

PROJECT REPORT

IoT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

S.Gunaseelan¹ P.Premnath² M.TataPravin³ S.Vikram⁴ R.Manikandan⁵

**KARAIKUDI INSTITUTE OF TECHNOLOGY AND KARAIKUDI INSTITUTE OF
MANAGEMENT,
KARAIKUDI.**

1. INTRODUCTION

1.1 Project Overview

This project is based on Internet Of Things (IoT), that can measure soil moisture, Humidity and temperature conditions for agriculture and crop protection using Watson IoT services and also protect crops from animals using PIR sensor and Ultrasonic sensor out of this we can able to find the weather condition also. Measure water level in the tank which store by rain water harvesting and also use ground water by automated and manual process. IoT is network that connects physical objects or things embedded with electronics, software and sensors through network connectivity that collects and transfers data using cloud for communication. Data is transferred through internet without human to human or human to computer interaction.

In this project we have not used any hardware. Instead of real soil moisture, Humidity and Temperature data obtained from sensors and motion detection sensor of ultrasonic and PIR sensor we make use of Simulator which can transmit these parameters as required.

1.2 Purpose

An intelligent crop protection system helps the farmers in protecting the crop from the animals and birds which destroy the crop in unwanted way. This system also helps farmers to monitor the soil moisture levels in the field and also he temperature and humidity values near the field. The motors and sprinklers in the field can be controlled using the mobile application.

2. LITERATURE SURVEY

Journal	Author	Concept
IOT Based Crop-Field Monitoring And Irrigation Automation	S,muthunpandian, S.Vigneshwaran , R.C Ranjitsabarinath , Y.Manoj kumar reddy	This paper has given an automated system has been designed and implemented on for crop field monitoring continuously. The system maintains the water levels in the crop field at power consumption in the crop field. Developed system is useful in the irrigation system.

Automated Irrigation System Using a Wireless Sensor Network and GPRS Module	Joaquin Gutierrez, Juan Francisco Villa-Medina, Alejandra Nieto- Garibay, and Miguel Angel Porta-Gandara	This paper showed automated irrigation system that reduces the water resources more effective by considering the timing of water scarcity. They shown water utilization is minimized and incorporated a solar power system to reduce power consumption. This was developed by smart phone operating by considering the sensors data via internet.
Field Monitoring and Automation using IOT in Agriculture Domain	Mohanraj I Kirthika Ashokumarb, Naren	This paper focuses on monitoring the data in farming cycle. The system contains ATMEL microcontroller based GSM operated sensors are used to monitors wind mill temperature variation PH level of water. After that an Arduino based IOT system are used however when we consider to monitor the huge number of Raspberry pi system is more suitable.
Risk Assessment on Raspberry PI using NIST Standards	Michael G Williams	This paper showed an irrigation system with raspberry pi system. Raspberry pi based systems for home automation, entertainment systems, security. Developing Raspberry systems are more interesting for thrusting environments

2.1 Existing Problem

Agriculture is a field which forms the basis of our economy. Yet it faces a lot of problems in terms of availability of resources, Irrigation, increasing rate of Pesticides, Climatic disasters, Insects which ruin the crops and makes a huge loss this sector. In agriculture water is needed for the crops for their growth. If the Soil gets dry it is necessary to supply water. But sometime if the farmer doesn't visit the field it is not possible to know the condition of soil. Sometimes over supply of water or less supply of water affects the growth of crops. Sometimes if the weather/temperature changes suddenly it is necessary to take certain actions. Specific crops grow better in specific conditions, they may get damaged due to bad weather.

2.2 References

- 1 <https://www.researchgate.net/publication/299447425>
- 2 [\(PDF\) IOT Based Crop Protection System against Birds and Wild Animal Attacks | IJIRT Journal - Academia.edu](#)
- 3 <https://www.researchgate.net/publication/325770625>
- 4 www.reuters.com/article/us-italy-boar/italy-hunts-for-solution-to-wild-boar-emergency-iduskcn0su1jn20151105.
- 5 www.telegraph.co.uk/news/worldnews/europe/italy/12105887/tuscanwine-makers-back-cull-of-250000-wild-boar-and-deer.html.
- 6 www.elsevier.com/locate/fgcs
- 7 [Implementation of IIoT based smart crop protection and irrigation system - IOPscience](#)
- 8 [SMART SECURITY AND MONITORING DEVICE FOR AGRICULTURAL USE | Tanmay Baranwal | 1 publications | Research Project \(researchgate.net\)](#)
- 9 [\(PDF\) Design and Implementation of Automatic Low Cost Organic Fertilizer and Insecticides Making Machine | IJIRT Journal - Academia.edu](#)
- 10 J. Faster R-CNN: Towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. **2015**, 28. Available online: <https://arxiv.org/abs/1506.01497> (accessed on 27 April 2021).
- 11 Sugam Sharma, U S Tim, Shashi Gadia, and Johnny Wong.(2015). “*Growing Cloud Density and as-a-Service Modality and OTH-Cloud Classification in IOT Era*” Available at www.public.iastate.edu/~sugamsha/articles/OTH-Cloud/OTH-Cloud/in/IoT
- 12 <https://wokwi.com/projects/322410731508073042>
- 13 Budikartiwa, yayanapriyana & harissyahbuddin, Indonesia Production and Quality enhancement of mango using fan jet sprayer irrigation technique naniheryani. Indonesian Journal of Agricultural Science Vol. 17 No. 2 October 2016: 41–48 DOI: <http://dx.doi.org/10.21082/ijas.v17n2.2016.p41-48>.
- 14 Gwo-Jiun Horng ; Min-Xiang Liu Chao-Chun Chen ; The Smart Image Recognition Mechanism for Crop Harvesting System in Intelligent IEEE sensors Journals Year: 2020
- 15 Nanda, I., & Adhikari, N. (2019). Accelerator design for ethernet and HDMI IP systems for IoT using xilinx vivado 18.X. International Journal of Innovative Technology and Exploring Engineering, 8(10), 652–656.

2.3 Problem Statement Definition

Defining the Problem:

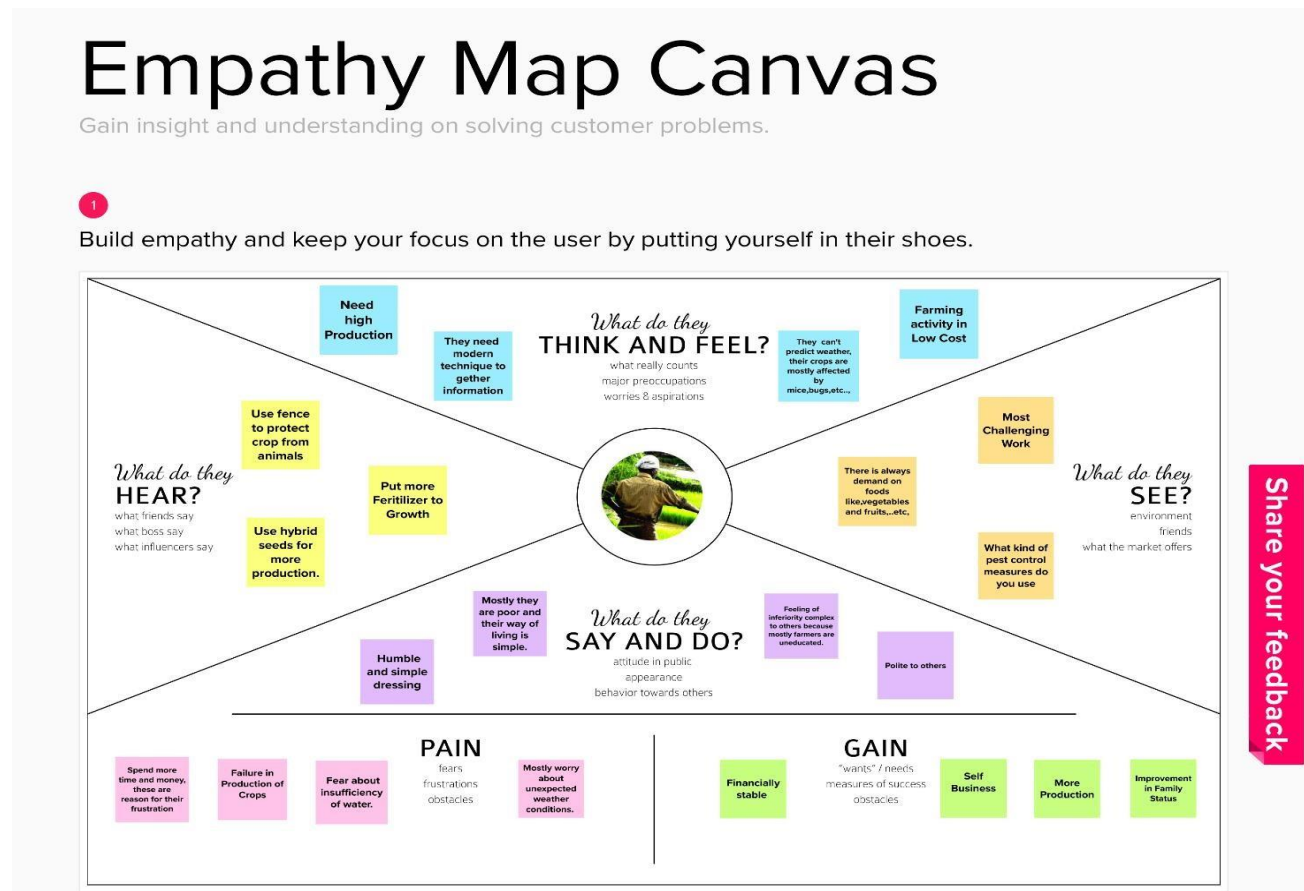
Farmer needs a way to monitor and protect his agricultural land so that he will get a good yield.

Problem Statement:

Ganesh is a busy businessman who needs a way to not spend his time on monitoring, watering the plant and protecting his agricultural land so that he doesn't want to hire labor for these works.

3. IDEATION & PROPOSED SOLUTION:

3.1 Empathy Map Canvas



3.2 Ideation & Brain Storming

What do they think and feel?

As its name may imply, smart farming is the use of technology in animal agriculture, and it's something that's been around since the Industrial Revolution. The biggest difference between then and now, though? "Motorized devices are being replaced with IOT".

What do they hear?

Smart farming is about using the new technologies which have arisen at the dawn of the Fourth Industrial Revolution in the areas of agriculture and cattle production to increase production quantity and quality, by making maximum use of resources and minimizing the environmental.

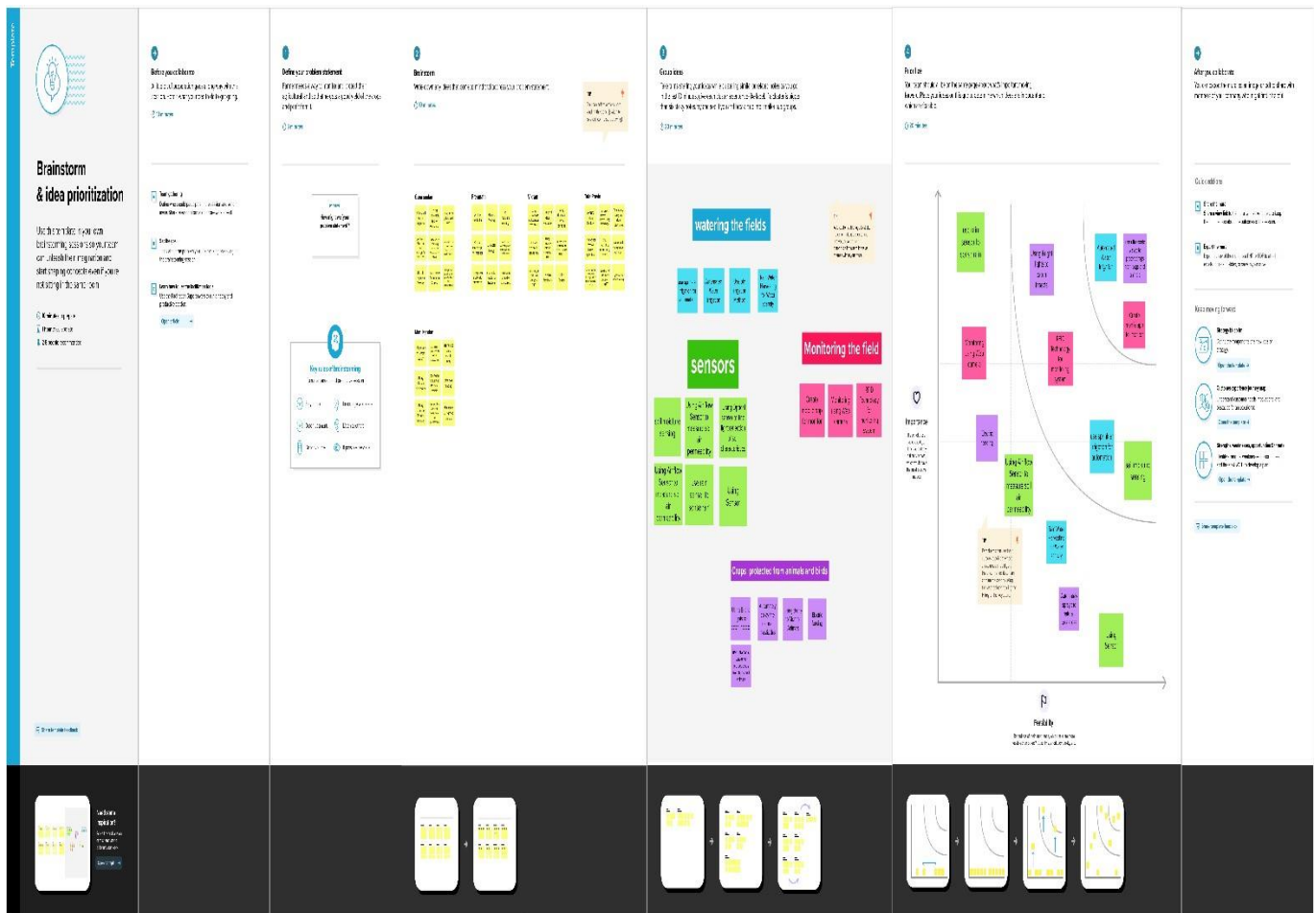
What do they see?

Smart farming is a management concept focused on providing the agricultural industry with the infrastructure to leverage advanced technology – including big data, the cloud and the internet of things (IoT) – for tracking, monitoring, automating and analyzing operations.

What do they say and do?

□ The aim of this technology is to make the most of all the data collected by various tools, by converting them into real sources of information in order to then define ways of simplifying agricultural work. It also allows for accurate and predictive analysis of all situations that may affect the farms, such as weather conditions (temperature, humidity, etc.) and sanitary or economic situations, for example. This makes it easier to organize the supply of energy, water, livestock feed and fertilizer.

□ In its most advanced form, smart farming facilitates the exchange of information between different farms, creating a real network of connected farms accessible from a smartphone .



3.3 Proposed Solution

Defining the Problem:

Farmer needs a way to monitor and protect his agricultural land so that he will get a good yield.

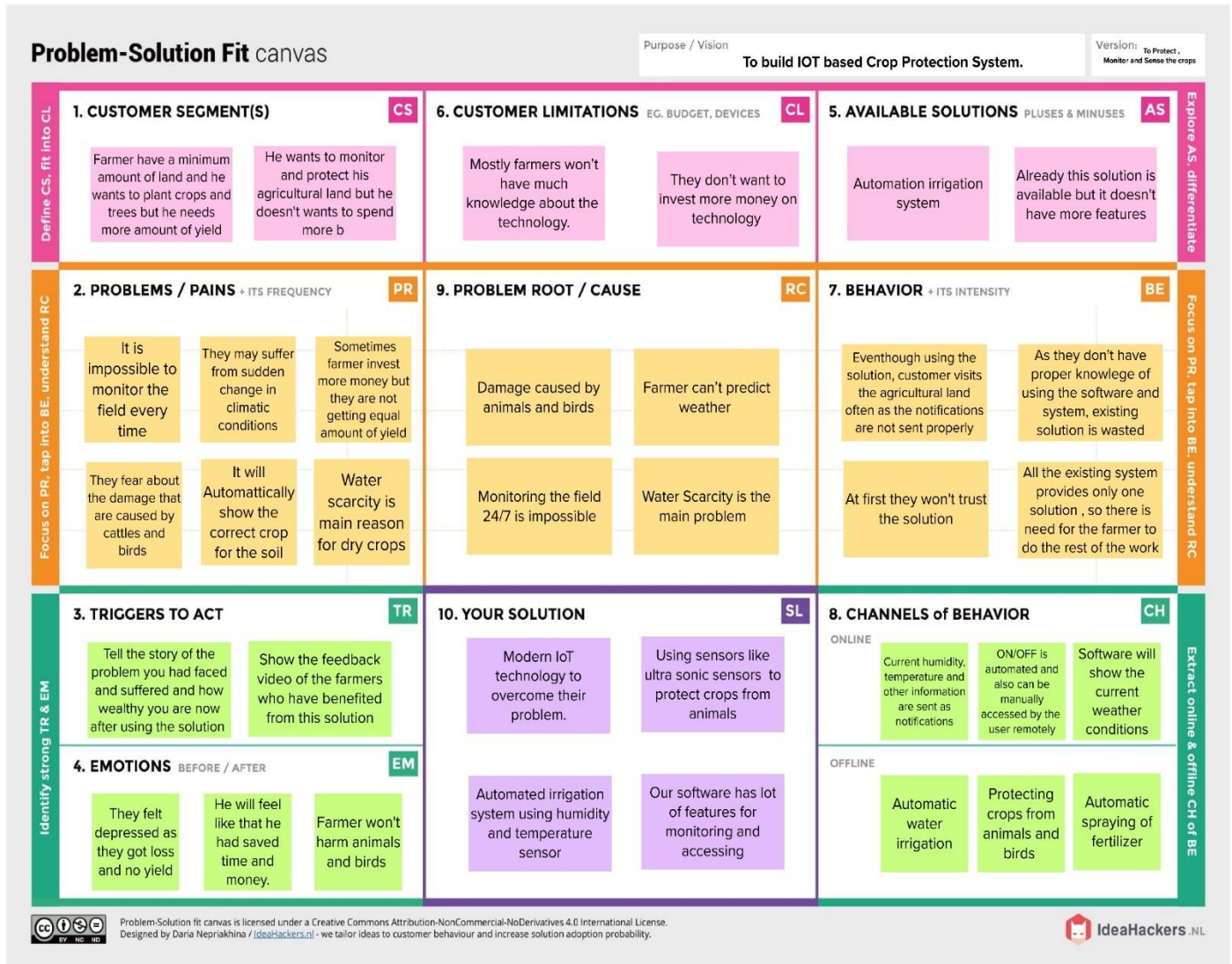
Proposed Solution Template:

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Krishna is a busy businessman who needs a way to spend his time on monitoring, watering the plant and protecting his agricultural land so that he doesn't want to hire labour for these work.
2.	Idea / Solution description	Idea for watering the fields Automated crop protection using sensor Automated water irrigation Use sprinkler irrigation method Rain Water Harvesting for water scarcity Idea using Sensors Soil moisture sensing Use rain sensor to sense rain Idea for Monitoring the field Create mobile app for monitor Using advanced technology for monitoring Ideas for Crops protection from animals and birds Using bright lights to catch insects during night time Automatically spraying the natural pesticides Use ultra-sonic waves to protect crops from bugs and animals

3.	Novelty / Uniqueness	<p>This project helps the farmer to reduce their work and time.</p> <p>While comparing with other method, this is the most efficient and effective method. Here farmer need not monitor his agricultural land every time, everything is automated and can be done manually if needed. It can easily be handled by both educated and uneducated people. This project is not particularly made for killing or disturbing the animals and bugs but it keeps them away from the crop. In this way our system differs from other system.</p>
4.	Social Impact / Customer Satisfaction	<p>Through this project the customer can easily handle, monitor and control all the equipment and sensors used for crop growth and protection.</p> <p>It avoids unnecessary wastage of water to harvest the plants.</p> <p>Compared to the normal farming techniques it is more accurate and secure. The customer also gets notified about each and every actions happening in the agricultural land, so need for the customer to visit the agricultural land often.</p> <p>Compared to normal agriculture we can spent minimum of cost and control at anywhere at any time. It improves protection from animals and insects. It gives high production.</p>
5.	Business Model (Revenue Model)	<p>As this project requires less manpower which leads to spending less amount and increases gain.</p> <p>Compared to human being, sensors protects the land efficiently, so there will no chance for damage of crops which leads to high yield and profit.</p>
6.	Scalability of the Solution	<p>As every farmer is nowadays suffering from loss, our project will definitely help them. This project can be taken to regional and national level, as there is need for such project.</p>

3.4 Problem 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	Software Installation	Installing the software in customer mobile phone
FR-2	Land Enquiry	Enquires land size Enquires type of planting Enquires surroundings

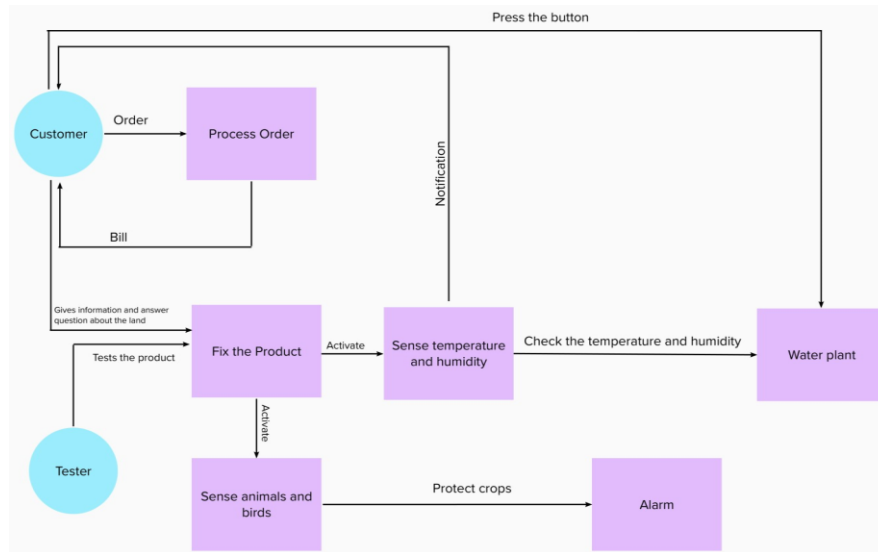
FR-3	Monitoring Field	Monitors weather Monitors movement of cattle and birds
FR-4	Weather Prediction	Predicts weather using temperature sensor
FR-5	Watering Plant	Watering the plants automatically Watering the plants manually using app Watering the plants using Rain water harvesting Watering the plants using ground water
FR-6	Notification	Sends notification to customer mobile phone by app

4.2 Non Functional Requirements

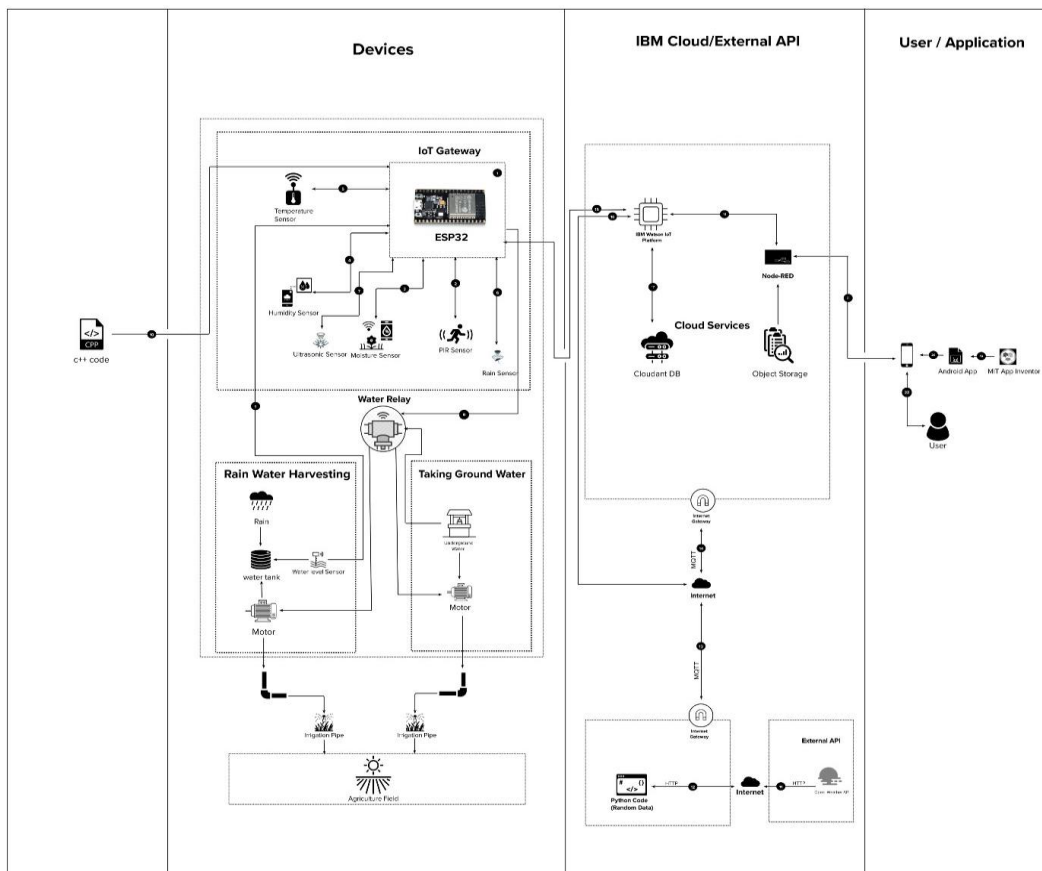
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	People with no understanding of English must be able to use the product
NFR-2	Security	Access permission of the software installed on the customer mobile can only be changed by the customer
NFR-3	Reliability	The software must send notification to the user if any failure occurs in the hardware
NFR-4	Performance	The system should send the notification through internet within 5 seconds
NFR-5	Availability	Adding new sensor to the existing system should not affect the existing sensor
NFR-6	Scalability	The system must be scalable to be implemented up to 20 acres land

5.1 Data Flow Diagram



5.2 Solution & Technical Architecture



Guidelines:

1. Connects all the IOT devices
2. Indicates sensor for motion detection
3. Indicates sensor for detecting soil moisture
4. Indicates sensor for detecting the humidity
5. Indicates sensor for detecting the temperature
6. Indicates sensor for predicting rain
7. Indicates sensor for detecting the motion range
8. Indicates water relay for switching on the motor
9. Indicates sensor for detecting the water level in tank
10. Indicates code for written for Arduino Uno for functioning of all the sensor
11. Indicates request sent to external API through HTTP protocol
12. Indicates response from external API through HTTP protocol
13. Indicates internet connection through MQTT protocol
14. Indicates internet connection through MQTT protocol
15. Indicates publish and subscribe of data to IBM Watson IoT platform
16. Indicates publish and subscribe of data to IBM Watson IoT platform
17. Indicates IBM Watson IoT platform data stored in Cloudant DB
18. Connection between IBM Watson IoT Platform and Node-Red
19. Indicates Android App created using MIT App Inventor
20. Indicates Android App installed in Mobile Phone
21. Indicates connection between Node-Red and Mobile Phone
22. Indicates user using Mobile phone

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Mobile App	Android
2.	Application Logic-1	Logic for sensing the motion detection	Python/C++
3.	Application Logic-2	Logic for sensing the soil moisture	Python/C++
4.	Application Logic-3	Logic for sensing the humidity	Python/C++
5.	Application Logic-4	Logic for sensing the temperature	Python/C++
6.	Application Logic-5	Logic for switching on the motor if the moisture is low	Python/C++
7.	Application Logic-6	Logic for sensing the water level in the tank	Python/C++
8.	Application Logic-7	Logic for sending message to the user mobile	Python/C++
9.	External API-1	Purpose of the API is to access the current weather data	Weather API

10.	Cloud Database	Database Service on Cloud	IBM Cloudant DB
11.	File Storage	Storing the data	Object Storage
12.	Programming tool	Purpose of the tool is for wiring hardware and APIs	Node-Red

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Node-Red	IBM Cloud
2.	Scalable Architecture	3-tier Architecture	Technology used
3.	Performance	The client sends and receives two or more messages per second	5G Network

5.3 User Stories

User Type	Functional Requirements (Epic)	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Customer	Login	USN-1	As a user, I can login to the Android app using username and password created by seller for me	I can only access my app	High	Sprint-1
Customer	Watering plant	USN-2	As a user, I can water the plant by pressing the button in the application installed in the mobile	I can also control the watering of plants	High	Sprint-2
		USN-3	As a user, I automate the process of watering the plant	I need not do any work	Medium	Sprint-2
		USN-4	As a user, I can water the plant using Rain Water Harvesting	I can save the Ground Water	High	Sprint-2
		USN-5	As a user, I can water the plants using the Ground Water if the water saved in the tank is empty	I can also use the Ground Water if needed to water the plant	Medium	Sprint-2
	Monitoring Field	USN-6	As a user, I can monitor the weather condition	I will know about the current weather condition	High	Sprint-3

		USN-7	As a user, I can monitor the humidity	I will know about the current humidity	Low	Sprint-3
		USN-8	As a user, I can monitor the temperature	I will know about the current temperature	Low	Sprint-3
		USN-9	As a user, I can monitor the soil moisture	I can decide whether to water the plants or not	High	Sprint-3
		USN-10	As a user, I can monitor the movement of animals and birds in the agricultural land	I can protect the land	High	Sprint-3
		USN-11	As a user, I can monitor the water level in the Rain Water Harvesting tank	I can decide whether to water the plant using Rain Water Harvesting or Ground Water	High	Sprint-3
	Rain Prediction	USN-12	As a user, I can predict the rain using temperature	I can decide whether to water the plants or not	Low	Sprint-4
	Information	USN-13	As a user, I can get information through Mobile App	I will get Information if water level is low in the Rain Water Harvesting tank	Low	Sprint-4

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	Login	USN-1	As a user, I can login to the Android app using username and password created by seller for me	3	High	Premnath
Sprint-2	Watering plant	USN-2	As a user, I can water the plant by pressing the button in the application installed in the mobile	4	High	Gunaseelan

Sprint-2		USN-3	As a user, I automate the process of watering the plant	4	Medium	Gunaseelan
Sprint-2		USN-4	As a user, I can water the plant using rain water harvesting	2	High	Gunaseelan
Sprint-2		USN-5	As a user, I can water the plants using the ground water if the water saved in the tank is empty	4	Medium	Gunaseelan
Sprint-3	Monitoring Field	USN-6	As a user, I can monitor the weather condition	2	High	Manikandan
Sprint-3		USN-7	As a user, I can monitor the humidity	3	Low	Tata Pravin
Sprint-3		USN-8	As a user, I can monitor the temperature	2	Low	Manikandan
Sprint-3		USN-9	As a user, I can monitor the soil moisture	3	High	Tata Pravin
Sprint-3		USN-10	As a user, I can monitor the movement of animals and birds in the agricultural land	3	High	Manikandan
Sprint-3		USN-11	As a user, I can monitor the water level in the Rain Water Harvesting tank	3	High	Gunaseelan
Sprint-4	Rain Prediction	USN-12	As a user, I can predict the rain using temperature	2	Low	Vikram
Sprint-4	Information	USN-13	As a user, I can get information through Android App	2	High	Premnath

6.2 Sprint Delivery Schedule

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	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	04 Nov 2022	20	04 Nov 2022
Sprint-3	20	6 Days	05 Nov 2022	10 Nov 2022	20	10 Nov 2022
Sprint-4	20	6 Days	11 Nov 2022	16 Nov 2022	20	16 Nov 2022

6.3 Reports from JIRA

		NOV	DEC	JAN
Sprints	IBCP...	IBCP... IBCP... IBCP...		
▸ IBCPSFA-15 Monitoring Field				
▸ IBCPSFA-17 Login				
▸ IBCPSFA-19 Watering Plant	DONE			
▸ IBCPSFA-20 Rain Prediction	DONE			
▸ IBCPSFA-34 Information	DONE			

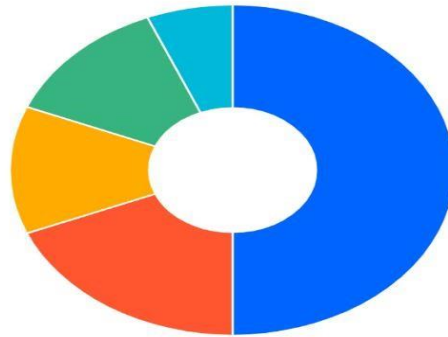
		T		
Sprints			IBCP...	IB...
▾ IBCPSFA-17 Login				
IBCPSFA-21 As a user, I can login to the Android app using username and password created by seller for me	DONE	CREATIVEPR...		

		T		
Sprints			IBCP...	IB...
▾ IBCPSFA-19 Watering Plant	DONE			
IBCPSFA-23 As a user, I automate the process of watering the plant	DONE	GUNASEEL...		
IBCPSFA-22 As a user, I can water the plant by pressing the button in the application installed in the mobile	DONE	GUNASEEL...		
IBCPSFA-24 As a user, I can water the plant using Rain Water Harvesting	DONE	GUNASEEL...		
IBCPSFA-25 As a user, I can water the plants using the Ground Water if the water saved in the tank is empty	DONE	GUNASEEL...		

		T		
Sprints			IBCP...	IB...
▾ IBCPSFA-15 Monitoring Field				
IBCPSFA-26 As a user, I can monitor the weather condition	DONE	RBMANIKA...		
IBCPSFA-27 As a user, I can monitor the humidity	DONE	TATAPRAVIN		
IBCPSFA-28 As a user, I can monitor the temperature	DONE	RBMANIKA...		
IBCPSFA-30 As a user, I can monitor the movement of animals and birds in the agricultural land	DONE	RBMANIKA...		
IBCPSFA-29 As a user, I can monitor the soil moisture	DONE	TATAPRAVIN		
IBCPSFA-34 As a user, I can monitor the water level in the Rain Water Harvesting tank	DONE	GUNASEEL...		

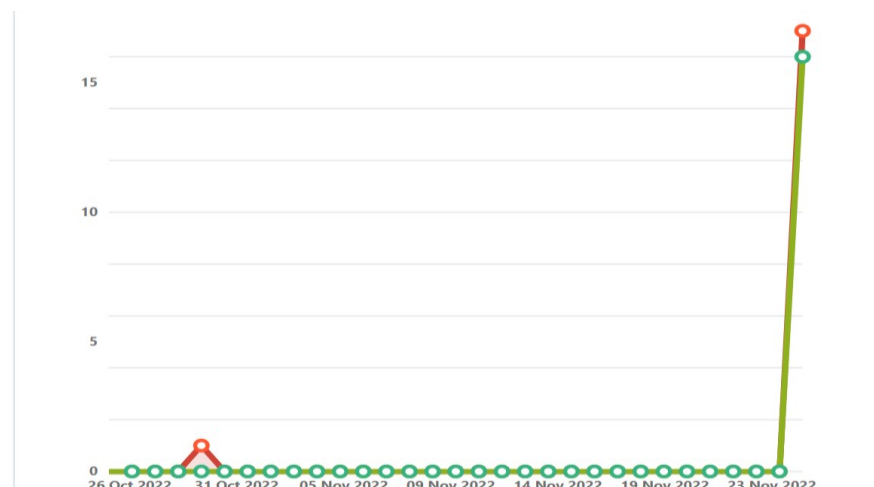
		T		
Sprints			IBCP...	IB...
▾ IBCPSFA-20 Rain Prediction	DONE			
IBCPSFA-32 As a user, I can predict the rain using temperature	DONE	VIKRAMSET...		
▾ IBCPSFA-34 Information	DONE			
IBCPSFA-33 As a user, I can get information through Android App	DONE	CREATIVEPR...		

Pie Chart: workload

**Assignee**

Total Issues: 16

Gunaseelan	8
rbmanikandan03	3
Tatapraavin	2
creativepremnath	2
vikramsethupathi2001	1



7. CODING & SOLUTIONING:

CODING:

Wokwi Code

```
#include "SPI.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h" #include
"DHTesp.h"
#include "WiFi.h"//library for wifi
#include "PubSubClient.h"//library for MQTT
#include "HTTPClient.h"
#include "Arduino_JSON.h"
```

```
#define PIN_TRIG 27
#define PIN_ECHO 26
```

```

#define TFT_DC 2
#define TFT_CS 15
#define SOUND_SPEED 0.034

const int DHT_PIN = 13;
int pirpin = 12;
int BUZZER_CHANNEL = 0;
int ultrabuzzer=33; int
temperaturepin=35; const
float BETA = 3950; int
pirState = LOW; String
payload; const String
endpoint =
"https://api.openweathermap.org/data/2.5/weather?lat=10.0695426&lon=78.7642444&appid="; const
String key = "dff65ece7cd56e5609a961090c1815a5";

DHTesp dhtSensor;
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

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

#define ORG "o18g4e"
#define DEVICE_TYPE "temp"
#define DEVICE_ID "1234"
#define TOKEN "12345678"
String data3;
float h, t;

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/event_1/fmt/json"; char
subscribtopic[] = "iot-2/cmd/command/fmt/String"; char
authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);

void setup() { Serial.begin(9600);
dhtSensor.setup(DHT_PIN, DHTesp::DHT22); tft.begin();
ledcSetup(BUZZER_CHANNEL, 8000, 12); pinMode(PIN_TRIG,
OUTPUT); pinMode(PIN_ECHO, INPUT);
ledcAttachPin(ultrabuzzer, BUZZER_CHANNEL);
analogReadResolution(10); pinMode(temperaturepin,INPUT);
wificonnect(); mqttconnect();
}

```

```

int value=0; void loop() {
  HTTPClient http;
  http.begin(endpoint + key);
  int httpCode = http.GET();
  if (httpCode > 0) {
    String payloadweather = http.getString();
    Serial.println(httpCode);
    Serial.println("Http Status Code: "+httpCode);
    Serial.println(payloadweather);
    JSONVar myObject = JSON.parse(payloadweather);
    if (JSON.typeof(myObject) == "undefined") {
      Serial.println("Parsing input failed!"); return;
    }
    Serial.print("Area: ");
    Serial.println(myObject["name"]);
    Serial.print("Weather_Details: ");
    Serial.println(myObject["weather"]);
  }

  else {
    Serial.println("Error on HTTP request");
  }
  http.end(); //Free the resources
  TempAndHumidity data = dhtSensor.getTempAndHumidity();
  float t=data.temperature; float
  h=data.humidity; payload =
  "{"temperature\":"; payload
  += t; payload += ","
  "\"Humid\":"; payload += h;
  payload += ","; if
  (!client.loop()) {
    mqttconnect();
  }

  int state = digitalRead(pirpin);
  if (state == HIGH) { if
  (pirState == LOW)
  {
    tft.fillScreen(ILI9341_BLACK);
    tft.setTextColor(ILI9341_RED);
    tft.setTextSize(3); tft.setCursor(0,
    120); tft.println("Motion
    Detected"); pirState = HIGH;
    ledcWriteTone(BUZZER_CHANNEL,500);
    payload += "\"PIR_Sensor\":"; payload
    += "\"Motion Detected\""; payload += ",";

```

```

} } else if (pirState == HIGH) {
tft.fillScreen(ILI9341_BLACK);
tft.setTextColor(ILI9341_RED);
tft.setTextSize(3);
tft.setCursor(0, 120);
tft.println("Motion ended");
ledcWrite(BUZZER_CHANNEL, 0);
pirState = LOW; payload +=
"\\"PIR_Sensor\\": "; payload
+="\\"Motion Ended\\""; payload +=
"; "; } else { payload +=
"\\"PIR_Sensor\\": "; payload +=\\"No
Motion Detected\\"";
payload += "; ";
}

```

```

digitalWrite(PIN_TRIG, HIGH);
delayMicroseconds(10);
digitalWrite(PIN_TRIG, LOW); int duration
= pulseIn(PIN_ECHO, HIGH); int distance
= duration * SOUND_SPEED/2;
delay(1000); if(distance<100) {
tft.fillScreen(ILI9341_BLACK);
tft.setTextColor(ILI9341_RED);
tft.setTextSize(3);
tft.setCursor(1, 120);
tft.println("Motion Detected in
100 cm range");
ledcWriteTone(BUZZER_CHA
NNEL,300);

```

```

payload += "\\"Ultrasonic_Sensor\\": "; payload +=
"\\"Motion Detected in 100cm range\\""; payload +=
"; ";
} else
{
ledcWrite(BUZZER_CHANNEL, 0);
tft.fillScreen(ILI9341_BLACK); payload
+= "\\"Ultrasonic_Sensor\\": "; payload +=
"\\"No Motion Detected\\"";
payload += "; ";
}
int analogValue = analogRead(temperaturepin);
float celsius = 1 / (log(1 / (1023. / analogValue - 1)) / BETA + 1.0 / 298.15) - 273.15;
if(celsius>10 && celsius<=30)
{
tft.fillScreen(ILI9341_BLACK); tft.setTextColor(ILI9341_RED);

```

```

tft.setTextSize(3); tft.setCursor(26,
120); tft.print(celsius); tft.print("
Celsius");
tft.setTextColor(ILI9341_GREEN);
tft.print(" Normal Rain"); payload
+= "\"Rain_Sensor\":"; payload +=
 "\"Normal Rain\":"; payload += "}";
}
else if(celsius>30 && celsius<=50)
{
tft.fillScreen(ILI9341_BLACK); tft.setTextColor(ILI9341_RED);
tft.setTextSize(3); tft.setCursor(26,
120); tft.print(celsius); tft.print("
Celsius");
tft.setTextColor(ILI9341_GREEN);
tft.print(" Drizzling"); payload +=
 "\"Rain_Sensor\":"; payload +=
 "\"Drizzling\":"; payload += "}";
}
else if(celsius<=10)
{
tft.fillScreen(ILI9341_BLACK);
tft.setTextColor(ILI9341_RED);
tft.setTextSize(3); tft.setCursor(26,
120); tft.print(celsius); tft.print("
Celsius");
tft.setTextColor(ILI9341_GREEN);
tft.print(" Heavy Rain"); payload
+= "\"Rain_Sensor\":"; payload +=
 "\"Heavy Rain\":"; payload += "}";
} else
{
tft.fillScreen(ILI9341_BLACK);
tft.setTextColor(ILI9341_RED);
tft.setTextSize(3); tft.setCursor(26,
120); tft.print(celsius); tft.print("
Celsius");
tft.setTextColor(ILI9341_GREEN);
tft.print(" No Rain"); payload +=
 "\"Rain_Sensor\":"; payload +=
 "\"No Rain\":"; payload += "}"; }
publish(payload);
//delay(100); } 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++) {
data3 += (char)payload[i]; Serial.println("data:
"+ data3); if(data3=="motoron")
{
Serial.println(data3); tft.fillScreen(ILI9341_BLACK);
tft.setTextColor(ILI9341_RED);
tft.setTextSize(3); tft.setCursor(26,
120);
tft.println("Motor on by farmer");
} else
{
Serial.println(data3); tft.fillScreen(ILI9341_BLACK);
tft.setTextColor(ILI9341_RED);
tft.setTextSize(3); tft.setCursor(26,
120);

tft.println("Motor off by farmer");
} data3="";
}
}

```

```

}
void publish(String payload)
{

mqttconnect();
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");
Serial.println(client.publish(publishTopic, (char*) payload.c_str()));
} delay(10);
}

```

Python code:

```

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
import requests,json

#Provide your IBM Watson Device Credentials
organization = "o18g4e" deviceType = "rain"
deviceId = "12345" authMethod = "token"
authToken = "RJXd1kAhbObGRo)U(p"
complete_url="https://api.openweathermap.org/data/2.5/weather?lat=10.069542&lon=78.7642445&appid
=dff65ece7cd56e5609a961090c1815a5"

# Initialize GPIO
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
status=cmd.data['command'] if status=="motoron":
print ("Motor is on") elif status == "motoroff": print
("Motor is off") else :
print ("please send proper command")

try:
deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method":
authMethod, "auth-token": authToken} deviceCli = ibmiotf.device.Client(deviceOptions)
#.....

except Exception as e:

```

```

        print("Caught exception connecting device: %s" % str(e))
        sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting"
10 times

deviceCli.connect()

while True:

    motorstatus = ""
    waterlevel=random.randint(1,20)
    Waterusing=""    if waterlevel < 2:
        waterusing =str(waterlevel) + " feet Water Level Low"
    else:
        waterusing =str(waterlevel) + " feet Using rain water"

    soilmoisture=random.randint(0,872)    if
    soilmoisture <200:
        motorstatus="Motor on automatically"

    else:
        motorstatus="Motor off automatically"

    response = requests.get(complete_url)
    x = response.json()
    if x["cod"] != "404":
        y=x["name"]        z
        = x["weather"]

        weather_description = z[0]["description"]
        data = { 'Waterlevel' : waterusing, 'SoilMoisture': soilmoisture, 'MotorStatus':motorstatus,
        'Area':str(y),'WeatherDescription':str(weather_description) }

    #print data    def myOnPublishCallback():    print ("Published Waterlevel = %s " % waterusing,
    "SoilMoisture = %s %%" % soilmoisture, "MotorStatus = %s" % motorstatus," Area="+str(y)," Weather
    description : " +str(weather_description), "to IBM Watson")
    success = deviceCli.publishEvent("event_1", "json", data, qos=0,
    on_publish=myOnPublishCallback)    if not success:

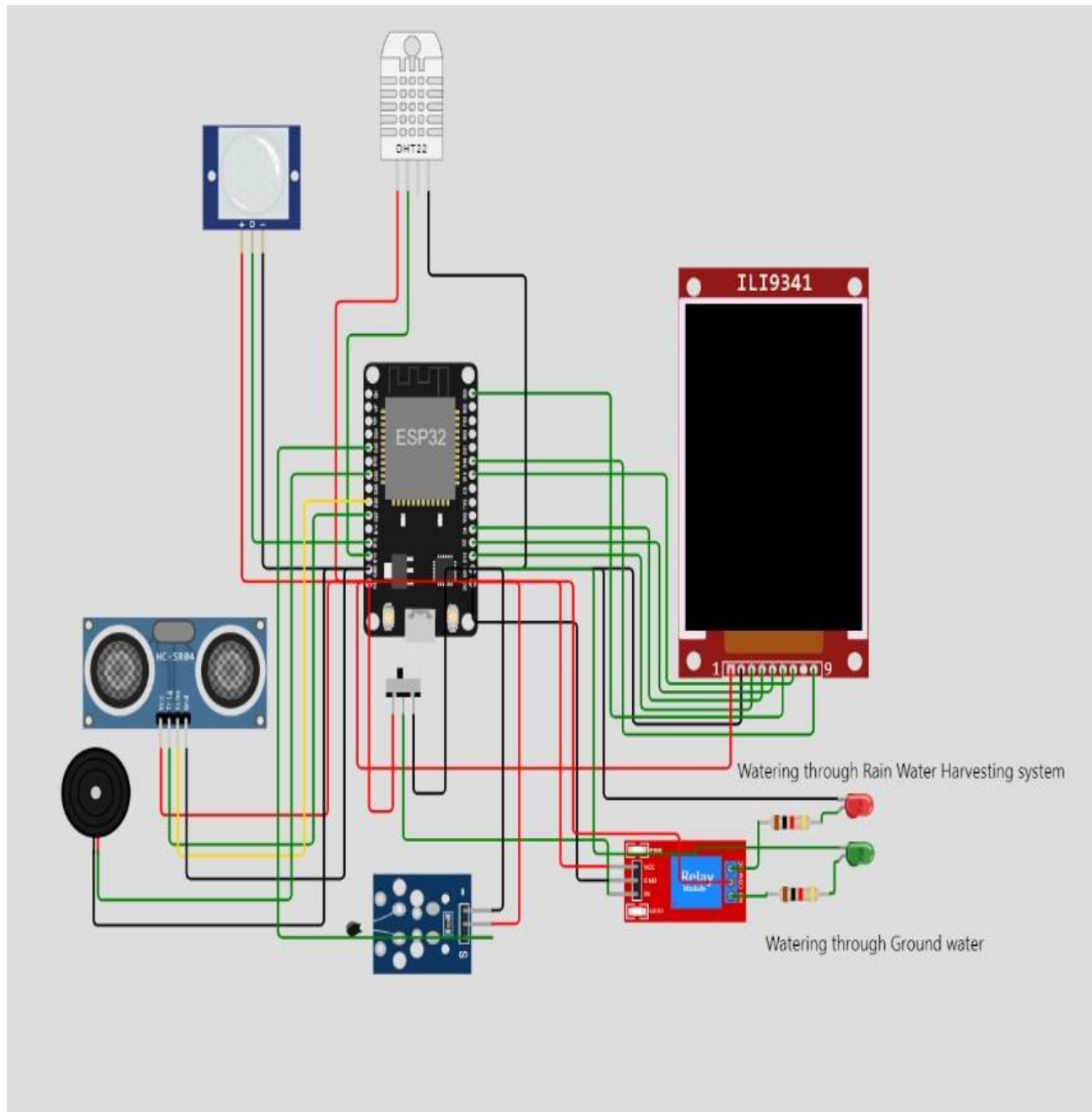
        print("Not connected to IoTF")    time.sleep(10)

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud deviceCli.disconnect()

```


SOLUTION:

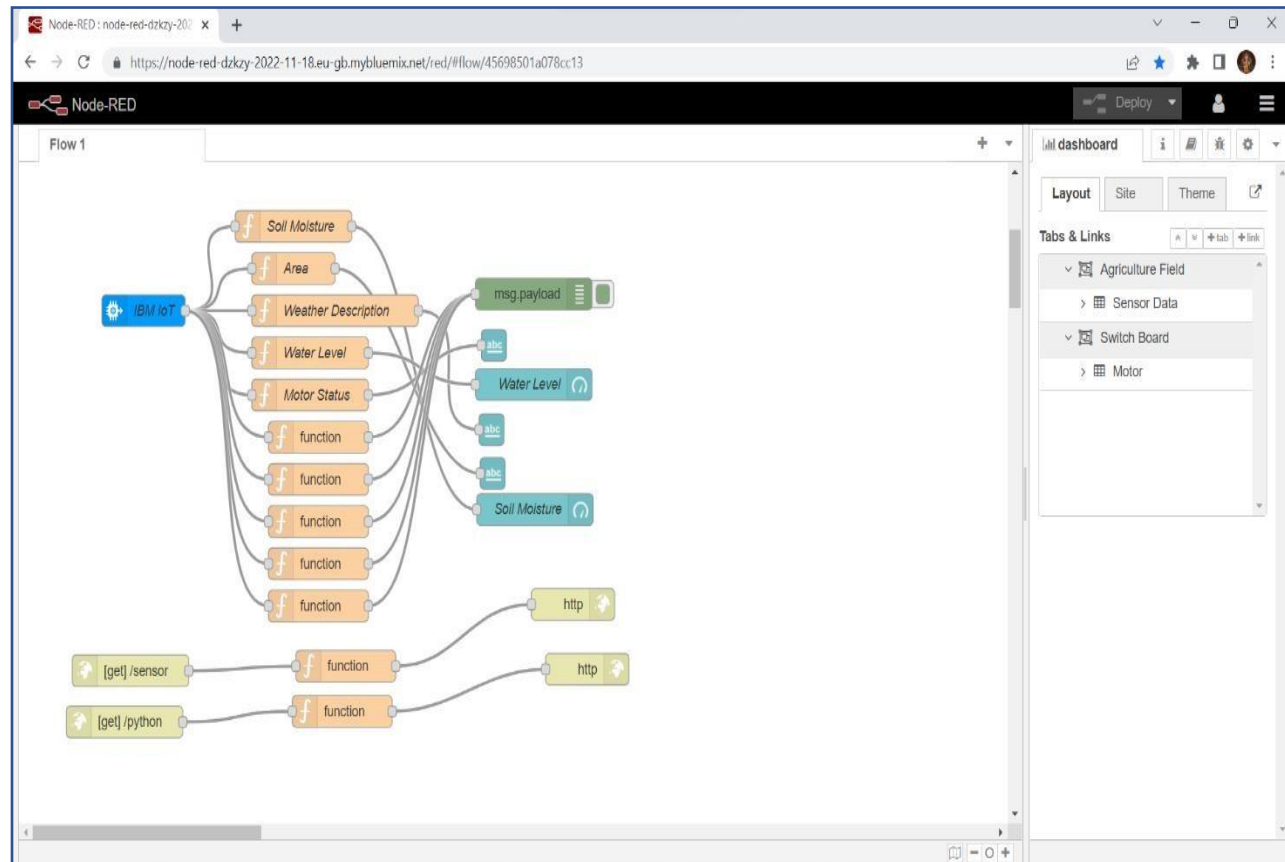
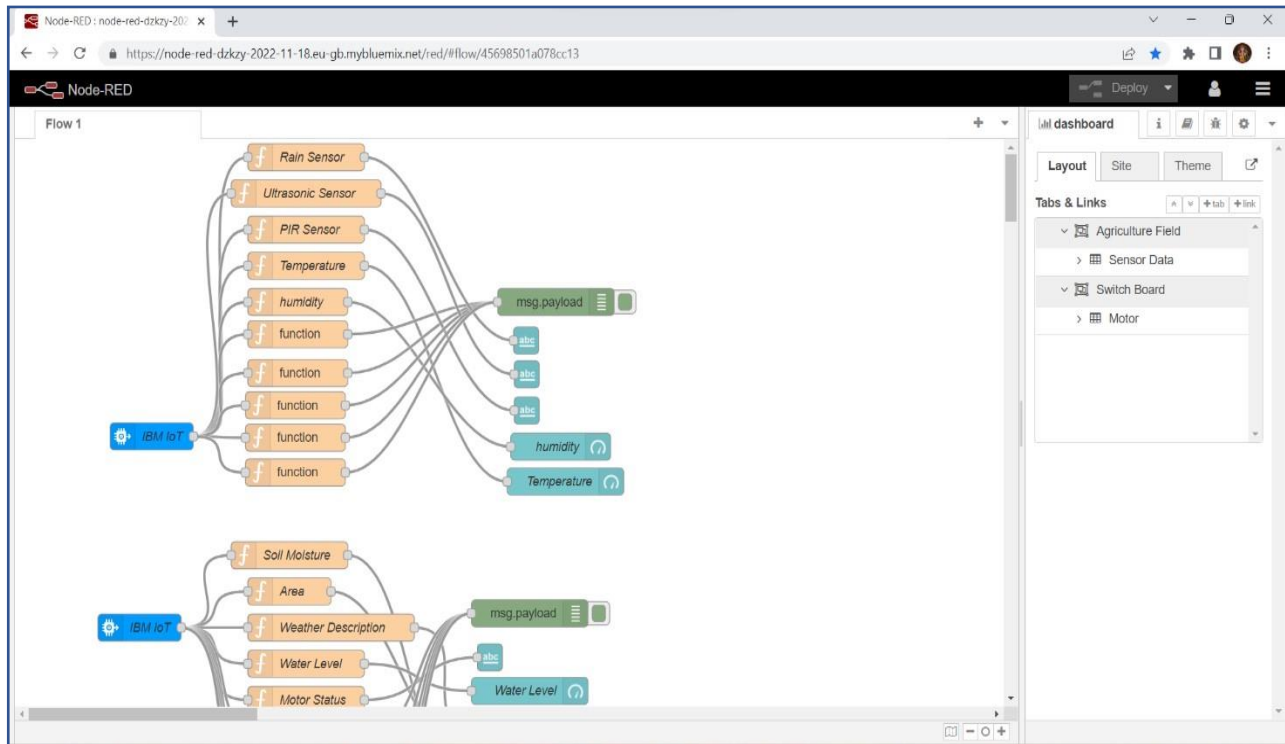


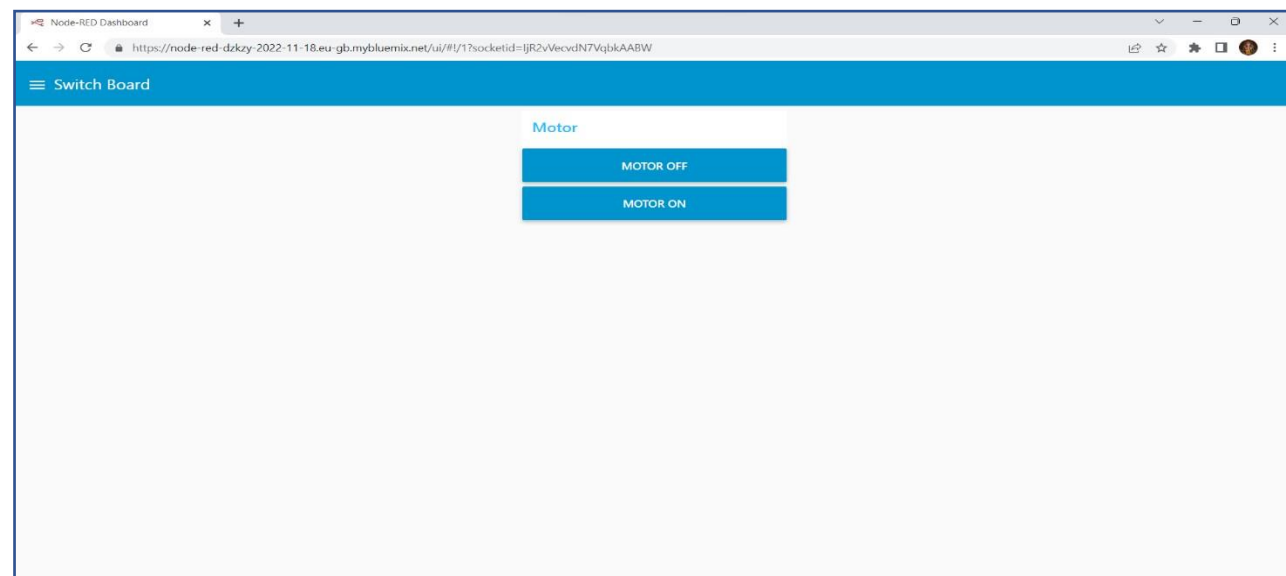
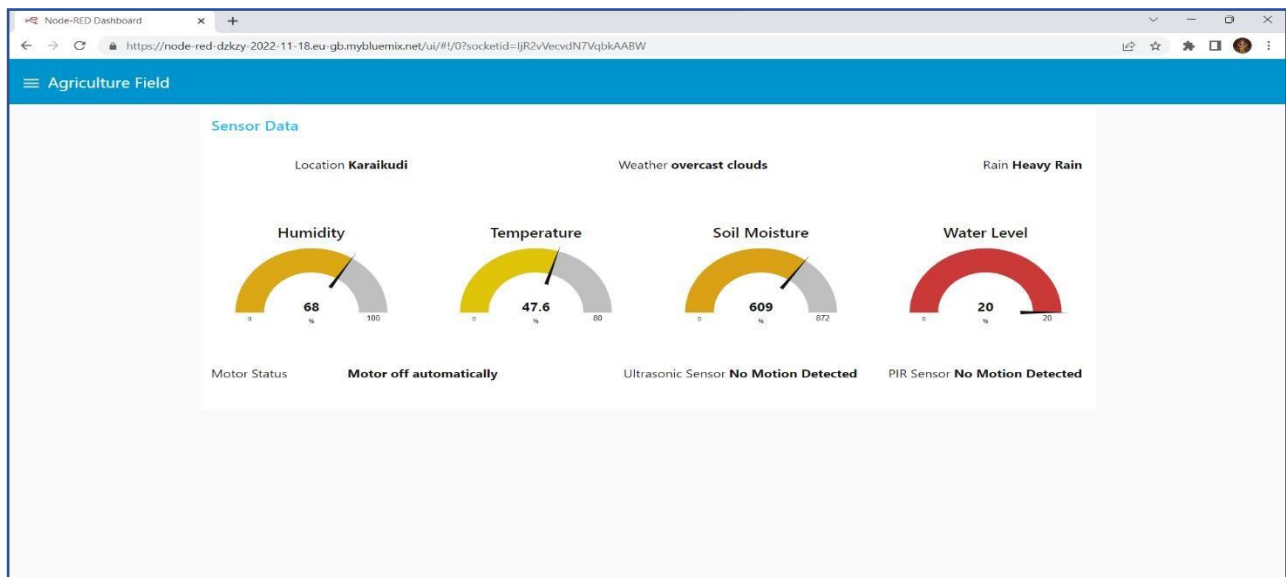
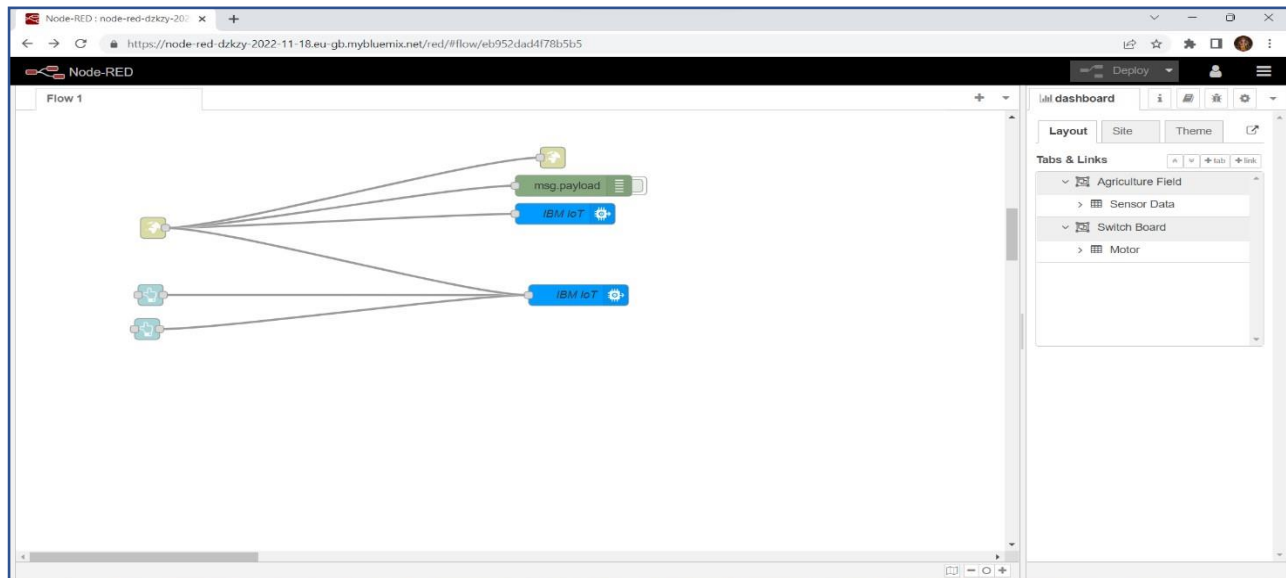
These devices are used to frame to build project

Devices

- ☐ PIR Sensor
- ☐ Ultrasonic Sensor
- ☐ Rain Sensor
- ☐ Water Relay
- ☐ Wifi Module (ESP32)
- ☐ Temperature and Humidity Sensor (DHT22) and etc.

Node-Red Layout:





7.1 Feature 1

Wokwi to Simulate IoT devices

WOKWI

SAVE

SHARE

Docs

esp32-ili9341-hello.ino

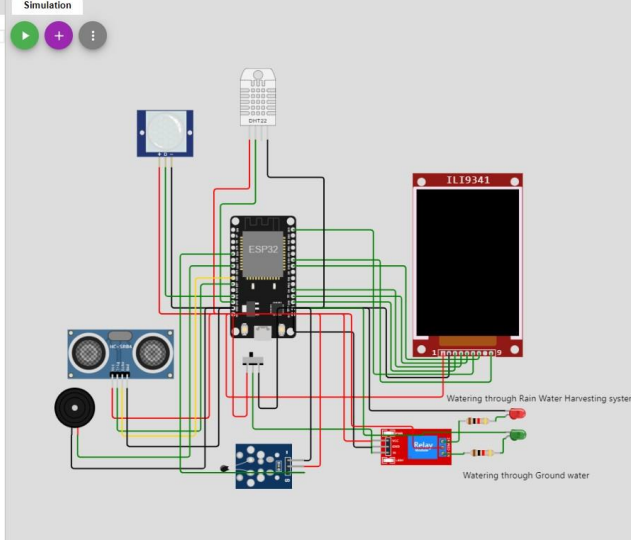
diagram.json

libraries.txt

Library Manager

```
1 #include "SPI.h"
2 #include "Adafruit_GFX.h"
3 #include "Adafruit_ILI9341.h"
4 #include "DHTesp.h"
5 #include "WiFi.h"//library for wifi
6 #include "PubSubClient.h"//library for MQTT
7 #include "HTTPClient.h"
8 #include "Arduino_JSON.h"
9
10 #define PIN_TRIG 27
11 #define PIN_ECHO 26
12 #define TFT_DC 2
13 #define TFT_CS 15
14 #define SOUND_SPEED 0.034
15
16 const int DHT_PIN = 13;
17 int pirpin = 12;
18 int BUZZER_CHANNEL = 0;
19 int ultrabuzzer=33;
20 int temperaturepin=35;
21 const float BETA = 3950;
22 int pirstate = LOW;
23 String payload;
24
25
26
27
28 const String endpoint = "https://api.openweathermap.org/data/2.5/weather?lat=10.2776&lon=78.5424&appid=";
29 const String key = "dfff6Sece7cd56e509a961890c1815a5";
30
31 DHTesp dhtSensor;
32 Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
33
34 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
35
36 #define ORG "o18g4e"//IBM ORGANITION ID
37 #define DEVICE_TYPE "temp"//Device type mentioned in ibm watson IOT Platform
38 #define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
```

Simulation



Using IBM Cloud IoT Platform to store payload

⚡

🔒

⚙️

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
event_1	{"Waterlevel": "10 feet Using rain water", "SoilMoi...	json	a few seconds ago
event_1	{"Waterlevel": "10 feet Using rain water", "SoilMoisture": 670, "MotorStatus": "Motor off automatically", "Area": "Karakudi", "WeatherDescription": "overcast clouds"} {"Waterlevel": "14 feet Using rain"		

⚡

🔒

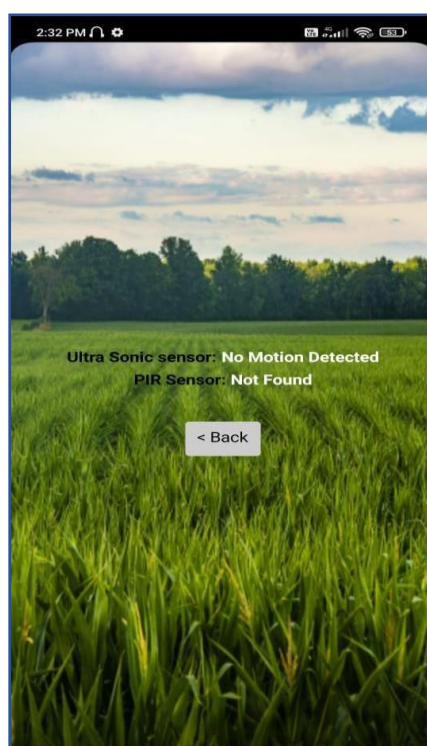
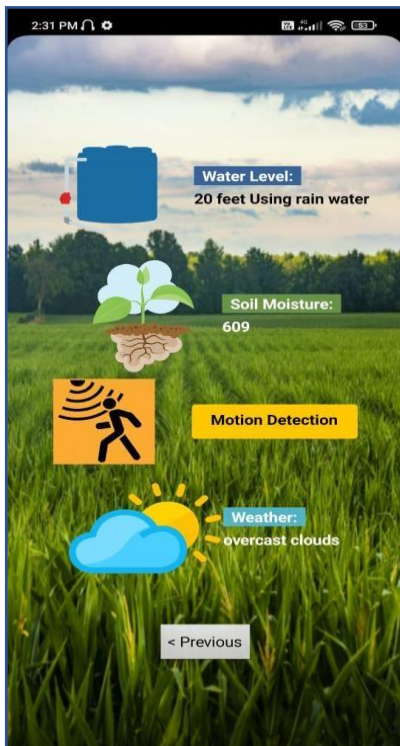
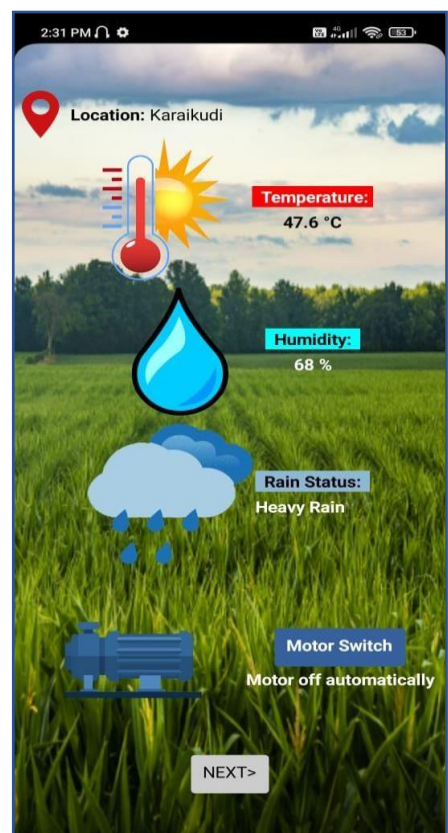
⚙️

The recent events listed show the live stream of data that is coming and going from this device.

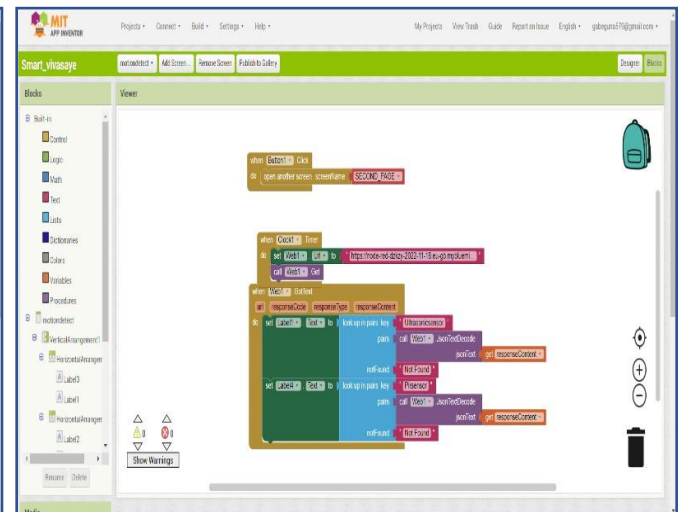
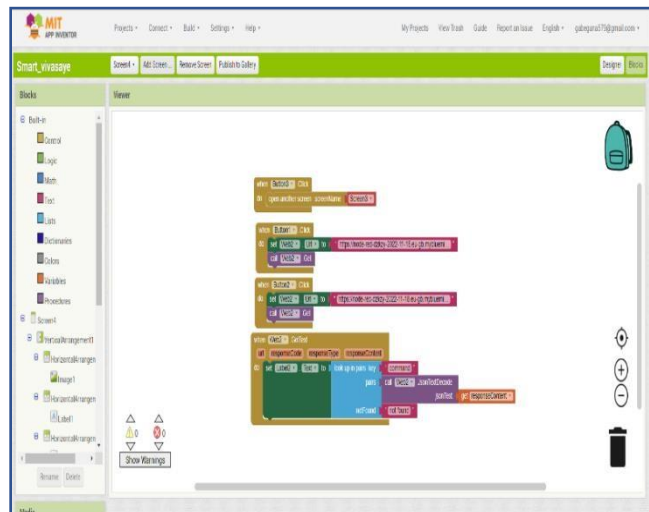
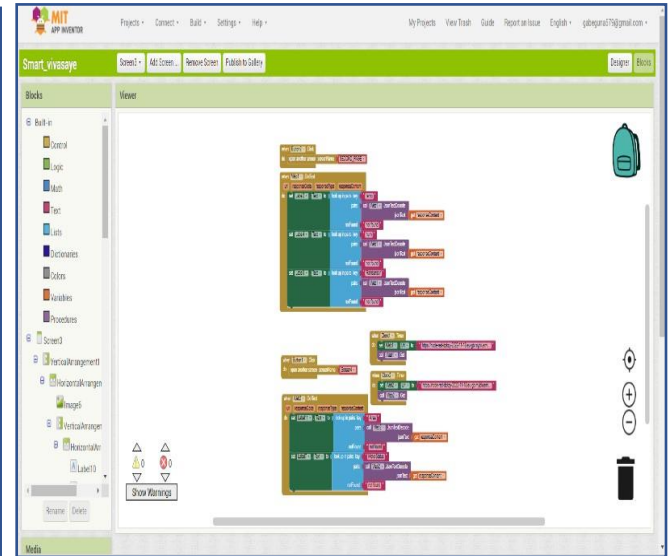
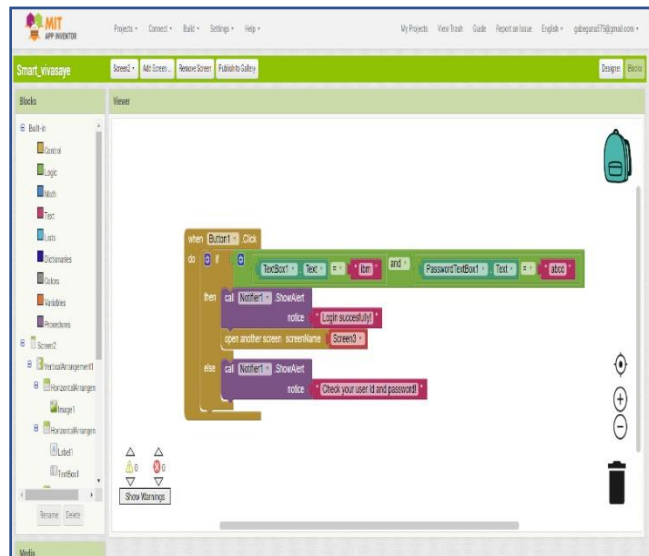
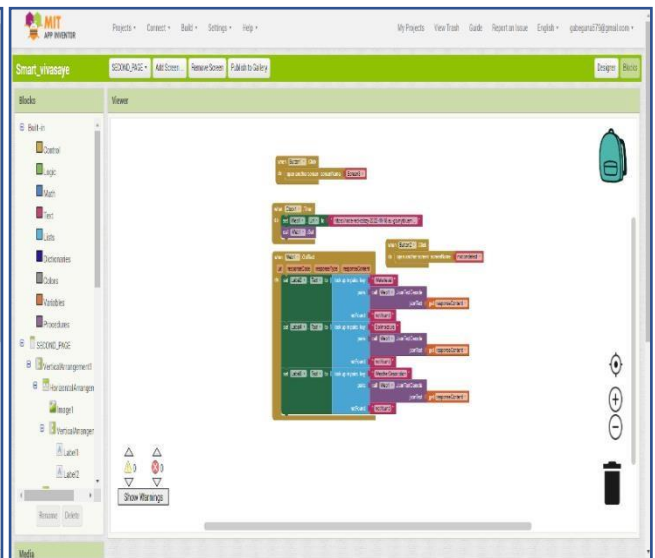
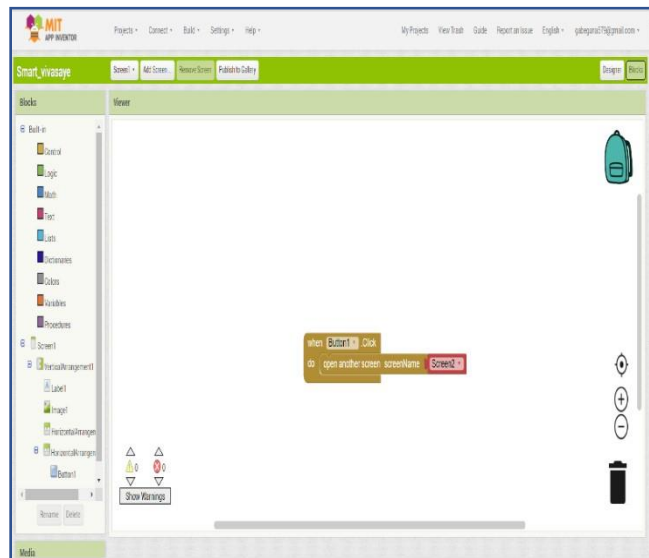
Event	Value	Format	Last Received
event_1	{"temperature": 47.6, "Humid": 68, "PIR_Sensor": "...	json	a few seconds ago
	{"temperature": 47.6, "Humid": 68, "PIR_Sensor": "No Motion Detected", "Ultrasonic_Sensor": "No Motion Detected", "Rain_Sensor": "Normal Rain"}		

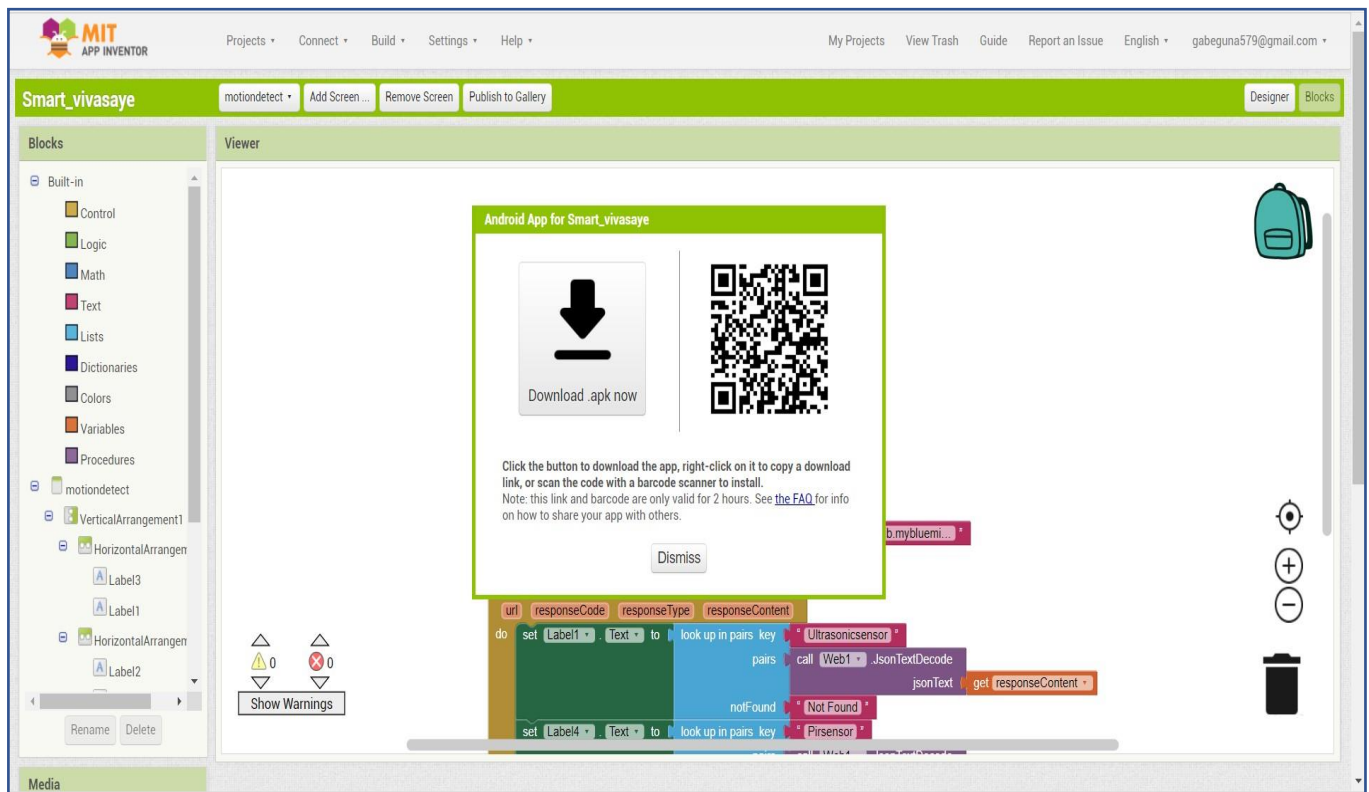
7.2 Feature 2

MIT APP inventor to design the APP



Customize the App interface to Display the Values





8.1 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	15	4	2	3	24
Duplicate	2	1	0	0	3
External	2	2	1	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	0	1
Won't Fix	1	5	2	1	9
Totals	31	14	11	25	81

8.2 Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Client Application	23	0	0	23
Security	1	0	0	1
Exception Reporting	20	0	0	20
Final Report Output	6	0	0	6
Version Control	2	0	0	2

9. RESULT:

We have successfully built an IOT Based Smart Crop Protection System for Agriculture and used Node-Red , IBM IoT Cloud Platform to accomplish it.

10. ADVANTAGES & DISADVANTAGES:

10.1 Advantages

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

10.2 Disadvantages

- Smart Crop Protection requires internet connectivity continuously, but rural part can not fulfill this requirement.
- Any faults in the sensors can cause great loss in the agriculture, due to wrong records and the actions of automated processes.
- IoT devices need much money to implement.

11. CONCLUSION:

IoT based smart Crop Monitoring System for Agriculture for Live Monitoring of Temperature and Soil Moisture and to control motor and light remotely has been proposed using Node Red and IBM CloudPlatform. The System has high efficiency and accuracy in fetching the live data of temperature and soil moisture. The IoT based smart farming System being proposed via this project will assist farmers in increasing the agriculture yield and take efficient care of food production as the System will always provide helping hand to farmers for getting accurate live feed of environmental temperature and soil moisture with more than 99% accurate results. Therefore, the project proposes a thought of consolidating the most recent innovation into the agrarian field to turn the customary techniques for water system to current strategies in this way making simple profitable and temperate trimming.

12. FUTURE SCOPE:

In future due to more demand of good and more farming in less time, for betterment of the crops and reducing the usage of extravagant resources like electricity and water IoT can be implemented in most of the places. In this project we have used simulator instead of IoT devices, in the future it will be implemented in real time. The email will be sent to user in case of any critical situation in the Agricultural land. This project will be implemented in the agricultural field in real time. The use of water from Rain Water Harvesting or Ground water to water the plants is not automated, in the future it will be automated.

13. APPENDIX:

GITHUB LINK : <https://github.com/IBM-EPBL/IBM-Project-43417-1660716788>

WOKWI LINK : <https://wokwi.com/projects/349284450026127956>

APP QR CODE :

