RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CB23332 **SOFTWARE ENGINEERING LAB**
Laboratory Record Note Book

Name : . . . .

Year / Branch / Section : . . . .

Register No. : . . . .

Semester : . . . .

Academic Year : . . . .

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

RAJALAKSHMI NAGAR, THANDALAM – 602-105

BONAFIDE CERTIFICATE

NAME:______________________**REGISTER NO.:** _______________

ACADEMIC YEAR: 2024-25 **SEMESTER:** III **BRANCH:**_________B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the

CB23332-SOFTWARE ENGINEERING - Laboratory during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on ____________

Internal Examiner

External Examiner

INDEX

S.No.	Name of the Experiment	Expt. Date	Faculty Sign
1.	Preparing Problem Statement		
2.	Software Requirement Specification (SRS)		
3.	Entity-Relational Diagram		
4.	Data Flow Diagram		
5.	Use Case Diagram		
6.	Activity Diagram		
7.	State Chart Diagram		
8.	Sequence Diagram		
9.	Collaboration Diagramt		
10.	Class Diagram		

EX NO:1	WRITE THE COMPLETE PROBLEM STATEMENT
DATE:	

AIM:

To prepare a **PROBLEM STATEMENT** for the project *Smart Agriculture System for Irrigation*.

ALGORITHM:

1. The problem statement is the initial starting point for a project.
2. A problem statement describes what needs to be done without describing how.
3. It is typically a one-to-three-page document that all stakeholders agree on, describing the project goals at a high level.
4. The problem statement is intended for a broad audience and should be written in non-technical terms.
5. It helps technical and non-technical personnel communicate effectively by providing a clear description of the problem.
6. It does not describe the solution to the problem.

INPUT:

1. The input to requirement engineering is the problem statement prepared by the customer.
2. It may provide an overview of the existing system along with broad expectations from the new system.
3. The first phase of requirements engineering begins with **requirements elicitation**, i.e., gathering information about requirements.
4. Requirements are identified with the help of the customer and existing system processes.

PROBLEM:

The current irrigation practices in agriculture rely heavily on traditional methods, which are inefficient and often lead to over-irrigation or under-irrigation. These issues result in excessive water wastage, increased costs, and suboptimal crop yields. Additionally, farmers face difficulties in adapting to changing climatic conditions due to the lack of real-time monitoring and automated control systems. As water scarcity becomes a growing global concern, there is a pressing need for a sustainable, smart irrigation system that utilizes technology to optimize water usage and enhance agricultural productivity.

BACKGROUND:

Many farmers depend on manual or semi-automated irrigation systems that fail to adjust to real-time environmental conditions, such as soil moisture, humidity, and weather forecasts. These outdated systems not only waste water but also limit the ability of farmers to respond effectively to changing agricultural needs. Furthermore, traditional systems often require substantial manual labor and constant supervision, making them less practical for large-scale farming operations. The lack of integration with modern technology also means missed opportunities to leverage data insights for better decision-making, thereby hindering sustainable farming practices.

RELEVANCE:

Efficient irrigation management is critical for sustainable agriculture, particularly in regions prone to drought or water scarcity. A poorly managed system results in resource wastage, reduced crop productivity, and financial losses for farmers. By modernizing irrigation practices, farmers can enhance their yield, reduce water usage, and contribute to environmental conservation. The introduction of smart irrigation systems has the potential to revolutionize agriculture by promoting precision farming, enabling remote management, and supporting broader sustainability goals.

OBJECTIVES:

The primary objective of this project is to develop a **Smart Agriculture System for Irrigation** that leverages technology to optimize water usage, improve crop health, and promote sustainable farming practices. Specific objectives include:

1. **Analyzing Current Practices**: Conducting an assessment of traditional irrigation methods to identify inefficiencies and areas for improvement.
2. **Deploying IoT Sensors**: Installing sensors to monitor real-time soil moisture, temperature, and humidity data for informed decision-making.
3. **Integrating Weather Data**: Incorporating weather forecasts into irrigation scheduling to avoid unnecessary water usage.
4. **Automating Irrigation**: Developing an automated system to control water flow based on sensor data and environmental conditions.
5. **Real-Time Monitoring and Alerts**: Implementing a system to notify farmers of abnormal conditions, such as low soil moisture or extreme weather, to enable timely action.
6. **Remote Control**: Providing farmers with mobile or web-based access to monitor and control irrigation systems remotely.
7. **Data Analytics and Reporting**: Offering insights into water usage patterns, crop yield, and overall system efficiency to support long-term planning.
8. **Improving Water Efficiency**: Designing algorithms to optimize water allocation, ensuring no resource is wasted.
9. **Scalability**: Ensuring the system can accommodate varying farm sizes and future agricultural needs.
10. **User Training**: Training farmers and agricultural staff to effectively use the system and leverage its full potential.

Result:

EX NO:2	WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT
DATE:	

AIM:

To conduct **requirement analysis** and develop a **Software Requirement Specification Sheet (SRS)** for the project *Smart Agriculture System for Irrigation*.

ALGORITHM:

An SRS shall address the following:

1. **Functionality**: What is the software supposed to do?
2. **External Interfaces**: How does the software interact with users, hardware, and other systems?
3. **Performance**: Speed, availability, and response time for system operations.
4. **Attributes**: Considerations for portability, correctness, maintainability, security, etc.
5. **Design Constraints**: Standards, implementation language, database policies, resource limits, and operating environments.

1. Introduction

1.1 Purpose

This document outlines the requirements for a *Smart Agriculture System for Irrigation*. The system aims to automate irrigation based on real-time environmental data, reduce water usage, and enhance crop productivity.

1.2 Document Conventions

- IoT: Internet of Things
- UI: User Interface
- API: Application Programming Interface
- KPI: Key Performance Indicator
- RBAC: Role-Based Access Control

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- **Developers**: To understand technical requirements for implementation.
- **Project Managers**: To monitor timelines and deliverables.
- **Stakeholders**: To verify alignment with operational goals.

Quality Assurance: To validate system functionality and performance

1.4 Project Scope

The *Smart Agriculture System for Irrigation* will provide real-time monitoring of environmental conditions, automated irrigation control, and user notifications. The system will optimize water usage, reduce costs, and support sustainable farming practices.

1.5 References

- Sustainable Water Management Guidelines.
- IoT Sensor Deployment Manuals.
- Agricultural Best Practices for Crop Water Needs.

2. Overall Description

2.1 Product Perspective

The system will function as a web-based and IoT-enabled application. It integrates with soil moisture sensors, weather APIs, and irrigation controllers to automate water management.

2.2 Product Features

Key features include:

- Automated irrigation control based on predefined thresholds.
- Weather integration to adjust irrigation schedules dynamically.
- Remote access through web and mobile interfaces.
- Notifications for abnormal conditions or maintenance alerts.
- Analytics for water usage and crop yield improvement.

2.3 User Classes and Characteristics

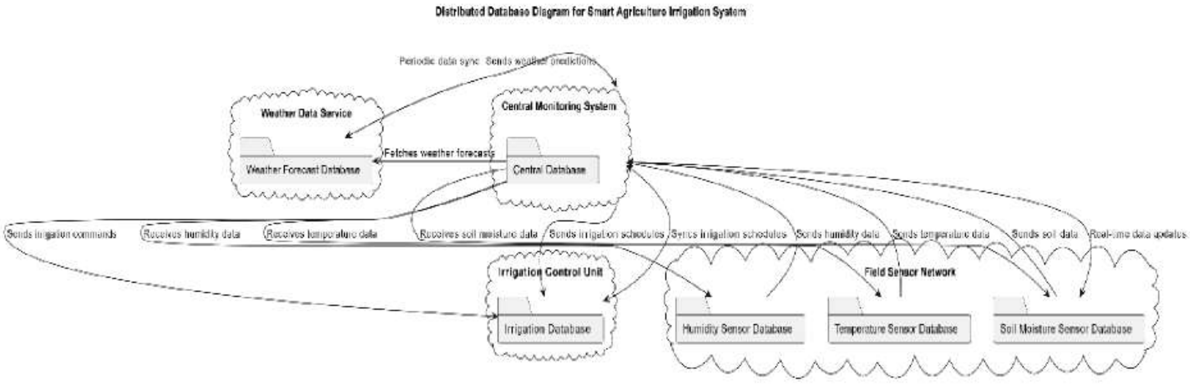
- **Farmers**: Access real-time data, manage irrigation schedules, and receive alerts.
- **Technicians**: Maintain hardware and ensure system functionality.
- **Administrators**: Manage system configurations and access controls.

2.4 Operating Environment

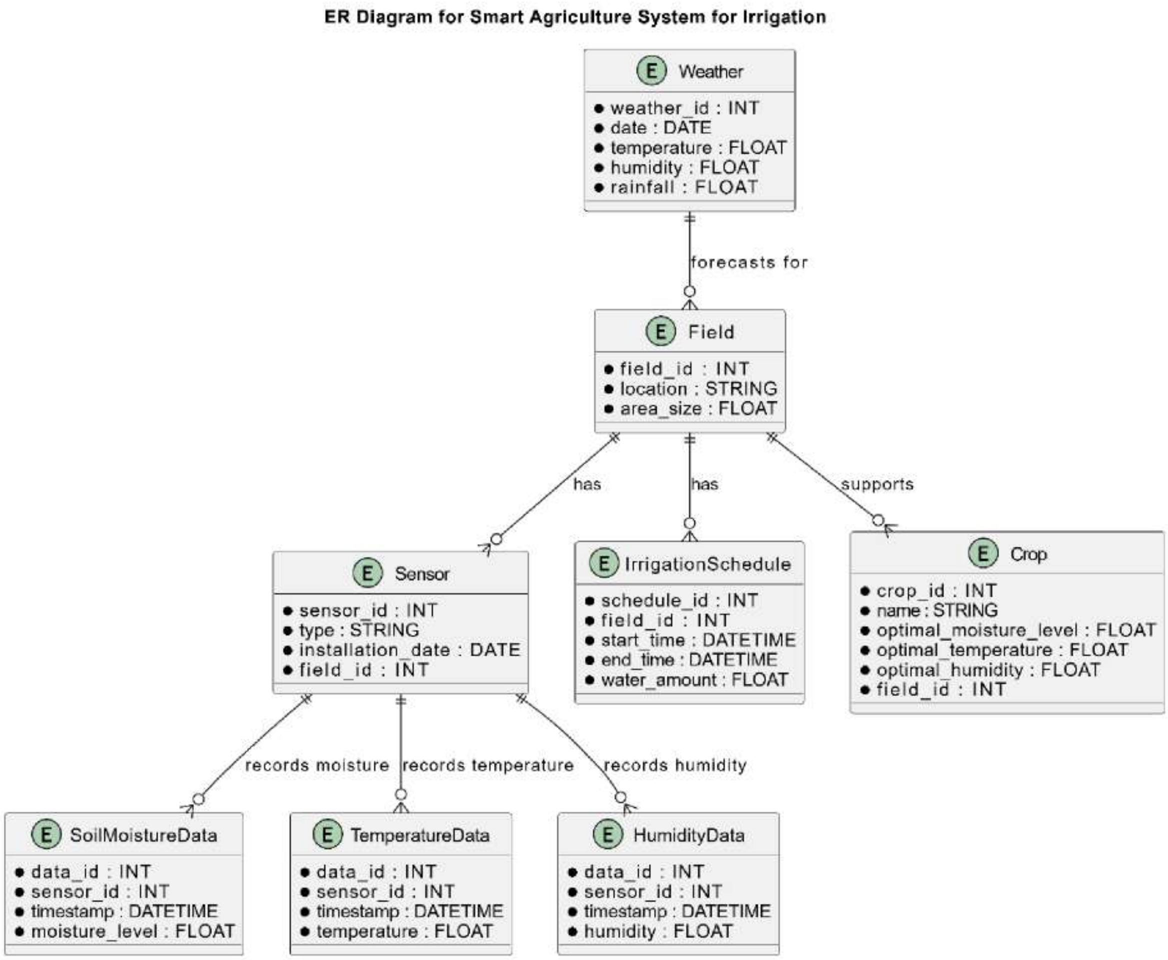
The system will operate in the following environment:

- IoT-enabled sensors for data collection.
- Distributed database for storing environmental and irrigation data.
- Cloud-based architecture accessible via modern web browsers and mobile platforms.

Distributed Diagram:



ER Diagram:



2.5 Design and Implementation Constraints

- Integration with IoT sensors for soil and weather monitoring.
- Secure APIs for data exchange and irrigation control.
- Scalability to support multiple farms or large agricultural areas.

2.6 Assumptions and Dependencies

- IoT sensors and devices are deployed across the farm.
- Reliable internet connectivity is available for real-time data updates.
- Farmers and users have access to smartphones or PCs for remote management.

3. Specific Requirements

Description and Priority

The system prioritizes optimizing irrigation efficiency, minimizing water waste, and supporting sustainable farming.

Stimulus/Response Sequence

- **Soil moisture drops below a threshold**: Irrigation is automatically activated.
- **Weather forecast indicates rain**: Irrigation is paused.
- **Abnormal soil conditions detected**: Notifications are sent to farmers.

Functional Requirements

1. **Real-Time Monitoring**: IoT sensors collect data on soil moisture, temperature, and humidity.
2. **Automated Irrigation Control**: Activate or deactivate irrigation based on sensor data and weather forecasts.
3. **Remote Access**: Farmers can monitor and control irrigation schedules through a mobile or web app.
4. **Notifications**: Alerts for abnormal conditions, such as low soil moisture or extreme weather, are sent in real time.

4. External Interface Requirements

4.1 User Interfaces

- **Frontend**: Python with Streamlit for web-based dashboards.
- **Backend**: SQL database for storing sensor and irrigation data.

4.2 Hardware Interfaces

- **IoT Devices**: Soil moisture sensors, temperature sensors, and irrigation controllers.
- **Operating System**: Windows or Linux for the server.

4.3 Software Interfaces

- Programming Language: Python for its flexibility and rich library support.
- Database: MySQL for storing environmental and irrigation data.
- Weather API: Integration with weather services for forecast data.

4.4 Communication Interfaces

- Web application accessible via modern browsers (Chrome, Firefox, Edge).
- Secure HTTPS protocol for data exchange.

5. Additional Requirements

5.1 Performance Requirements

- ER Diagram: Entities include farms, sensors, crops, and irrigation schedules.
- Normalization: Ensures efficient data management and minimal redundancy.
- System response time: Actions like irrigation activation must occur within 2 seconds of data threshold breach.

5.2 Safety Requirements

- Data backups ensure recovery in case of failures.
- The system detects hardware malfunctions and notifies technicians.

5.3 Security Requirements

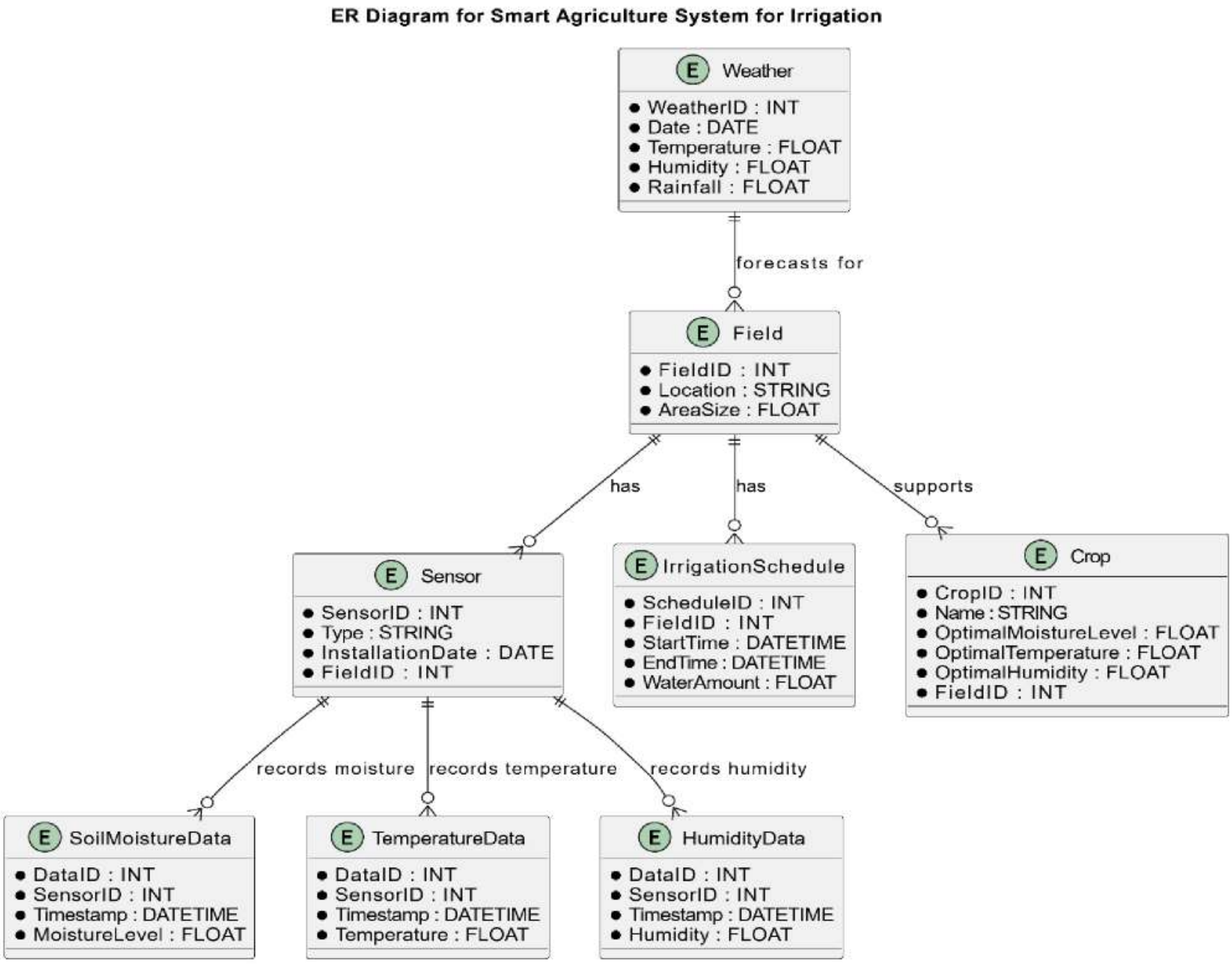
- Role-based access control (RBAC) for farmers, administrators, and technicians.
- Encryption for sensitive data during transmission and storage.

5.4 Software Quality Attributes

- Availability: The system must have 99.9% uptime.
- Correctness: Accurate data collection and irrigation decisions.
- Maintainability: Easy to update and fix bugs.
- Usability: User-friendly interfaces suitable for non-technical users.

Result:

ER Diagram:


ER Diagram for Smart Agriculture System for Irrigation
Weather
WeatherID : INT
Date : DATE
Temperature : FLOAT
Humidity : FLOAT
Rainfall : FLOAT
forecasts for
Field
FieldID : INT
Location : STRING
AreaSize : FLOAT
has
has
supports
Sensor
SensorID : INT
Type : STRING
InstallationDate : DATE
FieldID : INT
IrrigationSchedule
ScheduleID : INT
FieldID : INT
StartTime : DATETIME
EndTime : DATETIME
WaterAmount : FLOAT
Crop
CropID : INT
Name : STRING
OptimalMoistureLevel : FLOAT
OptimalTemperature : FLOAT
OptimalHumidity : FLOAT
FieldID : INT
records moisture
records temperature
records humidity
SoilMoistureData
DataID : INT
SensorID : INT
Timestamp : DATETIME
MoistureLevel : FLOAT
TemperatureData
DataID : INT
SensorID : INT
Timestamp : DATETIME
Temperature : FLOAT
HumidityData
DataID : INT
SensorID : INT
Timestamp : DATETIME
Humidity : FLOAT

EX NO:3	DRAW THE ENTITY RELATIONSHIP DIAGRAM
DATE:	

AIM:

To Draw the Entity Relationship Diagram for smart agriculture system for irrigation

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

Entities

Entity Relationship Matrix

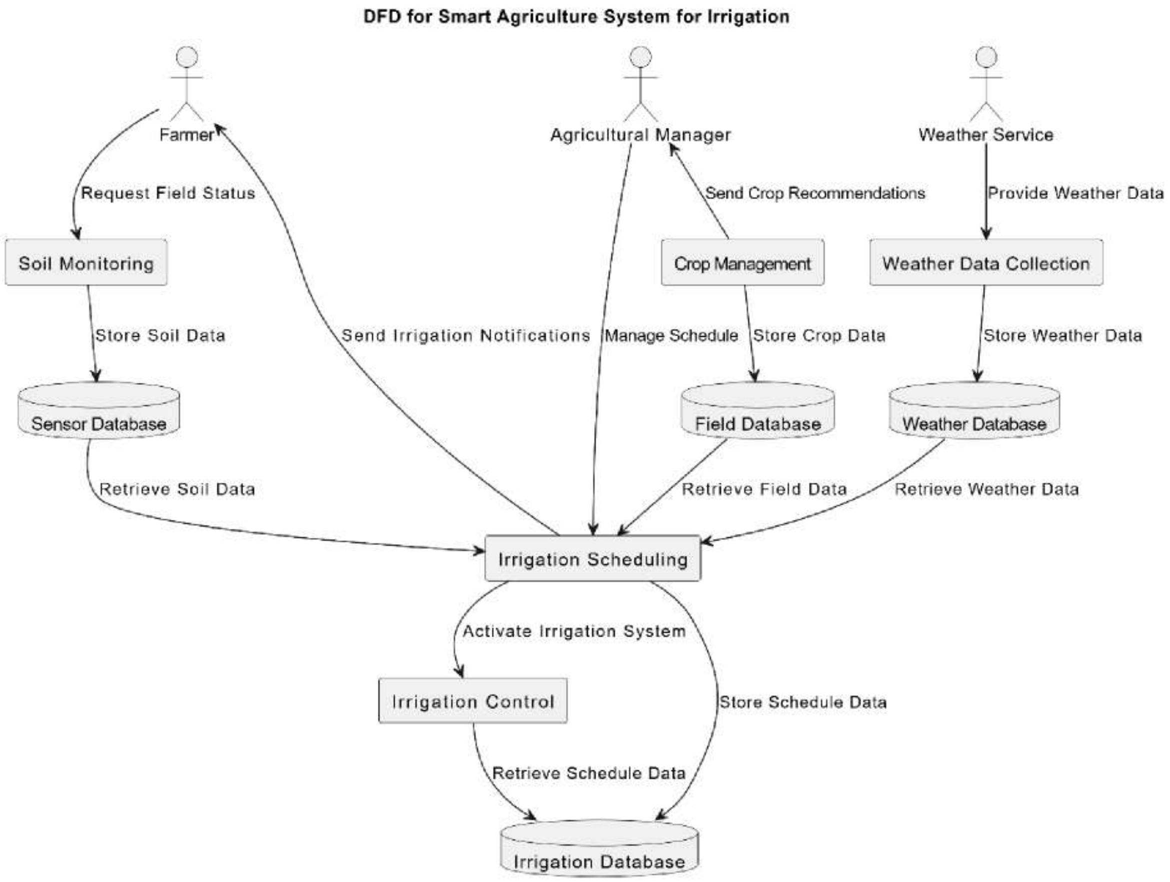
Primary Keys

Attributes

Mapping of Attributes with Entities

Result:

DFD Diagram:



DFD for Smart Agriculture System for Irrigation
Farmer
Agricultural Manager
Weather Service
Request Field Status
Send Crop Recommendations
Provide Weather Data
Soil Monitoring
Crop Management
Weather Data Collection
Store Soil Data
Send Irrigation Notifications
Manage Schedule
Store Crop Data
Store Weather Data
Sensor Database
Field Database
Weather Database
Retrieve Soil Data
Retrieve Field Data
Retrieve Weather Data
Irrigation Scheduling
Activate Irrigation System
Irrigation Control
Store Schedule Data
Retrieve Schedule Data
Irrigation Database

EX NO:4	DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1
DATE:	

AIM:

To Draw the Data Flow Diagram for smart agriculture system for irrigation and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

INPUT:

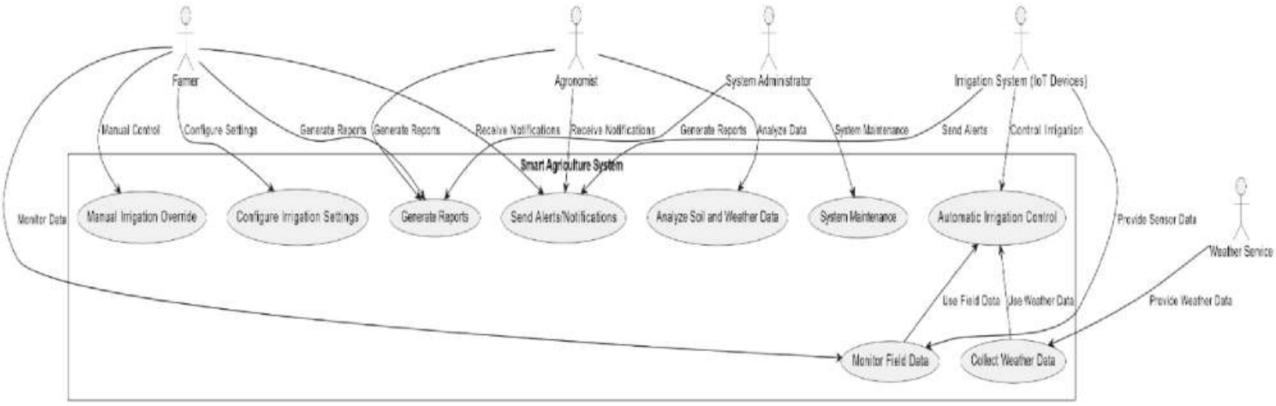
Processes

Datastores

External Entities

Result:

Use Case diagram:



EX NO:5	DRAW USE CASE DIAGRAM
DATE:	

AIM:

To Draw the Use Case Diagram for smart agriculture system for irrigation

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

INPUTS:

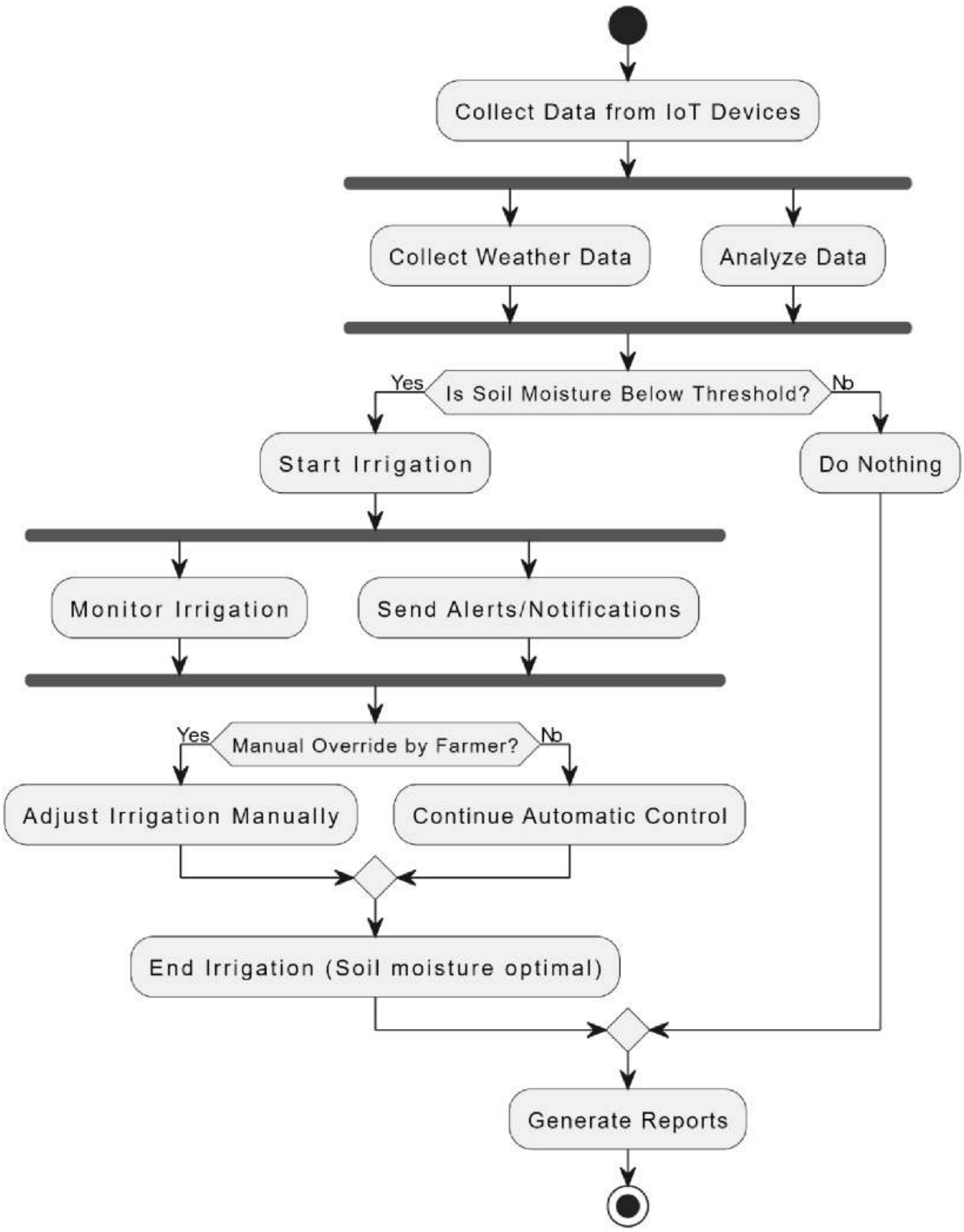
Actors

Use Cases

Relations

Result:

Activity Diagram:



EX NO:6	DRAW ACTIVITY DIAGRAM OF ALL USE CASES.
DATE:	

AIM:

To Draw the activity Diagram for smart agriculture system for irrigation

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

Decision Points

Guards

Parallel Activities

Conditions

Result:

State Chart Diagram:

EX NO:7	DRAW STATE CHART DIAGRAM OF ALL USE CASES.
DATE:	

AIM:

To Draw the State Chart Diagram for smart agriculture system for irrigation

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

INPUTS:

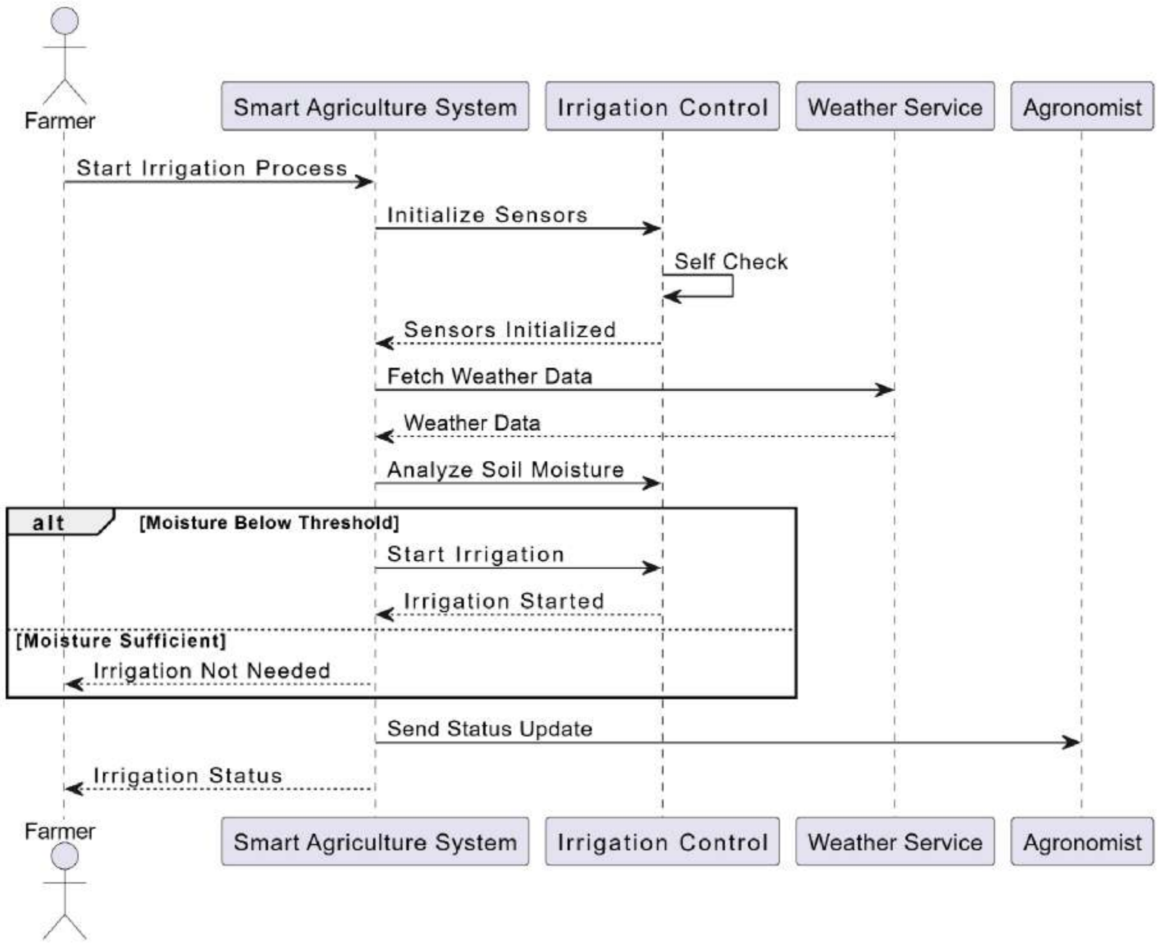
Objects

States

Events

Result:

Sequence Diagram:



EX NO:8	DRAW SEQUENCE DIAGRAM OF ALL USE CASES.
DATE:	

AIM: To Draw the Sequence Diagram for smart agriculture system

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

Objects taking part in the interaction.

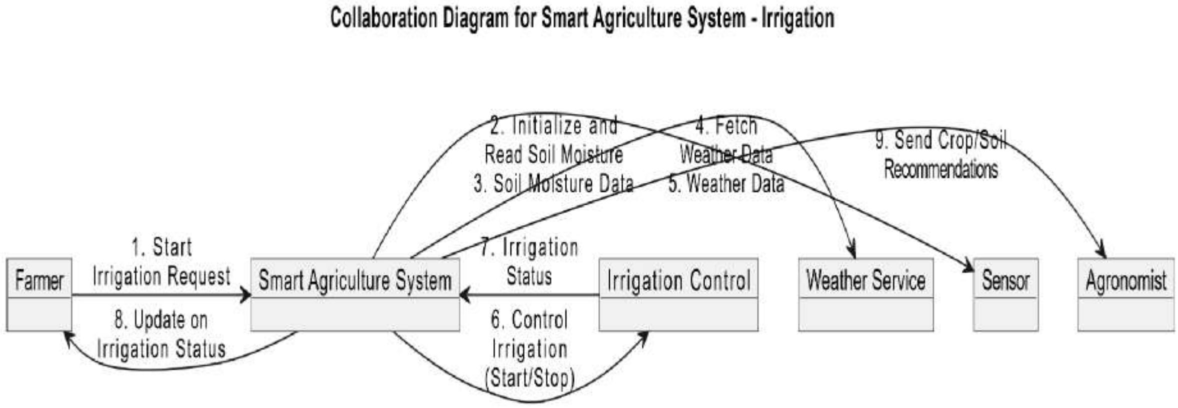
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

Collaboration Diagram:



EX NO:9	DRAW COLLABORATION DIAGRAM OF ALL USE CASES
DATE:	

AIM:

To Draw the Collaboration Diagram for smart agriculture system for irrigation

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

Objects taking part in the interaction.

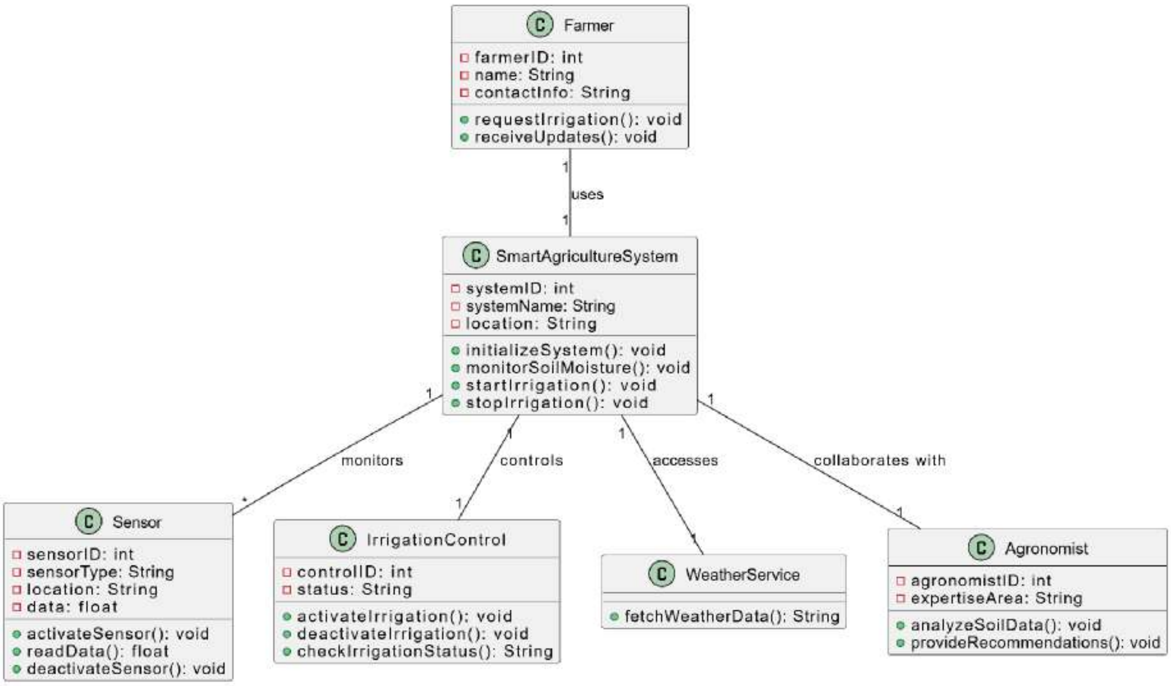
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

Class Diagram:



EX NO:10	ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.
DATE:	

AIM:

To Draw the Class Diagram for any project

ALGORITHM:

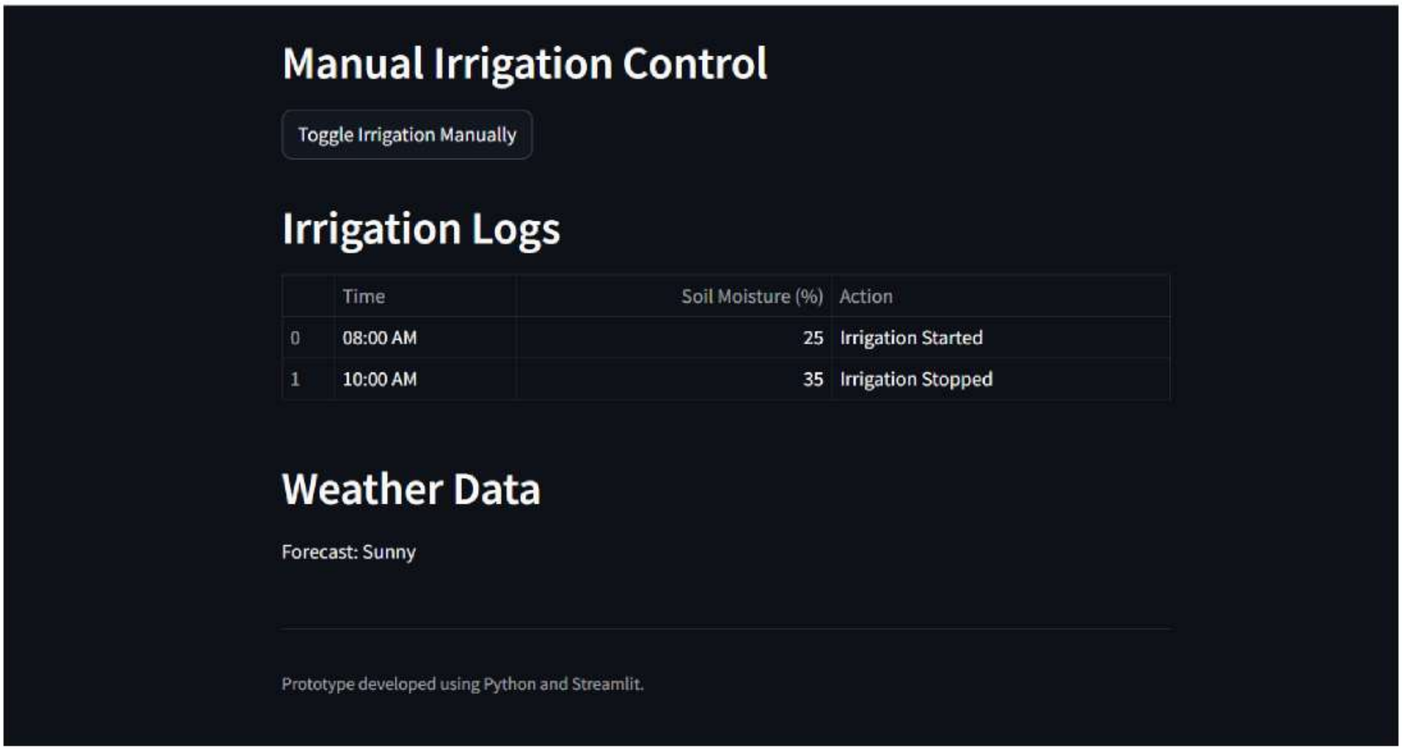
1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:


Sensor and Weather Data
Soil Moisture Level (%)
33
0
100
Temperature (°C)
34
-10
50
Humidity (%)
68
0
100
Weather Forecast
Sunny
Smart Agriculture System for Irrigation
Real-Time Sensor Data
Parameter
Value
0 Soil Moisture (%) 33
1 Temperature (°C) 34
2 Humidity (%) 68
Irrigation Decision
Irrigation Deactivated: Soil moisture is sufficient.
Current Irrigation Status: OFF


Manual Irrigation Control
Toggle Irrigation Manually
Irrigation Logs
Time
Soil Moisture (%)
Action
0 08:00 AM 25 Irrigation Started
1 10:00 AM 35 Irrigation Stopped
Weather Data
Forecast: Sunny
Prototype developed using Python and Streamlit.

EX NO:11	MINI PROJET-SMART AGRICULTURE SYSTEM FOR IRRIGATION
DATE:	

Aim:

To develop a **Smart Agriculture System for Irrigation** using **Streamlit** and **MySQL** that allows farmers and agricultural staff to efficiently monitor environmental conditions, automate irrigation schedules, track water usage, and notify users via email about critical alerts or maintenance. The focus is to optimize irrigation processes, enhance crop productivity, and ensure a sustainable, user-friendly, and reliable solution for managing irrigation in agricultural fields.

Algorithm:

1. **Start**: Initialize the Streamlit app and connect to the MySQL database.
2. **Input Data**: Gather soil moisture, temperature, humidity, and weather data.
3. **Check Conditions**:

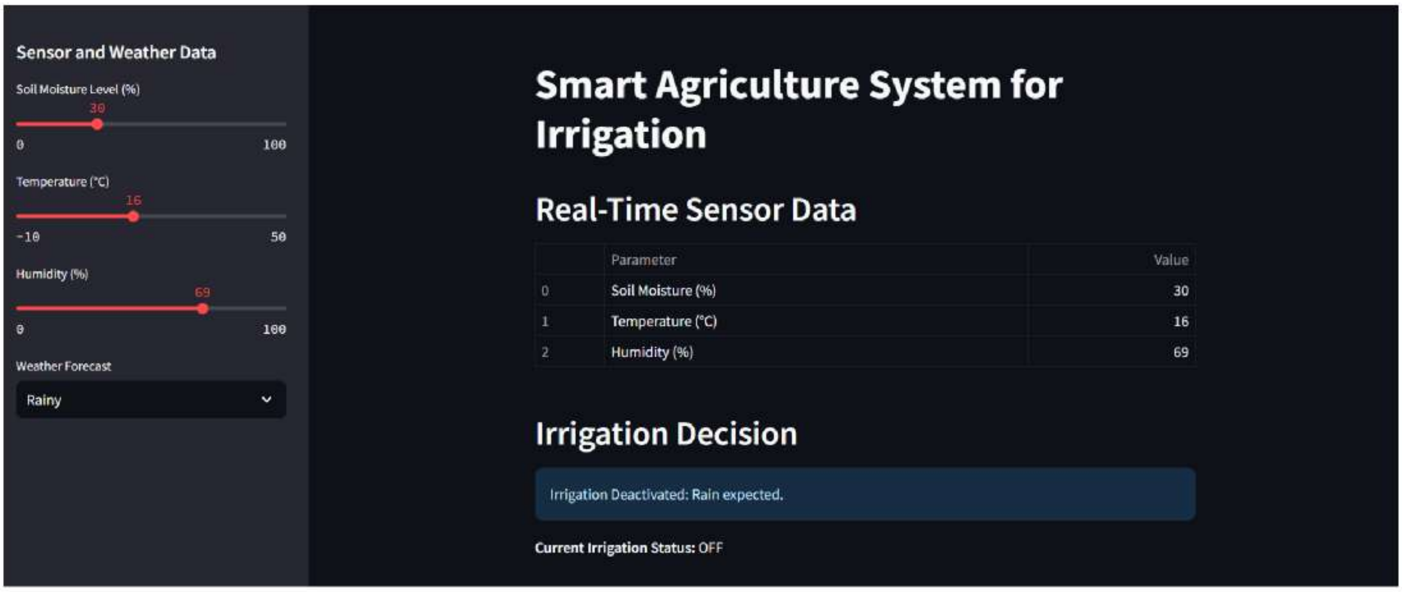
- If soil moisture is low, activate irrigation.
- If adequate or rain is forecasted, deactivate irrigation.

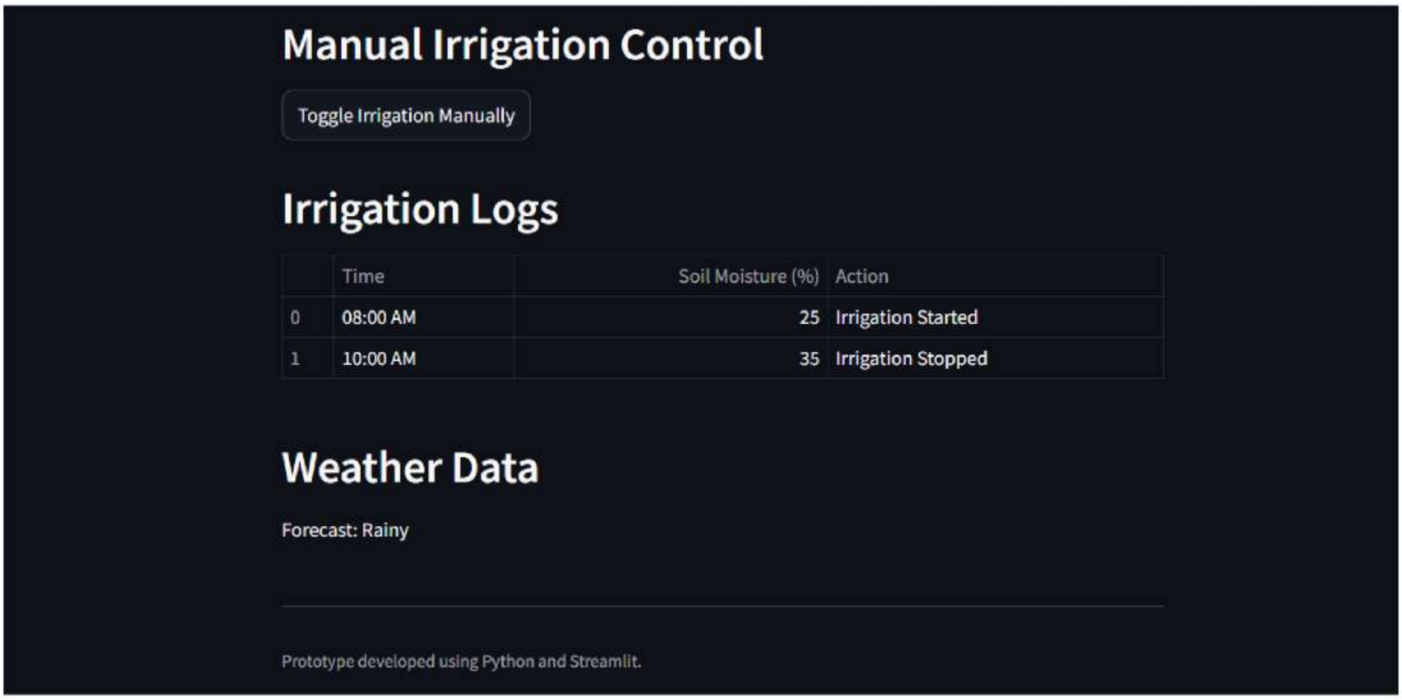
4. **Manual Override**: Allow users to control irrigation manually via the app.
5. **Send Alerts**: Notify users via email about abnormal conditions or maintenance.
6. **Log Data**: Store irrigation actions and environmental data in the database.

7.End: Ensure the system runs continuously for real-time monitoring.

Program:

```
import streamlit as st

import pandas as pd

import random


# Title

st.title("Smart Agriculture System for Irrigation")


# Sidebar for Input

st.sidebar.header("Sensor and Weather Data")

soil_moisture = st.sidebar.slider("Soil Moisture Level (%)", 0, 100, random.randint(20, 50))

temperature = st.sidebar.slider("Temperature (°C)", -10, 50, random.randint(15, 35))

humidity = st.sidebar.slider("Humidity (%)", 0, 100, random.randint(30, 70))

weather_forecast = st.sidebar.selectbox("Weather Forecast", ["Sunny", "Rainy", "Cloudy", "Stormy"])
```


Sensor and Weather Data
Soil Moisture Level (%)
30
0
100
Temperature (°C)
16
-10
50
Humidity (%)
69
0
100
Weather Forecast
Rainy
Smart Agriculture System for Irrigation
Real-Time Sensor Data
Parameter
Value
0 Soil Moisture (%) 30
1 Temperature (°C) 16
2 Humidity (%) 69
Irrigation Decision
Irrigation Deactivated: Rain expected.
Current Irrigation Status: OFF


Manual Irrigation Control
Toggle Irrigation Manually
Irrigation Logs
Time
Soil Moisture (%)
Action
0 08:00 AM 25 Irrigation Started
1 10:00 AM 35 Irrigation Stopped
Weather Data
Forecast: Rainy
Prototype developed using Python and Streamlit.

```
# Simulated Sensor Data
st.header("Real-Time Sensor Data")
sensor_data = pd.DataFrame({
    "Parameter": ["Soil Moisture (%)", "Temperature (°C)", "Humidity (%)"],
    "Value": [soil_moisture, temperature, humidity],
})
st.table(sensor_data)

# Automated Decision Logic
st.header("Irrigation Decision")
if soil_moisture < 30:
    irrigation_status = "ON"
    st.success("Irrigation Activated: Soil moisture is low.")
elif weather_forecast == "Rainy":
    irrigation_status = "OFF"
    st.info("Irrigation Deactivated: Rain expected.")
else:
    irrigation_status = "OFF"
    st.warning("Irrigation Deactivated: Soil moisture is sufficient.")

st.write(f"**Current Irrigation Status:** {irrigation_status}")

# Manual Override
st.header("Manual Irrigation Control")
manual_override = st.button("Toggle Irrigation Manually")
if manual_override:
    irrigation_status = "ON" if irrigation_status == "OFF" else "OFF"
    st.write(f"**Irrigation Status Manually Set to:** {irrigation_status}")

# Logs and Reports
st.header("Irrigation Logs")
logs = [
    {"Time": "08:00 AM", "Soil Moisture (%)": 25, "Action": "Irrigation Started"},
    {"Time": "10:00 AM", "Soil Moisture (%)": 35, "Action": "Irrigation Stopped"},
```

```python
]
log_df = pd.DataFrame(logs)
st.table(log_df)


st.header("Weather Data")
st.write(f"Forecast: {weather_forecast}")


# Footer
st.write("---")
st.caption("Prototype developed using Python and Streamlit.")
```

Conclusion

The **Smart Agriculture System for Irrigation** provides an efficient and sustainable solution for modern farming. By leveraging Streamlit for user-friendly interfaces and MySQL for reliable data management, the system automates irrigation based on real-time environmental data, optimizing water usage and enhancing crop productivity. It simplifies farm management, reduces resource wastage, and empowers farmers with timely notifications and actionable insights, contributing to sustainable agricultural practices.