K.L.N COLLEGE OF ENGINEERING,POTTAPALAYAM

(An Autonomous institution, affiliated to Anna University, Chennai)

Problem statement: SmartFarmer - IoT Enabled SmartFarming Application

Team ID: PNT2022TMID11484

Team Leader : Bala Vignesh Kumar M

Team Members :

1. V Hari Prasath
2. P Dhilip
3. S L Senthil Sachin

Faculty Mentor: B. BUVANESWARI

Evaluator: L. MEENAKSHI

Industry Mentor: BHARADWAJ

ANNA UNIVERCITY : CHENNAI 600025

BONAFIDE CERTIFICAT

Certified that this project report **"SMARTFARMER-IOT ENABLED SMART FARMING APPLICATION"** is the bonafide work of

M.BALA VIGNESH KUMAR (910619106008)
V.HARI PRASATH(910619106020)
P.DHILIP (910619106013)
S.L .SENTHIL SACHIN(910619106307)

Who carried out the project work under our supervision.

SIGNATURE	**SIGNATURE**
FACULTY MENTOR	**FACULTY EVALUATOR**
DR. B. BUVANESHWARI	MEENAKSHI L
ASSISTANT PROFESSOR	ASSISTANT PROFESSOR (Sr.Gr)
ELECTRONICS AND COMMUNICATION ENGINEERING	ELECTRONICS AND COMMUNICATION ENGINEERING
K.L.N COLLEGE OF ENGINEERING	K.L.N COLLEGE OF ENGINEERING

SIGNATURE

DR.V.KEJALAKSHMI

HEAD OF THE DEPARTMENT

ELECTRONICS AND COMMUNICATION ENGINEERING
K.L.N COLLEGE OF ENGINEERING

ABSTRACT

Agriculture is the broadest economic sector and plays an important role in the overall economic development of a nation. Technological advancements in the arena of agriculture will ascertain to increase the competence of certain farming activities. In this paper, we have proposed a novel methodology for smart farming by linking a smart sensing system and smart irrigator system through wireless communication technology. Our system focuses on the measurement of soil's temperature and humidity and overall lumination, temperature, weather condition of the farm yard.

LIST OF FIGURES

LIST OF TABLE- FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. PROJECT OVERVIEW:

This is a Smart Agriculture System project based on Internet Of Things (IoT), that can measure soil moisture, Humidity and temperature conditions for agriculture using Watson IoT services. 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.

- **Project Requirements** : Node-RED, IBM Cloud, IBM Watson IoT, Python 3.8, Open Weather API platform,MIT app inventor, Wokei .
- **Project Deliverables** : Application for IoT based Smart Agriculture System

1.2. PURPOSE

IoT based farming is grooming nowadays because it improves the entire agriculture system by monitoring the field in real-time. With the help of IoT in agriculture not only saves the time but also reduces the extravagant use of resources such as water and electricity.

Sometimes due to over or less supply of water in the agricultural field crops may not grow proper. Using IoT supply of water and growth of plants can be satisfied to a greater extent. The flow of water can be controlled from the application.

Thus this approach towards Agriculture will help the farmers to get better yield at low cost and without much usage of resources.

CHAPTER 2

LITERATURE SURVEY

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.
- The paper based on smart agriculture using IOT [1] uses the data from sensors to detect the humidity, soil moisure content and temperature for finding adequate quantity of water used for irrigation.
- The project that uses IoT-enabled system architecture and platform [2] that provides cloud-based IoT irrigation control solution, it helps the user to remotely operate and control the irrigation system from the mobile phone.

2.2. REFERENCES

[1] Kasara Sai Pratyush Reddy, Mohana Roopa Dr.Y, Kovvada Rajeev Lakshmi Narasimha and Narra Sai Nandan, “IoT based Smart Agriculture using Machine Learning”, *Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, July 2020.

[2] Schneider Electric, "WaterForce launch IIoT solution for New Zealand farms", *https://enterpriseiotinsights.com/20170802/smart-farm/20170802internet-of-thingsschneider-electric-waterforce-launch-iiot-solutions-new-zealand-farms-tag23,* 2017.

2.3. PROBLEM STATEMENT DEFINITION

There are many cases of crop failures and most of the cases have the same reason behind it and that is lack of water which causes the irrigation system inadequate for the crop field. The main concern is inefficient farming practices which leads to crop failures. We can overcome this by providing proper technology/smart technology to the farmers and for the farming activity, which in turn increases the yield .And the challenge now is to develop a cheap, but accurate system that will provide the farmer with the adequate amount of information related to the moisture of the soil, the temperature, humidity and all required elements which play an important role in the vegetation yield. Also motor controls can be given to them so that they can access the motor from anywhere through the web app and also through the phone application.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1. EMPATHY MAP CANVAS

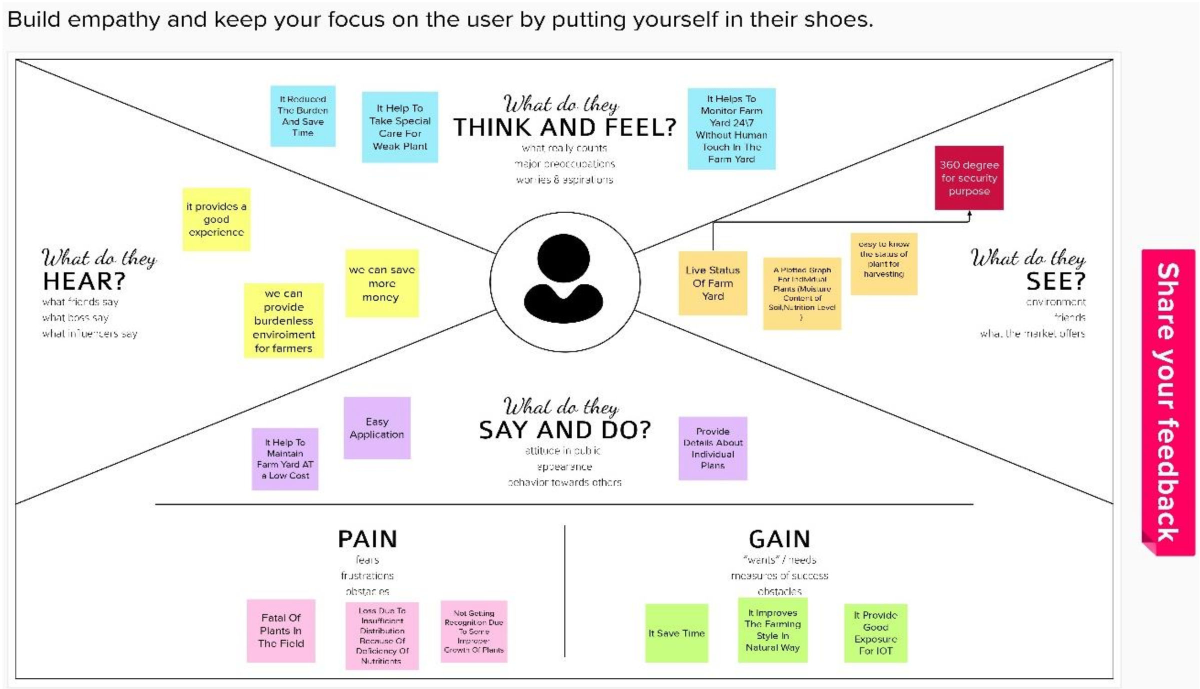


Figure 3.1 Empathy Map Canvas

3.2. IDEATION & BRAINSTORMING

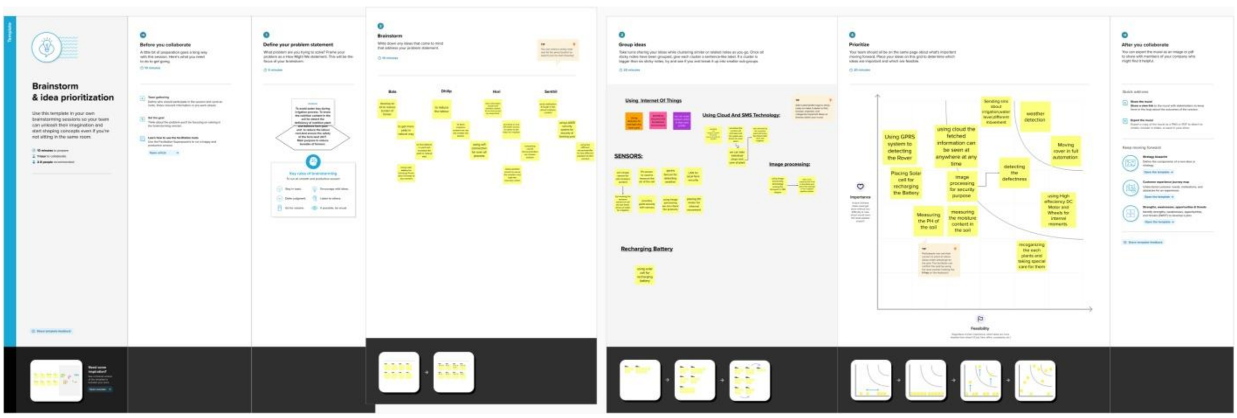


Figure 3.2 Ideation & Brainstorming

3.3. PROPOSED SOLUTION

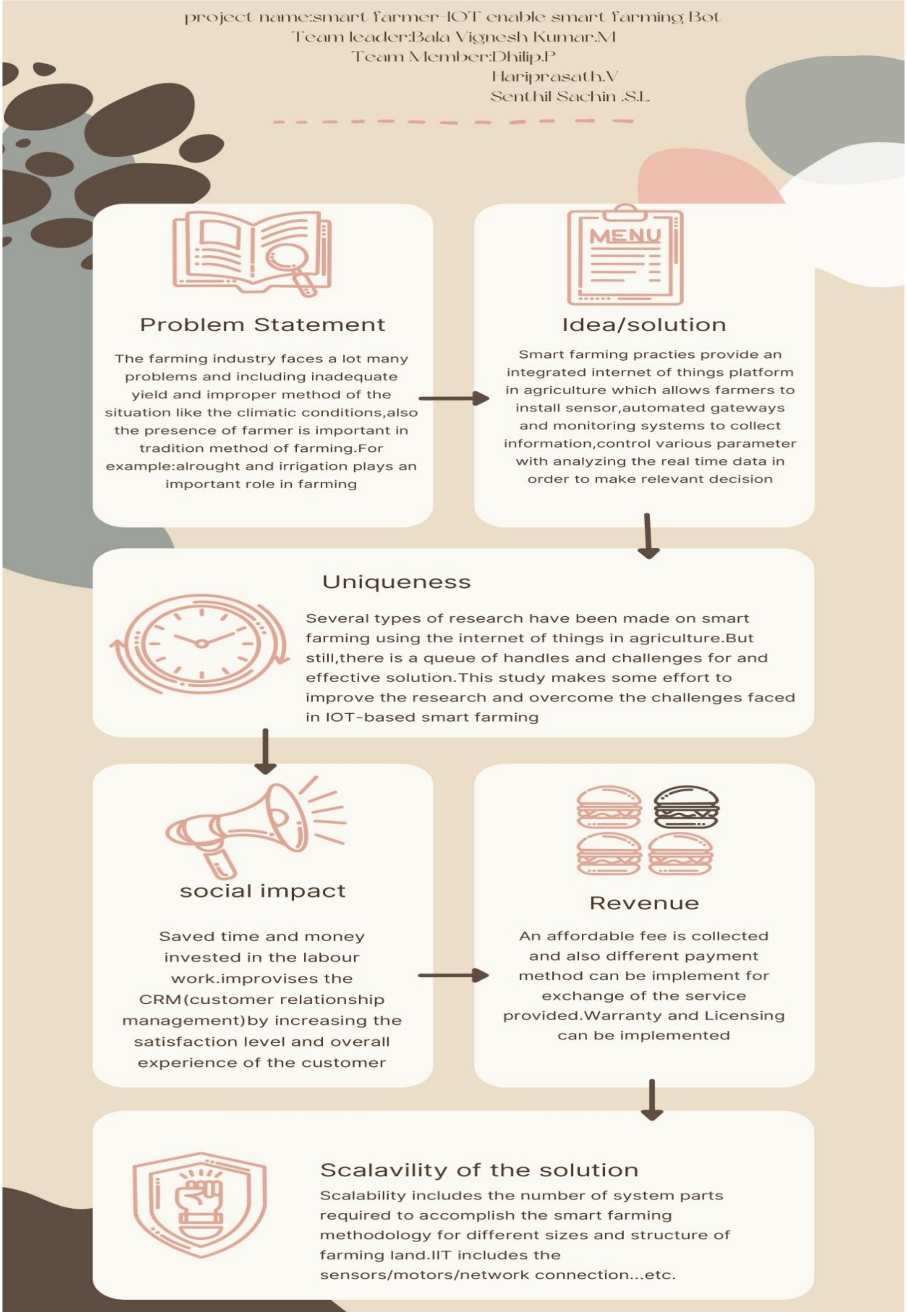


Figure 3.3 Proposed Solution

3.4. PROBLEM SOLUTION FIT

Figure 3.4 Solution Fit

CHAPTER 4

REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENT

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/Sub-Task)
FR-1	User Registration/ Login	Phone Application and Wi-fi Module.
FR-2	User Permission	User permission for irrigation via Mobile Application and software Web UI Application
FR-3	Login/ App	Check Id/ user name and check role access.
FR-4	Check weather details	Temperature, Humidity, Mositure, water, motor runtime, motor (On/Off), internal security.
FR-5	Logout	Exit

Table Figure 4.1 Functional Requirement

4.2. NON-FUNCTIONAL REQUIREMENTS

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User-friendly interface and easy to learn , efficiency in use, remember ability, lack of errors in operation
NFR-2	Security	Sensitive and private data must be protected from their production until the decision-making and storage stages
NFR-3	Reliability	The shared protection achieves a better trade-off between costs and reliability. Accuracy of data
NFR-4	Performance	The process of the usage is easy and simple which allows to monitor and control with application's stability and accuracy
NFR-5	Availability	Automatic adjustment of farming equipment made possible by linking information like crops and weather details.

Table Figure 4.2 Non Functional Requirement

CHAPTER 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.

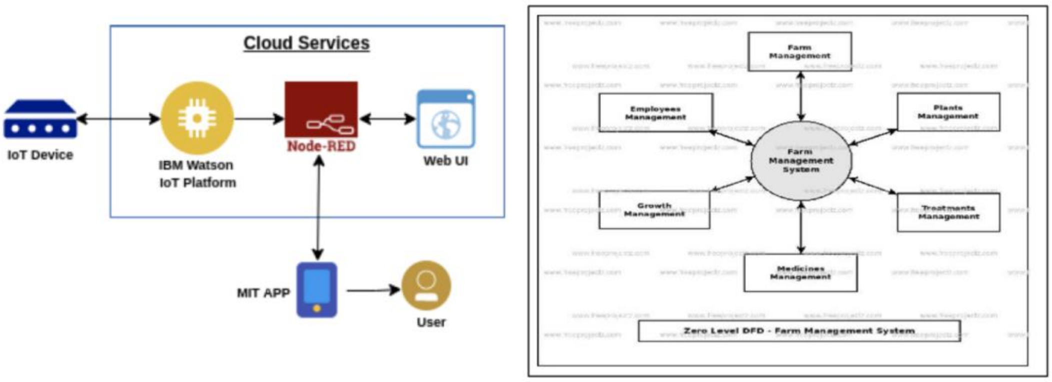


Figure 5.1 Data Flow Diagrams

5.2. SOLUTION & TECHNICAL ARCHITECTURE

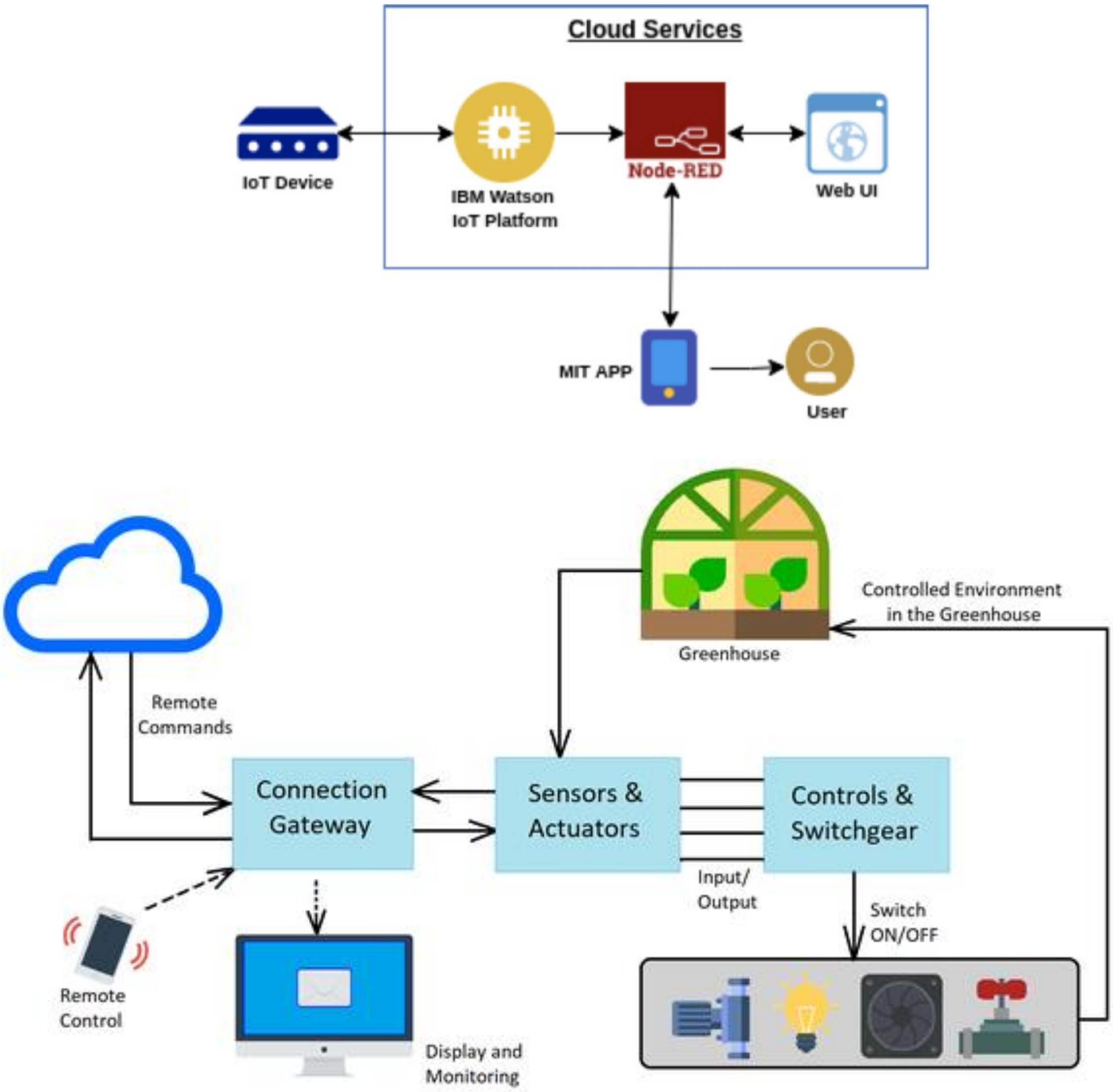


Figure 5.2 Solution & Technical Architecture

S.No	Component	Description	Technology
1.	User Interface	How user interacts with the application. Eg. Web UI, Mobile App, Chatbot, etc.,	MIT App
2.	Application Logic - 1	Logic for a process in the application.	Node red/ IBM Watson/ MIT

			App
3.	Application Logic -2	Logic for a process in the application.	Node red/ IBM Watson/ MIT App
4.	Application Logic -3	Logic for a process in the application.	Node red/ IBM Watson/ MIT App
5.	Database	Data type, configuration etc.,	MySQL, NoSQL
6.	Cloud Database	Database service on cloud	IBM Cloud
7.	Temperature sensor	Monitors the temperature of the crop	
8.	Humidity sensor	Monitors the humidity	
9.	Soil moisture sensor (Tensiometer)	Monitors the soil temperature	
10.	Weather sensor	Monitors the weather	

S.NO	Characteristics	Description	Technology
1.	Open-Source Framework	MIT App, Node-Red	Software
2.	Scalable Architecture	Drone technology, pesticide monitoring, mineral identification in soil	Hardware

Table Figure 5.2 Solution & Technical Architecture

5.3. USER STORIES

User type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
Customer (Mobile User)	Login	USN-1	As a Administrator, I need to give user idand passcode so that user can have access.	I can access my account	low
	Dashboard	USN-2	As a user, I can access option for viewing farm security detail, farmyard 3600 degree view, graphical representation of farm details, weather condition , light, humidity, temperature, soil moisture, and timing, control for rover so that I can look after their yard.	I can access the dashboard.	medium
Customer (Web user)	app	USN-3	As a user, I can access a mobile app, so that I can remotely oversee the farm land	I can register and access the app.	high
Customer Care Executive		USN-4	As executive, I can make user interact with software.	Database stored in the cloud.	High
	Logout	USN-5	Exit	Sign out.	

Table Figure 5.3 User Stories

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1. SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Login	USN-1	As a Administrator, I need to give user idand passcode so that user can have access.	10	low	Bala,hari
Sprint-1	Dashboard	USN-2	As a developer, I need to provide option for viewing farm security detail, farmyard 3600 degree view, graphical representation of farm details, weather condition , light, humidity, temperature, soil moisture, and timing, control for rover so that farmers can look after their yard.	10	medium	Bala,senthil
Sprint-2	app	USN-3	As a developer, I need to provide a mobile app, so that user can remotely oversee their farm land	20	high	Senthil,Hari, Dhilp,bala
Sprint-3	Simulation	USN-4	As a developer, I need to simulate and connect the sensors, so the input is provided to the app.	20	medium	Bala,hari
Sprint-4	Database	USN-5	As a developer, I need to develop a back end connectivity so that the app can retrieve data from database.	20	High	Dhilip,bala

Table Figure 6.1 Sprint Planning

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	6Days	24Oct2022	29Oct2022	20	29Oct2022
Sprint-2	20	6Days	31Oct2022	05Nov2022	20	05Nov2022
Sprint-3	20	6Days	07Nov2022	12Nov2022	20	12Nov2022

Sprint-4	20	6Days	14Nov2022	19Nov2022	20	19Nov2022

Table Figure 6.2 Sprint Delivery Schedule

6.3. REPORTS FROM JIRA

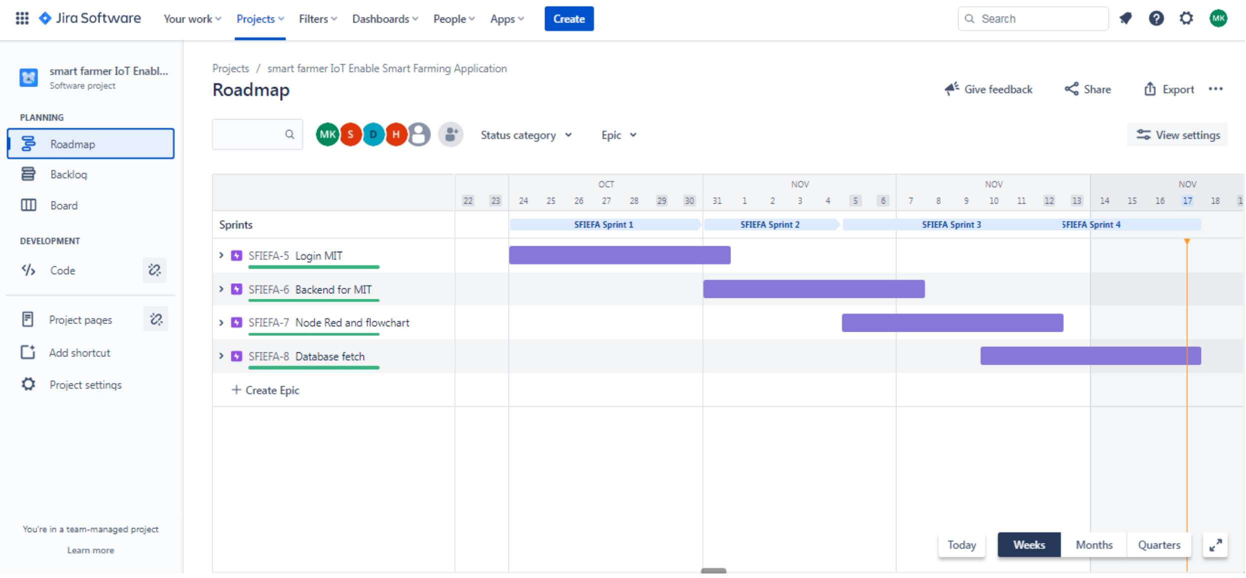


Figure 6.3 Jira Report

CHAPTER 7

CODING & SOLUTIONING

Python Code:

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



#Provide your IBM Watson Device Credentials
organization = "i3lxgz"
deviceType = "ESP32"
deviceId = "06020"
authMethod = "token"
authToken = "1234567890"

from bs4 import BeautifulSoup
import requests
headers = {
	'User-Agent': 'Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/58.0.3029.110 Safari/537.3'}
def weather(city):
	city = city.replace(" ", "+")
```

```
      res = requests.get(

      f'https://www.google.com/search?q={city}&oq={city
}&aqs=chrome.0.35i39l2j0l4j46j69i60.6128j1j7&sourceid=
chrome&ie=UTF-8', headers=headers)
      soup = BeautifulSoup(res.text, 'html.parser')
      location =
soup.select('#wob_loc')[0].getText().strip()
      time =
soup.select('#wob_dts')[0].getText().strip()
      info =
soup.select('#wob_dc')[0].getText().strip()
      weather =
soup.select('#wob_tm')[0].getText().strip()
      print(location, ',', time, ',', info, ',',
weather+'°C')
      #print(time)
      #print(info)
      #print(weather+"°C")

city = 'Madurai'
city = city+" weather"
weather(city)

# Initialize GPIO

def myCommandCallback(cmd):
    print("Command received: %s" %
cmd.data['command'])
```

```
    status=cmd.data['command']
    if status=="switchon":
        print ("Switch is on")
    else :
        print ("Switch is off")

    #print(cmd)

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

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

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

while True:
        #Get Sensor Data from DHT11

        temp=random.randint(0,100)
```

```
        Humid=random.randint(0,100)
        tottemp=random.randint(0,100)


        data = { 'temp' : temp, 'Humid': Humid,
"tottemp": tottemp}

        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" %
temp, "Humidity = %s %%" % Humid, "tottemp = %s %%" %
tottemp, "to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor",
"json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(1)

        deviceCli.commandCallback = myCommandCallback

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

ibmpro1.py - C:\Users\bala2\Downloads\ibmpro1.py (3.7.4)

File Edit Format Run Options Window Help

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


#Provide your IBM Watson Device Credentials
organization = "i31xgz"
deviceType = "ESP32"
deviceId = "06020"
authMethod = "token"
authToken = "1234567890"

from bs4 import BeautifulSoup
import requests
headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}
def weather(city):
        city = city.replace(" ", "+")
        res = requests.get(
                f'https://www.google.com/search?q={city}&oq={city}&aqs=chrome.0.35i39l2j0l4j46j69i60.6128j1j7&sourceid=chrome&ie=UTF-8', headers=headers)
        soup = BeautifulSoup(res.text, 'html.parser')
        location = soup.select('#wob_loc')[0].getText().strip()
        time = soup.select('#wob_dts')[0].getText().strip()
        info = soup.select('#wob_dc')[0].getText().strip()
        weather = soup.select('#wob_tm')[0].getText().strip()
        print(location, ',', time, ',', info, ',', weather+'°C')
        #print(time)
        #print(info)
        #print(weather+"°C")

city = 'Madurai'
city = city+" weather"
weather(city)

# Initialize GPIO

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="switchon":
        print ("Switch is on")
    else :
        print ("Switch is off")

    #print(cmd)

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

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

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

while True:
        #Get Sensor Data from DHT11

        temp=random.randint(0,100)
        Humid=random.randint(0,100)
        tottemp=random.randint(0,100)


        data = { 'temp' : temp, 'Humid': Humid, "tottemp": tottemp}

        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % Humid, "tottemp = %s %%" % tottemp, "to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(1)

        deviceCli.commandCallback = myCommandCallback

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

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\Users\bala2\Downloads\ibmpro1.py ================
Madurai, Tamil Nadu , Monday, 10:00 pm , Fog , 25°C
2022-11-21 22:17:37,496   ibmiotf.device.Client      INFO    Connected successfully: d:i31xgz:ESP32:06020
Published Temperature = 68 C Humidity = 24 % tottemp = 28 % to IBM Watson
Published Temperature = 80 C Humidity = 33 % tottemp = 93 % to IBM Watson
Published Temperature = 15 C Humidity = 74 % tottemp = 50 % to IBM Watson
Published Temperature = 10 C Humidity = 9 % tottemp = 11 % to IBM Watson
Published Temperature = 61 C Humidity = 68 % tottemp = 23 % to IBM Watson
Published Temperature = 83 C Humidity = 93 % tottemp = 17 % to IBM Watson
Published Temperature = 43 C Humidity = 81 % tottemp = 70 % to IBM Watson
Published Temperature = 48 C Humidity = 56 % tottemp = 65 % to IBM Watson
Published Temperature = 47 C Humidity = 52 % tottemp = 28 % to IBM Watson
Published Temperature = 28 C Humidity = 75 % tottemp = 12 % to IBM Watson
Published Temperature = 62 C Humidity = 48 % tottemp = 12 % to IBM Watson
Published Temperature = 51 C Humidity = 63 % tottemp = 20 % to IBM Watson
Published Temperature = 41 C Humidity = 58 % tottemp = 18 % to IBM Watson
Published Temperature = 45 C Humidity = 100 % tottemp = 45 % to IBM Watson
Published Temperature = 83 C Humidity = 60 % tottemp = 0 % to IBM Watson
Published Temperature = 11 C Humidity = 68 % tottemp = 57 % to IBM Watson
```

Figure 7.0 Python code and Output

7.1. FEATURE 1

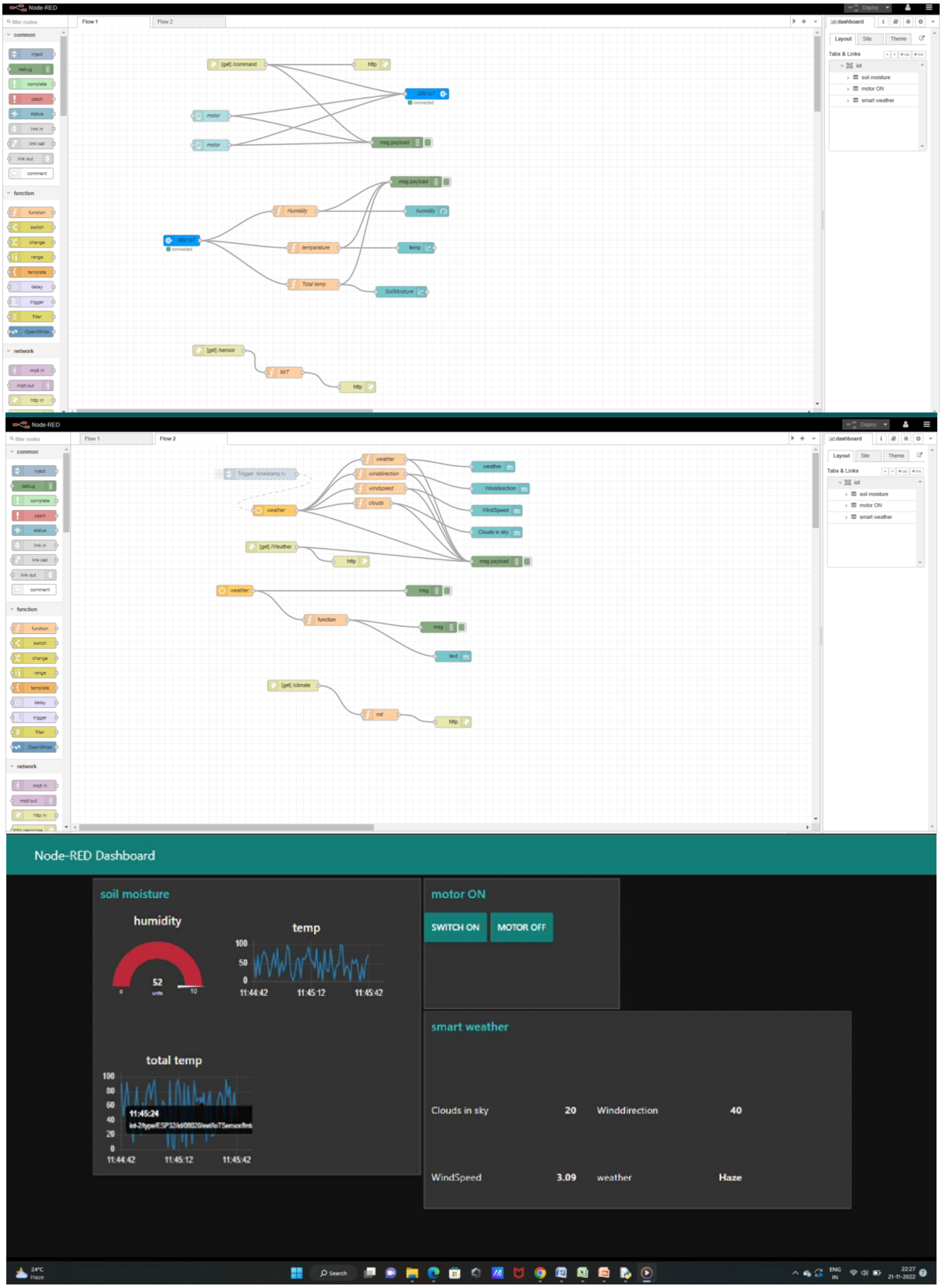


Figure 7.1 Node Red Dashboard

7.2. FEATURE 2

Code:

```
#include <WiFi.h>
#include <PubSubClient.h>
//void PubSubClient client(server, 1883, calloc, callback
,wifiClient);
void callback(char* subscribetopic, byte* payload, unsigned
int payloadLength);
//-------credentials of IBM Accounts------
#define ORG "pw4ndm"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm
watson IOT Platform
#define DEVICE_ID "06008"//Device ID mentioned in ibm
watson IOT Platform
#define TOKEN "123456789" //Token
String data3;
char server[] = ORG
".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/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);

#include "DHTesp.h"

const int DHT_PIN = 15;
const float BETA = 3950;
#define LDR_PIN 2

DHTesp dhtSensor;

void setup() {
  Serial.begin(115200);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  Serial.begin(9600);
  analogReadResolution(10);
  pinMode(15,INPUT);
  pinMode(4,OUTPUT);
```

```
   pinMode(2, INPUT);
}

void loop() {
  TempAndHumidity  data = dhtSensor.getTempAndHumidity();
  Serial.println("Temp: " + String(data.temperature, 2) +
"°C");
  Serial.println("Humidity: " + String(data.humidity, 1) +
"%");
  Serial.println("---");
  delay(1000);
  int analogValue = analogRead(15);
  float celsius = 1 / (log(1 / (1023. / analogValue - 1)) /
BETA + 1.0 / 298.15) - 273.15;
  Serial.print("Temperature: ");
  Serial.print(celsius);
  Serial.println(" ℃");
  Serial.println(": ");
  if (digitalRead(LDR_PIN) == LOW) {
    Serial.println("Light!");
  }
```

```
  else {
    Serial.println("ALERT!!");
    Serial.println("Dark  ");
    delay(1000);
    PublishData();
    delay(1000);
    if (!client.loop()) {
      mqttconnect();
    }
  }
  delay(100);
}
void PublishData() {
  mqttconnect();
  String payload = "{\"Cloudy\":";
  payload += ",\"ALERT!!\":""\"Cross check the weather and
keep the bot safe\"";
  payload += "}";
  Serial.print("Sending payload: ");
  Serial.println(payload);
```

```
  if (client.publish(publishTopic, (char*)
payload.c_str())) {
    Serial.println("Publish ok");
  }
  else {
    Serial.println("Publish failed");
  }
}
void mqttconnect() {
if (!client.connected()) {
Serial.print("Reconnecting client to ");
Serial.println(server);
while (!!!client.connect(clientId, authMethod, token)) {
Serial.print(".");
delay(500);
}
initManagedDevice();
Serial.println();
}
}
void wificonnect()
```

```
{
Serial.println();
Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
void initManagedDevice() {
if (client.subscribe(subscribetopic)) {
Serial.println((subscribetopic));
Serial.println("subscribe to cmd OK");
} else {
Serial.println("subscribe to cmd FAILED");
}
}
```

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

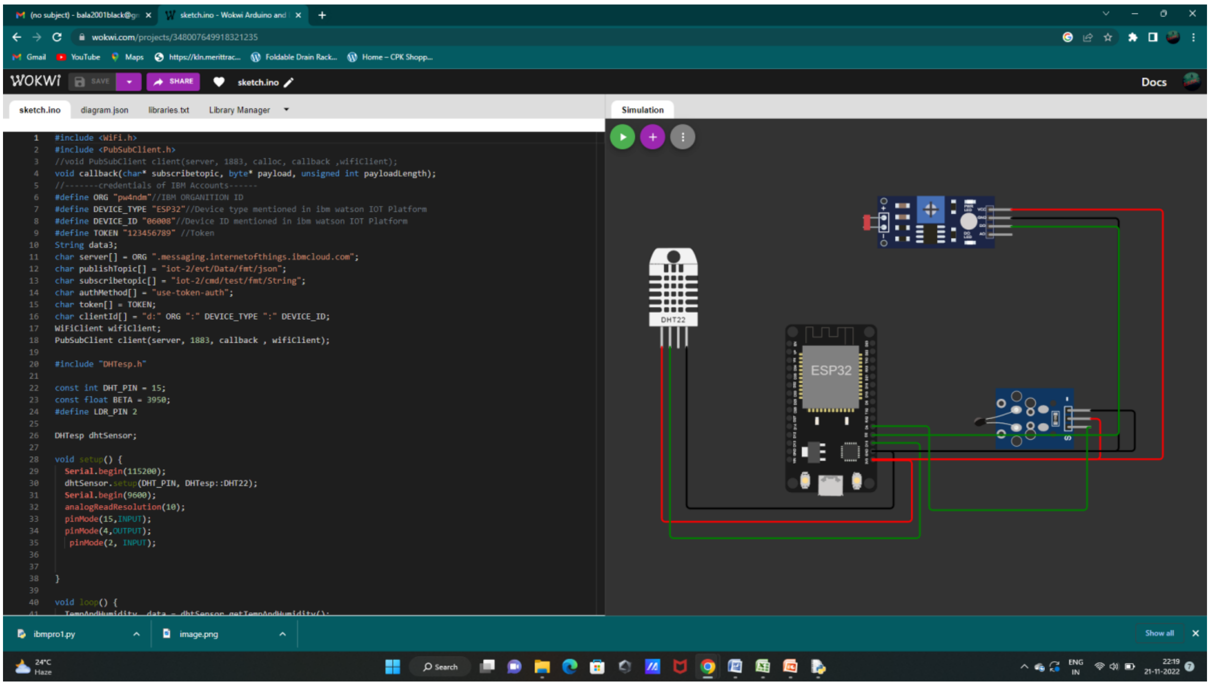


Figure 7.2.1 Wokwi Screen

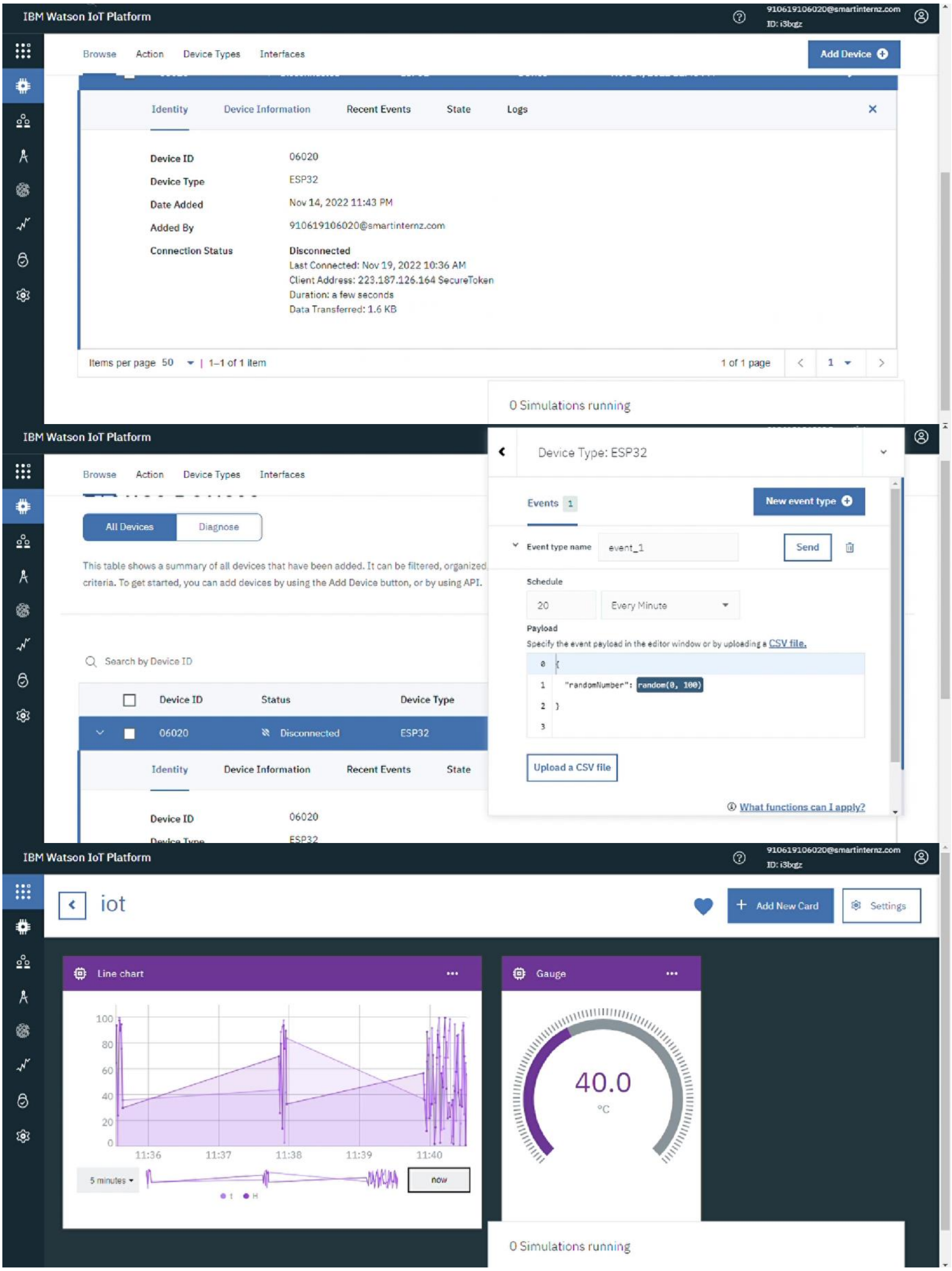
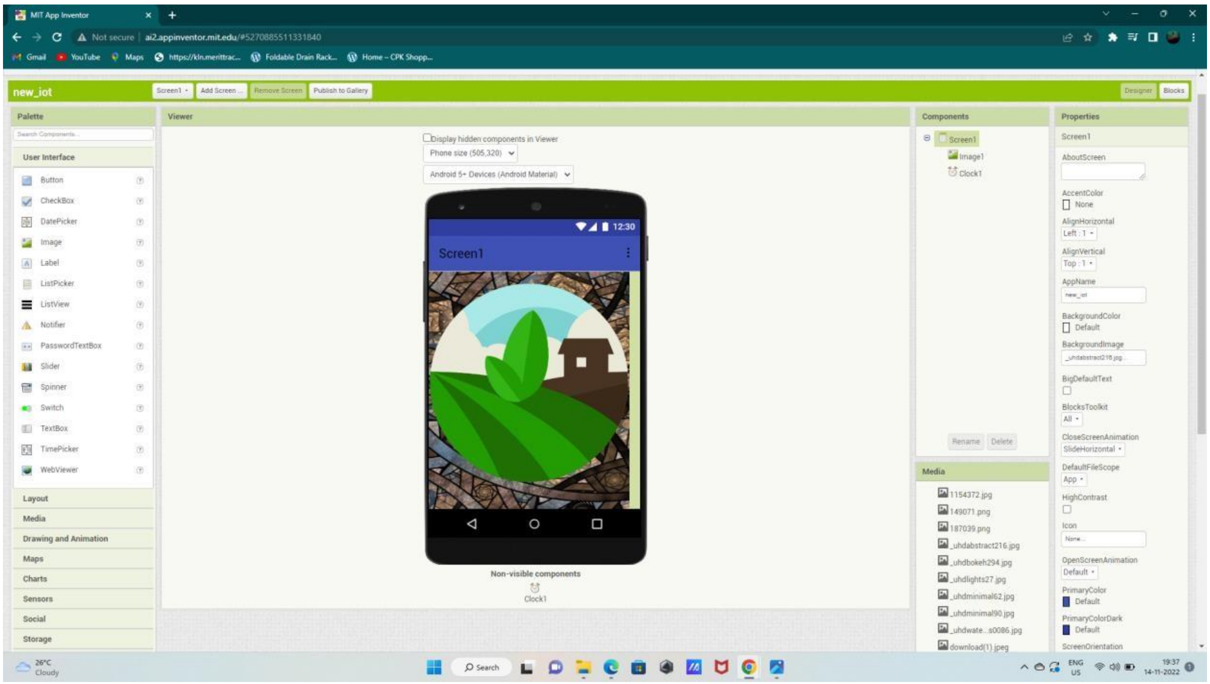
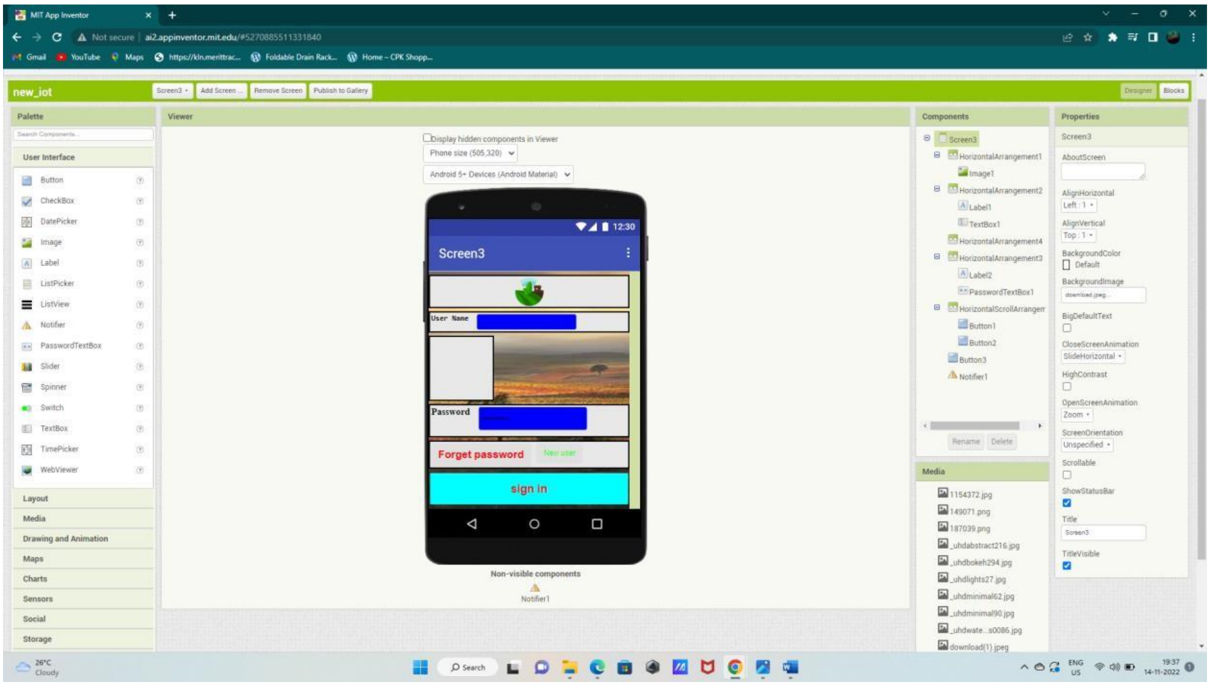


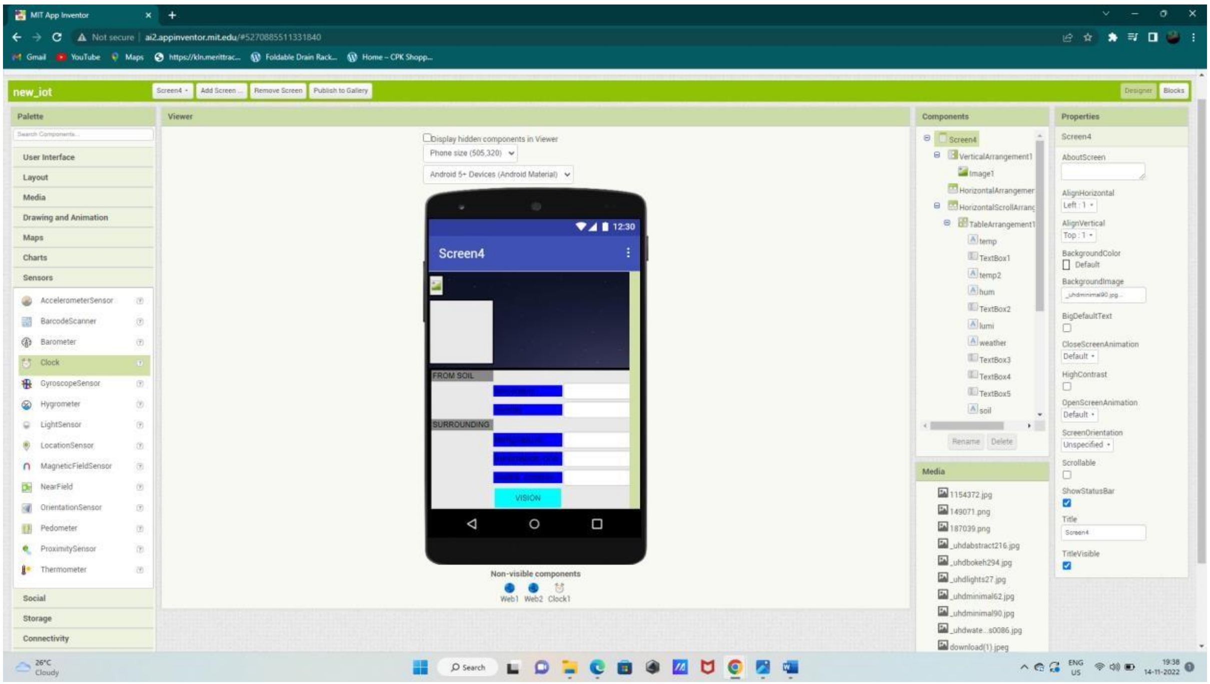
Figure 7.2.2 IBM Watson and Dashboard

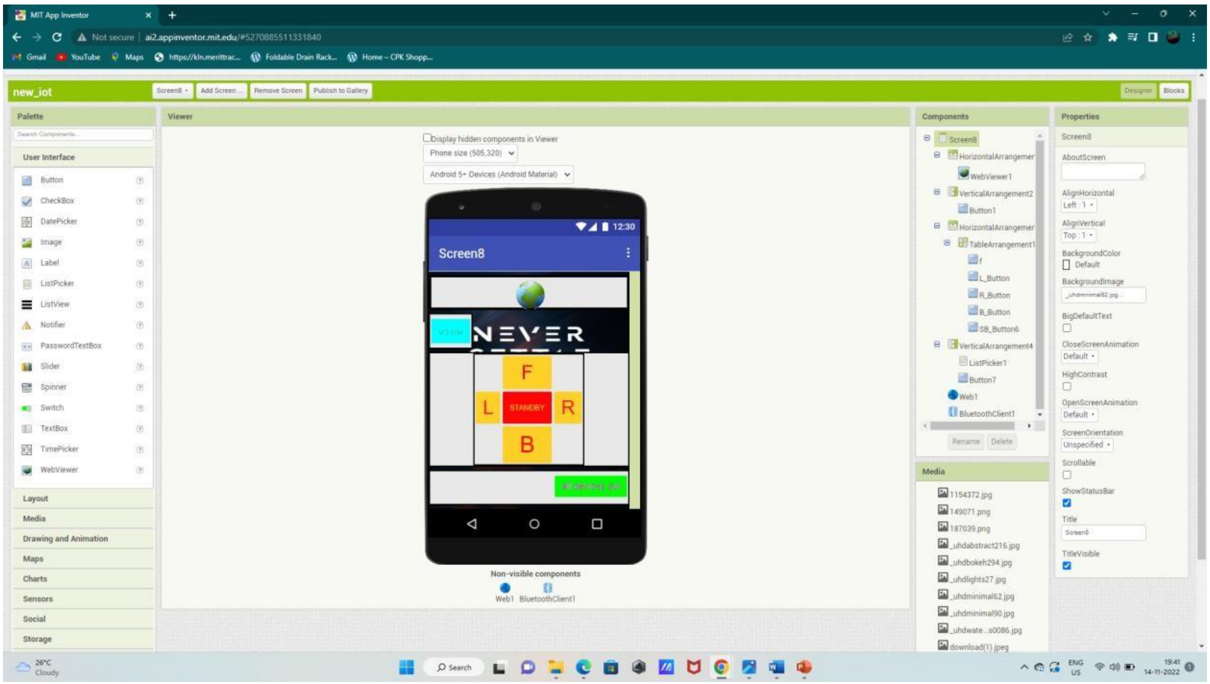
7.3. FEATURE 3:

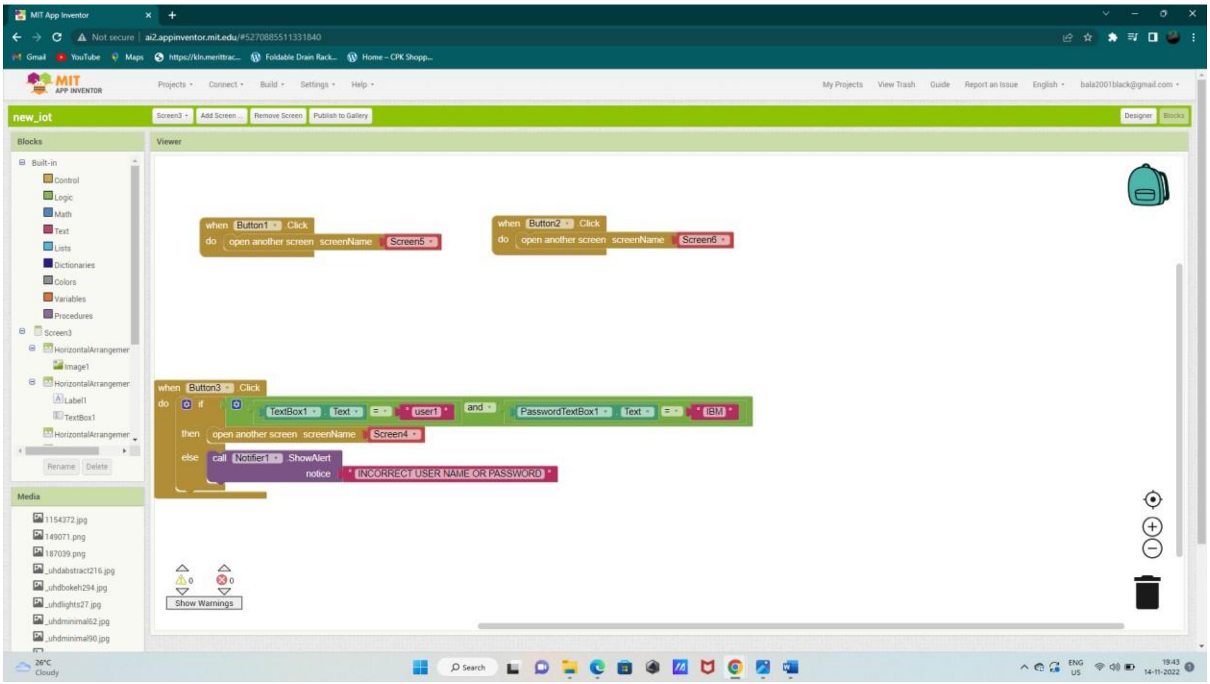
MIT APP:

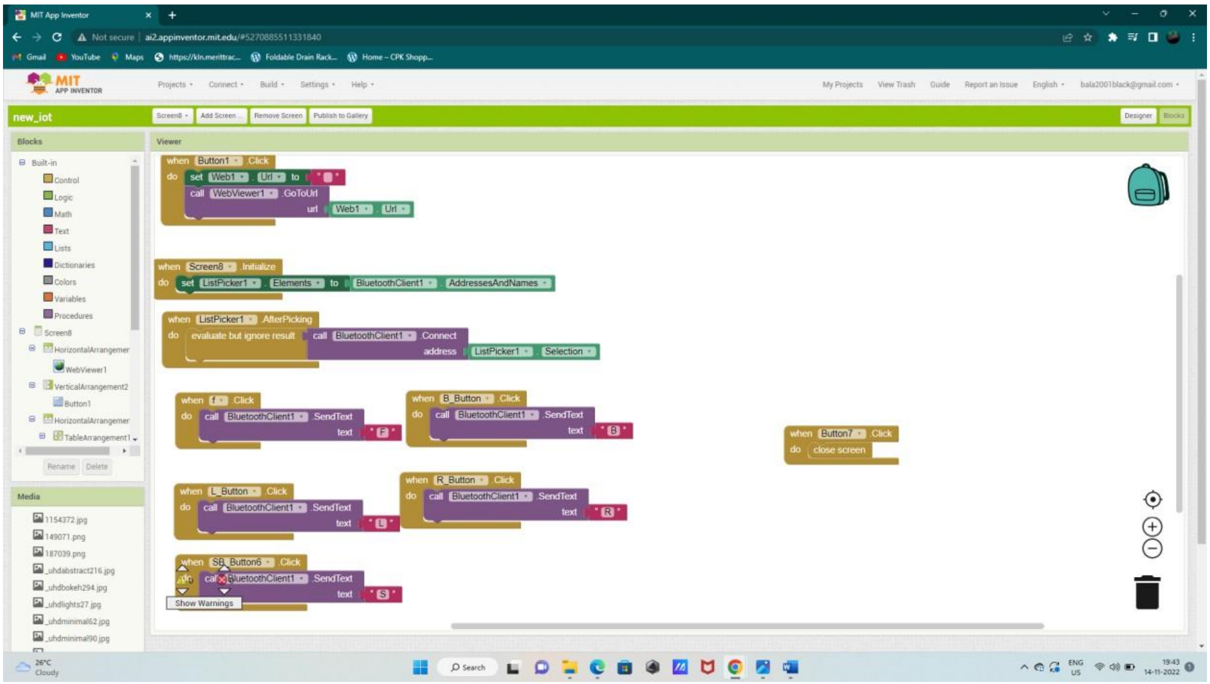


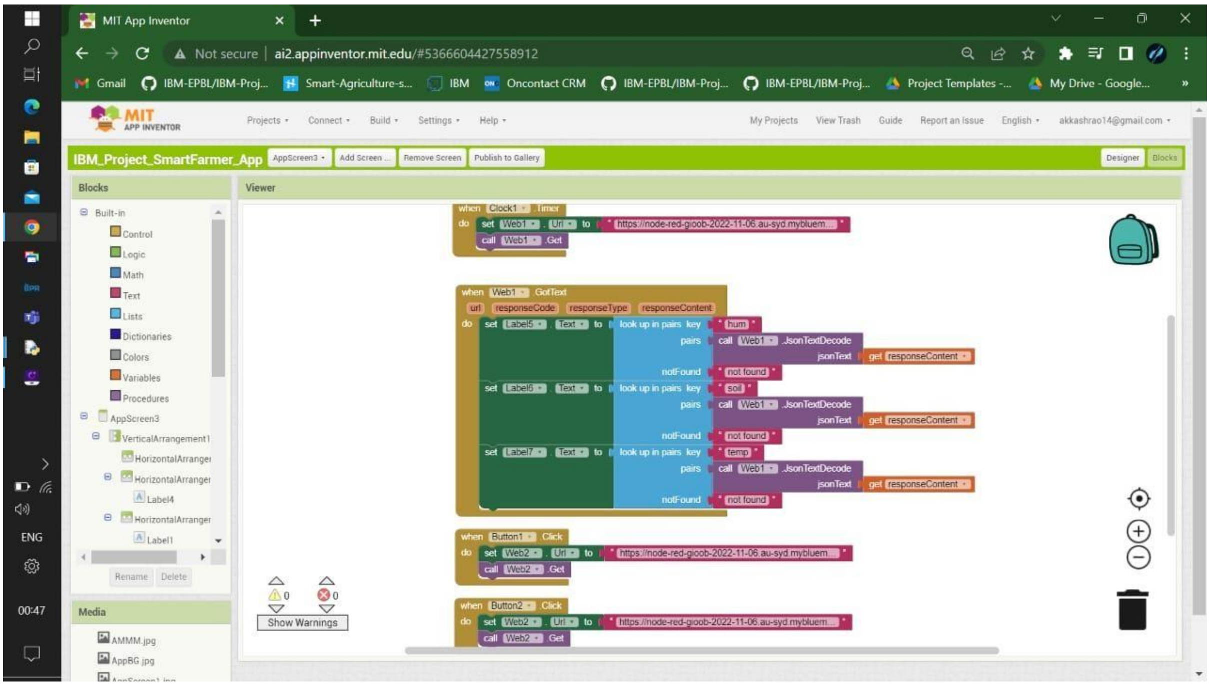



MIT App Inventor
ai2.appinventor.mit.edu/#5270885511331840
new_iot
Screen4
Add Screen ...
Remove Screen
Publish to Gallery
Designer
Blocks
Palette
Viewer
Components
Properties
User Interface
Layout
Media
Drawing and Animation
Maps
Charts
Sensors
AccelerometerSensor
BarcodeScanner
Barometer
Clock
GyroscopeSensor
Hygrometer
LightSensor
LocationSensor
MagneticFieldSensor
NearField
OrientationSensor
Pedometer
ProximitySensor
Thermometer
Social
Storage
Connectivity
Display hidden components in Viewer
Phone size (505,320)
Android 5+ Devices (Android Material)
Screen4
FROM SOIL
SURROUNDING
VISION
Non-visible components
Web1
Web2
Clock1
VerticalArrangement1
Image1
TableArrangement1
temp
TextBox1
temp2
hum
TextBox2
lumi
weather
TextBox3
TextBox4
TextBox5
soil
Rename
Delete
Media
1154372.jpg
149071.png
187039.png
_uhdabstract216.jpg
_uhdbokeh294.jpg
_uhdlights27.jpg
_uhdminimal62.jpg
_uhdminimal90.jpg
download(1).jpeg
AboutScreen
AlignHorizontal
AlignVertical
BackgroundColor
BackgroundImage
BigDefaultText
CloseScreenAnimation
HighContrast
OpenScreenAnimation
ScreenOrientation
Scrollable
ShowStatusBar
Title
TitleVisible

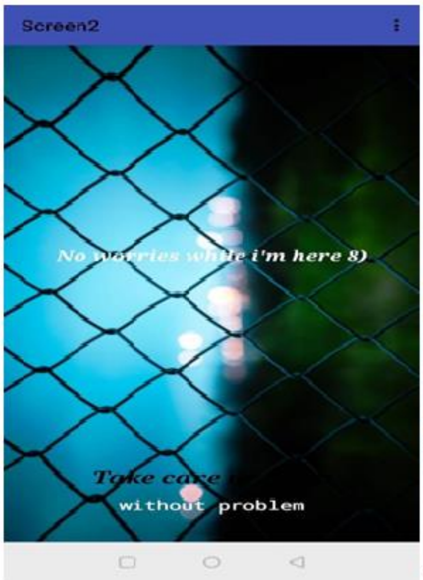

MIT App Inventor
ai2.appinventor.mit.edu/#5270885511331840
new_iot
Screen8
Add Screen ...
Remove Screen
Publish to Gallery
Designer
Blocks
Palette
Viewer
Components
Properties
User Interface
Button
CheckBox
DatePicker
Image
Label
ListPicker
ListView
Notifier
PasswordTextBox
Slider
Spinner
Switch
TextBox
TimePicker
WebViewer
Layout
Media
Drawing and Animation
Maps
Charts
Sensors
Social
Storage
Display hidden components in Viewer
Phone size (505,320)
Android 5+ Devices (Android Material)
Screen8
NEVER
F
L
STANDBY
R
B
Non-visible components
Web1
BluetoothClient1
WebViewer1
VerticalArrangement2
Button1
TableArrangement1
L_Button
R_Button
B_Button
SB_Button6
VerticalArrangement4
ListPicker1
Button7
Rename
Delete
Media
1154372.jpg
149071.png
187039.png
_uhdabstract216.jpg
_uhdbokeh294.jpg
_uhdlights27.jpg
_uhdminimal62.jpg
_uhdminimal90.jpg
download(1).jpeg
AboutScreen
AlignHorizontal
AlignVertical
BackgroundColor
BackgroundImage
BigDefaultText
CloseScreenAnimation
HighContrast
OpenScreenAnimation
ScreenOrientation
Scrollable
ShowStatusBar
Title
TitleVisible


when Button1 Click
do open another screen screenName "Screen5"
when Button2 Click
do open another screen screenName "Screen6"
when Button3 Click
do if TextBox1 Text = "user1" and PasswordTextBox1 Text = "IBM"
then open another screen screenName "Screen4"
else call Notifier1 ShowAlert
notice "INCORRECT USER NAME OR PASSWORD"


when Button1 Click
do set Web1 Url to
call WebViewer1 GoToUrl
url Web1 Url
when Screen8 Initialize
do set ListPicker1 Elements to BluetoothClient1 AddressesAndNames
when ListPicker1 AfterPicking
do evaluate but ignore result call BluetoothClient1 Connect
address ListPicker1 Selection
when f Click
do call BluetoothClient1 SendText
text "F"
when B_Button Click
do call BluetoothClient1 SendText
text "B"
when Button7 Click
do close screen
when L_Button Click
do call BluetoothClient1 SendText
text "L"
when R_Button Click
do call BluetoothClient1 SendText
text "R"
when SB_Button6 Click
do call BluetoothClient1 SendText
text "S"


MIT App Inventor
ai2.appinventor.mit.edu/#5366604427558912
IBM_Project_SmartFarmer_App
Blocks
Viewer
Show Warnings
Media


Screen1


Screen2
No worries while i'm here 8)
without problem


User Name
Password
Forget password
New user
sign in

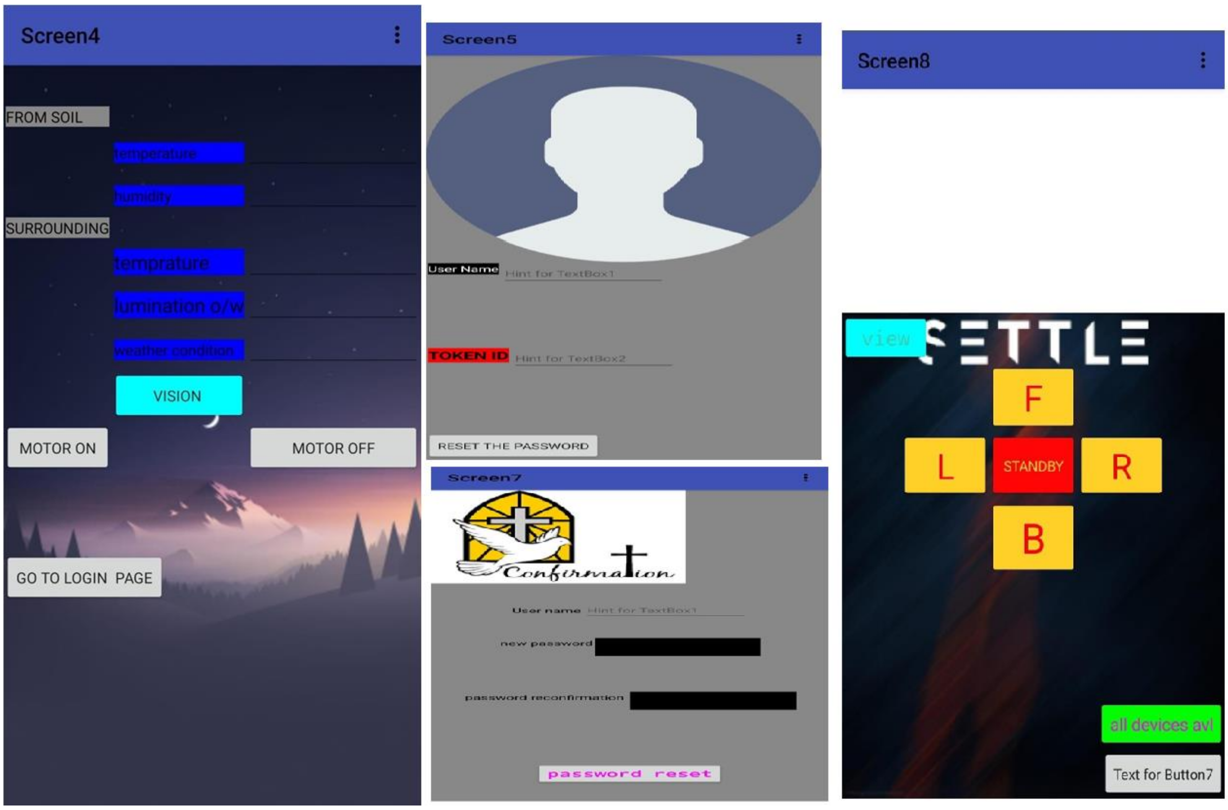


Figure 7.3 MIT App

CHAPTER 8

TESTING

8.1. TEST CASES

Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Executed By
Installation IBM IOT Platform	PC/Software	Installation of IBM IoT and Dashboard nodes for Node-Red and IBM Watson IoT Platform.	1.Open Node-red using IBM cloud 2.Installing package to connect with IBM watson and configure the node with the Authentication Key and ID using IBM watson iot platform.3 Arrange the functional nodes for the parameters and configure them 4.connect all nodes with msgpayload and deploy them. STEP 2 :Create an account in IBM cloud using your email ID 1.Create IBM Watson Platform in services in your IBM cloud account 2. Launch the IBM Watson IoT Platform 3.Create a new device 4.Give credentials like device type, device ID, Auth. Token 5.Create API key and store API key and token elsewhere		IBM watson And Node Red connecton must be established	Working as expected	Pass	Bala vignesh kumar , Senthil sachin
IBM Watson and Python Integration	IBM CLOUD platform /Python 3.62 application.	To Generate values for the parametres Temperature,Humidity,Soil moisture	STEP:1.Install Python 3.6.2 2.Install the package wiotp.sdk. 3.Import the package in python 4.provide the device credentials from IBM iot watson platform STEP 2:Open python 2.Write a program to generate variables for the parameters using random library.3.Run the program	python code	The code must run and the values must be generated.	Working as expected	pass	Hari prasath, Dhilip
Node-Red	IBM IOT/MIT/WEB PAGE	To establish connection to IBM iot watson platform and then configuring Node Red for the parameters	1.Open Node-red using IBM cloud 2.Installing package to connect with IBM watson and configure the node with the Authentication Key and ID using IBM watson iot platform.3 Arrange the functional nodes for the parameters and configure them 4.connect all nodes with msgpayload and deploy them.	Functional Flow Chart	From the Dashboard the specified values can be seen as layout or use the debug window to view the generated values .	working as expected	pass	Hari prasath , Bala vignesh kumar
Mit app(Front end and Back end)	Login page / Screen 1&2	Verify user is able to log into application with Valid credentials	1.Use the components given in the app to build the login page 2.components like text box(specified),Buttons and variations in color and allignments to be made.3.Enter URL(https://shopenzer.com/) and click go 4.Click on My Account dropdown button 5.Enter Valid username in dashboard box	Username: user1 password: IBM	Application should show 'Incorrect username or password ' validation message.Application should redirect to the next page if the password is correct,else it will show check your	working as expected	pass	Bala vignesh kumar , Dhilip
Mobile application	Final Screen	To verify the User name and password and decide what to perform.	1.Login to mobile application with user name and password. 2.values like temperature ,humidity and soil moisture will be displayed in mobile application.	Username: user1 password: IBM	Depending on the values of temperature,humidity,soil moisture, windspeed, wind degree, weather how much cloud present in sky ,when critical situation araises motor will ON or OFF.Application should show 'Incorrect username or password ' validation message.	Executed Successfully.	Pass	Bala vignesh kumar

Table Figure 8.1 Test case report

8.2. USER ACCEPTANCE TESTING

1. Purpose of Document

IOT-based agriculture system helps the farmer in monitoring different parameters of his field likesoil moisture, temperature, and humidity using some sensors and providing weather condition,
Wind speed and degree, how much cloud present in sky

Farmers can monitor all the sensor parameters by using a web or mobile application even if the farmer is not near his field. Watering the crop is one of the important tasks for the farmers.

2. TestCaseAnalysis

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

Section	Total Cases	Not Tested	Fail	Pass
Install python	10	0	5	5
Launch IBM Watson	15	0	7	8
IBM Watson and python integration	20	0	8	12
Install Node red	2	0	0	2
Interconnecting IBM Watson and node red	30	0	10	20
Web UI dashboard	10	0	0	10
MIT app design	50	10	20	20
To view the values in mobile application	5	0	0	5

Table Figure 8.2 User Acceptance testing

CHAPTER 9

RESULTS

9.1. PERFORMANCE METRICS

			NFT - Risk Assessment						
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Voluem Changes	Risk Score	Justification
1	SmartFarmer - IoT Enabled	IoT-based agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors and providing weather condition, Wind speed and degree, how much cloud present in sky	User permission for irrigation via Mobile Application And software Web UI Application	No Changes	Moderate	No changes	>50 to 100%	100% score will be displayed in mobile application with no risk(with internet or without internet	As we have seen the changes in connectivity issues

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	mptions/Dependencies/Risks	
1	SmartFarmer - IoT Enab	moderate	No risk	

			End Of Test Report					
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open	Approvals/SignOff
1	SmartFarmer - IoT Enablec	moderate	usability,security,realit	Pass	Pass	Nil	NIL	NIL

Table Figure 9.1 Performance Metrics

CHAPTER 10

ADVANTAGES & DISADVANTAGES

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.

DISADVANTAGES

- Smart Agriculture requires internet connectivity continuously, but rural parts 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.

CHAPTER 11

CONCLUSION

By using this system farmers can effectively produce more yield and can save water from wastage. With help of weather forecast service farmer can water their land as per weather. Farmer can also turn ON/OFF motor whenever required based on the water content in soil. They can also analyze the values and also predict the upcoming weather forecast and act accordingly. Experienced farmer can easily teach the new learners by speaking in terms of value and the change in the values. In total the agriculture is getting benefitted as well the Economy of India that is the GDP is getting benefitted.

CHAPTER 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.

IoT have the potential to transform agriculture in many aspects and these are the main ones. Data collected by smart agriculture sensors, in this approach of farm management, a key component are sensors, control systems, robotics, autonomous vehicles, automated hardware, variable rate technology, motion detectors, button camera, and wearable devices. This data can be used to track the state of the business in general as well as staff performance, equipment efficiency. The ability to foresee the output of production allows to plan for better product distribution. Agricultural Drones Ground-based and aerial-based drones are being used in agriculture in order to enhance various agricultural practices: crop health assessment, irrigation, crop monitoring, crop spraying, planting, and soil and field analysis. Livestock tracking and geofencing Farm owners can utilize wireless IoT applications to collect data regarding the location, well-being, and health of their cattle. Smart Greenhouses A smart greenhouse designed with the help of IoT intelligently monitors as well as controls the climate, eliminating the need for manual intervention. Predictive analytics for smart farming Crop predication plays a key role, it helps the farmer to decide future plan regarding the production of the crop, its storage, marketing techniques and risk management. To predict production rate of the crop artificial network use information collected by sensors from the farm. This information includes parameters such as soil, temperature, pressure, rainfall, and humidity. The farmers can get an accurate soil data either by the dashboard or a customized mobile application.

CHAPTER 13

APPENDIX

GitHub link: https://github.com/IBM-EPBL/IBM-Project-11064-1659258183

Video Link: https://youtu.be/BUq__W9ahuQ