**K.S.R COLLEGE OF ENGINEERING**

**(Autonomous)**

**TIRUCHENGODE-637 215**

**ANNA UNIVERSITY: CHENNAI 600 025**

**November 2022**

**PROJECT**

Smart Farmer - IoT Enabled SmartFarming Application

**SUBMITTED BY**

**TEAM ID :** **IBM-Project-37586-1660312808**

**ASHOKKUMAR K - 1915004**

**DINESH S - 1915006**

**MOHAN S - 1915033**

**PERUMAL R - 1915039**

**Project Report**

1. **INTRODUCTION**
   1. Project Overview 1.2 Purpose
2. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   1. Functional requirement
   2. Non-Functional requirements
5. **PROJECT DESIGN**
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
6. **PROJECT PLANNING & SCHEDULING**
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   1. Feature 1
   2. Feature 2
   3. Database Schema (if Applicable)
8. **TESTING**
   1. Test Cases
   2. User Acceptance Testing
9. **RESULTS**
   1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE** 13. **APPENDIX** Source Code GitHub & Project Demo Link

# Introduction

## Problem Statement – Smart Farmer : 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.

## 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.

## 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.

# Literature Survey

## Existing Problem

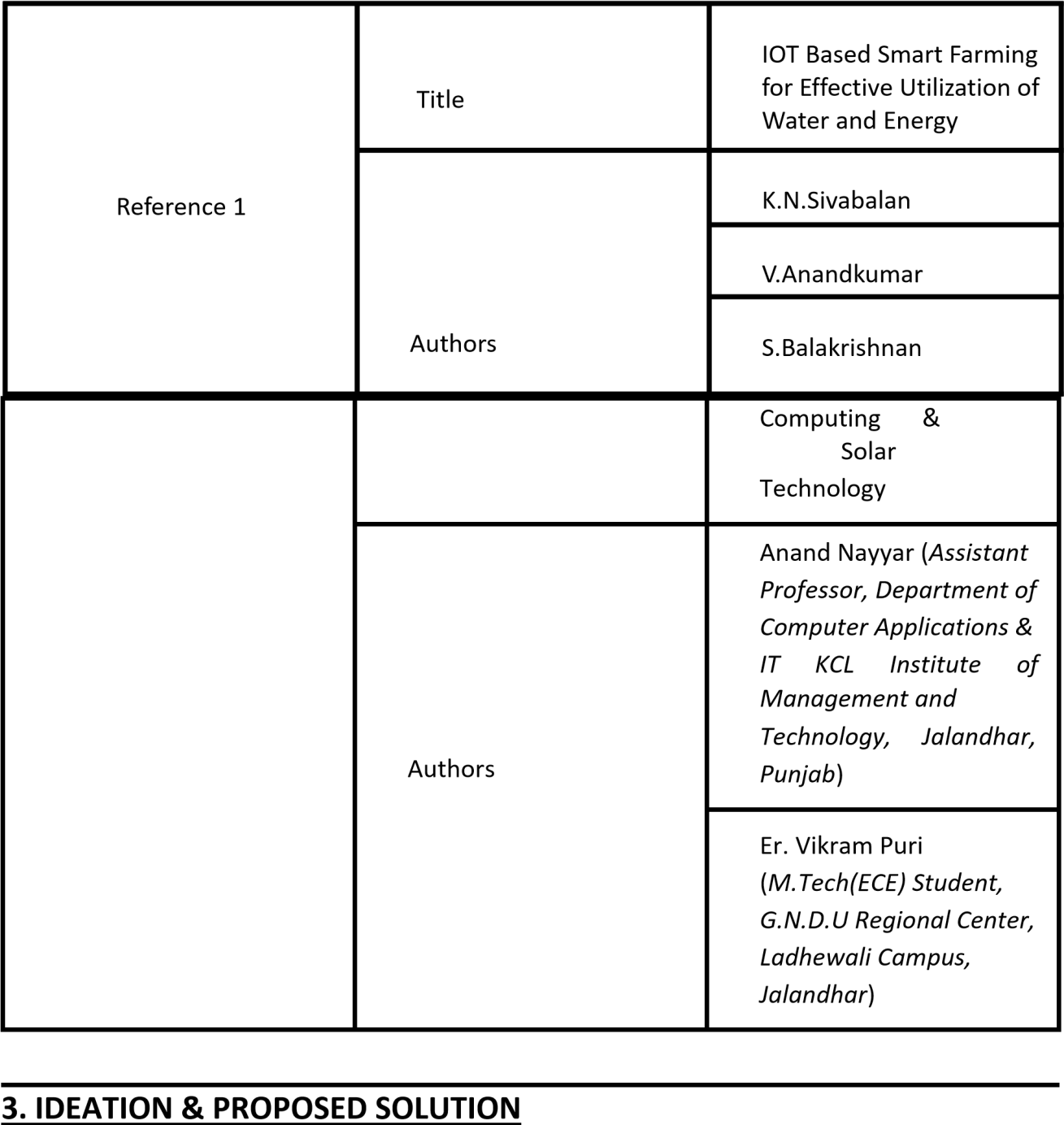
Farmers must meet the challenging needs of our planet and the experiences and the expectations of the regulations, consumers and food produced. There are increasing pressure from climate change, soil erosion and biodiversity loss and from consumers changing tastes in food and concerns about how it is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

The scenario of decreasing water tables, drying up of rivers and tanks, unpredictable environment present an urgent need of proper utilization of water. To cope up with this use of temperature and moisture sensors at suitable locations for monitoring of crops is implemented .An algorithm developed with threshold values of temperature and soil moisture can be programmed into micro controller based gateway to control water quantity. The system can be powered by Photo voltaic panels and can have duplex communication link based on cellular – Internet interface that allow data inspection and irrigation scheduling to be programmed through web. The technological development in open source software and hardware make it easy to develop the device which can make better monitoring and wireless sensor network made it possible to use in monitoring and control of green house parameter in precision agriculture.

## References

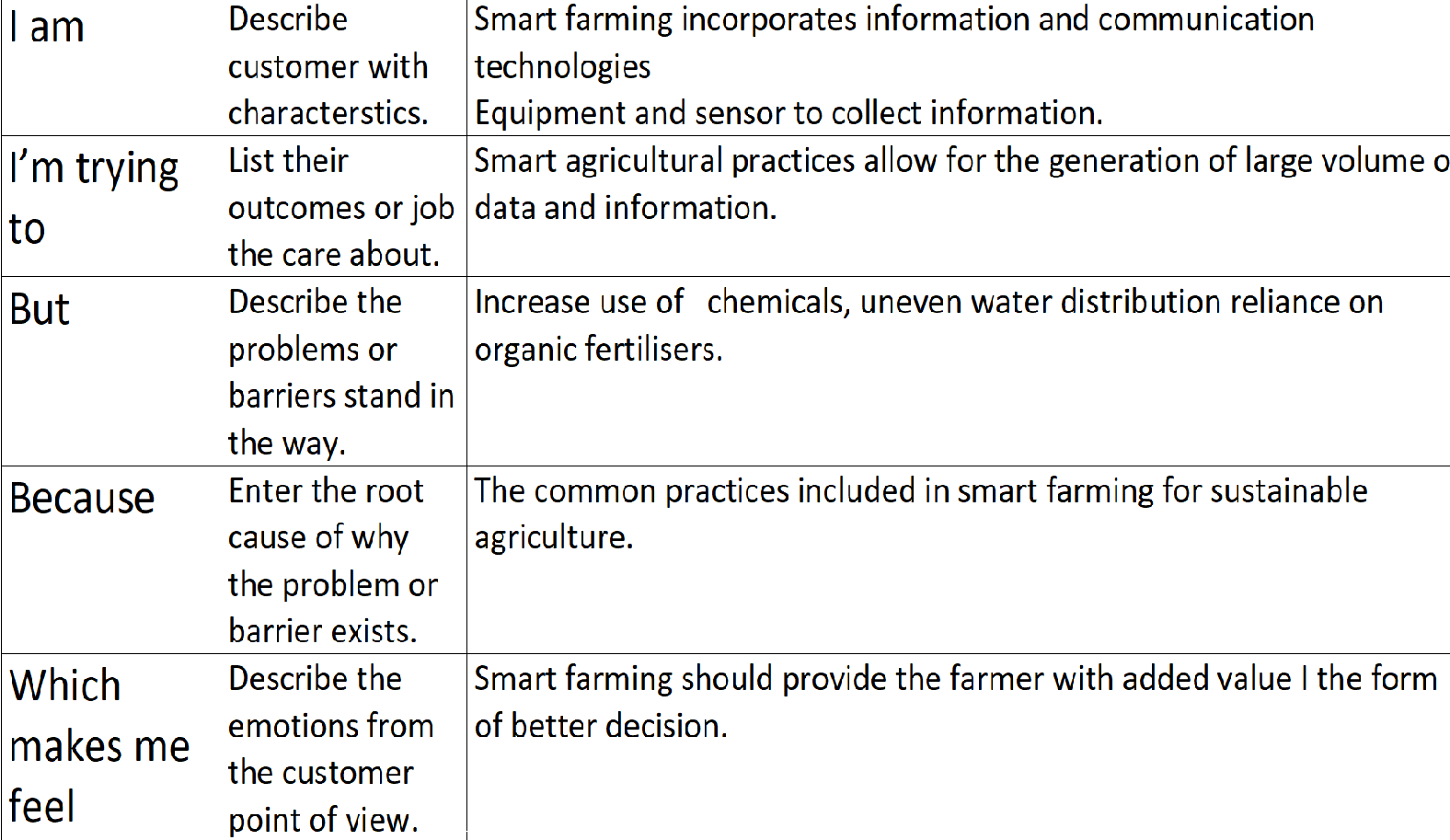


* 1. **Empathy Map Canvas**

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

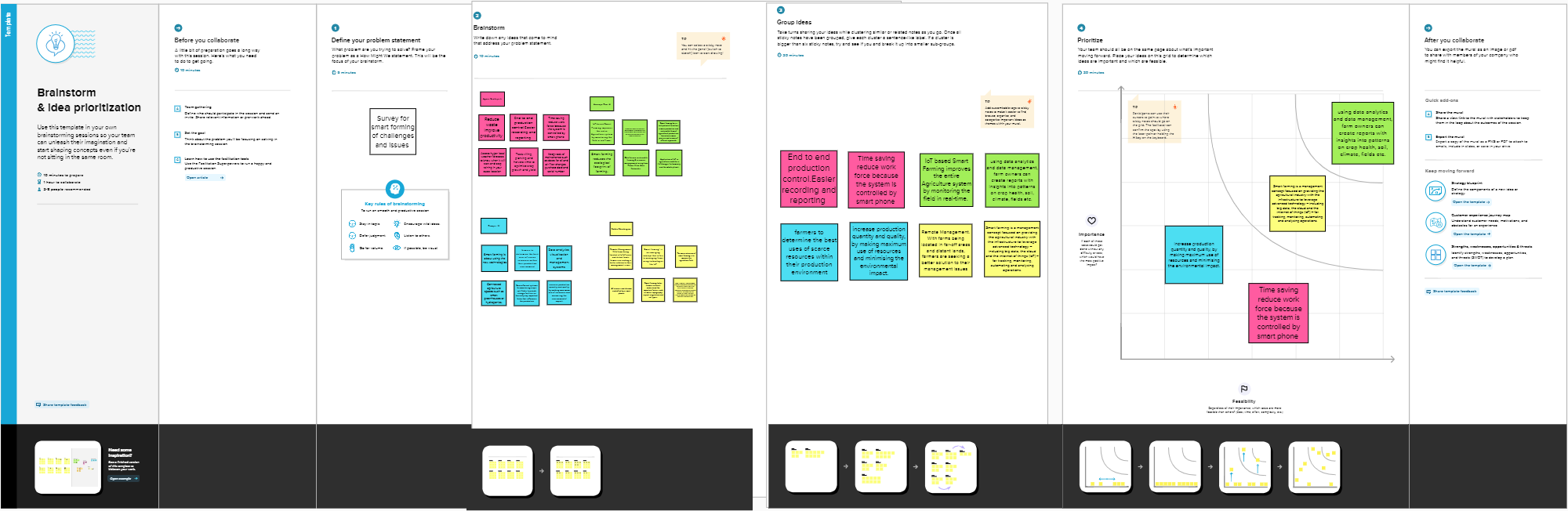
## Ideation & Brainstorming:

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



## 

**Step 2 : Brainstorming**

****

## Proposed Solution:

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
|  | Problem Statement (Problem to be solved) | The challenges of a smart agriculture system include the integration of these sensors and tying the sensor data to the analytics driving automation and response activities. |
|  | Idea / Solution description | To create affordable IoT based smart agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors. |
|  | Novelty / Uniqueness | Various eminent researchers have been making efforts for smart farming by using IoT concepts in agriculture. But, a bouquet of unfolded challenges is still in a queue for their effective solution. IoT helps in better crop management, better resource management, cost efficient agriculture, improved quality and quantity , crop monitoring |
|  | Social Impact / Customer Satisfaction | Consumer behaviour plays a major role in the Agriculture products segment. The marketers of agriculture products need to be innovative and dynamic in order to compete with the changing purchase behaviour in the Agriculture products market among urban residents. |
|  | Business Model (Revenue Model) |  |
|  | Scalability of the Solution | Scalability in smart farming refers to the adaptability of a system to increase the capacity, for example, the number of technology devices such as sensors and actuators, while enabling timely analysis .IoT technology provides a smart farming solution, enabling farmers to manage their fields remotely via smart gadgets. |

**3.4 Proposed Solution Fit**

|  |  |  |
| --- | --- | --- |
| 1. CUSTOMER SEGMENT(S) | 2. JOBS-TO-BE-DONE / PROBLEMS | 3. TRIGGERS |
| **We divide our customers into segments up based on land holding,**   * **Micro farmers** * **small or marginal farmers** * **emerging and large farmers** * **commercial farmers** | * **Insufficient Water Supply** * **Irrigation** * **Soil erosion** * climate change * Satisfy consumers * Poor Storage Facilities | * Agricultural policy, standards, laws * Prices * Market demands * opportunities * Technological progress * Environment |

|  |  |  |
| --- | --- | --- |
| 4. EMOTIONS: BEFORE / AFTER | 5. AVAILABLE SOLUTIONS | 6. CUSTOMER CONSTRAINTS |
| * *Farmers face myriad barriers to mental health services* * *Financial pressures* * Excessive workloads * Health, pain, or mobility issues | * Adopt and learn new technologies * Invest in farm productivity * Better Water Management * mobile technology | * Lack of proper irrigation facilities * production machinery * access to institutional credit * difficulties procuring inputs and storing products * negative impacts of climate |

|  |  |  |
| --- | --- | --- |
| 7. BEHAVIOUR | 8.CHANNELS of BEHAVIOUR | 9. YOUR SOLUTION |
| * Find the correct technology to improve farming * The hardware * Sensors, Arduino * Mobile applications * Data collection frequency | 8.1 ONLINE   * software for specialized applications and for enabling IoT-based automation | * We design arduino   With sensor embedded system based on IOT .  Tthis technology is end to end method,save time and more efficient method in now days. |
| 8.2 OFFLINE   * Farmers make their works perfectly with in the given time |

# 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. |

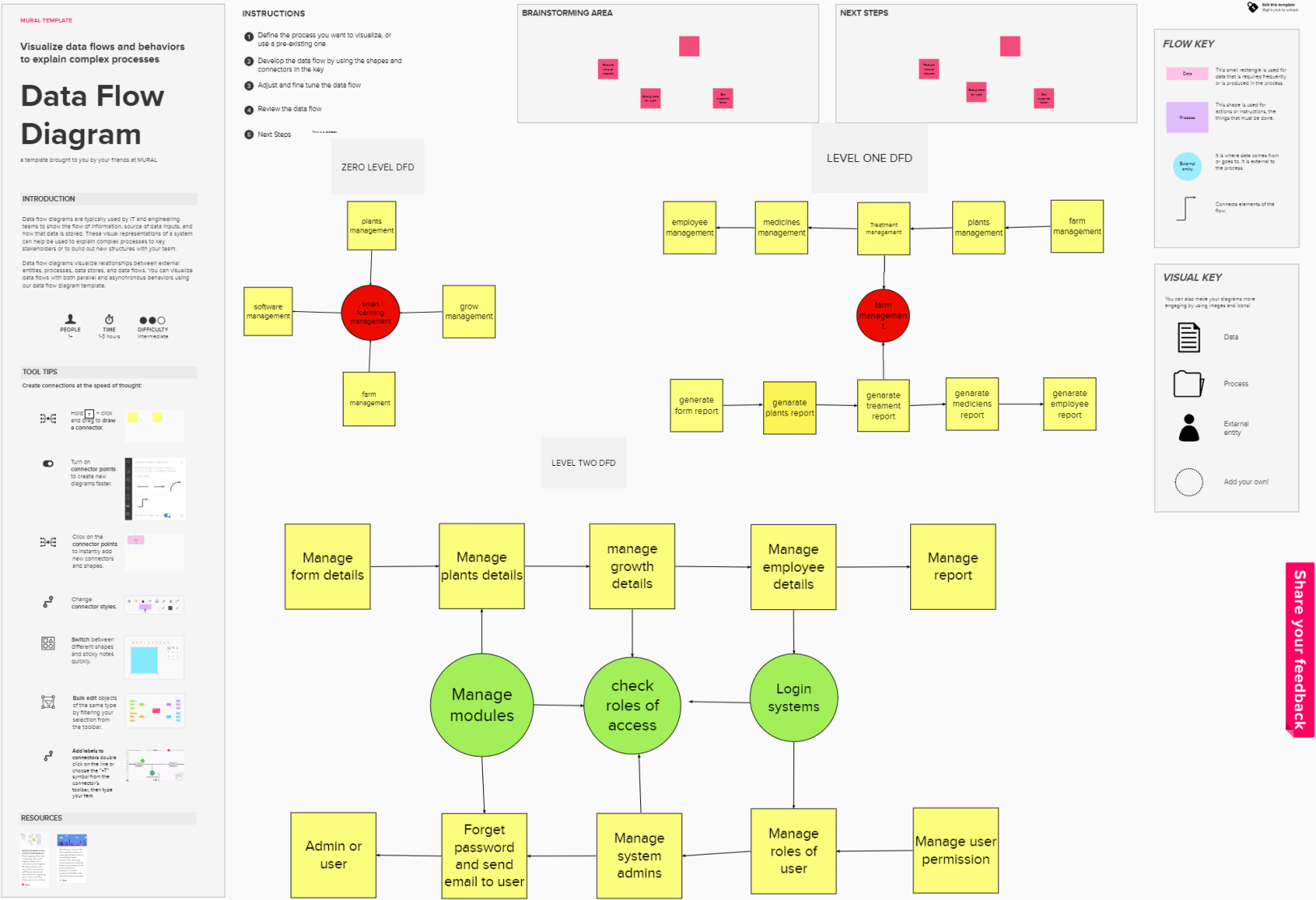
# 5. Project Design:

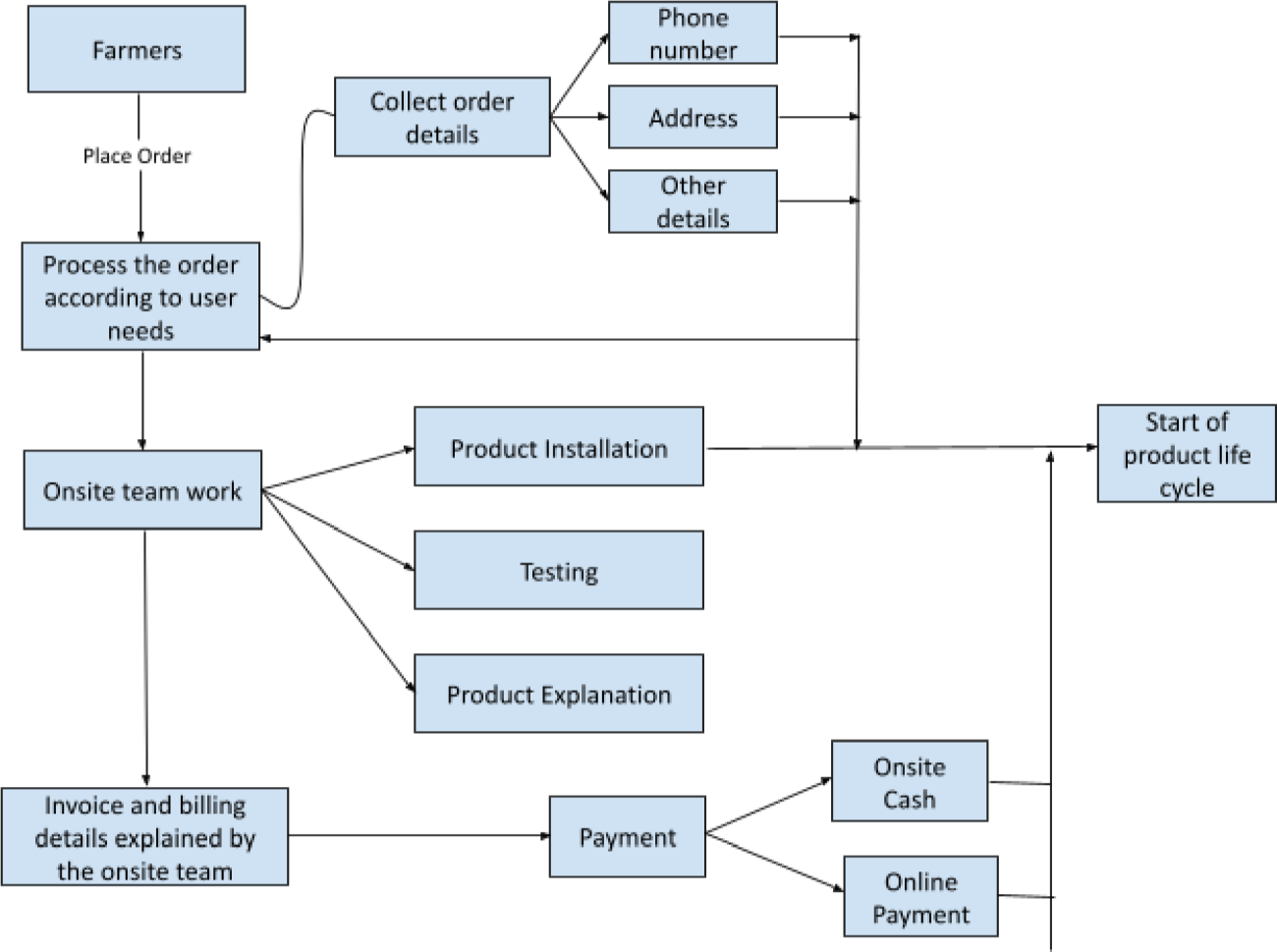
## 5.1Data Flow Diagrams:

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.





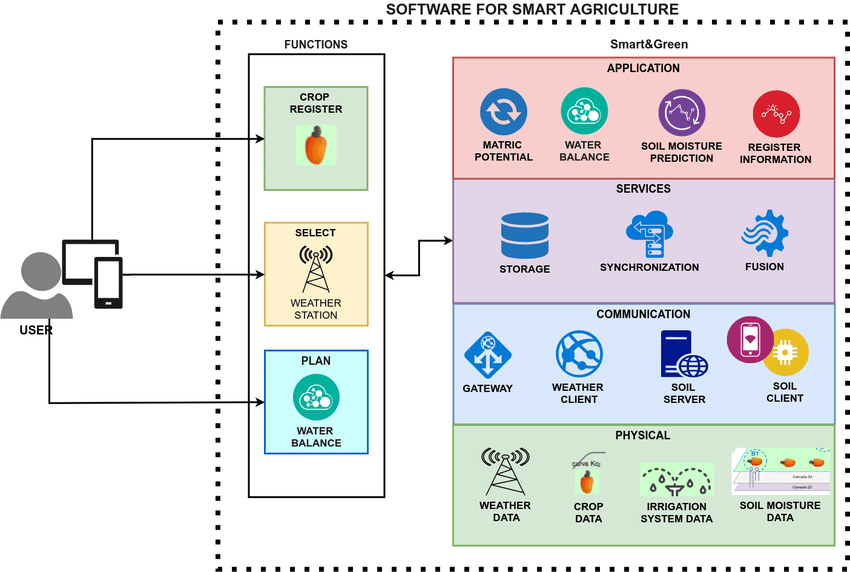
## 5.2 Solution & Technical Architecture:

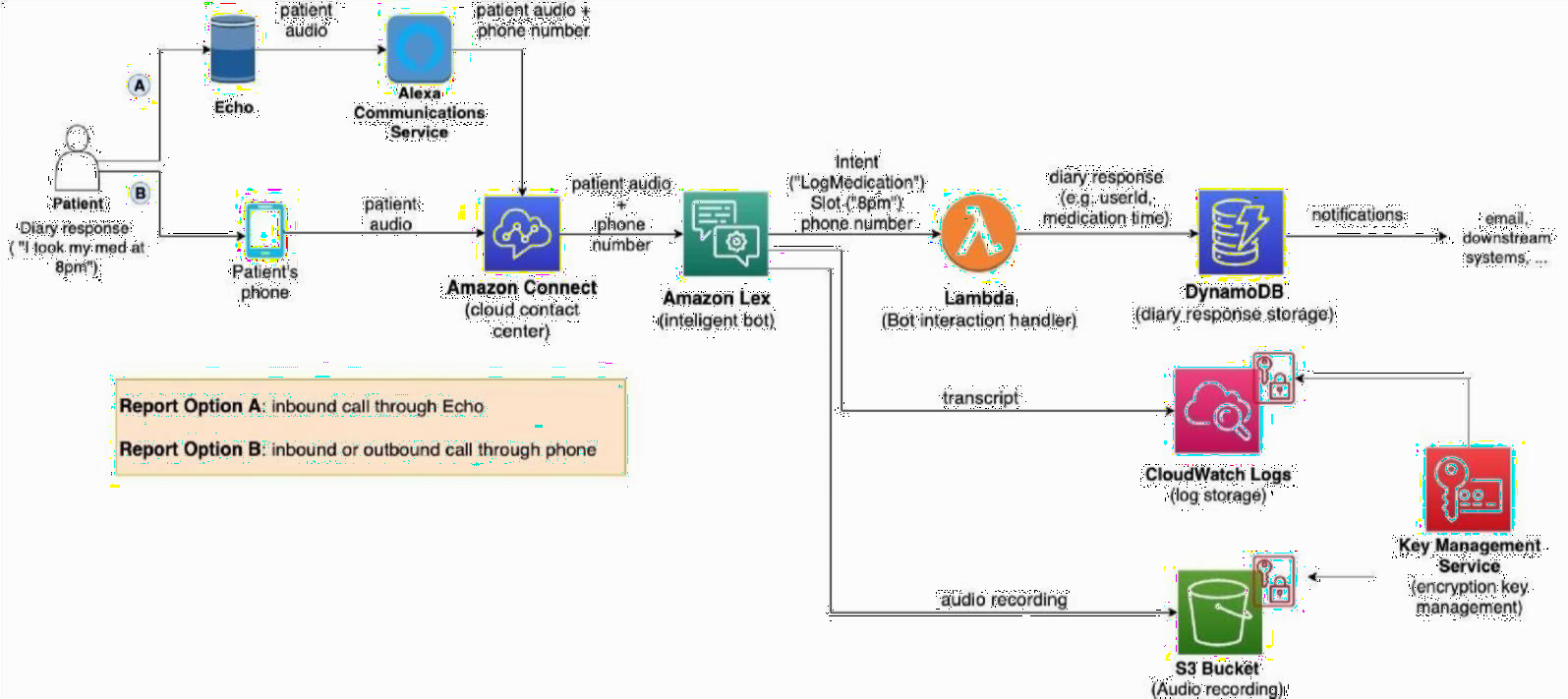
**Solution Architecture:**

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:





**5.3UserStories**

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

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 |

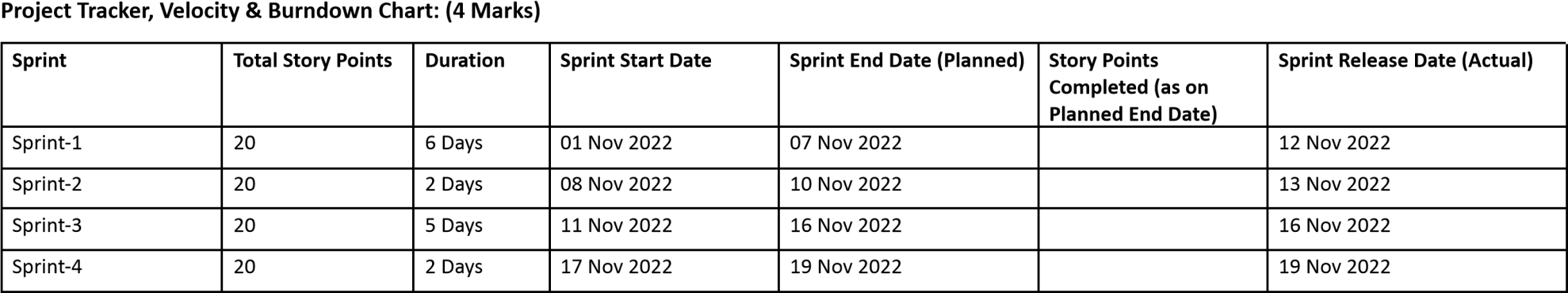
# 6. PROJECT PLANNING & SCHEDULING:

## 6.1 Sprint Planning & Estimation

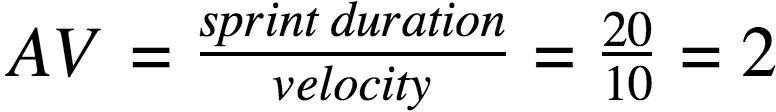
Use the below template to create product backlog and sprint schedule

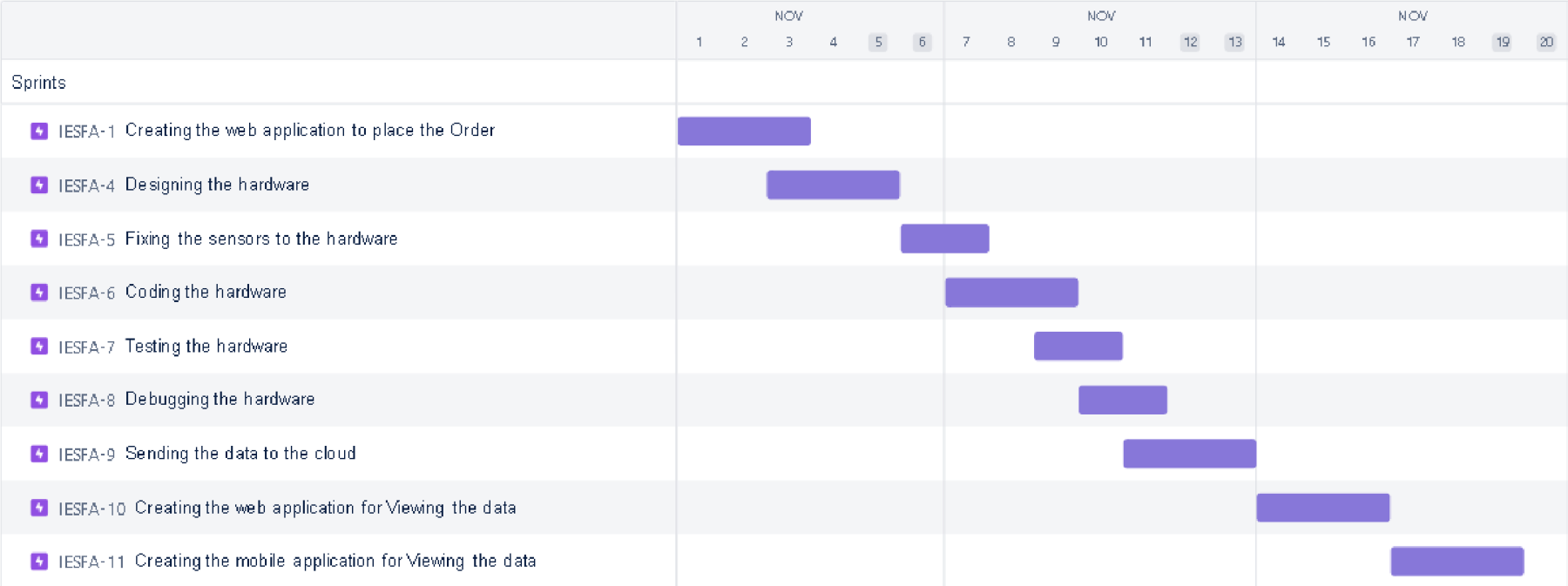
|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story**  **Number** | **User Story / Task** |
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. |
| Sprint-1 |  | USN-2 | As a user, I will receive confirmation email once I have registered for the application |
| Sprint-2 | Hardware Development phase - 1 | USN-3 | Monitoring and displaying humidity and temperature. |
| Sprint-2 | Hardware Development phase - 2 | USN-4 | Connecting the device online(Configuring wifi automatically through a web portal) |
| Sprint-3 | Hardware Development phase - 3 | USN-5 | Uploading data to the cloud. |
| Sprint-4 | Hardware Development phase - 4 | USN-6 | Establishing relay controls. |
| Sprint-4 | Mobile Application | USN-7 | Deployment of application |

**Velocity:**



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





**6.2. Sprint Delivery Schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROJECT  NAME | TEAM  WORKERS | PROJECT  START DATE | PROJECT  END DATE | TOTAL  WEEKS |
| **SmartFarmer - IoT Enabled Smart Farming Application** | **Agnes Santhiya L**  **Anusuya Rani K**  **Subha Shenbagam S**  **Ramya M** | **August 22, 2022** | **November 19,2022** | **13 Weeks** |
| PROJECT MILESTONES & ACTIVITIES | | | | |
| MILESTONES | ACTIVITIES | STATUS | EXPECTED DATE | ACTUAL  DATE |
| Preparation Space | Pre-requisites | Completed | 22 - 27 Aug ,2022 | Aug 25,2022 |
| Registration |
| Environment set-up |
| Ideation Phase | Literature Survey | Completed | 29 Aug - 3 Sep, 2022 | Sep 1,2022 |
| Empathy Map | Completed | 5 - 10 Sep, 2022 | Sep 7,2022 |
| Brainstroming | Completed | 12 - 17Sep,2022 | Sep 15,2022 |
| Project Design Phase - I | Proposed Solution Document | Completed | 19 - 24 Sep,2022 | Sep 21,2022 |
| Problem Solution Fit | Completed | 26 Sep - 1 Oct, 2022 | Sep 30,2022 |
| Solution Architecture |
| Project Design  Phase - II | Customer Journey | Completed | 3 - 8 Oct, 2022 | Oct 6,2022 |
| Functional Requirements | Completed | 10 - 15 Oct, 2022 | Oct 14,2022 |
| Data Flow Diagram |
| Technology Architecture |
| Projecct Planning Phase | Sprint Delivery Plan | Completed | 17 - 22 Oct, 2022 | Oct 20,2022 |
| Prepare Milestone & Activity List |
| Project Development Phase | Project Development Delivery of Sprit - I | In progress | 24 - 29 Oct, 2022 | - |
| Project Development Delivery of Sprit - II | In Progress | 31 Oct - 5 Nov, 2022 | - |
| Project Development  Delivery of Sprit - III | In Progress | 7 - 12 Nov, 2022 | - |
| Project Development Delivery of Sprit - IV | In Progress | 14 - 19 Nov, 2022 | - |

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

void setup()

{

Serial.begin(9600);

pinMode(A2,INPUT);

pinMode(12,OUTPUT);

}

void loop()

{

double data=analogRead(A2);

double n=data/1024;

Serial.print(" Converted Analog data: ");

Serial.println(n);

double volt=n\*5;

Serial.print(" Voltage data: ");

Serial.println(volt);

double off=volt-0.5;

Serial.print("After offset data: ");

Serial.println(off);

double temperature=off\*100;

Serial.print("Temperature data: ");

if(temperature>=60){

Serial.println(temperature);

tone(12,temperature);

delay(1000);}

else{

noTone(12);

}

}

void setup()

{

Serial.begin(9600);

pinMode(2,INPUT);

pinMode(12,OUTPUT);

}

void loop()

{

int motion=digitalRead(2);

Serial.print("Position is; ");

Serial.println("motion");

if(motion==1){

Serial.println("Motion detected");

tone(12,motion);

delay(1000);

}

else{

Serial.println("No Motion");

}

}

**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

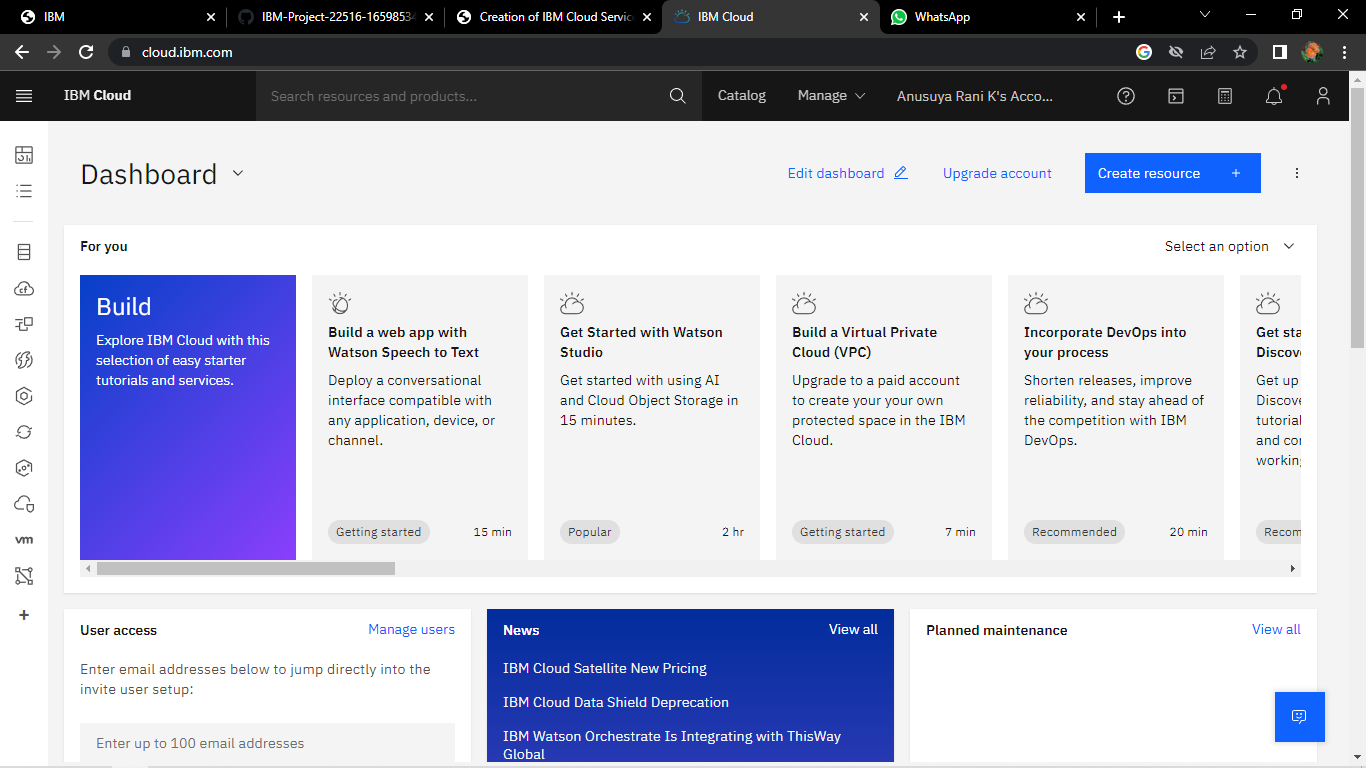
**7.3 Feature 3**

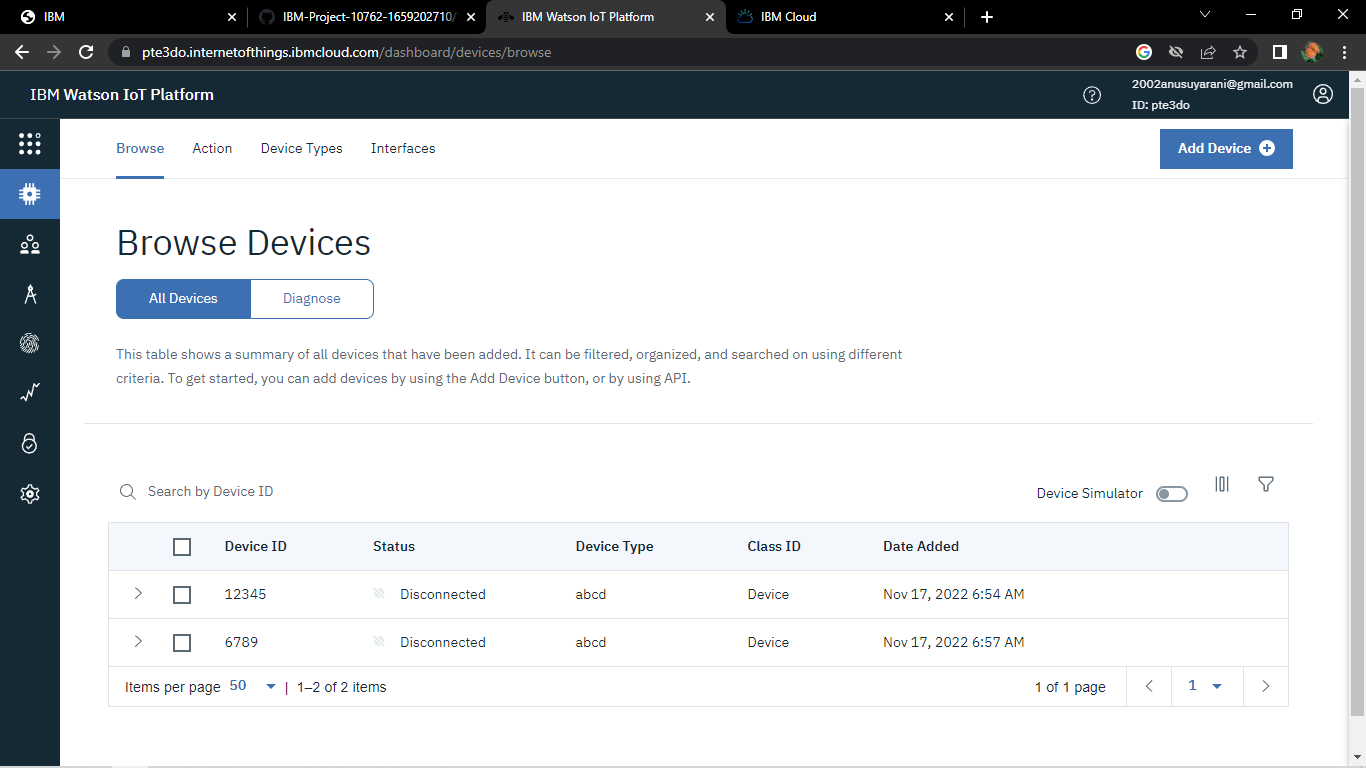
The user who is connected to the device can view the readings in mobile as well as the desktop. It is both mobile and web responsive so there is no need to install a separate mobile application in the mobil;e devices to view the device status.

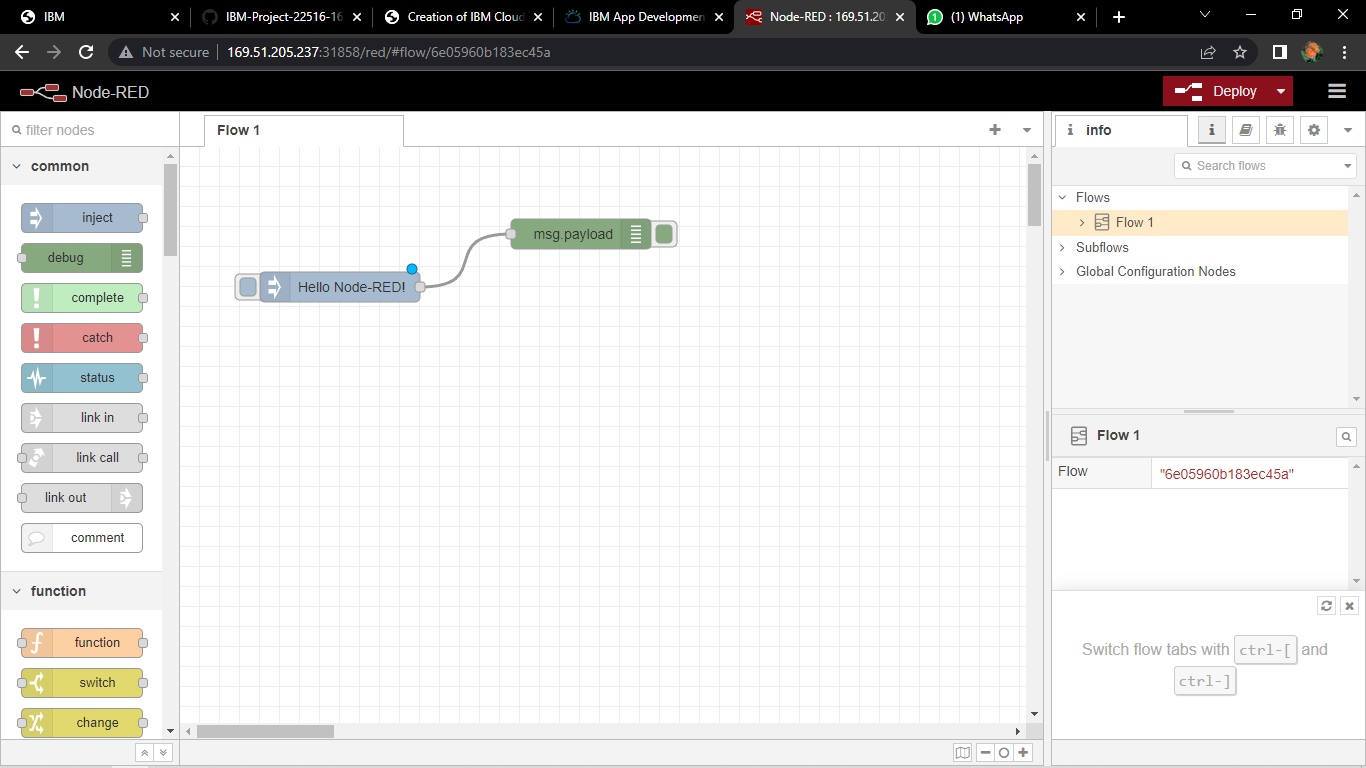
**9. RESULTS**

**9.1 Performance Metrics**

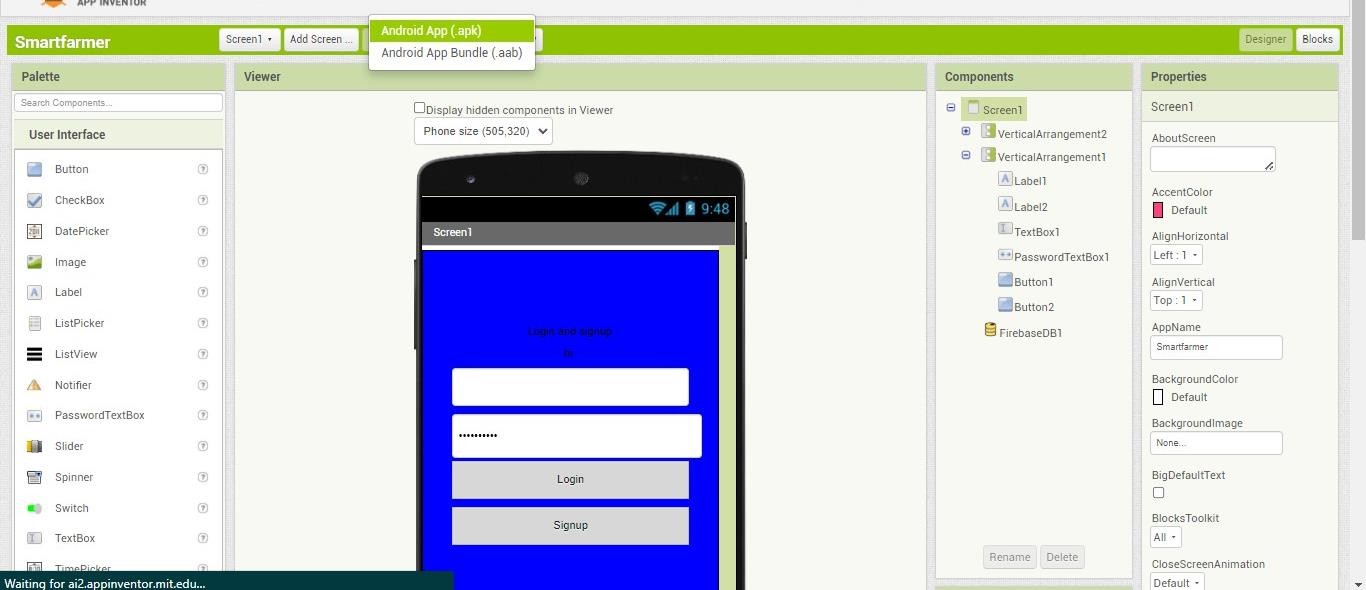
**Cloud Services & Node-Red services**

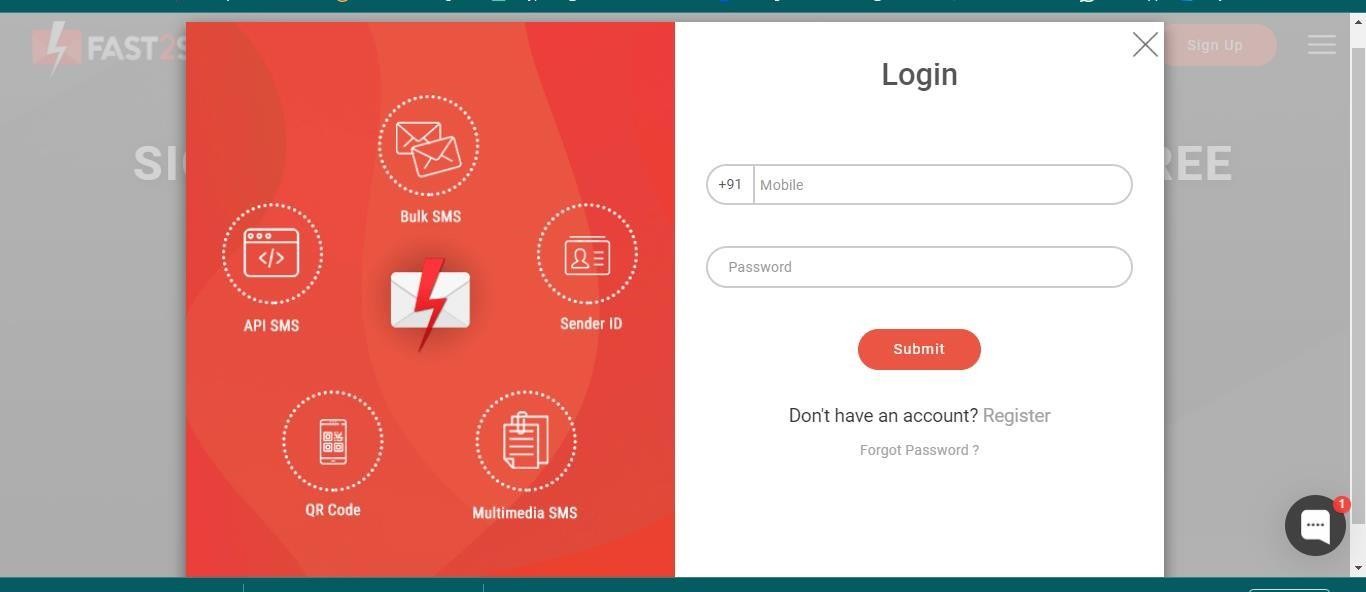


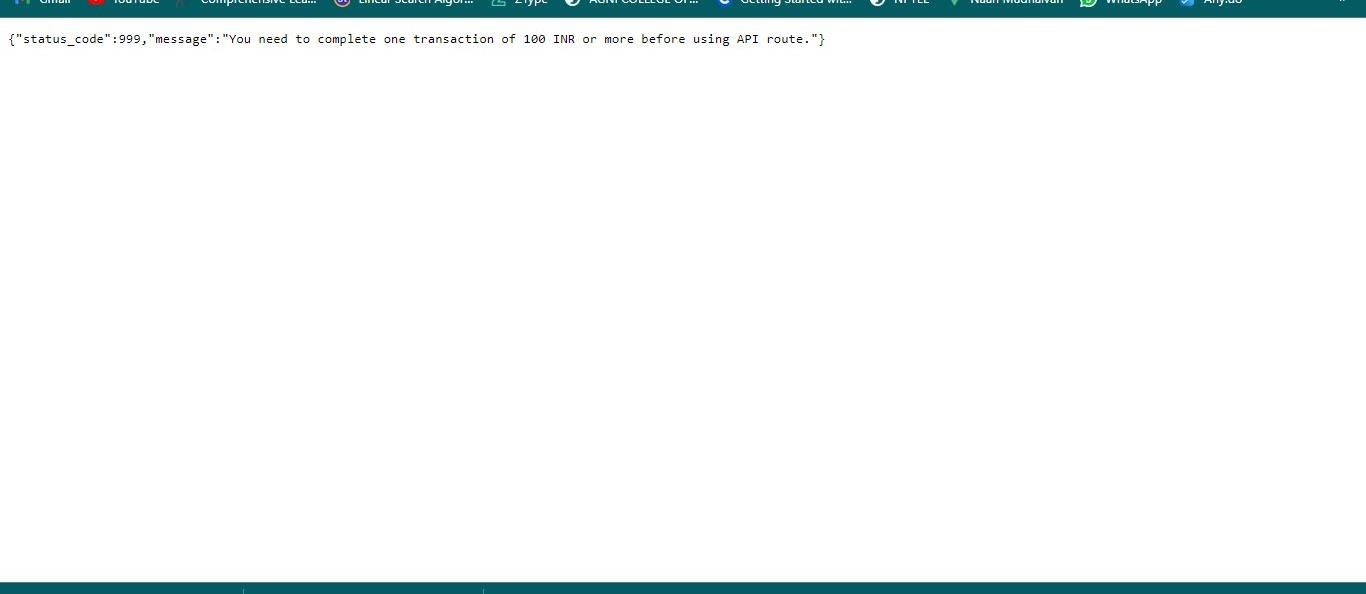




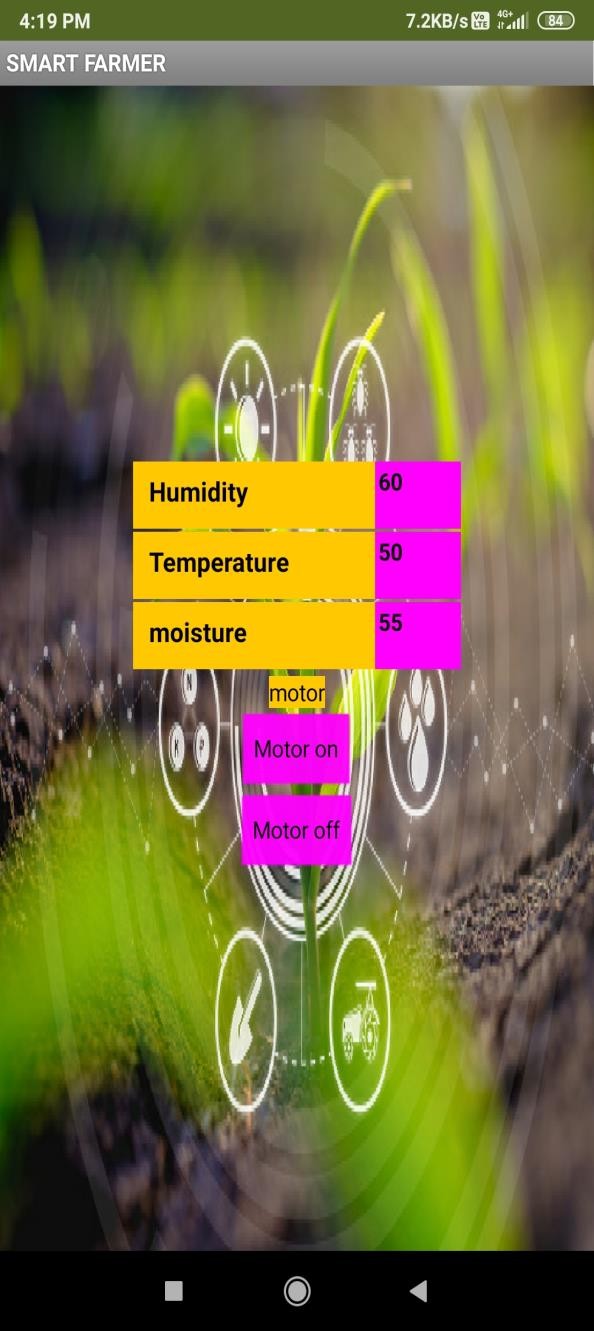
**Mit App Inventor & Fast2SMS**







**Mobile Application**



# 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.

# 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. Environment Humidity

2. Environmental Temperature

3. Soil Moisture

4. Rain Sensing

5. Water Management

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.

# 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.

Pest monitoring system.

* 1. Estimated yield calculator.
  2. Estimated time of cultivation.
  3. Individual cloud management dashboard.

# 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);

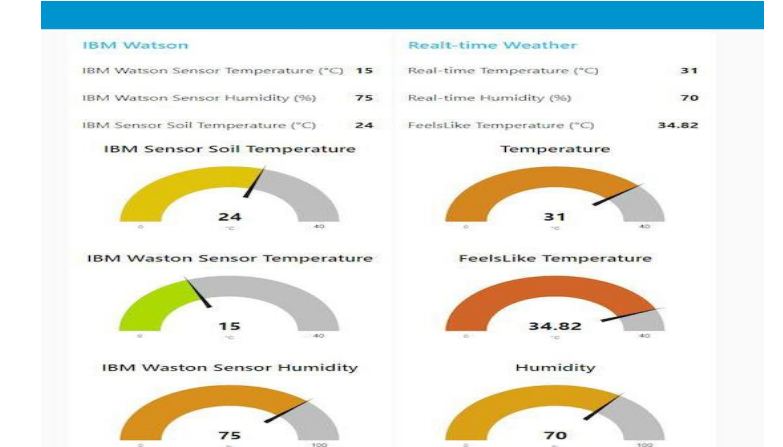
Serial.println();

// 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();

}

}

****

https://github.com/IBM-EPBL/IBM-Project-37586-1660312808

Demo link: https://drive.google.com/file/d/1XLvx-a40gL9lgi-erXGc6PoKuS9wrw5e/view?usp=drivesdk