PROJECT

SmartFarmer - IoT Enabled Smart Farming Application

DONE BY

TEAM ID: PNT2022TMID54115

SUSHMITHA.S(111919106060)

DURGA.D(111919106011)

HARIHARASUDHAN J(111919106018)

MUKUNDHAN JJ(11191906040)

Project Report

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
   - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams
   - 5.2 Solution & Technical Architecture
   - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 Sprint Planning & Estimation
   - 6.2 Sprint Delivery Schedule
   - 6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   - 7.1 Feature 1
   - 7.2 Feature 2
   - 7.3 Database Schema (if Applicable)
8. **TESTING**
   - 8.1 Test Cases

8.2 User Acceptance Testing

9. **RESULTS**

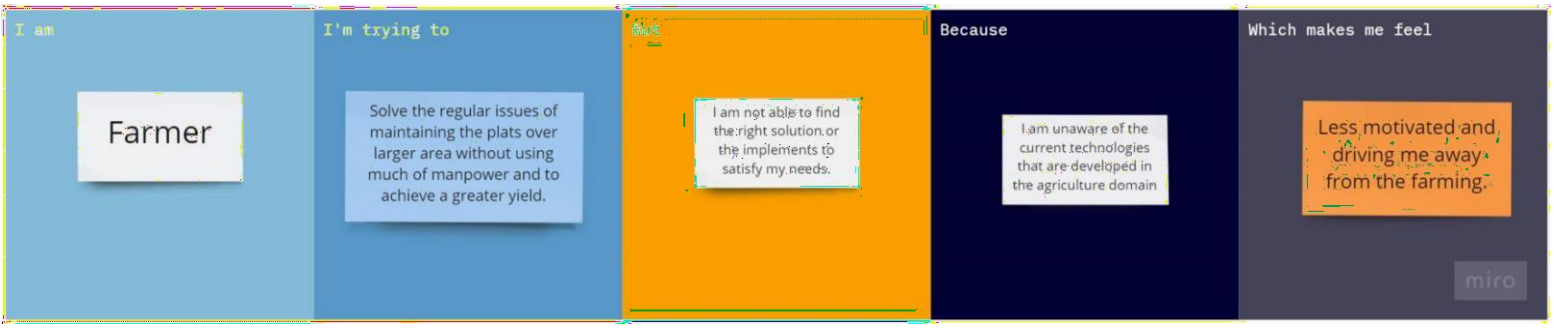
9.1 Performance Metrics

10. **ADVANTAGES & DISADVANTAGES**

11. **CONCLUSION**

12. **FUTURE SCOPE** 13. **APPENDIX** Source Code

GitHub & Project Demo Link



1. Introduction

Problem Statement - SmartFarmer : IoT Enabled Smart Farming Application

What is a problem statement - Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

Img 1.1

The above image 1.1 defines the problem statement of what the customer is undergoing and what solution I could provide for the above stated problem statement.

1.1 Project Overview

Agriculture, the major sector which defines the growth of the nation and any community that exists on this earth. It requires very intensive care, discipline and patience to reap the yield. In this evolving modern world the number of people who adopt agriculture are only a handful, in the next few years there will be very minimal people who do farming.

This project will help the farmers to grow crops in a whole new way so that they can reap good yield and healthy foods. This system does the following:

1. Autonomous crop monitoring.
2. Irrigation control.
3. Environment sensing.

1.2 Purpose

The purpose of this project is to help farmers to grow crops better and have yield, it emanates the need to manually look for the soil condition and decide whether to water the plant or not, this process becomes very difficult when plants are maintained over a very large area. This project helps the farmers to find a better solution and make them aware of their surroundings.

2. Literature Survey

2.1 Existing Problem

Farmers must meet the challenging needs of our planet and the experiences and the expectations of the regulations, consumers and food produced. There are increasing pressure from climate change, soil erosion and biodiversity loss and from consumers changing tastes in food and concerns about how it is produced and the natural world that farming works with plants, pests and diseases.

Problems that farmers face:

1. Cope up with climate change, soil erosion and biodiversity.
2. Satisfy the customer needs.
3. Meet rising demands for more food of higher quality.
4. Invest in farm productivity.
5. Adopt and learn new technologies.

Reference 2	Title	Smart Farming: IoT Based Smart Sensors Agriculture Stick for Live Temperature and Moisture Monitoring using Arduino, Cloud

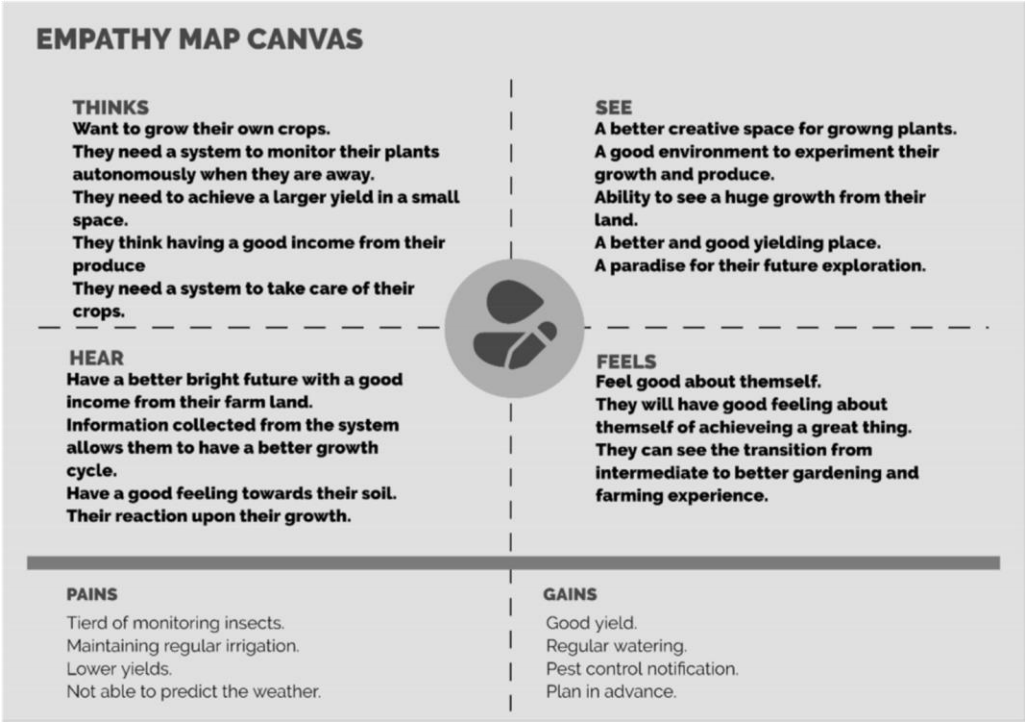
2.2 References

Reference 1	Title	IOT Based Smart Farming for Effective Utilization of Water and Energy
	Authors	K.N.Sivabalan
		V.Anandkumar
		S.Balakrishnan
		Computing & Solar Technology
	Authors	Anand Nayyar (*Assistant Professor, Department of Computer Applications & IT KCL Institute of Management and Technology, Jalandhar, Punjab*)
		Er. Vikram Puri (*M.Tech(ECE) Student, G.N.D.U Regional Center, Ladhewali Campus, Jalandhar*)

3. IDEATION & PROPOSED SOLUTION

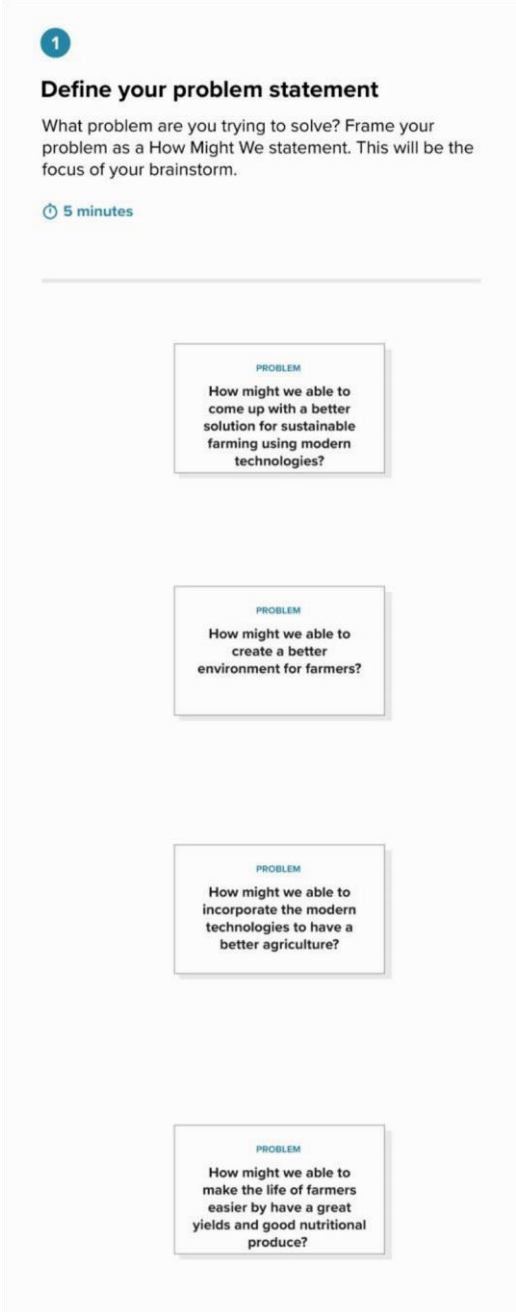
3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. **Empathy Map:**

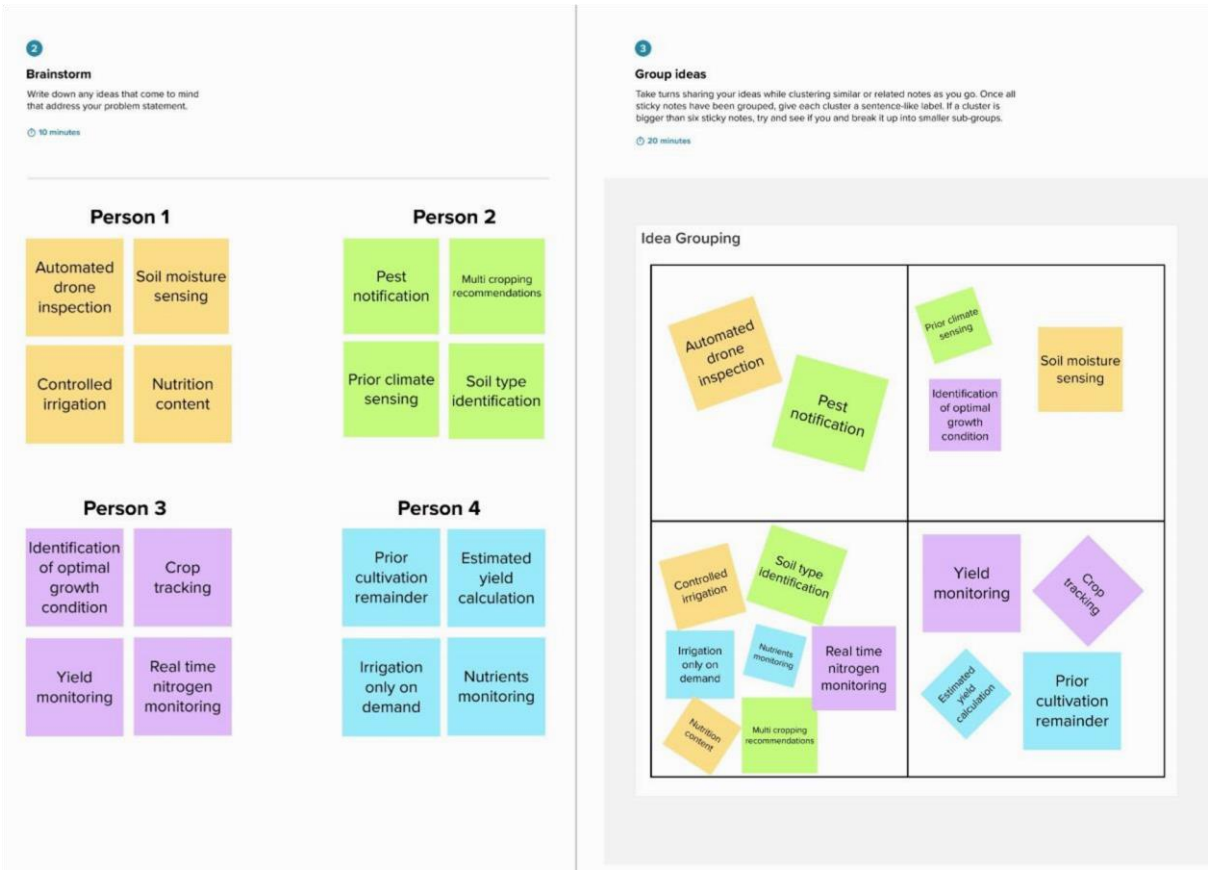


3.2 Ideation & Brainstorming:

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

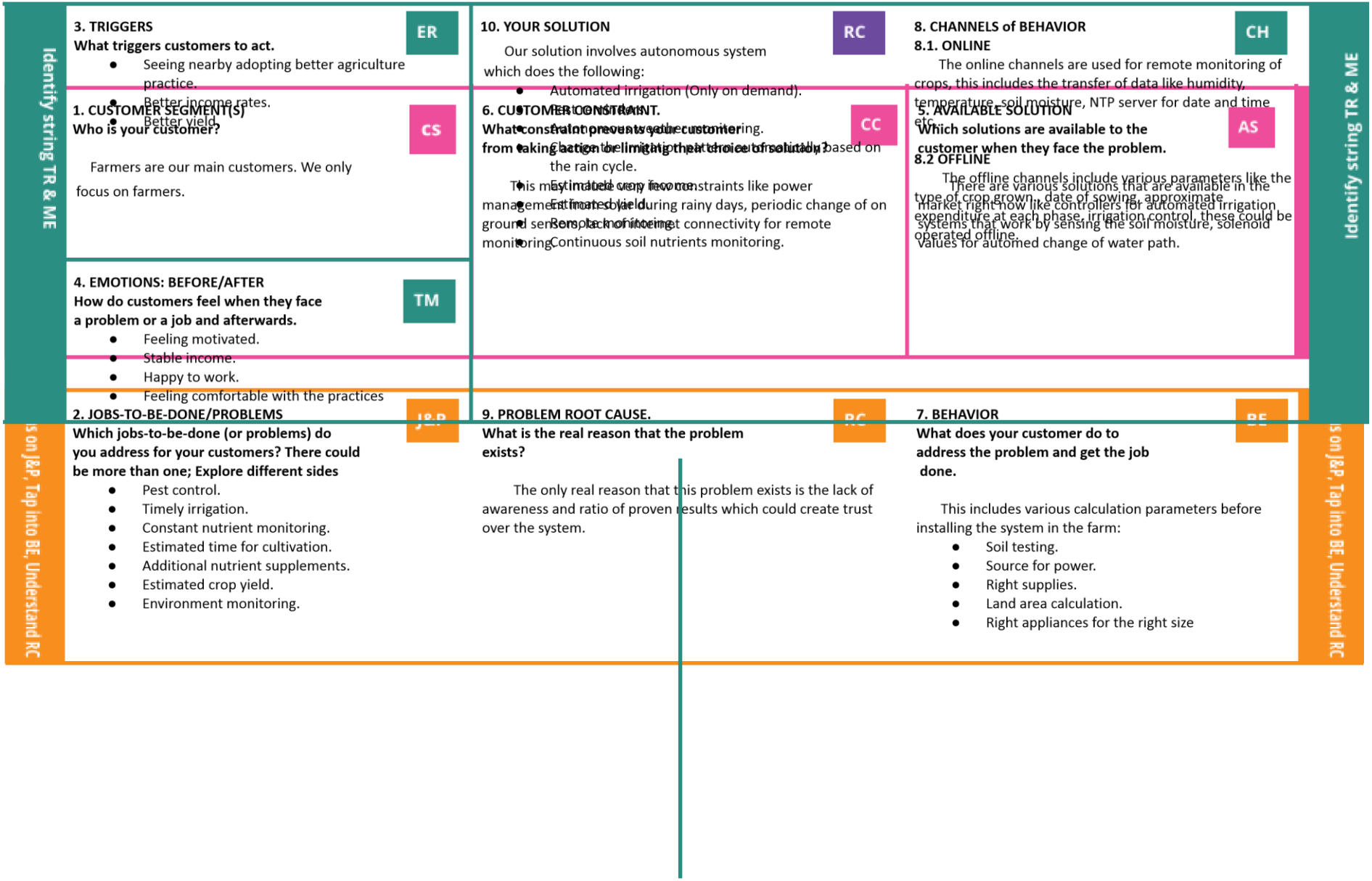


Step-3: Idea Prioritization

3.3 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	IoT-based agriculture systems help the farmer in monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors.
2.	Idea / Solution description	Our system comprises the following elements to come up with a solution: 1. Pest control. 2. Timely irrigation. 3. Constant nutrient monitoring. 4. Estimated time for cultivation. 5. Additional nutrient supplements. 6. Estimated crop yield. 7. Environment monitoring.
3.	Novelty / Uniqueness	Our system could function in both solar and battery mode. The inbuilt battery delivers power during the necessary times. It also delivers remote sensing facilities.
4.	Social Impact / Customer Satisfaction	Upon implementing customers feel : 1. Seeing nearby adopting better agriculture practice. 2. Better income rates. 3. Better yield. 4. Feeling motivated. 5. Stable income. 6. Happy to work. 7. Feeling comfortable with the practices

5.	Business Model (Revenue Model)	Our system comprises of hardware and software part: Hardware: 1. Controller(Brain) - 8000 2. Solenoid valves - 5000/piece 3. Pipe materials - needed to be provided by the land owner(May vary from place to place). 4. Cloud storage of data - 10000/Month(For n customers can be scaled up on demand) Roughly sums around - 25000
		Additionally we can generate income by increasing the number of controllers since it is limited to a specific area. Additional income - Ads posted on our mobile and web application platform
6.	Scalability of the Solution	This system of ours is like a lego which can be stacked and scaled up for a larger growth area.



3.4 Proposed Solution Fit

4. REQUIREMENT ANALYSIS:

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Account Creation	Users should create an account in the web application to order the product.
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Order Confirmation	Order is confirmed by a prompt message and redirected to the payment page.
FR-4	Payment	Payment can be made either through any online platform or can be paid at the time of delivery.
FR-5	Field Visit	A small field visit is made by the expert team to plan the product installation.
FR-6	Adding user data	A few data are added in the device before the installation to send sms and cloud storage.

4.2 Non-functional Requirements:

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

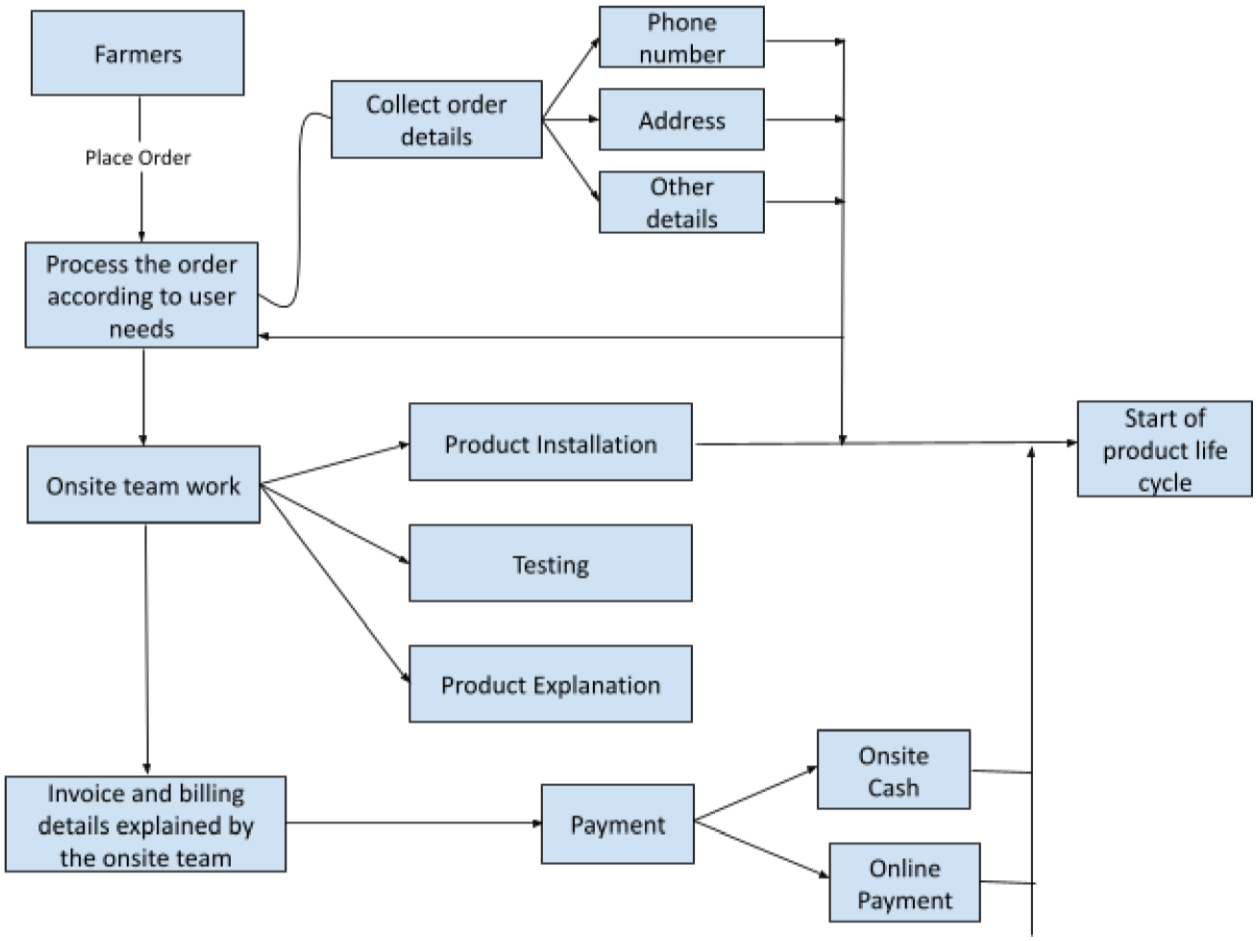
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Since the product is completely autonomous there is no need for extensive care.
NFR-2	**Security**	Since the has its own login credentials for both user and cloud login it makes it secure against data theft.

NFR-3	**Reliability**	The physical structure of the product is enclosed using waterproof and non corrosive materials it makes it reliable to use.
NFR-4	**Performance**	The performance of the device increases gradually based on the sensor data collected from the filed.
NFR-5	**Availability**	This system requires an active internet connection for better functioning and offsite control. But it can also work offline but users cannot monitor remotely.
NFR-6	**Scalability**	It can be scaled easily according to the area.

5. Project Design:

5.1 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.


Farmers
Place Order
Process the order according to user needs
Collect order details
Phone number
Address
Other details
Onsite team work
Product Installation
Testing
Product Explanation
Start of product life cycle
Invoice and billing details explained by the onsite team
Payment
Onsite Cash
Online Payment

5.2 Solution & Technical Architecture:

Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Example - Solution Architecture Diagram:

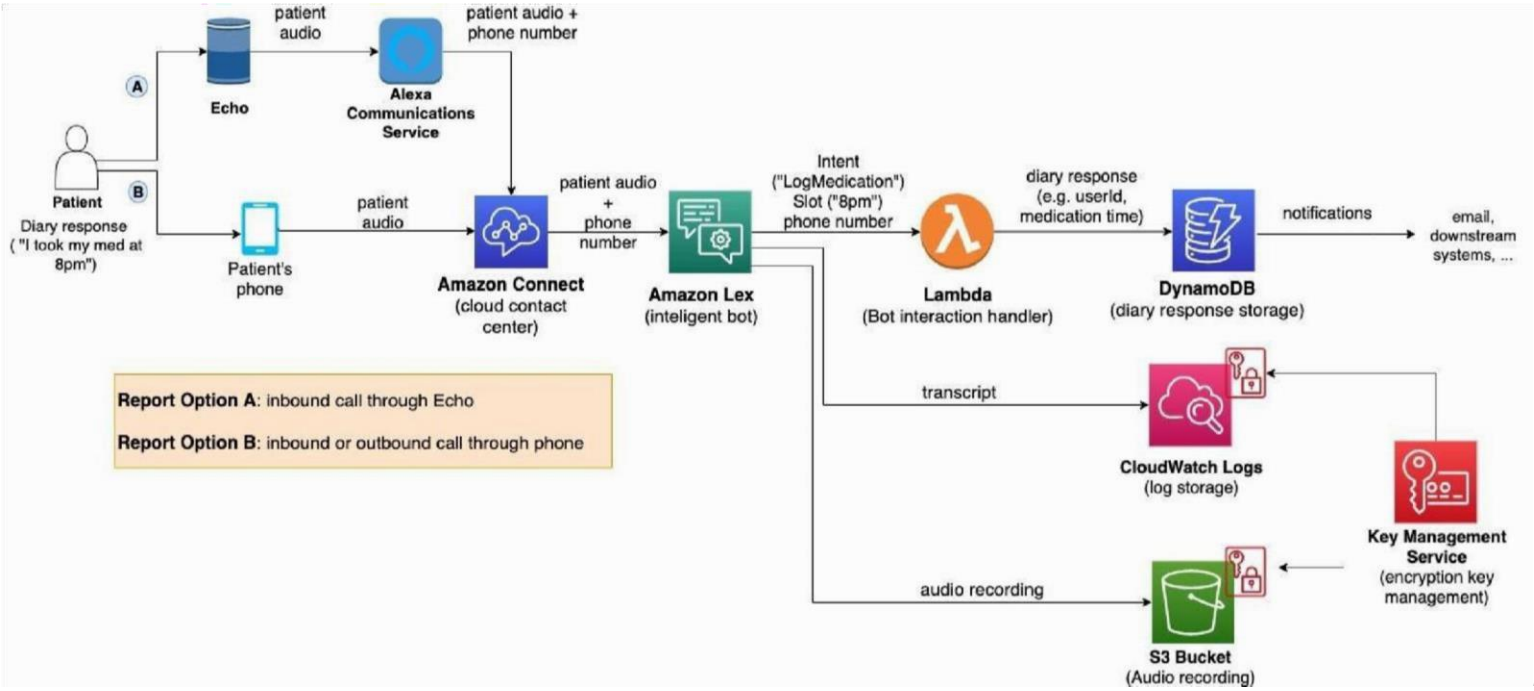


Figure 1: Architecture and data flow of the voice patient diary sample application

5.3 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
User	Registration	USN-1	As a farmer, they can register for the application by entering their email, password, and confirming the password.	I can access my account / dashboard	High	Sprint-2

	Login	USN-2	Product confirmation.	I will be receiving the confirmation email	High	Sprint-2
	Dashboard	USN-3	Product virtual demo through any online video sharing platform like Google Meet, Zoom or virtual tour. This is to show about the product which will be explained by our team.	I can register for the virtual demo or product tour.	Low	Sprint-2
	Dashboard	USN-4	Onsite inspection.	I can register for onsite inspection to have a pre planning of the product installation.	Medium	Sprint-4
	Dashboard	USN-5	Product tracking and payment.	I can track the shipment using the invoice number in the mail.	Low	Sprint-4
	Dashboard	USN-6	Payment confirmation.	Payment can be made either at the site by paying some advance during the ordering process or through online payment.	High	Sprint-2
Product Design	Working Hardware Demo	USN-7	Working model of the product.	The real time interface of the hardware.	High	Sprint-3
Product Care and Service	Web application	USN-8	Service or query	At times of product malfunction or in need of any assistance I can call the helpline	Medium	Sprint-2

6. PROJECT PLANNING & SCHEDULING:

6.1 Sprint Planning & Estimation

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	Chandru s
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Ramkumar T
Sprint-2	Hardware Development phase - 1	USN-3	Monitoring and displaying humidity and temperature.	2	Low	Sivaharish k
Sprint-2	Hardware Development phase - 2	USN-4	Connecting the device online(Configuring wifi automatically through a web portal)	2	High	Prabhakaran.A
Sprint-3	Hardware Development phase - 3	USN-5	Uploading data to the cloud.	1	High	Thaufiq ahmed.I
Sprint-4	Hardware Development phase - 4	USN-6	Establishing relay controls.	1	High	Chandru S
Sprint-4	Mobile Application	USN-7	Deployment of application	2	High	Ramkumar T

Velocity:

Project Tracker, Velocity & Burndown Chart: (4 Marks)

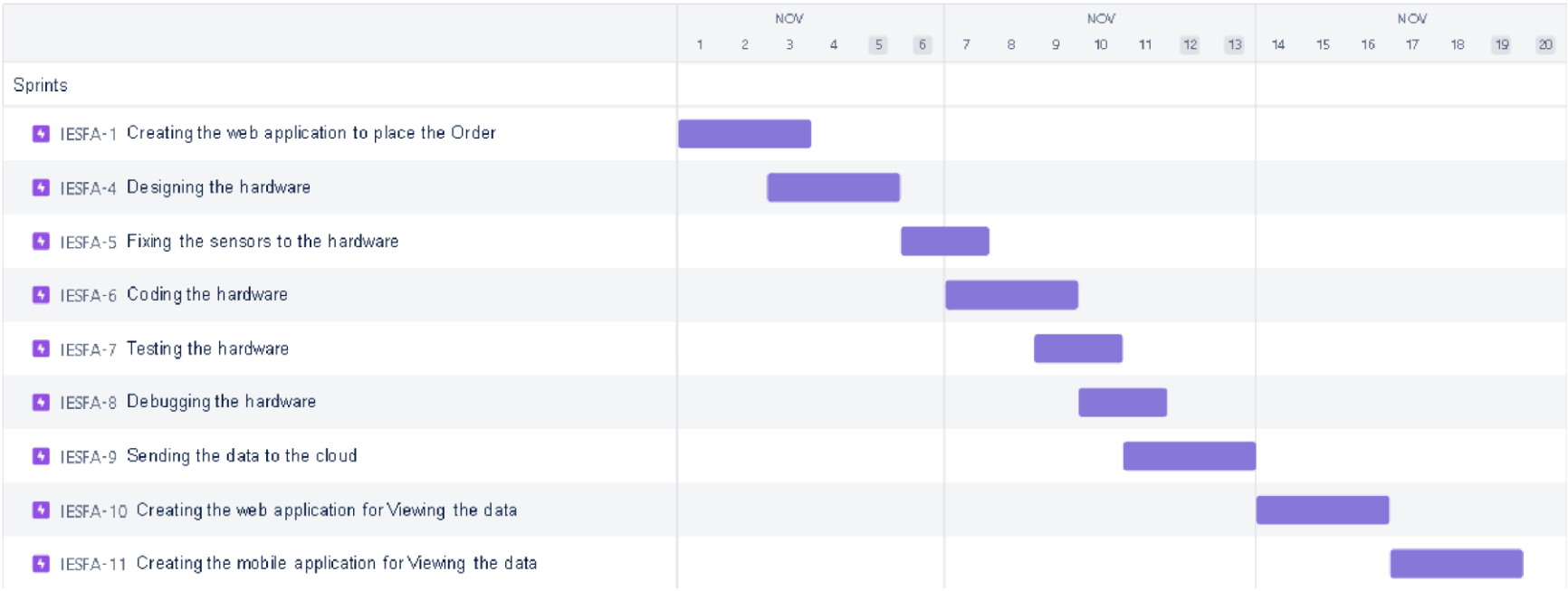
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	01 Nov 2022	07 Nov 2022		12 Nov 2022
Sprint-2	20	2 Days	08 Nov 2022	10 Nov 2022		13 Nov 2022
Sprint-3	20	5 Days	11 Nov 2022	16 Nov 2022		16 Nov 2022
Sprint-4	20	2 Days	17 Nov 2022	19 Nov 2022		19 Nov 2022

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)

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

Burndown Chart:

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



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

7.1 Feature 1

The unique feature of our project is that we can add up to 5 soil moisture and humidity sensors and we can add up to 4 pumps. The device is also capable of sustaining solar power so that it could operate without any power shortages during day time. It is capable of operating autonomously without any human intervention

7.2 Feature 2

The person who connected with the device can only view the data, other than the person connected with the device will not be able to view the sensor readings. It enables a simple device security principle that others can not view and control the sensor readings from the device.

7.3 Feature 3

The user who is connected to the device can view the readings in mobile as well as the desktop. It is both mobile and web responsive so there is no need to install a separate mobile application in the mobil;e devices to view the device status.

10. ADVANTAGES AND DISADVANTAGES :

Advantages:

1. Remote monitoring.
2. Autonomous watering system.
3. Environment monitoring.
4. Enables data security over sharing.
5. Can view the data in both mobile and desktop.

Disadvantages

1. Limited to small area - Requires similar units to cover larger area.
2. Few data discrepancies during bad weather.

11. CONCLUSION:

The aim of this project is to make the life and work of the farmer much easier. This can be achieved using the technique Precision Farming, this involves autonomous monitoring of crops and other environmental parameters which has an effect on the crop, these environmental conditions are:

1. Environmental Humidity 2.
   Environmental Temperature.
3. Soil Moisture.
4. Rain Sensing.

Above mentioned are some of the conditions monitored autonomously, threshold parameters for various crops are automatically set upon user input of crop variety to be monitored. By this system one could achieve a good yield and better nutritional crops in their agricultural produce.

12. FUTURE SCOPE:

Future scope of our project relies on the farmers and their feedbacks, in future we are planning to add the following features:

1. One device one farm - Cover the entire farm area with a single device.
2. Pest monitoring system.
3. Estimated yield calculator.
4. Estimated time of cultivation.
5. Individual cloud management dashboard.

13. APPENDIX :

Source Code:

IoT :

```
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <PubSubClient.h>
#include <Adafruit_BMP280.h>
#include <math.h>
#include <Wire.h>
#define BMP_SDA 21
#define BMP_SCL 22
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;
#define DHT11_PIN 4
#define rainAnalog 35
#define rainDigital 34
#define moistureDigital 32

Adafruit_BMP280 bmp280;
const char* ssid     = ""; const char*
```

```
password = ""; AsyncWebServer
server(80); AsyncEventSource
events("/events"); unsigned long
lastTime = 0; unsigned long timerDelay
= 1000;
int soil; int
rain; int
rainA;
float
temperat
ure; float
humidity;
float
pressure;
float
altitude;
long lastMsg
=
0; int
pumpRel
ayPin =
26;


#define ORG "6jw3v9"
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "######################" #define
TOKEN "######################"


char servers[] = ORG
".messaging.internetofthings.ibmcloud.com"; char pubTopic1[] =
"iot-2/evt/temperature/fmt/json"; char pubTopic2[] = "iot-
2/evt/humidity/fmt/json"; char pubTopic3[] = "iot-
2/evt/pressure/fmt/json"; char pubTopic4[] = "iot-
2/evt/altitude/fmt/json"; char authMethod[] = "use-token-auth"; char
token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;


WiFiClient wifiClient;
PubSubClient client(servers, 1883, NULL, wifiClient);
```

```
// Init BME280 void
initBME() {
  if (!bmp280.begin(0x76)) {
    Serial.println("Could not find a valid BMP280 sensor, check wiring!"); while
    (1);
  }
}

void getSensorReadings() {
  DHT.read(DHT11_PIN); temperature =
  DHT.temperature; humidity =
  DHT.humidity; pressure =
  bmp280.readPressure() / 100; soil =
  digitalRead(moistureDigital); rain =
  digitalRead(rainDigital); rainA =
  analogRead(rainAnalog); altitude =
  bmp280.readAltitude(1011.18); if(soil ==
  1){
    digitalWrite(pumpRelayPin, LOW);
  } else{ digitalWrite(pumpRelayPin,
  HIGH);
  }
}

// Initialize WiFi void
initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print('.'); delay(1000);
  }
  Serial.println(WiFi.localIP());
}
```

```
String processor(const String& var) {
  getSensorReadings(); //Serial.println(var);
  if (var ==
  "TEMPERATURE") {
    return String(temperature);
  } else if (var ==
  "HUMIDITY") { return
  String(humidity); } else if
  (var ==
  "PRESSURE") { return
    String(pressure);
  } else if (var ==
  "ALTITUDE") { return
    String(altitude);
  } else if (var ==
  "RAINING") { return
    String(rain);
  } else if (var ==
  "SOIL") { return String(soil);
  } return
  String();
}

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
< head >
  <title>Grow Greens Smart</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHp
vLbHG9Sr" crossorigin="anonymous">
  <link rel="icon" href="data:,">
  <style> html {font-family:     Arial; display:  inline-block;    text-align:       center;
background-color:#FCF8E8} p { font-size: 1.2rem;}
```

```
    body {  margin: 0;}
    .topnav { overflow: hidden; color: #6D9886; font-size: 1rem; }
    .content { padding: 20px; }
.card { background-color: #F2AA4CFF; box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5); border-radius: 30px;}
.cards { max-width: 800px; margin: 0 auto; display: grid; grid-gap: 2rem; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); }
    .reading { font-size: 1.4rem; }
  </style>
< /head >
< body >
  <div class="topnav">
    <h1>Grow Greens Smart</h1>
  < /div >
  <div class="content">
    <div class="cards">
      <div class="card">
        <p><i class="fas fa-thermometer-half" style="color:#101820FF; font-size:25px"></i> Temperature</p><p><span class="reading" style = "color:#101820FF"><span id="temp" style="font-size:1rem; font-weight:bolder;">%TEMPERATURE%</span>
&deg;C</span></p>
      < /div >
      <div class="card">
        <p><i class="fas fa-tint" style="color:#101820FF; font-size:25px"></i> Humidity</p><p><span class="reading" style="color:#101820FF; fontsize:1rem;"><span id="hum" style="font-size:1rem; fontweight:bolder;">%HUMIDITY%</span>
&percnt;</span></p> < /div
      >
      <div class="card">
        <p><i class="fas fa-angle-double-down" style="color:#101820FF; font-size:25px"></i> Pressure</p><p><span class="reading" style="color:#101820FF; font-size:1rem;"><span id="pres" style="font-size:1rem;
```

```
font-weight:bolder;">%PRESSURE%</span> hPa</span></p>
    < /div >
    <div class="card">
        <p><i class="fas fa-mountain" style="color:#101820FF; font-size:25px"></i>
Altitude</p><p><span class="reading" style="color:#101820FF"><span
id="alti" style="font-size:1rem; font-weight:bolder;">%ALTITUDE%</span>
m</span></p>
    < /div >
    <div class="card">
        <p><i class="fas fa-cloud-rain" style="color:#101820FF; font-size:25px"></i>
Raining</p><p><span class="reading" style="color:#101820FF"><span id="rain"
style="font-size:1rem; font-weight:bolder;">%RAINING%</span></span></p> < /div >
    <div class="card">
        <p><i class="fas fa-tree" style="color:#101820FF; font-
size:25px"></i> Moisture</p><p><span class="reading"
style="color:#101820FF"><span id="soil" style="font-size:1rem;
fontweight:bolder;">%SOIL%</span></span></p> < /div > < /div >
 < /div > <script> if
(!!window.EventSource) { var source =
new EventSource('/events');

source.addEventListener('open', function(e) {
  console.log("Events Connected");
}, false); source.addEventListener('error', function(e)
{
  if (e.target.readyState != EventSource.OPEN) {
    console.log("Events Disconnected");
  }
}, false);

source.addEventListener('message', function(e) {
  console.log("message", e.data);
}, false);

source.addEventListener('temperature', function(e) {
```

```
  console.log("temperature", e.data); document.getElementById("temp").innerHTML
  = e.data;
}, false);

source.addEventListener('humidity', function(e) {
  console.log("humidity", e.data);
  document.getElementById("hum").innerHTML = e.data;
}, false);

source.addEventListener('pressure', function(e) {
  console.log("pressure", e.data);
  document.getElementById("pres").innerHTML = e.data;
}, false);
source.addEventListener('altitude', function(e) {
  console.log("latitude", e.data);
  document.getElementById("alti").innerHTML = e.data;
}, false);

source.addEventListener('rain', function(e) { console.log("Rain",
e.data);
  if(e.data == '0')
    document.getElementById("rain").innerHTML = "Raining";
  else document.getElementById("rain").innerHTML = "Not
    Raining";
}, false);

source.addEventListener('soil', function(e) { console.log("Soil
Moisture", e.data);
  if(e.data == '1')
    document.getElementById("soil").innerHTML = "Less Water";
  else document.getElementById("soil").innerHTML = "Enough
    Water";
}, false);
```

```
}
</script>
< /body > </html>)rawliteral";


void setup() { Serial.begin(115200);
  pinMode(rainDigital, INPUT);
  pinMode(moistureDigital, INPUT);
  pinMode(pumpRelayPin, OUTPUT); initWiFi();
  initBME();
  // Handle Web Server server.on("/", HTTP_GET,
  [](AsyncWebServerRequest * request) { request->send_P(200,
    "text/html", index_html, processor);
  }) ;
  // Handle Web Server Events
  events.onConnect([](AsyncEventSourceClient * client) { if
  (client->lastId()) {
                        Serial.printf("Client reconnected! Last message ID that it got is: %u\n",
client->lastId());
    }
    // send event with message "hello!", id current millis
    // and set reconnect delay to 1 second client-
    >send("hello!", NULL, millis(), 10000);
  }) ;
  server.addHandler(&events)
  ; server.begin(); if
  (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(servers); while
    (!client.connect(clientId, authMethod, token)) {
    Serial.print("."); delay(500);
    }
    Serial.println("Bluemix connected");
  }
}
void loop() {
```

```
client.loop(); long now =
millis(); if (now - lastMsg
> 3000) { lastMsg
 = now;


 String payload =
 "{\"temperature\":"; payload += temperature;
 payload += "}";
 Serial.print("Sending payload: ");
 Serial.println(payload); if (client.publish(pubTopic1,
 (char*) payload.c_str())) { Serial.println("Publish ok");
 }  else  {
  Serial.println("Publish failed");
 }
 String payload1 = "{\"humidity\":";
 payload1 += humidity; payload1
 += "}";
 Serial.print("Sending payload: ");
 Serial.println(payload1); if (client.publish(pubTopic2,
 (char*) payload1.c_str())) { Serial.println("Publish ok");
 }  else  {
  Serial.println("Publish failed");
 }


 String payload2 = "{\"pressure\":";
 payload2 += pressure; payload2
 += "}";
 Serial.print("Sending payload: ");
 Serial.println(payload2); if (client.publish(pubTopic3,
 (char*) payload2.c_str())) { Serial.println("Publish ok");
 }  else  {
  Serial.println("Publish failed");
 }
```

```
  String payload3 = "{\"altitude\":";
  payload3 += altitude; payload3
  += "}";
  Serial.print("Sending payload: ");
  Serial.println(payload3); if (client.publish(pubTopic4,
  (char*) payload3.c_str())) { Serial.println("Publish ok");
  }  else  {
    Serial.println("Publish failed");
  }

 } if ((millis() - lastTime) > timerDelay)
 {
  getSensorReadings();
  Serial.printf("Temperature = %.2f ºC \n", temperature); Serial.printf("Humidity = %.2f
  \n", humidity);
  Serial.printf("Pressure = %.0f hPa \n", pressure);
  Serial.printf("Altitude = %.0f m \n", altitude);
  Serial.printf("Rain = %d\n", rain);
  Serial.printf("Rain = %d\n", rainA);
  Serial.printf("Soil = %d\n", soil);
  Serial.println();

  // Send Events to the Web Server with the Sensor Readings
  events.send("ping", NULL, millis());
  events.send(String(temperature).c_str(), "temperature", millis());
  events.send(String(humidity).c_str(), "humidity", millis());
  events.send(String(pressure).c_str(), "pressure", millis());
  events.send(String(altitude).c_str(), "altitude", millis());
  events.send(String(rain).c_str(), "rain", millis());
  events.send(String(soil).c_str(), "soil", millis());

  lastTime = millis();
 }
}
```

github link : https://github.com/IBM-EPBL24276-1659940929