Smart Farmer – IoT Enabled Smart Farming Application

Project Report

Submitted by

Team ID: **PNT2022TMID25993**

Team Leader - **Habib ur Rahman Z** (2019PITEC191)

Team Member - **Mohamed Irshath M** (2019PITEC216)

Team Member - **Prathap B** (2019PITEC221)

Team Member - **Logeshwaran T** (2019PITEC207)

CONTENT

1. INTRODUCTION

1.1 Project Overview:

Smart Farmer – IoT Enabled Smart Farming Application is an Internet enabled smart farming solution which enables the farmers and other person in the field of agriculture to monitor the crops, to control motors and other kind of activities remotely.

1.2 Purpose:

IoT-based agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature and humidity using some sensor. Farmer can monitor all the sensor parameters by using web or mobile application even if the farmer is not near his field. Watering the crops 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 sense parameters and controlling the motor pumps from the mobile application itself.

2. Literature Survey

2.1 Existing Problems:

Some of the problems exist in agriculture is lack of information for 24 hours this makes the farmers to monitor the crops for certain intervals. Other problems that exist is the young generation cannot able to get into agriculture this makes the low count in yielders and more count in consumers.

2.2 Reference:

Bahga, A., & Madisetti, V. (2014). Internet of Things: A Hands-on Approach. VPT. https://www.arduino.cc/en/Main/arduinoBoardMega2560 (Accessed on April 25, 2016)

Lee, M., Hwang, J., & Yoe, H. (2013, December). Agricultural Production System Based on IoT. In Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on (pp. 833-837). IEEE.

Suo, H., Wan, J., Zou, C., & Liu, J. (2012, March). Security in the internet of things: a review. In Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on (Vol. 3, pp. 648-651). IEEE.

2.3 Problem Statement Definitions:

Some of problem statement are

1. IoT enabled micro irrigation and farming land health logging system

History-based soil health parameters like soil moisture, pH level, temperature etc. are very essential of organic cultivation. IoT applications may assist in controlling the irrigation pump, opening and closing water flowing gates and also data logging the soil health conditions for present and future purpose. Further, with the help of IoT applications, provision for live guidance based on stored date of soil health from professional/experts to farmers in remote locations may be made available.

2. Digital Farming

Develop affordable app-based solution for Soil health monitoring and suggest which crop to be sown based on it. Create app-based solution to detect soil parameters like moisture content, temperature, relative humidity, nutrient, Ph, CEC, NPK etc. and provide crop suggestions to be produced based on soil parameters & environment values. Currently farmers follow Traditional Crop yielding pattern and irrespective of soil condition, farmers take routine crops. Farmers irrespective of whether soil nutrient requirement uses blanket fertilizers for crop. Because of these issues, losses in crop yielding and soil health gets affected. With the help of solution, farmer can plan which crop to take based on soil condition and plan quickly possible remedies for soil deficiencies.

3. Ideation and Proposed Solution:

3.1 Empathy Map Canvas

It comes in taking a picture before getting into actions. Empathy map canvas consist of following questions which are taking into picture. Some of them are

What do they THINK AND FEEL?

- Helps in handling many crops
- Remote Monitoring
- Cost Management

What do they HEAR?

- Easy to use
- Provides Accurate Results
- Time Management

What do they SEE?

- Current Temperature

- Weather Forecasting
- System Connection Status

What do they SAY AND DO?

- Hard in understanding metrics
- Increase Screen-ON time
- Suggest next steps

PAIN:

- Internet Availability
- Adopting new technology
- Sensor may get damaged

GAIN:

- Reduced consumables
- Increased yields
- Quality Improvement

3.2 Ideation and Brainstorming:

Brainstorming or ideation is part of the planning phase of the writing process. In Smart Farming we brainstormed some of the ideas

1. Developing a smart irrigation system that utilizes weather data to optimize watering schedules for crops.

2. Designing a mobile app for small farmers and hobbyists to sell their produce directly to consumers.

3. Implementing precision agriculture techniques to reduce water and fertilizer usage while maximizing crop yields.

4. Investigating the use of drones or other robotic technologies for crop mapping and yield analysis.

5. Creating a sensor system to monitor soil moisture levels, temperature, and other conditions in order to optimize growing conditions.

6. Developing a system to automatically identify and isolate pests and diseases in crops, then provide information on how to address them.

7. Investigating the use of alternative energy sources such as solar or wind power to run farm machinery or provide electricity for on-farm buildings.

8. designing a program that assists farmers in choosing the most efficient agricultural production strategies based on specific conditions and acreage.

9. Establishing a database of information on best practices for sustainable and profitable farming, which can be accessed by farmers worldwide.

10. Conducting research on new and innovative farming technologies and practices that can be adopted by farmers to improve efficiency and yields

3.3 Proposed Solutions:

1. Automated machine learning for crop monitoring: Automated machine learning can be used to build models that can detect, diagnose, and predict problems with crops. This can help farmers to identify problems early and take corrective action.

2. Predictive analytics for irrigation: Predictive analytics can be used to optimize irrigation schedules. This can help farmers to save water and energy, while also improving crop yields.

3. Soil moisture sensors: Soil moisture sensors can be used to monitor soil moisture levels. This information can be used to optimize irrigation schedules and improve crop yields.

4. Yield mapping: Yield mapping can be used to track and optimize crop yields. This information can be used to make decisions about seed selection, planting density, and fertilizer application.

5. Remote sensing: Remote sensing can be used to monitor crop growth and condition. This information can be used to make decisions about irrigation, pest management, and crop harvesting

3.4 Problem Solution Fit:

Problem solution fit describes the problem faces by the customer and how to overcome the solution. The respective analysis and solutions are documented an attached to the respective GIT repository in visual format.

4. Requirement Analysis

4.1 Functional Requirement:

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 Form Registration through Gmail Registration through LinkedIN

FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Verifying Email	Verification Email is sent to respective Email IDs for verification.
FR-4	Authentication	Using of Biometrics or PIN Authentication to perform some sensitive actions on the app.
FR-5	Exporting Reports	Allowing users to export their yearly, monthly, weekly stats.
FR-6	Sharing of data to third-party application	Allowing users to share some data via whatsapp and other medium.

4.2 Non-Functional Requirement:

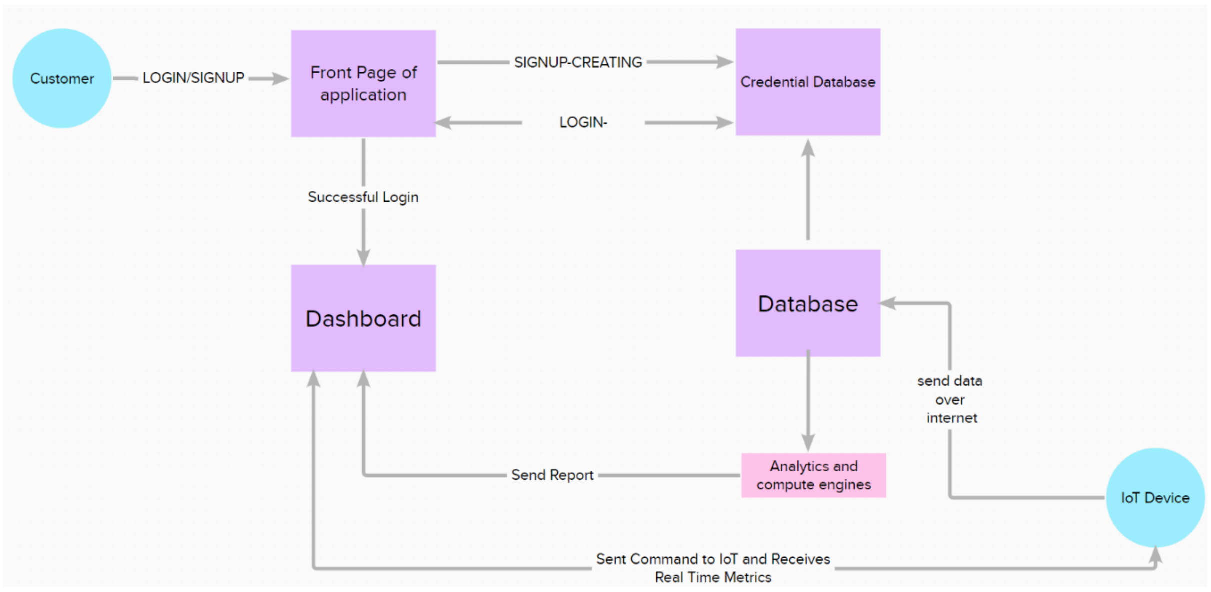
Following are the non-functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Dashboard must be simple, clean and customizable.
NFR-2	**Security**	Using two factor authentication and maintaining session period for some actions.
NFR-3	**Reliability**	Ensure the code is well and good before making it to production.
NFR-4	**Performance**	Writing a efficient code to give better performance to the low end devices too.
NFR-5	**Availability**	Deploying the application in two or more Availability Zones to ensure the availability SLA of 99.999%
NFR-6	**Scalability**	Using auto scalable services in cloud for database, compute, etc….

5. Project Design

5.1 Data Flow Diagram:

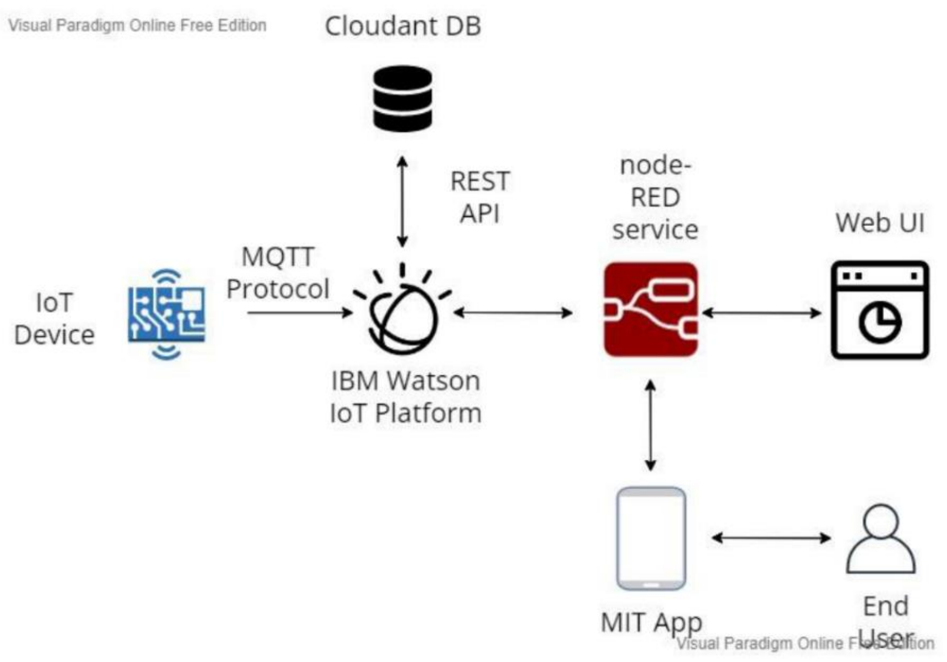
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



The above diagram shows how the data is handled from the device to the app and vice versa.

5.2 Solution and Technical Architecture:

Solution architecture shows the real-time data transmitted and received between devices are graphical represented in easier manner by using the components without driving deep into the components. Its shows the rough overview of the backend of the application.



5.2 Technical Architecture:

Technical Architecture shows the graphical representation of flows and integration of two or more components. It also suggests the protocol used, handling servers and other security aspects of the application.

The below architecture diagram represents the actual architecture used in the smart farming application. This architecture is entirely built on and for IBM Cloud platform the services used here are IBM Managed Services and Open Source Software.

Major Components are:

- Node-RED
- Cloudant DB
- IBM Watson IoT

Node-RED:

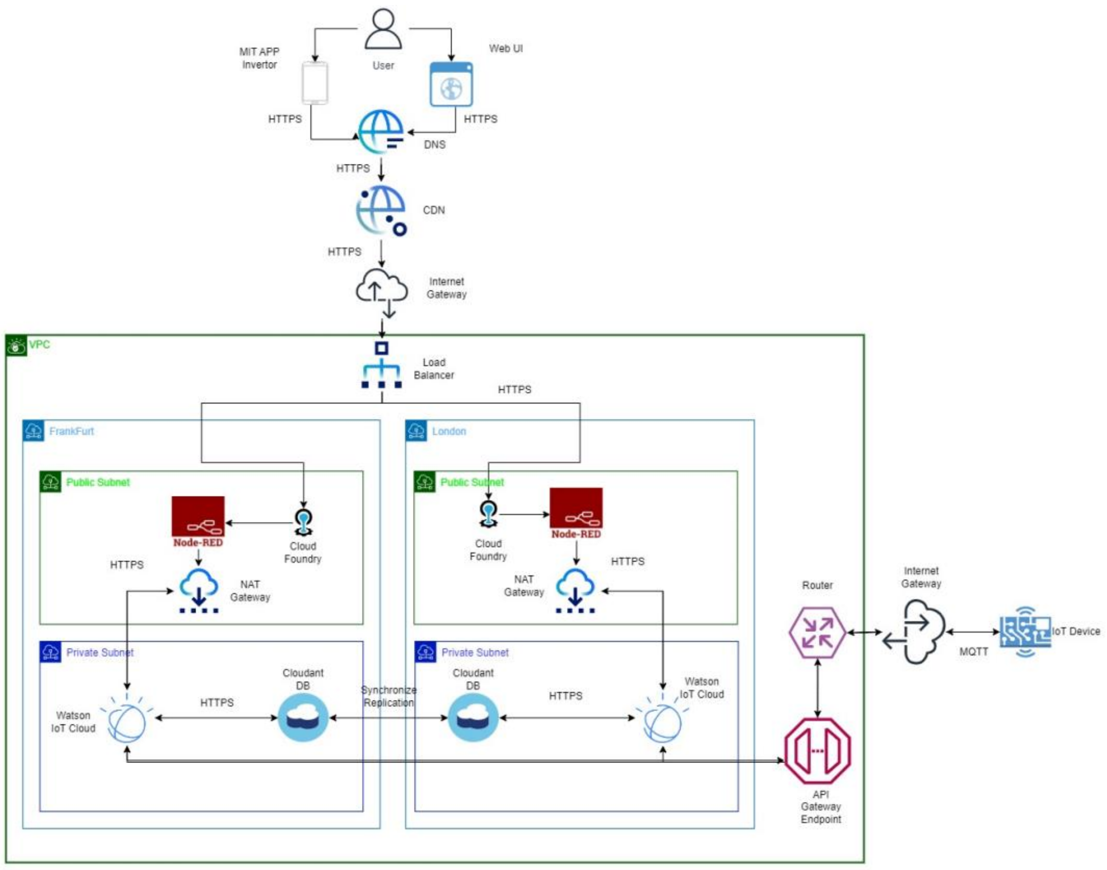
It the server-side scripting application which manages the WebUI and Request handing and pushing request.

Cloudant DB:

It is complete IBM Managed Database. It stores the information in the form of tables and JSON.

IBM Watson IoT:

IBM Native IoT Platform is Watson IoT it manages the IoT devices and also simulate the following devices.



5.3 User Stories:

Use the below table lists all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login
		USN-4	As a user, I can register for the application through Gmail	I can get registered in one tap
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login to the application

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**
	Dashboard	USN-6	As a user, I must watch my metrics displayed right in the front page of the application	I can may analysis from the dashboard itself
Customer (Web user)	Registration	USN-7	As a web user, I can fill the form right in the front page of the application	I can get registered without clicking and navigating to other page
Customer Care Executive	Dashboard	USN-8	As a customer care executive, I must provide with the text box to get the user information using unique no. for every individual user	I can get the user stats easily
	Chat Interface	USN-9	As a customer care executive, I must be indicated with the typing… indication	I can find whether the user on the other end is active or not
Administrator	Dashboard	USN-10	As an admin, I must be provided with stats of the instance running behind	I can figure out the health of the application
	Adding new employee	USN-11	As an admin, I must be provided with the option to add new employee	I can give the respective credential for the employee

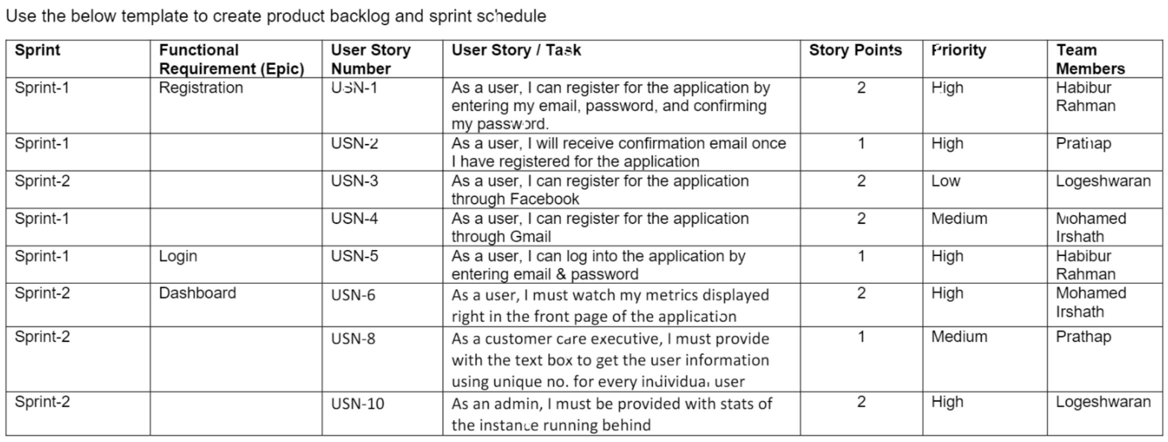
6. Project Planning and Scheduling

6.1 Sprint Planning and Execution:

This gives the overall planning task, execution and timeline of each sprint that has been carried during application creating process.

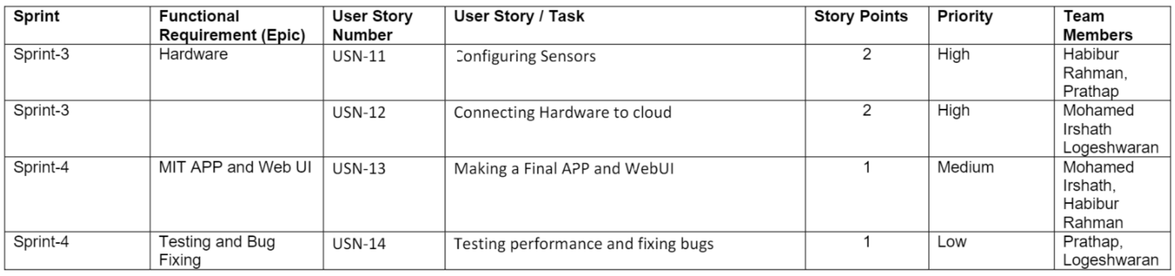
In the application creation process we are scheduling 4 sprints at a duration of 1 week each.

Each sprint has designed based on the epic. Epic consist of login page, dashboard, hardware configuration and finally testing.



Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Habibur Rahman
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Prathap
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	Logeshwaran
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Mohamed Irshath
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	Habibur Rahman
Sprint-2	Dashboard	USN-6	As a user, I must watch my metrics displayed right in the front page of the application	2	High	Mohamed Irshath
Sprint-2		USN-8	As a customer care executive, I must provide with the text box to get the user information using unique no. for every individual user	1	Medium	Prathap
Sprint-2		USN-10	As an admin, I must be provided with stats of the instance running behind	2	High	Logeshwaran



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Hardware	USN-11	Configuring Sensors	2	High	Habibur Rahman, Prathap
Sprint-3		USN-12	Connecting Hardware to cloud	2	High	Mohamed Irshath Logeshwaran
Sprint-4	MIT APP and Web UI	USN-13	Making a Final APP and WebUI	1	Medium	Mohamed Irshath, Habibur Rahman
Sprint-4	Testing and Bug Fixing	USN-14	Testing performance and fixing bugs	1	Low	Prathap, Logeshwaran

6.2 Sprint Delivery Schedule:

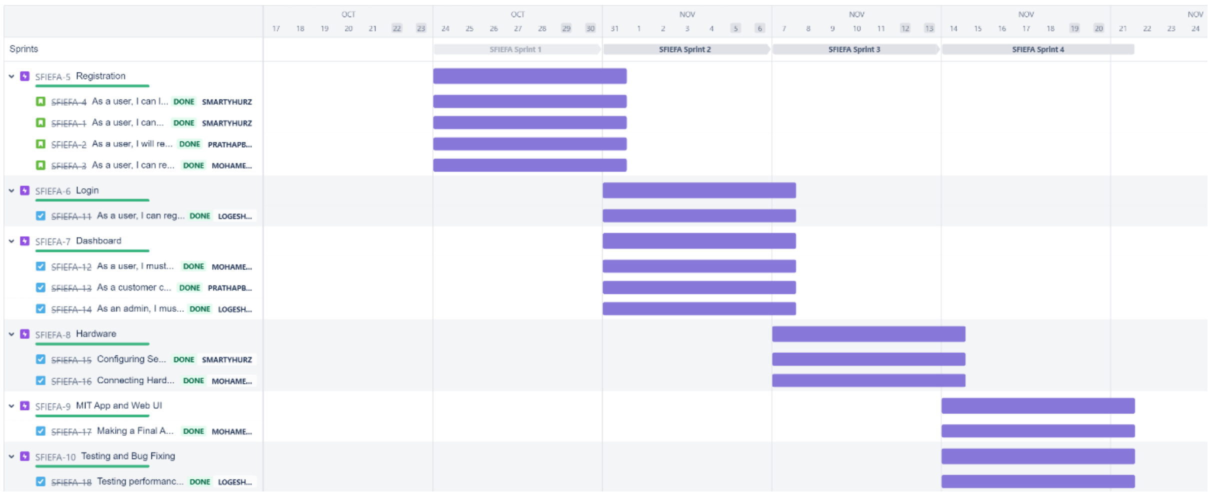
Below table suggest the sprint schedule

Sprint	Task	Commencing Date	Deadline
Sprint - 1	Creating Web UI and MIT App.	24 Oct 2022	29 Oct 2022
Sprint - 2	Hardware Configuration	31 Oct 2022	06 Nov 2022
Sprint - 3	Configuring IBM Watson Cloud	07 Nov 2022	14 Nov 2022
Sprint - 4	Testing and Bug Fixing	14 Nov 2022	19 Nov 2022

6.3 Report From JIRA:

During the sprint process JIRA Software from Atlassian is used as management tool. Using the application, the report is generated and timeline is created.

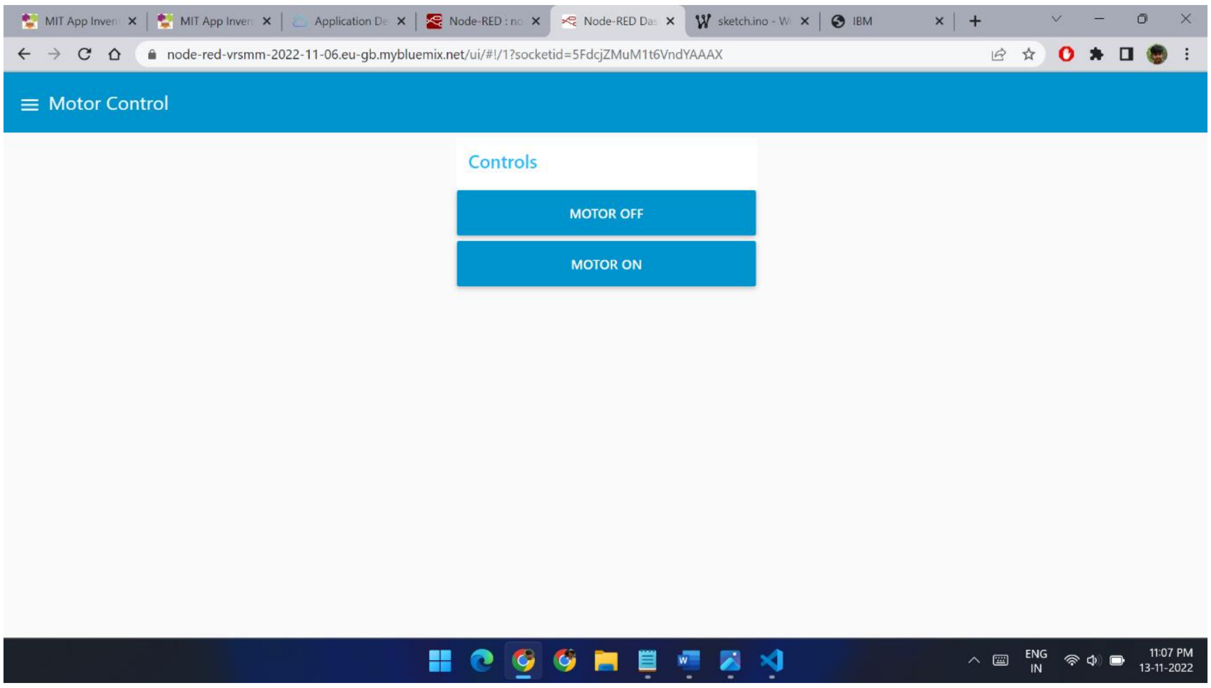
The report summary and timeline are attached below.



7. Coding and Solutioning:

7.1 Feature 1:

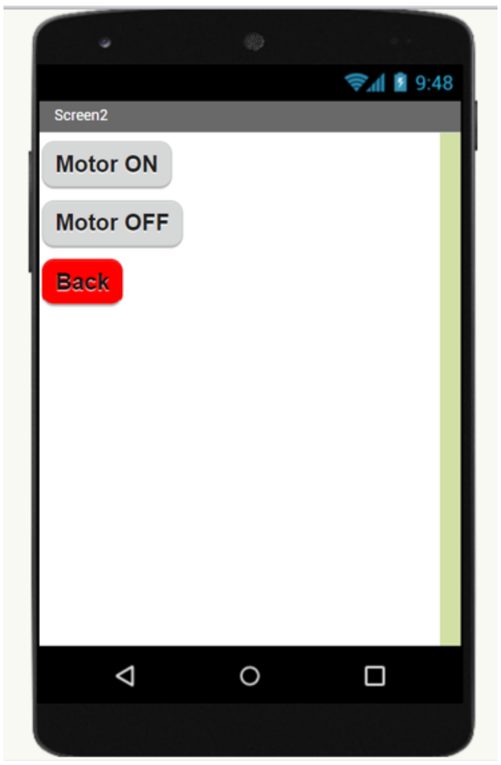
In Web UI Separate home layout has been added to display buttons on to the web.


node-red-vrsmm-2022-11-06.eu-gb.mybluemix.net/ui/#!/1?socketid=5FdcjZMuM1t6VndYAAAX
Motor Control
Controls
MOTOR OFF
MOTOR ON
11:07 PM
13-11-2022

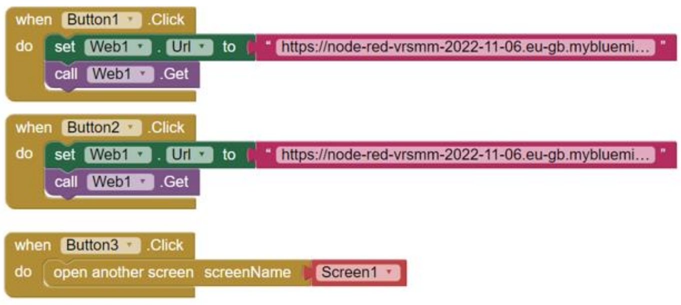
7.2 Feature 2:

Same as web UI Separate screen is used for displaying motor controls on the MIT application.

Screen 2:



Code Block for Screen 2:



```
when Button1 .Click
do   set Web1 . Url to " https://node-red-vrsmm-2022-11-06.eu-gb.mybluemi... "
     call Web1 .Get

when Button2 .Click
do   set Web1 . Url to " https://node-red-vrsmm-2022-11-06.eu-gb.mybluemi... "
     call Web1 .Get

when Button3 .Click
do   open another screen screenName Screen1
```

-

Database Schema:

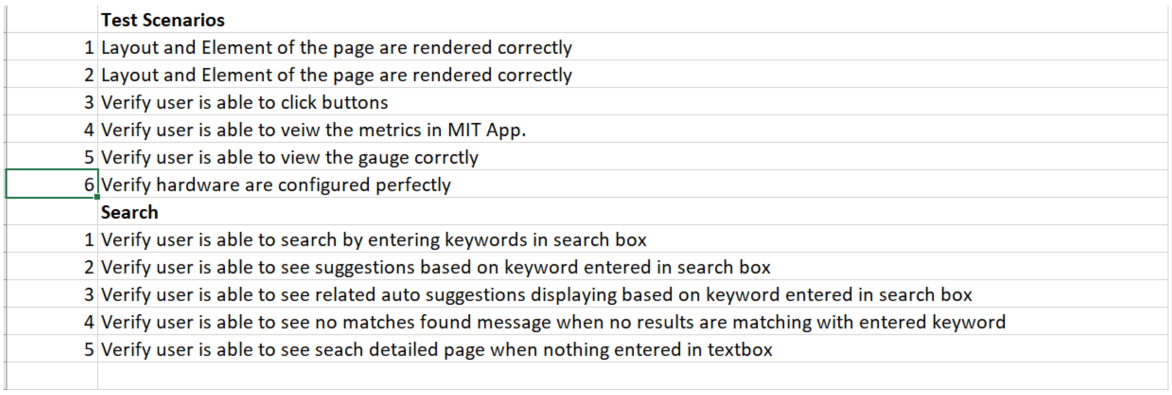
```
{
 "id": "043a92e731908d90360d415a6c
41041a",
 "key": "043a92e731908d90360d415a6
c41041a",
 "value": {
  "rev": "1-540c44f057da11ac62bb6e
c908f67d3a"
 },
 "doc": {
  "_id": "043a92e731908d90360d415a
6c41041a",
  "_rev": "1-540c44f057da11ac62bb6
ec908f67d3a",
  "topic": "iot-2/type/ESP32/id/12
3456789/evt/IoTSensor/fmt/json",
  "payload": {
   "temperature": 90,
   "humidity": 42,
   "soilmoisture": 20
  },
  "deviceId": "123456789",
  "deviceType": "ESP32",
  "eventType": "IoTSensor",
  "format": "json"
 }
}
```

Cloudant DB is a JSON Document store database. The schema for the database is attached here

8. Testing

8.1 Testcases:

Testcases that are taken into considerations and tested.



	Test Scenarios
1	Layout and Element of the page are rendered correctly
2	Layout and Element of the page are rendered correctly
3	Verify user is able to click buttons
4	Verify user is able to veiw the metrics in MIT App.
5	Verify user is able to view the gauge corrctly
6	Verify hardware are configured perfectly
	Search
1	Verify user is able to search by entering keywords in search box
2	Verify user is able to see suggestions based on keyword entered in search box
3	Verify user is able to see related auto suggestions displaying based on keyword entered in search box
4	Verify user is able to see no matches found message when no results are matching with entered keyword
5	Verify user is able to see seach detailed page when nothing entered in textbox

8.2 User Acceptance Testing:

Defect Analysis:

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	3	1	0	0	4
Duplicate	0	0	0	0	0
External	3	0	0	1	4
Fixed	5	0	0	0	5
Not Reproduced	0	0	0	0	0
Skipped	3	1	0	0	4
Won't Fix	4	3	0	1	8
Totals	18	5	0	2	25

Testcase Analysis:

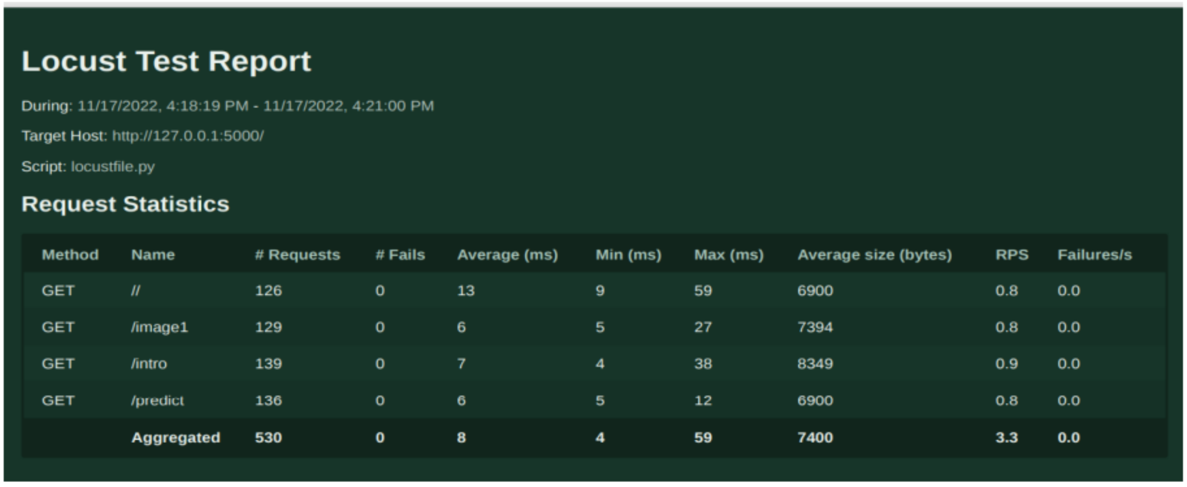
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	4	0	0	4
Client Application	5	0	0	5
Security	3	0	3	0

Outsource Shipping	0	0	0	0
Exception Reporting	1	0	1	0
Final Report Output	13	0	4	9
Version Control	2	0	0	2

9. Results

9.1 Performance Metrics:



Locust Test Report

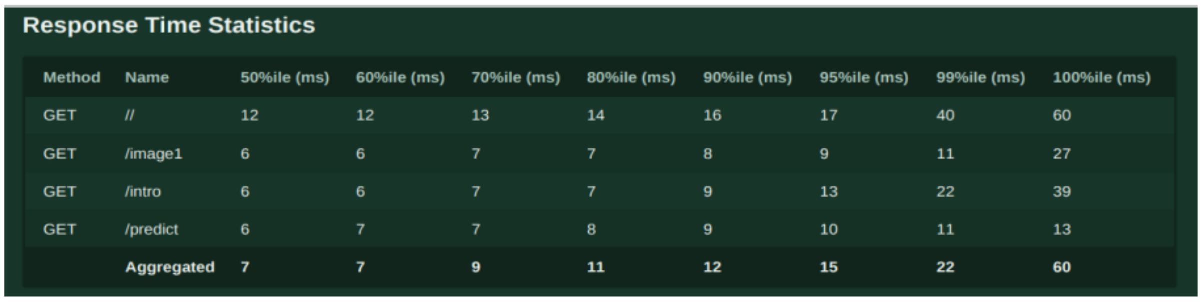
During: 11/17/2022, 4:18:19 PM - 11/17/2022, 4:21:00 PM

Target Host: http://127.0.0.1:5000/

Script: locustfile.py

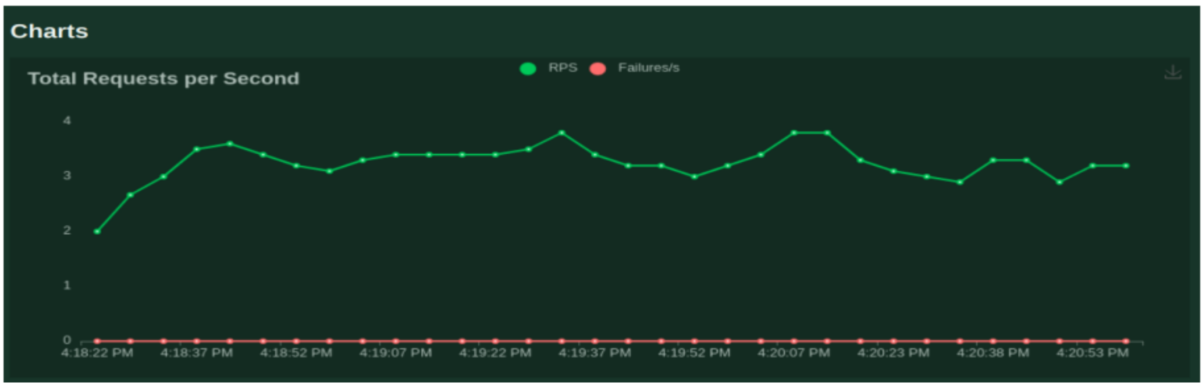
Request Statistics

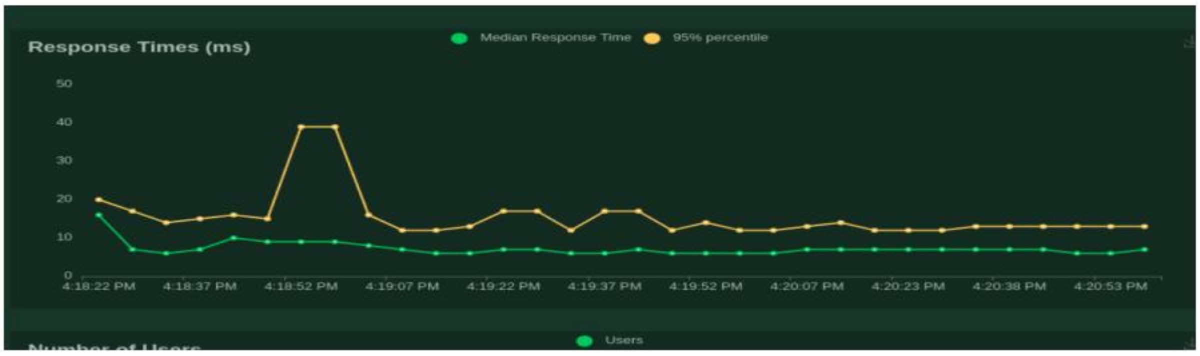
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	126	0	13	9	59	6900	0.8	0.0
GET	/image1	129	0	6	5	27	7394	0.8	0.0
GET	/intro	139	0	7	4	38	8349	0.9	0.0
GET	/predict	136	0	6	5	12	6900	0.8	0.0
	Aggregated	**530**	**0**	**8**	**4**	**59**	**7400**	**3.3**	**0.0**



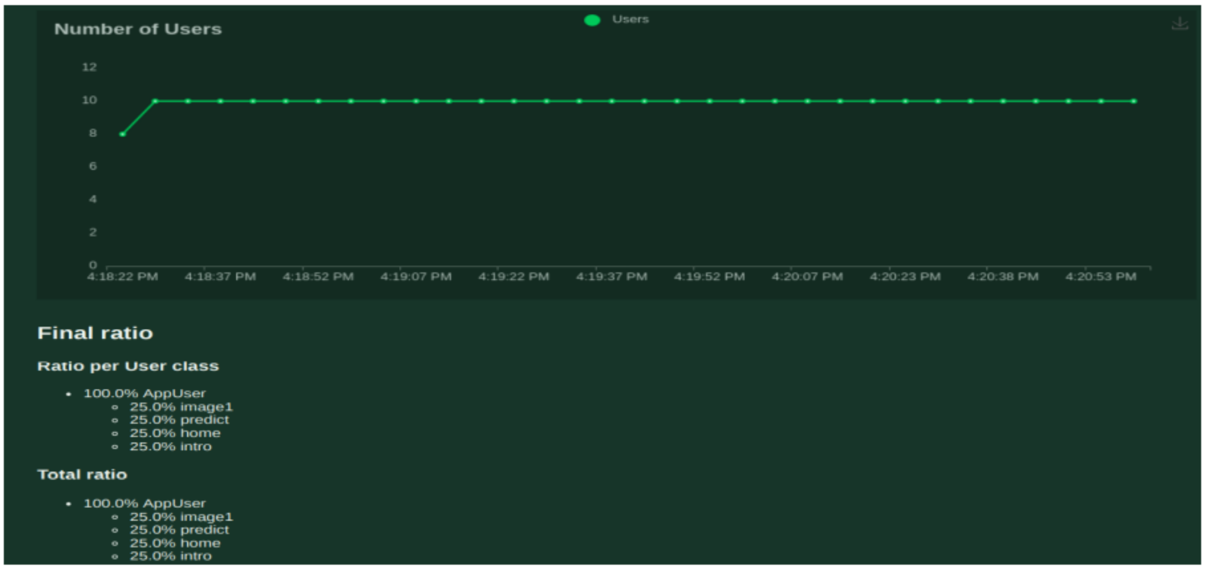
Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	12	12	13	14	16	17	40	60
GET	/image1	6	6	7	7	8	9	11	27
GET	/intro	6	6	7	7	9	13	22	39
GET	/predict	6	7	7	8	9	10	11	13
	Aggregated	**7**	**7**	**9**	**11**	**12**	**15**	**22**	**60**





9.2 Output:



Final ratio

Ratio per User class

- 100.0% AppUser
  - 25.0% image1
  - 25.0% predict
  - 25.0% home
  - 25.0% intro

Total ratio

- 100.0% AppUser
  - 25.0% image1
  - 25.0% predict
  - 25.0% home
  - 25.0% intro

10. Advantages and Disadvantages

Pros:

Smart agriculture is a term that describes an innovative approach to agriculture that uses information and communication technologies to improve agricultural production, profitability, sustainability, and food quality.

• Smart agriculture also refers to the use of information and communication technologies for the purposes of communication, such as information and data sharing, as well as for decision-making, data analysis and management

• Smart agriculture is a new energy-efficient agricultural technology that is being developed to help farmers, who often struggle with a lack of water and other resources, to produce more food with fewer resources. In some cases, farmers are able to grow their crops more efficiently, which is good for the environment.

• Additionally, farmers can plant their crops earlier in the season and produce them with less labour. On the other hand, in some regions, there is a decrease in food security

• Smart agriculture is the use of technology to manage the production of the agricultural production

• It uses sensors, analytics, and other technologies to improve the efficiency of the agricultural process. This enables producers to increase production, reduce costs and protect the environment.

Smart agriculture is, for example, able to take advantage of the latest weather forecasts, which can help to schedule planting, irrigation and harvesting

Cons:

One of the big advantages of smart agriculture is that it uses data to make decisions. The downside of smart agriculture is that it is not a perfect system.

• Smart agriculture has its disadvantages in the following areas: increased use of chemicals, uneven water distribution, reliance on organic fertilizers, and increased food miles.

• Smart agriculture system is not very flexible. It is also not very user-friendly. The system is also not very effective and is therefore not efficient. One of the major disadvantages is that smart agriculture can also have negative environmental impacts.

• Smart agriculture is not smart when it comes to environmental impact. One of the biggest disadvantages of using technology to help farmers is that it is not sustainable. It takes a lot of time and resources to implement the technology. - Another disadvantage is that if the technology fails, there are no more options for farmers to implement the technology.

The main disadvantage is the waste of time and resources that it takes to implement the technology. In order to make a smart agriculture system, It takes a lot of time and money.

11. Conclusion

Smart farming is a term used to describe the use of technology in agriculture to improve the efficiency and productivity of farms. Smart farming includes the use of sensors, drones, and other technology to collect data about crops and livestock, as well as the use of data analytics to make decisions about planting, irrigation, and other farm management activities.

12. Future Scope

There is a lot of scope for smart farming agriculture in the future. With the help of technology, farmers will be able to increase their productivity and yield. They will also be able to reduce the use of pesticides and fertilizers.

13. Appendix

Python Code:

```python
import sys
import random
import time
import ibmiotf.application
import ibmiotf.device

#IBM Cloud Credentials
organization = "jtp3hb"
deviceType = "ESP32"
deviceId = "123456789"
authMethod = "token"
authToken = "1234567890"

def myCommandCallback(cmd):
    print("Command Recived: %s" % cmd.data['command'])
    status = cmd.data['command']
    if status == "motoron":
        print("Motor is ON")
    else:
        print("Motor is OFF")

#Try and Except Statement for connecting cloud
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()

#Device CLI Connectivity
deviceCli.connect()


#Sensing and Alerting Cloud
while True:
    humidity = random.randint(0,100)
    temperature = random.randint(0,100)
    soilmoisture = random.randint(0,1000)

    data = {'temperature': temperature, 'humidity' : humidity, 'soilmoisture'
: soilmoisture}
    def myOnPublishCallback():
        print("Published Metrics", humidity, temperature, soilmoisture)
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
    if not success:
        print ("Not Connected to IoT")

    time.sleep(1)
    deviceCli.commandCallback = myCommandCallback

deviceCli.disconnect()
```

C++ Code:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#include <ESP32Servo.h>
#define DHTPIN 4     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define SERVOO 5
DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of
dht connected

Servo servo;

void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);


#define ORG "jtp3hb"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "123456789"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "1234567890"     //Token
String data3;
float h, t;
int sa = 56;

//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of
event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd  REPRESENT command
type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the
predefined client id by passing parameter like server id,portand
wificredential
```

```
void setup()// configureing the ESP32
{
  Serial.begin(115200);
  dht.begin();
  delay(10);
  Serial.println();
  servo.attach(SERVOO, 500, 2400);
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{

  h = dht.readHumidity();
  t = dht.readTemperature();

  PublishData(t, h);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }

}
void PublishData(float temp, float humid) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"temperature\":";
  payload += temp;
  payload += "," "\"humidity\":";
  payload += humid;
  payload += "," "\"soilmoisture\":";
  payload += sa;
  payload += "}";


  Serial.print("Sending payload: ");
  Serial.println(payload);


  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud
then it will print publish ok in Serial monitor or else it will print publish
failed
  } else {
    Serial.println("Publish failed");
  }

}
void mqttconnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }

     initManagedDevice();
     Serial.println();
  }
}
void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish
the connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{

  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic);
  for (int i = 0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]);
    data3 += (char)payload[i];
  }

  Serial.println("data: "+ data3);
  if(data3=="motoron")
  {
    int pos = 0;
    servo.write(pos);

  }

  else
  {
    int pos = 180;
    servo.write(pos);

  }
data3="";


}
```

GIT Repository:

https://github.com/IBM-EPBL/IBM-Project-2983-1658490672

Demo Video Link:

https://youtu.be/8iCSno5ij2k