FERTILIZER RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

1 INTRODUCTION

1.1 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.

Detection and recognition of plant diseases using machine learning are very efficient in providing symptoms of identifying diseases at its earliest. Plant pathologists can analyze the digital images using digital image processing for diagnosis of plant diseases. Application of computer vision and image processing strategies simply assist farmers in all of the regions of agriculture. Generally, the plant diseases are caused by the abnormal physiological functionalities of plants. Therefore, the characteristic symptoms are generated based on the differentiation between normal physiological functionalities and abnormal physiological functionalities of the plants. Mostly, the plant leaf diseases are caused by Pathogens which are positioned on the stems of the plants. These different symptoms and diseases of leaves are predicted by different methods in image processing. These different methods include different fundamental processes like segmentation, feature extraction and classification and so on. Mostly, the prediction and diagnosis of leaf diseases are depending on the segmentation such as segmenting the healthy tissues from diseased tissues of leaves.

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

1.2 PURPOSE

The proposed method uses SVM to classify tree leaves, identify the disease and suggest the fertilizer. The proposed method is compared with the existing CNN based leaf disease prediction. The proposed SVM technique gives a better result when compared to existing CNN.

2 LITERATURE SURVEY

2.1 EXISTING PROBLEM

2.1 Title: Healthy Harvest: Crop Prediction And Disease Detection System

Author: Sambhav Bhansali; Punit Shah; Jinay Shah;

Year: 2022

Description:

Economy of India highly depends on agriculture. Still traditional ways of recommendations are used for agriculture. Currently, farmers use traditional ways of approximations for amount of fertilizer used and the type of crop to be sown. Agriculture extremely depends on the type of soil and climatic condition of the region. Therefore, it becomes vital to create advancement in this field. With the help of Machine Learning and Deep Learning Techniques we will create a Web-App which will be one-stop solutions for information regarding the agriculture. Crop and fertilizer recommendation system will help the farmers in increasing their yield production. We are going to take the soil parameters along with the weathers API to figure out the most suitable crop for that region. Using the decision tree and navies bayes algorithm we will make the recommendation model which will use the N-K-P, Ph. value and rainfall as the parameters for training. Basis on the crop and region of farming we will recommend the fertilizer and its uses to boost the yield productivity for farmers. Sometimes due to unwanted excess of rainfall or the pest attack can cause disease to crops. We will use the image classification technique where the user can upload the picture of the affected plant/crop and the system will figure out the type of disease which will be done using Support Vector Machine (SVM) or using the neural network techniques.

2.2 Title: Neural Network Based Fertilizers Recommendation System For Disorder Classification And Prediction In Petal Images

Author: N. Valarmathi; M. Vengateshwaran; Kalaimani Shanmugam;

Year: 2021

Description:

The point of farming isn't just to take care of the ever-developing populace but at the same time is a basic wellspring of vitality and an answer for the emergency of an Earth-wide temperature boost. Determination of plant ailment is basic for early finding and control of it. The unaided eye method is generally utilized for the conclusion of ailments. This methodology requires experts who can recognize varieties in leaf shading. Ordinarily a similar malady is characterized by a few specialists as a different sickness. This arrangement is exorbitant, in light of the fact that it requires nonstop expert management. Makers need to follow their yields and perceive the primary signs at modest costs so as to abstain from spreading even a plant malady and spare a lot of income. Recruiting qualified ranchers can't be reasonable especially in far off geologically detached zones. AI calculations in an image can give a substitute strategy to following plants and an expert can deal with such a way to deal with offer their types of assistance at a lower cost. It incorporates picture division which incorporates the dynamic shape strategy and the picture arrangement approach which incorporates a neural system calculation for foreseeing various kinds of ailments. Or on the other hand grow the way to deal with suggest the composts dependent on the examination of power with estimations.

2.3 Title: KRISHI RAKSHAN - A Machine Learning based New Recommendation System to the Farmer

Author: D. N. V. S. L. S. Indira; M. Sobhana; A. H. L. Swaroop;

Year: 2022

Description:

Totally 54% of India's land area is deemed arable, making it the world's largest agrarian economy. Soil infertility owing to over fertilization, as well as a lack of access and awareness of contemporary agricultural practices, are the different factors that contribute to low agricultural production. The main purpose of this research work is to develop a machine learning-based recommendation system to increase agricultural productivity. A variety of datasets were used in this study to design and develop advanced models to estimate the crop, recommend fertiliser, and identify plant disease. An algorithm called MobileNet uses an image of a leaf to identify the disease present in a plant. The XGBoost model predicts a suitable crop based on the local soil nutrients and rainfall. Random Forest [RF] model was used to propose fertilizer and develop ideas for improving soil fertility depending on nutrients present in the soil. When compared to other approaches, the proposed model delivers a high level of accuracy. Moreover, this article suggests the farmer to increase the crop yield by entering the input values and local soil conditions, wherein the model suggests recommended crop for that soil with an accuracy of 99%.

2.4 Title: Predictor Analysis and Proliferation of Fertility and Production for Agriculturalists

Author: C. Shyamala Kumari; Rohit Kumar; Saurav Kumar Gupta; Shourjya Hazra;

Year: 2022

Description:

The world of technical innovation and experiments have brought a new technological movement all over the world. Despite of it a major portion of the agribusiness community is far away from technical aspects that can make farming easy and efficient. About 60% of total agriculturalists in India are poor and can't afford heavy robotics to take advantage of the technology. Farmers are sometimes oblivious of the disease in the crop and the market prices of the products. This is why they are paid less than what the actual cost is. As a solution, a multilingual platform has been proposed which can be accessed by all people and from where the farmers can easily get to know the current price of their crops in the market. The system is fed with reliable data from the government and is built on the anaconda platform under the TensorFlow environment. The system helps in the prognosis of crop diseases and also furnishes the reason and cure for the disease. It is also helpful to get recommendations about the correct fertilizer as per the quality of soil and other considerations. The system will be very much helpful for poor farmers who can't afford pricey tools to enhance their crop production. Also, it will keep them aware of the current prices of the crop they are reaping and suggest which crop is suitable for which weather condition, which ultimately will be a boon for them.

2.5 Title: Agro-Mate: A Virtual Assister to Maximize Crop Yield in Agriculture Sector

Author: Dayalini S; Sathana M; Navodya P.R. N; R.W.A.I.M.N Weerakkodi;

Year: 2022

Description:

This paper presents a decision support system that supports farmers to take accurate decisions and help them with soil quality determination, best crop selection, rice disease prediction, and disaster prediction for the wet zone of Sri Lanka. This project has incorporated technologies such as Deep Learning, Image Processing, the Internet of Things, and Machine Learning that can aid farmers or investors to maximize yield. 'Agro-Mate' consists of four components which are soil quality determination, best crop selection, rice disease prediction, and natural disaster prediction. Also, the application suggests fertilizer when soil is lacking quality and provides recommendations whenever rice diseases or natural disasters are identified. An android mobile application is developed which users will utilize to access the system and make use of it. The proposed system facilitates the farmer in accurate decision-making to gain more quality and quantity of crops. 'Agro-mate' is more likely to increase the productivity of crops. In the future, this paper will be included with the test and evaluations results to prove the proposed decision-making concept.

2.2 REFERENCES

1. S. Bhansali, P. Shah, J. Shah, P. Vyas and P. Thakre, "Healthy Harvest: Crop Prediction And Disease Detection System," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), 2022, pp. 1-5, doi: 10.1109/I2CT54291.2022.9825446.

2. N. Valarmathi, M. Vengateshwaran, K. Shanmugam and R. Sudha, "Neural Network Based Fertilizers Recommendation System For Disorder Classification And Prediction In Petal Images," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021, pp. 1532-1537, doi: 10.1109/ICOSEC51865.2021.9591715.

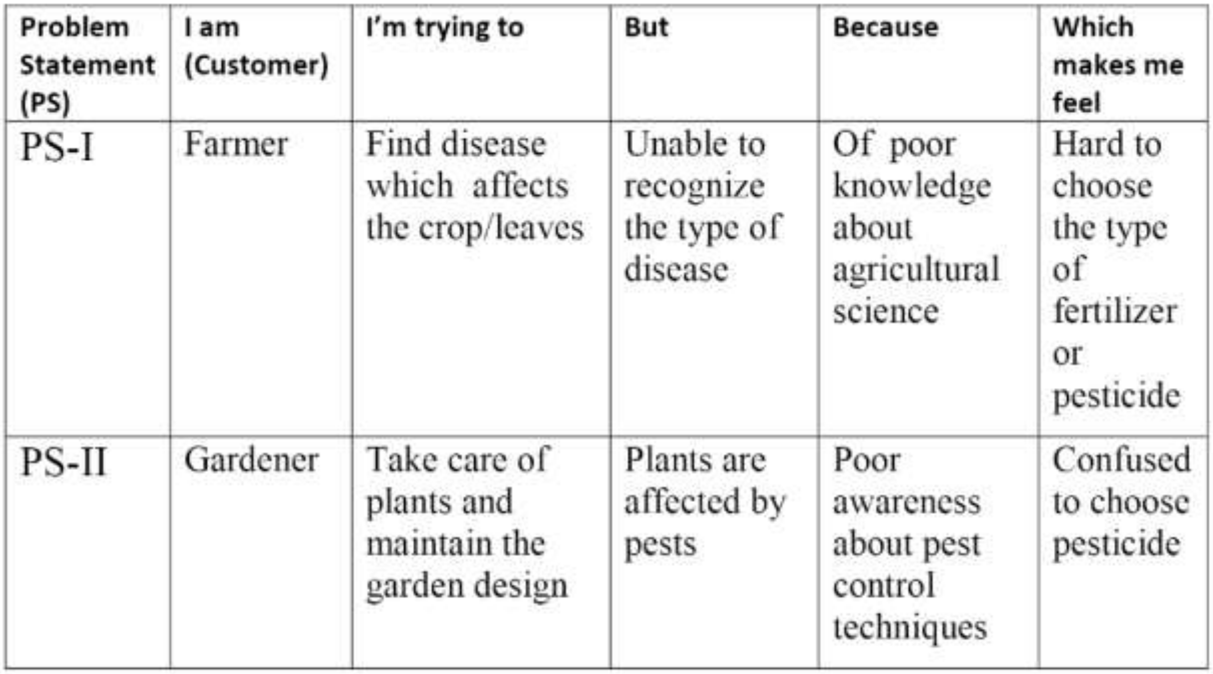
3. D. N. V. S. L. S. Indira, M. Sobhana, A. H. L. Swaroop and V. Phani Kumar, "KRISHI RAKSHAN - A Machine Learning based New Recommendation System to the Farmer," 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), 2022, pp. 1798-1804, doi: 10.1109/ICICCS53718.2022.9788221.

4. C. S. Kumari, R. Kumar, S. K. Gupta, S. Hazra and M. Paul, "Predictor Analysis and Proliferation of Fertility and Production for Agriculturalists," 2022 IEEE India Council International Subsections Conference (INDISCON), 2022, pp. 1-4, doi: 10.1109/INDISCON54605.2022.9862897.

5. D. S, S. M, N. P. R. N, R. W. A. I. M. N. Weerakkodi, A. Jayakody and N. Gamage, "Agro-Mate: A Virtual Assister to Maximize Crop Yield in Agriculture Sector," TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON), 2021, pp. 387-392, doi: 10.1109/TENCON54134.2021.9707199.

2.3 PROBLEM STATEMENT DEFINITION

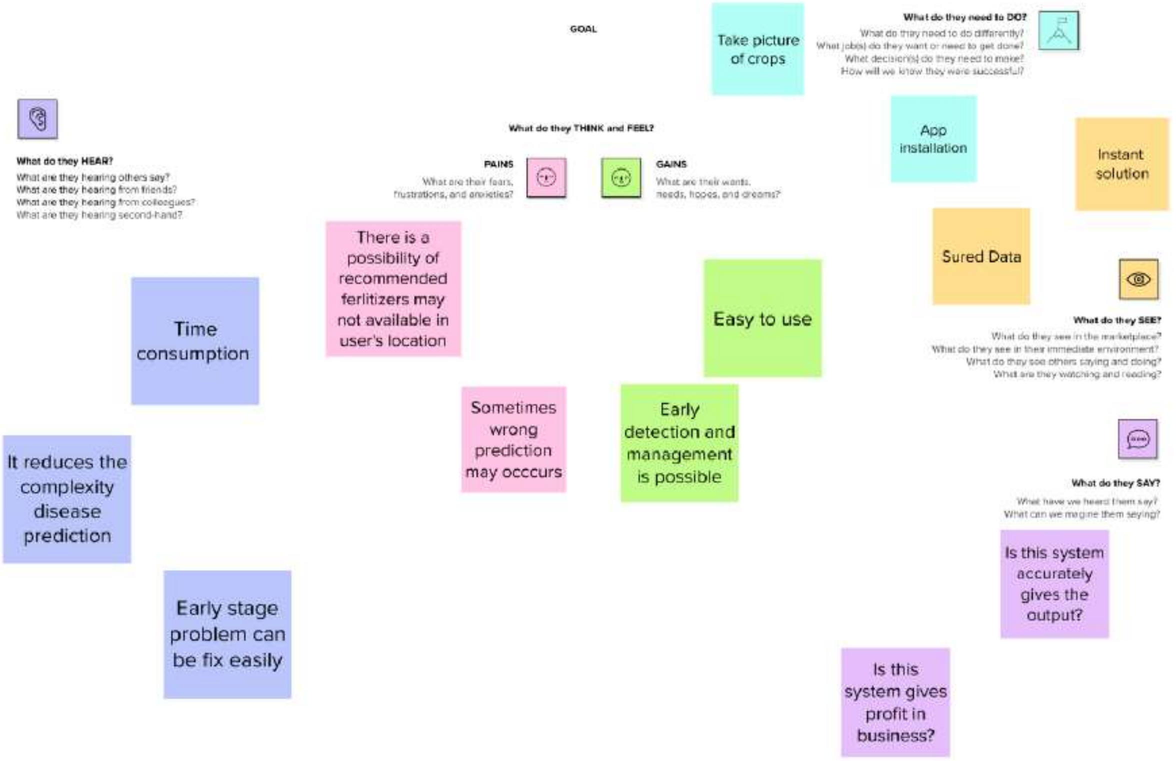
Most of the Indian population depends on Agriculture for their livelihood. Agriculture gives an opportunity of employment to the village people to develop a country like India on large scale and give a push in the economic sector. The majority of farmers face the problem of planting an inappropriate crop for their land based on a conventional or non-scientific approach. Because crops are highly affected by pest. So, the farmer’s profit is reduced in a considerable manner. Here we proposed the system which recognizes the disease which affect the crop and suggest the fertilizer. By using that farmer can increase the productivity and also their profit.



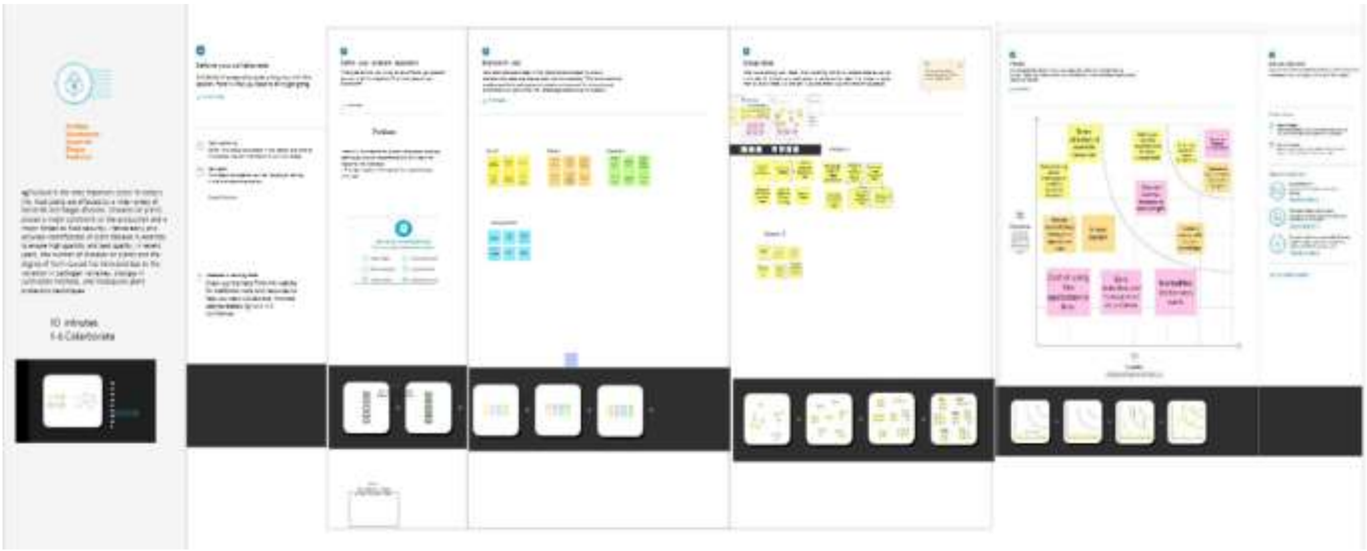
Problem Statement (PS)	I am (Customer)	I’m trying to	But	Because	Which makes me feel
PS-I	Farmer	Find disease which affects the crop/leaves	Unable to recognize the type of disease	Of poor knowledge about agricultural science	Hard to choose the type of fertilizer or pesticide
PS-II	Gardener	Take care of plants and maintain the garden design	Plants are affected by pests	Poor awareness about pest control techniques	Confused to choose pesticide

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:


GOAL
Take picture of crops
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 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?
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
App installation
Instant solution
Sured Data
There is a possibility of recommended ferlitizers may not available in user's location
Time consumption
Easy to use
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?
Sometimes wrong prediction may occcurs
Early detection and management is possible
It reduces the complexity disease prediction
What do they SAY?
What have we heard them say?
What can we imagine them saying?
Is this system accurately gives the output?
Early stage problem can be fix easily
Is this system gives profit in business?

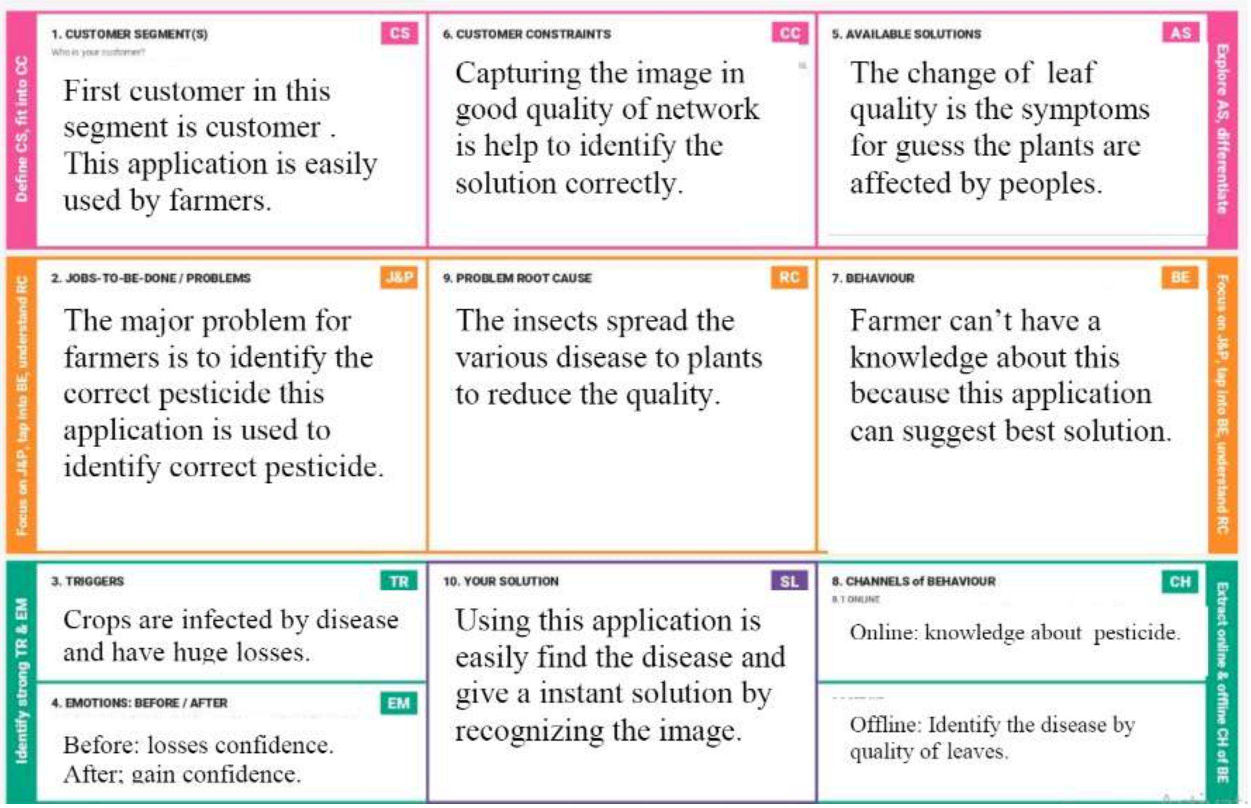
3.2 Ideation & Brainstorming:



3.3 Proposed Solution:

S.NO	PARAMETER	DESCRIPTION
1	Problem statement (problem to be solved)	Disease affects the quality and quantity of crops/plants. Difficult to identify the disease.
2	Idea/solution description	In early stage we have to find the disease and suggest the correct pesticide.
3	Novelty / uniqueness	It recognizes the image and suggest good fertilizer to curette disease.
4	Social impact/customer satisfaction	It helps to identify the disease and apply the correct pesticide and improve the plant quality.
5	Business model(revenue model)	In basic stage the application is recommends solutions to farmer.
6	Scalability of the solution	It introduces the online shopping for fertilizer.

3.4 Problem Solution fit:



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

First customer in this segment is customer . This application is easily used by farmers.

6. CUSTOMER CONSTRAINTS CC

Capturing the image in good quality of network is help to identify the solution correctly.

5. AVAILABLE SOLUTIONS AS

The change of leaf quality is the symptoms for guess the plants are affected by peoples.

Explore AS, differentiate

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

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

The major problem for farmers is to identify the correct pesticide this application is used to identify correct pesticide.

9. PROBLEM ROOT CAUSE RC

The insects spread the various disease to plants to reduce the quality.

7. BEHAVIOUR BE

Farmer can't have a knowledge about this because this application can suggest best solution.

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

Identify strong TR & EM

3. TRIGGERS TR

Crops are infected by disease and have huge losses.

4. EMOTIONS: BEFORE / AFTER EM

Before: losses confidence.
After; gain confidence.

10. YOUR SOLUTION SL

Using this application is easily find the disease and give a instant solution by recognizing the image.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

Online: knowledge about pesticide.

Offline: Identify the disease by quality of leaves.

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS:

4.1 Functional requirement:

Following are the functional requirements of the proposed solution

FR.NO	FUNCTIONAL REQUIREMENT	SUB REQUIREMENT (STORY/SUBTASK)
Fr-1	User registration process	Registration through the Gmail
Fr-2	User confirmation process	Confirmation via OTP/Email
Fr-3	Capturing image of leaf	Image of the leaf will be captured.
Fr-4	Image processing to prediction	To predict the disease of the leaf by uploading a image.
Fr-5	Leaf identification for suggestion	Identify and predict the disease.
Fr-6	Image description	Analyze the leaf and suggest suitable fertilizer/pesticide for the disease.

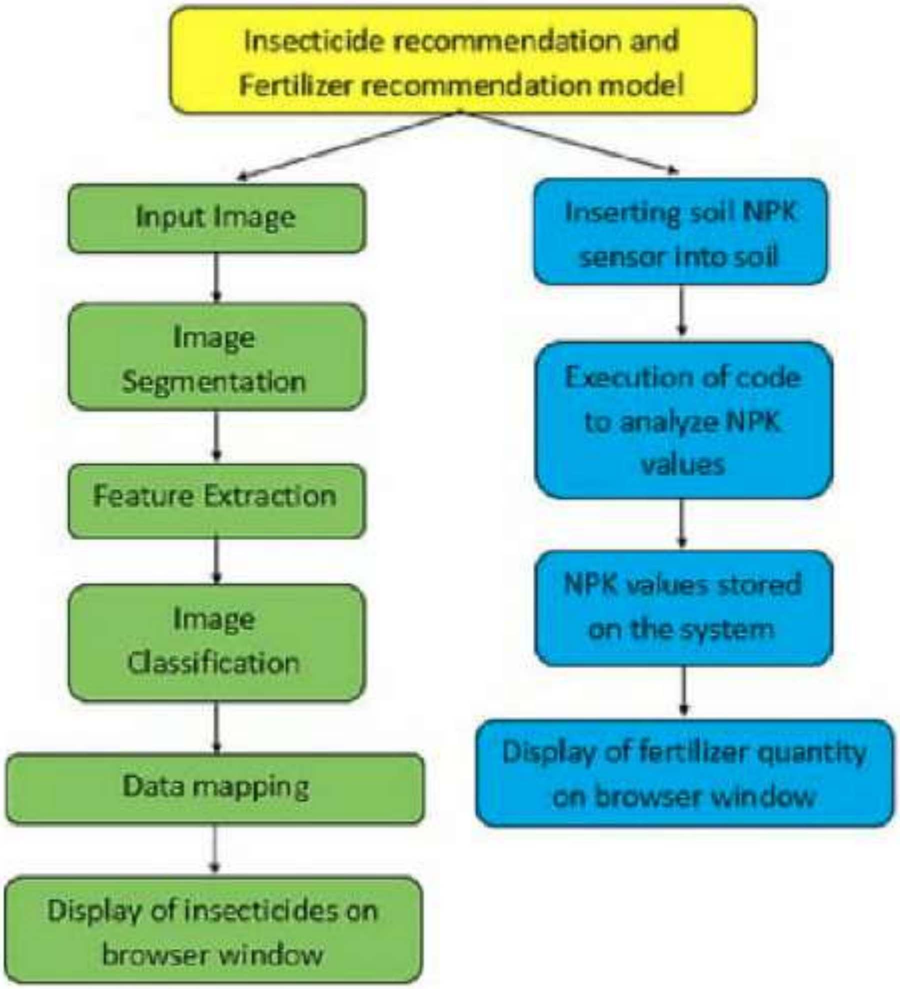
Non-functional requirement:

Following are the non-functional requirement of the proposed solution

FR.NO	**NON FUNCTIONAL REQUIREMENT**	**SUB REQUIREMENT (STORY/SUBTASK)**
Ntr-1	Usability	To detect the disease which present in the leaf by datasets.ser registration process
Nfr-2	Security	The information are encrypted and secured.
Nfr-3	Reliability	To predict the disease most important is leaf quality.
Nfr-4	Performance	Based on the quality of leaf.
Nfr-5	Availability	It is available for all to predict and get suggestion of disease.
Nfr-6	Scalability	Prediction of disease in leaves will be increased

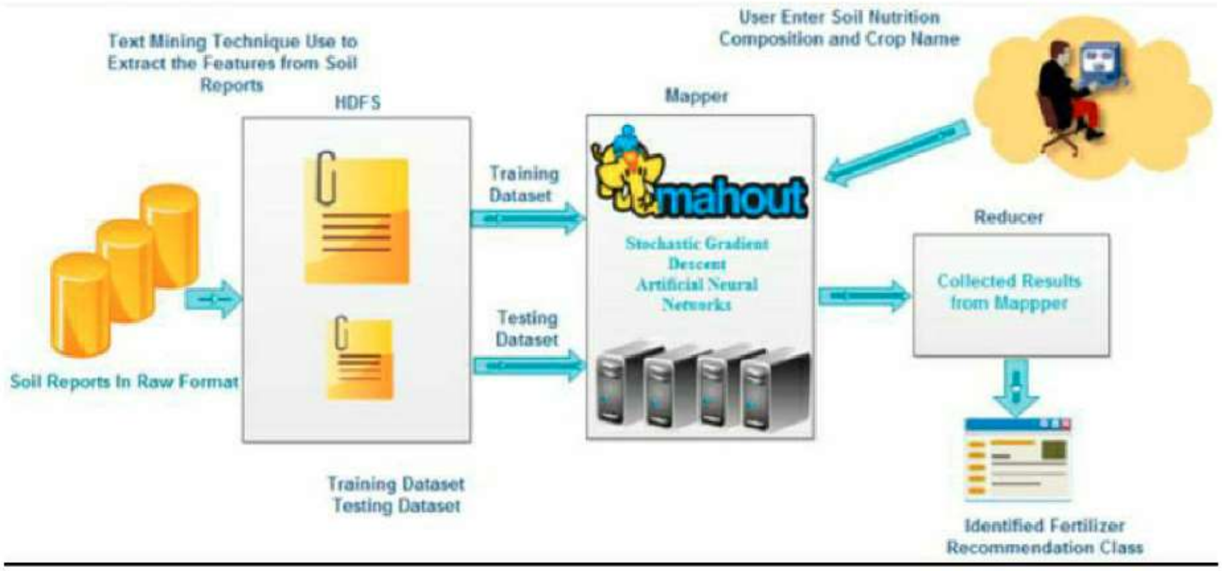
5. PROJECT DESIGN:

5.1 Data Flow Diagrams:


Insecticide recommendation and Fertilizer recommendation model
Input Image
Image Segmentation
Feature Extraction
Image Classification
Data mapping
Display of insecticides on browser window
Inserting soil NPK sensor into soil
Execution of code to analyze NPK values
NPK values stored on the system
Display of fertilizer quantity on browser window

5.2 Solution & Technical Architecture:

Solution Architecture:


Text Mining Technique Use to Extract the Features from Soil Reports
HDFS
Soil Reports In Raw Format
Training Dataset
Testing Dataset
Training Dataset
Testing Dataset
Mapper
mahout
Stochastic Gradient Descent
Artificial Neural Networks
User Enter Soil Nutrition Composition and Crop Name
Reducer
Collected Results from Mappper
Identified Fertilizer Recommendation Class

5.3 User Stories:

User	Type Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Application	USN-1	As a user I installed the application for day to day leaf disease detection and recognize it.	My app will be installed in home screen.	High	Sprint-1
		USN-2	As a user, I can register for the application identifying diseases at early stage	I can access my account /dashboard	High	Sprint-1
		USN-3	As a user I will detect the plant	I can register and access the	Medium	Sprint-2

			once I have registered for the particular process	dashboard with application account		

6. PROJECT PLANNING & SCHEDULING:

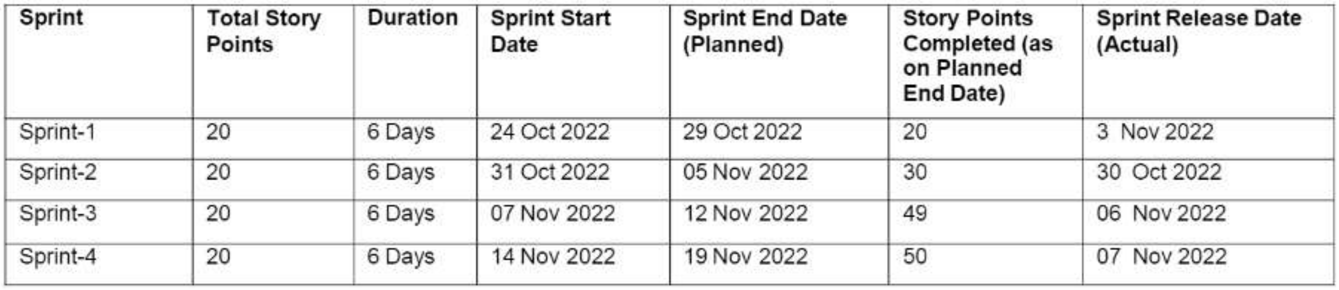
6.1 Sprint Planning & Estimation:

Use the below template to create product backlog and sprint schedule

Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	
Sprint -1	Registration	USN 1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	
Sprint -1		USN 2	As a user, I will receive confirmation email once I have registered for the application	1	High	

Sprint -2		USN 3	As a user, I can register for the application through Face book	2	Low	
Sprint -1		USN 4	As a user, I can register for the application through Gmail	2	Medium	
Sprint-1	Login	USN 5	As a user, I can log into the application by Entering email & password	1	High	

6.2 Sprint Delivery Schedule:

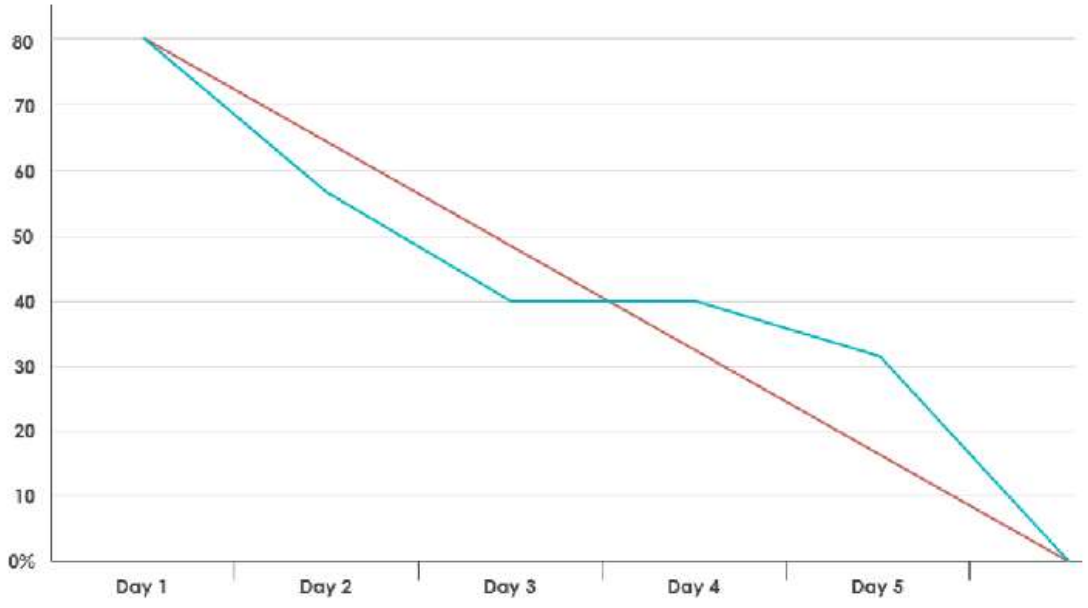


Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	3 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	30	30 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	49	06 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	50	07 Nov 2022

6.3 Reports from JIRA:

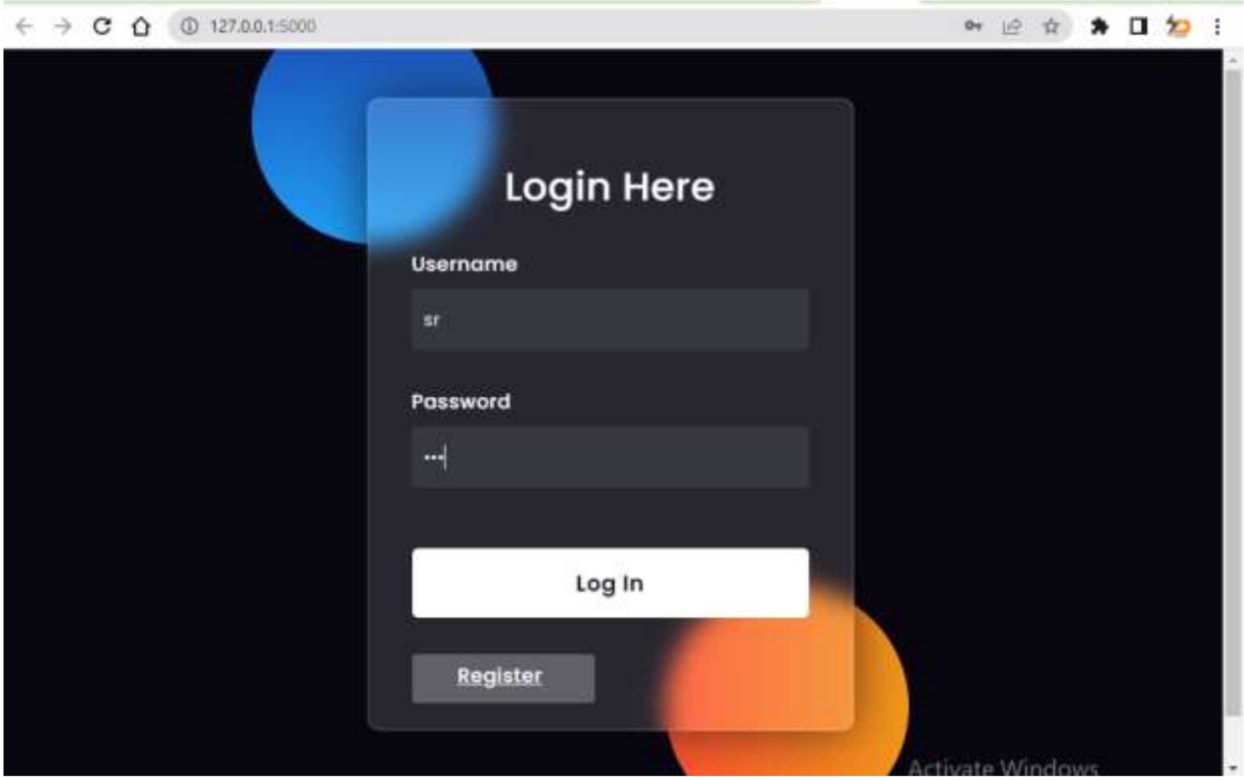
$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burndown Chart:

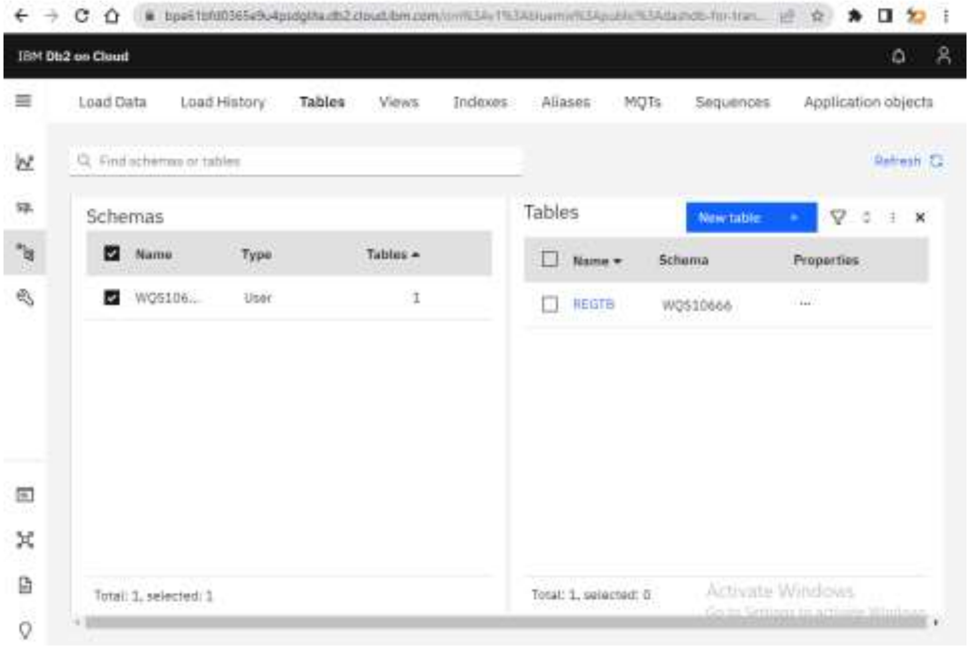


7. CODING & SOLUTIONING:

LOGIN HERE:



REGISTER PAGE



Schemas

☑	Name	Type	Tables ▲
☑	WQS106...	User	1

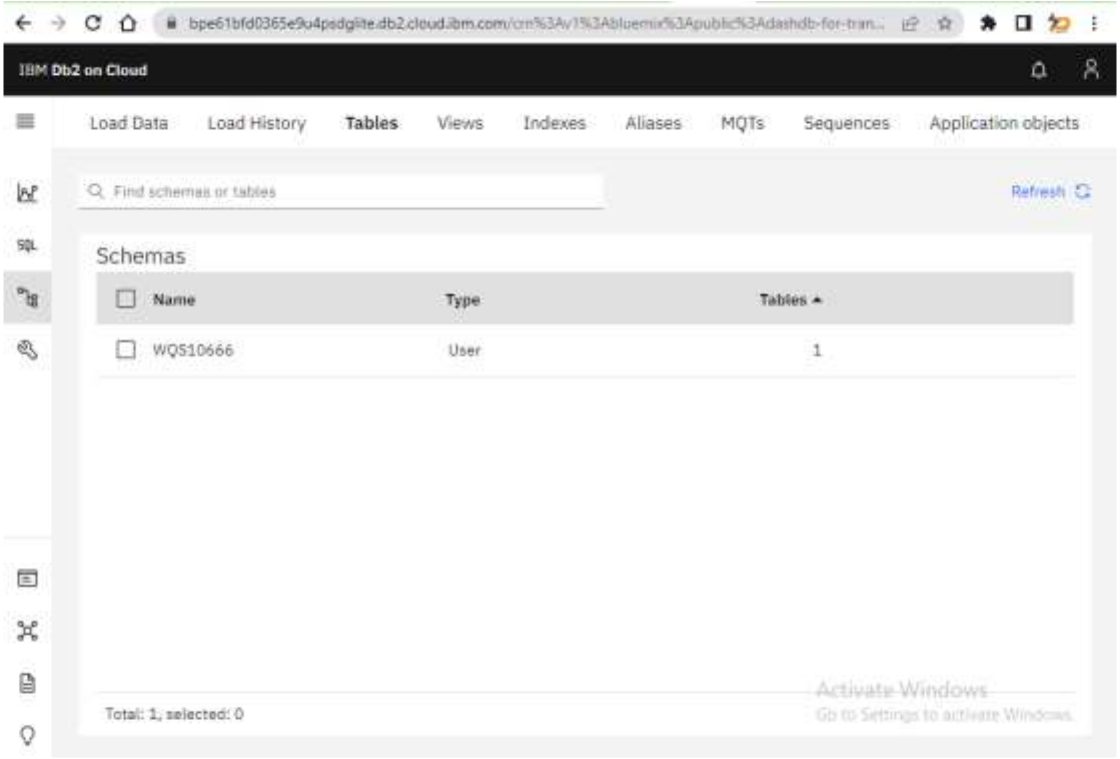
Total: 1, selected: 1

Tables

□	Name ▾	Schema	Properties
□	REGTB	WQS10666	...

Total: 1, selected: 0

TABLE PAGE

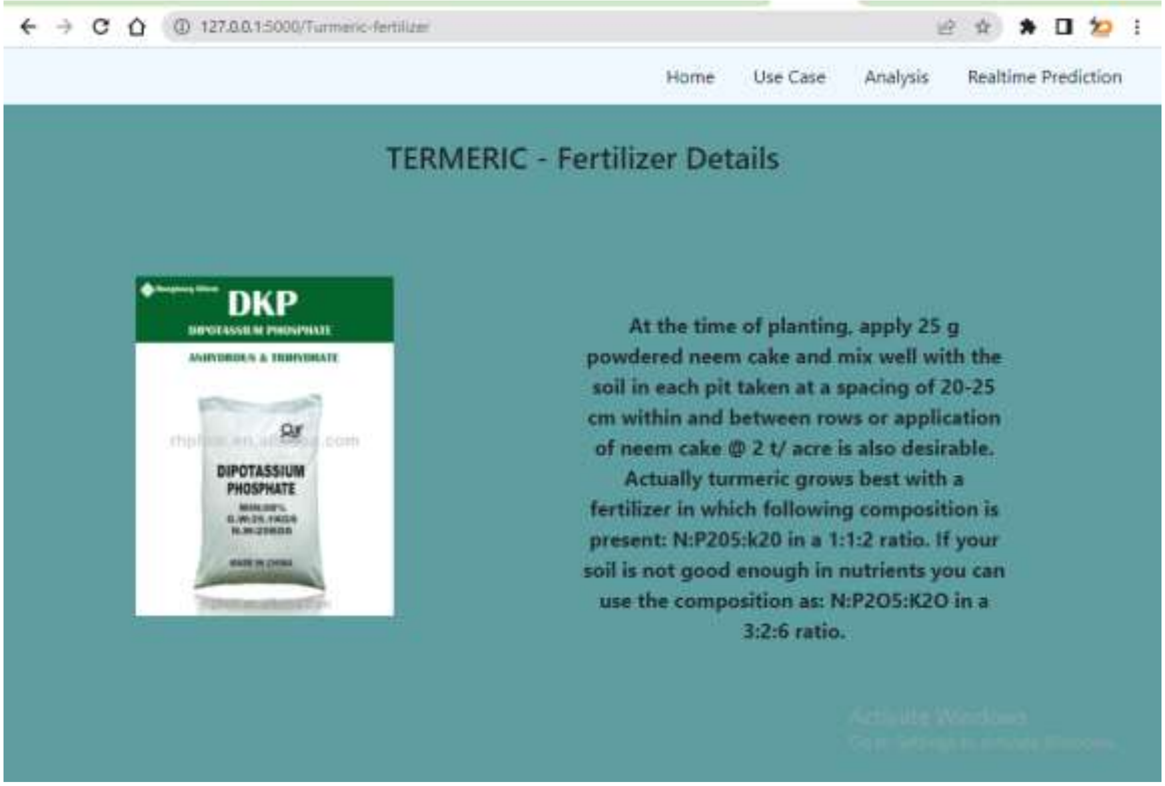


Schemas

□	Name	Type	Tables ▲
□	WQS10666	User	1

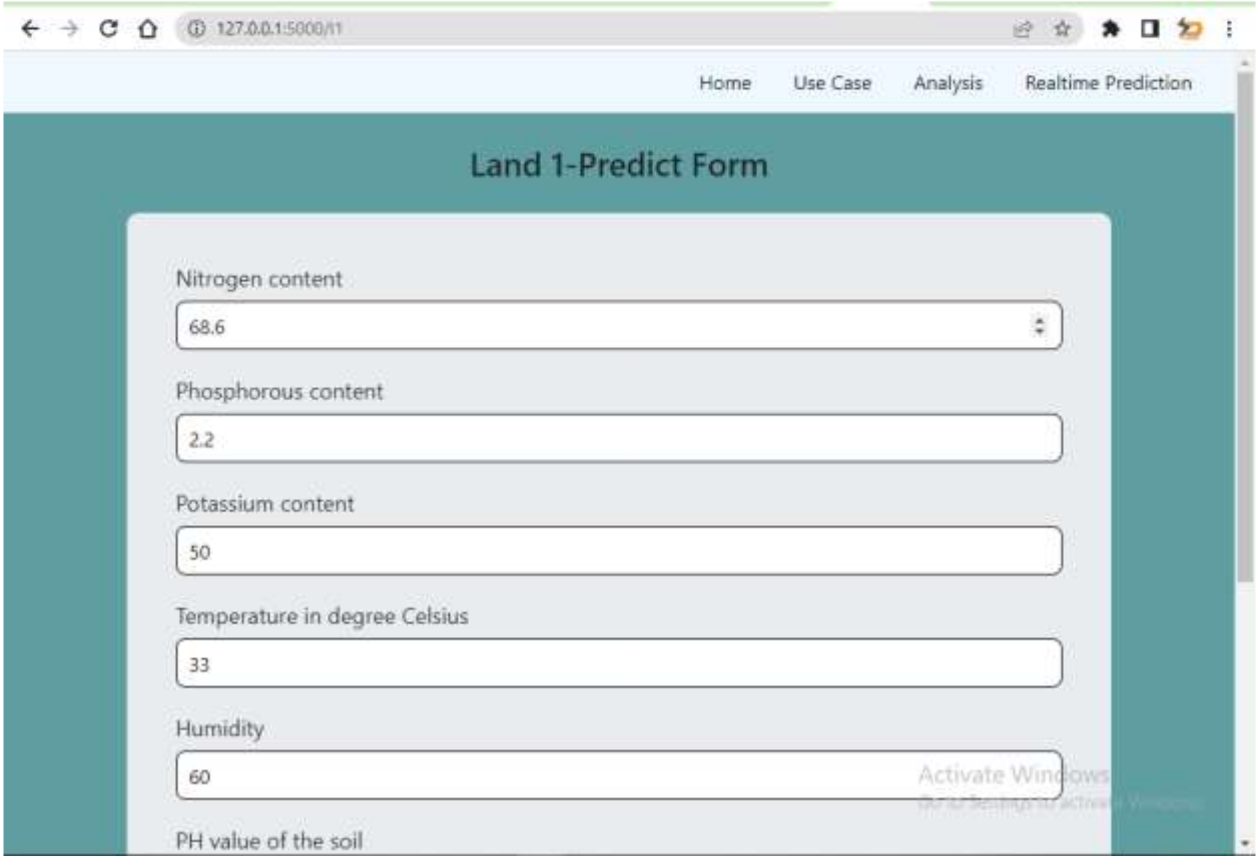
Total: 1, selected: 0

FERTIKIZER DETAILS



TERMERIC - Fertilizer Details

At the time of planting, apply 25 g powdered neem cake and mix well with the soil in each pit taken at a spacing of 20-25 cm within and between rows or application of neem cake @ 2 t/ acre is also desirable. Actually turmeric grows best with a fertilizer in which following composition is present: N:P205:k20 in a 1:1:2 ratio. If your soil is not good enough in nutrients you can use the composition as: $N:P_2O_5:K_2O$ in a 3:2:6 ratio.



Land 1-Predict Form

Nitrogen content

68.6

Phosphorous content

2.2

Potassium content

50

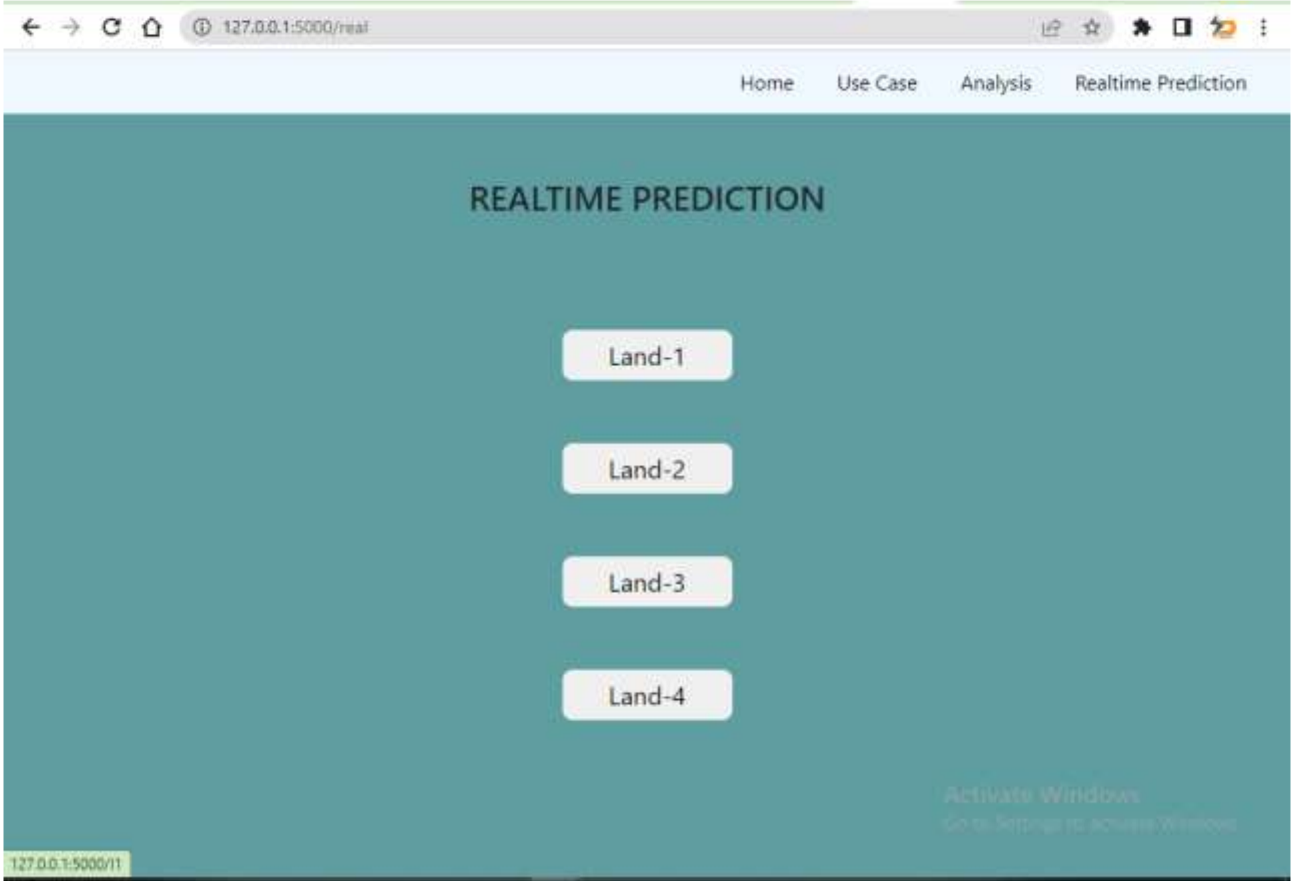
Temperature in degree Celsius

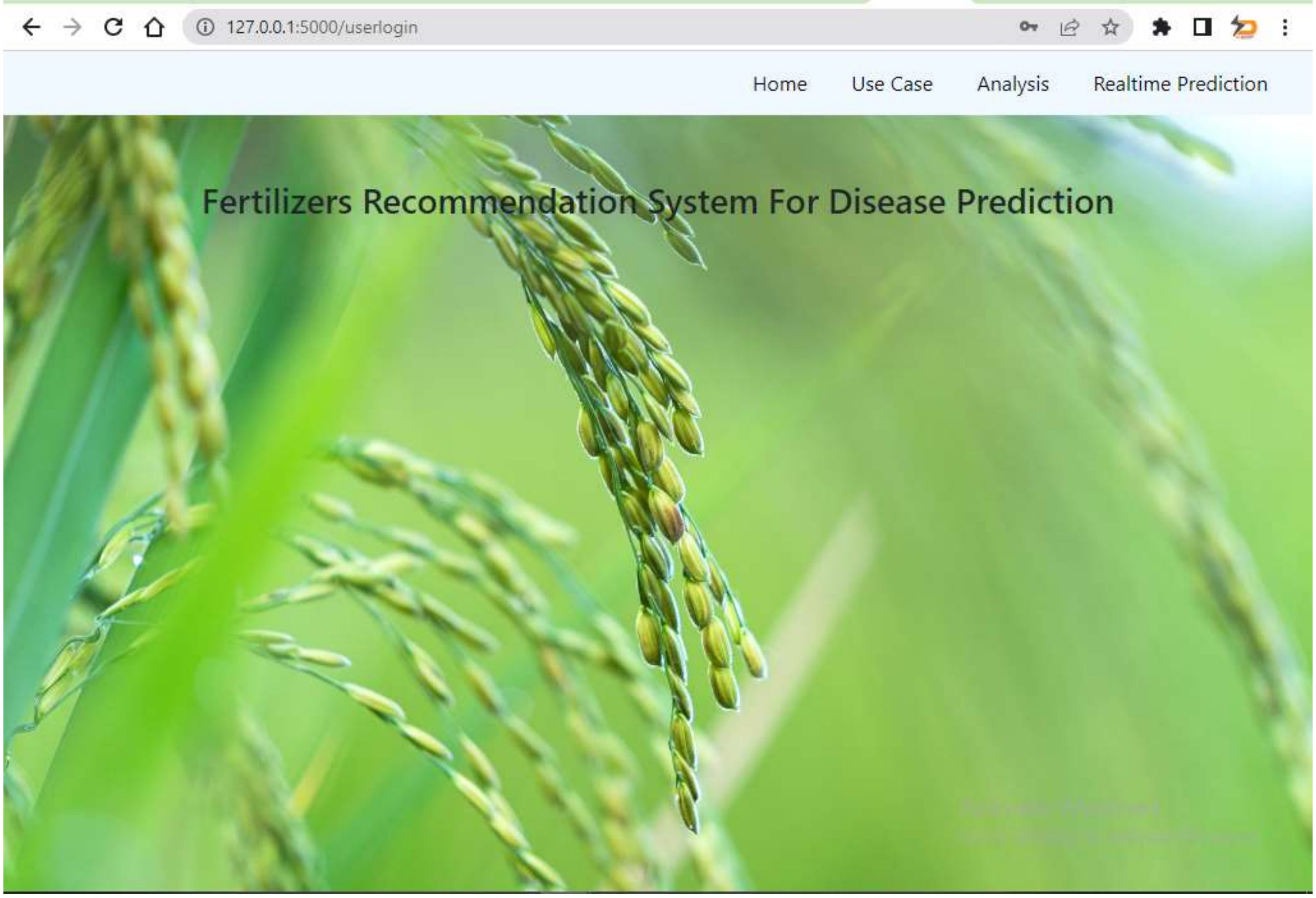
33

Humidity

60

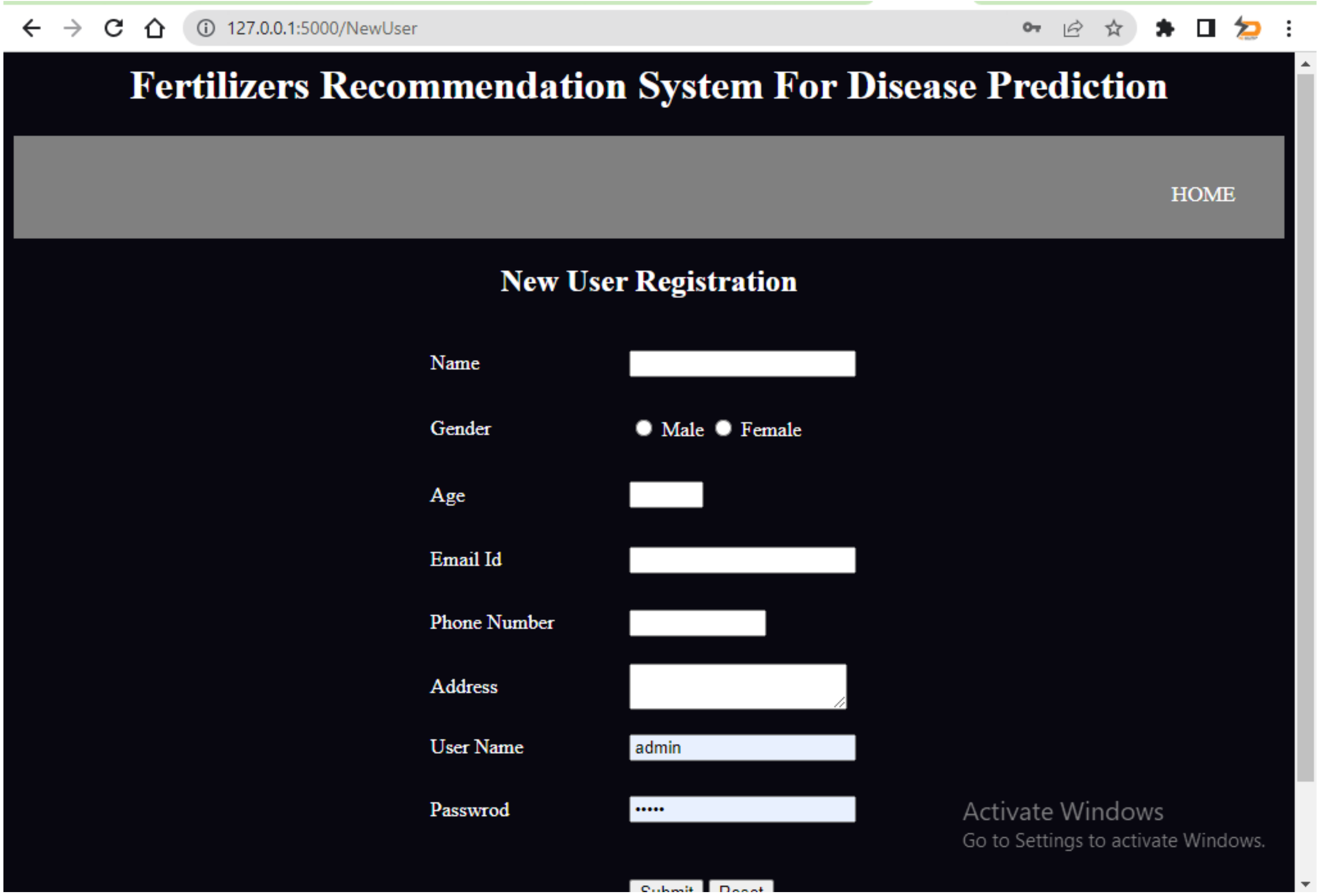
PH value of the soil


127.0.0.1:5000/real
Home
Use Case
Analysis
Realtime Prediction
REALTIME PREDICTION
Land-1
Land-2
Land-3
Land-4
127.0.0.1:5000/l1



127.0.0.1:5000/userlogin

Fertilizers Recommendation System For Disease Prediction



Fertilizers Recommendation System For Disease Prediction

New User Registration

Name

Gender ○ Male ○ Female

Age

Email Id

Phone Number

Address

User Name admin

Passwrod •••••

8 TESTING

8. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.1 TYPES OF TESTS

8.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test.

System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

8.2 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

8.2.1 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

8.2.2 Test objectives

- All field entries must work properly.

- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

8.2.3 Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

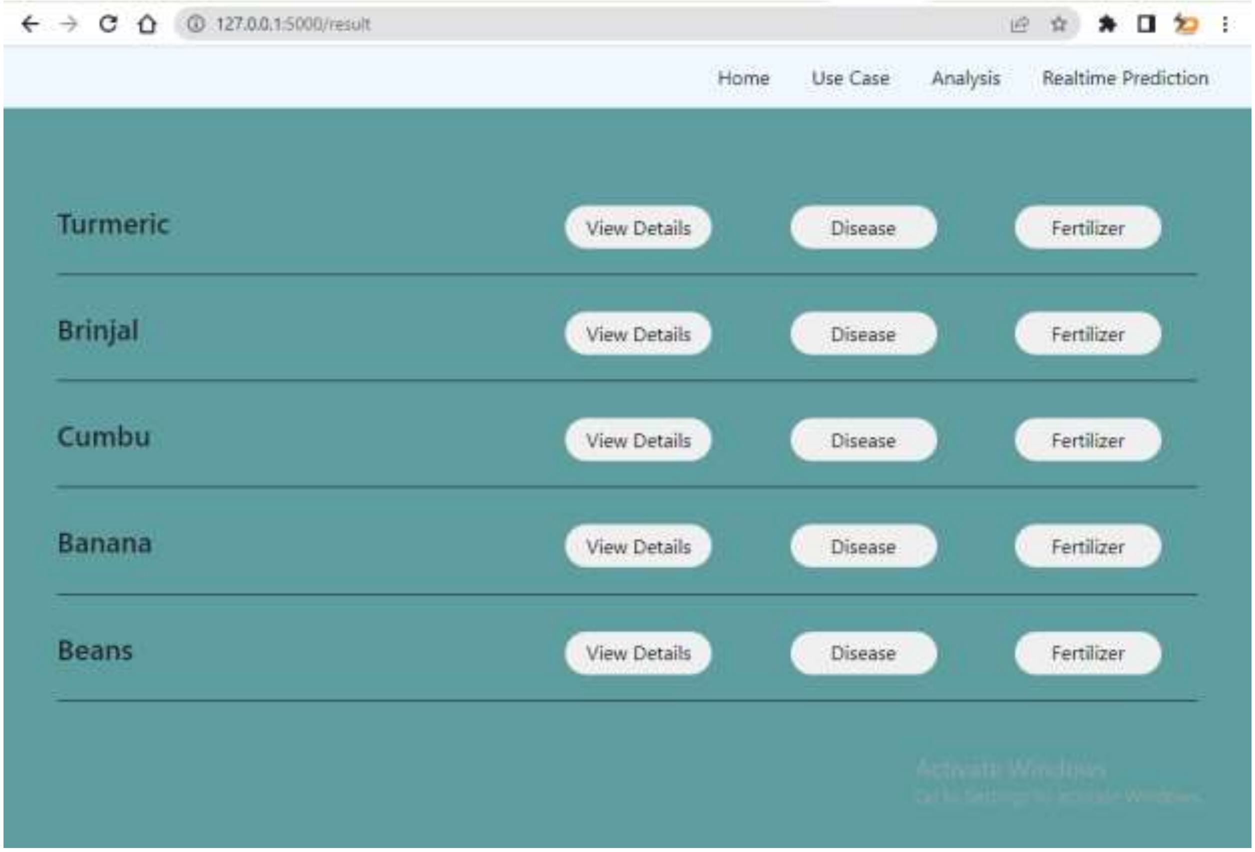
8.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9 RESULTS:

9.1 Performance Metrics:



10 ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

- High accuracy
- Increase overall performance

DISADVANTAGES:

- Less accuracy
- Low performance

11. CONCLUSION

In this paper, significance of management of crops was studied vastly. Farmers need assistance with recent technology to grow their crops. Proper prediction of crops can be informed to agriculturists in time basis. Many Machine Learning techniques have been used to analyze the agriculture parameters. Some of the techniques in different aspects of agriculture are studied by a literature study. Blooming Neural networks, Soft computing techniques plays significant part in providing recommendations. Considering the parameter like production and season, more personalized and relevant recommendations can be given to farmers which makes them to yield good volume of production.

11. FUTURE WORK

Fertilizers are applied to replace the essential nutrients for plant growth to the soil after they have been depleted. Excess amounts of fertilizers may enter streams creating sources of nonpoint pollution. Fertilizers most commonly enter water sources by surface runoff and leaching from agricultural lands.

13. APPENDIX

SOURCE CODE:

```
from flask import Flask, flash,render_template,request,session

import numpy as np

import csv

import joblib

import pandas as pd

from plotly.subplots import make_subplots

import plotly.graph_objects as go

import plotly

import  random


import ibm_db

import pandas

import ibm_db_dbi

from sqlalchemy import create_engine


engine = create_engine('sqlite://',
                  echo = False)

dsn_hostname = "19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "wqs10666"
dsn_pwd = "dERuKeEtxSIEGRrm"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "30699"
dsn_protocol = "TCPIP"
dsn_security = "SSL"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
    "SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid, dsn_pwd,dsn_security)



try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ", dsn_hostname)

except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )



app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'
```

```
@app.route('/')
def login():
    return render_template('UserLogin.html')


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


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

        name1 = request.form['name']
        gender1 = request.form['gender']
        Age = request.form['age']
        email = request.form['email']
        address = request.form['address']
        pnumber = request.form['phone']
        uname = request.form['uname']
        password = request.form['psw']


        conn = ibm_db.connect(dsn, "", "")


        insertQuery = "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 +
"','" + Age + "','" + email + "','" + pnumber + "','" + password + "','" + uname + "','"
+ address + "')"
        insert_table = ibm_db.exec_immediate (conn, insertQuery)
        print(insert_table)









    return render_template('userlogin.html')
@app.route("/userlogin", methods=['GET', 'POST'])
def userlogin():
    error = None
    if request.method == 'POST':
```

```
    username = request.form['uname']

    password = request.form['password']

    session['uname'] = request.form['uname']



    conn = ibm_db.connect(dsn, "", "")

    pd_conn = ibm_db_dbi.Connection(conn)


    selectQuery = "SELECT * from regtb where uname='" + username + "' and password='" + password + "'"

    dataframe = pandas.read_sql(selectQuery, pd_conn)


    if dataframe.empty:

        data1 = 'Username or Password is wrong'

        return render_template('goback.html', data=data1)

    else:

        print("Login")

        selectQuery = "SELECT * from regtb where uname='" + username + "' and password='" + password + "'"

        dataframe = pandas.read_sql(selectQuery, pd_conn)
        dataframe.to_sql('Employee_Data',
                con=engine,
                if_exists='append')

        # run a sql query
        print(engine.execute("SELECT * FROM Employee_Data").fetchall())

        return render_template('index.html', data=engine.execute("SELECT * FROM Employee_Data").fetchall())

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

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


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



    model = joblib.load('lightgbm_.pkl')


    classes = ['Paddy', 'Cholam', 'Cumbu', 'Ragi','Cotton', 'Sugarcane','Chilli', 'Pigeon Pea',
                'Coconut', 'Tobacco', 'Onion', 'Banana','Mangoes', 'Turmeric', 'Groundnut', 'BlackGram',
                'Maize', 'Tapioca','Tomoto', 'Brinjal', 'Carrot', 'Beans']


    values = []
    if request.method == 'POST':
        values.append(float(request.form.get('nitrogen')))
        values.append(float(request.form.get('phosphorous')))
        values.append(float(request.form.get('potassium')))
        values.append(float(request.form.get('temperature')))
        values.append(float(request.form.get('humidity')))
        values.append(float(request.form.get('ph')))
        values.append(float(request.form.get('rainfall')))

        # answer = model.predict([values])
        predict_pro = model.predict_proba([values])
        list_proba = []
        for i in [-1, -2, -3, -4, -5]:
            list_proba.append(classes[np.argsort(np.max(predict_pro, axis=0))[i]])
        # print(list_proba)
        return render_template('result.html',probab = list_proba)

@app.route('/analysis')
def analysis():
    df = pd.read_csv('data.csv')
    def intractive_plot(df, feature, name):
        colorarr = ['#0592D0','#Cd7f32', '#E97451', '#Bdb76b', '#954535', '#C2b280', '#808000','#C2b280', '#E4d008', '#9acd32', '#Eedc82', '#E4d96f',
               '#32cd32','#39ff14','#00ff7f', '#008080', '#36454f', '#F88379', '#Ff4500', '#Ffb347', '#A94064', '#E75480', '#Ffb6c1', '#E5e4e2',
               '#Faf0e6', '#8c92ac', '#Dbd7d2','#A7a6ba', '#B38b6d']


        df_label = pd.pivot_table(df, index=['label'], aggfunc='mean')

        df_label_feature = df_label.sort_values(by=feature, ascending = False)


        fig = make_subplots(rows = 1, cols = 2)


        top = {


            'y': df_label_feature[feature][:10].sort_values().index,

            'x': df_label_feature[feature][:10].sort_values()

        }

        last = {


            'y': df_label_feature[feature][-10:].sort_values().index,

            'x': df_label_feature[feature][-10:].sort_values()

        }


        fig.add_trace(
            go.Bar(top,
                   name='Least {} Needed'.format(name),
                   marker_color = random.choice(colorarr),
                   orientation = 'h',
                   text = top['x']
                  ),
            row = 1, col = 1
        )
        fig.add_trace(
            go.Bar(last,
                   name='Least {} Needed'.format(name),
                   marker_color = random.choice(colorarr),
                   orientation = 'h',
                   text = top['x']
                  ),
            row = 1, col = 2
        )

        fig.update_traces(texttemplate = '%{text}', textposition = 'inside')
```

```
    fig.update_layout(title_text = name,
                      plot_bgcolor = 'white',
                      font_size = 12,
                      font_color = 'black',
                      height = 500
                     )
    fig.update_xaxes(showgrid = False)
    fig.update_yaxes(showgrid = False)
    fig.show()

intractive_plot(df, feature = 'N', name = 'Nitrogen')
intractive_plot(df, feature = 'P', name = 'Phosphorous')
intractive_plot(df, feature = 'K', name = 'Potassium')
intractive_plot(df, feature = 'humidity', name = 'Humidity')
intractive_plot(df, feature = 'temperature', name = 'Temperature')
intractive_plot(df, feature = 'ph', name = 'ph')
intractive_plot(df, feature = 'rainfall', name = 'Rainfall')
return render_template('predict.html')
```

```
@app.route('/real')

def real():

    return render_template('real.html')



@app.route('/l1')

def l1():

    l1 = [[68.6,2.2,50,33,60,0,67]]

    return render_template('predict1.html',data=l1,msg='Land 1')



@app.route('/l2')

def l2():

    l2 = [[75.6,3,58.3,34,61,0,70]]

    return render_template('predict1.html',data=l2,msg='Land 2')



@app.route('/l3')

def l3():

    l3 = [[71.4,4,70.8,33,62,0,81]]

    return render_template('predict1.html',data=l3,msg='Land 3')
```

```
@app.route('/l4')

def l4():

    l4 = [[62.2,4.2,66.6,35,59,0,110]]

    return render_template('predict1.html',data=l4,msg='Land 4')



@app.route('/Paddy-details')

def paddy_detail():

    return render_template('paddy_details.html')



@app.route('/Paddy-disease')

def paddy_disease():

    return render_template('paddy_disease.html')



@app.route('/Paddy-fertilizer')

def paddy_ferti():

    return render_template('paddy_ferti.html')



@app.route('/Cholam-details')

def cholam_detail():
    return render_template('cholam_details.html')


@app.route('/Cholam-disease')
def cholam_disease():
    return render_template('cholam_disease.html')


@app.route('/Cholam-fertilizer')
def cholam_ferti():
    return render_template('cholam_ferti.html')


@app.route('/Cumbu-details')
def cumbu_detail():
    return render_template('cumbu_details.html')


@app.route('/Cumbu-disease')
def cumbu_disease():
    return render_template('cumbu_disease.html')


@app.route('/Cumbu-fertilizer')
```

```
def cumbu_ferti():

    return render_template('cumbu_ferti.html')



@app.route('/Ragi-details')

def ragi_detail():

    return render_template('ragi_details.html')



@app.route('/Ragi-disease')

def ragi_disease():

    return render_template('ragi_disease.html')



@app.route('/Ragi-fertilizer')

def ragi_ferti():

    return render_template('ragi_ferti.html')



@app.route('/Cotton-details')

def cotton_detail():

    return render_template('cotton_details.html')
```

```
@app.route('/Cotton-disease')
def cotton_disease():
    return render_template('cotton_disease.html')


@app.route('/Cotton-fertilizer')
def cotton_ferti():
    return render_template('cotton_ferti.html')


@app.route('/Sugarcane-details')
def sugarcane_detail():
    return render_template('sugarcane_details.html')


@app.route('/Sugarcane-disease')
def sugarcane_disease():
    return render_template('sugarcane_disease.html')


@app.route('/Sugarcane-fertilizer')
def sugarcane_ferti():
    return render_template('sugarcane_ferti.html')
```

```
@app.route('/Chilli-details')
def chilli_detail():
    return render_template('chilli_details.html')


@app.route('/Chilli-disease')
def chilli_disease():
    return render_template('chilli_disease.html')


@app.route('/Chilli-fertilizer')
def chilli_ferti():
    return render_template('chilli_ferti.html')


@app.route('/Pigeon Pea-details')
def pigeon_detail():
    return render_template('pigeon_details.html')


@app.route('/Pigeon Pea-disease')
def pigeon_disease():
    return render_template('pigeon_disease.html')


@app.route('/Pigeon Pea-fertilizer')
def pigeon_ferti():
    return render_template('pigeon_ferti.html')


@app.route('/Coconut-details')
def coconut_detail():
    return render_template('coconut_details.html')


@app.route('/Coconut-disease')
def coconut_disease():
    return render_template('coconut_disease.html')


@app.route('/Coconut-fertilizer')
def coconut_ferti():
    return render_template('coconut_ferti.html')


@app.route('/Tobacco-details')
```

```
def tobacco_detail():

    return render_template('tobacco_details.html')



@app.route('/Tobacco-disease')

def tobacco_disease():

    return render_template('tobacco_disease.html')



@app.route('/Tobacco-fertilizer')

def tobacco_ferti():

    return render_template('tobacco_ferti.html')



@app.route('/Onion-details')

def onion_detail():

    return render_template('onion_details.html')



@app.route('/Onion-disease')

def onion_disease():

    return render_template('onion_disease.html')
```

```
@app.route('/Onion-fertilizer')

def onion_ferti():

    return render_template('onion_ferti.html')



@app.route('/Banana-details')

def banana_detail():

    return render_template('banana_details.html')



@app.route('/Banana-disease')

def banana_disease():

    return render_template('banana_disease.html')



@app.route('/Banana-fertilizer')

def banana_ferti():

    return render_template('banana_ferti.html')



@app.route('/Mangoes-details')

def mango_detail():

    return render_template('mango_details.html')
```

```
@app.route('/Mangoes-disease')

def mango_disease():

    return render_template('mango_disease.html')



@app.route('/Mangoes-fertilizer')

def mango_ferti():

    return render_template('mango_ferti.html')



@app.route('/Turmeric-details')

def termeric_detail():

    return render_template('termeric_details.html')



@app.route('/Turmeric-disease')

def termeric_disease():

    return render_template('termeric_disease.html')



@app.route('/Turmeric-fertilizer')

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


@app.route('/Groundnut-details')
def ground_detail():
    return render_template('ground_details.html')


@app.route('/Groundnut-disease')
def ground_disease():
    return render_template('ground_disease.html')


@app.route('/Groundnut-fertilizer')
def ground_ferti():
    return render_template('ground_ferti.html')


@app.route('/BlackGram-details')
def black_detail():
    return render_template('black_details.html')


@app.route('/BlackGram-disease')
```

```
def black_disease():
    return render_template('black_disease.html')

@app.route('/BlackGram-fertilizer')
def black_ferti():
    return render_template('black_ferti.html')

@app.route('/Maize-details')
def maize_detail():
    return render_template('maize_details.html')

@app.route('/Maize-disease')
def maize_disease():
    return render_template('maize_disease.html')

@app.route('/Maize-fertilizer')
def maize_ferti():
    return render_template('maize_ferti.html')
```

```
@app.route('/Tapioca-details')

def topi_detail():

    return render_template('topi_details.html')



@app.route('/Tapioca-disease')

def topi_disease():

    return render_template('topi_disease.html')



@app.route('/Tapioca-fertilizer')

def topi_ferti():

    return render_template('topi_ferti.html')



@app.route('/Tomoto-details')

def tomoto_detail():

    return render_template('tomoto_details.html')



@app.route('/Tomoto-disease')

def tomoto_disease():

    return render_template('tomoto_disease.html')
```

```
@app.route('/Tomoto-fertilizer')

def tomoto_ferti():

    return render_template('tomoto_ferti.html')



@app.route('/Brinjal-details')

def brinjal_detail():

    return render_template('brin_details.html')



@app.route('/Brinjal-disease')

def brinjal_disease():

    return render_template('brin_disease.html')



@app.route('/Brinjal-fertilizer')

def brinjal_ferti():

    return render_template('brin_ferti.html')



@app.route('/Carrot-details')

def carrot_detail():
    return render_template('carrot_details.html')


@app.route('/Carrot-disease')
def carrot_disease():
    return render_template('carrot_disease.html')


@app.route('/Carrot-fertilizer')
def carrot_ferti():
    return render_template('carrot_ferti.html')


@app.route('/Beans-details')
def bean_detail():
    return render_template('bean_details.html')


@app.route('/Beans-disease')
def bean_disease():
    return render_template('bean_disease.html')


@app.route('/Beans-fertilizer')
```

```
def bean_ferti():

    return render_template('bean_ferti.html')



if __name__ == '__main__':

    app.run(debug=True)
```

DRIVE LINK : https://drive.google.com/file/d/1SIQJmsaFhGl_lf0nfcAYsQ_BShMwZZ2I/view?usp=drivesdk

GITHUB LINK : L/IBM-Project-43281-1ht//github.com/IBM-EPB660715142