PROJECT REPORT

Project Name: ***SMARTFARMER- IOT ENABLED SMART FARMING APPLICATION***

Team ID: PNT2022TMID04114

Team Members :

PADHMASHREE.S
JYOTI PAL
HEMAMALINI.S
HEMALATHA.G

1. INTRODUCTION
1.1 Project Overview
1.2 Purpose
2. LITERATURE SURVEY
2.1 Existing problem
2.2 References
2.3 Problem Statement Definition
3. IDEATION & PROPOSED SOLUTION
3.1 Empathy Map Canvas
3.2 Ideation & Brainstorming
3.3 Proposed Solution
3.4 Problem Solution fit
4. REQUIREMENT ANALYSIS
4.1 Functional requirement
5. PROJECT DESIGN
5.1 Data Flow Diagrams
5.2 Solution & Technical Architecture
6. PROJECT PLANNING & SCHEDULING
6.1 Sprint Planning, Schedule & Estimation
7. CODING & SOLUTIONING
7.1 Feature
8. TESTING
8.1 Test Cases
8.2 User Acceptance Testing
9. RESULTS
9.1 Performance Metrics
10. ADVANTAGES & DISADVANTAGES
11. CONCLUSION
12. FUTURE SCOPE
13. APPENDIX
Source Code
GitHub & Project Demo Link

INTRODUCTION:

Smart Farmer - IoT Enabled Smart Farming Application:

IoT-based agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors. Farmers can monitor all the sensor parameters by using a web or mobile application even if the farmer is not near his field. Watering the crop is one of the important tasks for the farmers. They can make the decision whether to water the crop or postpone it by monitoring the sensor parameters and controlling the motor pumps from the mobile application itself.

Project Flow:

1. The parameters like temperature, humidity, and soil moisture are updated to theWatson IoT platform.
2. The device will subscribe to the commands from the mobile application andcontrol the motors accordingly.
3. APIs are developed using Node-RED service for communicating with MobileApplication.
4. A mobile application is developed using the MIT App inventor to monitor thesensor parameters and control the motors.

To accomplish this, we have to complete all the activities and tasks listed below:

1. Create and configure IBM Cloud Services
2. Create IBM Watson IoT Platform
3. Create a device & configure the IBM IoT Platform
4. Create Node-RED service
5. Create a database in Cloudant DB to store all the sensor parameters
6. Develop a python script to publish and subscribe to the IBM IoT platform
7. Configure the Node-RED and create APIs for communicating with mobileapplication
8. Develop a mobile application to display the sensor parameters and control themotors

LITERATURE SURVEY:

[1] Kolli Revanth, Shaik Mohammed Arshad, Prathibhamol C.P Department of Computer Science and Engineering, Amrita VishwaVidyapeetham, Amritapuri, India

[2] CH Nishanthi1, Dekonda Naveen , Chiramdasu Sai Ram , KommineniDivya, Rachuri Ajay Kumar1 Associate Professor, ECE Dept., Teegala Krishna Reddy Engineering College, Hyderabad, India, Teegala KrishnaReddy Engineering College, Hyderabad, India

[3] Jirapond Muangprathub, Nathaphon Boonnam, Siriwan Kajornkasirat,Narongsak Lekbangpong, Apirat Wanichsombat, and Pichetwut Nil- laor. IoT and agriculture data analysis for the smart farm.Computers and electronics in agriculture.

IDEATION PHASE:

REFER:

https://github.com/IBM-EPBL/IBM-Project-22030-1659801442/blob/main/Project%20Design%20%26%20Planning/Ideation%20Phase/Ideation_Process.pdf

BRAINSTORMING:

https://github.com/IBM-EPBL/IBM-Project-22030-1659801442/blob/main/Project%20Design%20%26%20Planning/Ideation%20Phase/Brainstorm.pdf

EMPATHY MAP:

https://github.com/IBM-EPBL/IBM-Project-22030-1659801442/blob/main/Project%20Design%20%26%20Planning/Ideation%20Phase/Empathy%20map.pdf

PROBLEM SOLUTION FIT:

https://github.com/IBM-EPBL/IBM-Project-22030-1659801442/blob/main/Project%20Design%20%26%20Planning/Project%20Design%20Phase%201/Problem%20Solution%20Fit.pdf

FUNCTIONAL REQUIREMENT:

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Log in to system	Check Roles of Access. Check Credentials
FR-4	Manage Modules	Manage System Admins Manage Roles of User Manage User permission
FR-5	Check whether details	Temperature details Humidity details
FR-6	Log out	Exit

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

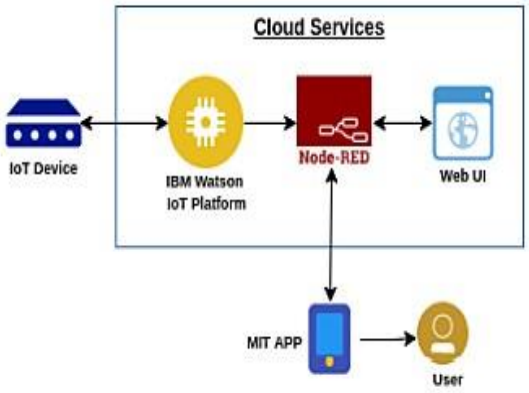
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Usability is defined as the ability to learn quickly, use something effectively, remember something, operate something without making a mistake, and enjoy something.
NFR-2	**Security**	Private and confidential information must be kept secure at all times, including during collection, processing, and storage.
NFR-3	**Reliability**	A superior cost-to-reliability trade-off is achieved with shared protection. To prevent agricultural service interruptions, the approach employs specialised and shared protection methods.
NFR-4	**Performance**	It will be more effective to monitor farming operations overall if integrated sensors are used to measure soil and ambient characteristics.
NFR-5	**Availability**	By tying information about crops, weather, and equipment together, it is feasible to automatically alter temperature, humidity, and other factors in farming equipment.

PROJECT DESIGN AND PLAN:

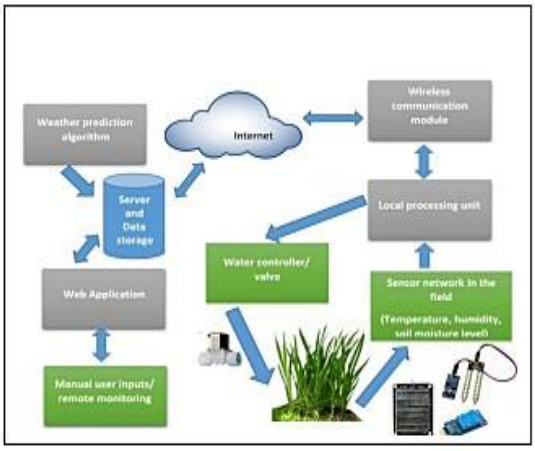
Data Flow Diagrams:

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

Example: (Simplified)



Example: DFD Level 0



User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	IoT devices	USN-1	Sensors and wi-fi module		High	Sprint-1
Customer	Software	USN-2	IBM Watson IoT platform, Workflows for IoT scenarios using Node-red		High	Sprint-2
Customer	MIT app	USN-3	To develop an application using MIT		High	Sprint-3
Customer	Web UI	USN-4	To make the user to interact with the software.	User can access the app for the services.	High	Sprint-4

PROJECT PLANNING AND SCHEDULING:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Hardware and Software	USN-1	Sensors without wifi - Simulation. .	2	High	Padhma shree, Jyoti Pal, Hema Malini, Hema Latha
Sprint-2	Software	USN-2	IBM Watson IoT platform, Workflows for IoTscenarios using Node-red	2	High	Padhma shree, Jyoti Pal, Hema Malini, Hema Latha
Sprint-3	MIT app	USN-3	To develop an mobile application using MIT	2	High	Padhma shree, Jyoti Pal, Hema Malini, Hema Latha
Sprint-4	Web UI	USN-4	To make the user to interact with software.	2	High	Padhma shree, Jyoti Pal, Hema Malini, Hema Latha

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	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		5thNOV 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12thNOV 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		14thNOV 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

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

CODING AND SOLUTIONING:

To make the user to interact with software:

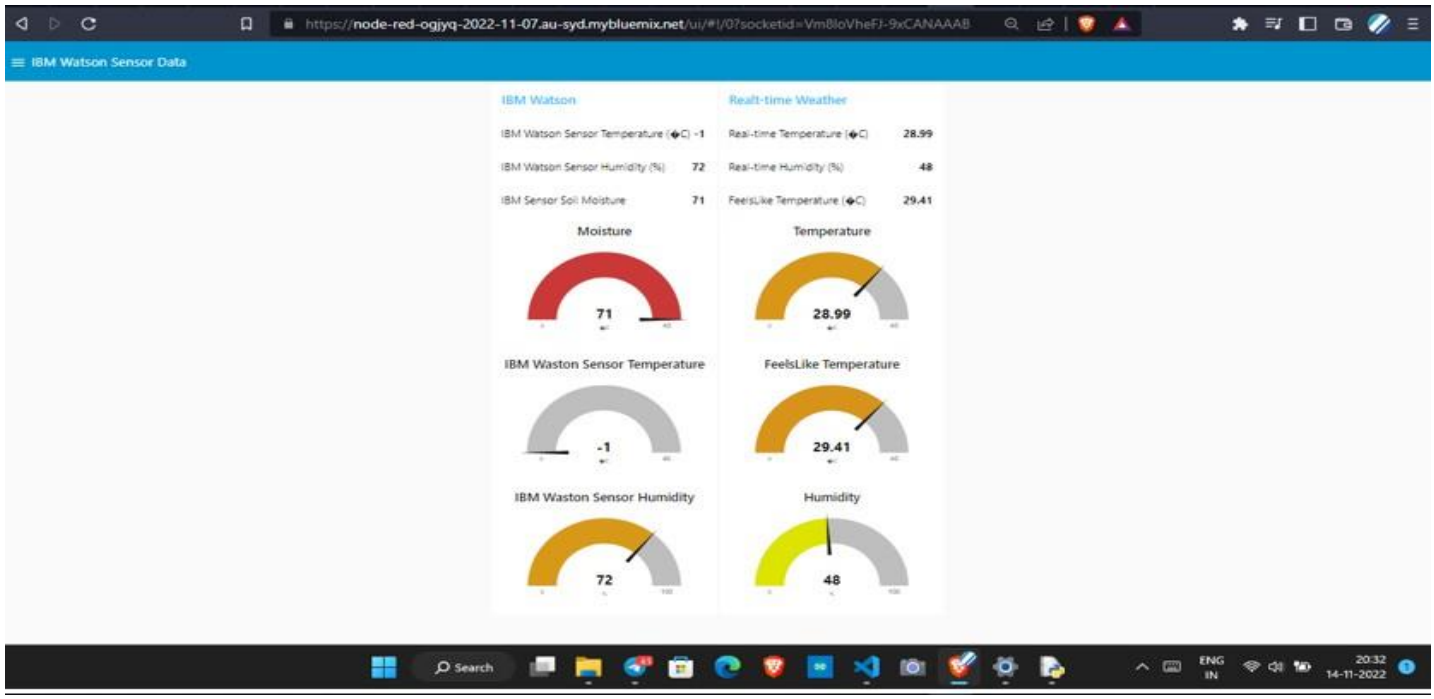
Receiving commands from IBM cloud using Python program:

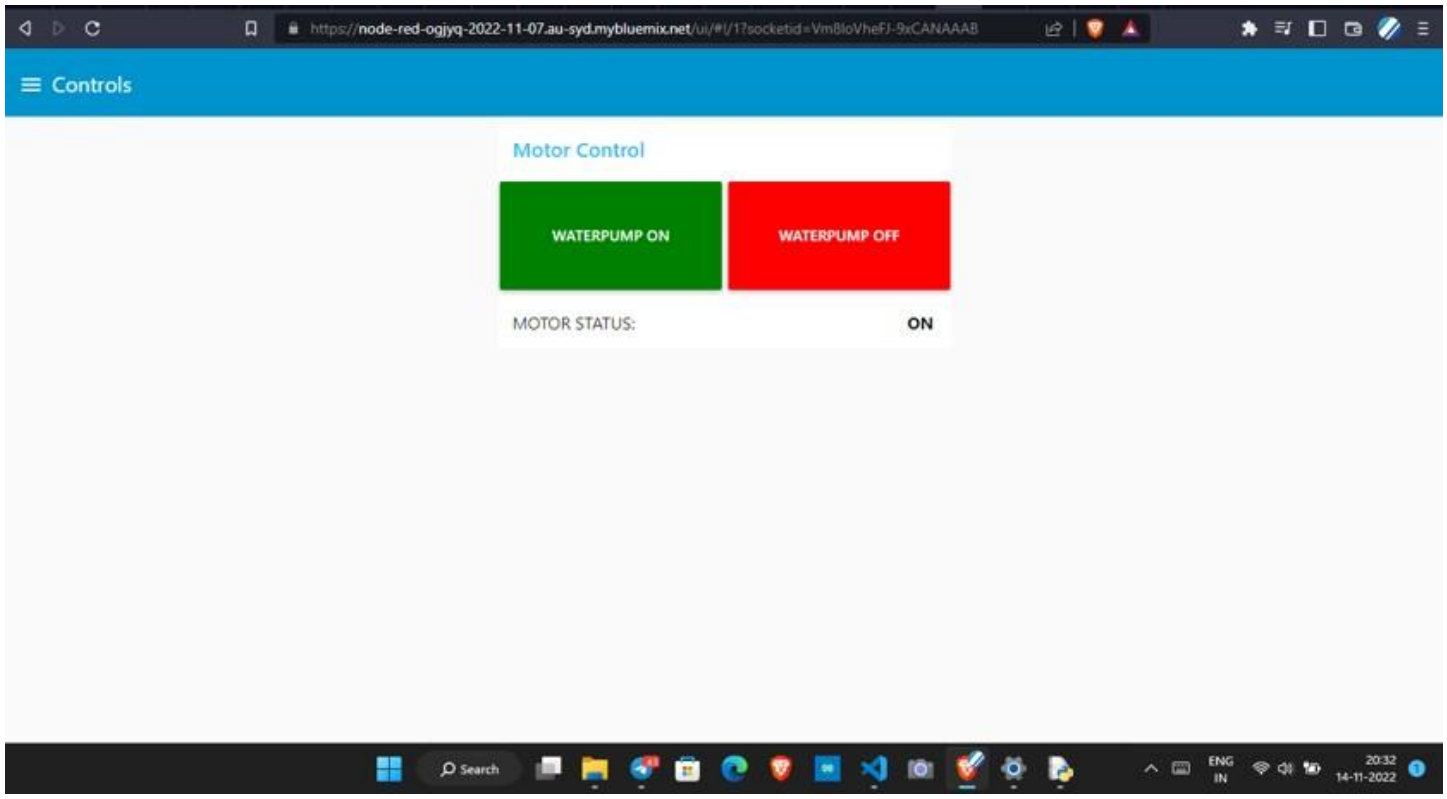
```
import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig = { "identity":{
"orgId": "04gt4e",
"typeId": "NodeMCU",
"deviceId": "12345"
},
"auth": {
"token": "123456789" }
}
client = wiotp.sdk.device.DeviceClient(config=myConfig,
logHandlers=None)
client.connect ()
def myCommandCallback (cmd) :
```

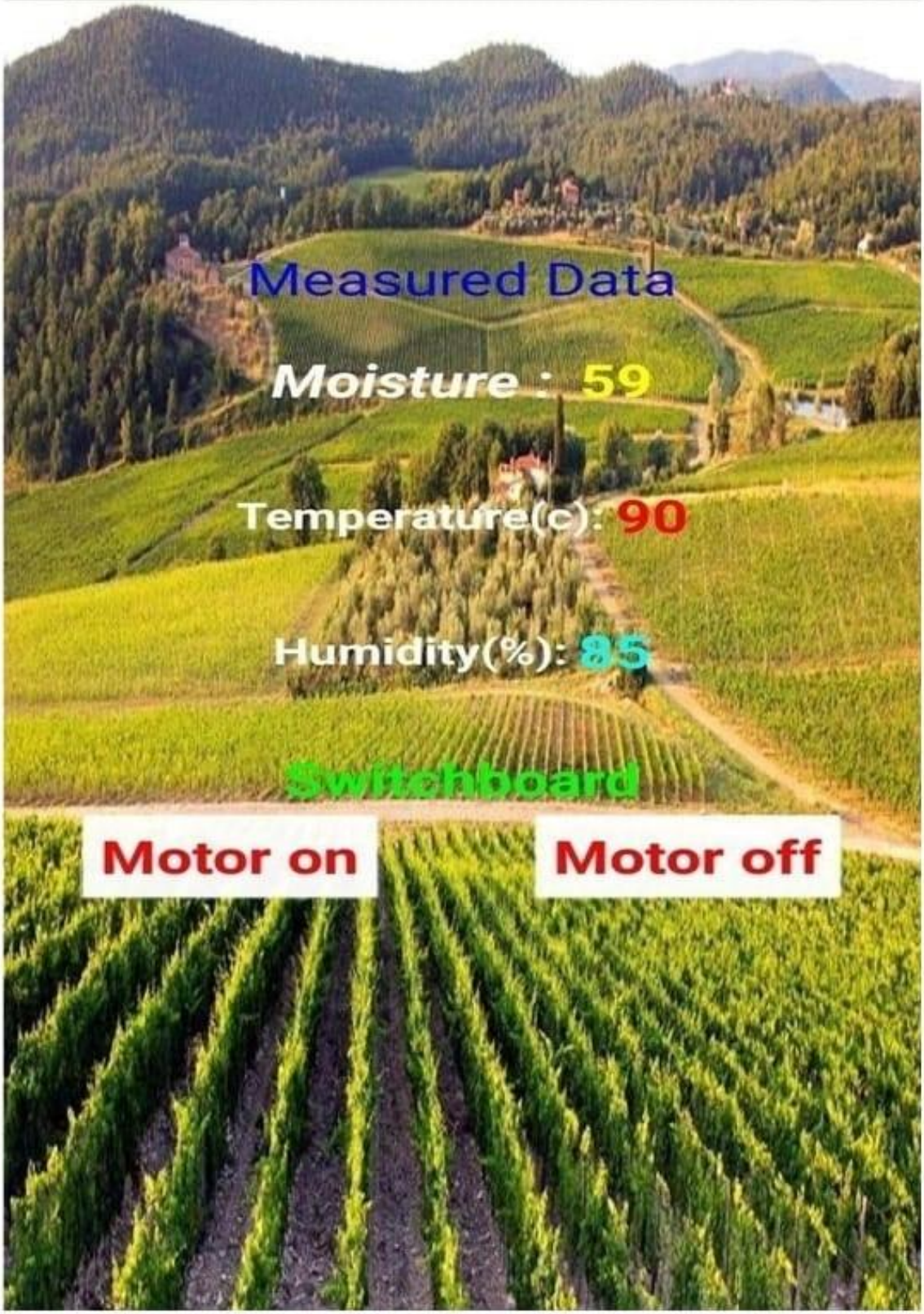
```
    print ("Message received from IBM IoT Platform: %s" %
cmd.data['command'])
    m=cmd.data['command']
    if(m=="motoron"):
        print ("Motor is switched on")
    elif(m=="motoroff"):
        print ("Motor is switched OFF")
    print(" ")


while True:
    soil=random.randint(10,100)
    temp=random.randint(-20, 125)
    hum=random.randint(0, 100)
    myData={'soil_moisture': soil, 'temperature':temp,
'humidity':hum}
    client.publishEvent (eventId="status", msgFormat="json",
data=myData, qos=0 , onPublish=None)
    print("Published data Successfully: %s", myData)
    time.sleep(2)
    client.commandCallback = myCommandCallback
client.disconnect()
```

USER INTERFACE – WEB APPLICATION:






Measured Data
Moisture : 59
Temperature(c): 90
Humidity(%): 85
Switchboard
Motor on
Motor off

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Smart Farmer - IoT Enabled Smart Farming Application project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and howthey were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested.

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

FEATURES:

- Comparative real time data from the internet
- Visual graph for easier understanding
- Separate tab for motor control and voice alert on commands
- SMS notification once the value falls below the threshold limit.

ADVANTAGES:

- Farms can be monitored and controlled remotely.
- Increase in convenience to farmers.
- Less labour cost.
- Better standards of living.

DISADVANTAGES:

- Lack of internet/connectivity issues.
- Added cost of internet and internet gateway infrastructure.
- Farmers wanted to adapt the use of Mobile App.

CONCLUSION:

Thus the objective of the project to implement an IOT system in order to help farmers to control and monitor their farms has been implemented successfully.

FUTURE SCOPE:

1. Yield prediction using ML models and cloud data.
2. More UI changes and features to improve user accessibility.

Github link:

https://github.com/IBM-EPBL/IBM-Project-22030-1659801442

Demo Video link:

https://drive.google.com/drive/u/0/folders/1K_FQK12NbGaFHHkuNV4R-5YoANu_nKV7?lfhs=2