PROJECT REPORT

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

Team ID: PNT2022TMID15629

Team:

ARUN R – TEAM LEAD

DILIPAN B – TEAM MEMBER 1

DINESH P – TEAM MEMBER 2

GOKUL R – TEAM MEMBER 3

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 food print of farming. Minimized or sitespecific application of inputs, such as fertilizers and pesticides, in precision agriculture systems will mitigate leaching problems as well as the emission ofgreenhouse gases.

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

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

2.2 REFERENCES:

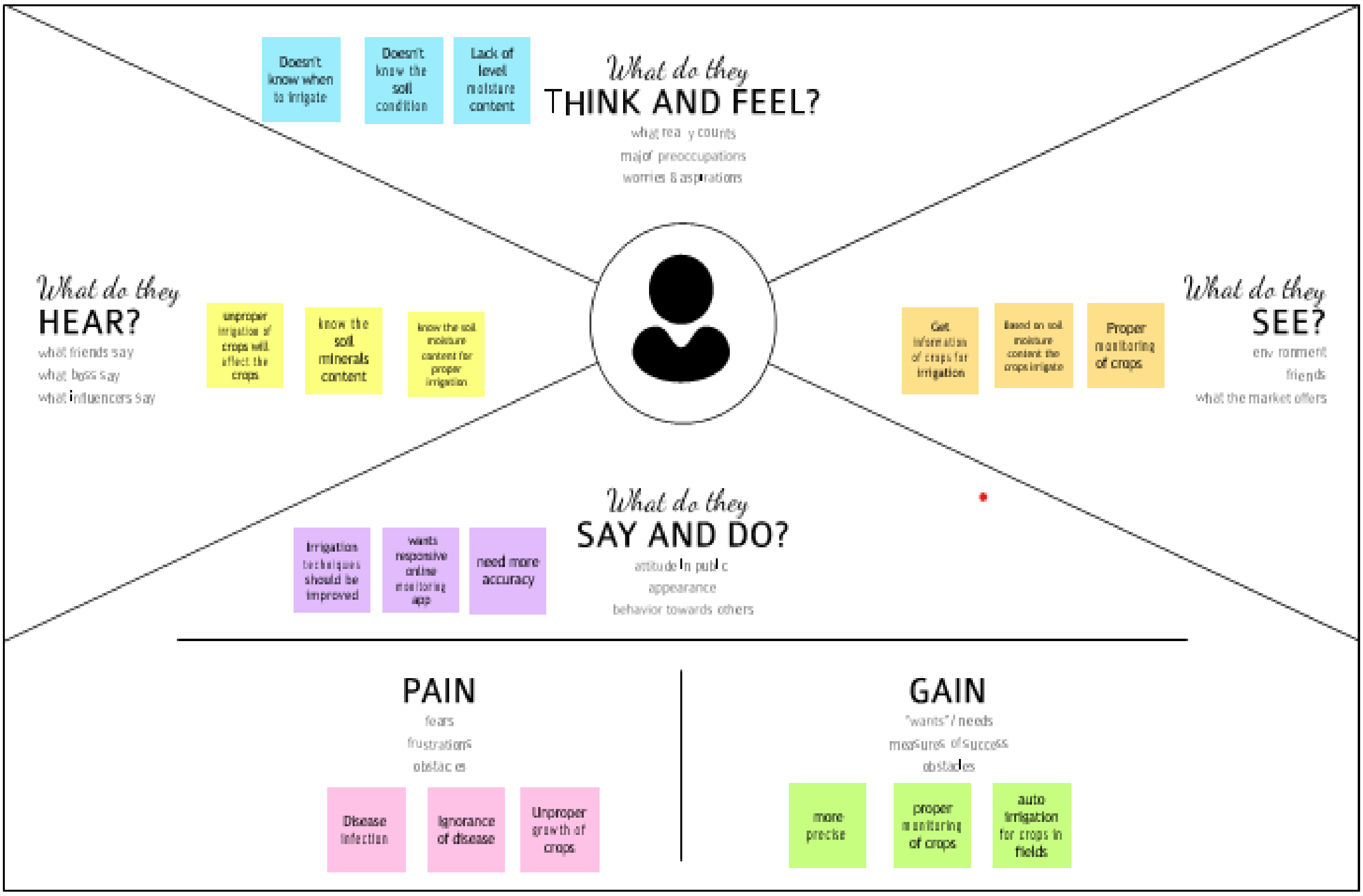
It is the application of modern ICT (Information and Communication Technologies) into agriculture. In IOT- based smart farming, asystem is built for monitoring the crop field with the help of sensors (light, humidity, temperature, soil moisture, etc.). The farmers can monitor the field conditions from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

2.3 PROBLEM STATEMENT DEFINITION:

Overuse of pesticides and fertilizer in agricultural fields leads to destruction ofthe 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:

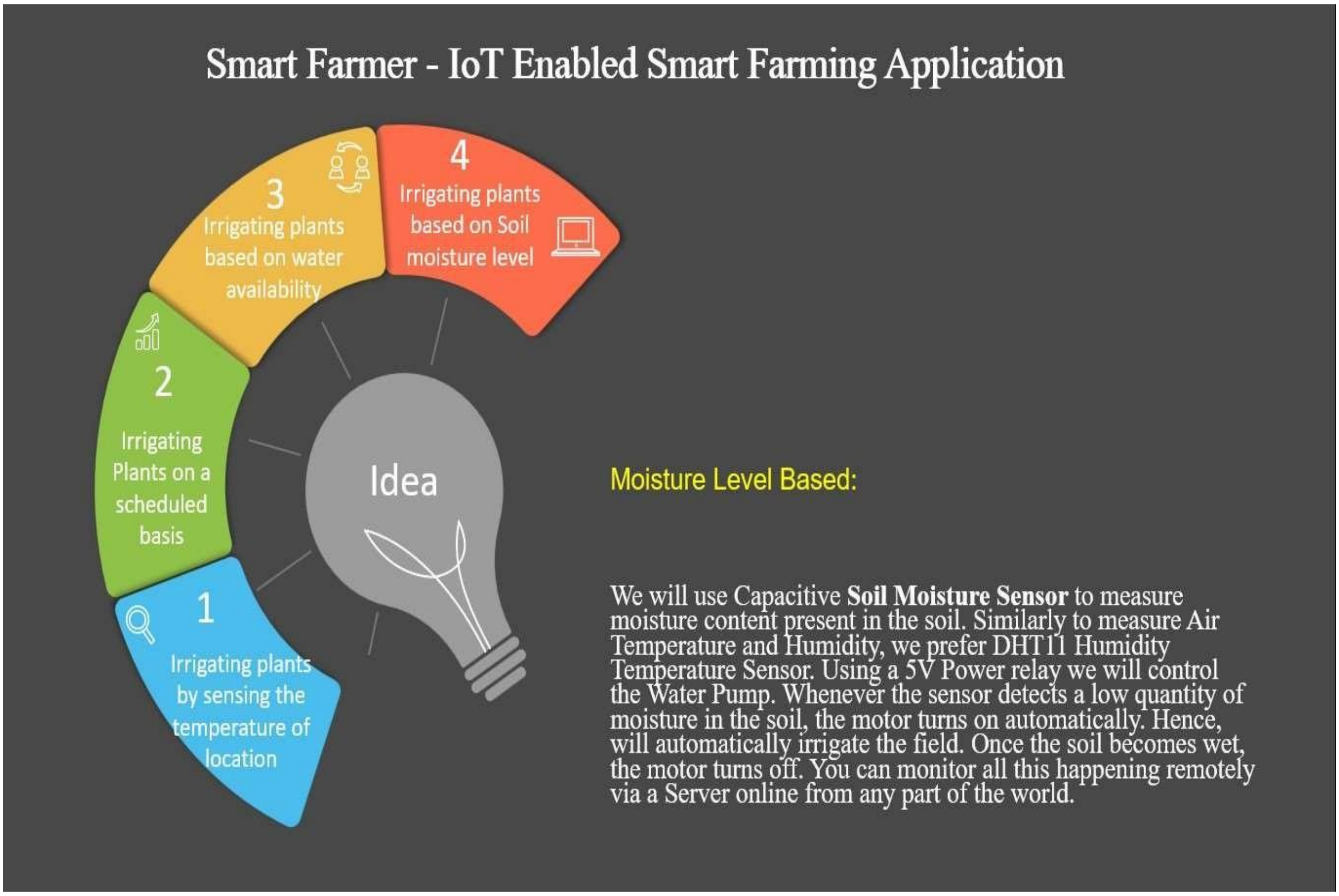


3.2 IDEATION & BRAINSTORMING:

Ideation is the create process of generating, developing, andcommunicating new ideas, where an is idea understood as a basic element of thought that can beeither 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 (Problemto be solved)	To make farming easier by choosing several constraints in agriculture andto 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 systemwhich helps you to monitor temperature, moisture, humidity using smart sensors.
4.	Social Impact / Customer Satisfaction	It is better than the present modernirrigation 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 increasesand hence need for the application also increases, which increases the revenue of the business.
6.	Scalability of the Solution	It is definitely scalable we ca increase the constraints when the problemarises.

3.4 PROBLEM SOLUTIONS FIT:

Project Title: Smart Farmer – IoT Enabled Smart Farming Application

Project Design Phase- I - Solution Fit

Team ID: PNT2022TMID15629

Define CS, fit into CC

1. Customer Segment(S) CS

Who is your customer? i.e. working parents of 0-5 y.o. kids

The customer for this product is a farmer who grows crops. Our goal is to help them, monitor field parameters remotely. This product saves agriculture from extinction.

6. Customer Constrains CC

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

Using many sensors is difficult. An unlimited or continuous internet connection is required for success.

5. AVAILABLE SOLUTIONS AS

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

The irrigation process is automated using IoT. Meteorological data and field parameters were collected and processed to automate the irrigation process. Disadvantages are efficiency only over short distances, and difficult data storage.

Explore AS, differentiate

Focus on J&P, tap into BE, underst

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

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

The purpose of this product is to use sensors to acquire various field parameters and process them using a central processing system. The cloud is used to store and transmit data using IoT. The Weather API is used to help farmers make decisions. Farmers can make decisions through mobile applications.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?

Frequent changes and unpredictable weather and climate made it difficult for farmers to engage in agriculture. These factors play an important role in deciding whether to water your plants. Fields are difficult to monitor when the farmer is not at the field, leading to crop damage.

7. BEHAVIOUR BE

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)

Use a proper drainage system to overcome the effects of excess water from heavy rain. Use of hybrid plants that are resistant to pests.

Focus on J&P, tap into BE, underst

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	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Log in to system	Check Credentials Check Roles of Access.
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

4.2 NON-FUNCTIONAL REQUIREMENTS:

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

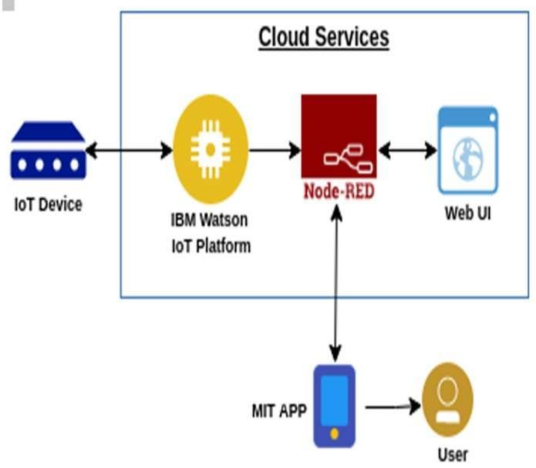
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Usability includes easy learn ability, efficiency in use, remember ability, lack of errors in operation and subjective pleasure.
NFR-2	**Security**	Sensitive and private data must be protected fromtheir production until the decision-making and storage stages.
NFR-3	**Reliability**	The shared protection achieves a better trade-off between costs and reliability. The model uses dedicated and shared protectionschemes to avoid farm service outages.
NFR-4	**Performance**	the idea of implementing integrated sensors with sensing soil and environmental or ambient parametersin farming will be more efficient for overall monitoring.
NFR-5	**Availability**	Automatic adjustment of farming equipment made possible by linking information like crops/weather and equipment to auto-adjust temperature, humidity, etc.
NFR-6	**Scalability**	Scalability is a major concern for IoT platforms. It has shown that different architectural choices of IoT platforms affect system scalability, and that automatic real time decision-making is feasible in an environment composed of dozens of thousand.

5. PROJECT DESIGN:

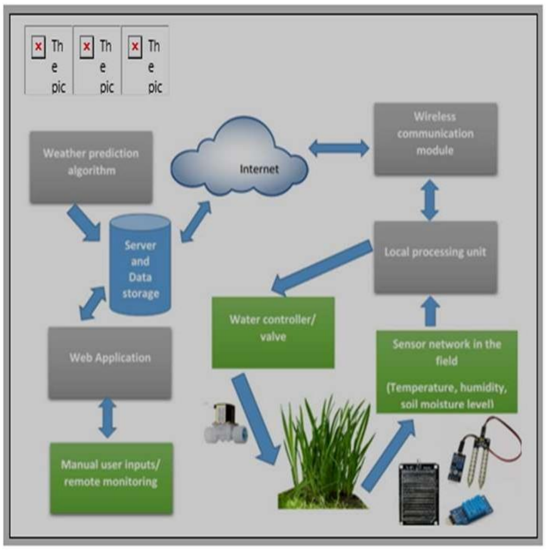
5.1 DATA FLOW DAIGRAMS AND USER STORIES:

A Data Flow Diagram (DFD) is a traditional visual representation of the informationflows 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

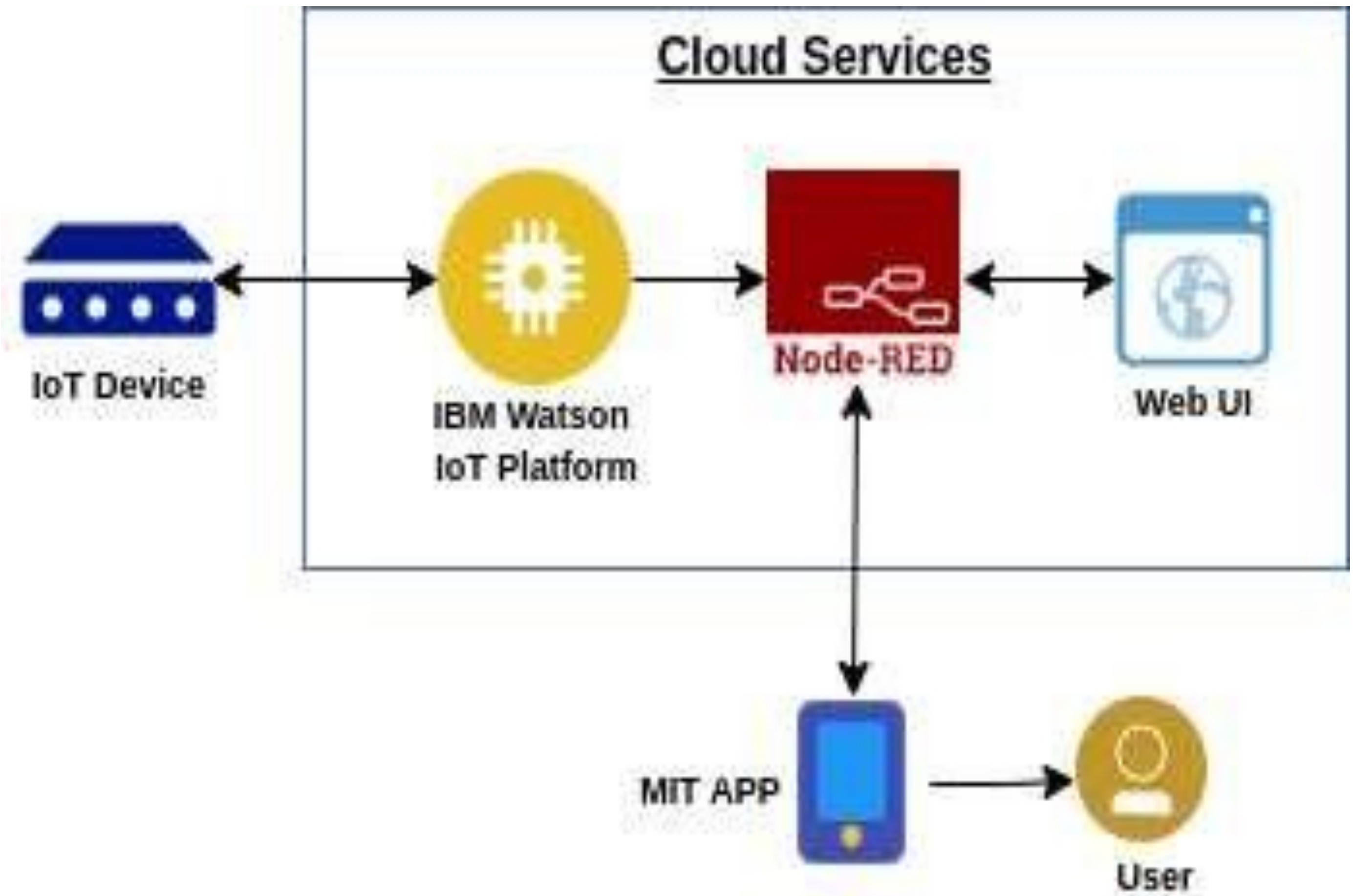


Table-1: Components & Technologies:

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, Configurationsetc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM cloud.
7.	Temperature sensor	Monitors the temperature ofthe crop	
8.	Humidity sensor	Monitors the humidity	
9.	Soil moisture sensor (Tensiometers)	Monitors the soiltemperature	
10.	Weather sensor	Monitors the weather	.
11.	Solar panel		.
12.	RTC module	Date and time configuration	
13.	Relay	To get the soil moisturedata	

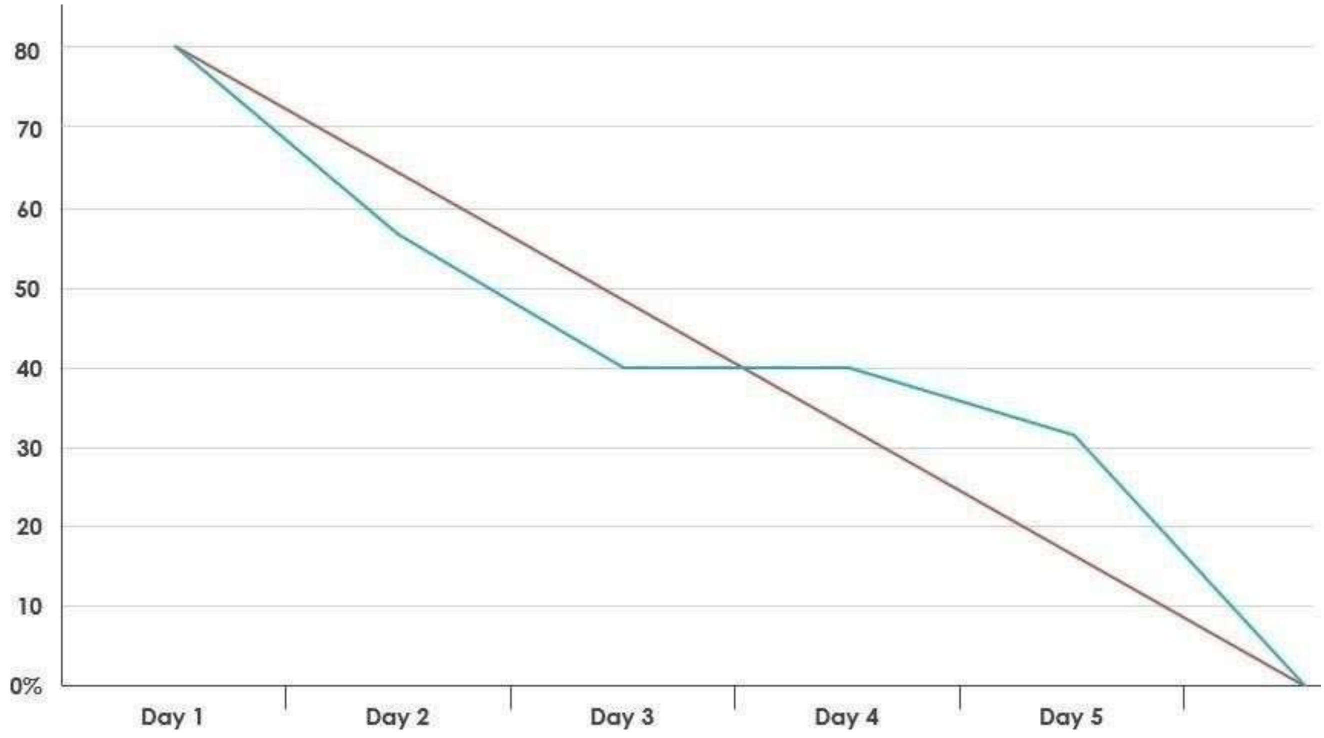
Table-2: Application Characteristics:

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

6. PROJECT PLANNING AND SCHEDULING:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint - 1	Creating Hardware Simulation	USN - 1	Connect Sensors and Wi - Fi modules by usingPython code	2	High	Arun, Dilipan, Dinesh, Gokul
Sprint - 2	Using Software	USN - 2	Creating device in the IBM Watson IOT platform, to making workflow of IOT scenarios using Node – Red service	2	High	Arun, Dilipan, Dinesh, Gokul
Sprint - 3	MIT App Inventor	USN - 3	Develop a mobile application for the SmartFarmer project using MIT App Inventor	2	High	Arun, Dilipan, Dinesh, Gokul
Sprint - 4	Web UI	USN - 4	To make the user to interact with software	2	High	Arun, Dilipan, Dinesh, Gokul

Burndown Chart:



7. CODING & SOLUTIONS:

FEATURE:

ibmiotpublishsubscribe.py - C:\Users\ARUN\Downloads\ibmiotpublishsubscribe.py (3.7.0)

File Edit Format Run Options Window Help

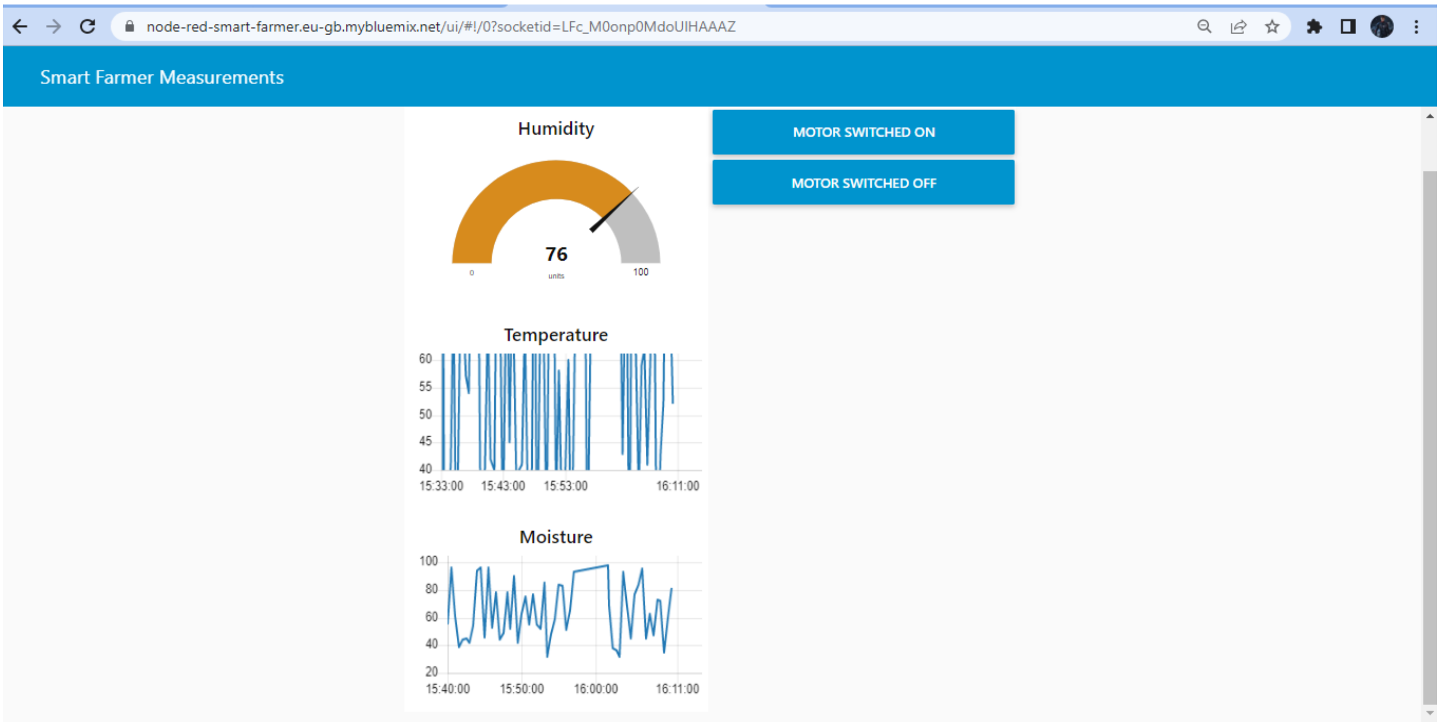
```python
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "bafc8x"
deviceType = "Smart-Farmer"
deviceId = "Smart-farmer"
authMethod = "token"
authToken = "8bxi1t(&hrcGQg2sd*"
# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("Led is on")
    else :
        print ("Led is off")
    #print(cmd)
try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMe
        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 an event of type "greeting"
deviceCli.connect()
```

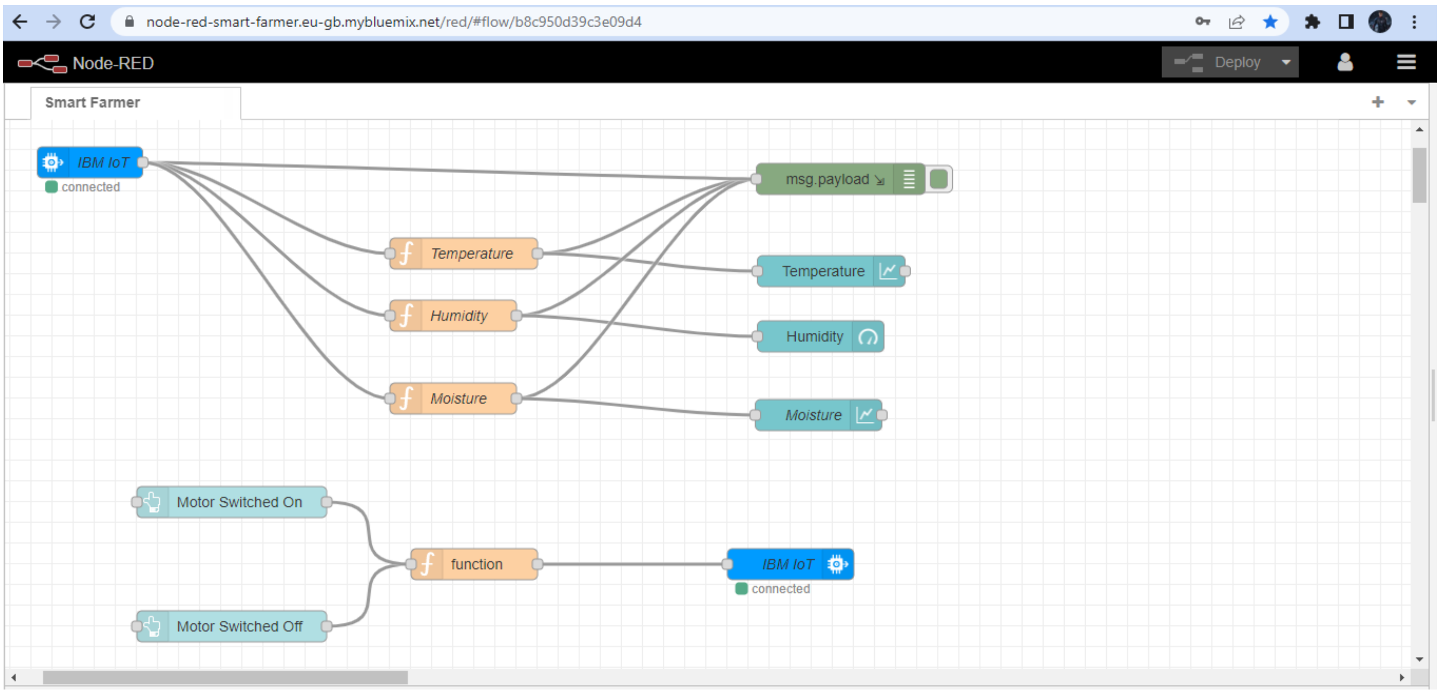
Ln: 42 Col: 53

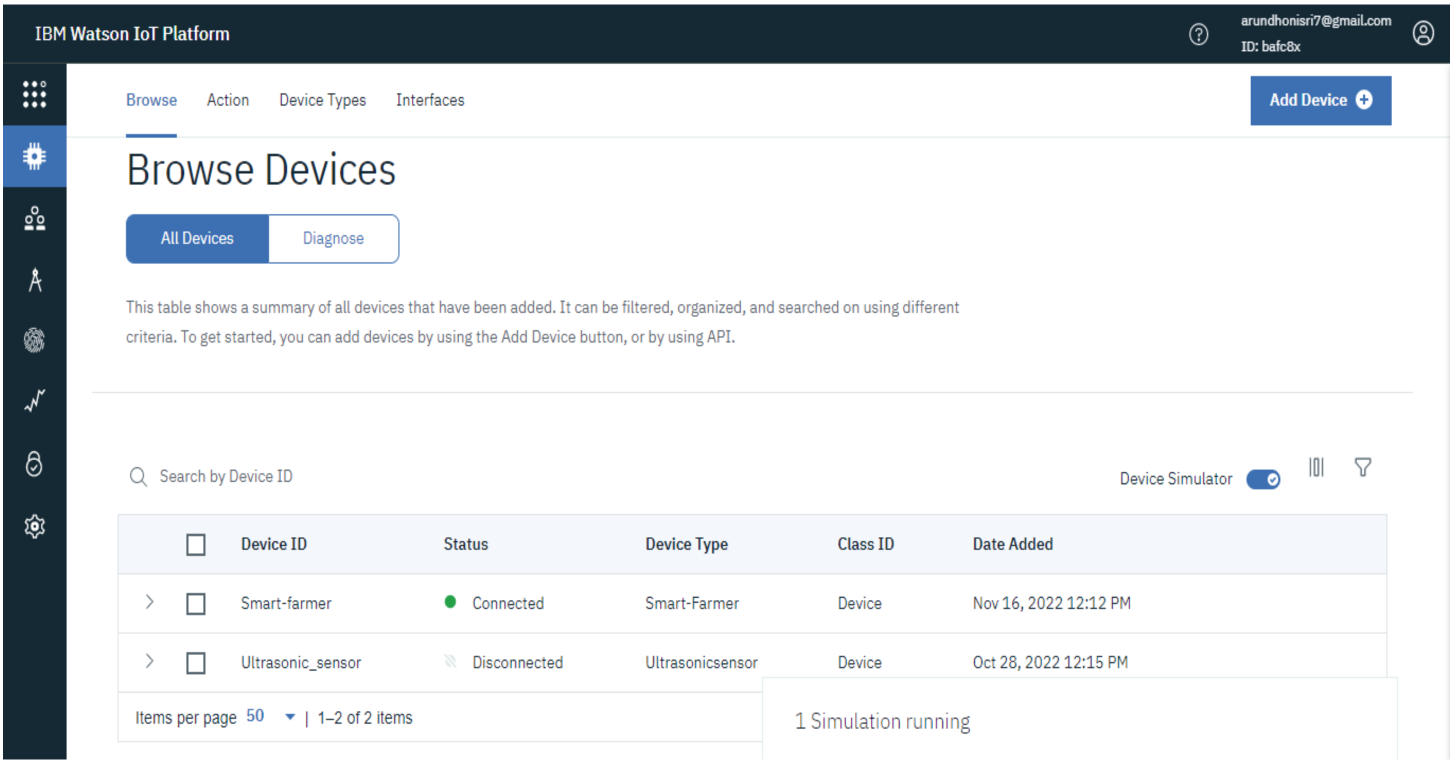
TESTING:

7.1 TEST CASE:

Web application using Node-RED







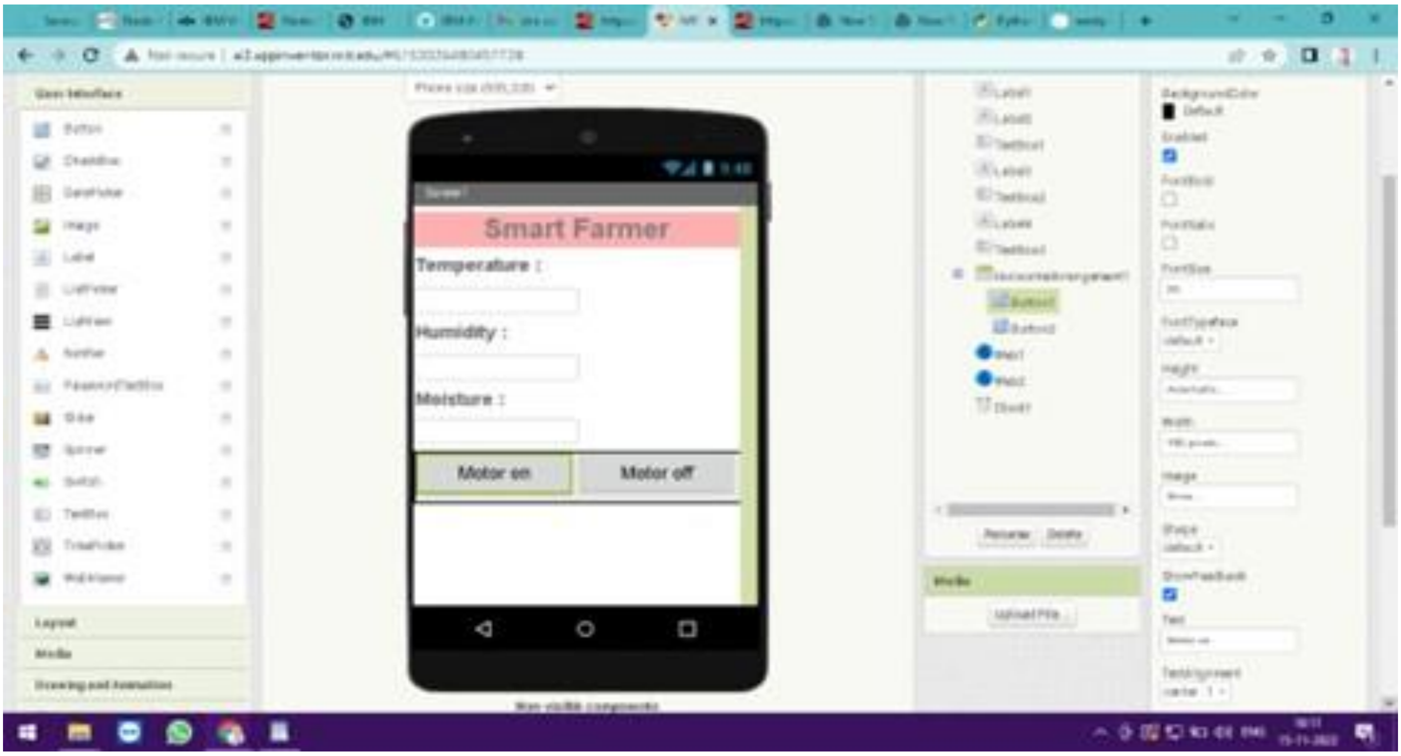
ibmiotpublishsubscribe.py - C:\Users\ARUN\Downloads\ibmiotpublishsubscribe.py (3.7.0)

File Edit Format Run Options Window Help

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "bafc8x"
deviceType = "Smart-Farmer"
deviceId = "Smart-farmer"
authMethod = "token"
authToken = "8bxi1t(&hrcGQg2sd*"
# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("Led is on")
    else :
        print ("Led is off")
    #print(cmd)
try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMe
        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 an event of type "greeting"
```

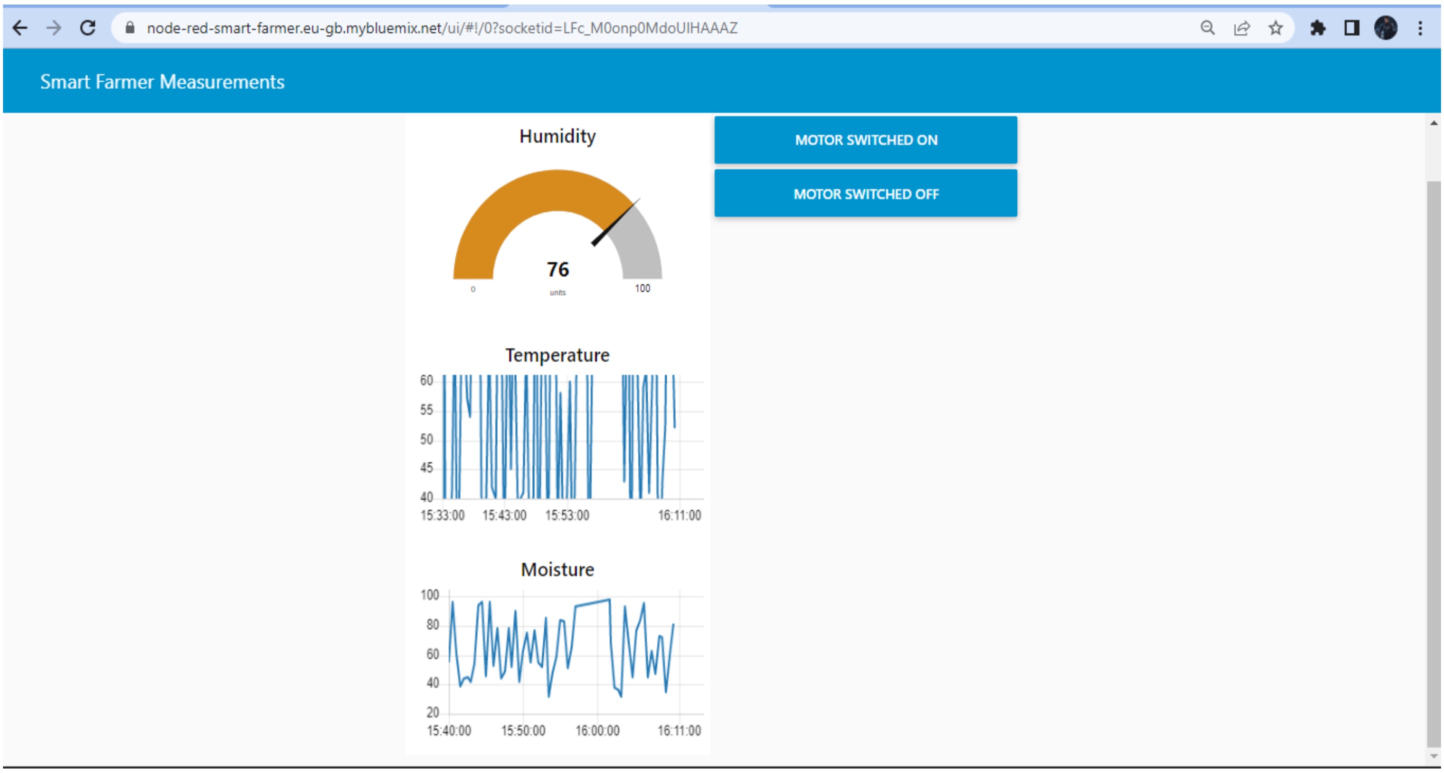
Ln: 42 Col: 53

8.3 User Acceptance Testing



8. RESULT:

9.1 Performance Metrics



9. ADVANTAGES AND DISADVANTAGES:

9.1ADVANTAGES:

- All the data like climatic conditions and changes in them, soil orcrop 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.

9.2 DISADVANTAGES:

- Smart Agriculture requires internet connectivity continuously, butrural 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.

10. CONCLUSION:

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

11. 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 implementedin most of the place

12. APPENDIX:

SOURCE CODE:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "bafc8x"
deviceType = "Smart-Farmer"
deviceId = "Smart-farmer"
authMethod = "token"
authToken = "8bxi1t(&hrcGQg2sd*"
# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("Led is on")
    else :
        print ("Led is off")
    #print(cmd)
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 an event of type
"greeting" 10 times
deviceCli.connect()
while True:
        #Get Sensor Data from DHT11
        temp=random.randint(0,100)
        Humid=random.randint(0,100)
        data = { 'temp' : temp, 'Humid': Humid }
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % Humid, "to IBM
Watson")
        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
```

```
        if not success:
            print("Not connected to IoTF")
        time.sleep(1)
        deviceCli.commandCallback = myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

OUTPUT:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:\Users\ARUN\Downloads\ibmiotpublishsubscribe.py =========
2022-11-19 10:08:32,297   ibmiotf.device.Client      INFO    Connected successfully: d:bafc8x:Smart-Fa
rmer:Smart-farmer
Published Temperature = 16 C Humidity = 63 % to IBM Watson
Published Temperature = 100 C Humidity = 9 % to IBM Watson
Published Temperature = 96 C Humidity = 4 % to IBM Watson
Published Temperature = 0 C Humidity = 26 % to IBM Watson
Published Temperature = 62 C Humidity = 43 % to IBM Watson
Published Temperature = 78 C Humidity = 96 % to IBM Watson
Published Temperature = 62 C Humidity = 22 % to IBM Watson
Published Temperature = 99 C Humidity = 10 % to IBM Watson
Published Temperature = 0 C Humidity = 39 % to IBM Watson
Published Temperature = 59 C Humidity = 65 % to IBM Watson
Published Temperature = 32 C Humidity = 32 % to IBM Watson
Published Temperature = 3 C Humidity = 16 % to IBM Watson
Published Temperature = 23 C Humidity = 6 % to IBM Watson
Published Temperature = 32 C Humidity = 17 % to IBM Watson
Published Temperature = 43 C Humidity = 65 % to IBM Watson
Published Temperature = 21 C Humidity = 51 % to IBM Watson
Published Temperature = 23 C Humidity = 62 % to IBM Watson
Published Temperature = 19 C Humidity = 30 % to IBM Watson
Published Temperature = 91 C Humidity = 51 % to IBM Watson
```

GitHub :

https://github.com/IBM-EPBL/IBM-Project-13760-1659529400

Project Demo Link:

https://photos.google.com/photo/AF1QipNkoKOkbdCOOGNetcSaQeFnq3H9N2SlCywJVSaO