PROJECT REPORT

Fertilizers Recommendation System

For Disease Prediction

By team : PNT2022TMID35128

ROHINI COLLEGE ENGINEERING OF TECHNOLOGY

SEBIN AGNES.A(963319106088)
RAJA SUBHA.R(9633191060)
SHUNMUGA PRIYA.M(963319106094)
VANMATHI NISHA.P(963319106119)

Under the guidance of

Industry Mentor Name: Durga Prasad
Faculty Mentor Name: S.Abisha

INTRODUCTION

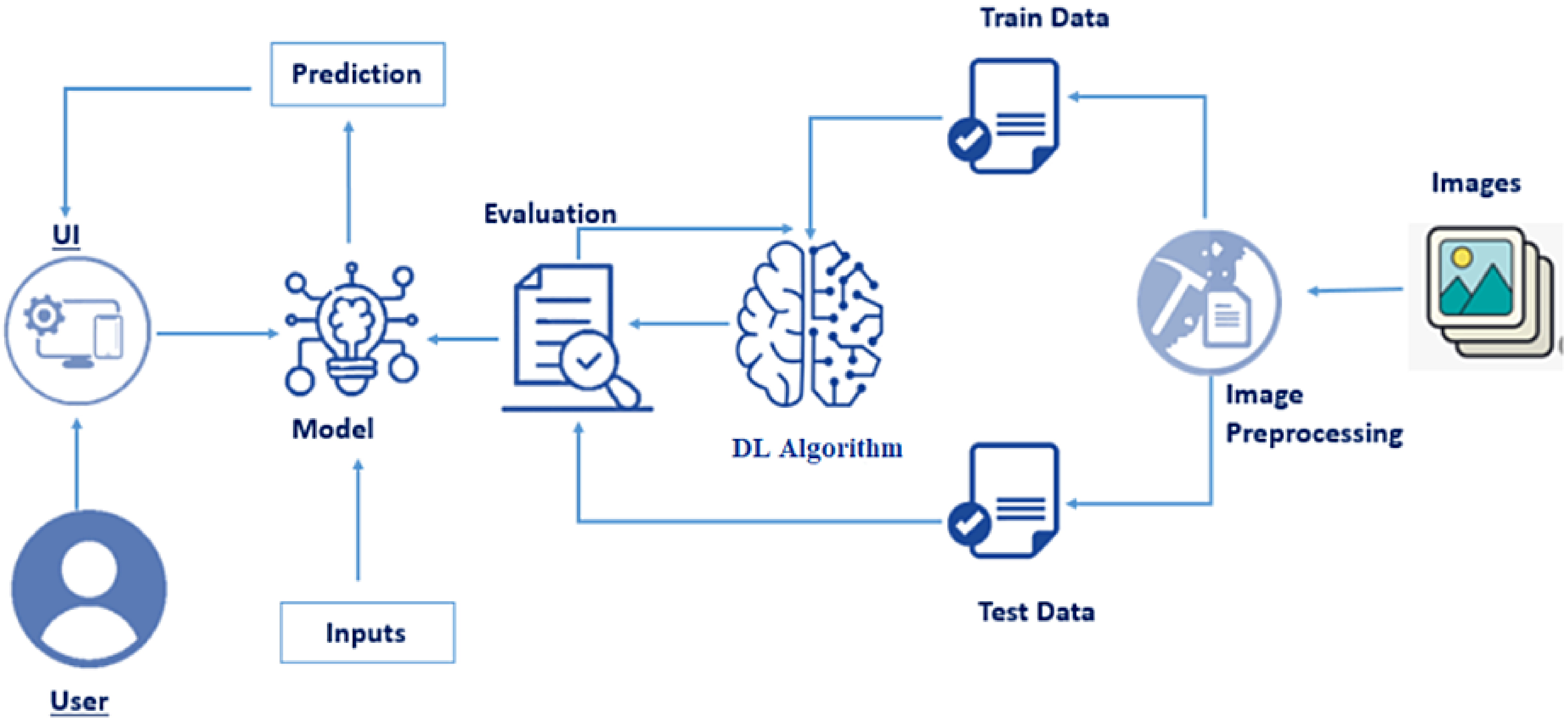
Project Overview

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.

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

Technical Architecture



Purpose

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

Project Objective

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

- An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

Project Flow

- A web Application is built where,
  - ○ Farmers can interact with the portal build.
  - ○ Interacts with the user interface to upload images of diseased leaf.
  - ○ Our model built analyses the Disease and suggests the farmer with fertilizers are to be used
- To accomplish the above task you must complete the below activities and tasks;

Download the dataset.

Classify the dataset into train and test sets.

Add the neural network layers.

Load the trained images and fit the model.

Test the model.

Save the model and its dependencies.

Build a Web application using a flask that integrates with the model built.

Prior Knowledge

Supervised and unsupervised learning:

In Supervised Learning, a machine is trained using ' labeled' data. Datasets are said to be labeled when they contain both input and output parameters. In other words, the data has already been tagged with the correct answer.

Unsupervised learning, also known uses machine learning algorithms to analyze and cluster unlabeled datasets.

These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

Project Structure

The dataset folder contains two folders for the fruit and vegetable dataset which again contains a test and train folder, each of them have images of different diseases.

The Flask folder has all the files necessary to build the flask application. the static folder has the images, style sheets, and scripts that are needed in building the web page. templates folder has the HTML pages. uploads folder has the uploads made by the user. app.py is the python script for server-side computing. .h5 files are the model files that are to be saved after model building. precautions excel files contain the precautions for all kinds of diseases.

Fruit-Training. ipynb, Vegetable-Training, and Plant-Disease-Testing. ipynb are the training and testing notebooks.

Data Collection:

The first step is to download the dataset

Create Train and Test folders with each folder having subfolders with leaf images of different plant diseases. You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc. The folder contains the provided in the project structure section has the link from where you can download datasets that can be used for training. Two datasets will be used, we will be creating two models one to detect vegetable leaf diseases like tomato, potato, and pepper plants and the second model would be for fruits diseases like corn, peach, and apple.

Image Preprocessing

Now that we have all the data collected, let us use this data to train the model. before training the model you have to preprocess the images and then feed them on to the model for training. We make use of Ker as Image Data Generator class for image preprocessing.

Image Pre-processing includes the following main tasks

Import Image Data Generator Library.

Configure Image Data Generator Class.

Applying Image Data Generator functionality to the train set and test set.

Data Collection:

The first step is to download the dataset

Create Train and Test folders with each folder having subfolders with leaf images of different plant diseases. You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc. The folder contains the provided in the project structure section has the link from where you can download datasets that can be used for training. Two datasets will be used, we will be creating two models one to detect vegetable leaf diseases like tomato, potato, and pepper plants and the second model would be for fruits diseases like corn, peach, and apple.

Image Preprocessing

Now that we have all the data collected, let us use this data to train the model. before training the model you have to preprocess the images and then feed them on to the model for training. We make use of Ker as Image Data Generator class for image preprocessing.

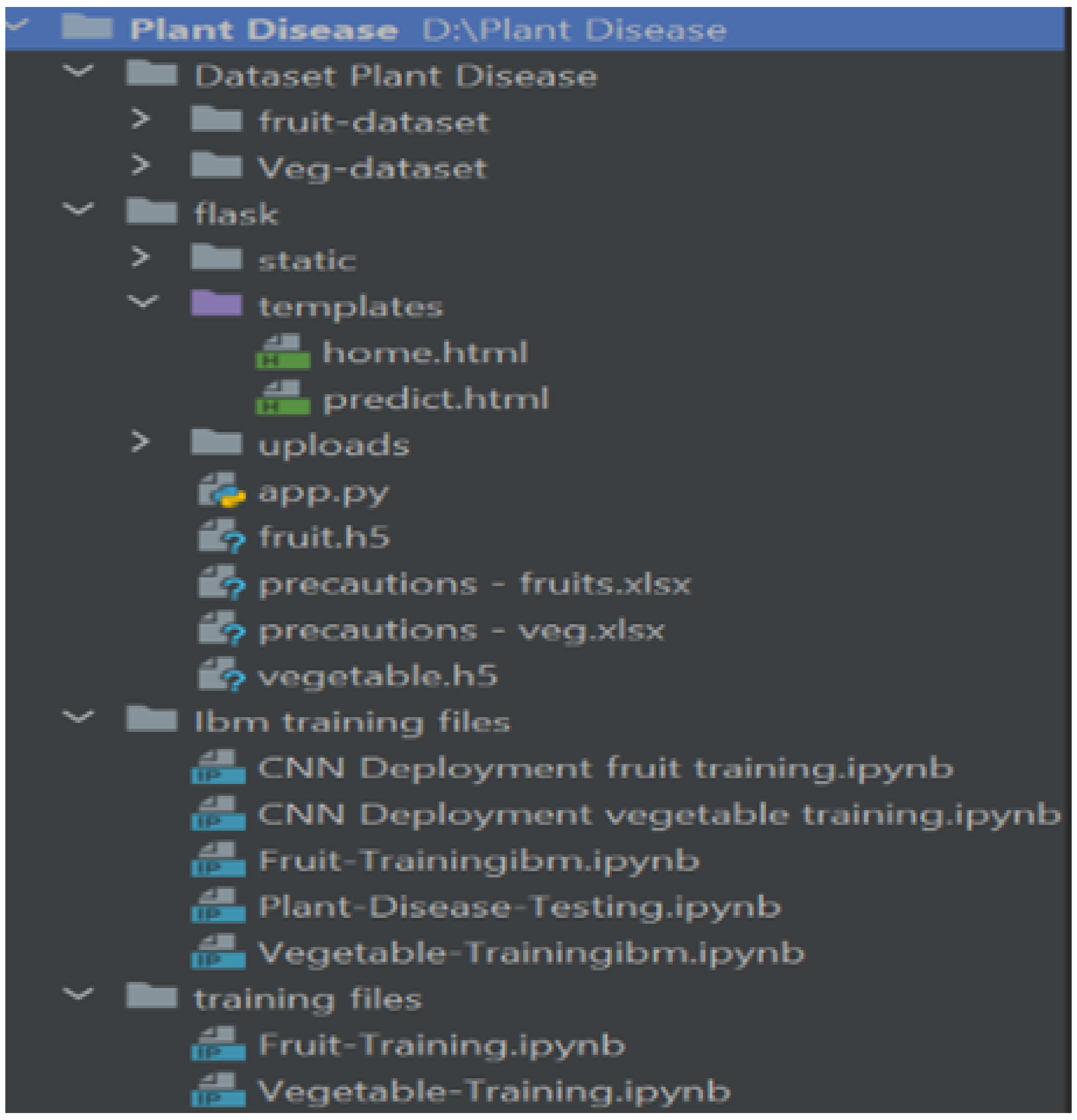
Image Pre-processing includes the following main tasks

Import Image Data Generator Library.

Configure Image Data Generator Class.

Applying Image Data Generator functionality to the train set and test set.

IBM folder contains IBM deployment files



```
Plant Disease  D:\Plant Disease
  Dataset Plant Disease
    > fruit-dataset
    > Veg-dataset
  flask
    > static
    templates
        home.html
        predict.html
    > uploads
      app.py
      fruit.h5
      precautions - fruits.xlsx
      precautions - veg.xlsx
      vegetable.h5
  Ibm training files
      CNN Deployment fruit training.ipynb
      CNN Deployment vegetable training.ipynb
      Fruit-Trainingibm.ipynb
      Plant-Disease-Testing.ipynb
      Vegetable-Trainingibm.ipynb
  training files
      Fruit-Training.ipynb
      Vegetable-Training.ipynb
```

Data Collection:

The first step is to download the dataset

Create Train and Test folders with each folder having subfolders with leaf images of different plant diseases. You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc. The folder contains the provided in the project structure section has the link from where you can download datasets that can be used for training. Two datasets will be used, we will be creating two models one to detect vegetable leaf diseases like tomato, potato, and pepper plants and the second model would be for fruits diseases like corn, peach, and apple.

Image Preprocessing

Now that we have all the data collected, let us use this data to train the model. before training the model you have to preprocess the images and then feed them on to the model for training. We make use of Ker as Image Data Generator class for image preprocessing.

Image Pre-processing includes the following main tasks

Import Image Data Generator Library.

Configure Image Data Generator Class.

Applying Image Data Generator functionality to the train set and test set.

LITERATURE SURVEY

Existing problem

Indumathi proposed a method for leaf disease detection and suggest fertilizers to cure leaf diseases\1\. But the method involves less number of train and test sets which results in poor accuracy. Pandiselvi \2\ proposed a simple prediction method for soil based fertilizer recommendation system for predicted crop diseases.

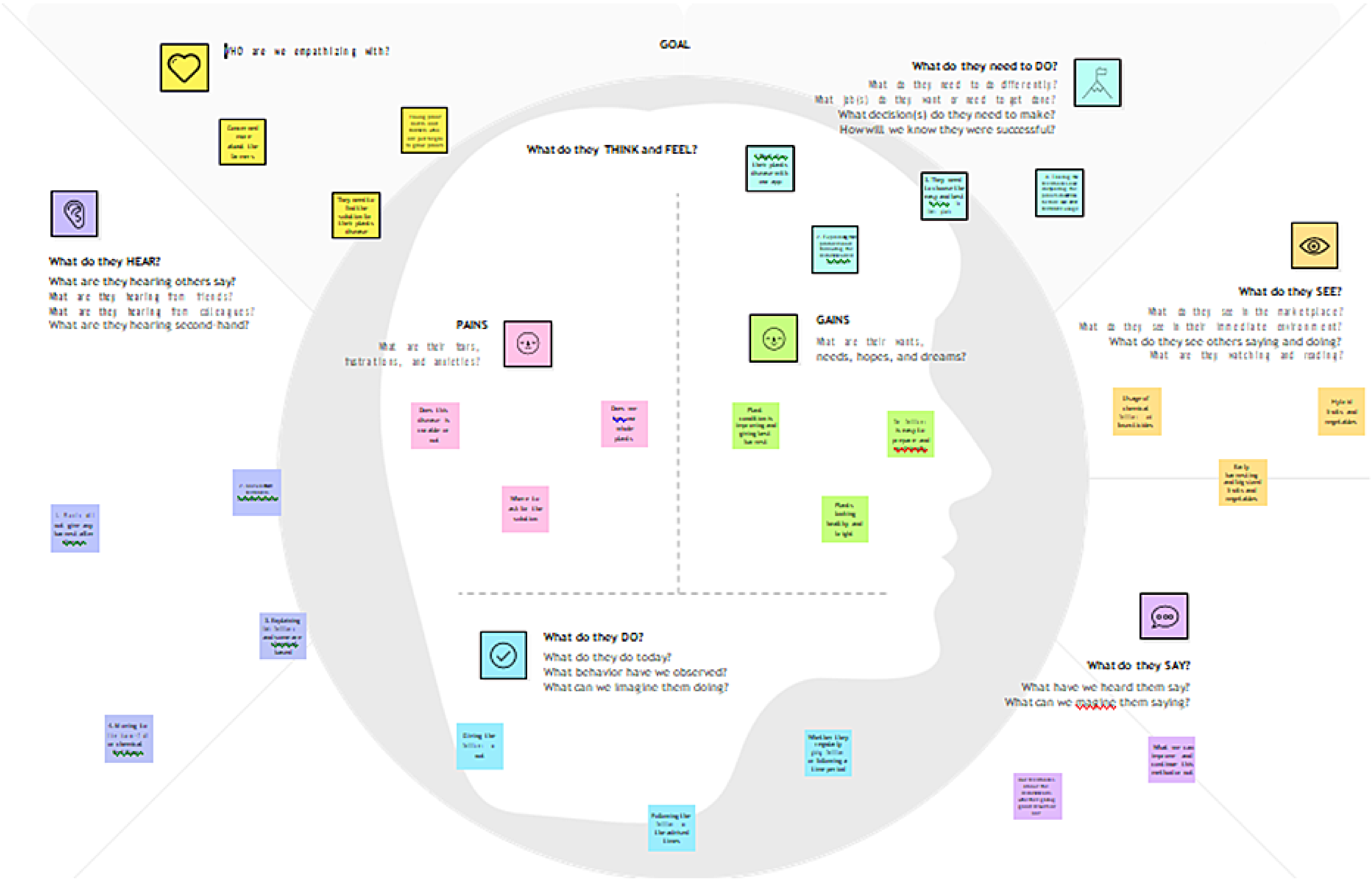
This method gives less accuracy and prediction. Shiva Reddy \3\ proposed an IOT based system for leaf disease detection and fertilizer recommendation which is based on Machine Learning techniques yields less 80 percentage accuracies.

Proposed solution

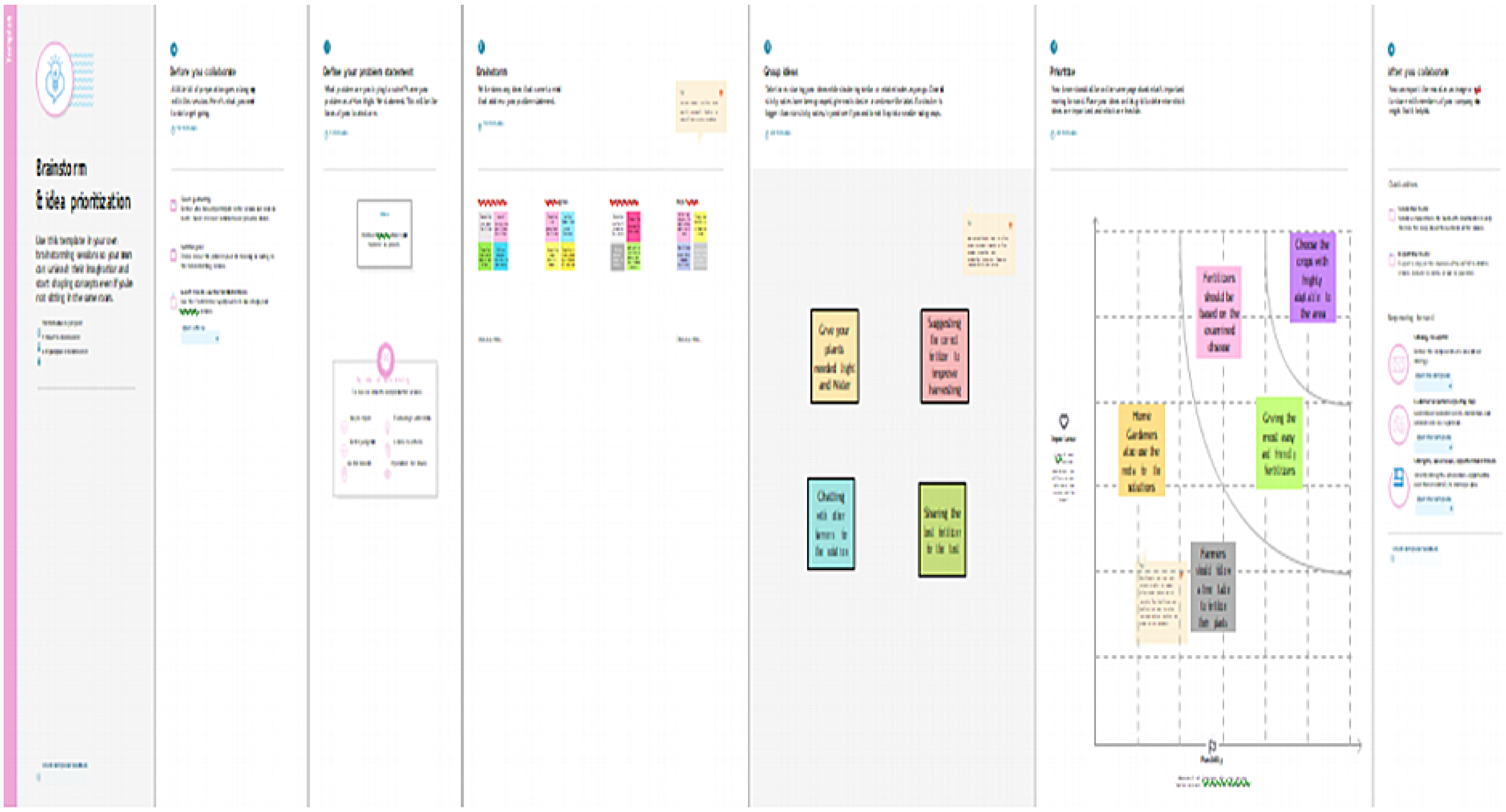
In this project work, a deep learning based neural network is used to train the collected datasets and test the same. The deep learning based neural network is CNN which gives more than 90% classification accuracies. By increasing the more number of dense layers and by modifying hyper parameters such as number of epochs, batch size, the accuracy rate can be increased to 95% to 98%.

IDEATION PHASE

Empathy Map Canvas


WHO are we empathizing with?
GOAL
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
What do they THINK and FEEL?
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
What do they SAY?
What have we heard them say?
What can we imagine them saying?

Ideation and Brainstorming


Brainstorm
& idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
Before you collaborate
Define your problem statement
Brainstorm
Group ideas
Give your plants needed light and Water
Suggesting the correct fertilizer to improve harvesting
Chatting with other farmers for the solution
Sharing the best fertilizer for the land
Prioritize
Fertilizers should be based on the examined disease
Choose the crops with highly available in the area
Home Gardeners also use the mobile to find solutions
Giving the most easy and friendly fertilizers
Farmers should follow a few tips to fertilize their plants
Importance
Feasibility
After you collaborate

PROJECT DESIGN PHASE -1

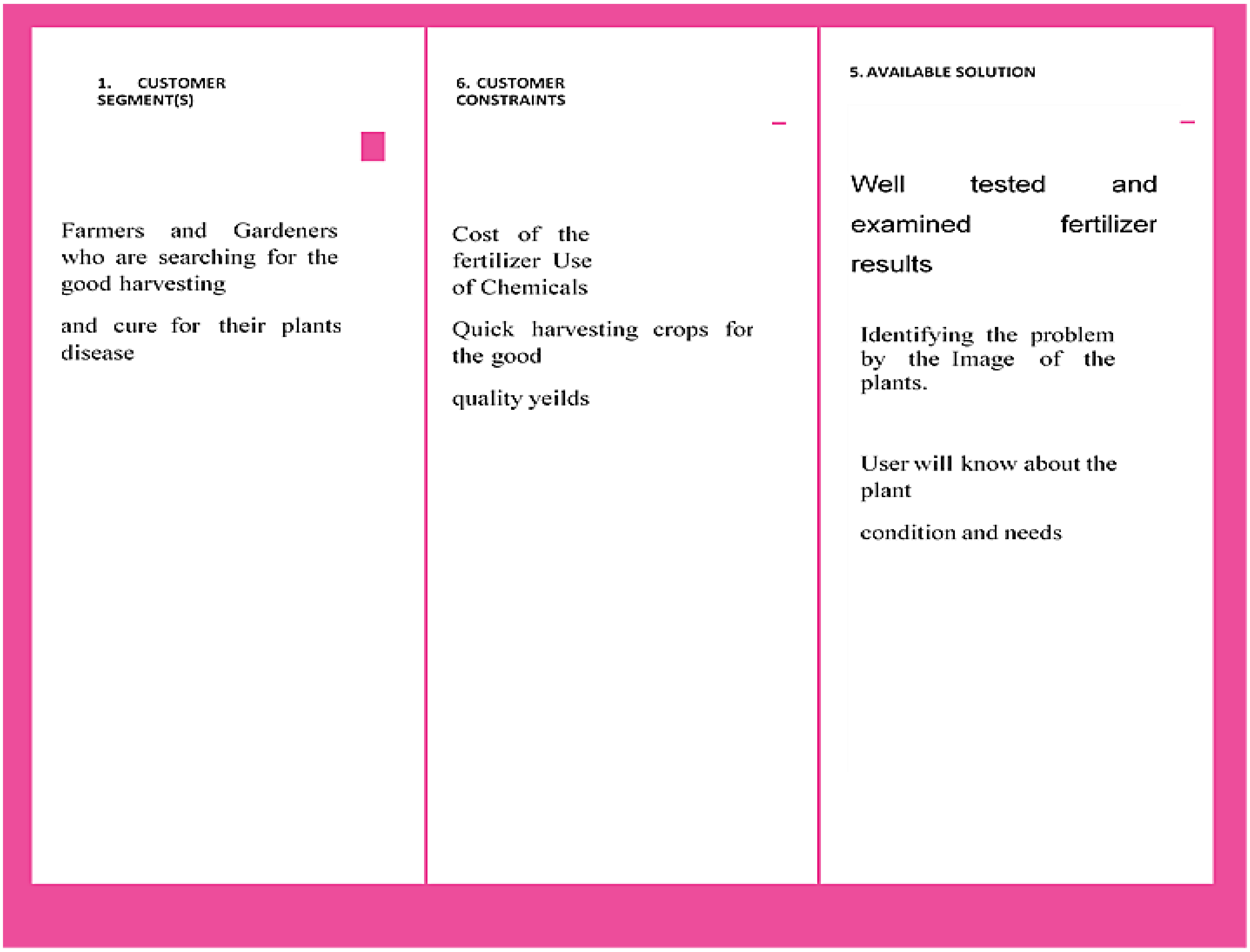
Proposed Solution

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

S.NO	Parameter	Description
1.	ProblemStatement(Problem to be solved)	Disease and pests affecting the plants yield.
2.	Idea / Solution description	Suggesting the best fertilizer to cure the disease

3.	Novelty / Uniqueness	Identifying the plants problem by the image of the plants and comments from the farmers .
4.	SocialImpact/Customer Satisfaction	Giving the examined fertilizer and noticing the Nutrition deficiency.
5.	Business Model (Revenue Model)	Increasing the users by socializing the farmers and gardeners in each area.
6.	Scalability of the Solution	Obtaining the users reviews and feedback for the recommended fertilizer

Problem Solution Fit



1. CUSTOMER SEGMENT(S)

Farmers and Gardeners who are searching for the good harvesting

and cure for their plants disease

6. CUSTOMER CONSTRAINTS

Cost of the fertilizer Use of Chemicals

Quick harvesting crops for the good

quality yeilds

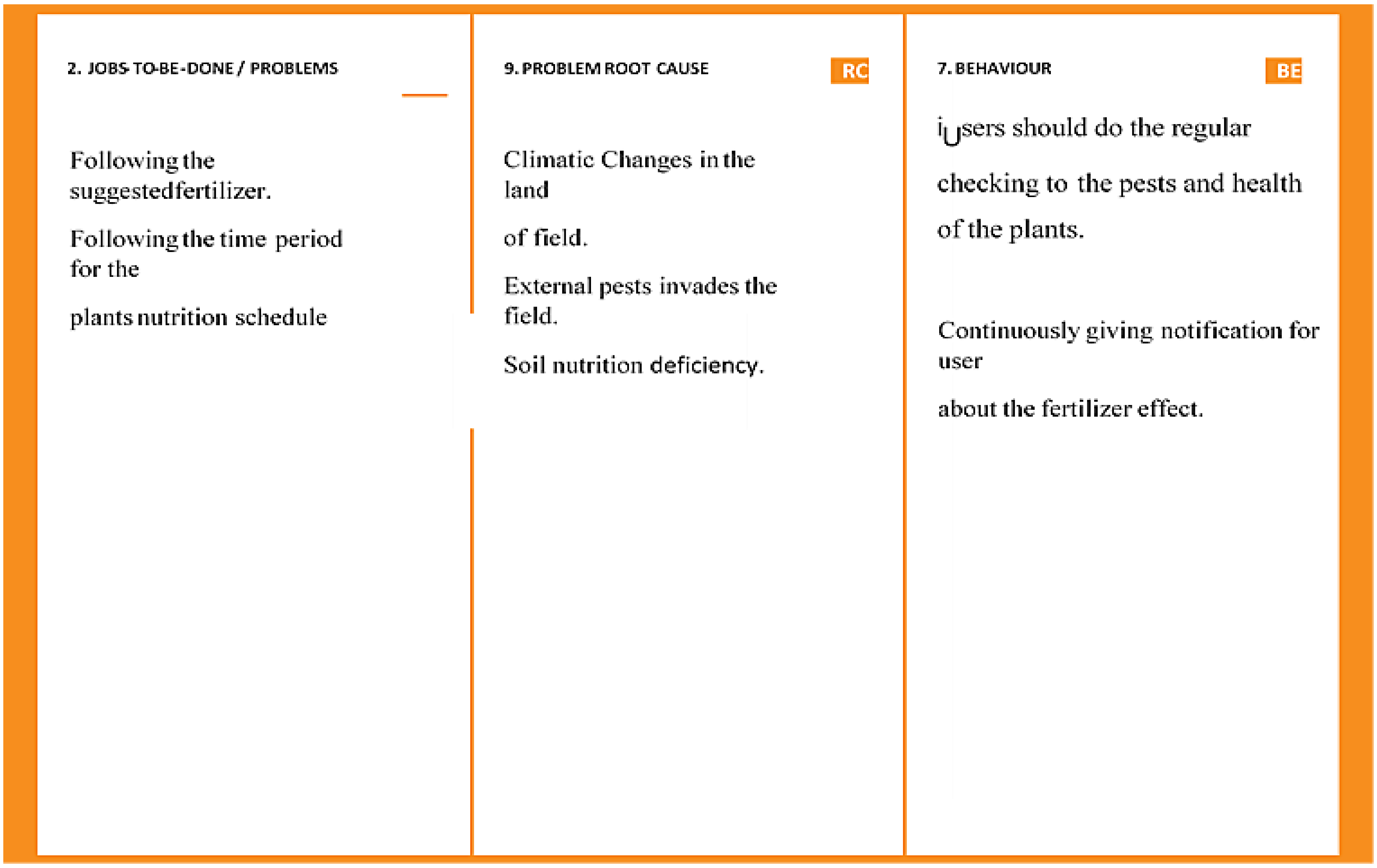
5. AVAILABLE SOLUTION

Well tested and examined fertilizer results

Identifying the problem by the Image of the plants.

User will know about the plant

condition and needs



2. JOBS-TO-BE-DONE / PROBLEMS

Following the suggestedfertilizer.

Following the time period for the

plants nutrition schedule

9. PROBLEM ROOT CAUSE

RC

Climatic Changes in the land

of field.

External pests invades the field.

Soil nutrition deficiency.

7. BEHAVIOUR

BE

iUsers should do the regular

checking to the pests and health

of the plants.

Continuously giving notification for user

about the fertilizer effect.

3. TRIGGERS TR

Continuously givingthe notification. thatremember the timing for the fertilizer.

4. EMOTIONS: BEFORE / AFTER

10. YOUR SOLUTION SL

Suggesting the tested fertilizer and consultingthe other farmers.

Suggesting the good quality seeds and examiningthesoil.

8. CHANNELS of BEHAVIOUR

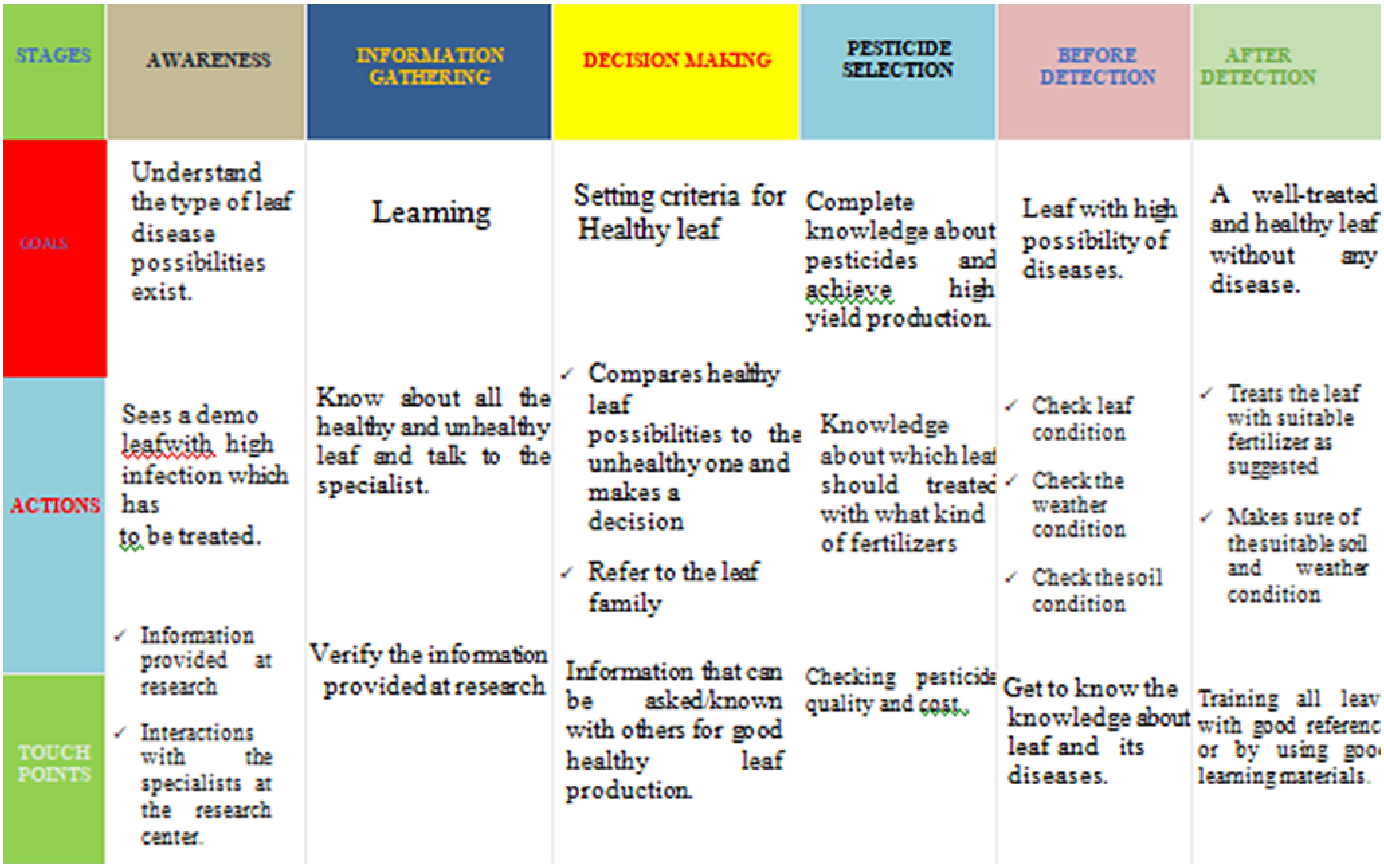
Giving thenotification for theuser about theScheduled fertilizer.

Following the recommended fertilizer for the plants.

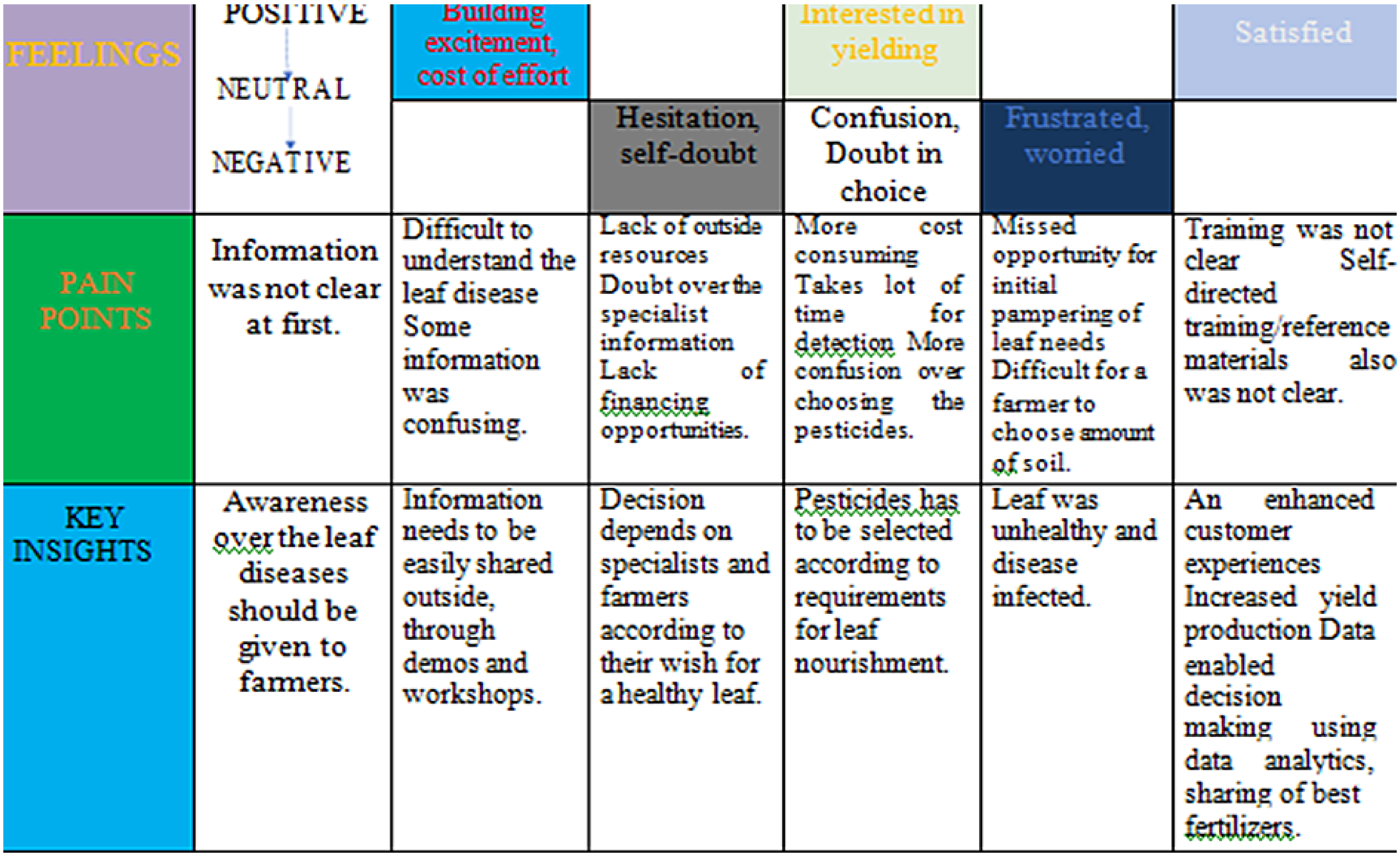
EM Worrying aboutthe plant condition and curing Methods. Excited about the improvement in the plants health.		

Requirement Analysis

Functional Requirement



STAGES	AWARENESS	INFORMATION GATHERING	DECISION MAKING	PESTICIDE SELECTION	BEFORE DETECTION	AFTER DETECTION
GOALS	Understand the type of leaf disease possibilities exist.	Learning	Setting criteria for Healthy leaf	Complete knowledge about pesticides and achieve high yield production.	Leaf with high possibility of diseases.	A well-treated and healthy leaf without any disease.
ACTIONS	Sees a demo leafwith high infection which has to be treated.	Know about all the healthy and unhealthy leaf and talk to the specialist.	✓ Compares healthy leaf possibilities to the unhealthy one and makes a decision ✓ Refer to the leaf family	Knowledge about which lea should treate with what kind of fertilizers	✓ Check leaf condition ✓ Check the weather condition ✓ Check the soil condition	✓ Treats the leaf with suitable fertilizer as suggested ✓ Makes sure of the suitable soil and weather condition
TOUCH POINTS	✓ Information provided at research ✓ Interactions with the specialists at the research center.	Verify the information provided at research	Information that can be asked/known with others for good healthy leaf production.	Checking pesticid quality and cost.	Get to know the knowledge about leaf and its diseases.	Training all leav with good referenc or by using goo learning materials.

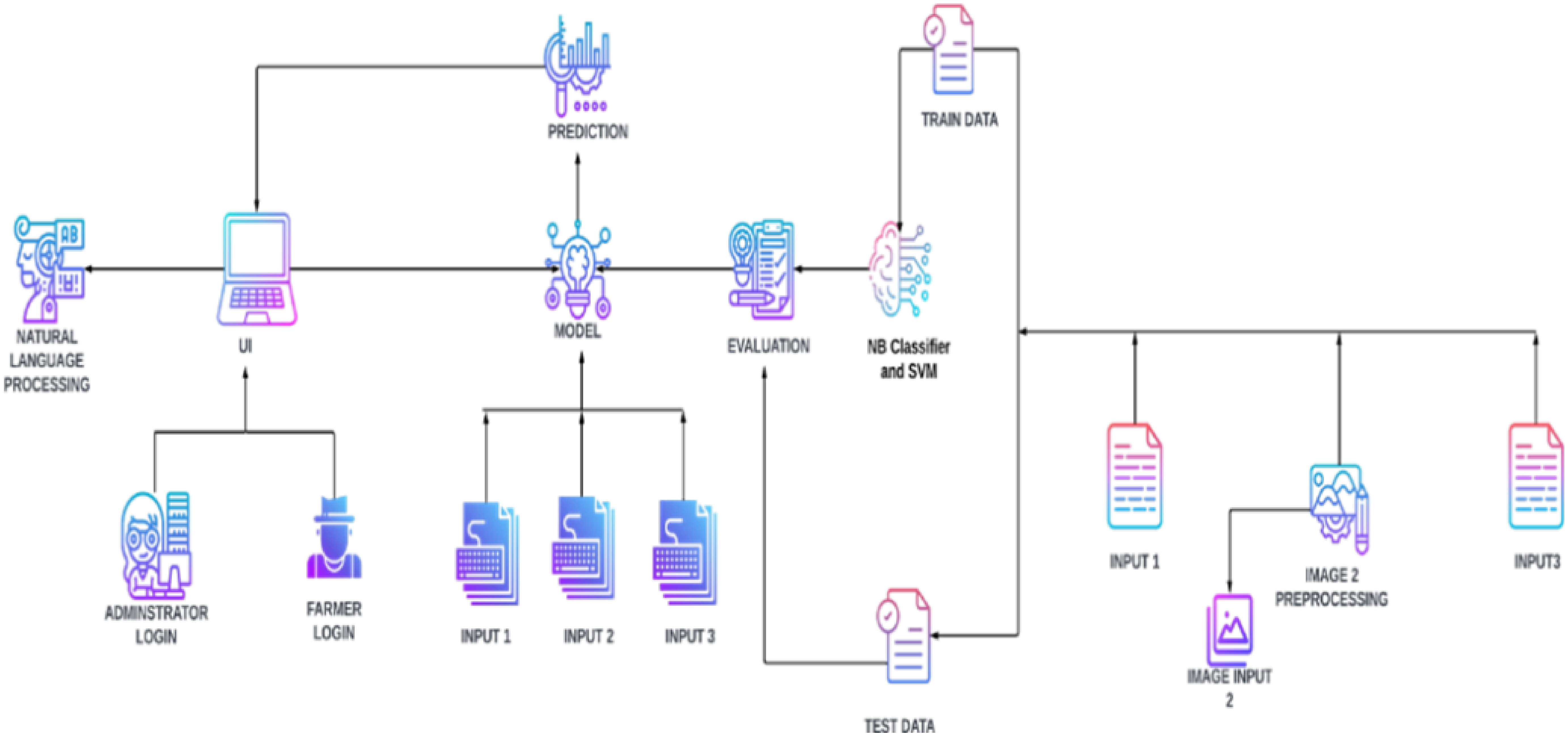


FEELINGS	POSITIVE NEUTRAL NEGATIVE	Building excitement, cost of effort		Interested in yielding		Satisfied
			Hesitation, self-doubt	Confusion, Doubt in choice	Frustrated, worried	
PAIN POINTS	Information was not clear at first.	Difficult to understand the leaf disease Some information was confusing.	Lack of outside resources Doubt over the specialist information Lack of financing opportunities.	More cost consuming Takes lot of time for detection More confusion over choosing the pesticides.	Missed opportunity for initial pampering of leaf needs Difficult for a farmer to choose amount of soil.	Training was not clear Self-directed training/reference materials also was not clear.
KEY INSIGHTS	Awareness over the leaf diseases should be given to farmers.	Information needs to be easily shared outside, through demos and workshops.	Decision depends on specialists and farmers according to their wish for a healthy leaf.	Pesticides has to be selected according to requirements for leaf nourishment.	Leaf was unhealthy and disease infected.	An enhanced customer experiences Increased yield production Data enabled decision making using data analytics, sharing of best fertilizers.

Project Design Phase-II

Data Flow Diagram

Table-1: Components & Technologies



S.NO	Component	Description	Technology
1.	User Interface	How the user interacts with the application .To depict the human-computer interaction and communication.	HTML, CSS,JSP

2.	Application Logic-1	A page to upload images as input	Python
3.	Application Logic-2	To use the Machine Learning model and predicting the result	Python
4.	Database	Structured data-images	CNN
5.	Cloud Database	Database that typically runs on a cloud computing platform and access to the database is provided as-a- service	IBM Cloud Databases for MySQL

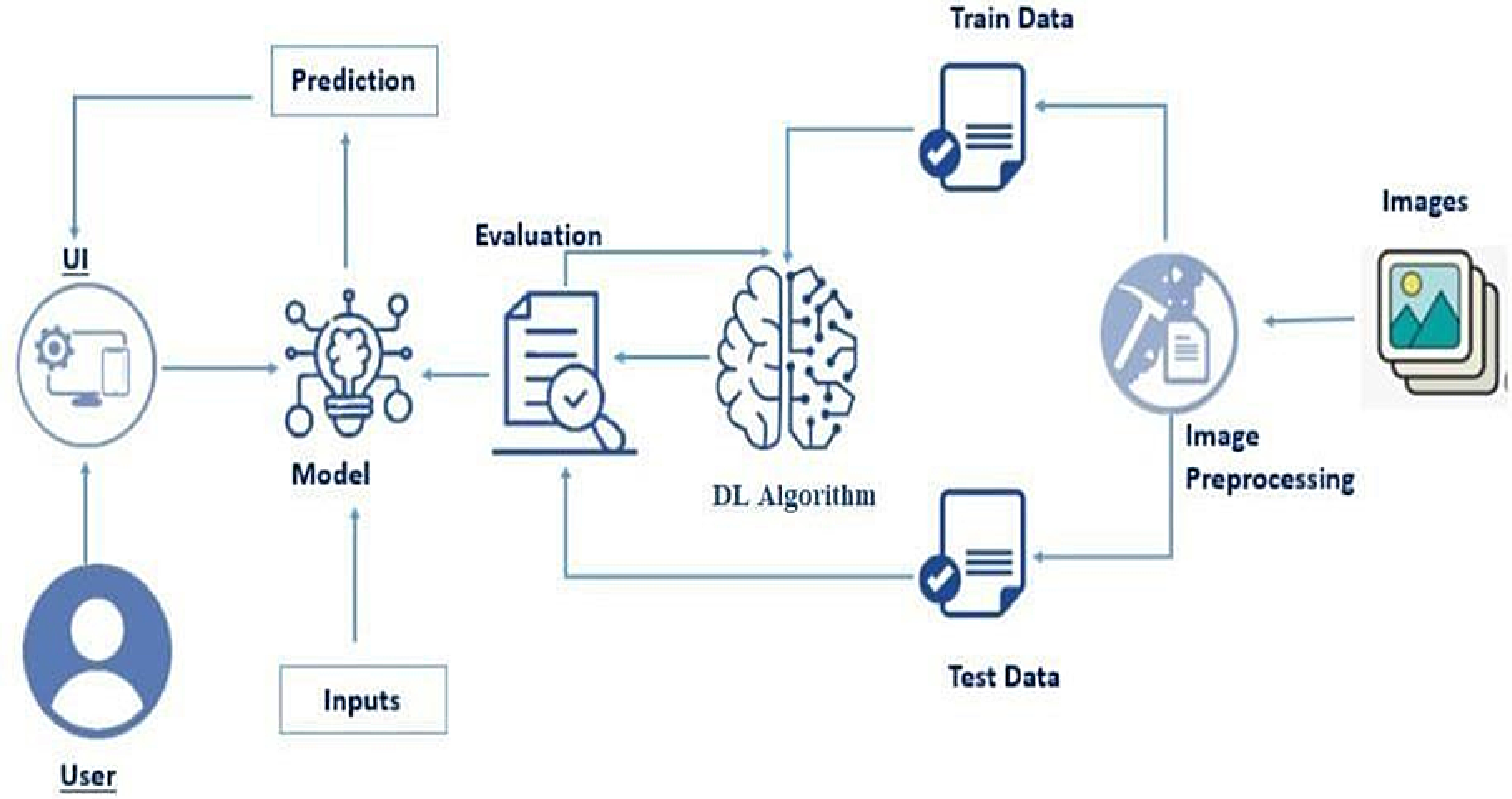
6.	FileStorage	To storestructure data in a hierarchical	LocalFile system
7.	Machine Learning Model	Here, we use a Support VectorMachine Algorithm that is used widelyin Classification and Regression problems.	Random Forest ,XG Boost

Table-2: Application Characteristics

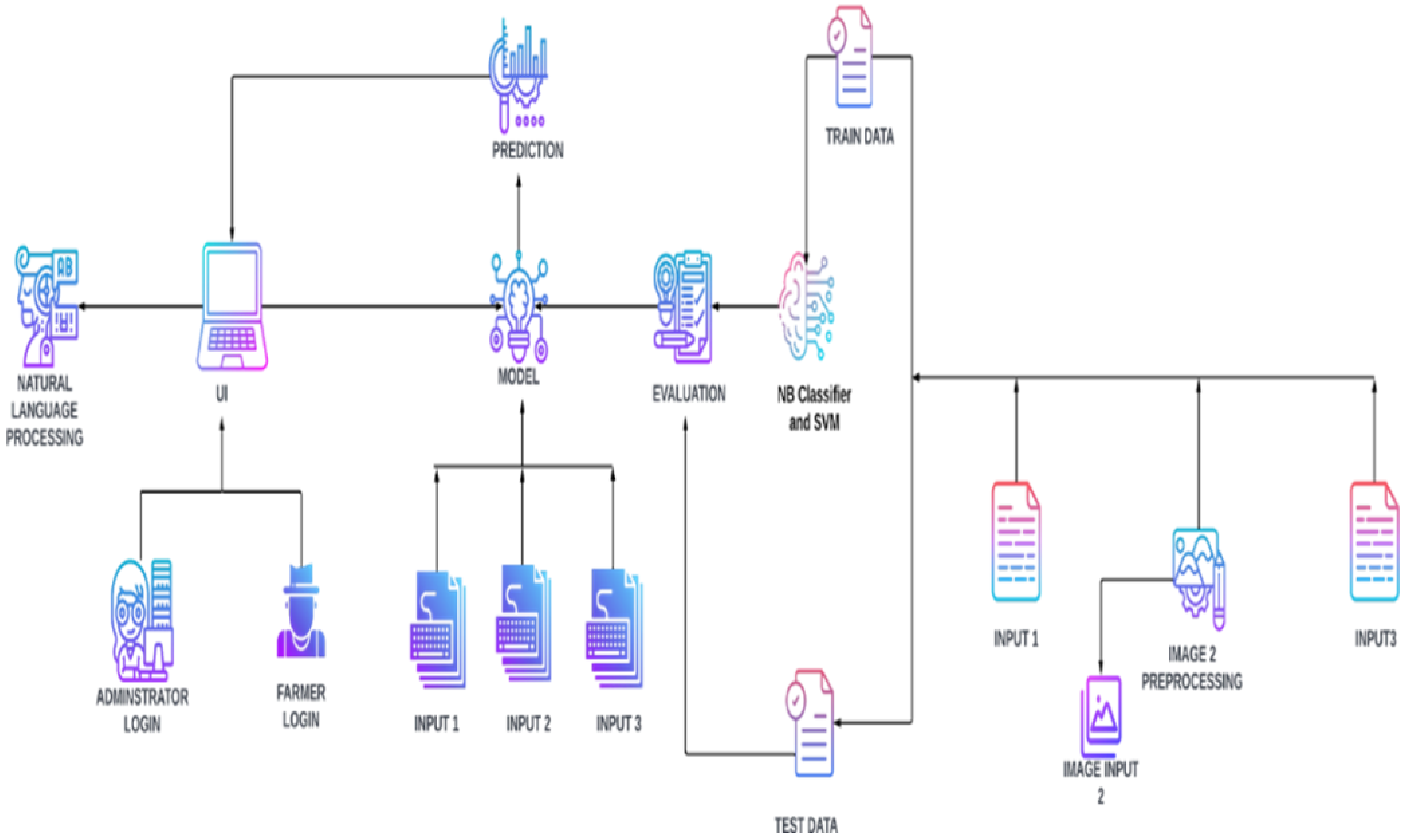
S. No	Characteristics	Description	Technology
1.	Open-SourceFrameworks	Flask micro web framework	Written in Python. It is classified as a micro frame work because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where preexisting third- party libraries provide commonfunctions.
2.	Security Implementations	With all aspects of the job, including detectingmalicious attacks, analyzing the network, endpoint protection an vulnerability assessment, Sign i encryption	IBM CloudApp ID Services

3.	Availability	Available for alldata size	-
4.	Performance	Can extend the storage according to our needs	Python,Angular JS

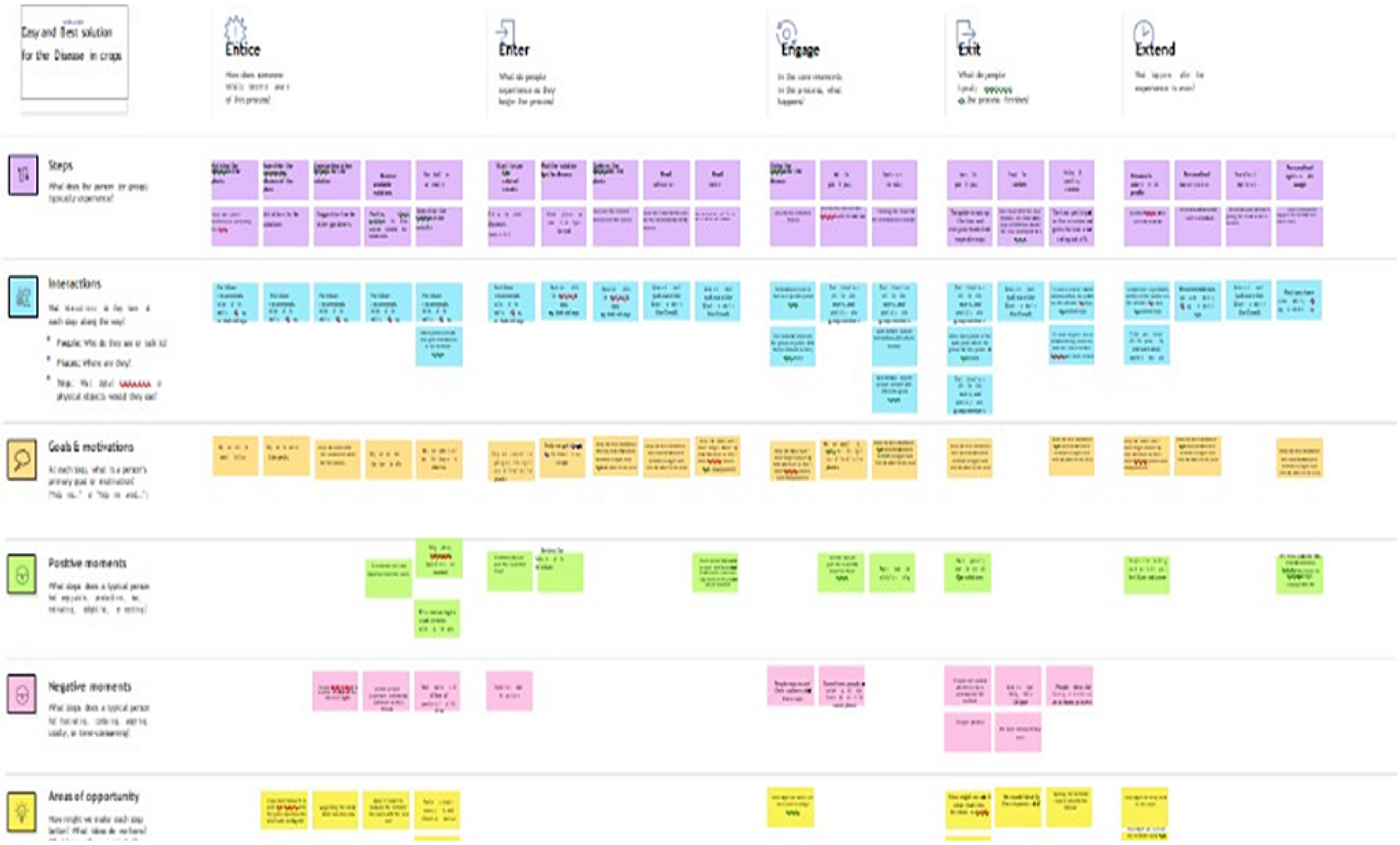
Solution Architecture



Technical Architecture



User Stories


Easy and Best solution for the Disease in crops
Entice
How does someone initially become aware of this process?
Enter
What do people experience as they begin the process?
Engage
In the core moments in the process, what happens?
Exit
What do people typically experience as the process finishes?
Extend
What happens after the experience is over?
Steps
What does the person (or group) typically experience?
Interactions
What interactions do they have at each step along the way?
People: Who do they see or talk to?
Places: Where are they?
Things: What digital touchpoints or physical objects would they use?
Goals & motivations
At each step, what is a person's primary goal or motivation? ("Help me..." or "Help me avoid...")
Positive moments
What steps does a typical person find enjoyable, productive, fun, motivating, delightful, or exciting?
Negative moments
What steps does a typical person find frustrating, confusing, angering, costly, or time-consuming?
Areas of opportunity
How might we make each step better? What ideas do we have?

Project Planning and Scheduling

Milestone and Activity Plan

	MILESTONE	DESCRIPTION	DURATION	STATUS
	Prerequisites	Prerequisites are all the needs at the requirement levelneeded for the execution of the different phasesof a project.	1WEEK	

			IBM Cloud provides solutions		Wheretheproject'skeyfeatures,structure, criteria for success, and major deliverables are
	Create	&	that enable higher levels of compliance, security, and	1WEEK	planned out. The aim is to develop one or more
	IBM		management, with proven		designs that can be used to achieve the desired
	cloud		architecture patterns andmethods		Project goals.
	services		for rapid delivery for running		
			mission-critical workloads.		

	Ideation phase	Ideation is the process where you generate ideas and solutions through sessions such as Sketching, Prototyping, Brainstorming, Brain writing, Worst Possible Idea, and a wealth of other ideation techniques.	1WEEK	COMPLETED

	Projectphases	design	Project design is an early phase of a project		
			where the project'skey features structure, criteria for success, and major deliverables are	1WEEK	COMPLETED
			plannedout.Theaim is to develop one or more		
			designs that can be used to achieve the desired		
			Project goals.		

		In the Planning		

	Project planningphase	Phase, the Project Manager works with the project team to create the technical design, task list, resource plan, communications plan, budget, and initial schedule for the project, and establishes the roles and responsibilities of the project	1WEEK	COMPLETED

	Project development phase	Project development is the process of planning and allocating resources to fully develop a project or product from concept to go-live.	4WEEKS	IN PROGRESS

	Develop the	APython script is a set of commands included in a filethatis intendedtoberun similarlytoa program. The concept isthatthe file will be run or performed from the command line or from within a Python interactive shell to perform a particular activity. Of course, the file includes methodsandimports different modules.	4WEEKS	IN PROGRESS
	Develop web application	A web application (or web app) is application software that runs in a web browser, unlike software programs that run locally and natively on the operating system(OS) of thedevice.	4WEEKS	IN PROGRESS

Sprint Delivery Schedule

Sprint	Functional Requirement(Epic)	User Story Number	User Story/ Task	StoryPoints	Priorty	TeamMembers

Sprint	Functional Requirement(Epic)	User Story Number	User Story/ Task	StoryPoints	Priorty	TeamMembers
Sprint-1	User Confirmation	USN-2	As a farmer, I will receive confirmation email once I haveregisteredforthe application	1	Medium	M.Shunmugapriya P.VanmathiNisha
Sprint-1	Login	USN-3	As a farmer, I can log into theapplication by entering email & password	2	High	A.Sebin Agnes

Sprint-2	Data Collection	USN-1	Collect datasetsfrom differentopenSources. TwoDatasets willbe used, we will be creating twomodels one todetect vegetabledisease and another for fruit Disease.	2	High	R.RajaSubha P.Vanmathi Nisha
Sprint-3	Coding (Data Processing)	USN-1	Coding isa setof instructions used to manipulate	2	High	A.Sebin Agnes R.Raja Subha
Sprint-4	Web Application	USN-1	We notifythe Information about the disease leaf and recommended Fertilizer.	1	Medium	P.VanmathiNisha M. Shunmuga Priya R.Raja Subha

MODEL BUILDING FOR

FRUIT DISEASE PREDICTION

Import the libraries

```
 from keras.preprocessing.image importImageDataGenerator

train_datagen = ImageDataGenerator (rescale = 1./255, shear_range= 0.2,zoom_range= 0.2, horiz

test_datagen =ImageDataGenerator (rescale = 1)

x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-datas

x_test = test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-dataset

Found 5384 images belonging to 6 classes.

Found 1686 images belonging to 6 classes.

x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset

x_test = test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset\tes

Found 11386 images belonging to 9 classes.

Found 3416 images belonging to 9 classes.

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

 model=Sequential()
```

Initializing the model

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator (rescale = 1./255, shear_range= 0.2,zoom_range= 0.2, horiz

test_datagen =ImageDataGenerator (rescale =1)
```

Double-click (or enter) to edit

```
x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-datas

x_test = test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-dataset

Found 5384 images belonging to 6 classes.

Found 1686 images belonging to 6 classes.

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

model=Sequential()

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

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

model.add(Flatten())

model.add(Dense(units=40,kernel_initializer='uniform',activation='relu'))

model.add(Dense(units=70,kernel_initializer='random_uniform',activation='relu'))

model.add(Dense(units=6,kernel_initializer='random_uniform',activation='softmax'))

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

model.fit(x_train,steps_per_epoch=168,epochs=3,validation_data=x_test,validation_steps=5
Epoch 1/3

168/168 [==============================] - 173s 1s/step - loss: 0.8498 - accuracy: 0.68

Epoch 2/3

168/168 [==============================] - 100s 597ms/step - loss: 0.3500 - accuracy: 0

Epoch 3/3

168/168 [==============================] - 102s 606ms/step - loss: 0.2901 - accuracy: 0

<keras.callbacks.History at 0x18b98c7b9d0>
```

```
model.save(r"C:\Users\DELL\Desktop\fruit-dataset.h5")

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

ADD CNN LAYER

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

```
train_datagen = ImageDataGenerator (rescale = 1./255, shear_range= 0.2,zoom_range= 0.2, horiz

test_datagen =ImageDataGenerator (rescale =1)
```

Double-click (or enter) to edit

```
x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-datas

x_test = test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-dataset
```

```
Found 5384 images belonging to 6 classes.

Found 1686 images belonging to 6 classes.
```

```
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

model=Sequential()

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

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

model.add(Flatten())

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

model.save(r"C:\Users\DELL\Desktop\fruit-dataset.h5")

model.summary()
```

```
Model: "sequential_1"
```


```
Layer (type)    Output Shape  Param #
=================================================================

conv2d_1 (Conv2D)   (None, 126, 126, 32)   896


max_pooling2d_1 (MaxPooling          (None, 63, 63, 32)        0

2D)



flatten_1 (Flatten)        (None, 127008)            0



=================================================================

Total params: 896

Trainable params: 896

Non-trainable params: 0
```


ADD DENSE LAYER

```
from keras.preprocessing.image importImageDataGenerator

train_datagen = ImageDataGenerator (rescale = 1./255, shear_range= 0.2,zoom_range= 0.2, horiz

test_datagen =ImageDataGenerator (rescale = 1)
```

Double-click (or enter) to edit

```
x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-datas

x_test = test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-dataset
```

```
Found 5384 images belonging to 6 classes.

Found 1686 images belonging to 6 classes.
```

```
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
```

```
model=Sequential()

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

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

model.add(Flatten())

model.add(Dense(units=40,kernel_initializer='uniform',activation='relu'))

model.add(Dense(units=70,kernel_initializer='random_uniform',activation='relu'))

model.add(Dense(units=6,kernel_initializer='random_uniform',activation='softmax'))

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(x_train,steps_per_epoch=168,epochs=3,validation_data=x_test,validation_steps=52)

Epoch 1/3

168/168 \==============================\ - 173s 1s/step - loss: 0.8498 - accuracy: 0.68

Epoch 2/3

168/168 \==============================\ - 100s 597ms/step - loss: 0.3500 - accuracy: 0

Epoch 3/3

168/168 \==============================\ - 102s 606ms/step - loss: 0.2901 - accuracy: 0

<keras.callbacks.History at 0x18b98c7b9d0>
```

Train and Save the model

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

train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,vertical_test_datagen

=ImageDataGenerator(rescale=1./255)

x_train=train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset\tra

Found 11386 images belonging to 9 classes.

x_test=test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset\test_Found

3416 images belonging to 9 classes.

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten

model=Sequential()

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

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

model.add(Flatten())

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
```


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

model.add(Dense(9,activation='softmax'))

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

len(x_train)356

1238/24

51.583333333333336

model.fit(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_
```

Epoch 1/10 356/356 \============================= =\	- 265s	742ms/step	- loss:	1.3768	- accuracy:	0
Epoch 2/10						
356/356 \============================= =\ Epoch 3/10	- 140s	394ms/step	- loss:	0.5650	- accuracy:	0
356/356 \============================= =\	- 140s	393ms/step	- loss:	0.4425	- accuracy:	0

Epoch 4/10 356/356 \============================= =\	- 224s	629ms/step	- loss:	0.3639	- accuracy:	0
Epoch 5/10 356/356 \============================= =\	- 271s	760ms/step	- loss:	0.3104	- accuracy:	0
Epoch 6/10						
356/356 \============================= =\	- 271s	760ms/step	- loss:	0.2766	- accuracy:	0
Epoch 7/10 356/356 \============================= =\	- 273s	766ms/step	- loss:	0.2531	- accuracy:	0
Epoch 8/10 356/356 \============================= =\	- 327s	917ms/step	- loss:	0.2388	- accuracy:	0
Epoch 9/10						
356/356 \============================= =\ Epoch 10/10	- 274s	770ms/step	- loss:	0.2280	- accuracy:	0

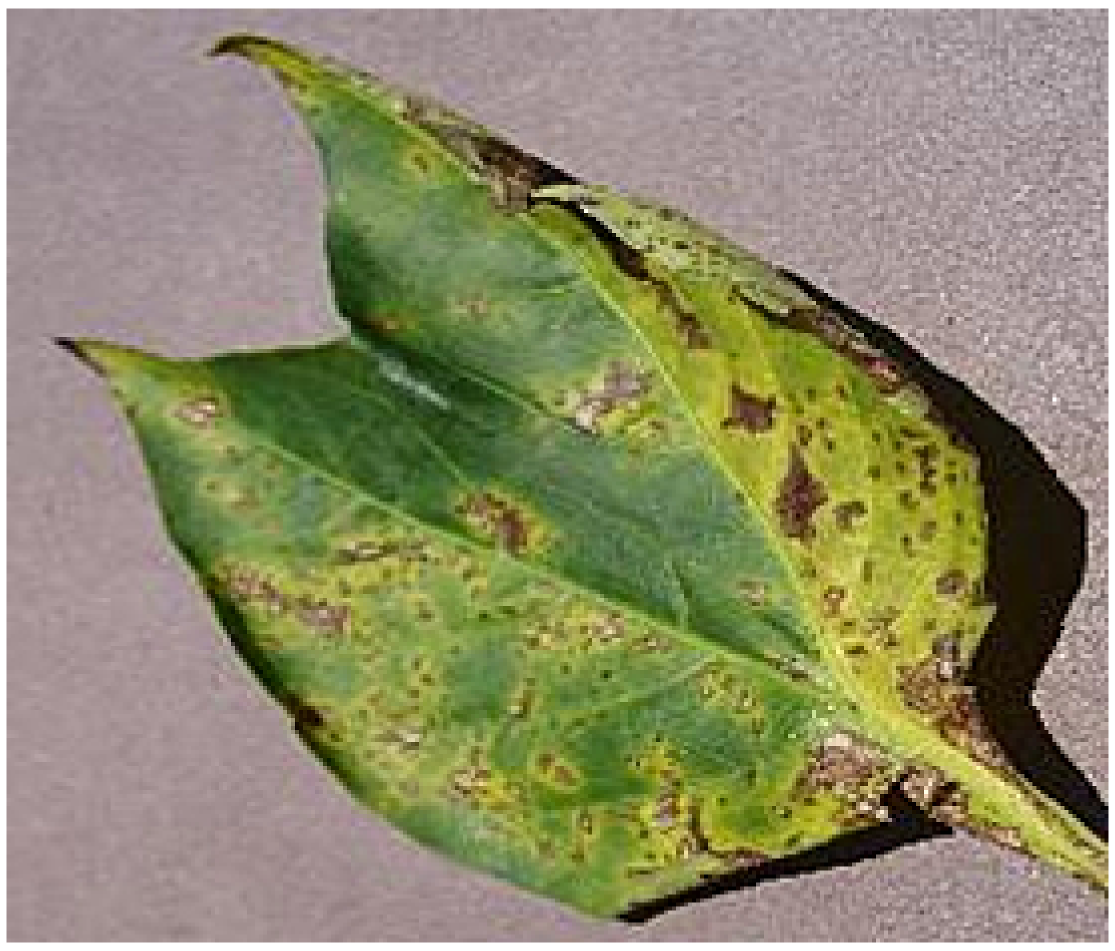
356/356 \============================= =\	- 274s	769ms/step	- loss:	0.2055	- accuracy:	0

```
<keras.callbacks.History at 0x19e7f04a790>
```

```
model.save('Veg-dataset.h5')

import numpy as np

from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

model=load_model('Veg-dataset.h5')

img=image.load_img(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset\test_set\Pepper,_bell__Baimg
```


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

```
img=image.load_img(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset\test_set\Pepper,_bell__Baimg
```




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

```
x array([[[152., 137., 142.],
```

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

xarray([[[[152., 137., 142.],

[144.,	132.,	136.],	
[158.,	146.,	150.]],	
[[153.,	138.,	143.],	
[154.,	139.,	144.],	
[154., ...,	139.,	144.],	
[177.,	165.,	169.],	
[162.,	150.,	154.],	
[170.,	158.,	162.]],	
[[155.,	140.,	145.],	
[155.,	140.,	145.],	
[156., ..., [159.,	141., 147.,	146.], 151.],	
[157.,	145.,	149.],	
[157.,	145.,	149.]],	
...,			
[[158.,	139.,	141.],	
[157.,	138.,	140.],	
[157.,	138.,	140.],	
...,			

\160.,	142.,	142.\,	
\162.,	144.,	144.\,	
\148.,	130.,	130.\\,	
\\160.,	141.,	143.\,	
\159.,	140.,	142.\,	
\158.,	139.,	141.\,	
..., \136.,	 118.,	 116.\,	
\170.,	152.,	150.\,	
\163.,	145.,	143.\\,	
\\165.,	146.,	148.\,	
\163.,	144.,	146.\,	
\161., ...,	142.,	144.\,	
\147.,	129.,	125.\,	
\171.,	153.,	151.\,	
\155.,	137.,	135.\\\,	dtype=float32)

\144.,	132.,	136.\,	
\158.,	146.,	150.\\,	
\\153.,	138.,	143.\,	
\154.,	139.,	144.\,	
\154., ...,	139.,	144.\,	

\177.,	165.,	169.\,	
\162.,	150.,	154.\,	
\170.,	158.,	162.\\,	
\\155.,	140.,	145.\,	
\155.,	140.,	145.\,	
\156., ..., \159.,	141., 147.,	146.\, 151.\,	
\157.,	145.,	149.\,	
\157.,	145.,	149.\\,	
...,			
\\158.,	139.,	141.\,	
\157.,	138.,	140.\,	
\157.,	138.,	140.\,	
..., \160.,	142.,	142.\,	
\162.,	144.,	144.\,	
\148.,	130.,	130.\\,	
\\160.,	141.,	143.\,	
\159.,	140.,	142.\,	
\158.,	139.,	141.\,	
..., \136.,	118.,	116.\,	
\170.,	152.,	150.\,	
\163.,	145.,	143.\\,	

[[165.,	146.,	148.],	
[163.,	144.,	146.],	
[161., ...,	142.,	144.],	
[147.,	129.,	125.],	
[171.,	153.,	151.],	
[155.,	137.,	135.]]]],	dtype=float32)

```
x_train.class_indices
```

```
{'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}
```

APPLICATION BUILDING

BUILD HTML CODE

Index

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>home page</title>
    <style>
        body{
        margin: 0;
        padding: 0;
        }
        .container{
            padding: 30px 70px 30px 70px;


            left: 20px;
            right:20px;
            background-color:rgb(163, 192, 120);
            font-size: 20pt;
            font-family: 'Times New Roman';


        }

        .card{
            font: optional;
            display: flex;

        }
        #h1{
            font-size: 50pt;
        }
        .menu{
            background-color:black;



        }
        #abc{
            color: white;
        }

    </style>
</head>
<body><div class="menu">
    <ul>
                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
                <li id="abc"> plant Disease
Prediction&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<a href="firstpage.html" id="abc"> home</a>
```

```
                    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<a href="predict.html" id="abc">predict</a></li></ul>

    <div class="container" >

        <h1 id="h1"><center><b> Detect if your plant is infected!! </b></center></h1>
        <div class="card" >
          <p > 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.</p>
        <img src="img.jpg" height="300" width="300">
    </div>
    </div>
    </div>
</body>
</html>
```

Predict

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>predict</title>
</head>

<style>
```

```
    .container{
        display: flex;
        padding: 60px 70px 60px 70px;
    }
    .card{
        padding: 70px 80px 70px 80px;
    }
    .menu{
        padding: 10px 10px 10px 10px;
        background-color: black;
        color: white;
        font-size: 15pt;
    }
</style>
<body>
    <div class="menu">
        <ul ><li>Plant disease Prediction</li></ul></div>
    <div class="container">

        <img src="img1.jpg">
        <div class="card">
        <form>
            <h1>Drop in the image to get the Prediction </h1><br><br>
            <label><select name="Fruit" id="plant">
                <option value="fruit" id="fruit">Fruit</option>
                <option value="vagitable" id="vig">vegitable</option>
                </select>
            </label><br><br><br>
            <input id="default-btn" type="file" name="" onchange="document.getElementById('output').src=window.URL.createObjectURL(this.files[0])"><br><br><br>
            <img src="" id="output">

           <button id="button" onclick ="display()" >Predict!</button><br><br>




        </form>
```

```
</body>
</html>
```

RUN THE CODE

Fruit Dataset Output

```
from keras.preprocessing.image importImageDataGenerator

train_datagen = ImageDataGenerator (rescale = 1./255, shear_range= 0.2,zoom_range= 0.2, ho

test_datagen =ImageDataGenerator (rescale = 1)
```

Double-click (or enter) to edit

```
x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-da x_test=
test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-data
Found 5384 images belonging to 6 classes.
Found1686 images belonging to 6 classes.
```

```
from keras.models import
Sequential fromkeras.layers import
Dense
from keras.layers import
Convolution2Dfrom keras.layers import
MaxPooling2D from keras.layers
importFlatten
from tensorflow.keras.models import load_model


model=Sequential()



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


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


model.add(Flatten())



model.add(Dense(units=40,kernel_initializer='uniform',activation='relu'))


model.add(Dense(units=70,kernel_initializer='random_uniform',activation='relu'))


model.add(Dense(units=6,kernel_initializer='random_uniform',activation='softmax'))


model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(x_train,steps_per_epoch=168,epochs=3,validation_data=x_test,validation_steps=52)Epoch 1/3

    168/168 [==============================] - 102s 598ms/step - loss: 0.8219 - accuracy

    Epoch 2/3
    168/168 [==============================] - 101s 600ms/step - loss: 0.3760 - accuracy
    Epoch 3/3
    168/168 [==============================] - 98s 584ms/step - loss: 0.2877 - accuracy:
    <keras.callbacks.History at 0x2a3360db430>
```

```
model.save(r"C:\Users\DELL\Desktop\fruit-dataset.h5")
```

```
model.fit(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len
```

Epoch 1/10 169/169 [============================= =]	-	100s 591ms/step - loss: 0.2747 - accuracy			
Epoch 2/10					
169/169 [============================= =] Epoch 3/10	- 99s	584ms/step	- loss:	0.2373	- accuracy:
169/169 [============================= =] Epoch 4/10 169/169 [============================= =]	- 66s - 51s	386ms/step 304ms/step	1. loss: 2. loss:	0.2049 0.1796	1. accuracy: 2. accuracy:
Epoch 5/10 169/169 [============================= =]	- 51s	299ms/step	- loss:	0.1603	- accuracy:
Epoch 6/10					

169/169 \============================= =\ Epoch 7/10 169/169 \============================= =\	- 46s - 48s	270ms/step 283ms/step	1. loss: 2. loss:	0.1337 0.1304	1. accuracy: 2. accuracy:
Epoch 8/10 169/169 \============================= =\	- 46s	273ms/step	- loss:	0.1310	- accuracy:
Epoch 9/10 169/169 \============================= =\	- 48s	284ms/step	- loss:	0.1168	- accuracy:
Epoch 10/10					
169/169 \============================= =\	- 48s	281ms/step	- loss:	0.1182	- accuracy:

<keras.callbacks.History at 0x2a3395af760>

```
model.save('fruit-dataset.h5')
```

```
model.summary()
```

Model: "sequential"

```
 Layer (type)          Output Shape          Param #
=================================================================
```

conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0

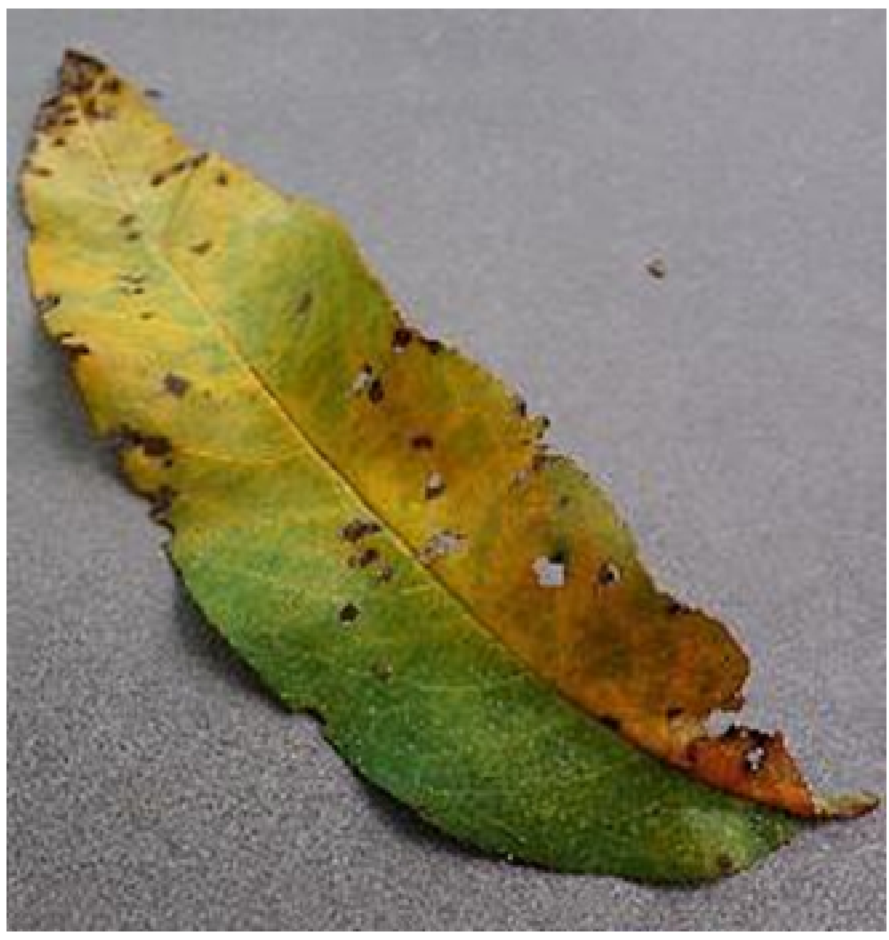
flatten (Flatten)	(None, 127008)	0
dense (Dense)	(None, 40)	5080360
dense_1 (Dense)	(None, 70)	2870
dense_2 (Dense)	(None, 6)	426

```
=================================================================

Total params: 5,084,552
Trainable params: 5,084,552
Non-trainable params: 0
```

```
model=load_model('fruit-dataset.h5')
```

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

```
img=image.load_img(r"C:\Users\DELL\Desktop\test\Peach_Bacterial_spot\0c1c3a6a-3454-4154-img
```



img=image.load_img(r"C:\Users\DELL\Desktop\test\Peach Bacterial_spot\0c1c3a6a-3454-4154-img

x=image.img_to_arra
y(img)x

array([[[148.,	149.,	153.],
[148.,	149.,	153.],
[148.,	149.,	153.],

..., [153.,	150.,	157.],	
[153.,	150.,	157.],	
[153.,	150.,	157.]],	

[[146.,	147.,	151.],	
[146.,	147.,	151.],	
[146., ..., [152.,	147., 149.,	151.], 156.],	
[152.,	149.,	156.],	
[153.,	150.,	157.]],	
[[145.,	146.,	150.],	
[145.,	146.,	150.],	
[145.,	146.,	150.],	
..., [150.,	147.,	154.],	
[152.,	149.,	156.],	
[154.,	151.,	158.]],	
...,			
[[122.,	118.,	119.],	
[131.,	127.,	128.],	
[103., ...,	99.,	100.],	
[141.,	136.,	140.],	
[120.,	115.,	119.],	
[139.,	134.,	138.]],	
[[134.,	130.,	131.],	
[82.,	78.,	79.],	
[108., ...,	104.,	105.],	
[129.,	124.,	128.],	
[130.,	125.,	129.],	
[133.,	128.,	132.]],	
[[127.,	123.,	124.],	
[121.,	117.,	118.],	
[87., ..., [119.,	83., 114.,	84.], 118.],	
[127.,	122.,	126.],	
[135.,	130.,	134.]]],	dtype=float32)

x=np.expand_dims(x,axi
s=0)x

array([[[[148.,	149.,	153.],
[148.,	149.,	153.],
[148.,	149.,	153.],
..., [153.,	150.,	157.],
[153.,	150.,	157.],
[153.,	150.,	157.]],
[[146.,	147.,	151.],

[146.,	147.,	151.],	
[146.,	147.,	151.],	
..., [152.,	149.,	156.],	
[152.,	149.,	156.],	
[153.,	150.,	157.]],	
[[145.,	146.,	150.],	
[145.,	146.,	150.],	
[145.,	146.,	150.],	
..., [150.,	147.,	154.],	
[152.,	149.,	156.],	
[154.,	151.,	158.]],	
...,			
[[122.,	118.,	119.],	
[131.,	127.,	128.],	
[103., ...,	99.,	100.],	
[141.,	136.,	140.],	
[120.,	115.,	119.],	
[139.,	134.,	138.]],	
[[134.,	130.,	131.],	
[82.,	78.,	79.],	

\108., ..., \129.,	104., 124.,	105.\, 128.\,	
\130.,	125.,	129.\,	
\133.,	128.,	132.\\,	
\\127.,	123.,	124.\,	
\121.,	117.,	118.\,	
\ 87., ..., \119.,	83., 114.,	84.\, 118.\,	
\127.,	122.,	126.\,	
\135.,	130.,	134.\\\\,	dtype=float32)

x

array(\\\\148.,	149.,	153.\,	
\148.,	149.,	153.\,	
\14 8., ...,	149.,	153.\,	
\153.,	150.,	157.\,	
\153.,	150.,	157.\,	
\153.,	150.,	157.\\,	
\\146.,	147.,	151.\,	
\146.,	147.,	151.\,	
\14 6., ...,	147.,	151.\,	
\152.,	149.,	156.\,	
\152.,	149.,	156.\,	
\153.,	150.,	157.\\,	

\\145.,	146.,	150.\,	
\145.,	146.,	150.\,	
\145., ..., \150.,	146., 147.,	150.\, 154.\,	
\152.,	149.,	156.\,	

[154.,	151.,	158.]],	
...,			
[[122.,	118.,	119.],	
[131.,	127.,	128.],	
[103.,	99.,	100.],	
..., [141.,	136.,	140.],	
[120.,	115.,	119.],	
[139.,	134.,	138.]],	
[[134.,	130.,	131.],	
[82.,	78.,	79.],	
[108.,	104.,	105.],	
..., [129.,	124.,	128.],	
[130.,	125.,	129.],	
[133.,	128.,	132.]],	
[[127.,	123.,	124.],	
[121.,	117.,	118.],	
[87., ...,	83.,	84.],	
[119.,	114.,	118.],	
[127.,	122.,	126.],	
[135.,	130.,	134.]]]],	dtype=float32)

BUILD PYTHON CODE

```
from keras.preprocessing.image importImageDataGenerator
```

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_
test_datagen=ImageDataGenerator(rescale=1)
```

```
x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-datase
x_test= test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset\
```

```
    Found 11386 images belonging to 9 classes.
    Found 3416 images belonging to 9 classes.
```

```
from keras.models import
Sequentialfromkeras.layers import
Dense
from keras.layers import
Convolution2Dfrom keras.layers import
MaxPooling2D from keras.layers
importFlatten
```

```
from keras.preprocessing.image importImageDataGenerator
```

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_
test_datagen=ImageDataGenerator(rescale=1)
```

```
x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-datase
x_test= test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset\
```

```
    Found 11386 images belonging to 9 classes.
    Found 3416 images belonging to 9 classes.
```

```
from keras.models import
Sequentialfromkeras.layers import
Dense
from keras.layers import
Convolution2Dfrom keras.layers import
```

```
MaxPooling2D from keras.layers
importFlatten
```

```
from keras.preprocessing.image importImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_
test_datagen=ImageDataGenerator(rescale=1)
```

```
x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-datase
x_test= test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\Veg-dataset\Veg-dataset\
```

```
Found 11386 images belonging to 9 classes.
Found 3416 images belonging to 9 classes.
```

```
model=Sequential()
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(units=300,kernel_initializer='uniform',activation='relu'))
```

```
model.add(Dense(units=150,kernel_initializer='uniform',activation='relu'))
model.add(Dense(units=75,kernel_initializer='uniform',activation='relu'))
model.add(Dense(units=9,kernel_initializer='uniform',activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer="adam",metrics=["accuracy"])
model.fit(x_train,steps_per_epoch=89,epochs=20,validation_data=x_test,validation_steps=27)
```

Epoch 89/89	1/20 \========================= =====\	- 72s	788ms/step	- loss:	1.9629	- accuracy:	0

Epoch 89/89	2/20 \========================= =====\	- 70s	789ms/step	- loss:	1.4169	- accuracy:	0
Epoch	3/20						
89/89 Epoch	\========================= =====\4/20	- 69s	776ms/step	- loss:	1.1186	- accuracy:	0
89/89 Epoch 89/89	\========================= =====\5/20 \========================= =====\	- 69s - 69s	772ms/step 771ms/step	1. 2.	0.8775 0.7720	1. accuracy 2. accuracy	0 0
Epoch 89/89	6/20 \========================= =====\	- 69s	769ms/step	- loss:	0.6852	- accuracy:	0
Epoch	7/20						
89/89 Epoch 89/89	\========================= =====\8/20 \========================= =====\	- 68s - 70s	767ms/step 780ms/step	1. 2.	0.6419 0.6209	1. accuracy 2. accuracy	0 0
Epoch 89/89	9/20 \========================= =====\	- 70s	781ms/step	- loss:	0.5679	- accuracy:	0
Epoch	10/20						
89/89 Epoch	\========================= =====\11/20	- 69s	773ms/step	- loss:	0.4980	- accuracy:	0
89/89 Epoch 89/89	\========================= =====\12/20 \=========================	- 70s	779ms/step 772ms/step	1.	0.5424 0.4508	1. accuracy 2. accu	0 0

	=====\	- 69s		2.		racy	
Epoch 89/89	13/20 \========================= =====\	- 69s	775ms/step	- loss:	0.4316	- accuracy:	0
Epoch	14/20						
89/89 Epoch 89/89	\========================= =====\15/20 \========================= =====\	- 69s - 72s	772ms/step 805ms/step	1. 2.	0.4464 0.3720	1. accu racy 2. accu racy	0 0
Epoch 89/89	16/20 \========================= =====\	- 69s	775ms/step	- loss:	0.3279	- accuracy:	0
Epoch 89/89	17/20 \========================= =====\	- 69s	772ms/step	- loss:	0.3478	- accuracy:	0
Epoch	18/20						
89/89 Epoch 89/89	\========================= =====\19/20 \========================= =====\	- 70s - 69s	781ms/step 775ms/step	1. 2.	0.3886 0.3481	1. accu racy 2. accu racy	0 0
Epoch 89/89	20/20 \========================= =====\	- 69s	779ms/step	- loss:	0.3707	- accuracy:	0

<keras.callbacks.History at 0x1b0daf122e0>

```
x_train = train_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-dax_test=
test_datagen.flow_from_directory(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-data
```

```
    Found 5384 images belonging to 6 classes.

    Found 1686 images belonging to 6 classes.
```

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

img=image.load_img(r"C:\Users\DELL\Desktop\fruit-dataset\fruit-dataset\test\Apple
healthimg
```

```
import numpy as nps
x=image.img_to_arra
y(img)
```

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


pred=(model.predict(x)> 0.5).astype("int32")


pred


import requests
from tensorflow.keras.preprocessing import image
model.save(r"C:\Users\DELL\Desktop\fruit-dataset.h5
")model.save('fruit-dataset.h5')
model=load_model('fruit-dataset.h5')


from tensorflow.keras.models import load_model
importnumpy as np
import pandas as pd
import tensorflow as tf
from flask import Flask, request , render_template, redirect, url_forimport os
from werkzeug.utils import secure_filename
from tensorflow.python.keras.backend importset_session


app= Flask(__name__)
model =
load_model("fruit-dataset.h5")
@app.route('/')

def home():
    return render_template('home.html')

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

```
@app.route('/predict',methods=['POST']
)def predict():
    if request.method=='POST':
        f= request.files['images']
        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['pl
        ant']print(plant)
        if(plant=="fruit"):
            preds=model.predict_classess
            (x)print(preds)
            df=pd.read_excel('precautions-veg.xlsx')
            print(df.iloc[preds[0]]['cautions'])
        else:
            pred=model1.predict_classes(x)
            df=pd.read_excel('precautions-fruits.xlsx')
            print(df.iloc[preds[0]]['caution'])
            return df.iloc[preds[0]]['caution']



if__name=="main_":
    app.run(debug=False)
```