SMART FARMER-IOT ENABLED SMART FARMING APPLICATION

PROJECT REPORT

IBM-Project-13788-1659530472 TEAM ID: PNT2022TMID48383

Submitted by

KAVIYA	**913319106010**
ARNIKASHREE	**913319106005**
ARUN	**913319106006**
THIRUMURUGAN	**913319106038**
PRIYADHARSHINI	**913319106024**

BACHELOR OF ENGINEERING

ELECTRONICS AND COMMUNICATION ENGINEERING

VAIGAI COLLEGE OF ENGINEERING
MADURAI

PROJECT REPORT

1. INRODUCTION

1.1 Project Overview

IOT- Enabled Smart Farming agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature, humidity using some sensors. Farmer 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 task for the farmers. They can make the decision whether to water the crop or postpone it by monitoring the sensor parameters and control the motor pumps from the mobile application itself. All the sensor parameters are stored in the IBM Cloudant DB

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 and temperature conditions, sensors IBM IoT Simulator is used which can transmit soil moisture temperature as required..

Project requirements: Node-RED, IBM Cloud, IBM Watson IoT, Node.js, IBM Device, IBM IoT Simulator, Python 3.7, Open Weather API platform.

Project Deliverables: Application for IoT based Smart Agriculture System

1.2 Purpose

IoT based farming 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.

Smart agriculture is a farming system which uses IoT technology. This emerging system increases the quantity and quality of agricultural products. IoT devices provide information about nature of farming fields and then take action depending on the farmer input**.**

The main goal of my project is to use IoT in the agriculture field in order to collect data instantly (soil Moister, temperature, humidity…), which will help one to monitor some environment conditions remotely, effectively and enhance tremendously the production and therefore the income of farmers. The present prototype is developed using Arduino technology, which comprise specific sensors, and a WIFI module that helps to collect instant data online. Worth mentioning the testing of this prototype generated, highly accurate data because while we were collecting them remotely any environmental changes were detected instantly and taking in consideration to make decisions.

2. LITERATURE SURVEY

2.1 Existing Problem

Watering the field is a difficult process, Farmers have to wait in the field until the water covers the whole farm field. Power Supply is also one of the problems. In Village Side, the power supply may vary. The Biggest Challenges Faced by IoT in the Agricultural Sector are Lack of Information, High Adoption, Cost and Security Concerns, etc The farmers do not have that much knowledge on the internet of things and good internet connection is required. So farmers don't know how to use the web application and to make a connection if any component get failed.

2.2 Problem Statement Definitions

The Biggest Challenges Faced by IoT in the Agricultural Sector are Lack of Information, High Adoption, Cost and Security. The farmers do not have that much knowledge on the internet of things and good internet connection is required. Power Supply is also one of the problems In Village Side, the power supply may vary. So farmers don't know how to use the web application and to make a connection if any component get failed.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges

Empathy Map

Reference:

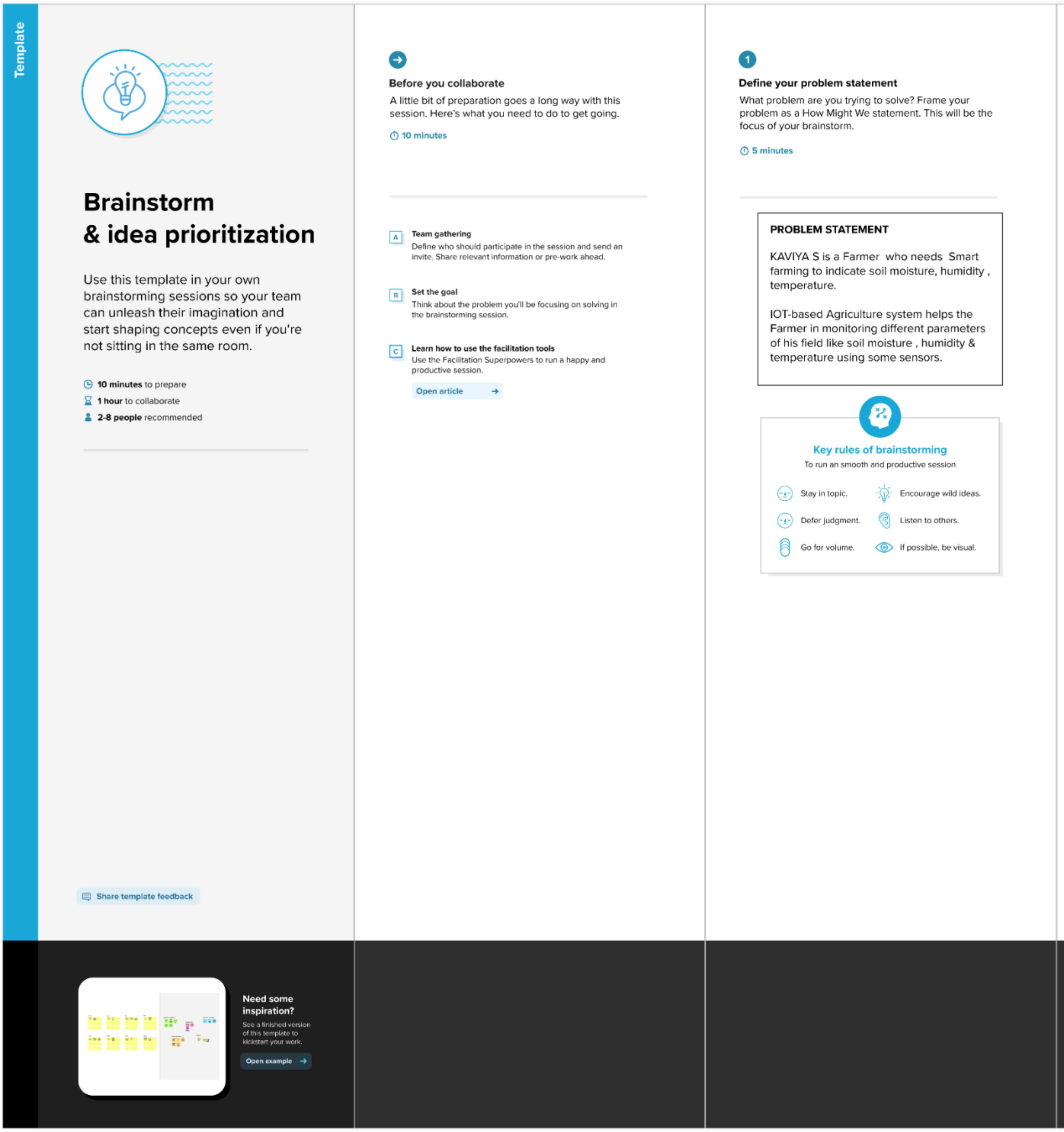
https://app.mural.co/embed/47db0758-2c69-4f64-8df8-065f67eae5c0

3.2 Ideation and Brainstorming

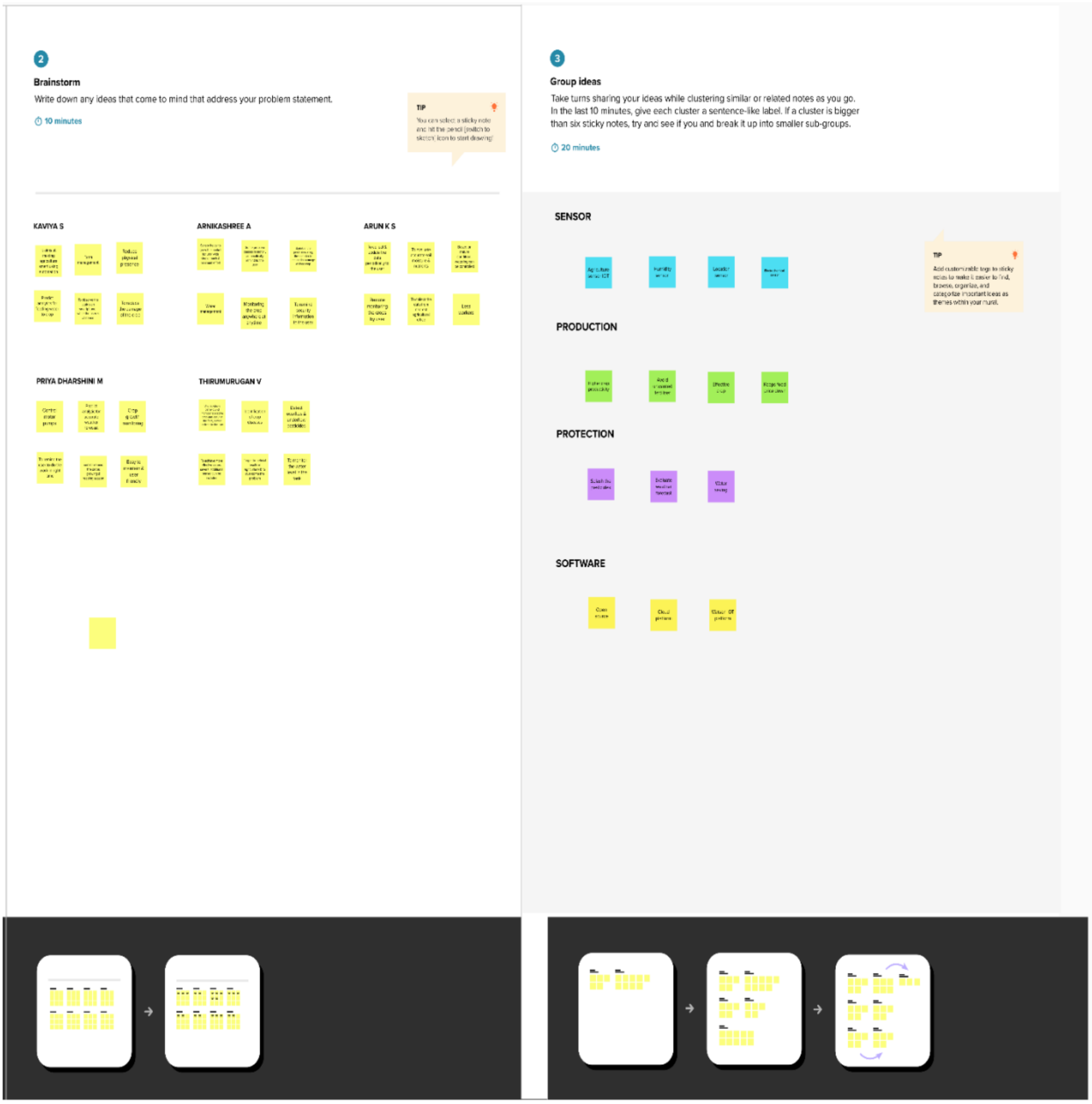
Reference :

https://app.mural.co/embed/79fb6052-e4ac-4672-b0a5-35104389dba6

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



Step-2: Brainstorm ,Idea Listing and Grouping



Step-3: Idea Prioritization

3.3 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To develop web application which automatically sense and monitor the field even if the farmer near the field. whether the farmer want to postpone watering the crop, which can be done by mobile application itself.
2.	Idea / Solution description	Our project aim at developing a web application that built for sensing or monitoring information, such as soil condition, temperature and the prediction of natural factors like rainfall and weather, with the help of various sensor like light, temperature, humidity, soil moisture, crop health etc.. By using this application farmer can monitor the field conditions from anywhere.
3.	Novelty / Uniqueness	The unique feature of our application is easy to operate. When some problem causes in the farm, the sensor indicate us by the application.
4.	Social Impact / Customer Satisfaction	It will help the people with providing high yield and healthy crops. Our

		application indicate us before any hazards occurs.
5.	Business Model (Revenue Model)	Social media is the best way to spread our application in a good manner and with influencers we can attract the normal people.
6.	Scalability of the Solution	It provides service for the user or farmer which is monitored 24/7

3.4 Problem Solution fit

1. CUSTOMER SEGMENT(S) * Persons who have less number of farming knowledge to monitor or manage one or more farms.	**6.Customer Constraints** * Network connection, high adoption costs, and security concerns.	**5. AVAILABLE SOLUTIONS** *To increase the quantity and quality of agriculture products.

2. JOBS-TO-BE-DONE / PROBLEMS * Cope with climate change, soil erosion and biodiversity loss.	**9. PROBLEM ROOT CAUSE** * To alleviate security concerns, we use sensors to detect real-time status.	**7. BEHAVIOUR** *With the help of IOT devices you can know the real-time status of the crops.

3. TRIGGERS TR	10. YOUR SOLUTION SL	8. CHANNELS of BEHAVIOUR CH
* Meeting other who have better cost management by using smart farming application. * Watching more benefits from using smart farming application in social media. **4. EMOTIONS: BEFORE / AFTER** * Before - High paid cost spending more time in farms to manage. Fear about sudden climate change. *After – Satisfied. Feeling secured. Better understanding about factors such as water, climate changing etc….	* Our patented sensors technology requires no batteries or wires and communicates wirelessly to a reader over a distance of as much as 19 meters. * The sensors can sense applicators to apply less nitrogen to healthy plants and more nitrogen to weaker, unhealthy plants.	8.1 ONLINE *Easy to monitoring from anywhere, controlling resources easily and effectively. 8.2 OFFLINE * Spending more time to manage crops in farms, appoint people with salary to monitor farms.

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	Github and slackAccount	Registering the account in both.
FR-2	IBM Account	Registering the account and login in.
FR-3	Node-RED	Creating the account and made the connection.
FR-4	Python	Encode the python code.
FR-5	Open Weather API	Get the data and access the resource.
FR-6	MIT app inventor	Control the motor through the application.

4.2 Non-functional Requirements:

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

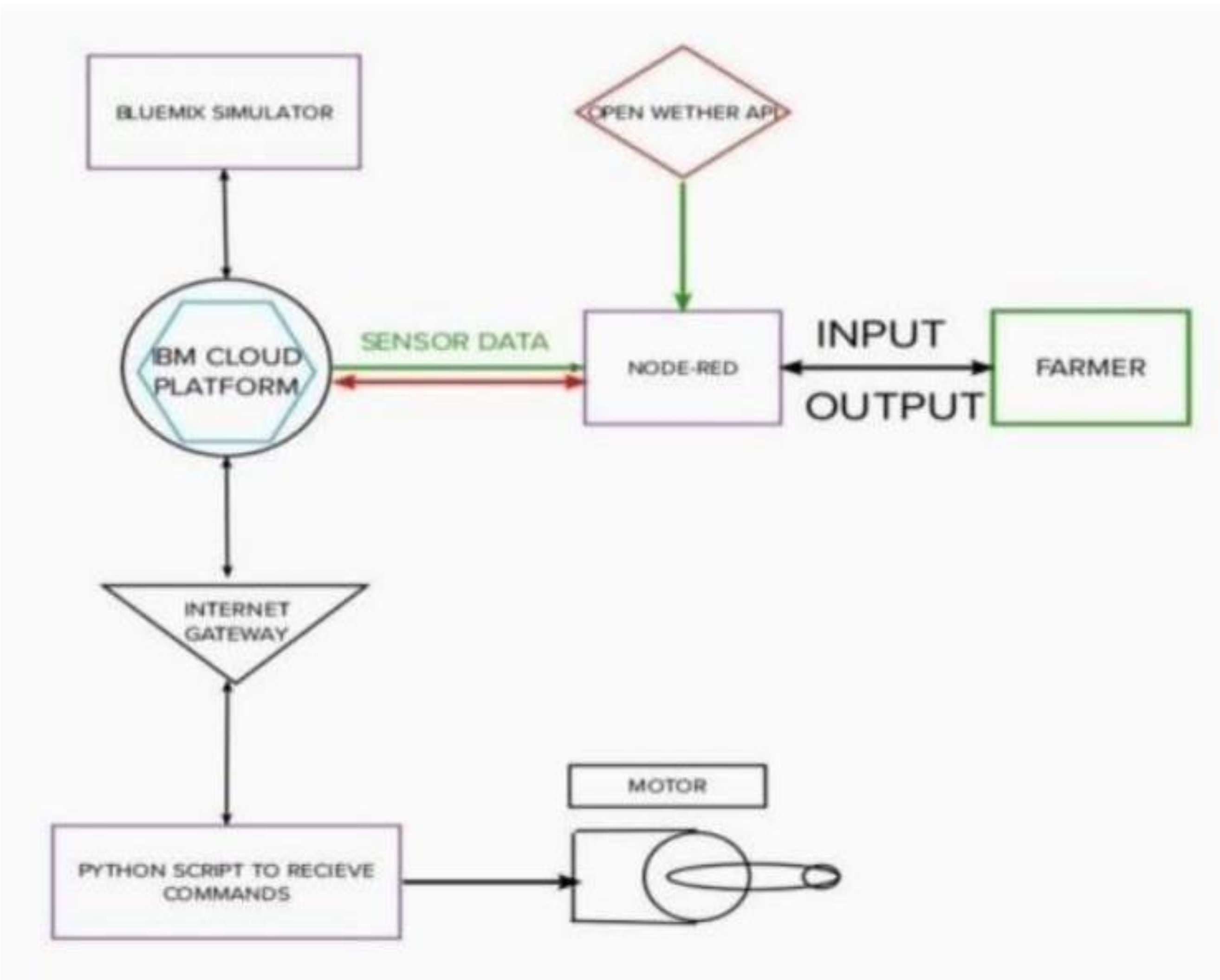
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The mobile application can monitor the temperature, humidity, pressure and soil moisture parameters along with weather forecasting details. Based on these details he can water the crops by controlling the motors through the app and the app gives an alert message if temperature or humidity goes beyond a threshold

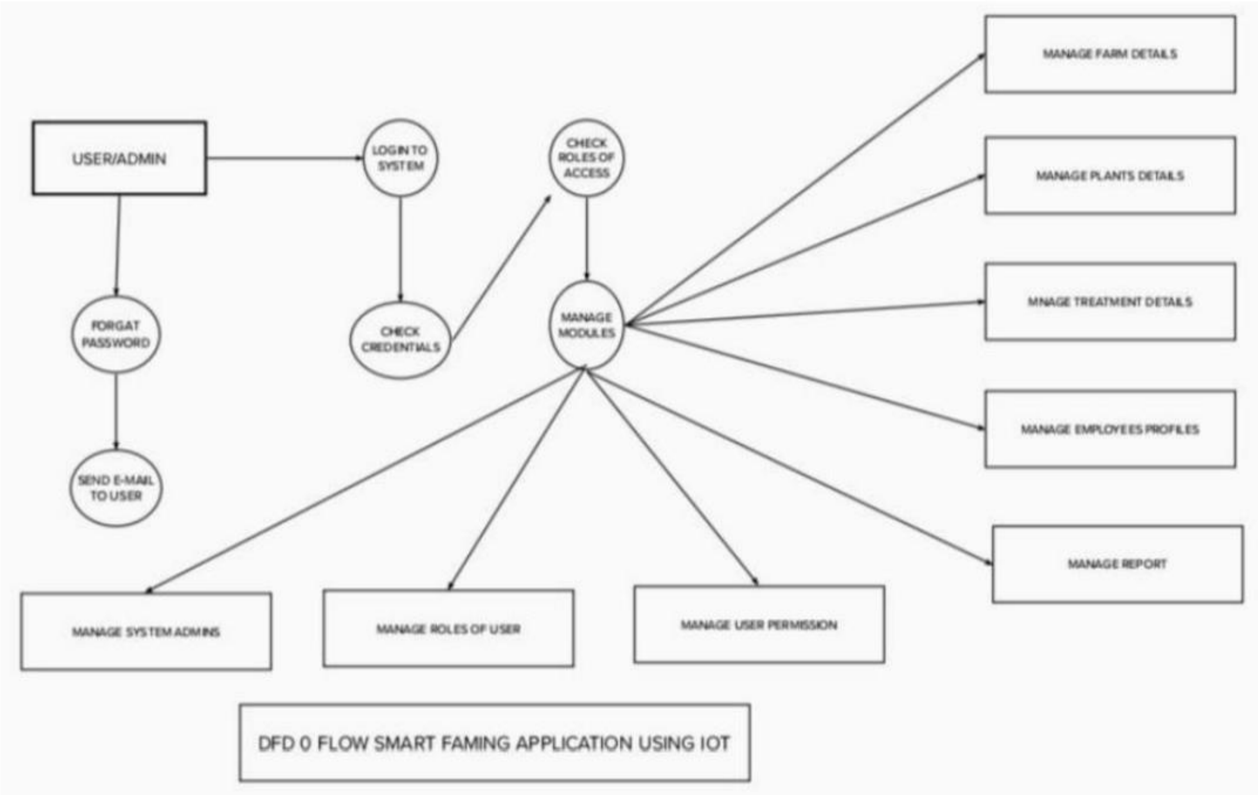
		value.
NFR-2	**Security**	The system needs the patient to recognize herself or himself using the phone. Any users who make use of the system.
NFR-3	**Reliability**	It specifies how likely the system or its element would run without a failure.
NFR-4	**Performance**	The system provides acknowledgment in just one second.The user interface acknowledges

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, andwhere data is stored.





1. The different soil parameters temperature, soil moistures and then humidity are sensedusing different sensors and obtained value is stored in the Ibm cloud.

2. Arduino UNO is used as a processing Unit that process the data obtained from thesensors and whether data from the weather API.

3. NODE-RED is used as a programming tool to write the hardware, software and APIs.The MQTT protocol is followed for the communication.

4. All the collected data are provided to the user through a mobile application that was developed using the MIT app inventor. The user could make a decision through an app, weather to water the crop or not depending upon the sensor values. By using theapp they can remotely operate to the motor switch.

5.2 Solution & Technical Architecture

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2.

Technologies used in Smart Farming:

- Edge-based sensor systems in smart farming
- Low energy machine-learning algorithms for edge-based sensors in smart farming
- Energy harvesting (image) sensor systems in smart farming
- Advanced image processing techniques and applications in smart farming
- Emerging IoT-based sensor applications in smart farming
- Sensing hardware platforms in smart farming
- Security solutions for sensing hardware in smart farming
- Energy efficient network-based analysis in smart farming
- Low energy wireless connectivity solutions for smart farming (LoRa, NB-IoT, etc.)
- Multimodal sensor integration in smart farming
- Emerging sensing methods (hyperspectral imaging, compressed sensing, etc.)

Example: Smart farming using IOT

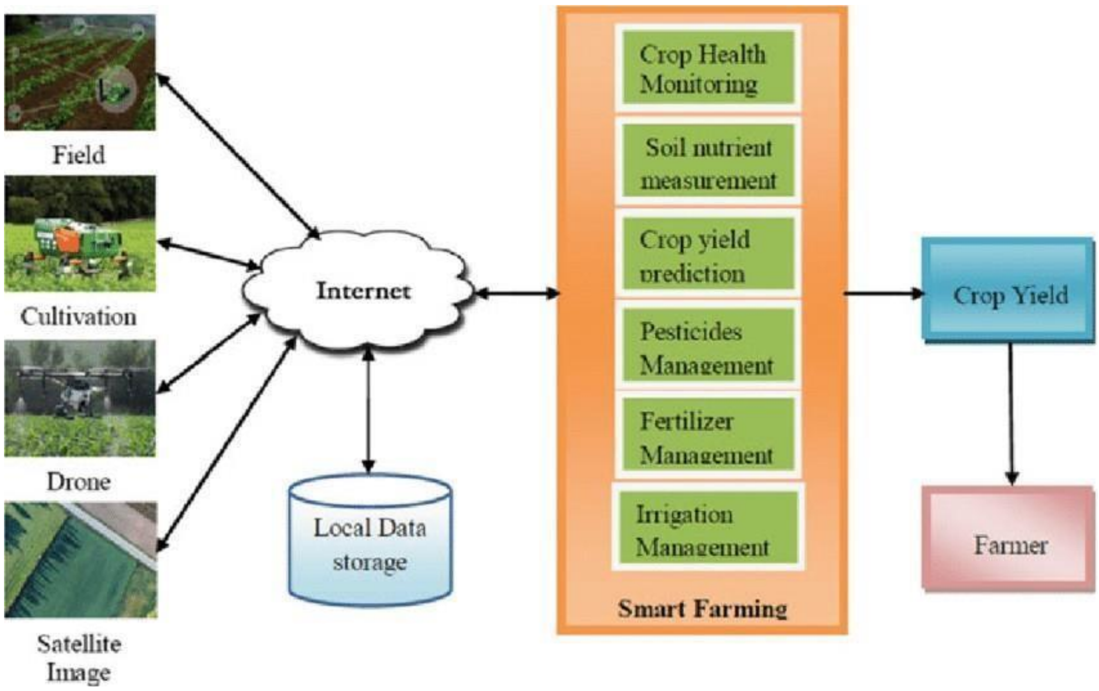


Table-1 :

Component	Description	Technology

Components & Technologies: **S.No**

1.	**User Interface**	user interface designed for smart farms which are controlled by mobile phones and applications user requirements and experience as a strategy for defining the scope and structure of the crop field	Internet Of Things(IOT), Artificial Intelligence (AI)
2.	**Application Logic-1**	monitoring the water levels in tanks. tracking of seed-growth	IOT, Cloud computing AI, Machine learning
3.	**Application Logic-2**	crop health, crop monitoring, planting, crop spraying, and field analysis.	Ground and Aerial drones
4.	**Application**	to maintain	data

	Logic-3	the quality of crops and fertility of the land, thus enhancing the product volume and quality.	analytics, Cloud, IOT
5.	**Database**	massive quantities of data, such as streaming data, time-series data, RFID data, and sensory data, among other things.	SQLite Database, MySQL
6.	**Cloud Database**	Database Service On Cloud	IBM DB2, IBM Cloudant etc.
7.	**File Storage**	Monitoring, Sensors and requirements	IBM Block Storage or Other Storage Service or Local File system
8.	**External API-1**	API done well are the most efficient way to connect data,	IBM Weather API, Robotics, AI etc.

		thereby enhancing the overall IOT value	
9.	**External API-2**	the logical connectors that allow applications to communicate with each manufacturer's IOT devices	API and IOT
10.	**Machine Learning Model**	collect the data, train the systems and predict the results	Machine Learning
11.	**Infrastructure (Server / Cloud)**	Application Deployment on Local System / Cloud Local Server Configuration: TCP and UDP Cloud Server Configuration : CM	Local servers , Cloud, Wireless Sensor Network (WSN)

Table-2: Application Characteristics: S.No	Characteristics	Description	Technology
1.	**Open-Source**	Things Board,	HTTP, Web Socket,

	Frameworks	Thing Speak, My Devices	edge computing
2.	**Security Implementations**	GSM, Firewall	Confidentiality, Integrity and Availability Triad
3.	**Scalable Architecture**	Collaborate and Connect	Artificial Intelligence
4.	**Availability**	Monitoring greenhouses	PS and GIS

5.3 User Stories

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Simulation creation and Python code development	USN-1	Connect Sensors Wi-fi Module with python code and pubsub python code.	2	High	kaviya
Sprint-2	Connecting python code with IBM Watson platfor	USN-2	Creating device in the IBM Watson IoT platform,	2	High	Arnikashree

	m and node-red work flow		workflow for IoT scenarios using Node-Red			
Sprint-3	Creating MIT App Inventor and designing front end like username and password	USN-3	Developing an application for the Smart farmer project using MIT App Inventor	2	High	Arun
Sprint-3	Developing the backend of the mit app using blocks	USN-3	Design the Modules and test the app	2	High	Thirumurugan
Sprint-4	Web UI	USN-4	To make the user to interact with software.	2	High	priyadharshini

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint planning & Estimation

Title	Description	Duration
Literature Survey & Information Gathering	Literature survey on the selected project & gatheringinformation by referring the, technical papers, research publications etc.	29 August-3rd September 2022
Prepare Empathy Map	Prepare Empathy Map Canvasto capture the user Pains & Gains, Prepare list of problem statements.	5-10 th September 2022
Brainstorming ideas	List the ideas by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	12-17 September 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	19-24 September 2022
Problem Solution Fit	Prepare problem - solution Fitdocument.	26 September-01 October 2022
Solution Architecture	Prepare solution Architecturedocument.	26 September-01 October 2022
Customer Journey	Prepare the customer journeymaps to understand the user interactions & experiences with the application	03-08 October 2022
Data Flow Diagrams	Draw the data flow Diagramsand submit for review.	10-15 October 2022
Technology	Architecture diagram.	10-15 October 2022

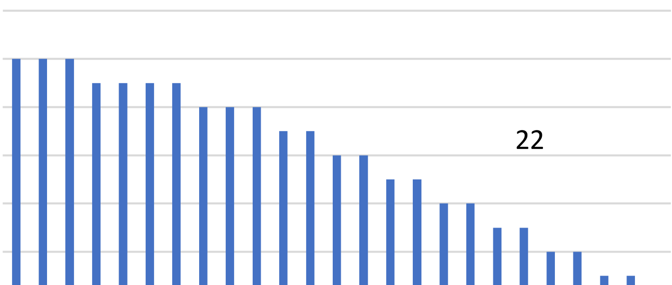
Architecture		

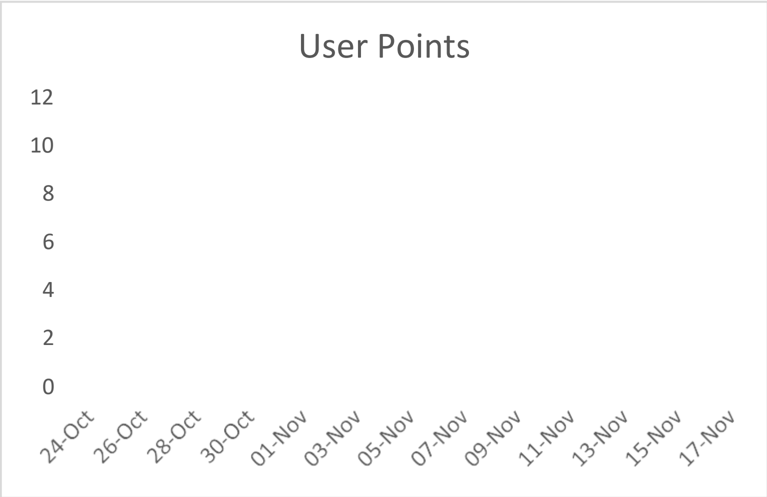
Milestone & Activity List	Prepare the milestones & Activity list of the project.	17-22 October 2022
Sprint Delivery	Prepare the Sprint delivery onNumber of Sprint planning meetings organized, Minutes of meeting recorded.	17-22 October 2022
Project Development Deliveryof Sprint-1,2,3&4	Develop & submit the developed code by testing it.	In Progress

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	28 Oct 2022
Sprint-2	20	5 Days	31 Oct 2022	04 Nov 2022	20	03 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	4 Days	14 Nov 2022	17 Nov 2022	20	16 Nov 2022

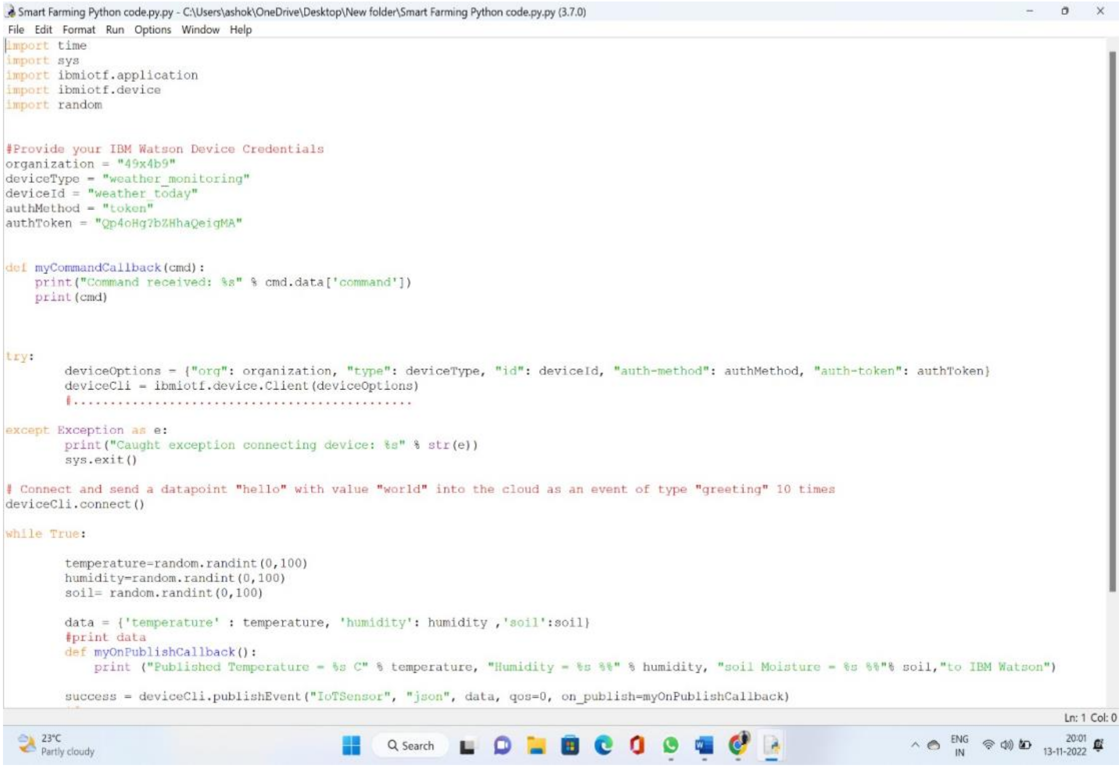
6.3 Report from JIRA




User Points
12
10
8
6
4
2
0
24-Oct
26-Oct
28-Oct
30-Oct
01-Nov
03-Nov
05-Nov
07-Nov
09-Nov
11-Nov
13-Nov
15-Nov
17-Nov

7. CODING & SOLUTIONING

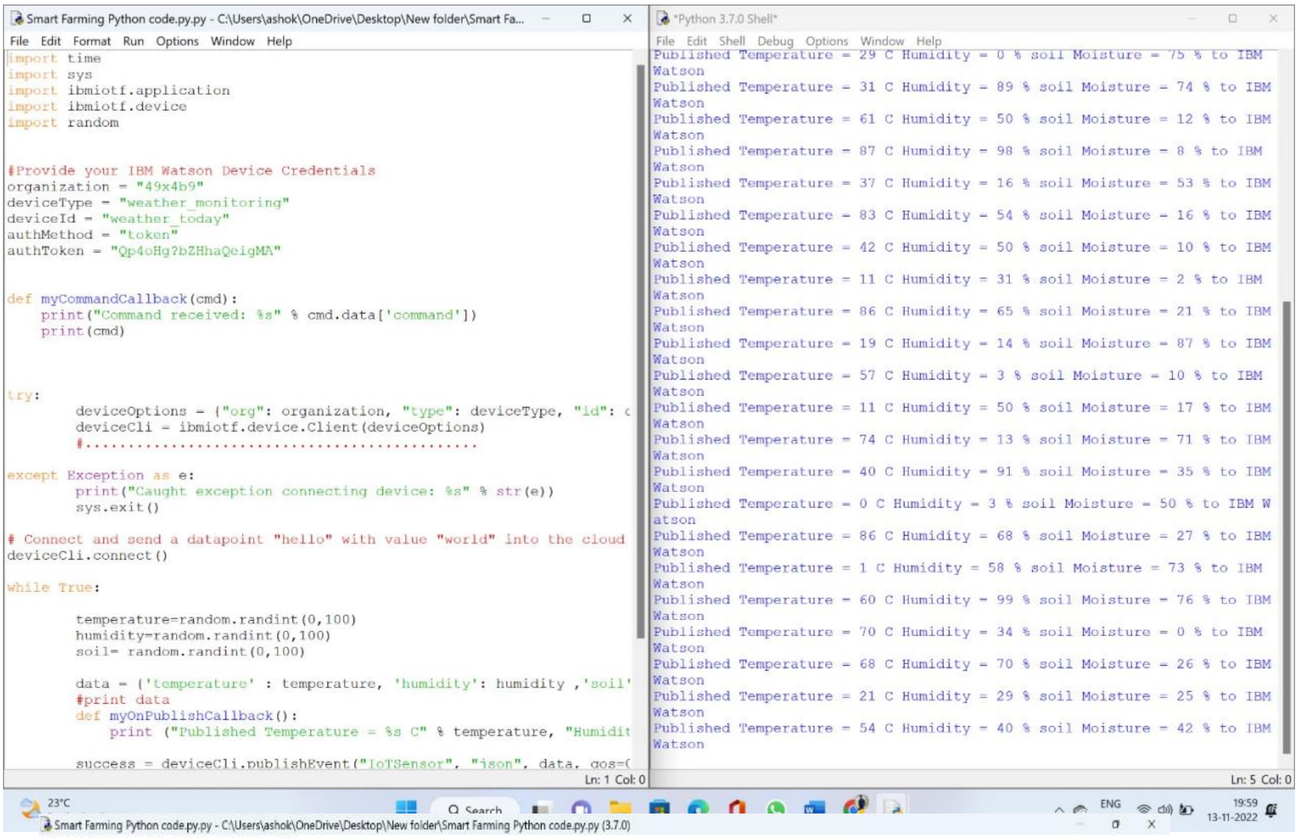
7.1 Feature 1

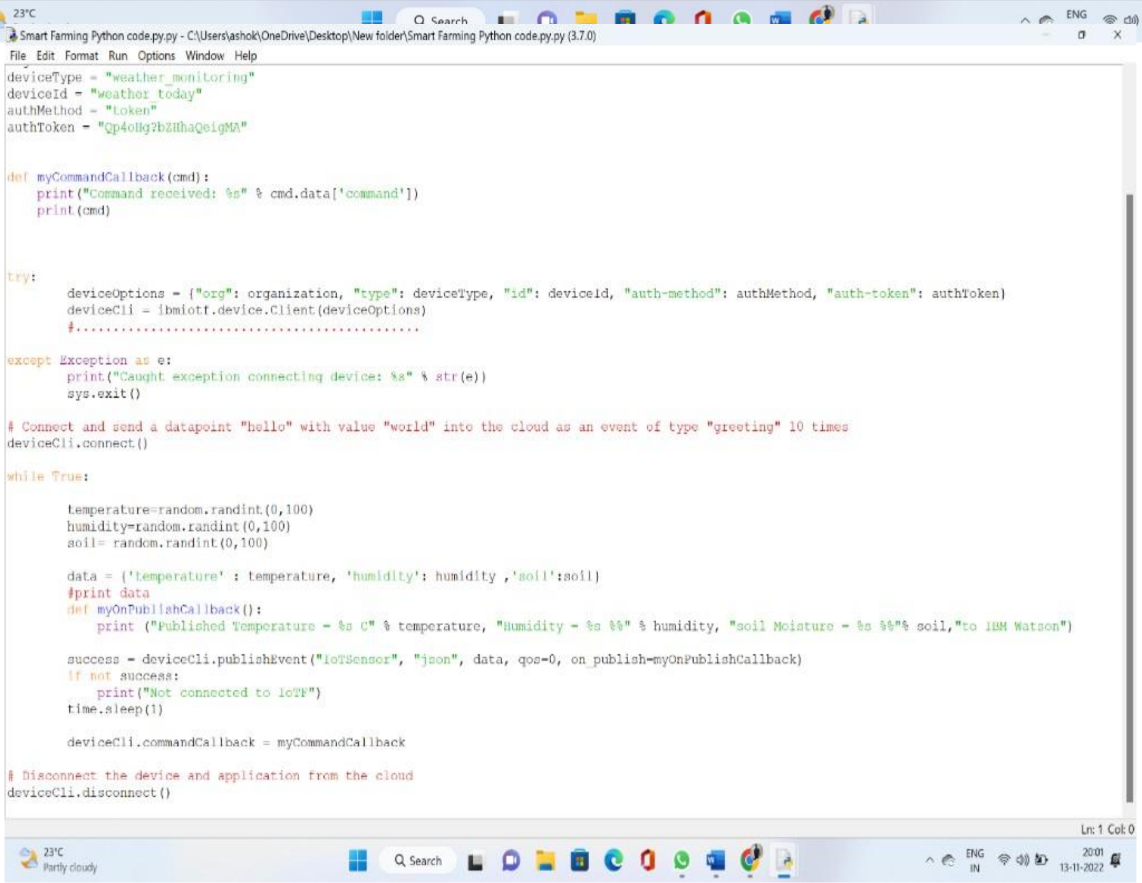

Smart Farming Python code.py.py - C:\Users\ashok\OneDrive\Desktop\New folder\Smart Farming Python code.py.py (3.7.0)

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 = "49x4b9"
deviceType = "weather_monitoring"
deviceId = "weather_today"
authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"


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

        temperature=random.randint(0,100)
        humidity=random.randint(0,100)
        soil= random.randint(0,100)

        data = {'temperature' : temperature, 'humidity': humidity ,'soil':soil}
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temperature, "Humidity = %s %%" % humidity, "soil Moisture = %s %%"% soil,"to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
```

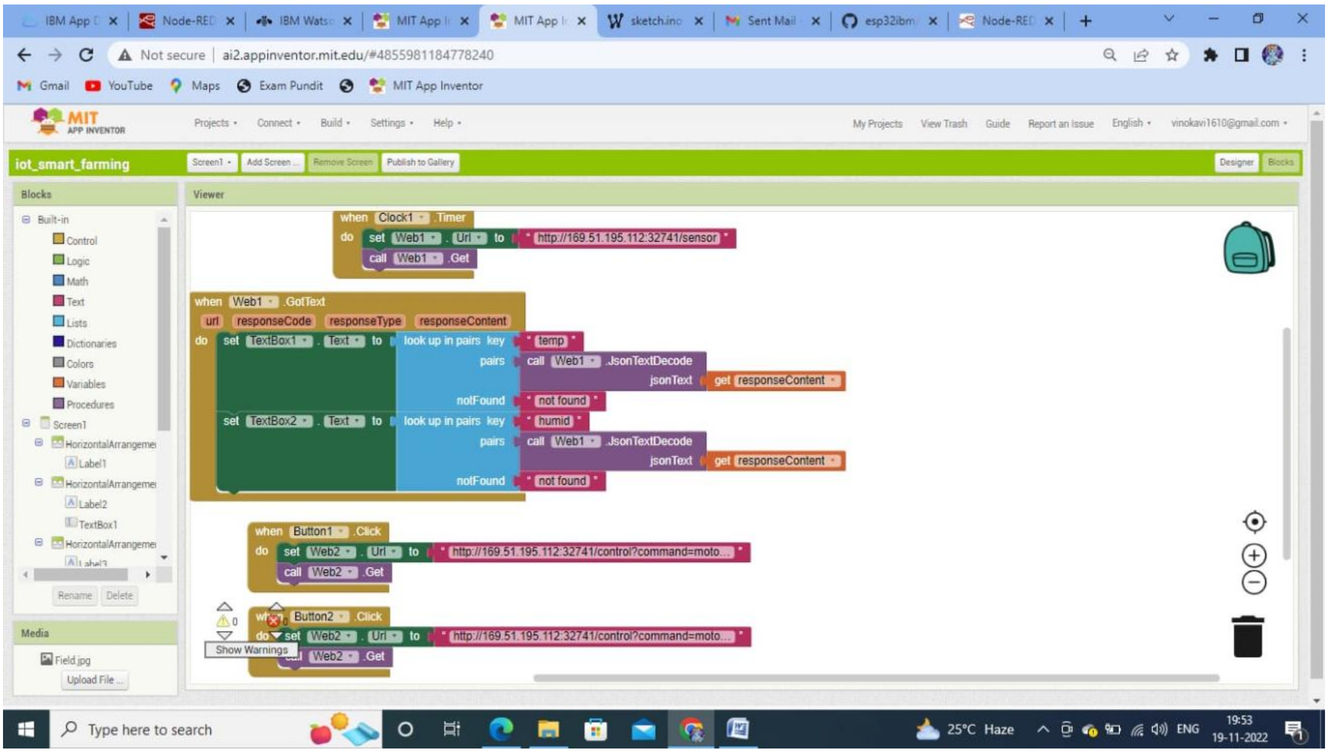
Ln: 1 Col: 0


Smart Farming Python code.py.py - C:\Users\ashok\OneDrive\Desktop\New folder\Smart Fa...
File Edit Format Run Options Window Help
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "49x4b9"
deviceType = "weather_monitoring"
deviceId = "weather_today"
authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
print(cmd)
try:
deviceOptions = {"org": organization, "type": deviceType, "id": c
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
deviceCli.connect()
while True:
temperature=random.randint(0,100)
humidity=random.randint(0,100)
soil= random.randint(0,100)
data = {'temperature' : temperature, 'humidity': humidity ,'soil'
#print data
def myOnPublishCallback():
print ("Published Temperature = %s C" % temperature, "Humidit
success = deviceCli.publishEvent("IoTSensor", "json", data, qos=(
Ln: 1 Col: 0
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published Temperature = 29 C Humidity = 0 % soil Moisture = 75 % to IBM Watson
Published Temperature = 31 C Humidity = 89 % soil Moisture = 74 % to IBM Watson
Published Temperature = 61 C Humidity = 50 % soil Moisture = 12 % to IBM Watson
Published Temperature = 87 C Humidity = 98 % soil Moisture = 8 % to IBM Watson
Published Temperature = 37 C Humidity = 16 % soil Moisture = 53 % to IBM Watson
Published Temperature = 83 C Humidity = 54 % soil Moisture = 16 % to IBM Watson
Published Temperature = 42 C Humidity = 50 % soil Moisture = 10 % to IBM Watson
Published Temperature = 11 C Humidity = 31 % soil Moisture = 2 % to IBM Watson
Published Temperature = 86 C Humidity = 65 % soil Moisture = 21 % to IBM Watson
Published Temperature = 19 C Humidity = 14 % soil Moisture = 87 % to IBM Watson
Published Temperature = 57 C Humidity = 3 % soil Moisture = 10 % to IBM Watson
Published Temperature = 11 C Humidity = 50 % soil Moisture = 17 % to IBM Watson
Published Temperature = 74 C Humidity = 13 % soil Moisture = 71 % to IBM Watson
Published Temperature = 40 C Humidity = 91 % soil Moisture = 35 % to IBM Watson
Published Temperature = 0 C Humidity = 3 % soil Moisture = 50 % to IBM Watson
Published Temperature = 86 C Humidity = 68 % soil Moisture = 27 % to IBM Watson
Published Temperature = 1 C Humidity = 58 % soil Moisture = 73 % to IBM Watson
Published Temperature = 60 C Humidity = 99 % soil Moisture = 76 % to IBM Watson
Published Temperature = 70 C Humidity = 34 % soil Moisture = 0 % to IBM Watson
Published Temperature = 68 C Humidity = 70 % soil Moisture = 26 % to IBM Watson
Published Temperature = 21 C Humidity = 29 % soil Moisture = 25 % to IBM Watson
Published Temperature = 54 C Humidity = 40 % soil Moisture = 42 % to IBM Watson
Ln: 5 Col: 0
23°C
Search
ENG
19:59
13-11-2022


Smart Farming Python code.py.py - C:\Users\ashok\OneDrive\Desktop\New folder\Smart Farming Python code.py.py (3.7.0)
File Edit Format Run Options Window Help
deviceType = "weather_monitoring"
deviceId = "weather_today"
authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
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:
temperature=random.randint(0,100)
humidity=random.randint(0,100)
soil= random.randint(0,100)
data = {'temperature' : temperature, 'humidity': humidity ,'soil':soil}
#print data
def myOnPublishCallback():
print ("Published Temperature = %s C" % temperature, "Humidity = %s %%" % humidity, "soil Moisture = %s %%"% soil,"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()
Ln: 1 Col: 0
23°C
Partly cloudy
Search
ENG
IN
20:01
13-11-2022

7.2 Feature 2

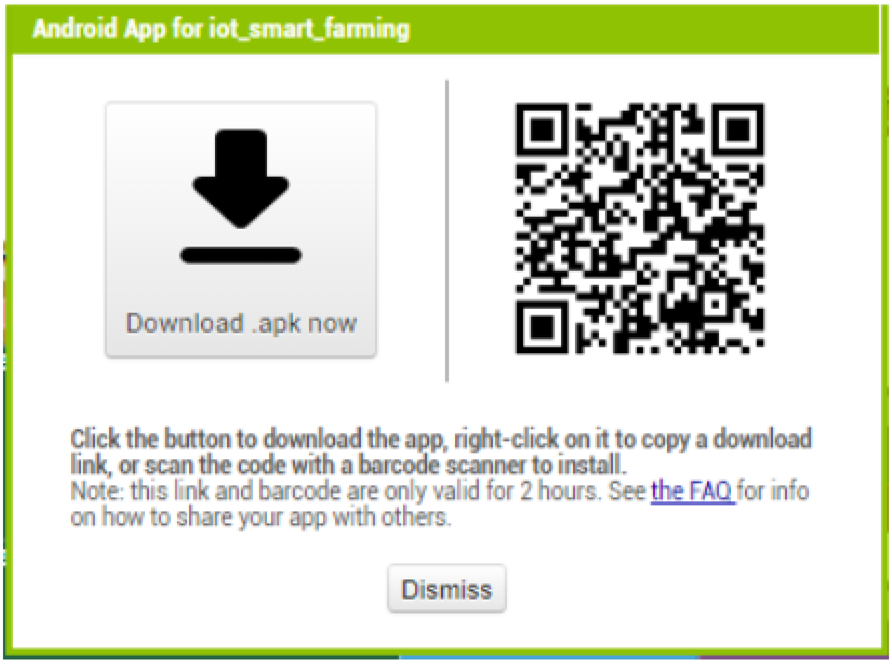
These are the blocks of the login and signup page of mobile application.



8. TESTING

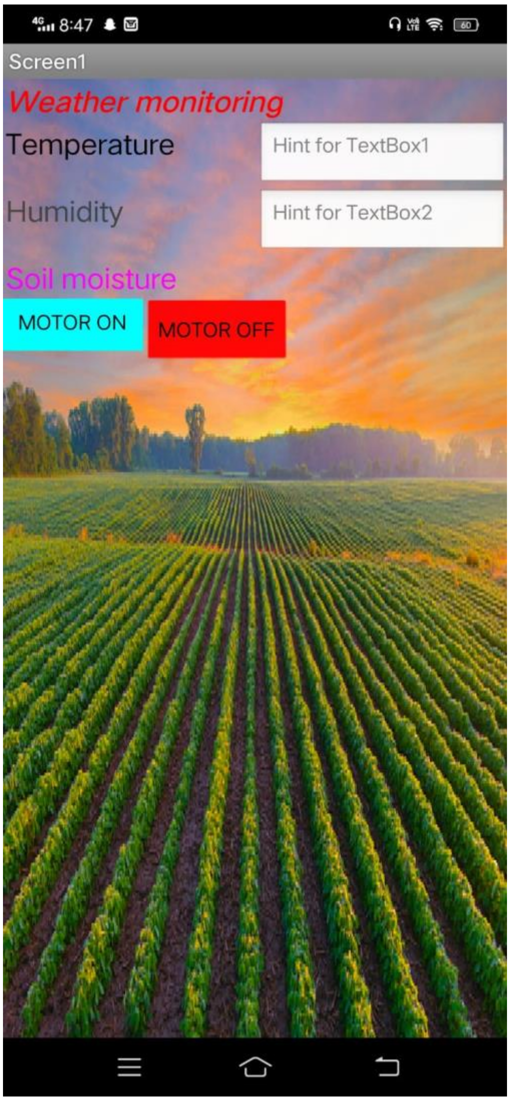
8.1 Test Cases

Step-1: First user need to download the android APK file from MIT appinventor where we developed our mobile application and install in their mobiles.



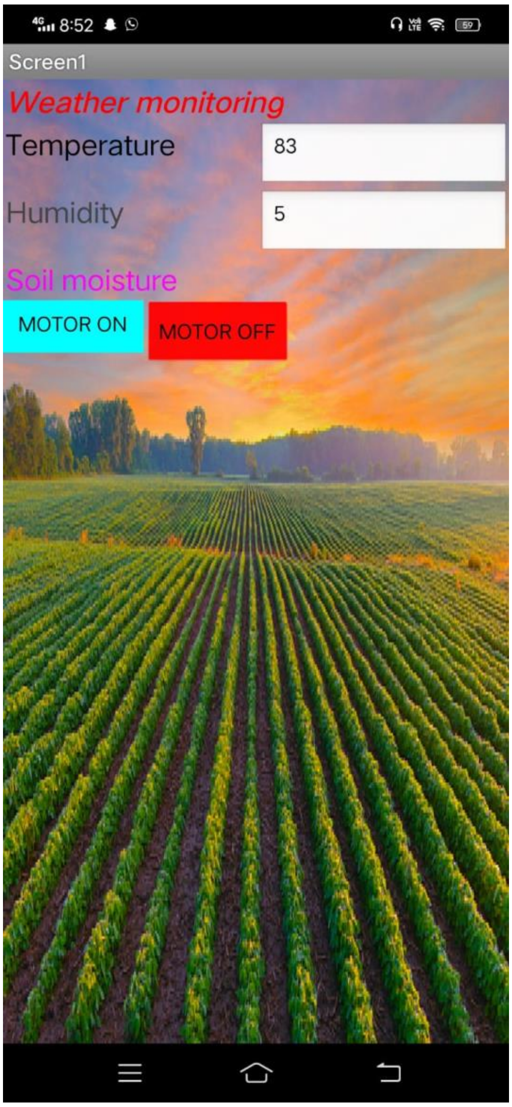
Step-2: After successful installation we can find app icon in our mobile asshown below.

Step-3: After clicking the app icon. The usercan see interface like these as shown below.


8:47
Screen1
Weather monitoring
Temperature
Hint for TextBox1
Humidity
Hint for TextBox2
Soil moisture
MOTOR ON
MOTOR OFF

8.2 User Acceptance Testing

In that page we can see thereal time temperature , humidity and motor ON and motor OFF control button also as shown below.



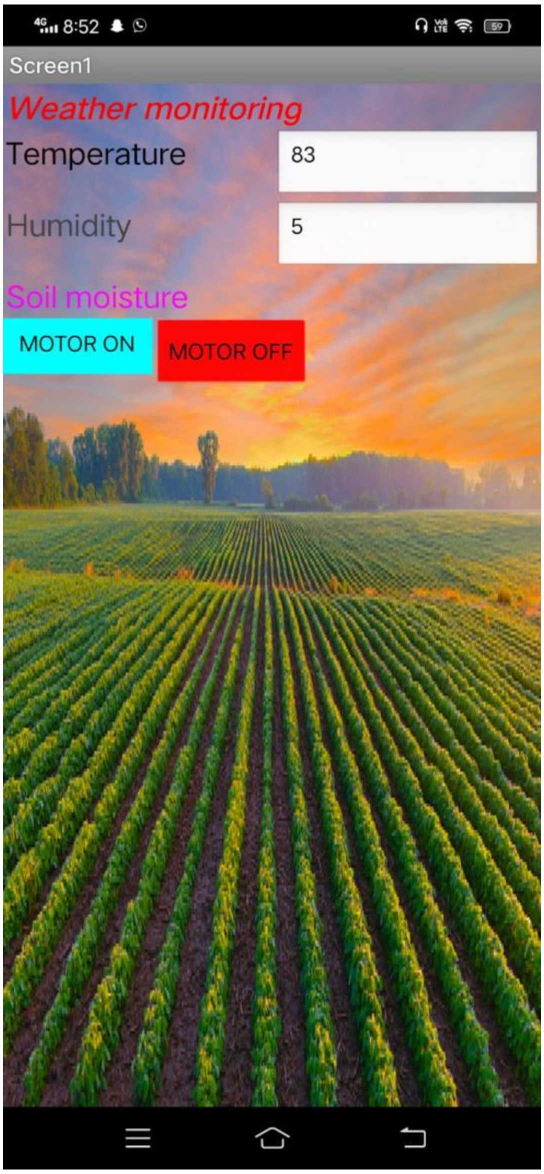
we are successfully created the IOT enabled smart farming application.

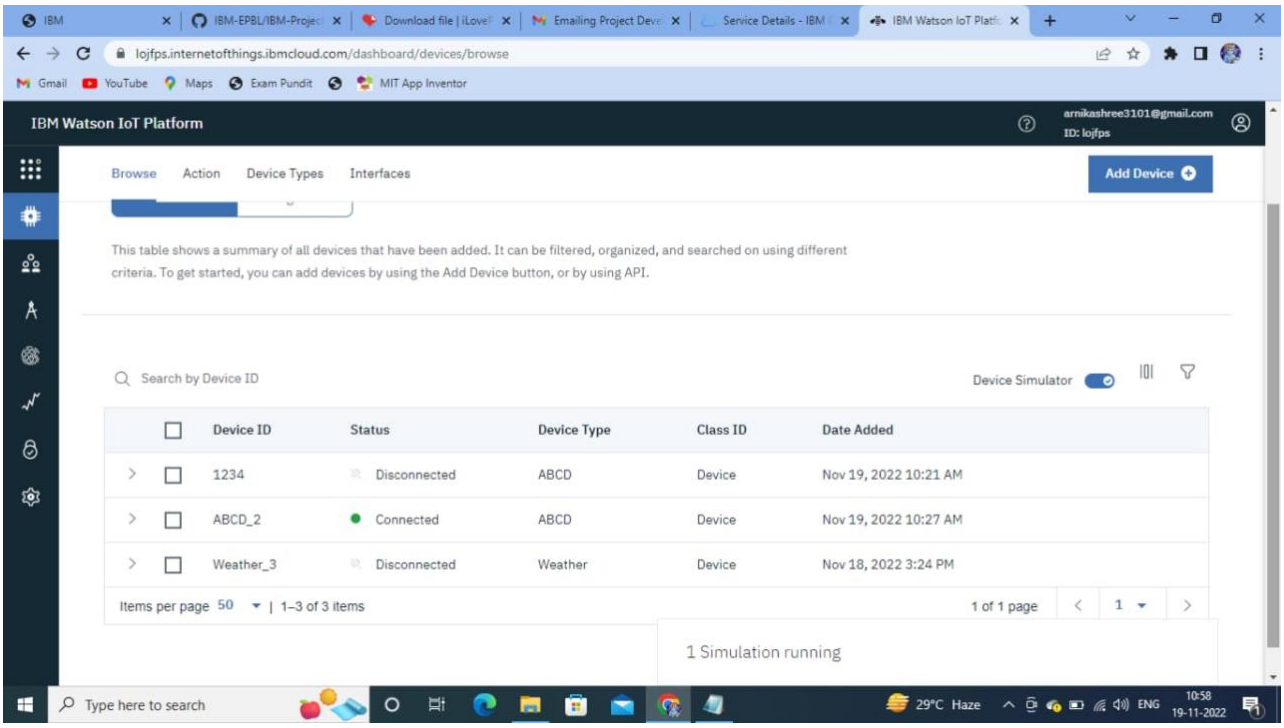
9. RESULTS

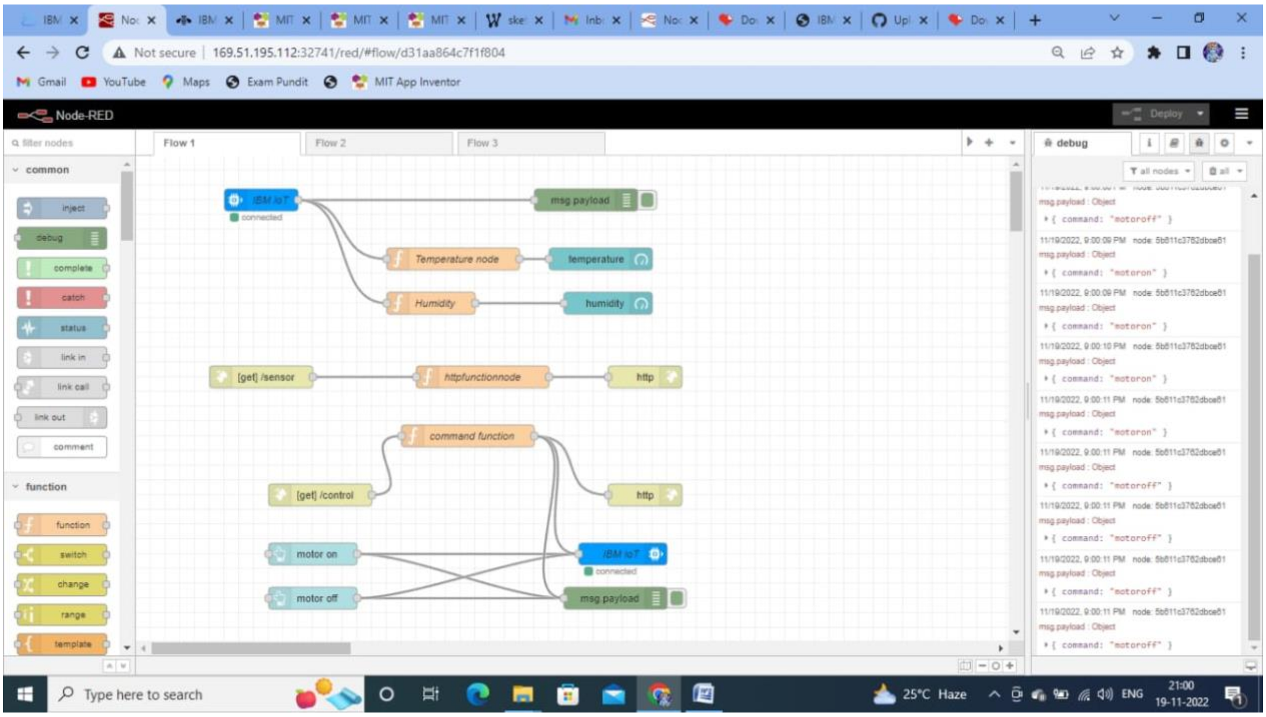
9.1 Performance Metrics

So finally when we run the python code it is going to connect the IBM Watson platform and connecting to the node-red after that is going to connect the mobile application.so we can seeoutput in the fourth window.

```
*Python 3.7.0 Shell*
File Edit Shell Debug Options Window Help
Published Temperature = 29 C Humidity = 0 % soil Moisture = 75 % to IBM
Watson
Published Temperature = 31 C Humidity = 89 % soil Moisture = 74 % to IBM
Watson
Published Temperature = 61 C Humidity = 50 % soil Moisture = 12 % to IBM
Watson
Published Temperature = 87 C Humidity = 98 % soil Moisture = 8 % to IBM
Watson
Published Temperature = 37 C Humidity = 16 % soil Moisture = 53 % to IBM
Watson
Published Temperature = 83 C Humidity = 54 % soil Moisture = 16 % to IBM
Watson
Published Temperature = 42 C Humidity = 50 % soil Moisture = 10 % to IBM
Watson
Published Temperature = 11 C Humidity = 31 % soil Moisture = 2 % to IBM
Watson
Published Temperature = 86 C Humidity = 65 % soil Moisture = 21 % to IBM
Watson
Published Temperature = 19 C Humidity = 14 % soil Moisture = 87 % to IBM
Watson
Published Temperature = 57 C Humidity = 3 % soil Moisture = 10 % to IBM
Watson
Published Temperature = 11 C Humidity = 50 % soil Moisture = 17 % to IBM
Watson
Published Temperature = 74 C Humidity = 13 % soil Moisture = 71 % to IBM
Watson
Published Temperature = 40 C Humidity = 91 % soil Moisture = 35 % to IBM
Watson
Published Temperature = 0 C Humidity = 3 % soil Moisture = 50 % to IBM W
atson
Published Temperature = 86 C Humidity = 68 % soil Moisture = 27 % to IBM
Watson
Published Temperature = 1 C Humidity = 58 % soil Moisture = 73 % to IBM
Watson
Published Temperature = 60 C Humidity = 99 % soil Moisture = 76 % to IBM
Watson
Published Temperature = 70 C Humidity = 34 % soil Moisture = 0 % to IBM
Watson
Published Temperature = 68 C Humidity = 70 % soil Moisture = 26 % to IBM
Watson
Published Temperature = 21 C Humidity = 29 % soil Moisture = 25 % to IBM
Watson
Published Temperature = 54 C Humidity = 40 % soil Moisture = 42 % to IBM
Watson
```


8:52
Screen1
Weather monitoring
Temperature
83
Humidity
5
Soil moisture
MOTOR ON
MOTOR OFF


IBM Watson IoT Platform
Browse
Action
Device Types
Interfaces
Add Device
This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.
Search by Device ID
Device Simulator
Device ID
Status
Device Type
Class ID
Date Added
1234
Disconnected
ABCD
Device
Nov 19, 2022 10:21 AM
ABCD_2
Connected
ABCD
Device
Nov 19, 2022 10:27 AM
Weather_3
Disconnected
Weather
Device
Nov 18, 2022 3:24 PM
Items per page 50 | 1–3 of 3 items
1 of 1 page
1 Simulation running


Node-RED
Flow 1
Flow 2
Flow 3
msg payload
Temperature node
temperature
Humidity
humidity
[get] /sensor
httpfunctionnode
http
command function
[get] /control
http
motor on
motor off
msg payload
debug

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

All the data like climatic conditions and changes in them, soil or crop conditions everythingcan 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 ofcrops 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 cannot fulfillthis requirement.

Any faults in the sensors can cause great loss in the agriculture, due to wrong records and theactions of automated processes.

IoT devices need much money to implement.

11. CONCLUSION: So finally we build A IoT Web Application for smart agricultural system using Watson IoT platform, Watson simulator, IBM cloud and Node-RED and MITapp Inventor

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.

13. APPENDIX

Source Code:

```
import timeimport sys
import ibmiotf.applicationimport ibmiotf.device import random


#Provide your IBM Watson Device Credentialsorganization = "49x4b9"
deviceType = "weather_monitoring"deviceId = "weather_today"
 authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"


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

temperature=random.randint(0,100)humidity=random.randint(0,100)
 soil= random.randint(0,100)

data = {'temperature' : temperature, 'humidity':humidity
 ,'soil':soil}
#print data
def myOnPublishCallback():
print ("Published Temperature = %s C" % temperature, "Humidity =
 %s %%" % humidity, "soil Moisture =
%s %%"% soil,"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()
```

GitHub:

Name	GitHub(User Name)
Team Leader(kaviya)	Kaviya1067
Team Member(Arnikashree)	Arnikashree31
Team Member(Arun)	Arun@123k
Team Member(Thirumurugan)	Thirumurugan1234
Team Member(Priyadharshini)	Priyashini

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-41508-1660642542

Project Demonstration Video Link:

https://drive.google.com/file/d/1w3XN9ncWyuuUjYM6fstZXlTvqQfYBgbX/view?usp=sharing