CROP SMART AND LOCATION-SPECIFIC FARMING RECOMMENDATIONS FOR AGRICULTURE

A PROJECT REPORT

Submitted by

KOWSALYA J	**-**	**421621104070**
LATHIKA V	**-**	**421621104073**
ROSELINE PRINCY V	**-**	**421621104120**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

MAILAM ENGINEERING COLLEGE

MAILAM

ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY:: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"CROP SMART AND LOCATION-SPECIFIC FARMING RECOMMENDATIONS FOR AGRICULTURAL "** is the Bonafide work of **KOWSALYA . J (421621104070) , ROSELINE PRINCY. V (421621104120) , LATHIKA .V (421621104073)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Mrs.E.LAVANYA,ME.,	**Dr.T.PRIYARADHIKADEVI,M.Tech,Ph.D,**
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Associate professor,	Professor and Head,
Computer Science and Engineering,	Computer Science and Engineering,
Mailam Engineering College,	Mailam Engineering College,
Mailam-604 304.	Mailam-604 304.

Submitted for the project and Viva-Voce Examination held on___________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

With the blessings of almighty, I express my immense gratitude to my beloved parents.

Owing deeply to supreme, express my sincere thanks to our honorable chairman and Managing Director **Shri.M.DHANASEKARAN,** Our beloved Vice Chairman

Shri. S.V.SUGUMARAN, and Secretary of our college **Dr.K.NARAYANASAMY M.B.B.S.,** Treasurer **Mr.D.RAJARAJAN** who entered his generosity to give the facilities to undergo the project work till its completion.

I am very much revered to our Director **Dr.S.SENTHIL,M.E,Ph.D.,** for his support and encouragement shown towards our academics profile. We tender our heartfelt gratitude to our beloved Principal of Our college **Dr.R.RAJAPPAN, M.Tech., Ph.D., MISTE** for his valuable and encouragement throughout this endeavor.

I extend my sincere thanks and gratitude to our Head of The Department. **Dr.T.PRIYARADHIKADEVI,M.Tech,Ph.D.,** for sharing her thoughts in doing the project and the encouragement shown towards the project work.

I am very much obliged to my project supervisor **Mrs.E.LAVANYA,M.E,Associate Professor.,** for his motivation, guidelines and continuous support for conducting the reviews at each and every level of work.

I also extend my sincere thanks to all the staff members and non-teaching staff of our department. I sincerely thank all of them.

ABSTRACT

This project introduces a Smart Agri is a comprehensive, software-based, location-aware decision support system designed to advance sustainable and precision agriculture without relying on IoT or physical sensor networks. The system leverages a combination of geospatial technologies, satellite imagery, weather forecasting models, historical climate data, and agricultural databases to offer intelligent, location-specific insights to farmers and agricultural planners. Through the integration of GIS mapping and data analytics, Smart Agri facilitates informed decision-making in crop selection, planting schedules, soil and land assessment, irrigation planning, and pest/disease risk management. The platform employs rule-based logic and machine learning techniques to process diverse datasets and generate customized recommendations aligned with sustainable practices. Its user-centric interface ensures accessibility for users with varying technical expertise, while its modular architecture supports scalability and regional adaptability. By reducing dependency on hardware infrastructure and emphasizing cost-effective, software-driven solutions, Smart Agri empowers farmers—especially in resource-constrained environments—to enhance productivity, conserve natural resources, and mitigate the environmental impact of agricultural activities. The system stands as a vital tool for promoting climate-resilient, data-driven farming in the digital agent.

ஆய்வுச்சுறுக்கம்

SmartAgri என்பது தோட்டச்செய்கை மற்றும் விவசாய நிர்வாகத்தில் நிலைத்தன்மையும் துல்லியமும் அடிப்படையாகக் கொண்ட, தோற்றவியல் அடிப்படையிலான, மென்பொருள் மூலமான முடிவு எடுக்கும் ஆதரவு அமைப்பாகும். இந்த அமைப்பு IoT அல்லது பைதியியல் உணரிகள் போன்ற உடைமை சார்ந்த தொழில்நுட்பங்களைப் பயன்படுத்தாமல், செயற்கைக்கோள் படங்கள், நிலத்தோற்ற தகவல்கள் (GIS), வானிலை கணிப்புகள், வரலாற்று காலநிலை பதிவுகள் மற்றும் விவசாயத் தரவுத்தொகுப்புகளை ஒருங்கிணைத்து செயல்படுகிறது. SmartAgri, பயிர் தேர்வு, விதைபோதும் கால அட்டவணை, நில நிலை மதிப்பீடு, பாசன திட்டமிடல் மற்றும் பூச்சி/நோய் அபாயக் கணிப்பில் விவசாயிகளுக்கும் திட்டமாளர்களுக்கும் மதிப்புள்ள பரிந்துரைகளை வழங்குகிறது. இம்முறை, விதிமுறை அடிப்படையிலான தரவாய்வு மற்றும் மெஷின் லெர்னிங் தொழில்நுட்பங்களைப் பயன்படுத்தி, நிலவிய தரவுகளை பகுப்பாய்வு செய்து, சுற்றுச்சூழல் நடத்தை மற்றும் நிலைத்த சேவைகளை முன்னெடுக்கும் வகையில் பரிந்துரைகள் வழங்கப்படுகின்றன. பயன்படுத்த எளிய இடைமுகம் மற்றும் நிலைபேறான கட்டமைப்பைக் கொண்ட SmartAgri, தொழில்நுட்ப அறிவு குறைவான பயனாளர்களுக்கும் பயன்படுத்தும் வாய்ப்பை வழங்குகிறது. குறைந்த செலவில், மென்பொருள் சார்ந்த தீர்வுகளை மையமாகக் கொண்டு, இது பசுமை விவசாயத்தை ஊக்குவிக்கிறது மற்றும் வளங்களை பாதுகாத்து, உற்பத்தி திறனை அதிகரிக்க உதவுகிறது. குறைந்த வளம் கொண்ட பகுதிகளில் கூட, தரவுகள் அடிப்படையிலான, காலநிலை நெருக்கடியை சமாளிக்கும் விவசாயத்தை முன்னெடுக்க இது ஒரு முக்கிய சாதனமாக விளங்குகிறது.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Agriculture, especially in developing regions, continues to face significant challenges due to the lack of localized, data-driven decision-making tools. Most farmers still rely on traditional practices and generalized recommendations that do not consider the unique environmental, climatic, and soil conditions of individual plots of land. As a result, issues such as inefficient resource usage, reduced crop yields, soil degradation, and vulnerability to climate change persist.

While precision agriculture technologies have emerged to address these challenges, they often depend on costly hardware solutions like IoT sensors, drones, and automated machinery. These technologies, although effective, are financially and logistically out of reach for small and marginal farmers, who represent a significant portion of the agricultural community.

Moreover, the digital tools that are available are either too complex for widespread adoption or lack the ability to integrate and process location-specific data in a meaningful way. This creates a technological gap where affordable and scalable software solutions are missing—tools that could otherwise provide tailored recommendations based on factors like soil type, crop history, weather patterns, and geographic features.

There is a clear need for an accessible, software-only decision support system that can leverage freely available data sources—such as satellite imagery, GIS databases, and historical weather records—to deliver intelligent, location-aware insights for better farming decisions. Such a system would bridge the digital divide in agriculture and promote sustainability, productivity, and resilience without the need for physical infrastructure investments.

1.2 SCOPE OF THE PROJECT

The Smart Agri project aims to develop a software-based, location-aware decision support system that empowers farmers and agricultural stakeholders to adopt sustainable and precision farming practices without relying on physical sensors or IoT infrastructure.

Key areas covered within the scope include:

1.Geospatial Data Integration:
Integration of GIS data, satellite imagery, and topographical information to analyze the physical and environmental characteristics of farmland.

2.Weather and Climate Analysis:
Utilization of historical and real-time weather data to assess climate suitability for various crops and recommend optimal planting and harvesting times.

3.Crop Recommendation System:
A rule-based or data-driven module that suggests suitable crops based on soil type, climatic conditions, and previous crop cycles for a given location.

4.Resource Optimization:
Software-generated recommendations for efficient use of water, fertilizer, and land to reduce waste and improve yield potential.

5.User-Friendly Interface:
A simple, intuitive web or mobile interface that allows users to input location or select maps and receive personalized, actionable advice in their preferred language.

6.Scalability and Accessibility:
The system is designed to be scalable across different regions and adaptable to regional agricultural needs and languages, making it suitable for both individual farmers and agricultural agencies.

7.Sustainability Focus:
All recommendations will be aligned with sustainable agricultural practices, promoting long-term soil health, biodiversity, and reduced environmental impact.

1.3 PROPOSED SYSTEM HIGHLIGHTS

Smart Agri is a software-driven, location-aware decision support system developed to assist farmers and agricultural planners in making smarter, data-informed decisions for sustainable and precision farming. Unlike traditional agricultural tools that depend heavily on physical sensors and IoT infrastructure, Smart Agri functions purely on software by integrating and analyzing openly available data such as satellite imagery, GIS data, weather forecasts, soil maps, and historical agricultural records.

The system is designed to provide personalized recommendations for crop selection, sowing time, resource management, and land-use planning based on the user's specific geographic location and environmental conditions. By doing so, Smart Agri aims to improve agricultural productivity, reduce resource wastage, and support environmentally responsible practices.

Smart Agri is particularly suited for regions where access to advanced agricultural technology is limited. With its user-friendly interface, localized support, and cost-effective approach, the project strives to bridge the digital divide in agriculture and empower small and marginal farmers with the benefits of precision farming—without the need for expensive hardware or technical expertise.

1.4 PURPOSES

The primary purpose of the Smart Agri project is to develop a cost-effective, software-based decision support system that enables farmers to make informed, location-specific agricultural decisions. The project aims to bridge the gap between traditional farming practices and modern data-driven agriculture by providing a user-friendly platform that leverages existing digital data sources—such as satellite imagery, GIS mapping, weather data, and soil databases.

Smart Agri is designed to assist in critical farming decisions such as crop selection, planting schedules, and efficient resource utilization. By offering region-specific and climate-aware recommendations, the system promotes sustainable agriculture, improves crop productivity, and minimizes environmental impact.

The overarching goal is to empower small and marginal farmers—who may not have access to expensive smart farming technologies—with a practical tool that supports precision farming using only software-based intelligence. In doing so, Smart Agri contributes to the advancement of climate-resilient agriculture and supports food security through improved farming outcomes.

CHAPTER 2

SYSTEM ANALYSIS

System analysis is the process of gathering facts, interpreting facts and diagnosing problem by using the information for improvements on the system

The act, process or profession of studying an activity (such as a procedure, a business, or a physiological function) typically by mathematical means in order to define its goals or purposes and to discover operations and procedures for accomplishing them most efficiently.

2.1 EXISTING SYSTEM

Most farmers rely on experience-based or generalized recommendations without considering specific environmental or geographical conditions. This often leads to inefficient resource usage, lower yields, and environmental degradation.

- High in cost and maintenance
- Complex to use
- Inaccessible to small or marginal farmers in rural or resource-constrained areas

Limited Accessibility and Localization:
Many existing tools are not user-friendly, lack support for regional languages, and do not cater to the local context in terms of crops, climate.

2.2 PROPOSED SYSTEM

- Smart Agri eliminates the need for hardware by leveraging readily available datasets such as satellite imagery, GIS data, historical weather patterns, and soil maps to deliver smart, location-aware insights.

- Localized and Personalized Recommendations:
  The system provides specific advice on:

    - Suitable crops based on soil and climate
    - Optimal planting and harvesting periods
    - Efficient resource management practices

- Cost effective and Scalable:

Designed to be affordable and accessible, Smart Agri is suitable for small-scale farmers and can be deployed across regions without significant infrastructure requirements.

- User-Friendly Interface:

The platform features an intuitive interface with support for local languages, allowing farmers to interact with the system easily, regardless of their technical background.

- Promotes Sustainability:

By focusing on precision without hardware, the system helps reduce overuse of water, fertilizers, and pesticides, thereby promoting environmentally sustainable practices.

2.3 FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This system can used to marking attendance in the fraction of second. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

CHAPTER 3
SYSTEM SPECIFICATION

A system requirement is the analysis which is needed for the purposed system to run ona particular machine. It gives the minimum requirements to execute the instruction in themachine. They are,

1. Hardware requirements
2. Software requirements

3.1 Hardware Requirement

A system requirement is the analysis. This is need for the proposed requirement to execute the instruction in the machine.

1.	Speed	**:** 2.00 GHz
2.	Primary memory	**:** 4 GB RAM
3.	Hard disk	**:** 1 GB
4.	Mouse	: three button mouse
5.	Camera	:Webcam

3.2 Software Requirement

The software requirement deal with the analysis of the software and operating system and the tools those are essential for the project.

1.	Processor	**:** AMD A4
2.	Operating System	**:** Windows family
3.	Tool	**:** Phycharm 3.7
4.	Front End	**:** Phython,HTML
5.	Back End	**:** SQL (Xampp)

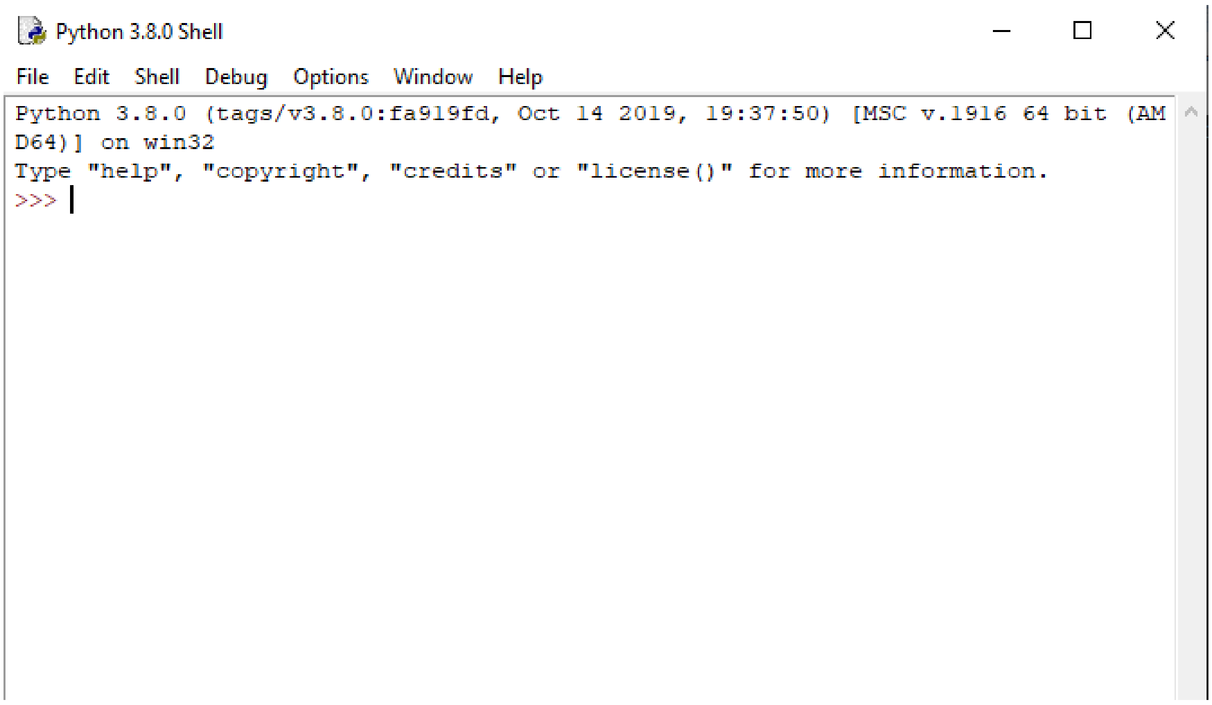
CHAPTER 4

SYSTEM DESCRIPTION

4.1 TOOL DESCRIPTION

Python 3.8 IDE

The python integrated development environment is a creative launching pad that we can use to edit, debug and write code and then run the system. An integrated development environment (IDE) is a feature rich program that can be used for many aspects of software development.

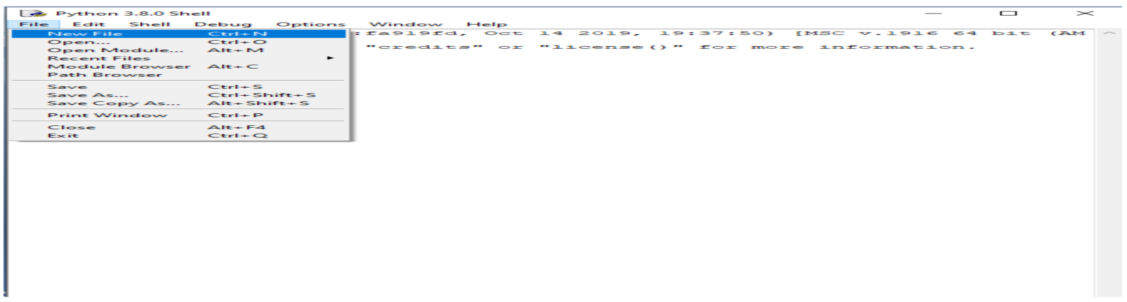


```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

FIG 4.1.1. PYTHON 3.8 IDLE

The image has several tools that we would use are:

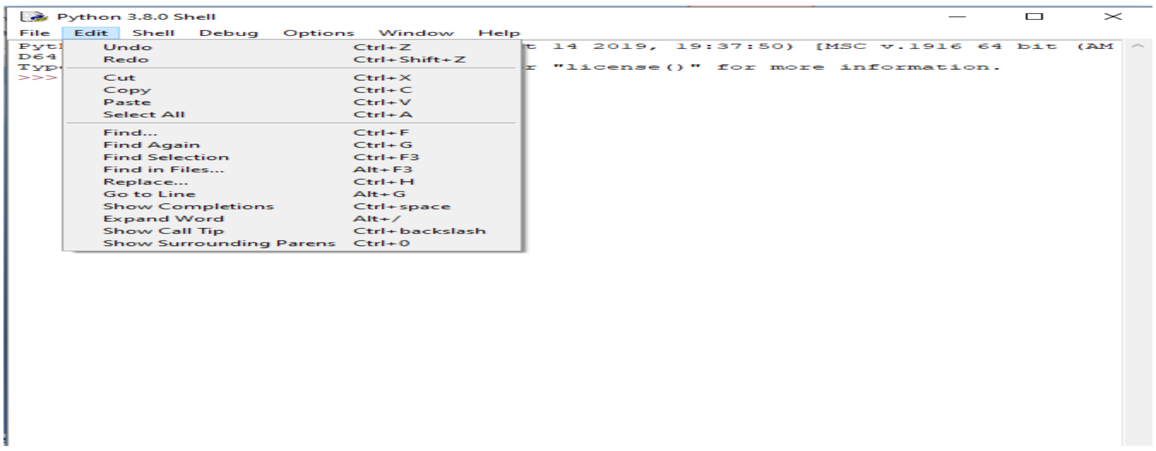
- File tab consists of create new file, open the existing file, save the file with save and save us, print the file and close the file



```
:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
"credits" or "license()" for more information.
```

FIG 4.1.2. FILE TAB

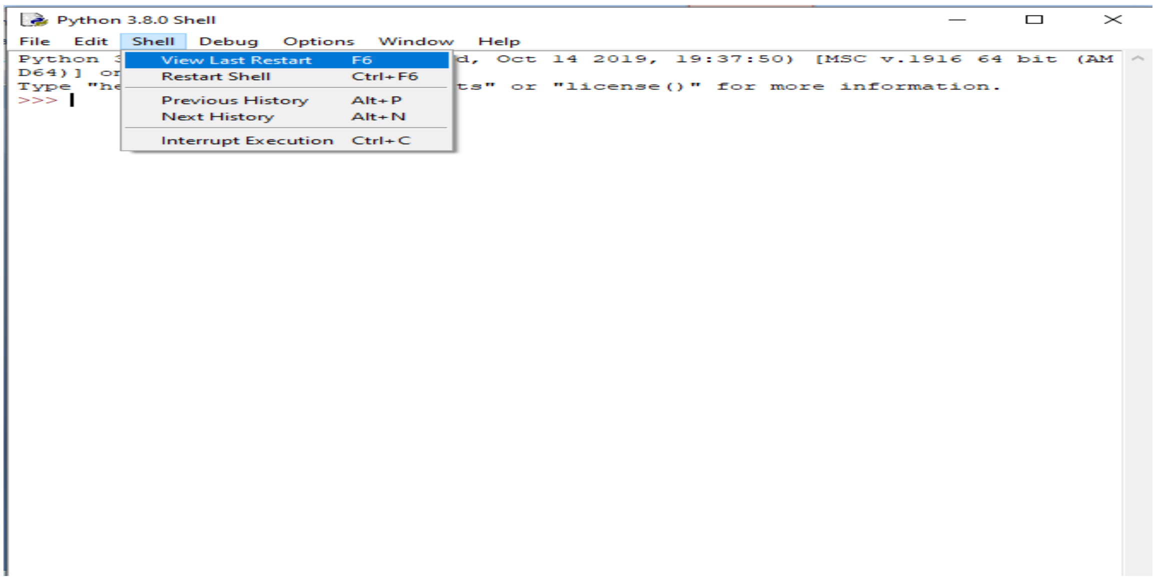
- Edit tab consist of undo, redo, find and replace, goto, show call tip, show surroundings



```
Pyt                                       t 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64
Typ                                       r "license()" for more information.
>>>
```

FIG 4.1.3 EDIT TAB

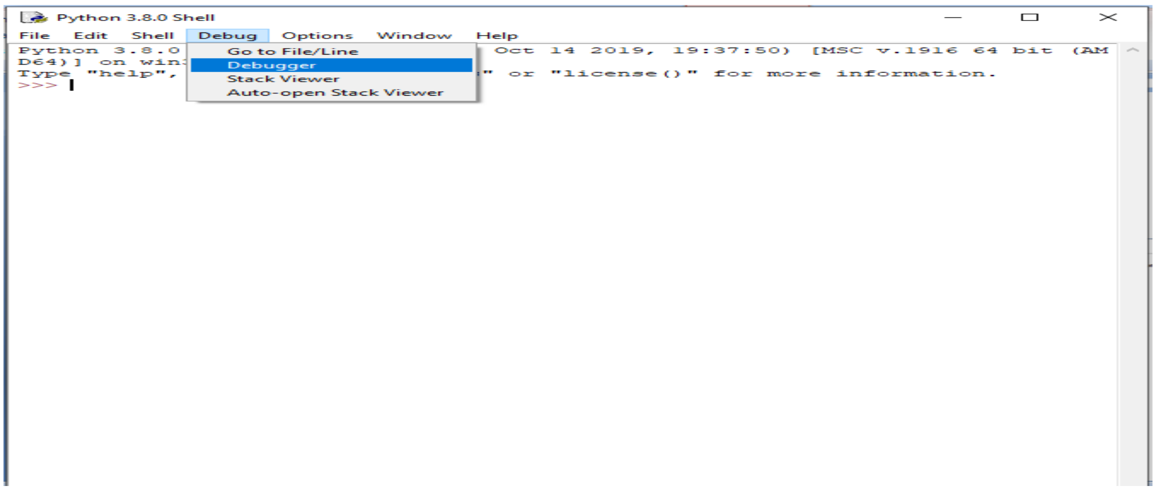
- Shell tab is used to view the recent and previous history and restart the shell



```
Python 3                          d, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] or
Type "he                          ts" or "license()" for more information.
>>> |
```

FIG 4.1.4 SHELL TAB

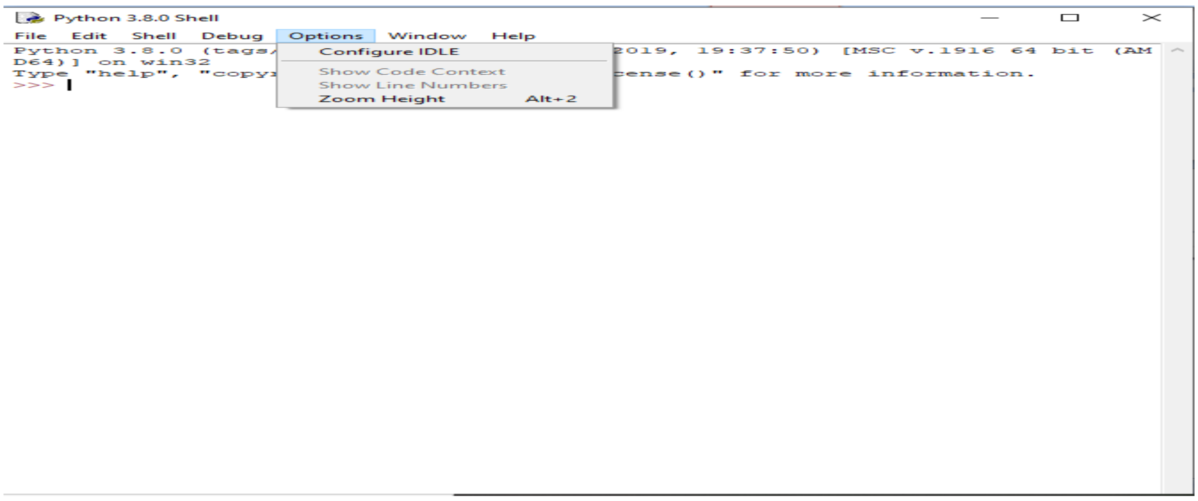
- Debug tab used to debug the program and it consist of goto, stack viewer, auto-open stack viewer



```
Python 3.8.0                 Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win
Type "help",               " or "license()" for more information.
>>> 
```

FIG 4.1.5. DEBUG TAB

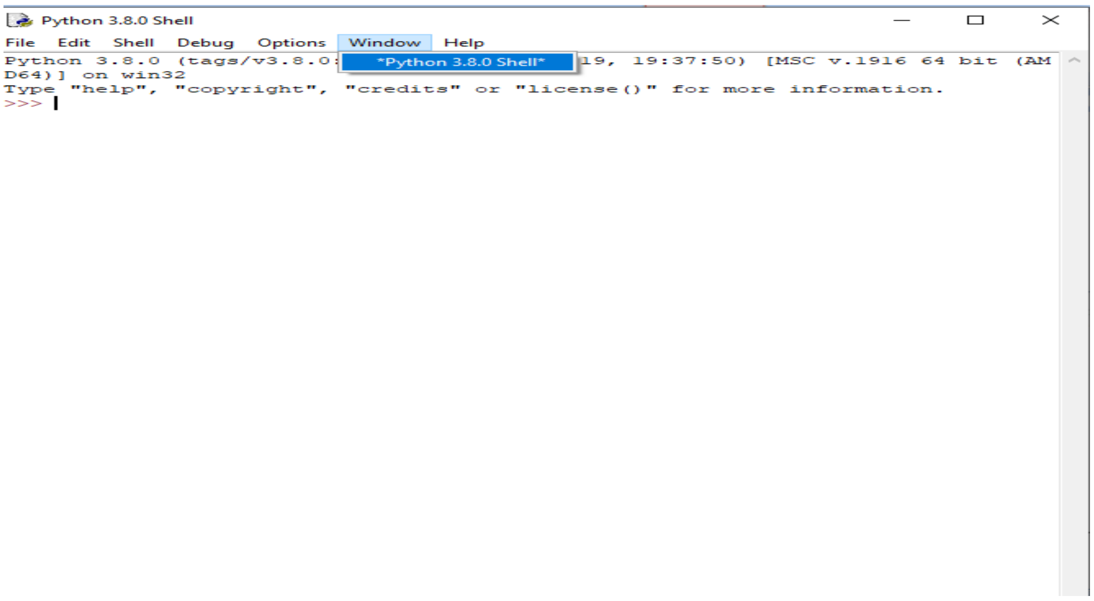
- Option tab consist of configure IDE, zoom height, Show line number and code context



```
Python 3.8.0 (tags/                      2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyr                      ense()" for more information.
>>> 
```

FIG 4.1.6. OPTION TAB

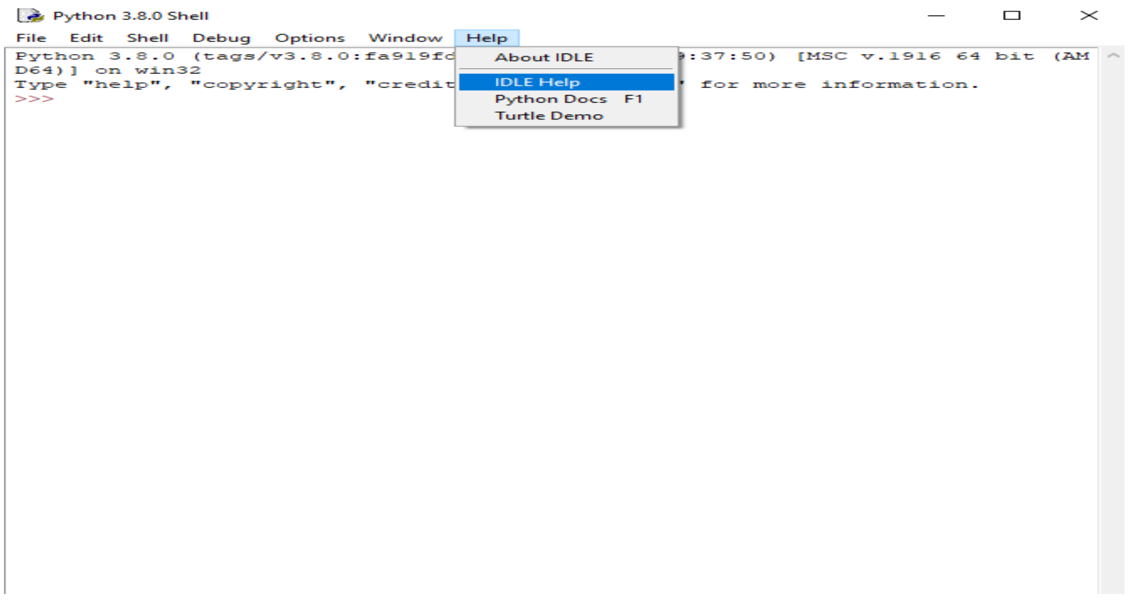
- Window tab consist of python 3.8 shell which activate shell window



```
Python 3.8.0 (tags/v3.8.0:                 19, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

FIG 4.1.7. WINDOW TAB

- Help tab allow the user to seek help from the IDE and it provide demo in it

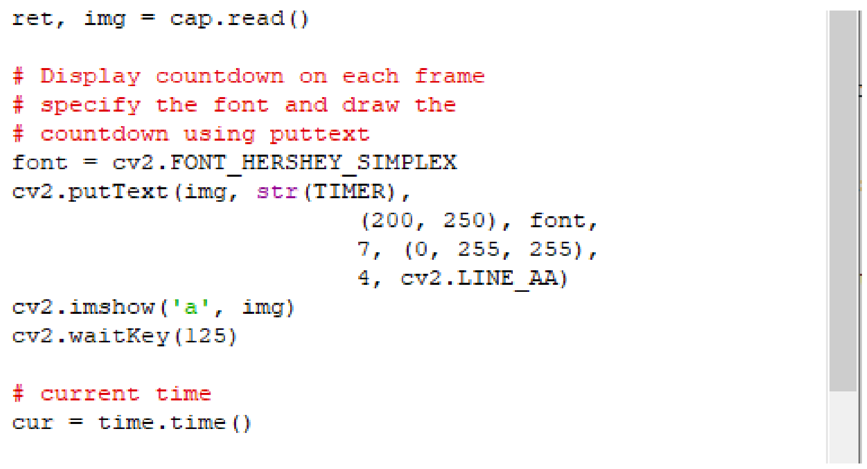


```
Python 3.8.0 (tags/v3.8.0:fa919fd          :37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credit            " for more information.
>>>
```

FIG 4.1.8. HELP TAB

Features of Python:

The image can be captured by the following coding in python.It can capture the image of the person in gray scale . Then save the image with name of the faculty



```
ret, img = cap.read()

# Display countdown on each frame
# specify the font and draw the
# countdown using puttext
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img, str(TIMER),
                        (200, 250), font,
                        7, (0, 255, 255),
                        4, cv2.LINE_AA)
cv2.imshow('a', img)
cv2.waitKey(125)

# current time
cur = time.time()
```

FIG 4.1.9 CAPTURE IMAGE

Packages:

- **TK-TOOLS**
  - Tk/Tcl has long been an integral part of Python. It provides a robust and platform independent windowing toolkit, that is available to Python programmers using the tkinter package, and its extension, the tkinter. tix and the tkinter. ttk modules. The tkinter package is a thin object-oriented layer on top of Tcl/Tk.
  - “pip install tk-tools”
- **OPENCV-CONTRIB-PYTHON**
  - It's a package that contains pre-built OpenCV with dependencies and Python bindings, so there's no need to install OpenCV separately. ... We're proud to bring opencv-python, opencv-python-headless, opencv-contrib-python and opencv-contrib-python-headless home at OpenCV.org.
  - “pip install opencv-contib-python”
- **DATETIME**
  - Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps
  - “pip install datetime”
- **PYTEST-SHUTIL**
  - The shutil module helps you automate copying files and directories. This saves the steps of opening, reading, writing and closing files when there is no actual

processing. It is a utility module which can be used to accomplish tasks, such as: copying, moving, or removing directory trees
    - “pip install pytest-shutil”

- **PYTHON-CSV**
    - CSV (Comma Separated Values) is a simple file format used to store tabular data, such as a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. ... For working CSV files in python, there is an inbuilt module called csv.
    - “pip install python-csv”
- **NUMPY**
    - NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices
    - “pip install numpy”

- **PILLOW**
    - Pillow is a Python Imaging Library (PIL), which adds support for opening, manipulating, and saving images. The current version identifies and reads a large number of formats. Write support is intentionally restricted to the most commonly used interchange and presentation formats.
    - “pip install pillow”
- **PANDAS**
    - Pandas is a widely-used data analysis and manipulation library for Python. It provides numerous functions and methods that expedite the data analysis and preprocessing steps.
    - “pip install pandas”
- **DLIB (Direct Laser Interference Patterning)**
    - It's a landmark's facial detector with pre-trained models, the dlib is used to estimate the location of 68 coordinates (x, y) that map the facial points on a person's face like image
    - “pip install dlib”

Haar cascade classifier

Haar Cascade classifiers are an effective way for object detection. This method was proposed by Paul Viola and Michael Jones in their paper Rapid Object Detection using a Boosted Cascade of Simple Features .Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

- **Positive images –** These images contain the images which we want our classifier to identify.
- **Negative Images –** Images of everything else, which do not contain the object we want to detect.

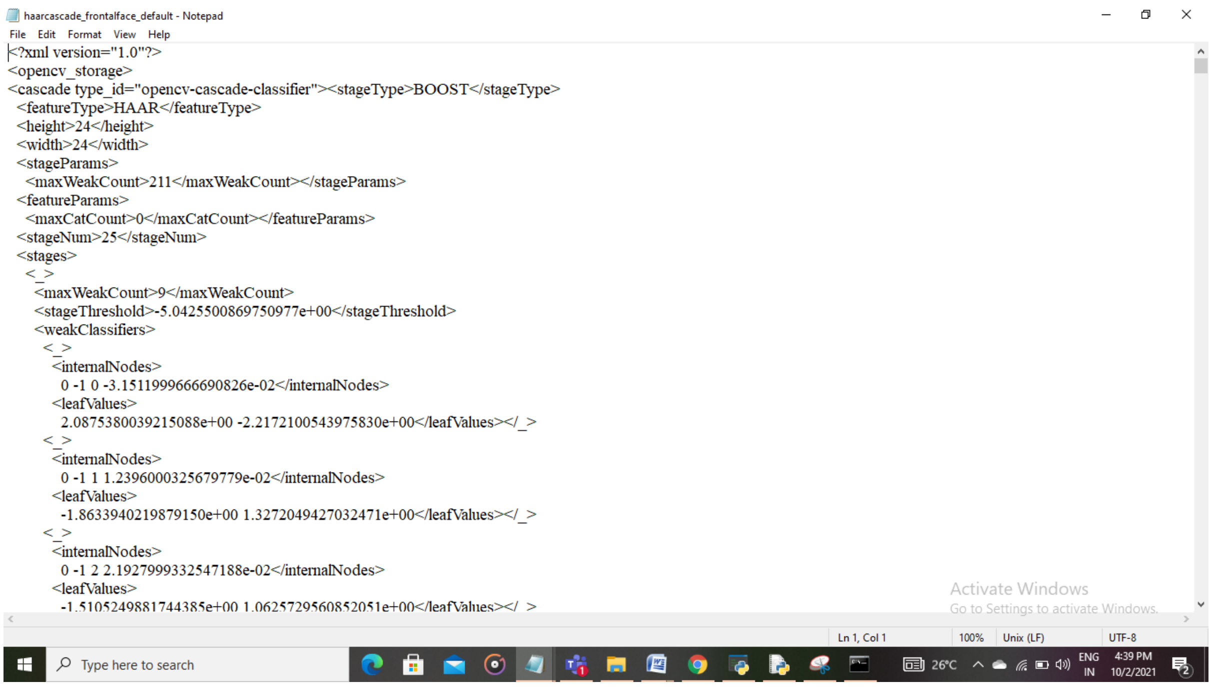

```
<?xml version="1.0"?>
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          <leafValues>
            2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 1.2396000325679779e-02</internalNodes>
          <leafValues>
            -1.8633940219879150e+00 1.3272049427032471e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 2.1927999332547188e-02</internalNodes>
          <leafValues>
            -1.5105249881744385e+00 1.0625729560852051e+00</leafValues></_>
```
haarcascade_frontalface_default - Notepad
File Edit Format View Help
Activate Windows
Go to Settings to activate Windows.
Ln 1, Col 1
100%
Unix (LF)
UTF-8
Type here to search
26°C
ENG IN
4:39 PM 10/2/2021

FIG 4.1.10. HAARCASCADE CLASSIFIER

CHAPTER 5

MODULE DESCRIPTION

Primary database creation and training

Location-Based Crop Selection Module

Covers:

- Collecting user location
- Linking location to region-specific soil, climate, and topographical data
- Identifying crops best suited to the user's local environment
- Ensuring alignment with historical cropping patterns and soil suitability

Rainfall Prediction Module

Covers:

- Integration of weather APIs and historical rainfall datasets
- Predicting short-term and seasonal rainfall trends for the selected region

Yield Prediction Module

Covers:

- Estimating potential yield based on selected crop, location, and weather patterns
- Considering soil quality, rainfall forecasts, and historical yield data
- Providing yield range (minimum-maximum) to aid planning and risk management

Crop Recommendation Module

Covers:

- Recommending suitable crops based on location-specific conditions
- Dynamic recommendations based on seasonal weather and land use
- Factoring in sustainability, profitability, and crop rotation practices

Fertilizer Recommendation Module

Covers:

- Suggesting appropriate type and quantity of fertilizer for selected crops
- Aligning recommendations with soil fertility, crop stage, and rainfall conditions
- Encouraging balanced fertilizer use to reduce cost and environmental harm
- Recommending organic or alternative options where possible

Feedback and Advisory Module

Covers:

- Allowing users to submit feedback on crop performance, weather accuracy, and recommendations .

CHAPTER 6

SYSTEM DESIGN

6.1 USECASE DIAGRAM:

Use case diagram is used to represent the flow of sequence of the project. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization

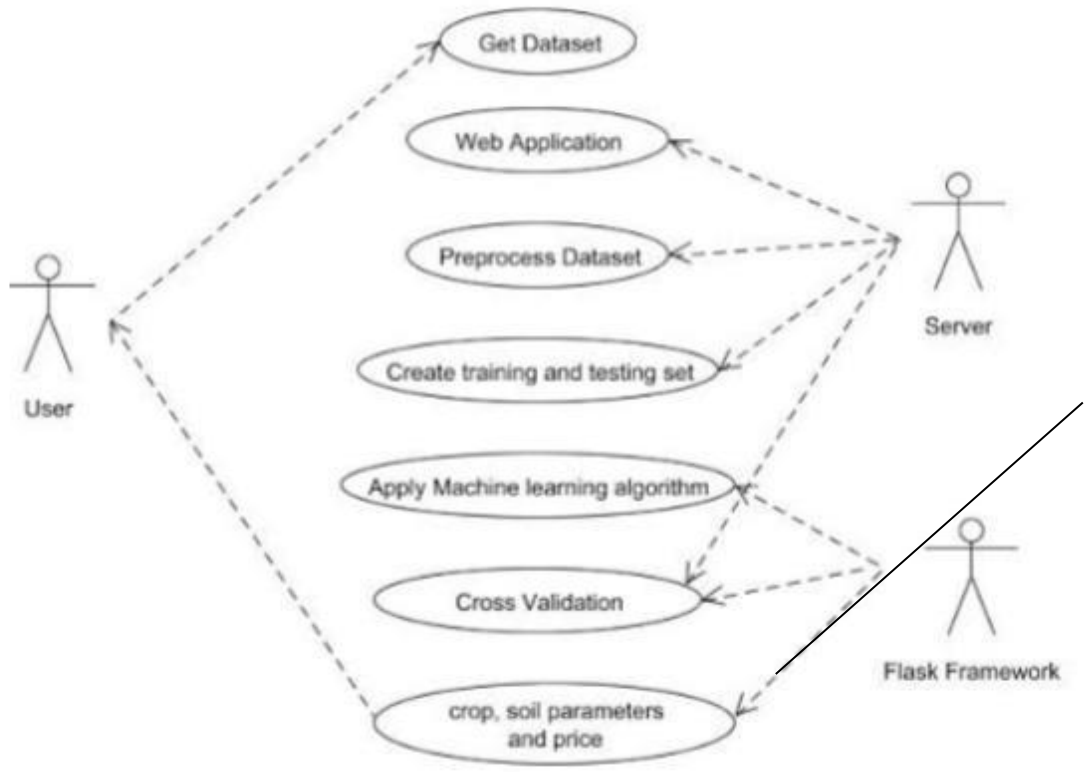

Get Dataset
Web Application
Preprocess Dataset
Create training and testing set
Apply Machine learning algorithm
Cross Validation
crop, soil parameters and price
User
Server
Flask Framework

FIG 6.1. USECASE DIAGRAM

6.2 Class Diagram

Class diagram is used to represent the object, classes, functions used in the project. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.

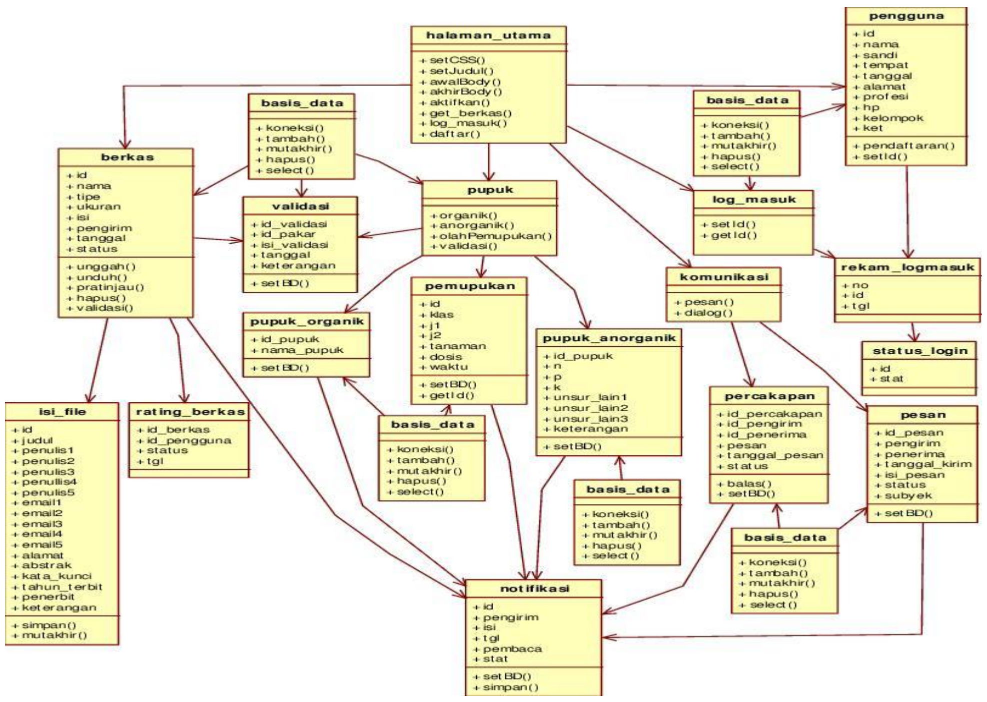


FIG 6.2. CLASS DIAGRAM

6.3 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements

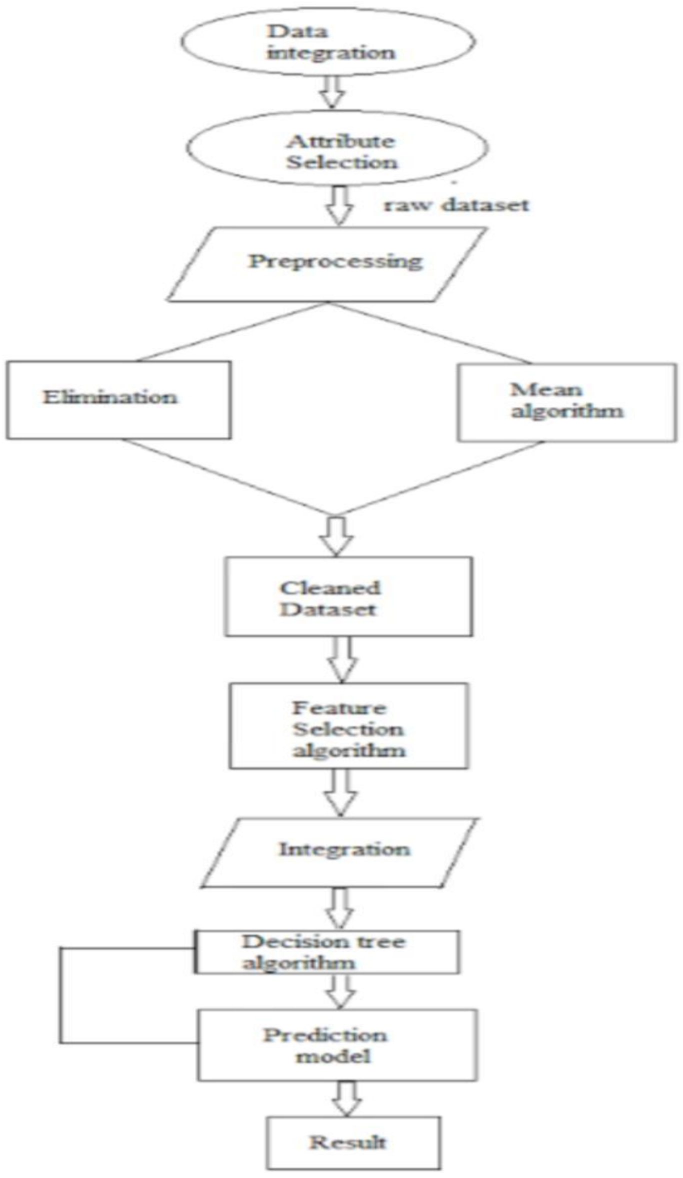


FIG 6.3. ACTIVITY DIAGRAM

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 SOURCE CODE

```
# from flask import Flask, request, render_template
# import pickle
# import numpy as np
# app = Flask(__name__)
# # Load models
# def load_models():
#     try:
#         nb_model = pickle.load(open('models/naive_bayes_model.pkl', 'rb'))
#         rf_model = pickle.load(open('models/random_forest_model.pkl', 'rb'))
#     except Exception as e:
#         print(f"Error loading models: {e}")
#         raise
#     return nb_model, rf_model
# nb_model, rf_model = load_models()
# @app.route('/', methods=['GET', 'POST'])
# def index():
#     if request.method == 'POST':
#         try:
#             # Extracting input data from the form
#             N = float(request.form['N'])
#             P = float(request.form['P'])
#             K = float(request.form['K'])
#             temperature = float(request.form['temperature'])
#           humidity = float(request.form['humidity'])
#           ph = float(request.form['ph'])
#           rainfall = float(request.form['rainfall'])
#           # Preprocess the data and predict
#           crop_features = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
#           crop_prediction = nb_model.predict(crop_features)[0]
#           fertilizer_features = np.array([[N, P, K]])
#           fertilizer_prediction = rf_model.predict(fertilizer_features)[0]
#           # Update fertilizer prediction to actual names
#           fertilizer_dict = {
#               0: 'Urea',
#               1: 'DAP',
#               2: 'Fourteen-Thirty Five-Fourteen',
#               3: 'Twenty Eight-Twenty Eight',
#               4: 'Seventeen-Seventeen-Seventeen',
#               5: 'Twenty-Twenty',
#               6: 'Ten-Twenty Six-Twenty Six'
#               #, , , ,
#           }
#           fertilizer_name = fertilizer_dict.get(fertilizer_prediction, "Unknown Fertilizer")


#           return render_template('index.html', crop_prediction=crop_prediction,
fertilizer_prediction=fertilizer_name)
#       except Exception as e:
#           return render_template('index.html', error=str(e))
#   return render_template('index.html')
# if __name__ == '__main__':
#     app.run(debug=True)
from flask import Flask, request, render_template
import pickle
import numpy as np
app = Flask(__name__)
# Load models
def load_models():
    try:
        nb_model = pickle.load(open('models/naive_bayes_model.pkl', 'rb'))
        rf_model = pickle.load(open('models/random_forest_model.pkl', 'rb'))
    except Exception as e:
        print(f"Error loading models: {e}")
        raise
    return nb_model, rf_model
# Mapping from number to crop name
crop_dict = {
    1: 'rice',
    2: 'maize',
    3: 'jute',
    4: 'cotton',
    5: 'coconut',
    6: 'papaya',
    7: 'orange',
    8: 'apple',
    9: 'muskmelon',
    10: 'watermelon',
    11: 'grapes',
    12: 'mango',
    13: 'banana',
    14: 'pomegranate',
    15: 'lentil',
    16: 'blackgram',
    17: 'mungbean',
    18: 'mothbeans',
    19: 'pigeonpeas',
    20: 'kidneybeans',
    21: 'chickpea',
    22: 'coffee'
}

nb_model, rf_model = load_models()
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        try:
            # Extracting input data from the form
            N = float(request.form['N'])
            P = float(request.form['P'])
            K = float(request.form['K'])
            temperature = float(request.form['temperature'])
            humidity = float(request.form['humidity'])
            ph = float(request.form['ph'])
            rainfall = float(request.form['rainfall'])
            # Preprocess the data and predict
            crop_features = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
            crop_prediction_num = nb_model.predict(crop_features)[0]
            crop_name = crop_dict.get(crop_prediction_num, "Unknown Crop")
           # Convert number to name

            fertilizer_features = np.array([[N, P, K]])
            fertilizer_prediction = rf_model.predict(fertilizer_features)[0]

           # Update fertilizer prediction to actual names
            fertilizer_dict = {
                0: 'Urea',
                1: 'DAP',
                2: 'Fourteen-Thirty Five-Fourteen',
                3: 'Twenty Eight-Twenty Eight',
                4: 'Seventeen-Seventeen-Seventeen',
                5: 'Twenty-Twenty',
                6: 'Ten-Twenty Six-Twenty Six'
            }
            fertilizer_name = fertilizer_dict.get(fertilizer_prediction, "Unknown Fertilizer")
            return render_template('index.html', crop_prediction=crop_name,
fertilizer_prediction=fertilizer_name)
        except Exception as e:
            return render_template('index.html', error=str(e))
return render_template('index.html')
if __name__ == '__main__':
 app.run(debug=True)


import pandas as pd
import numpy as np
from flask import Flask, url_for, request, render_template, Markup
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
import pickle, json
import warnings
import requests
warnings.filterwarnings('ignore')
from fertilizer import fertilizer_dic
with open('Model.pkl', 'rb') as f:
    model = pickle.load(f)
with open('labels.pkl', 'rb') as f:
    labels = pickle.load(f)
with open('StatesSeason.json', 'r') as f:
    common_label = json.load(f)


print(model.classes_)
app = Flask(__name__,static_url_path='/static')


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


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

```
def choose():
      return render_template('Choose.html')
@app.route('/details')
def details():
   return render_template('Details.html')


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


@app.route('/predict', methods=['GET', 'POST'])
def prediction():
   temp = float(request.form['temp'])
   humidity = float(request.form['humidity'])
   ph = float(request.form['ph'])
   rainfall = float(request.form['rainfall'])
   season = str(request.form['season'])
   state = str(request.form['state'])
   nitrogen = float(request.form['nitrogen'])
   phosphorous = float(request.form['phosphorous'])
   potassium = float(request.form['potassium'])


   #Encode
   print("Enocde")
   season = common_label[season]
   state  = common_label[state]
print("Encode")
```

```
    query = np.array([temp, humidity, ph, rainfall, season, state, nitrogen, phosphorous,
potassium]).reshape(1, -1)

    print(query)

    # Model

    result = model.predict_proba(query)[0]

    # Top 5 loosers

    top_gainers = np.argsort(-result)[:5]

    top_gainers = labels.inverse_transform(top_gainers)

    # Top 5 ganiers

    top_loosers = np.argsort(result)[:5]

    top_loosers = labels.inverse_transform(top_loosers)

    # f"Top Gainers:{top_gainers}<br>Top Loosers:{top_loosers}"

    return render_template('Crop_data.html', top_gainers =
top_gainers,top_loosers=top_loosers )


@app.route('/fertilizers', methods=['GET', 'POST'])

def index():

    if request.method == 'POST':

        crop_name = str(request.form['crop'])

        N = int(request.form['nitrogen'])

        P = int(request.form['phosphorous'])

        K = int(request.form['potassium'])

        # ph = float(request.form['ph'])

        df = pd.read_csv(r"E:\Projects\Smart_Harvest\fertilizer.csv")


        nr = df[df['Crop'] == crop_name]['N'].iloc[0]

        pr = df[df['Crop'] == crop_name]['P'].iloc[0]

        kr = df[df['Crop'] == crop_name]['K'].iloc[0]
```

```
       n = nr - N
        p = pr - P
        k = kr - K
        temp = {abs(n): "N", abs(p): "P", abs(k): "K"}
        max_value = temp[max(temp.keys())]
        if max_value == "N":
            if n < 0:
                key = 'NHigh'
            else:
                key = "Nlow"
        elif max_value == "P":
            if p < 0:
                key = 'PHigh'
            else:
                key = "Plow"
        else:
            if k < 0:
                key = 'KHigh'
            else:
                key = "Klow"

        response = Markup(str(fertilizer_dic[key]))
        return render_template('Fertilizer_result.html', response = response)
        return render_template('Fertilizer_details.html')

# Weather
@app.route('/weather', methods=['GET', 'POST'])
```

```
def weather():
    if request.method == 'POST':
        print("Entered....")
        zip_code = str(request.form['zip'])
        # state = str(request.form['state'])
        api_key = "56765cc0691fe16b12cf69c42ce61c10"

        headers = {
            "apikey": "90c125c0-e455-11ee-aee3-13a3b378e2b5"}

        params = (
            ("codes", zip_code),
            ("country","in"),
        )

        response = requests.get('https://app.zipcodebase.com/api/v1/search', headers=headers,
params=params);
        response = json.loads(response.text)
        results = response['results'][zip_code]
        latitude = results[0]['latitude']
        longitude = results[0]['longitude']
 # using Latitude and Longitude
Resp=requests.get(f"https://api.openweathermap.org/data/2.5/weather?lat={latitude}&lon={l
ongitude}&appid={api_key}")
        data = json.loads(resp.text)

        # Extract temperature, humidity, longitude, and latitude
        temperature = round(data['main']['temp']-273.15, 2)
        humidity = data['main']['humidity']

        # longitude = data['coord']['lon']

        # latitude = data['coord']['lat']

        state = "".join(results[0]['state'].lower().split())

        province = results[0]['province']

        print(state)

        print(province)

        print(data['main']['temp']-273.15)

        return render_template('Details.html', temperature = str(temperature), humidity=
str(humidity), state=state)

    return render_template('Details.html', temperature = '', humidity= '', state='')


@app.route('/crop_info')

def crop_info():

    return render_template('Crop_info.html')


@app.route("/goback")

def goback():

    return render_template('Choose.html')


if __name__ == '__main__':

    app.run(debug=True)


import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier
```

```
import pickle, json
from flask import Flask, render_template, url_for, request
import warnings
warnings.filterwarnings('ignore')

with open('Model.pkl', 'rb') as f:
   model = pickle.load(f)
with open('labels.pkl', 'rb') as f:
   labels = pickle.load(f)
with open('StatesSeason.json', 'r') as f:
   common_label = json.load(f)

# i = np.array([[ 21.9989826 ,  56.31006755,   6.98571967, 136.8274312 ,
#         20.        ,  19.        ,  18.        ,  55.        ,
#         23.        ]])
# r = model.predict_proba(i)
# print(r)
# print(labels.inverse_transform([8]))
# print(common_label)

#Flask
app = Flask(__name__)
@app.route('/')
def home():
   return render_template('index.html')

@app.route('/predict', methods=['GET', 'POST'])
```

```
def prediction():

    temp = float(request.form['temp'])

    humidity = float(request.form['humidity'])

    ph = float(request.form['ph'])

    rainfall = float(request.form['rainfall'])

    season = str(request.form['season'])

    state = str(request.form['state'])

    nitrogen = float(request.form['nitrogen'])

    phosphorous = float(request.form['phosphorous'])

    potassium = float(request.form['potassium'])


    #Encode

    print("Enocde")

    season = common_label[season]

    state  = common_label[state]

    print("Encode")

    query = np.array([temp, humidity, ph, rainfall, season, state, nitrogen, phosphorous,
potassium]).reshape(1, -1)

    print(query)

    # Model

    result = model.predict_proba(query)[0]

    # Top 5 loosers

    top_gainers = np.argsort(-result)[:5]

    top_gainers = labels.inverse_transform(top_gainers)

    # Top 5 ganiers

    top_loosers = np.argsort(result)[:5]

    top_loosers = labels.inverse_transform(top_loosers)
```

```
# f"Top Gainers:{top_gainers}<br>Top Loosers:{top_loosers}"
    return render_template('crop_results.html', top_gainers =
top_gainers,top_loosers=top_loosers )


if __name__ == "__main__":
    app.run(debug=True)
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
import pickle, json
from flask import Flask, render_template, url_for, request
import warnings
warnings.filterwarnings('ignore')


with open('Model.pkl', 'rb') as f:
    model = pickle.load(f)
with open('labels.pkl', 'rb') as f:
    labels = pickle.load(f)
with open('StatesSeason.json', 'r') as f:
    common_label = json.load(f)



# i = np.array([[ 21.9989826 ,  56.31006755,   6.98571967, 136.8274312 ,
#          20.        ,  19.        ,  18.        ,  55.        ,
#          23.        ]])
# r = model.predict_proba(i)
# print(r)
# print(labels.inverse_transform([8]))
# print(common_label)

#Flask
app = Flask(__name__)
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['GET', 'POST'])
def prediction():
    temp = float(request.form['temp'])
    humidity = float(request.form['humidity'])
    ph = float(request.form['ph'])
    rainfall = float(request.form['rainfall'])
    season = str(request.form['season'])
    state = str(request.form['state'])
    nitrogen = float(request.form['nitrogen'])
    phosphorous = float(request.form['phosphorous'])
    potassium = float(request.form['potassium'])

   #Encode
    print("Enocde")
    season = common_label[season]
    state  = common_label[state]
    print("Encode")
    query = np.array([temp, humidity, ph, rainfall, season, state, nitrogen, phosphorous,
potassium]).reshape(1, -1)

    print(query)


    # Model

    result = model.predict_proba(query)[0]

    # Top 5 loosers

    top_gainers = np.argsort(-result)[:5]

    top_gainers = labels.inverse_transform(top_gainers)

    # Top 5 ganiers

    top_loosers = np.argsort(result)[:5]

    top_loosers = labels.inverse_transform(top_loosers)


    # f"Top Gainers:{top_gainers}<br>Top Loosers:{top_loosers}"

    return render_template('crop_results.html', top_gainers =
top_gainers,top_loosers=top_loosers )




if __name__ == "__main__":

    app.run(debug=True)
```

7.2 SCREENSHOT

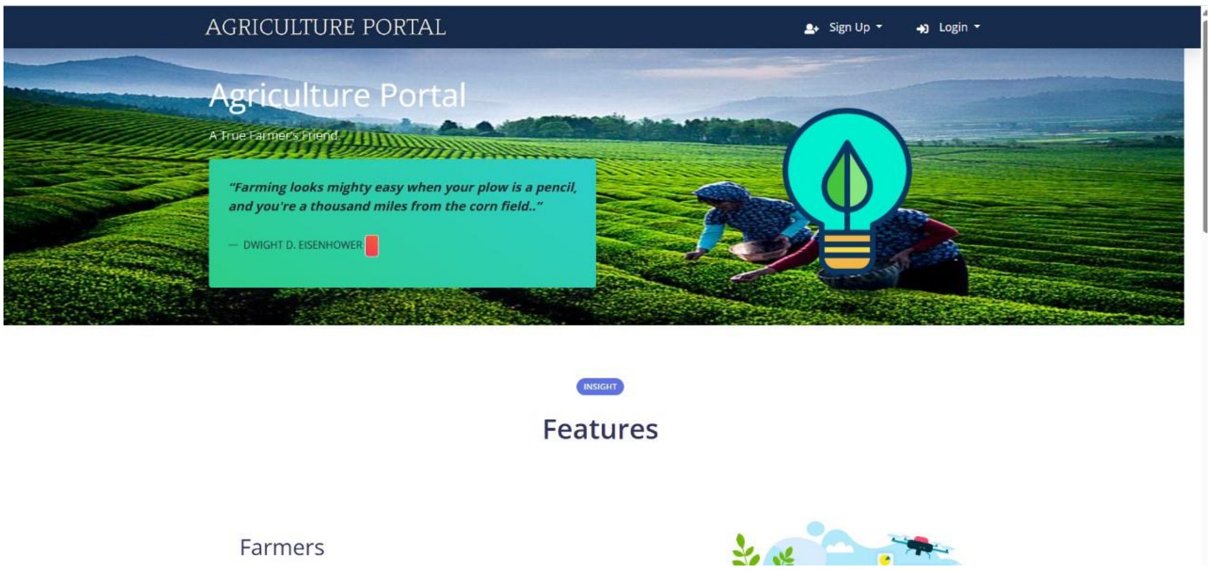
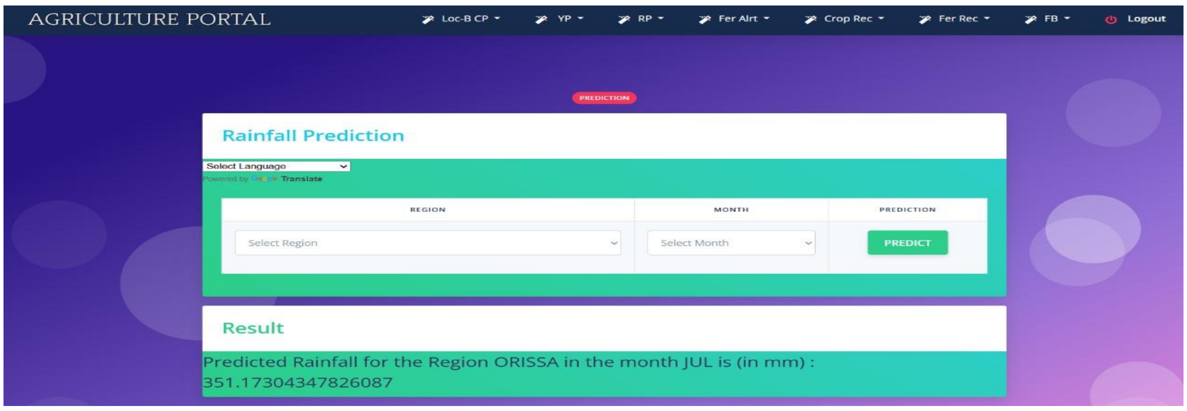
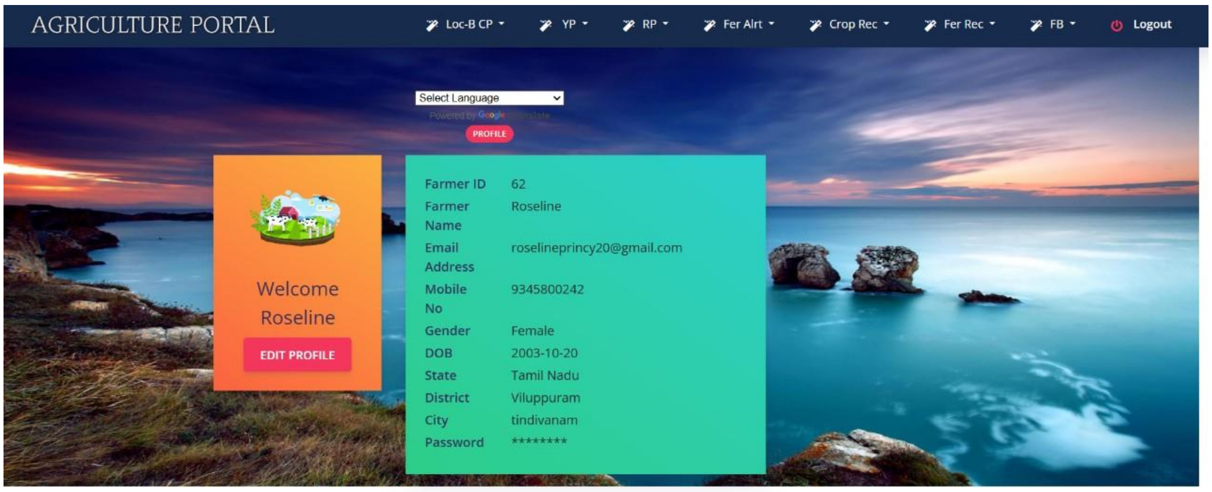


FIG7.2.1 Home Page



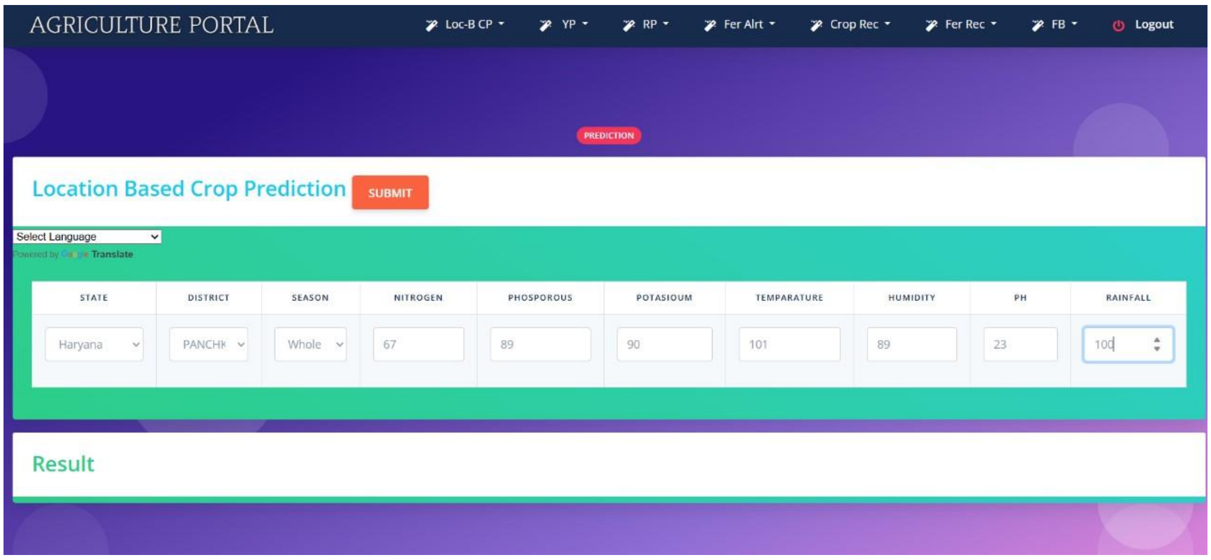
Predicted Rainfall for the Region ORISSA in the month JUL is (in mm) : 351.17304347826087

FIG 7.2.2 Prediction Page



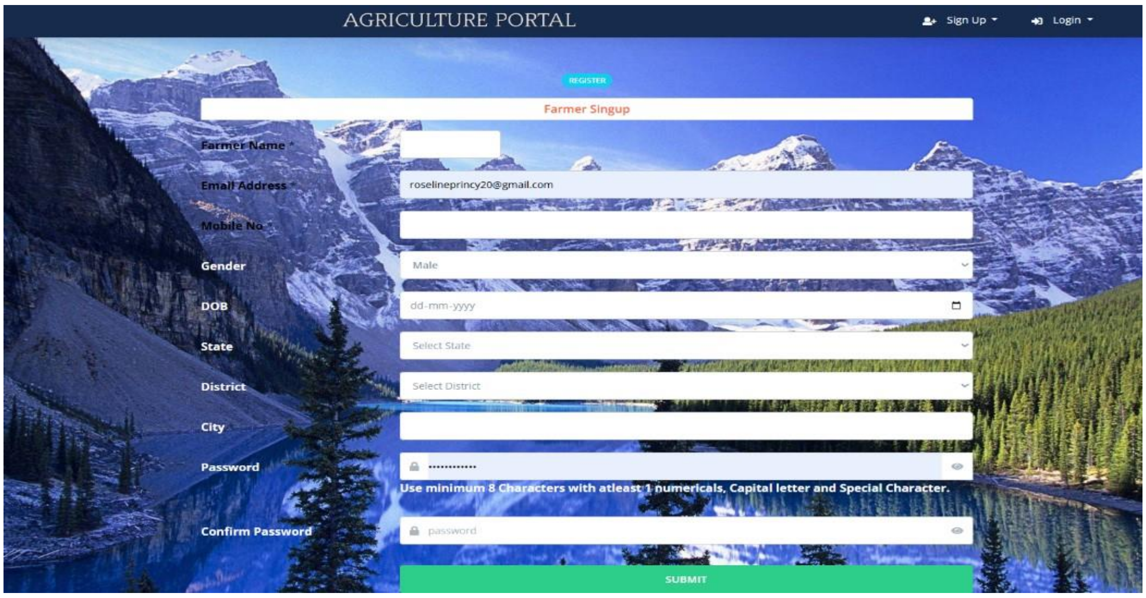
Farmer ID	62
Farmer Name	Roseline
Email Address	roselineprincy20@gmail.com
Mobile No	9345800242
Gender	Female
DOB	2003-10-20
State	Tamil Nadu
District	Viluppuram
City	tindivanam
Password	********

FIG 7.2.3 Profile Page



STATE	DISTRICT	SEASON	NITROGEN	PHOSPOROUS	POTASIOUM	TEMPARATURE	HUMIDITY	PH	RAINFALL
Haryana	PANCHK	Whole	67	89	90	101	89	23	100

FIG 7.2.4 Crop Recommendation Page


AGRICULTURE PORTAL

Sign Up ▾ Login ▾

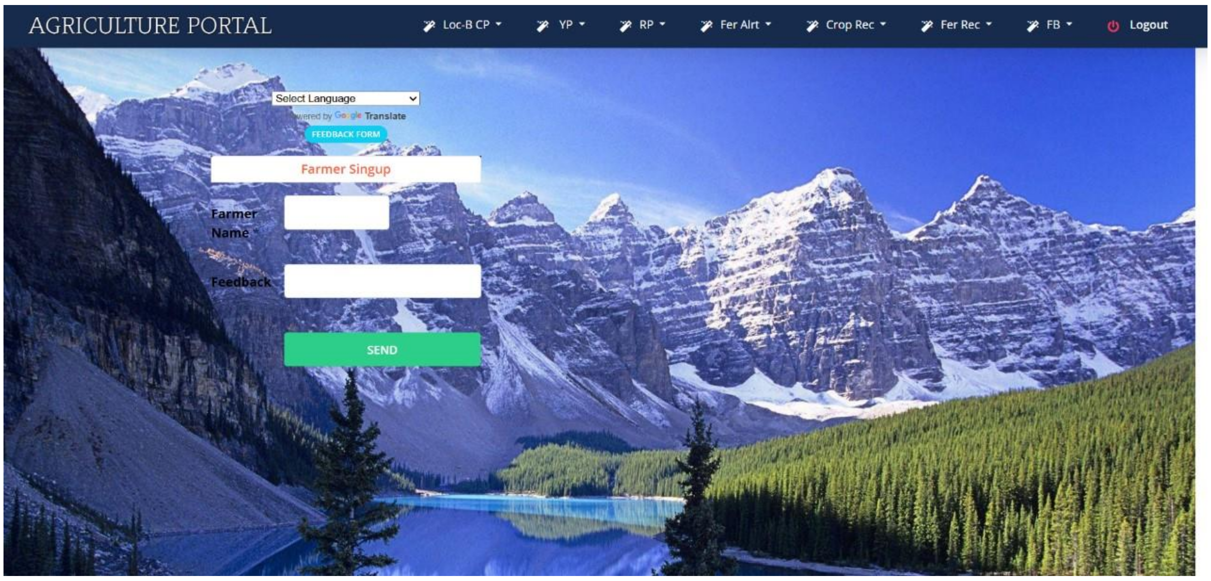
REGISTER

Farmer Singup

Farmer Name *	
Email Address *	roselineprincy20@gmail.com
Mobile No *	
Gender	Male
DOB	dd-mm-yyyy
State	Select State
District	Select District
City	
Password	••••••••••••
	Use minimum 8 Characters with atleast 1 numericals, Capital letter and Special Character.
Confirm Password	password

SUBMIT

FIG 7.2.5 Signup Page


AGRICULTURE PORTAL

Loc-B CP ▾ YP ▾ RP ▾ Fer Alrt ▾ Crop Rec ▾ Fer Rec ▾ FB ▾ Logout

Select Language

Powered by Google Translate

FEEDBACK FORM

Farmer Singup

Farmer Name	
Feedback	

SEND

FIG 7.2.6 Feedback Page

CHAPTER 8

SYSTEM TESTING

8.1 Unit Testing

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.

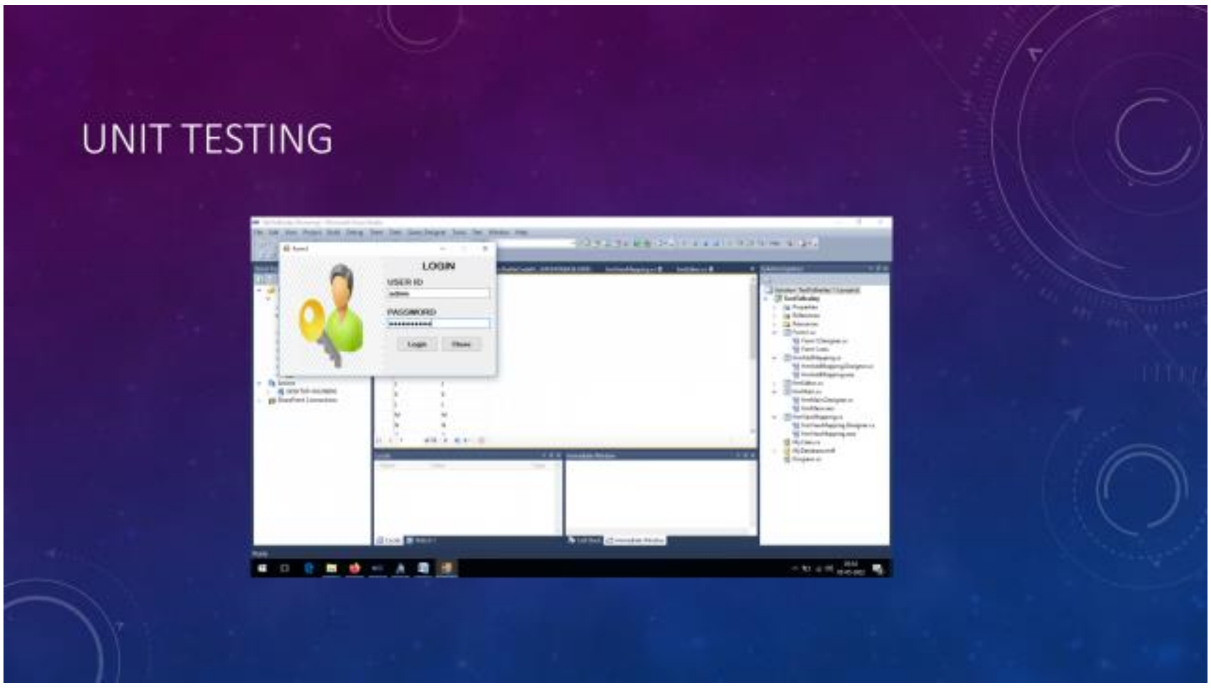


FIG8.1 UNIT TESTING

8.2 Black box Testing

A simple login screen of software or a web application will be tested for seamless user login. The login screen has two fields, username and password as an input and the output will be to enable access to the system. A black box testing will not consider the specifications of the code, and it will test the valid username and password to login to the right account.

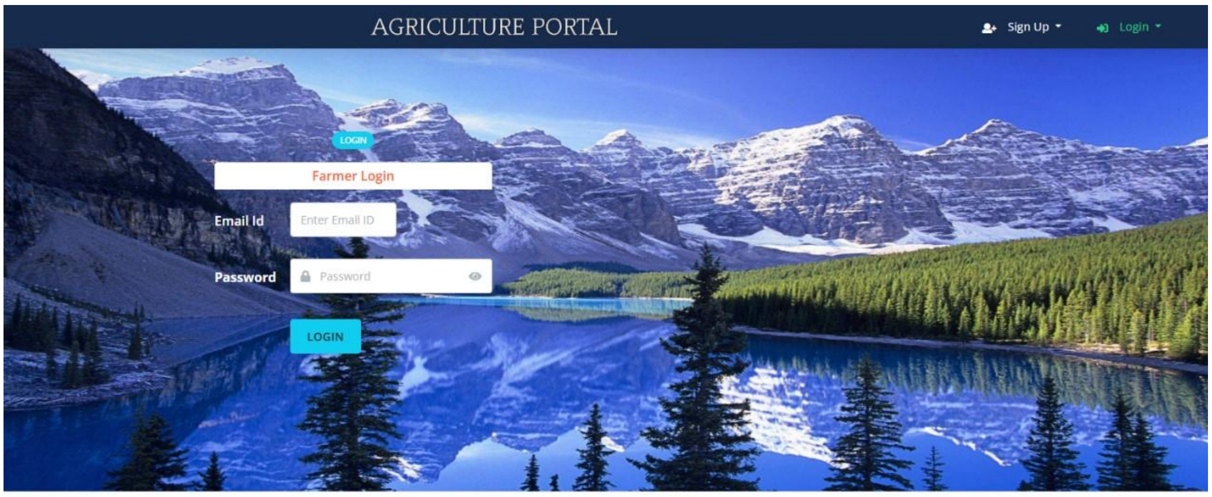


FIG8.2 BLACK BOX TESTING

8.3 INTERGRATION TESTING:

Integration testing is nothing that the system integration has been tested .

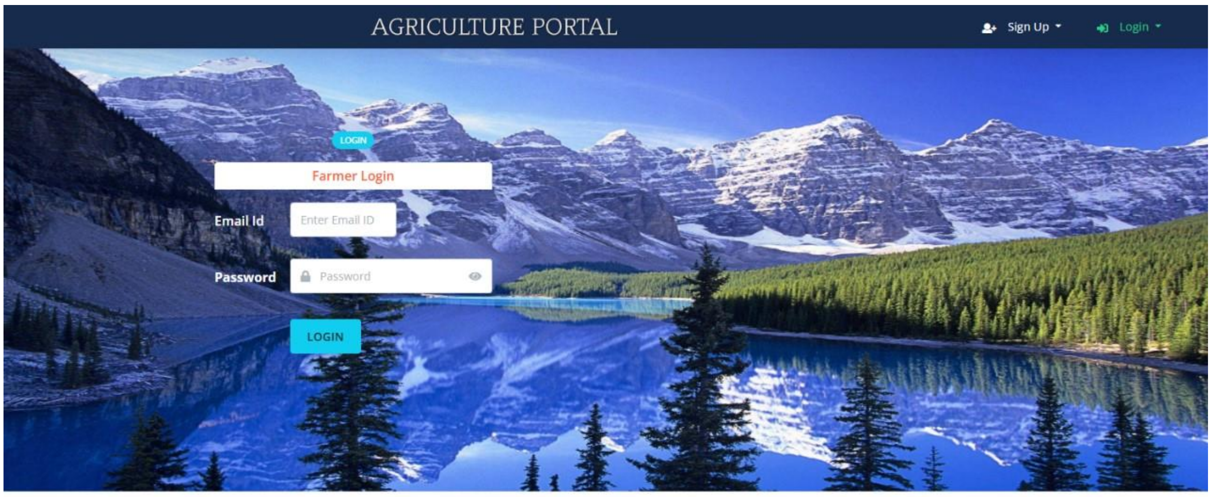


FIG8.3 INTEGRATION TESTING

CHAPTER 9

CONCLUSION

The Smart Agri project successfully demonstrates the potential of a software-based, location-aware decision support system for sustainable and precision farming. By eliminating the dependency on expensive IoT devices and focusing purely on accessible digital data sources such as GIS, weather, and crop databases, the system provides practical and personalized solutions to farmers—especially those in resource-limited areas.

Smart Agri offers intelligent modules for location-based crop selection, rainfall prediction, yield estimation, crop and fertilizer recommendations, and user feedback integration. These features collectively empower farmers to make informed decisions that optimize productivity, conserve resources, and reduce environmental impact.

This project not only addresses key challenges faced by traditional farming but also promotes the adoption of digital agriculture in a cost-effective and scalable manner. With future enhancements, such as multilingual support and integration with government or agricultural advisory systems, Smart Agri has the potential to make a significant contribution to the future of smart and sustainable farming.

CHAPTER 10

FUTURE ENHANCEMENT

The Smart Agri project has considerable room for further development to improve its capabilities and broaden its impact. One of the key enhancements would be the integration of IoT sensors and real-time data collection, allowing the system to provide more accurate insights on factors like soil moisture, temperature, and crop health. This would enable the platform to offer more granular, on-the-ground data for better decision-making.

CHAPTER 11

REFERENCES

1. Kumar, P., & Sharma, A. (2021). Precision agriculture: IoT-based solutions for smart farming. *Journal of Agricultural Engineering*, 58(2), 120-132. https://doi.org/10.1016/j.jageng.2021.02.003
2. Hussain, M., & Javed, R. (2022). Machine learning techniques for crop prediction and yield estimation: A review. *Agricultural Systems*, 191, 103164. https://doi.org/10.1016/j.agsy.2022.103164
3. Singh, R., & Singh, A. (2021). Data-driven crop recommendation systems for sustainable agriculture: A review. *Environmental Impact Assessment Review*, 89, 106573. https://doi.org/10.1016/j.eiar.2021.106573
4. Basu, A., & Das, S. (2022). Application of GIS and remote sensing technologies for precision agriculture: A review of recent advancements. *Precision Agriculture*, 23(1), 215-232. https://doi.org/10.1007/s11119-021-09825-5
5. Meena, S. K., & Saravanan, A. (2021). Role of artificial intelligence and machine learning in precision agriculture. *Computers and Electronics in Agriculture*, 181, 105970. https://doi.org/10.1016/j.compag.2021.105970
6. Jha, R., & Soni, S. (2023). IoT-based monitoring systems for sustainable farming: An overview and future prospects. *Agricultural Water Management*, 268, 107595. https://doi.org/10.1016/j.agwat.2023.107595
7. Agarwal, R., & Kapoor, R. (2021). A review of machine learning techniques in agricultural crop prediction. *Artificial Intelligence in Agriculture*, 4, 37-50. https://doi.org/10.1016/j.aiia.2021.01.001
8. Liu, Y., & Zhang, L. (2022). Rainfall prediction models for enhancing agricultural decision-making: A comparative study. *Journal of Hydrology*, 597, 126246. https://doi.org/10.1016/j.jhydrol.2021.126246
9. Khan, M. A., & Islam, S. (2021). Smart farming and its future in precision agriculture. *Farming Systems Research Journal*, 15(3), 48-63. https://doi.org/10.1007/s11828-021-09213-9
10. Patel, S., & Patel, M. (2022). Predictive analytics and big data for crop yield estimation and decision support in agriculture. *Journal of Agricultural and Food Chemistry*, 70(4), 1210-1221. https://doi.org/10.1021/acs.jafc.1c07215