
- It ensures the causes in advance and provides solutions before the damage happens.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

- Seeing their crops being infected by disease and facing a huge loss in quality and quantity of crops.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

- It reduces the farmers unwanted workload, stress, time and money.
- Before – Losing self-confidence, Stress
- After – Gaining Self-confidence, Relief

10. YOUR SOLUTION SL

- By building an AI, ML based web application, issues/problems can be resolved within seconds.
- Using fertilizers is one of the solutions for the disease in plants. Our application uses the images of the infected plant by identifying the disease and suggests good fertilizer for the diseases.

8. CHANNELS of BEHAVIOUR CH

Online – Basic Knowledge on the plant and fertilizer.

Offline – People try to identify the disease by the quality of the leaves. It improves the crop production and reduces the loss.

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1. Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via OTP
FR-3	User Profile	Filling the profile page after logging in
FR-4	Upload Dataset	Images of the leaves are uploaded
FR-5	Request solution	Uploaded images are compared with a pre-defined model and a solution is generated.
FR-6	Download solution	The solution document contains the recommendations of fertilizers and the possible diseases.

4.2. Non-Functional requirements

NFR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
NFR-1	Usability	The system allows the user to perform the tasks efficiently and effectively.
NFR-2	Security	Assuring whether all the data inside the system or its parts will be protected against malicious attacks or

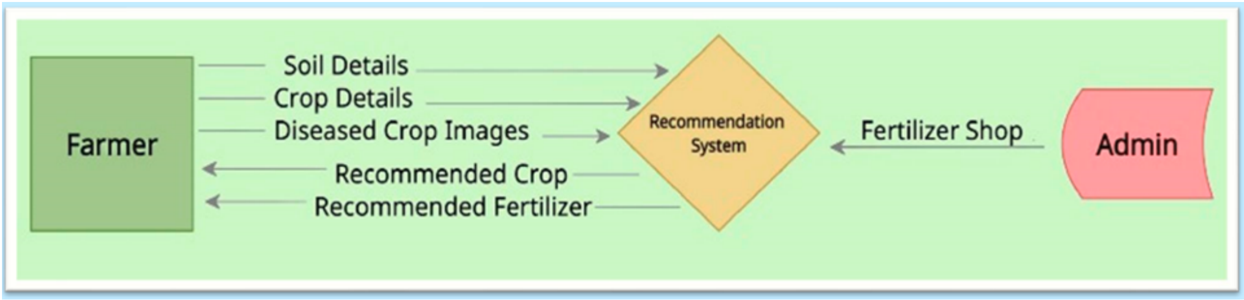
		unauthorized access.
NFR-3	Reliability	The website takes time to recover from failure quickly as the application is running in the single server
NFR-4	Performance	Response Time and Net Processing Time is fast
NFR-5	Availability	The system will be available up to 95% of the time
NFR-6	Scalability	The website is scalable

5. PROJECT DESIGN

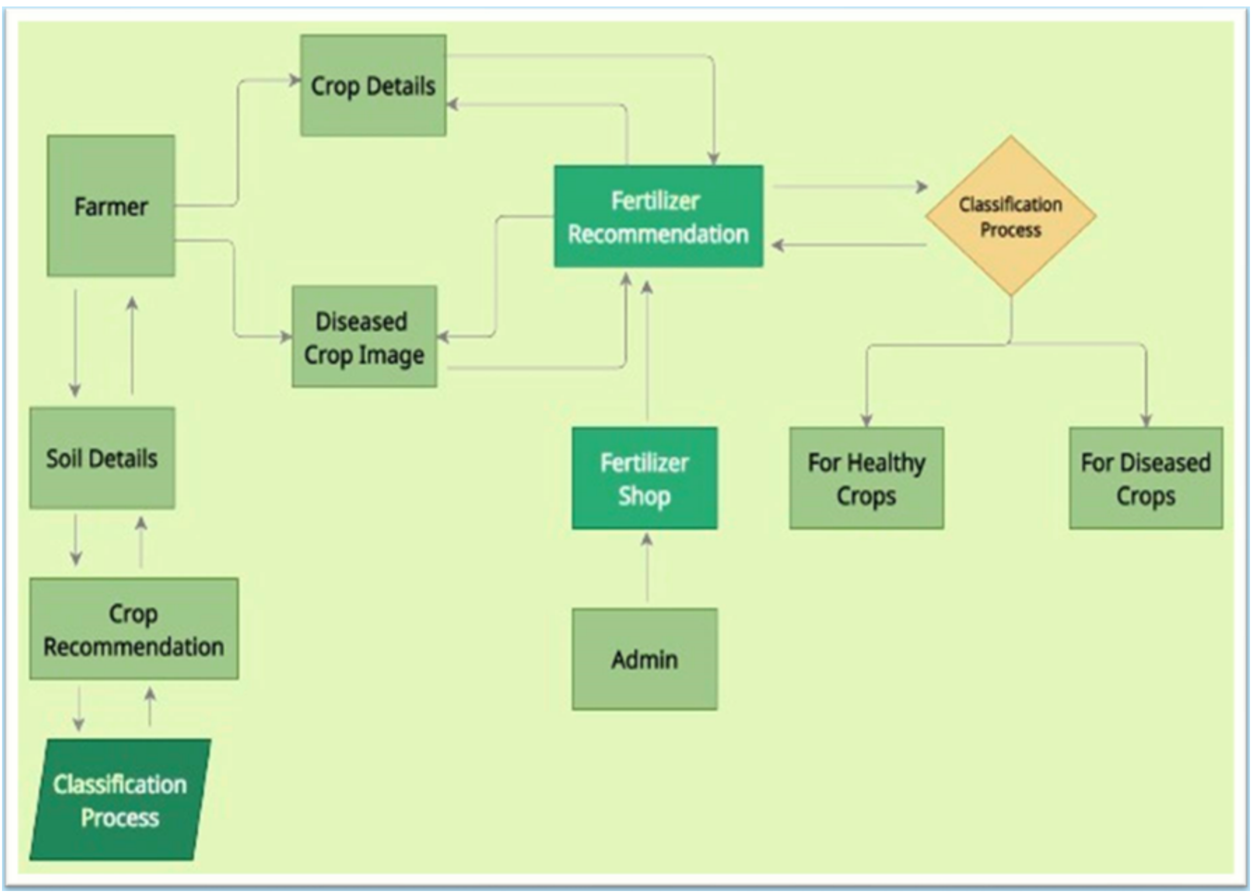
5.1. Data Flow Diagrams

Data Flow Diagrams(DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information and where is stored.

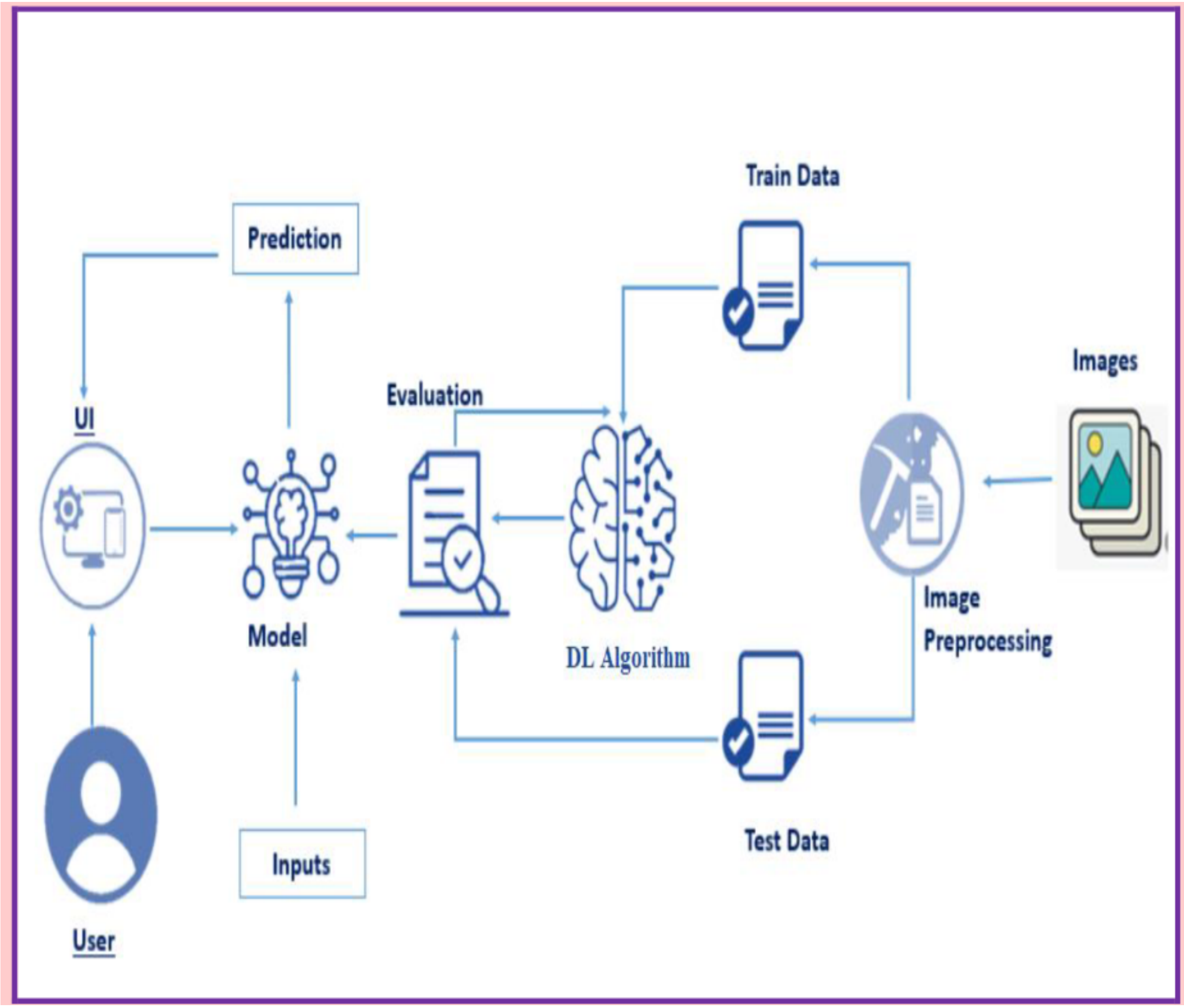
DFD LEVEL 0



DFD LEVEL 1



5.2. Solution & Technical Architecture



5.3. User Stories

[redacted]

6. PROJECT PLANNING & SCHEDULING

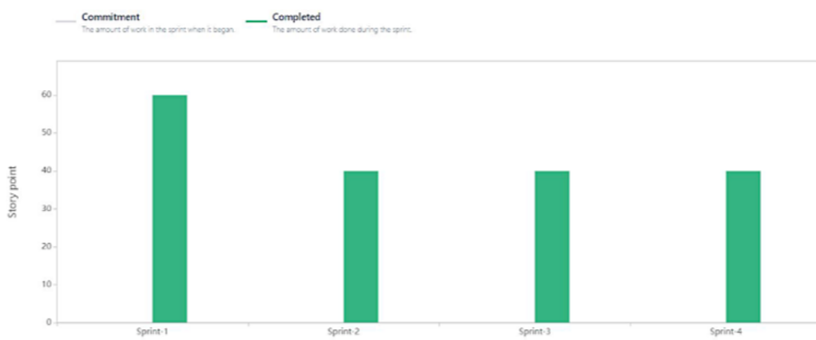
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint- 1	Data Collection	USN-1	Collecting dataset for pre-processing	1	High	Jerina A Emima S Dino J Seshathri N
Sprint- 1		USN-2	Data pre-processing- Used to transform the data into useful format.	1	Medium	Jerina A Emima S Dino J Seshathri N
Sprint- 2	Model Building	USN-3	Model building for fruit and vegetable disease prediction	1	High	Jerina A Emima S Dino J Seshathri N
Sprint- 2		USN-4	Splitting the data into training and testing from the entire dataset.	2	Medium	Jerina A Emima S Dino J Seshathri N
Sprint- 3	Training and Testing	USN-5	Training the model and testing the performance of the model	2	Medium	Jerina A Emima S Dino J Seshathri N
Sprint- 4	Implementation of Web page	USN-6	Implementing the web page for collecting the data from user	2	High	Jerina A Emima S Dino J Seshathri N
Sprint- 4		USN-7	Deploying the model using IBM Cloud and IBM Watson Studio	2	Medium	Jerina A Emima S Dino J Seshathri N

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint- 1	20	6 Days	22 Oct 2022	27 Oct 2022	20	28 Oct 2022
Sprint- 2	20	6 Days	29 Oct 2022	03 Nov 2022	20	04 Nov 2022
Sprint- 3	20	6 Days	05 Nov 2022	10 Nov 2022	20	11 Nov 2022
Sprint- 4	20	6 Days	12 Nov2022	17 Nov 2022	20	18 Nov 2022

6.3 Reports from JIRA



Sprint	Commitment	Completed
Sprint-1	0	60
Sprint-2	0	40
Sprint-3	0	40
Sprint-4	0	40

8. TESTING

8.1 Test Cases

Section	Total Cases	Not Tested	Fail	Pass
Yellow Leaves	20	0	0	20
Blights	43	0	0	43
Fruit rots	9	0	0	9
Leaf spots	5	0	0	5
Mosaic leaf pattern	19	0	0	19
Fruit Spots	2	0	0	2
Leaves misshapen	4	0	0	4

8.2 User Acceptance Testing

1. **Purpose of Document**

   The purpose of this document is to briefly explain the test coverage and open issues of the [Fertilizer 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
Yellow Leaves	10	4	5	15	34
Blights	1	5	2	4	12
Fruit rots	3	1	0	2	6
Leaf spots	9	2	4	18	33
Mosaic leaf pattern	3	9	6	6	24
Fruit Spots	3	1	5	1	10
Leaves misshapen	0	7	2	1	10
Totals	29	29	24	47	129

9. RESULTS

9.1 Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Model Summary	-	-
2.	Accuracy	**1) Fruit Dataset:** Training Accuracy - 98.8 Validation Accuracy - 64.8 **2) Vegetable Dataset:** Training Accuracy - 96.8 Validation Accuracy - 35.3	**1) Fruit Dataset:**  **2) Vegetable Dataset:** 
3.	Confidence Score (Only Yolo Projects)	Class Detected - NA Confidence Score - NA	NA

10. ADVANTAGES & DISADVANTAGES

Advantages:

- The proposed model here produces very high accuracy of classification.
- Very large datasets can also be trained and tested.
- Images of very high can be resized within the proposed itself.

Disadvantages:

- For training and testing, the proposed model requires very high computational time.
- The neural network architecture used in this project work has high complexity.

11. CONCLUSION

The model proposed involves image classification of fruit datasets and vegetable datasets. The following points are observed during model testing and training:

- The accuracy of classification increased by increasing the number of epochs.
- For different batch sizes, different classification accuracies are obtained.
- The accuracies are increased by increasing more convolution layers.
- The accuracy of classification also increased by varying dense layers.
- Different accuracies are obtained by varying the size of kernel used in the convolution layer output.
- Accuracies are different while varying the size of the train and test

datasets.

12. FUTURE SCOPE

The proposed model in this project work can be extended to image recognition. The entire model can be converted to application software using python to exe software. The real time image classification, image recognition and video processing are possible with help OpenCV python library. This project work can be extended for security applications such as figure print recognition, iris recognition and face recognition.

13. APPENDIX

Source Code

1. Fruit_data.ipynb

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```
In [94]:
```
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_
flip=True,vertical_flip=False)
```

In [95]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [119]:
```
x_train=train_datagen.flow_from_directory(r"/content/drive/MyDrive/Dataset/
fruit-dataset/fruit-dataset/train",target_size=(128,128),

class_mode='categorical',batch_size=24)
Found 5393 images belonging to 6 classes.
```

In [120]:
```
x_test=test_datagen.flow_from_directory(r"/content/drive/MyDrive/Dataset/fr
uit-dataset/fruit-dataset/test",target_size=(128,128),

class_mode='categorical',batch_size=24)
Found 1686 images belonging to 6 classes.
```

In [121]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import
Dense,Convolution2D,MaxPooling2D,Flatten
```

In [122]: `model=Sequential()`

In [123]:
```
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'
))
```

In [124]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 126, 126, 32)      896

 max_pooling2d_3 (MaxPooling  (None, 63, 63, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 127008)            0

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

In [125]:
```
32*(3*3*3+1)
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
```

In [126]:
```
model.add(Dense(6,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['a
ccuracy'])
len(x_train)
```

Out[126]:
```
225
```

In [127]: `1238/24`

Out[127]: `51.583333333333336`

In [ ]:

```
model.fit(x_train,steps_per_epoch=len(x_train),validation_data=x_test,valid
ation_steps=len(x_test),epochs=10)
Epoch 1/10
225/225 [==============================] - 1121s 5s/step - loss: 0.9793 -
accuracy: 0.7894 - val_loss: 0.2668 - val_accuracy: 0.9081
Epoch 2/10
225/225 [==============================] - 172s 762ms/step - loss: 0.2471
- accuracy: 0.9156 - val_loss: 0.1873 - val_accuracy: 0.9365
Epoch 3/10
225/225 [==============================] - 167s 744ms/step - loss: 0.2002
- accuracy: 0.9299 - val_loss: 0.2329 - val_accuracy: 0.9217
Epoch 4/10
225/225 [==============================] - 169s 750ms/step - loss: 0.1448
- accuracy: 0.9462 - val_loss: 0.1871 - val_accuracy: 0.9448
Epoch 5/10
225/225 [==============================] - 173s 768ms/step - loss: 0.1256
- accuracy: 0.9570 - val_loss: 0.2469 - val_accuracy: 0.9199
Epoch 6/10
225/225 [==============================] - 174s 772ms/step - loss: 0.1346
- accuracy: 0.9570 - val_loss: 0.1130 - val_accuracy: 0.9614
Epoch 7/10
225/225 [==============================] - 173s 771ms/step - loss: 0.1263
- accuracy: 0.9590 - val_loss: 0.1862 - val_accuracy: 0.9407
Epoch 8/10
225/225 [==============================] - 170s 755ms/step - loss: 0.0960
- accuracy: 0.9676 - val_loss: 0.1038 - val_accuracy: 0.9644
Epoch 9/10
225/225 [==============================] - 169s 749ms/step - loss: 0.1030
- accuracy: 0.9625 - val_loss: 0.1487 - val_accuracy: 0.9514
Epoch 10/10
225/225 [==============================] - 169s 751ms/step - loss: 0.0839
- accuracy: 0.9711 - val_loss: 0.1949 - val_accuracy: 0.9484
```

In [139]:
```
model.save('fruitdata.h5')
```

In [140]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [142]:
```
model=load_model('fruitdata.h5')
```

In [143]:
```
img=image.load_img(r"/content/0bb2ddc5-d1f4-4fc2-be6b-
6b63c60790df___RS_HL 7550.JPG")
```

```
img
```

Out[143]:

In [144]:

```
img=image.load_img(r"/content/0bb2ddc5-d1f4-4fc2-be6b-6b63c60790df___RS_HL
7550.JPG",target_size=(128,128))
img
```

Out[144]:

In [145]:

```
x=image.img_to_array(img)
x
```

Out[145]:

```
array([[[150., 161., 189.],
        [145., 156., 184.],
        [138., 149., 177.],
        ...,
        [228., 220., 235.],
```

```
 [139., 131., 146.],
 [201., 193., 208.]],

[[145., 156., 184.],
 [150., 161., 189.],
 [140., 151., 179.],
 ...,
 [195., 187., 202.],
 [171., 163., 178.],
 [255., 247., 255.]],

[[141., 152., 180.],
 [137., 148., 176.],
 [140., 151., 179.],
 ...,
 [150., 142., 157.],
 [178., 170., 185.],
 [164., 156., 171.]],

...,

[[157., 172., 203.],
 [155., 170., 201.],
 [148., 163., 194.],
 ...,
 [127., 133., 165.],
 [141., 147., 179.],
 [108., 114., 146.]],

[[161., 176., 207.],
 [162., 177., 208.],
 [159., 174., 205.],
 ...,
 [ 67.,  73., 105.],
 [ 95., 101., 133.],
 [ 86.,  92., 124.]],

[[153., 168., 199.],
 [159., 174., 205.],
 [163., 178., 209.],
 ...,
 [ 93.,  99., 131.],
 [ 92.,  98., 130.],
```

```
        [110., 116., 148.]]], dtype=float32)
```

In [146]:

```
x=np.expand_dims(x,axis=0)
x
```

Out[146]:

```
array([[[[150., 161., 189.],
         [145., 156., 184.],
         [138., 149., 177.],
         ...,
         [228., 220., 235.],
         [139., 131., 146.],
         [201., 193., 208.]],

        [[145., 156., 184.],
         [150., 161., 189.],
         [140., 151., 179.],
         ...,
         [195., 187., 202.],
         [171., 163., 178.],
         [255., 247., 255.]],

        [[141., 152., 180.],
         [137., 148., 176.],
         [140., 151., 179.],
         ...,
         [150., 142., 157.],
         [178., 170., 185.],
         [164., 156., 171.]],

        ...,

        [[157., 172., 203.],
         [155., 170., 201.],
         [148., 163., 194.],
         ...,
         [127., 133., 165.],
         [141., 147., 179.],
         [108., 114., 146.]],

        [[161., 176., 207.],
         [162., 177., 208.],
```

```
         [159., 174., 205.],
         ...,
         [ 67.,  73., 105.],
         [ 95., 101., 133.],
         [ 86.,  92., 124.]],

        [[153., 168., 199.],
         [159., 174., 205.],
         [163., 178., 209.],
         ...,
         [ 93.,  99., 131.],
         [ 92.,  98., 130.],
         [110., 116., 148.]]]], dtype=float32)
```

In [147]:

```
x
```

Out[147]:

```
array([[[[150., 161., 189.],
         [145., 156., 184.],
         [138., 149., 177.],
         ...,
         [228., 220., 235.],
         [139., 131., 146.],
         [201., 193., 208.]],

        [[145., 156., 184.],
         [150., 161., 189.],
         [140., 151., 179.],
         ...,
         [195., 187., 202.],
         [171., 163., 178.],
         [255., 247., 255.]],

        [[141., 152., 180.],
         [137., 148., 176.],
         [140., 151., 179.],
         ...,
         [150., 142., 157.],
         [178., 170., 185.],
         [164., 156., 171.]],

        ...,
```

```
        [[157., 172., 203.],
         [155., 170., 201.],
         [148., 163., 194.],
         ...,
         [127., 133., 165.],
         [141., 147., 179.],
         [108., 114., 146.]],

        [[161., 176., 207.],
         [162., 177., 208.],
         [159., 174., 205.],
         ...,
         [ 67.,  73., 105.],
         [ 95., 101., 133.],
         [ 86.,  92., 124.]],

        [[153., 168., 199.],
         [159., 174., 205.],
         [163., 178., 209.],
         ...,
         [ 93.,  99., 131.],
         [ 92.,  98., 130.],
         [110., 116., 148.]]]], dtype=float32)
```

In [148]:

```
y=np.argmax(model.predict(x),axis=1)
1/1 [==============================] - 0s 97ms/step
```

In [149]:

```
x_train.class_indices
```

Out[149]:

```
{'Apple___Black_rot': 0,
 'Apple___healthy': 1,
 'Corn_(maize)___Northern_Leaf_Blight': 2,
 'Corn_(maize)___healthy': 3,
 'Peach___Bacterial_spot': 4,
 'Peach___healthy': 5}
```

In [150]:

```
index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf
_Blight','Corn_(maize)___healthy','Peach___Bacterial_spot','Peach___health
y']
```

In [151]:
```
index[y[0]]
```

Out[151]:
```
'Corn_(maize)___healthy'
```

In [152]:
```
img=image.load_img(r"/content/drive/MyDrive/Dataset/fruit-dataset/fruit-dataset/test/Apple___healthy/011d02f3-5c3c-4484-a384-b1a0a0dbdec1___RS_HL7544.JPG",target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn_(maize)___healthy','Peach___Bacterial_spot','Peach___healthy']
index[y[0]]
1/1 [==============================] - 0s 39ms/step
```

Out[152]: `'Corn_(maize)___healthy'`

2. Veg.ipynb

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive
```

In [3]:
```
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
```

In [4]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [5]:
```
x_train=train_datagen.flow_from_directory(r"/content/drive/MyDrive/Dataset/Veg-dataset/train_set",target_size=(128,128),
class_mode='categorical',batch_size=24)
Found 11396 images belonging to 9 classes.
```

In [6]:

```
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/Veg-
dataset/test_set',target_size=(128,128),class_mode='categorical',batch_size=24)
Found 3342 images belonging to 9 classes.
```

In [7]:

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

In [8]:

```
model=Sequential()
```

In [9]:

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

In [10]:

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

In [11]:

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

In [12]:

```
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
_________________________________________________________________
```

In [13]:

```
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
```

In [14]: `model.add(Dense(9,activation='softmax'))`

In [15]:

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

In [16]: `len(x_train)`

Out[16]:

```
475
```

In [17]: `1238/24`

Out[17]:

```
51.583333333333336
```

In [ ]:

```
model.fit(x_train,steps_per_epoch=len(x_train),validation_data=x_test,valid
ation_steps=len(x_test),epochs=10)
Epoch 1/10
475/475 [==============================] - 2205s 5s/step - loss: 1.3317 -
accuracy: 0.6106 - val_loss: 0.6637 - val_accuracy: 0.7609
Epoch 2/10
475/475 [==============================] - 339s 713ms/step - loss: 0.6256
- accuracy: 0.7823 - val_loss: 0.4600 - val_accuracy: 0.8360
Epoch 3/10
475/475 [==============================] - 342s 721ms/step - loss: 0.4488
- accuracy: 0.8419 - val_loss: 0.5843 - val_accuracy: 0.7840
Epoch 4/10
475/475 [==============================] - 340s 716ms/step - loss: 0.3748
- accuracy: 0.8673 - val_loss: 0.2802 - val_accuracy: 0.9093
Epoch 5/10
475/475 [==============================] - 343s 720ms/step - loss: 0.3358
- accuracy: 0.8827 - val_loss: 0.2708 - val_accuracy: 0.9057
Epoch 6/10
475/475 [==============================] - 339s 714ms/step - loss: 0.3143
- accuracy: 0.8896 - val_loss: 0.1795 - val_accuracy: 0.9381
Epoch 7/10
475/475 [==============================] - 341s 718ms/step - loss: 0.2563
- accuracy: 0.9079 - val_loss: 0.2930 - val_accuracy: 0.8917
Epoch 8/10
475/475 [==============================] - 337s 710ms/step - loss: 0.2534
- accuracy: 0.9087 - val_loss: 0.2140 - val_accuracy: 0.9282
Epoch 9/10
475/475 [==============================] - 337s 708ms/step - loss: 0.2396
```

```
- accuracy: 0.9198 - val_loss: 0.2777 - val_accuracy: 0.9042
Epoch 10/10
475/475 [==============================] - 338s 711ms/step - loss: 0.1997
- accuracy: 0.9307 - val_loss: 0.1548 - val_accuracy: 0.9551
```

In [18]:
```
model.save('vegetabledata.h5')
```

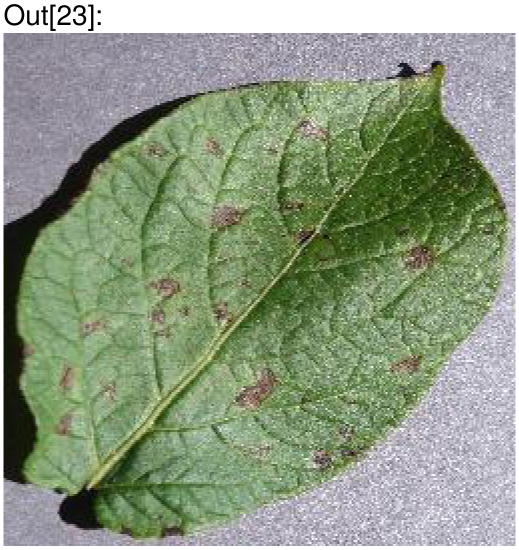
In [19]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [20]:
```
model=load_model('vegetabledata.h5')
```

In [22]:
```
img=image.load_img(r"/content/b817817e-a6b1-4123-88e7-
db98b453ce17___RS_Early.B 6880.JPG")
```

In [23]:
```
img
```

Out[23]:



In [24]:
```
x=image.img_to_array(img)
```

In [25]:
```
img=image.load_img(r"/content/b817817e-a6b1-4123-88e7-
db98b453ce17___RS_Early.B 6880.JPG",target_size=(128,128))
img
```

Out[25]:

In [26]: `x=image.img_to_array(img)`

In [27]: `x`

Out[27]:

```
array([[[135., 131., 145.],
        [134., 130., 144.],
        [133., 129., 143.],
        ...,
        [166., 164., 178.],
        [188., 186., 200.],
        [213., 211., 225.]],

       [[141., 137., 151.],
        [139., 135., 149.],
        [128., 124., 138.],
        ...,
        [201., 199., 213.],
        [157., 155., 169.],
        [172., 170., 184.]],

       [[136., 132., 146.],
        [135., 131., 145.],
        [141., 137., 151.],
        ...,
        [166., 164., 178.],
        [169., 167., 181.],
        [166., 164., 178.]],

       ...,

       [[163., 161., 175.],
        [154., 152., 166.],
        [160., 158., 172.],
        ...,
        [203., 201., 214.],
        [221., 219., 232.],
        [207., 205., 218.]],
```

```
        [[148., 146., 160.],
         [165., 163., 177.],
         [152., 150., 164.],
         ...,
         [176., 174., 187.],
         [192., 190., 203.],
         [189., 187., 200.]],

        [[162., 160., 174.],
         [155., 153., 167.],
         [141., 139., 153.],
         ...,
         [180., 178., 191.],
         [190., 188., 201.],
         [191., 189., 202.]]], dtype=float32)
```

In [28]: `x=np.expand_dims(x,axis=0)`

In [29]: `x`

Out[29]:

```
array([[[[135., 131., 145.],
         [134., 130., 144.],
         [133., 129., 143.],
         ...,
         [166., 164., 178.],
         [188., 186., 200.],
         [213., 211., 225.]],

        [[141., 137., 151.],
         [139., 135., 149.],
         [128., 124., 138.],
         ...,
         [201., 199., 213.],
         [157., 155., 169.],
         [172., 170., 184.]],

        [[136., 132., 146.],
         [135., 131., 145.],
         [141., 137., 151.],
         ...,
         [166., 164., 178.],
         [169., 167., 181.],
```

```
         [166., 164., 178.]],

        ...,

        [[163., 161., 175.],
         [154., 152., 166.],
         [160., 158., 172.],
         ...,
         [203., 201., 214.],
         [221., 219., 232.],
         [207., 205., 218.]],

        [[148., 146., 160.],
         [165., 163., 177.],
         [152., 150., 164.],
         ...,
         [176., 174., 187.],
         [192., 190., 203.],
         [189., 187., 200.]],

        [[162., 160., 174.],
         [155., 153., 167.],
         [141., 139., 153.],
         ...,
         [180., 178., 191.],
         [190., 188., 201.],
         [191., 189., 202.]]]], dtype=float32)
```

In [30]:

```
y=np.argmax(model.predict(x),axis=1)
1/1 [==============================] - 0s 432ms/step
```

In [31]:

```
x_train.class_indices
```

Out[31]:

```
{'Pepper,_bell___Bacterial_spot': 0,
 'Pepper,_bell___healthy': 1,
 'Potato___Early_blight': 2,
 'Potato___Late_blight': 3,
 'Potato___healthy': 4,
 'Tomato___Bacterial_spot': 5,
 'Tomato___Late_blight': 6,
```

```
 'Tomato___Leaf_Mold': 7,
 'Tomato___Septoria_leaf_spot': 8}
```

In [32]:

```
index=['Pepper,_bell___Bacterial_spot','Pepper,_bell___healthy','Potato___Early_blight','Potato___Late_blight','Potato___healthy','Tomato___Bacterial_spot','Tomato___Late_blight','Tomato___Leaf_Mold','Tomato___Septoria_leaf_spot']
```

In [ ]:

```
index[y[0]]
```

Out[ ]:

```
'Tomato___Septoria_leaf_spot'
```

In [33]:

```
img=image.load_img(r"/content/b817817e-a6b1-4123-88e7-db98b453ce17___RS_Early.B 6880.JPG",target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['Pepper,_bell___Bacterial_spot','Pepper,_bell___healthy','Potato___Early_blight','Potato___Late_blight','Potato___healthy','Tomato___Bacterial_spot','Tomato___Leaf_Mold','Tomato___Septoria_leaf_spot']
index[y[0]]
1/1 [==============================] - 0s 53ms/step
```

Out[33]:

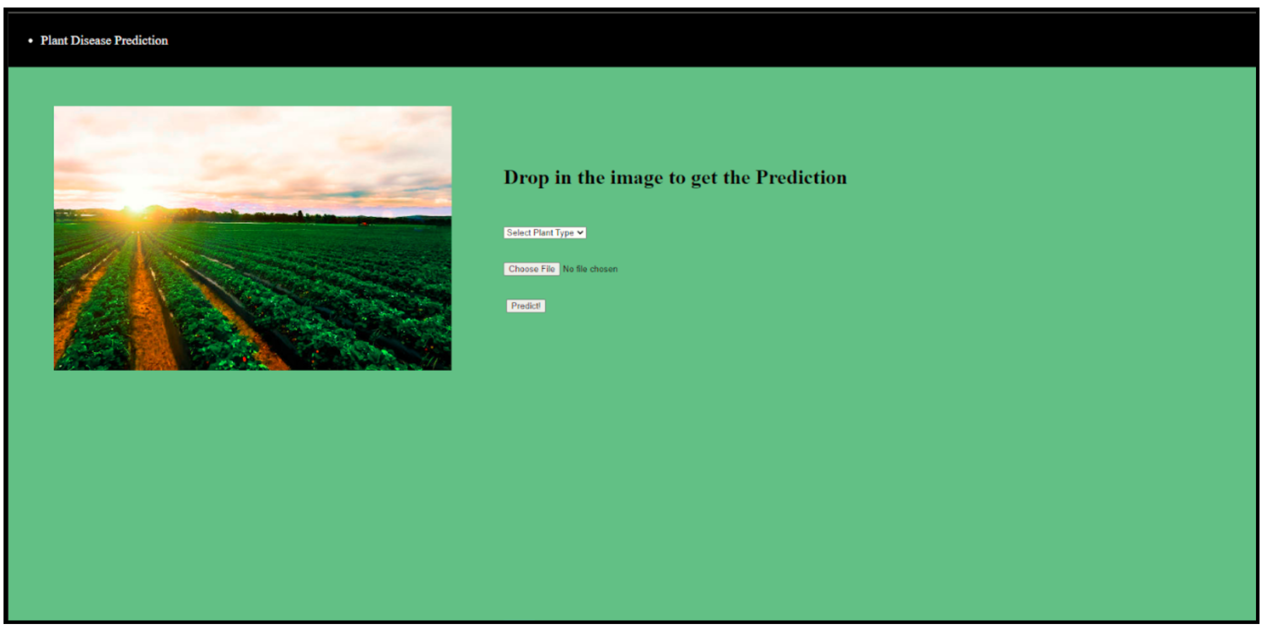
```
'Tomato___Septoria_leaf_spot'
```

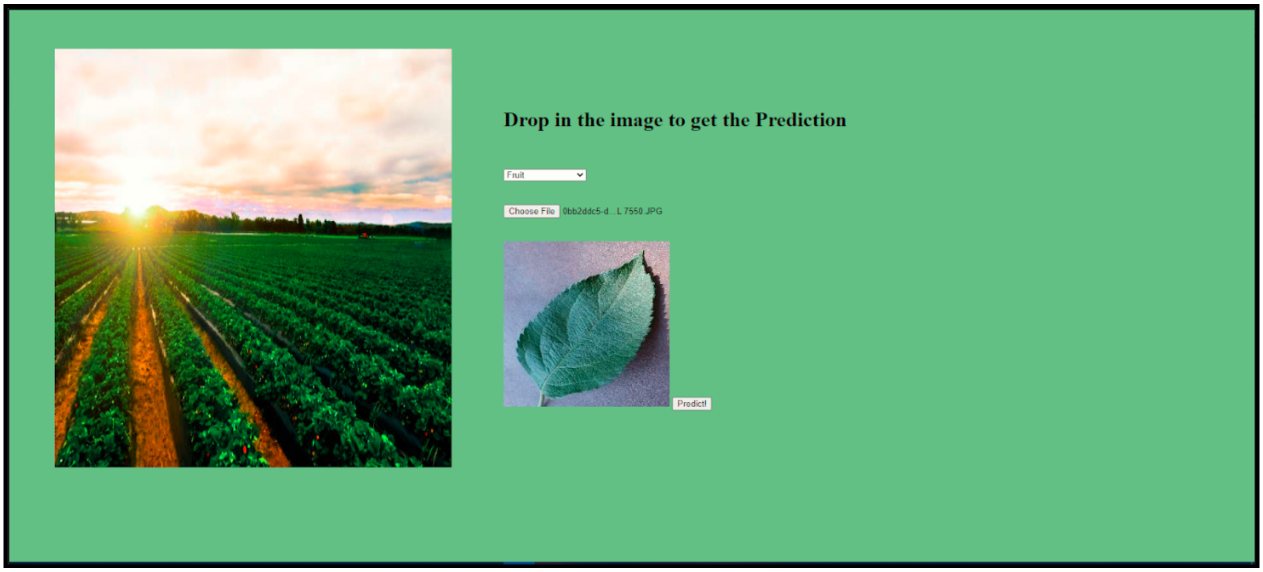
GitHub & Project Demo Link

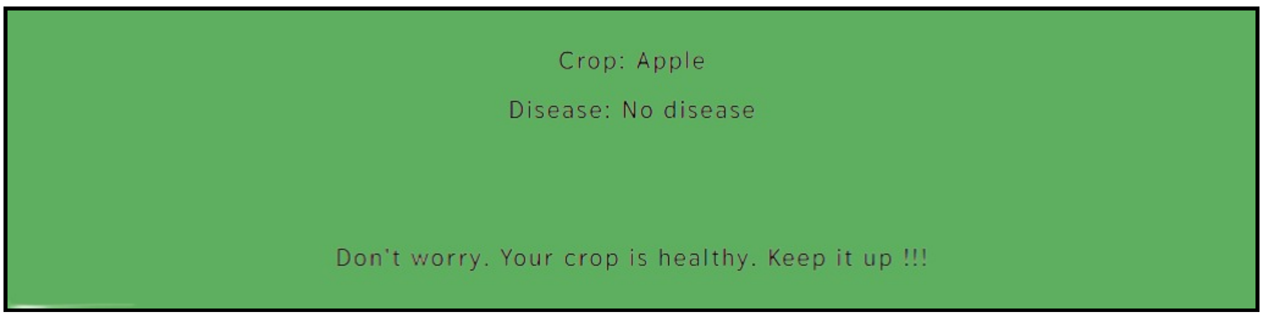
Github Link - https://github.com/IBM-EPBL/IBM-Project-47218-1660797299

Project Demo


Plant Disease Prediction
Home
Predict
Detect if your plant is infected!!
Agriculture is one of the major sectors works wide.Over the years it has developed and the use of new technologies and equipment replaced almost all the traditional methods of farming.The plant diseases effect the production.Identification of diseases and taking necessary precautions is all done through naked eye,which requires labour and laboratries.This application helps farmers in detecting the diseases by observing the spots on the leaves ,which inturn saves effort and labor costs.


Plant Disease Prediction
Drop in the image to get the Prediction
Select Plant Type
Choose File
No file chosen
Predict!


Drop in the image to get the Prediction
Fruit
Choose File
Predict!


Crop: Apple
Disease: No disease
Don't worry. Your crop is healthy. Keep it up !!!