



PROJECT REPORT

Project Name: SMART FARMER- IOT ENABLED SMART
FARMING APPLICATION.

1. INTRODUCTION

-  Project Overview
-  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
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams & User Stories
- 5.2 Solution & Technical Architecture

6. PROJECT PLANNING & SCHEDULING

- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

- 7.1 Feature
- 7.2 Database Schema (if Applicable)

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

SMART FARMING

1.INTRODUCTION:

PROJECT OVERVIEW:

This is system that enables framers to monitor and their forms with a web based application build with Node-RED.

It uses the IBM IOT Watson cloud platform as its Backend.

PURPOSE:

Smart Farming reduce the ecological foodprint of farming. Minimized or site specific application of inputs, such as fertilizers and pesticides ,in precision agriculture systems will mitigate leaching problems as well as the emission of greenhouse gases.

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

The biggest challenges faced by IoT in the agricultural sector are lack of information, high adoption costs , and security concerns , etc. Most of the farmers are not aware of the implementation of IoT in agriculture.

2.2 REFERENCES:

It is the application of modern ICT (Information and Communication Technologies) into agriculture. In IOT- based smart farming, a system is built for monitoring the crop field with the help of sensors (light, humidity, temperature, soil moisture, etc.). The farmers can monitor the field conditions from anywhere.

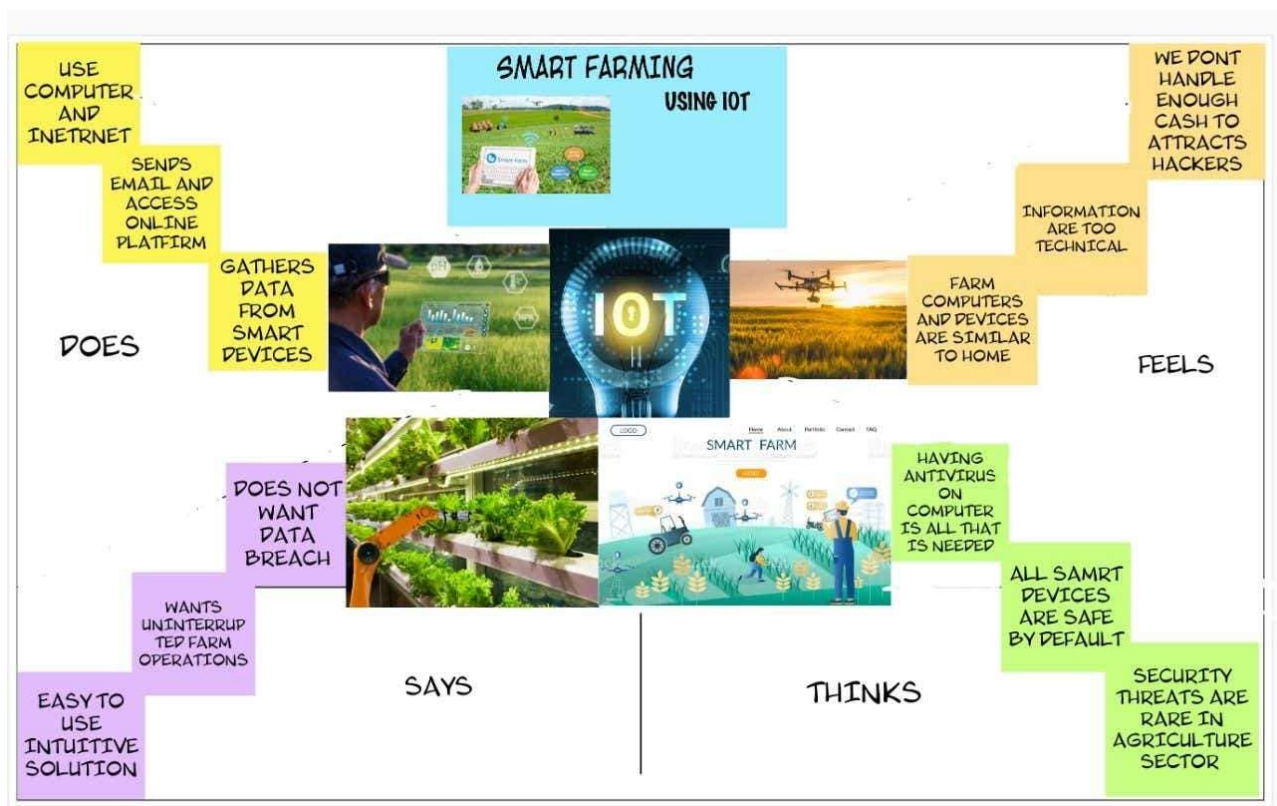
2.3 PROBLEM STATEMENT DEFINITION:

Overuse of pesticides and fertilizer in agricultural fields leads to destruction of the crop as well as reduces the efficiency of the field increasing the soil

vulnerability toward pest. IoT applications may be used to update the farmer/ user about type & quantity of pesticide required by the crop.

3. IDEATION & PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:

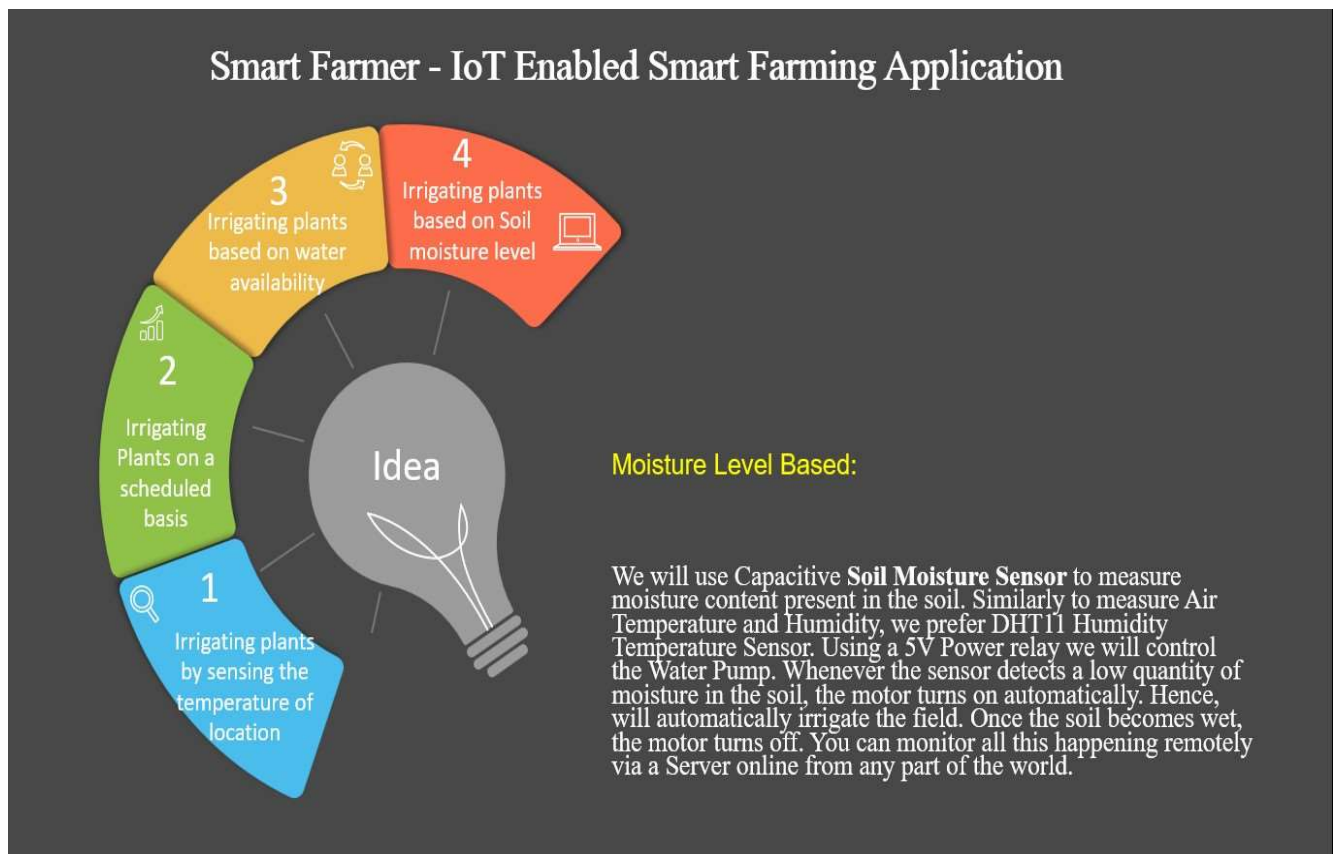


3.2 IDEATION & BRAINSTORMING:

Ideation is the create process of generating, developing, and communicating new ideas, where an idea is understood as a basic element of thought that can be either visual, concrete, or abstract.

Brainstorming is a group creative technique by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its members.

IDEATION PROCESS



3.3 Proposed Solution Template:

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To make farming easier by choosing several constraints in agriculture and to overcome those constraints, to increase production quality and quantity using IOT.
2.	Idea / Solution description	Using smart techniques like monitoring farms climate, smart irrigation and soil analysis.
3.	Novelty / Uniqueness	Solar power smart irrigation system which helps you to monitor temperature, moisture ,humidity using smart sensors.
4.	Social Impact / Customer Satisfaction	It is better than the present modern irrigation system by using this method we can control soil erosion. There will be better production yield.
5.	Business Model (Revenue Model)	As the productivity increases customer satisfaction also increases and hence need for the application also increases, which increases the revenue of the business.
6.	Scalability of the Solution	It is definetly scalable we ca increase the constraints when the problem arises.

3.4 PROBLEM SOLUTIONS FIT :

Project Title: Project Design Phase-I - Solution Fit Template

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) <small>Who is your customer? i.e. working parents of 8-5 yrs. kids</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">Farmers are our Customers</div>	6. CUSTOMER CONSTRAINTS <small>What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">The availability of device, proper Network facilities and budget are several constraints. Knowledge about the application.</div>	5. AVAILABLE SOLUTIONS <small>Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital networking</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">Most commonly used irrigation type is Drip irrigation the most common disadvantage is when the water is not filtered properly there will be clogs and the tubes will get affected easily. In smart farming we can use solar empowered smart irrigation system to overcome this.</div>	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS <small>Which jobs to be done (or problems) do you address for your customers? There could be more than one; explore different sides.</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">To make farming easier more quantitatively. 1. Monitoring farms climatic conditions. 2. Automatic systems for irrigation and Fertilization. 3. Soil analysis.</div>	9. PROBLEM ROOT CAUSE <small>What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations.</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">When there is no knowledge about the soil problem arises on what to be sowed, climatic conditions also play a major role. Knowledge on how to water the plants accordingly.</div>	7. BEHAVIOUR <small>What does your customer do to address the problem and get the job done? i.e. Directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">The customers will reach us when they don't have idea on how to analyse the soil and to improve the current irrigation system</div>	
Focus on J&P, tap into BE, understand RC	3. TRIGGERS <small>What triggers customers to act? i.e. seeing their neighbours installing solar panels, reading about a more efficient solution in the news.</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">To get correct accuracy on what to be done on the farm and to produce more crops and livestock quantitatively.</div>	10. YOUR SOLUTION <small>If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer job.</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">There will be less weed growth, Maximum use of water efficiently, Control of soil erosion and maximum crop yield.</div>	8. CHANNELS of BEHAVIOUR 8.1 ONLINE <small>What kind of actions do customers take online? Extract online channels from #7</small> 8.2 OFFLINE <small>What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">we will reach the customer directly ask about their problems and provide effective solutions if their problems match our application and provide them knowledge about our application to make their farming even more easier.</div>	Identify strong TR & EM
	4. EMOTIONS BEFORE / AFTER <small>How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure + confident, in control - use it in your communication strategy & design.</small> <div style="border: 1px dashed black; padding: 5px; margin-top: 10px;">As when the productivity increases farmers will be satisfied. They will not worry about the loss. Irrigation will be more efficient than before.</div>			

4.REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL ANALYSIS:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	IoT devices	Sensors and Wifi module.
FR-2	Software	Web UI, Node-red, IBM Watson, MIT app

4.2 NON FUNCTIONAL REQUIREMENTS:

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

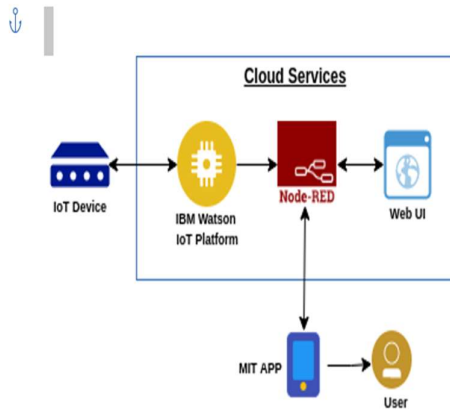
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Time consumability is less, Productivity is high.
NFR-2	Security	It has low level of security features due to integration of sensor data.
NFR-3	Reliability	Accuracy of data and hence it is Reliable.
NFR-4	Performance	Performance is high and highly productive.
NFR-5	Availability	With permitted network connectivity the application is accessible
NFR-6	Scalability	It is perfectly scalable many new constraints can be added

5. PROJECT DESIGN:

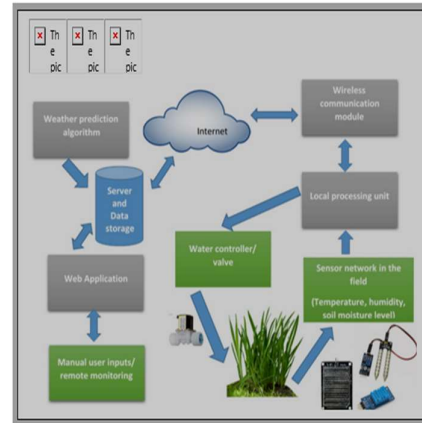
5.1 DATA FLOW DAIGRAMS AND USER STORIES:

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



5.2 SOLUTIONS AND TECHNICAL ARCHITECTURAL:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

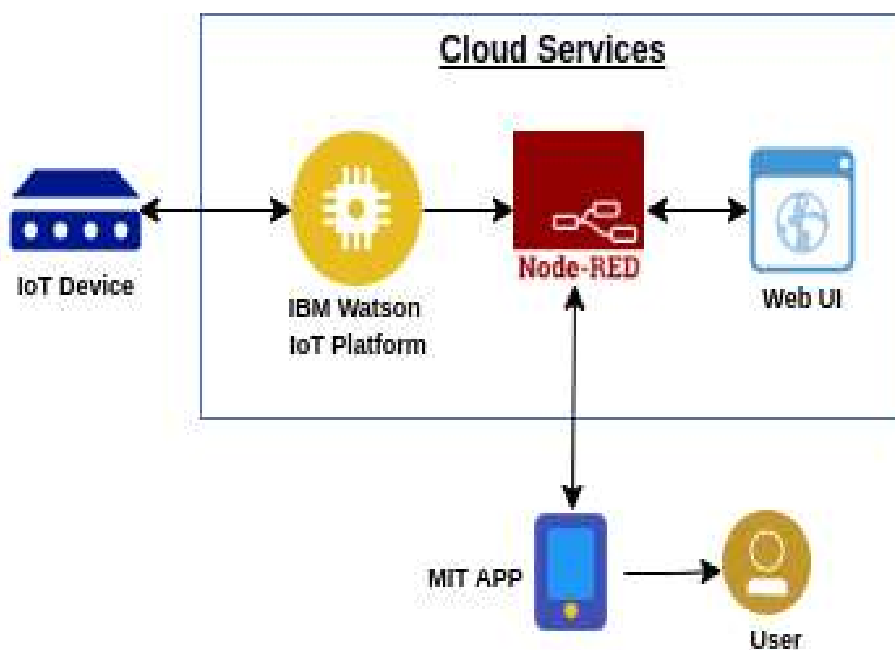


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	MIT app
2.	Application Logic-1	Logic for a process in the application	Node red/IBM Watson/MIT app
3.	Application Logic-2	Logic for a process in the application	Node red/IBM Watson/MIT app
4.	Application Logic-3	Logic for a process in the application	Node red/IBM Watson/MIT app
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM cloud.
7.	Temperature sensor	Monitors the temperature of the crop	
8.	Humidity sensor	Monitors the humidity	
9.	Soil moisture sensor (Tensiometers)	Monitors the soil temperature	
10.	Weather sensor	Monitors the weather	.
11.	Solar panel		.
12.	RTC module	Date and time configuration	
13.	Relay	To get the soil moisture data	

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	MIT app,Node-Red	Software
2.	Scalable Architecture	Drone technology, pesticide monitoring ,Mineral identification in soil	Hardware

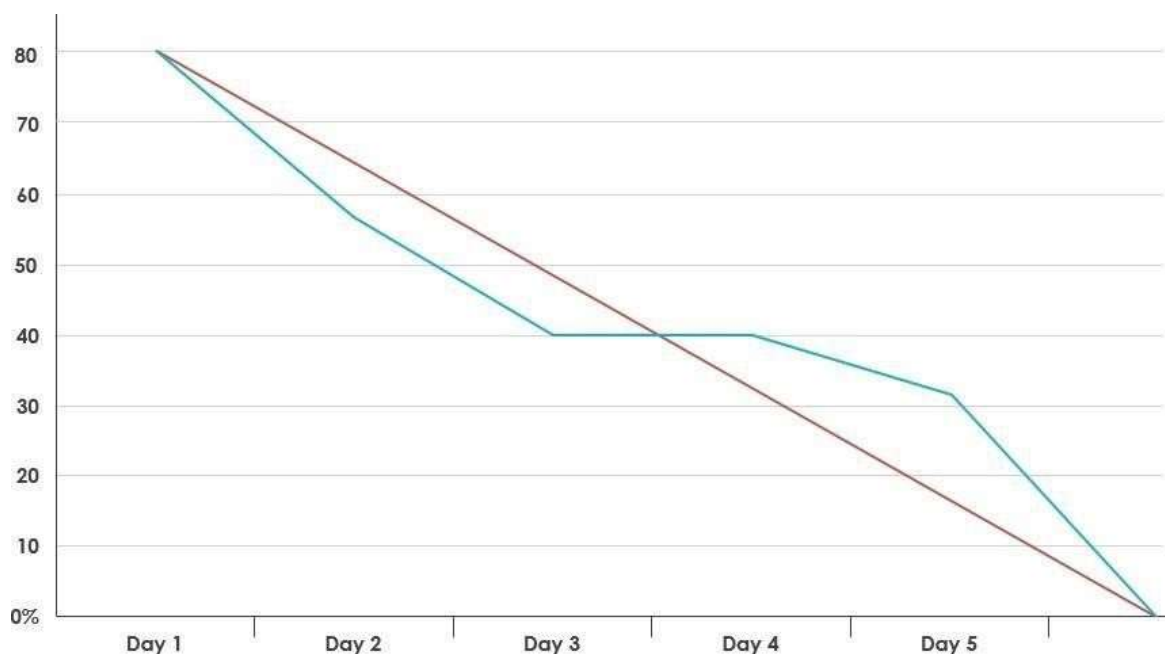
6.PROJECT PLANNING AND SCHEDULING:

Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task	Story Points	Priority	Team Member
Sprint-1	Registration (Farmer Mobile User)	UNS-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	A.Priyadharshini (Leader)
Sprint-1	Login	UNS-2	As a user, I will receive confirmation email once I have registered for the application	1	High	S.Santhanalakshmi (Member I)

Sprint-2	User Interface	UNS-3	As a user, I can register for the application through Facebook	3	Low	S.Kirubavathi (Member 2)
Sprint-1	Data Visualization	UNS-4	As a user, I can register for the application through GMAIL	2	Medium	G.Jaseema Begum (Member 3)
Sprint-3	Registration (Farmer - Web User)	USN - 1	As a user, I can log into the application by entering email and password	3	High	A.Priyadharshini (Leader)
Sprint - 2	Login	USN - 2	As a registered user, I need to easily login log into my registered account via the web page in minimum time	3	High	S.Santhanalakshmi (Member 1)
Sprint - 4	Web UI	USN - 3	As a user, I need to have a friendly user interface to easily view and access the resources	3	Medium	S.Kirubavathi (Member 2)
Sprint - 1	Registration (Chemical Manufacturer - Web user)	USN - 1	As a new user, I want to first register using my organization email and create a password for the account.	2	High	G.Jaseema Begum (Member 3)

Sprint - 4	Login	USN - 2	As a registered user, I need to easily log in using the registered account via the web page.	3	High	A.Priyadharshini (Leader)
Sprint - 3	Web UI	USN - 3	As a user, I need to have a user friendly interface to easily view and access the resources.	3	Medium	S.Santhanalakshmi (Member 1)
Sprint - 1	Registration (Chemical Manufacturer - Mobile User)	USN - 1	As a user, I want to first register using my email and create a password for the account.	1	High	S.Kirubavathi (Member 2)
Sprint - 1	Login	USN - 2	As a registered user, I need to easily log in to the application.	2	Low	G.Jaseema Begum (Member 3)

Burndown Chart:



7.CODING & SOLUTIONS:

FEATURE :

```
ibmiotpy - C:\Users\Priya\OneDrive\Desktop\users18\ibmiotpy (3.7.0)
File Edit Format Run Options Window Help

import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device
organization = "ck2ztf0"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "87654321"
# Initialize GPIO
def myCommandCallback(cmd):
    print("Commandreceived: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroff":
        print("motor is off")
    else :
        print ("please send proper command")

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
# .....
except Exception as e:
    print("Caught exception connecting device: %s" %str(e))
    sys.exit()
# Connect and send a datapoint "hello" with value "world" into the cloud as aneventof type "greeting" 10 times
deviceCli.connect()
while True:
    #Get Sensor Data fromDHT11
    temp=random.randint(90,110)
    Humid=random.randint(60,100)
    Mois=random.randint(20,120)
    data = { 'temp' : temp, 'Humid': Humid, 'Mois': Mois}
#Print data
def myOnPublishCallback():
    print ("Published Temperature = %s C" % temp, "humidity = %s %%" %Humid, "Moisture =%s deg c" % Mois, "to IBM Watson")
    success = deviceCli.publishEvent("IoTSensor", "json", data,qos=0,on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")
    time.sleep(10)
deviceCli.commandCallback = myCommandCallback
#Disconnect the device and application from the cloud
deviceCli.disconnect()
```

Ln: 49 Col: 22

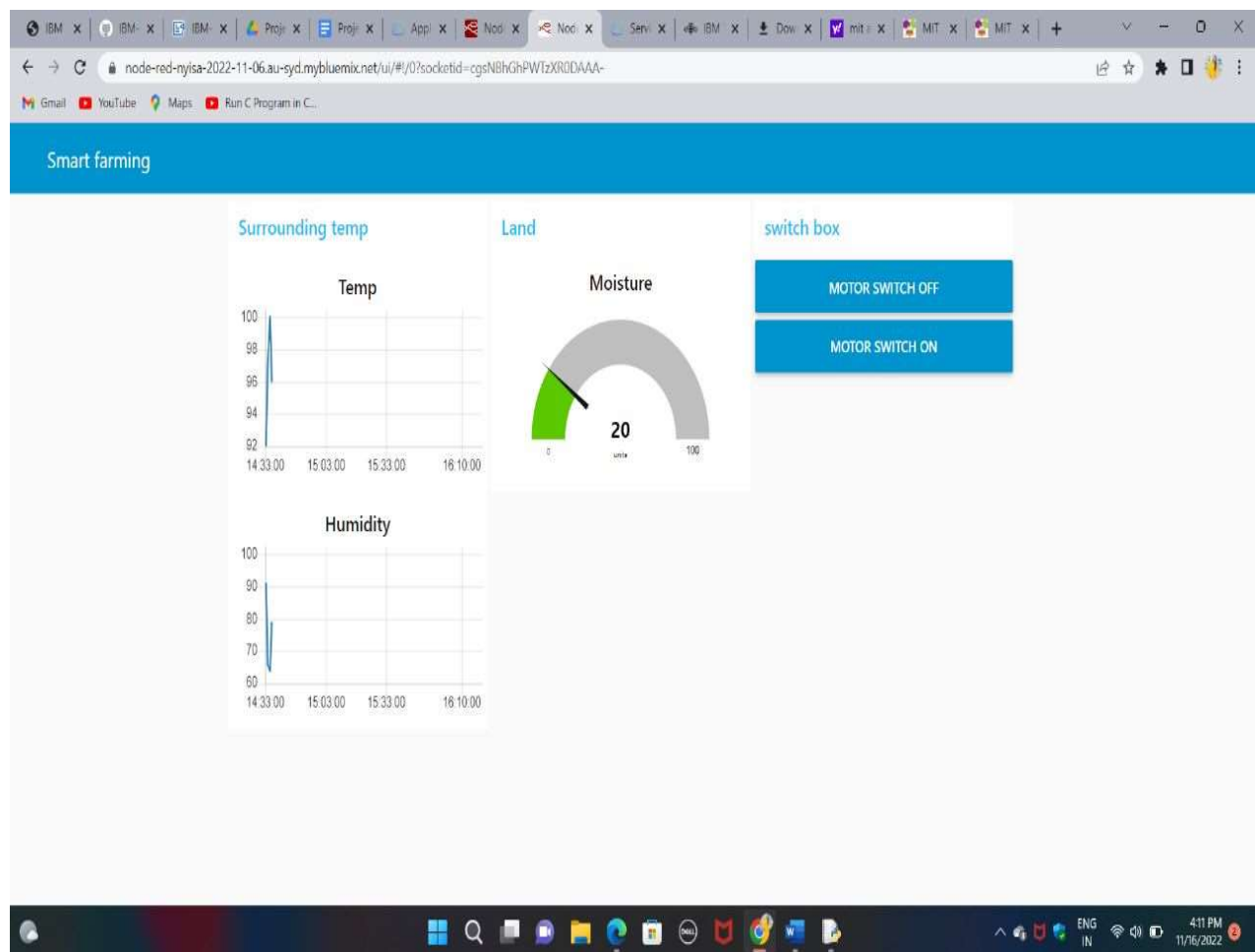
86°F
Mostly sunny

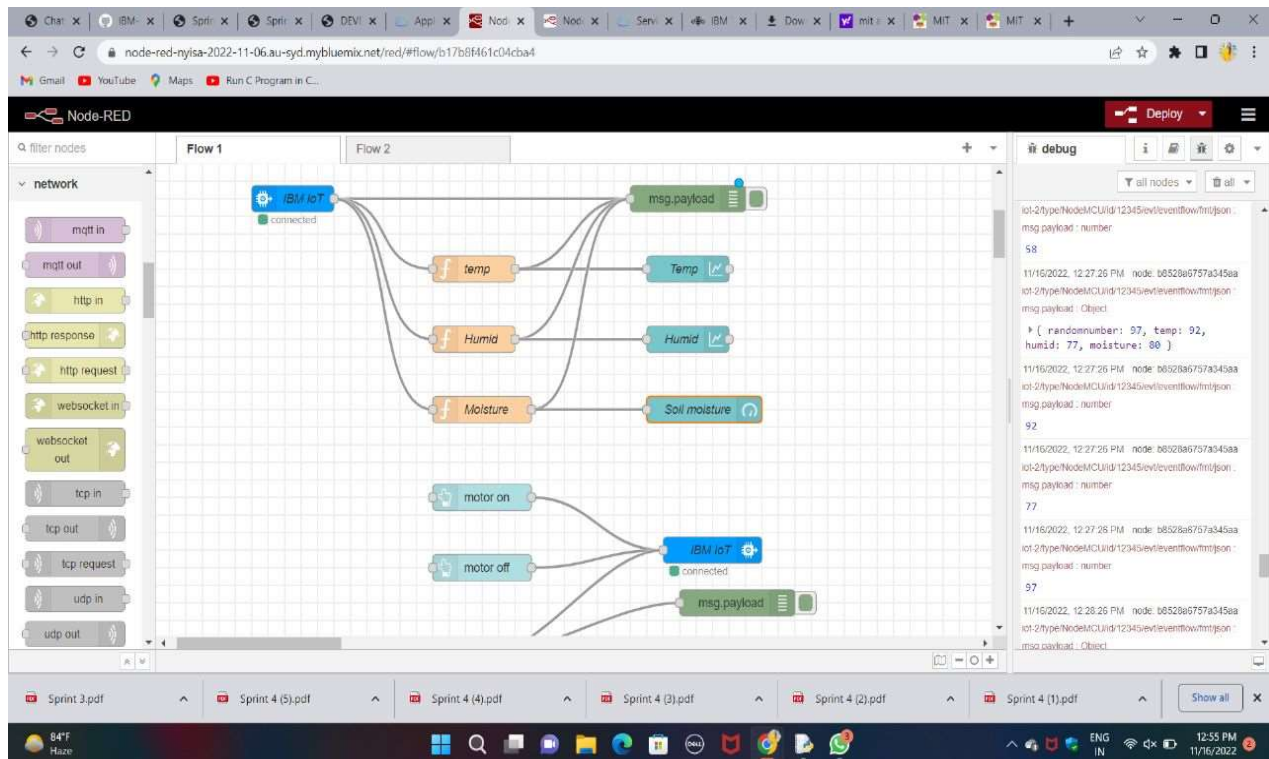
2:37 PM
11/7/2022

8.TESTING:

8.1 TEST CASE:

Web application using Node-RED.





ck2tf0.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

Browse Action Device Types Interfaces

Add Device

Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator

	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
>	123123	Disconnected	smartfarm	Device	16 Nov 2022 11:05	
>	12345	Connected	NodeMCU	Device	6 Nov 2022 16:26	

Items per page 50 | 1-2 of 2 items

1 Simulation running

Assignment 4 (1).pdf

Show all

86°F

Mostly sunny

3:55 PM

11/17/2022


```
ibmiot.py - C:\Users\Priya\OneDrive\Desktop\users18\ibmiot.py (3.7.0)
File Edit Format Run Options Window Help

import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device
organization = "ck2tf0"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "07654321"
# Initialize GPIO
def myCommandCallback(cmd):
    print("Commandreceived: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("Motor is on")
    elif status == "motorooff":
        print("Motor is off")
    else :
        print ("please send proper command")

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....
except Exception as e:
    print("Caught exception connecting device: %s" %str(e))
    sys.exit()

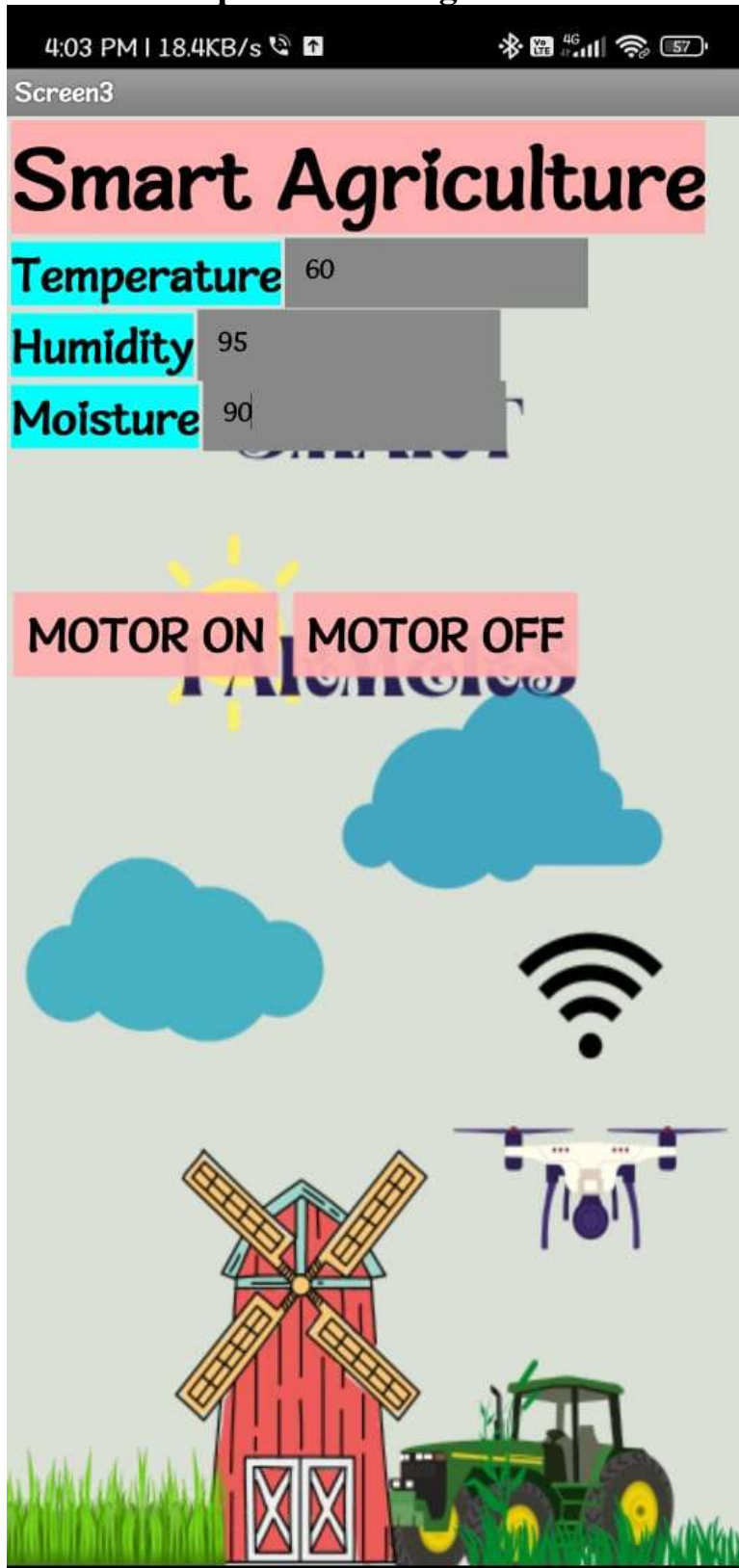
# Connect and send a datapoint "hello" with value "world" into the cloud as aneventof type "greeting" 10 times
deviceCli.connect()
while True:
    #Get Sensor Data fromDHT11
    temp=random.randint(90,110)
    Humid=random.randint(60,100)
    Mois=random.randint(20,120)
    data = { 'temp' : temp, 'Humid': Humid, 'Mois': Mois}
    #print data
    def myOnPublishCallback():
        print ("Published Temperature = %s c" % temp, "Humidity = %s %%" %Humid, "Moisture =%s deg c" % Mois, "to IBM Watson")
        success = deviceCli.publishEvent("IoTsensor", "json", data,qos=0,on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoT")
    time.sleep(10)
    deviceCli.commandCallback = myCommandCallback
#Disconnect the device and application from the cloud
deviceCli.disconnect()
```

Ln: 49 Col: 22

88°F Mostly sunny

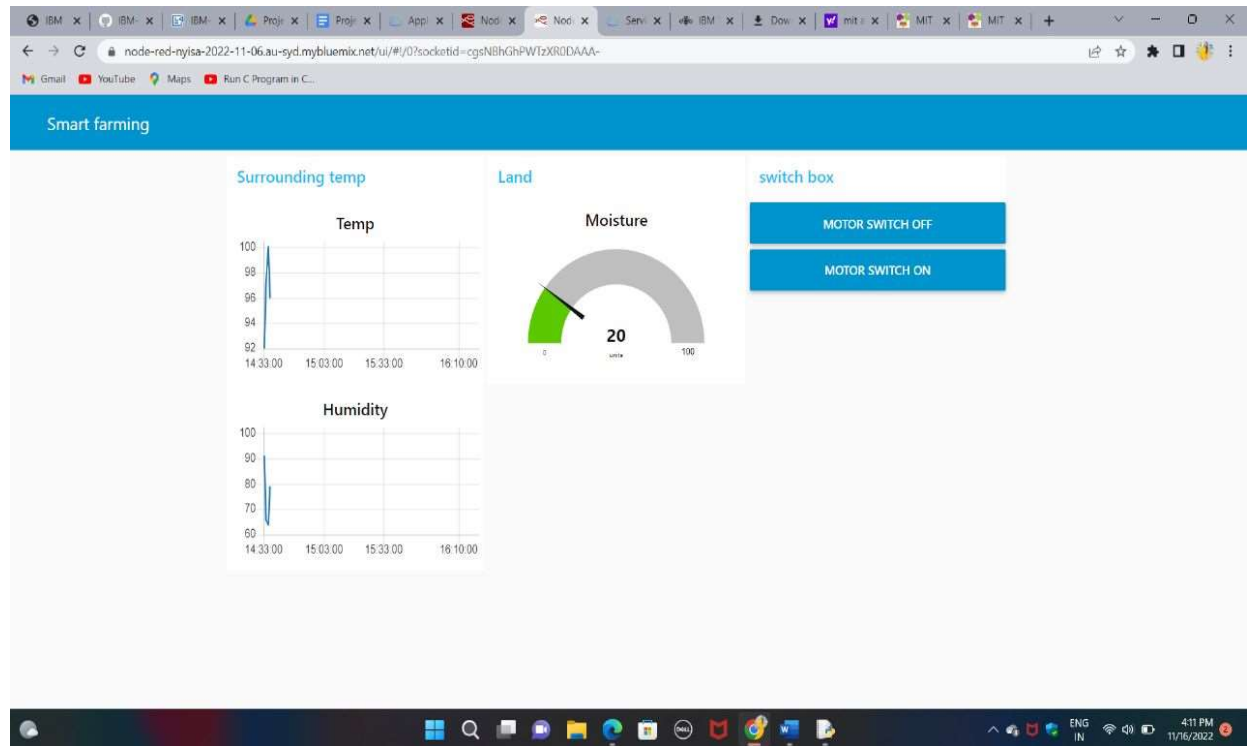
2:37 PM 11/17/2022

8.3 User Acceptance Testing



9. RESULT:

9.1 Performance Metrics



10.ADVANTAGES AND DISADVANTAGES:

10.1 ADVANTAGES:

- ❖ All the data like climatic conditions and changes in them, soil or crop conditions everything can be easily monitored.
- ❖ Risk of crop damage can be lowered to a greater extent.
- ❖ Many difficult challenges can be avoided making the process automated and the quality of crops can be maintained.
- ❖ The process included in farming can be controlled using the web applications from anywhere, anytime.

10.2 DISADVANTAGES:

- ❖ Smart Agriculture requires internet connectivity continuously, but rural parts cannot fulfil this requirement.
- ❖ Any faults in the sensors can cause great loss in the agriculture, due to wrong records and the actions of automated processes.
- ❖ IOT devices need much money to implement.

11.CONCLUSION:

An IOT based smart agriculture system using Watson IOT platform, Watson simulator, IBM cloud and Node-RED.

12.FUTURE SCOPE:

In future due to more demand of good and more farming in less time, for betterment of the crops and reducing the usage of extravagant resources like electricity and water IOT can be implemented in most of the places.

13.APPENDIX:

SOURCE CODE:

```
import wiotp.sdk.device
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device
organization = "ck2tf0"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken ="87654321"
# Initialize GPIO
def myCommandCallback(cmd):
    print("Commandreceived: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroff":
```

```

        print("motor is off")
    else :
        print ("please send proper command")

try:
    deviceOptions = {"org": organization, "type": deviceType, "id":
    deviceId, "auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....
except Exception as e:
    print("Caught exception connecting device: %s" %str(e))
    sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud
as aneventof type "greeting" 10 times
deviceCli.connect()
while True:
    #Get Sensor Data fromDHT11
    temp=random.randint(90,110)
    Humid=random.randint(60,100)
    Mois=random. randint(20,120)
    data = { 'temp' : temp, 'Humid': Humid , 'Mois': Mois}
    #print data
    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s %"
        %Humid, "Moisture =%s deg c" % Mois, "to IBM Watson")

```

```
    success = deviceCli.publishEvent("IoTSensor", "json",  
data,qos=0,on_publish=myOnPublishCallback)
```

if not success:

```
    print("Not connected to IoTTF")
```

```
time.sleep(10)
```

```
deviceCli.commandCallback = myCommandCallback
```

#Disconnect the device and application from the cloud

```
deviceCli.disconnect()
```

OUTPUT:

```
Published Moisture = 90 deg C Temperature = 96 C Humidity = 76 % to IBM Watson  
Published Moisture = 102 deg C Temperature = 110 C Humidity = 66 % to IBM Watson  
Published Moisture = 45 deg C Temperature = 99 C Humidity = 100 % to IBM Watson  
Command received: motoron  
motor is on  
Published Moisture = 77 deg C Temperature = 91 C Humidity = 85 % to IBM Watson  
Published Moisture = 73 deg C Temperature = 94 C Humidity = 86 % to IBM Watson  
Command received: motoroff  
motor is off  
Published Moisture = 101 deg C Temperature = 104 C Humidity = 87 % to IBM Watson
```

Github link : <https://github.com/IBM-EPBL/IBM-Project-51511-1660980188>

Project Demo link : <https://photos.app.goo.gl/AtvVJhyUrnQ1gEhSA>

THANK YOU....