

PROJECT REPORT

Team ID	PNT2022TMID14367
Project Name	SMARTFARMER – IOT ENABLED SMART FARMING APPLICATION

CHAPTER 1

INTRODUCTION

1.1 Project Overview

People who use the internet of things can live and work more intelligently and have total control over their life. IoT is crucial to business in addition to providing smart home automation devices. With the help of IoT, organizations can see in real time how their systems actually function, gaining insights into anything from equipment performance to supply chain and logistics activities. This paper presents an Internet of Things (IoT) based Smart Farmer - IoT Enabled Smart Farming Application. The system is implemented using an ultrasonic sensor which is connected to Arduino UNO as to monitor the water level. In this system, the depth level will be sent via Arduino Ethernet Shield with an Internet connection to the IBM Cloud.

1.2 Purpose

Since the dawn of human civilization, agriculture has been considered to be the most significant activity. Traditional irrigation techniques, such as flood irrigation and overhead sprinkler irrigation, are not very effective. They waste a significant amount of water, and the excessive moisture in the soil can also encourage the growth of diseases like fungus. Since water is a valuable resource and indirectly supports the survival of the farm, an automated irrigation system is crucial. Around 85% of all water resources are used exclusively for irrigation purposes worldwide. This need is anticipated to rise in the coming years due to population growth. We must implement new strategies that reduce the amount of water needed for irrigation in order to meet this demand. In an automation system, sensors are used to monitor the crop's access to water, and controlled irrigation is used to water the crop as needed.

IoT is a new platform that is extremely beneficial to people all over the world. IoT is at the heart of such revolutionary growth engines. IoT is possible because of adequate power supply and internet connectivity. The term "Internet of Things" is commonly used to describe a framework in which sensors are connected to objects and allow these objects to share their "digital voice" with the outside world via an internet connection. IoT has recently evolved into a collection of purpose-built networks.

Based on a wireless sensor network, this system created an automated irrigation system for farmers. The soil's moisture content, humidity, and temperature are all continuously monitored by this technology. The constant maintenance of the threshold soil moisture values was done using an algorithm. Depending on the soil's moisture content, the irrigation system either starts or stops. This method suggests a low cost data collecting system based on moisture sensors that is necessary for an automated watering system. Sensors rely on a change in impedance between two electrodes that are ke

CHAPTER -2

LITERATURE SURVEY

2.1 Existing Problem

S.NO	TITLE	AUTHOR AND YEAR OF PUBLICATIONS	METHODOLOGY USED
1.	Mobile Integrated Smart Irrigation Management and Monitoring System Using IOT	S. Vaishali et.al, 08 February 2018	In order to control and monitor the irrigation process, smart and automated irrigation systems developed, Implemented and tested. There is a need for automated irrigation system because it is simple and easy to install. This system uses values ON and OFF to control water motor. Python programming language is been used for automation purpose.
2.	IoT Based Smart Irrigation Monitoring And Controlling System	Shweta B. Saraf et.al, 15 January 2018	In this paper proposed system is based on IoT that uses real time input data. Smart farm irrigation system uses android phone for remote monitoring and controlling of drips through wireless sensor network. Zigbee is used for communication between sensor nodes and base station. Real time sensed data handling and demonstration on the server is accomplished using web based java graphical user interface.
3	Smart waste collection monitoring and alert system via IoT	<u>Zainal Hisham Che Soh</u> et.al, 24 June 2019	The system is implemented using an ultrasonic sensor which is connected to Arduino UNO as to monitor waste bin garbage level. In this system, waste bin depth level will be sent via Arduino Ethernet Shield with an Internet connection to the Ubidots IoT Cloud. The Ubidots store the collected waste bin level data into IoT database and display the waste bin depth level on online dashboard for real-time visualization.

2.2 References

1. Mobile Integrated Smart Irrigation Management and Monitoring System Using IOT
Date of Conference: 06-08
April 2017 Publisher: IEEE
Date Added to IEEE Xplore: 08
February 2018 DOI:
10.1109/ICCSP.2017.8286792
2. IoT Based Smart Irrigation Monitoring And Controlling System
Date Added to IEEE Xplore: 15
January 2018 ISBN Information:
Electronic ISBN: 978-1-5090...Date
of Conference: 19-20 May 2017
INSPEC Accession Number: 17504411
3. Smart Waste Collection Monitoring and Alert System via IoT
Date Added to IEEE Xplore:
24 June 2019 DOI:
10.1109/ISCAIE.2019.874376
Print on Demand(PoD) ISBN: 978-1-5386-854

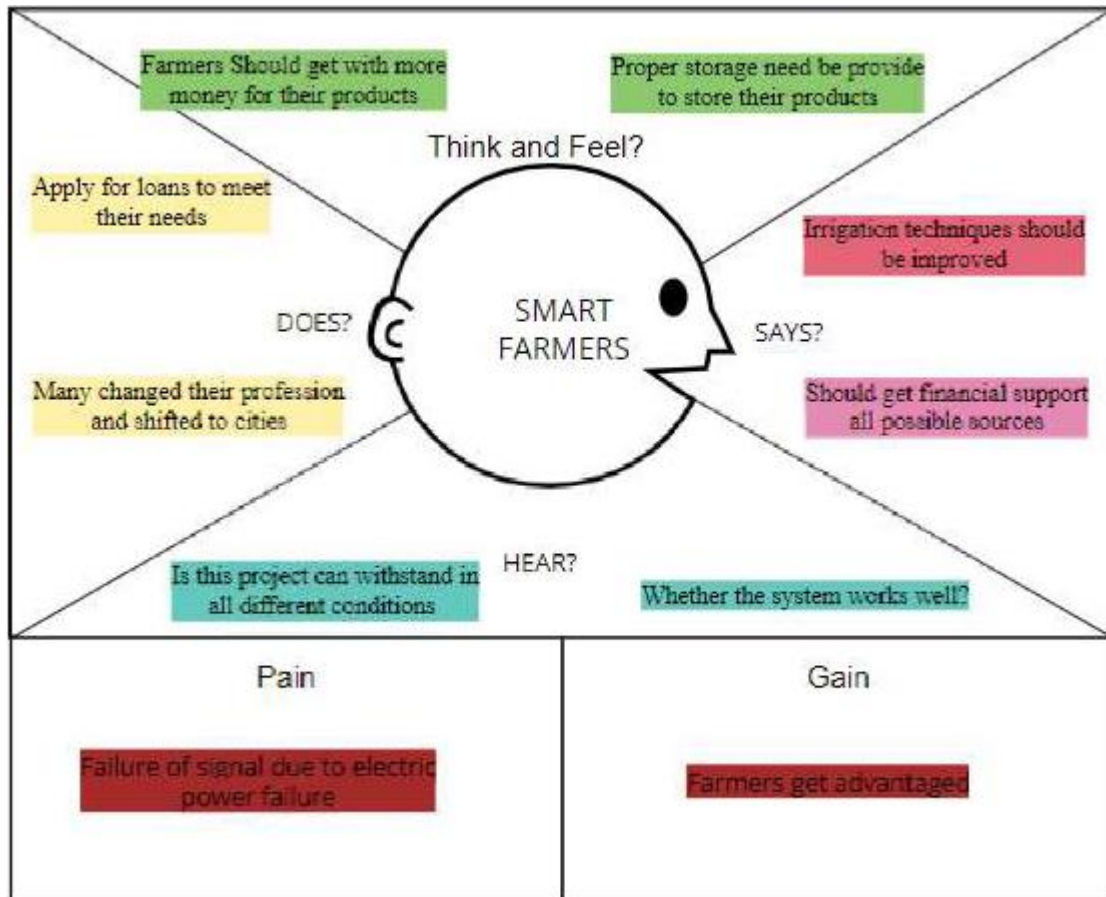
2.3 Problem Statement Definition

Agriculture is the Backbone of Our Country. Traditional methods that are used for irrigation. They results in a lot of wastage of water. About 85% of total available water resources across the world are solely used for the irrigation purpose. In upcoming years this demand is likely to increase because of increasing population. To meet this demand we must adopt new techniques which will conserve need of water for irrigation process. In this paper proposed system is based on IoT that uses real time input data. This Water Level Monitoring Irrigation system the excess availability of water in crop is monitored through sensors and reduces the water consumption. This idea is also to focus on parameters such as temperature and soil moisture. The main objective of this project is to control reduce the water supply, save the crops and monitor the plants. The system is implemented using an ultrasonic sensor which is connected to Arduino UNO as to monitor Farm Field level. In this system, Farm Field depth level will be sent via Arduino Ethernet Shield with an Internet connection to the IBM IoT Cloud. The IBM Cloud store the collected Farm field level data into IoT database and display the Farm Field depth level on online dashboard for real-time visualization. The IBM Event manager invoke a notification alert to the Owner of the farmer mobile phone via a SMS when the farm field is nearly filled and It automatically Switch Off the Water Motor.

CHAPTER 3


IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas






3.2 Ideation & Brainstorming

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




Brainstorm & idea prioritization


Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

 10 minutes to prepare
 1 hour to collaborate
 2-8 people recommended

[Share template feedback](#)

 **Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

 10 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.


B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →


1 Define your problem statement

During the irrigation process there is a possibility of wastage in water resources. So, we implemented an IOT set up to control and monitor the water supply for crops.

 5 minutes







PROBLEM


Can we identify the water level in the crop field and it can be controlled/monitored through mobile?



Key rules of brainstorming

To run a smooth and productive session

 Stay in topic.	 Encourage wild ideas.
 Defer judgment.	 Listen to others.
 Go for volume.	 If possible, be visual.



Need some inspiration?
See a finished version of this template to kickstart your work.

[Open example](#) →

Step-2: Brainstorm, Idea Listing and Grouping

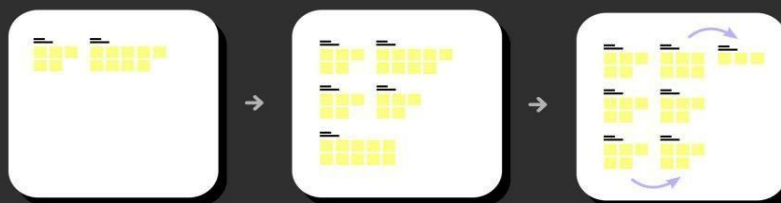
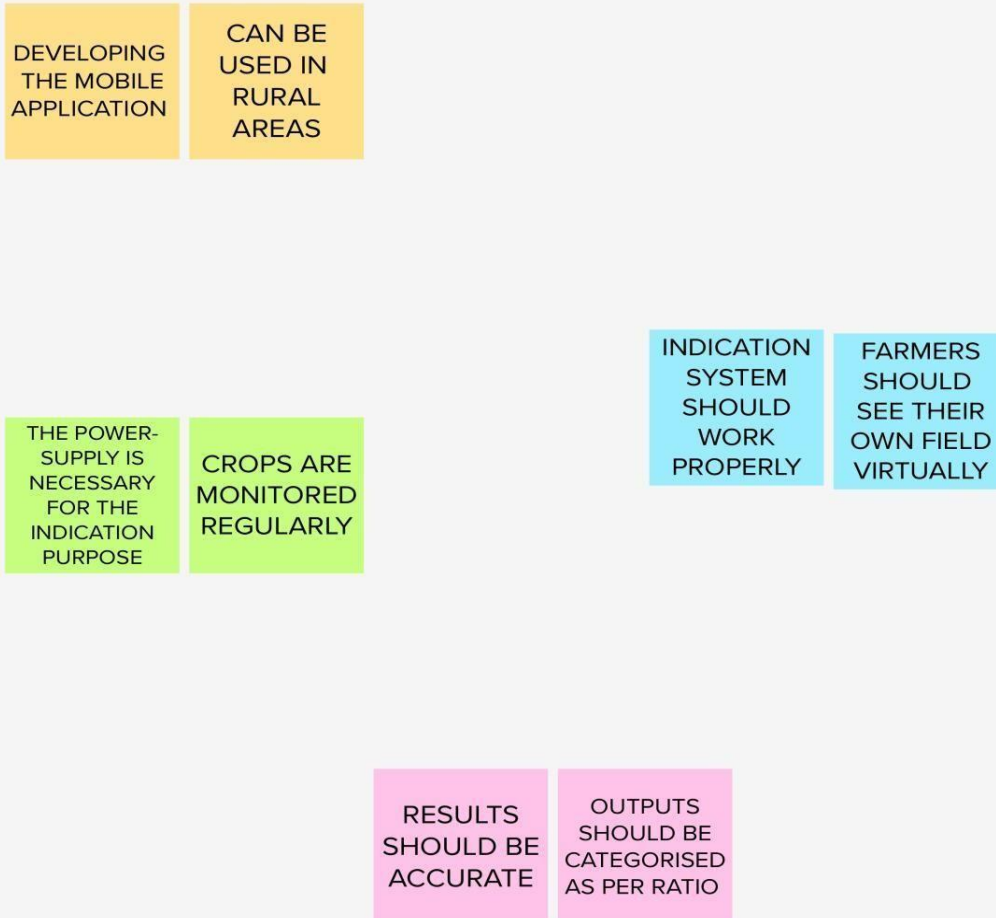
The image displays a sequence of six cards from a presentation, illustrating the 'Brainstorm & Idea Prioritization' process. Each card features a blue circular icon with a white symbol and a title. The cards are arranged in a row, with the first card on the left and the last on the right. The cards are numbered 1 through 6, indicating a sequential process. The first card is titled 'Brainstorm & Idea Prioritization' and includes a sub-header 'Use this template in your team brainstorming sessions to generate and prioritize ideas for your project. The template is designed to be used in a workshop setting.' The subsequent cards show various stages of the process, including brainstorming, idea generation, and prioritization. The last card shows a final prioritized list of ideas.

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes



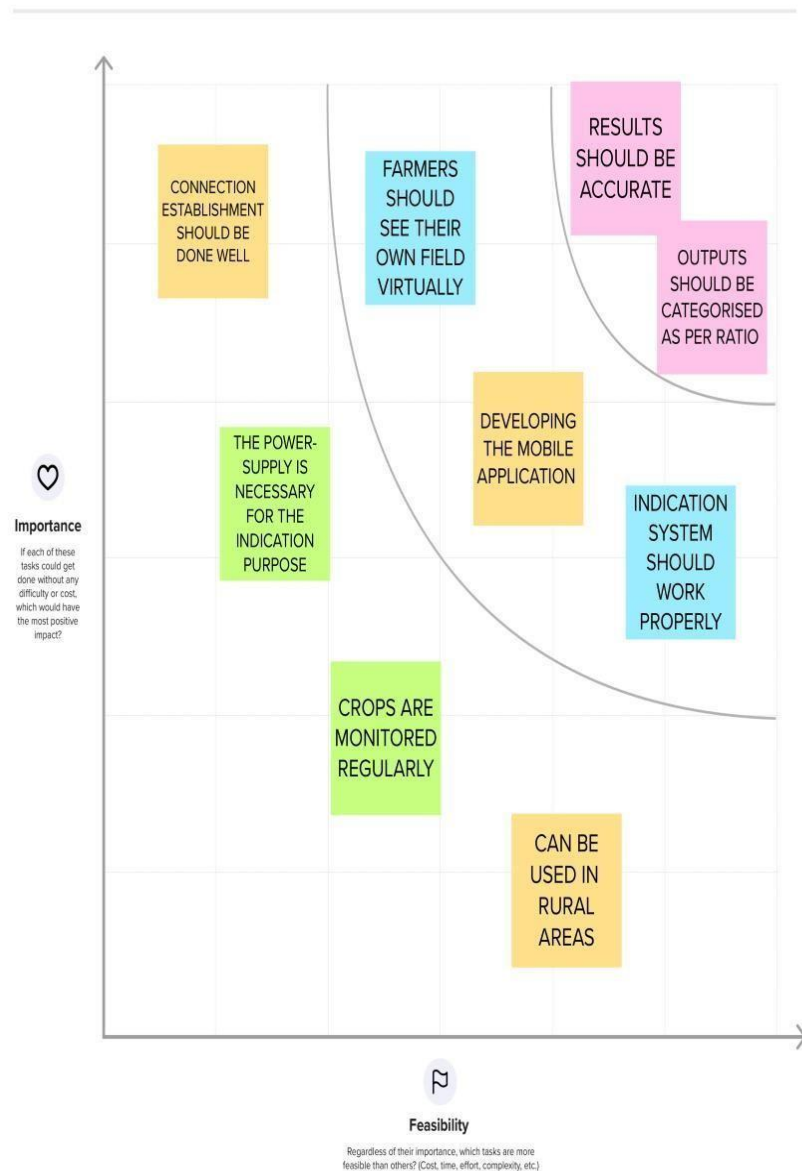
Step-3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

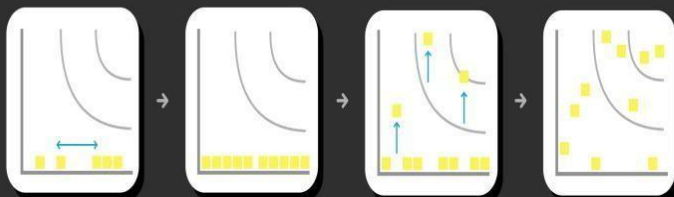
Quick add-ons

- A Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- B Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

- Strategy blueprint**
Define the components of a new idea or strategy.
[Open the template →](#)
- Customer experience journey map**
Understand customer needs, motivations, and obstacles for an experience.
[Open the template →](#)
- Strengths, weaknesses, opportunities & threats**
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
[Open the template →](#)

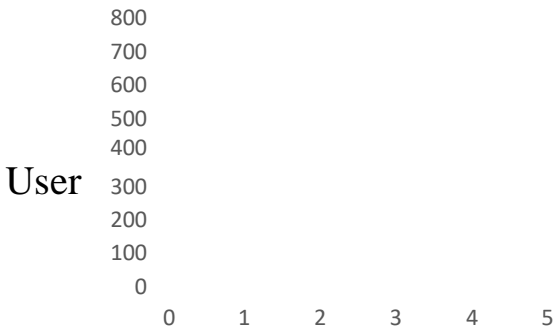
[Share template feedback](#)



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<ul style="list-style-type: none">• CLIMATIC CONDITIONS : The important drawback and problem faced by the farmers are the frequent changes in the climate.• PRECISION FARMING : The farm is covered by a vast area. It is a difficult task for the farmers to take care the entire farm and note the changes in each and every particular areas.• MONITORING AND WATER REQUIRMENTS : It's a difficult task for the farmer to monitor the cattle in all the areas. Water requirements are different in each places of the farms. If a particular place is watered completely, in case of excess water,the crops are affected and wasted.Hence it must be monitored with excess care.
2.	Idea / Solution description	<ul style="list-style-type: none">• With the help of IOT sensors,the real time climatic

		<p>conditions and monitoring can be done precisely .</p> <ul style="list-style-type: none"> • The Data collected by sensors, In terms of humidity, temperature, moisture, and dew detections help in determining the weather pattern in Farms. So cultivation is done for suitable crops.
3.	Novelty / Uniqueness	<ul style="list-style-type: none"> • IoT sensor nodes collect information from the farming environment, such as soil moisture, air humidity, temperature, nutrient ingredients of soil, pest images, and water quality, then transmit collected data to IoT backhaul devices. • It helps the farmer to operate the motor from anywhere.
4.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none"> • Reduces the wages for labors who work in the agricultural field. • It saves a lot of time. • IoT can help the each work to be completed precisely. • Easily identify maintenance needs, build better products. • The usage of water will be limited and also used only when needed.It reduces the wastage of excess water. • IoT can also help e-commerce businesses thrive and increase sales. • It make a wealthy society
5.	Business Model (Revenue Model)	Revenue (No. of Users vs Months)

		 <table><tr><th>X-axis</th><th>User</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>160</td></tr><tr><td>2</td><td>320</td></tr><tr><td>3</td><td>480</td></tr><tr><td>4</td><td>640</td></tr><tr><td>5</td><td>800</td></tr></table>	X-axis	User	0	0	1	160	2	320	3	480	4	640	5	800
X-axis	User															
0	0															
1	160															
2	320															
3	480															
4	640															
5	800															
6.	Scalability of the Solution	Scalability in smart farming refers to the adaptability of a system to increase the capacity, for example, the number of technology devices such as sensors and actuators, while enabling timely analysis.														

3.4 Problem Solution Fit

Problem-Solution Fit canvas

Team ID:PNT2022TMID14367

Define CS, Atmo CL	1. CUSTOMER SEGMENT(S) CS <ul style="list-style-type: none"> Farmers, Landowners, Gardeners Farmers who have farm field to yield crops who is like to save 80% of time. 	6. CUSTOMER CONSTRAINTS CL <small>EG. BUDGET, DEVICES</small> <ul style="list-style-type: none"> People those who are uneducated will find it difficult to operate smart phones Reduce over usage of resources 	5. AVAILABLE SOLUTIONS PLUSES & MINUSES <p>Farmers can monitor their land using smart phone which is integrated to IoT</p>
Focus on PR, tap into BL, understand RC	2. PROBLEMS / PAINS PR <small>+ ITS FREQUENCY</small> <ul style="list-style-type: none"> Irrigation Problems Meet rising demand for food of higher quality Cope with Climate change 	9. PROBLEM ROOT / CAUSE RC <ul style="list-style-type: none"> Irrigating the crops in the correct time Wasting a lot of time in the farm fields 	7. BEHAVIOR ITS INTENSITY <ul style="list-style-type: none"> Monitor the sensors regularly Collect the data from the field and analyze it Ensure the stable Internet Connectivity
Identify strong TR & EM	3. TRIGGERS TO ACT TR <p>Farmers want to save their time and to make their crops healthy, control them from anywhere and to reduce the wages of labors</p> <hr/> 4. EMOTIONS EM <small>BEFORE / AFTER</small> <p>Difficulty in predicting the weather and to monitor the crops from anywhere</p>	10. YOUR SOLUTION SL <ul style="list-style-type: none"> Monitoring the environmental condition towards the growth of high quality crops. 	8. CHANNELS of BEHAVIOR <div> ONLINE <p>Through online the customer can lively track the farm fields through mobile phones</p> </div> <hr/> <div> OFFLINE <p>In offline mode the customer can check the sensors regularly</p> </div>

CHAPTER 4

REQUIREMENT ANALYSIS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Reading Temperature	Using RTD (Resistance Temperature Detectors) or Thermistor
FR-2	Identify Humidity and Moisture Level	Hygrometers, are used to measure humidity levels in the atmosphere
FR-3	Transfer the Values	The Temperature and Humidity Values Passed through Sensor Networks
FR-4	IoT Enabled Smart Device	It helps to Monitor the sensor values and useful to operate Motor Pumps through Mobile App.
FR-5	Application	Mobile App Developed through MIT App Inventory For Accessing the Remote Operations

Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Given that it is IoT Enabled Device and Control through Mobile, even those who are with little experience can easily understand it.
NFR-2	Security	It provides assurance that all data transfer inside the system, so it will be secure from virus attacks and unapproved entry.
NFR-3	Reliability	The degree to which the result of a measurement, calculation, or specification can be depended on to be accurate.
NFR-4	Performance	It is easily adoptable, By adding new components with increased functionalities, the existing system can be simply improved
NFR-5	Availability	It protecting the functionality of support systems and ensuring data is fully available at the point in time (or period requirements) when it is needed by our farmers.
NFR-6	Scalability	It is ability of a network to cope with increasing workloads in a cost-effective and sustainable way, by expanding the network's bandwidth capacity and supporting its physical expansion like expanding the Monitoring Vision of Agriculture Field.

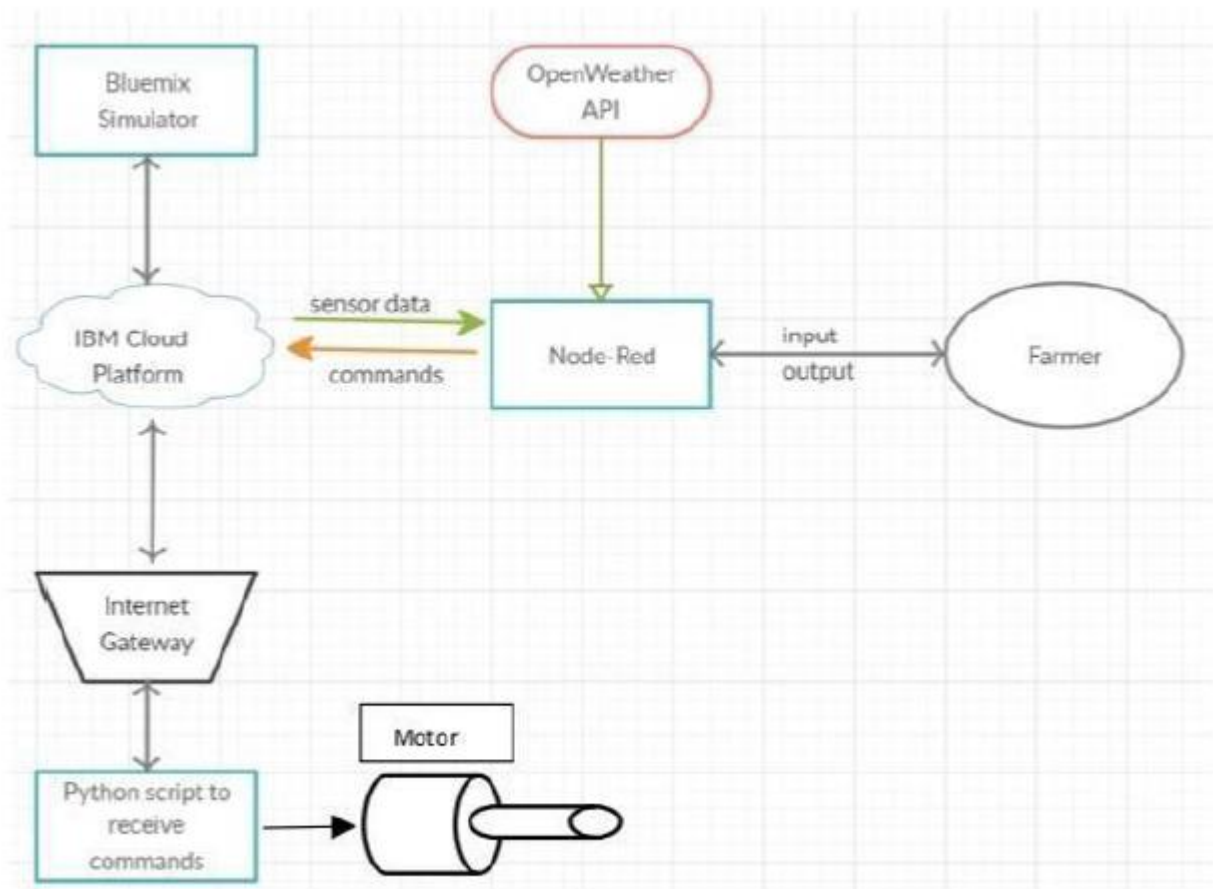
CHAPTER 5

PROJECT DESIGN

5.1 Data Flow Diagram

Data Flow Diagram :

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2 Solution and Technical Architecture

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

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

Solution Architecture Diagram:

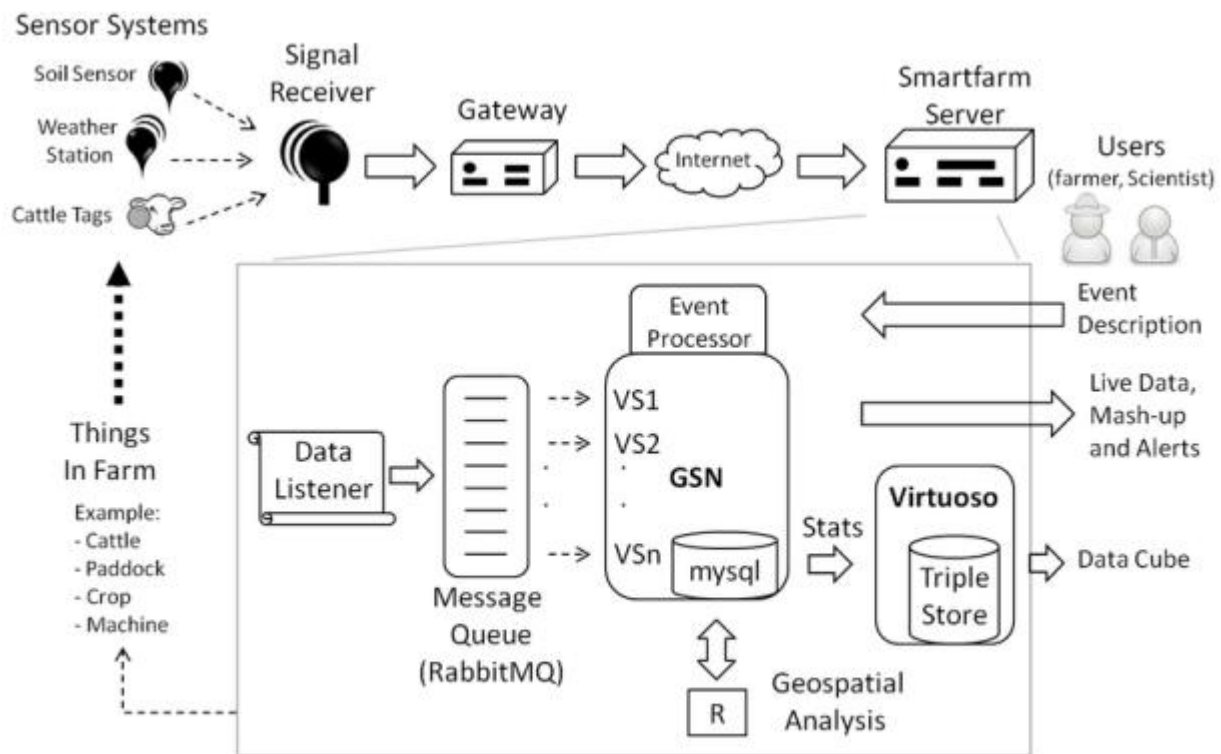


Figure 1: Architecture and data flow of the smart farming application

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Python IDE
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	GSM module
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Node red service
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	IBM Watson IoT Platform
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	NPK Sensors

5.3 User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
Customer (Mobile user)	Configure the Application and Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High
	Registration Method 2	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High
	Registration Method 3	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low
	Registration Method 4	USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Mail Login	Medium
	Login	USN-5	As a user, I can log into the application by entering email & password	I Logged in, and Check out my Dashboard	High
	Dashboard	USN-6	As a user, I can track, analyze and display data.	Authenticated Users are allowed to access	High
Customer (Web user)	As per the Mobile Application View	USN-7	User Friendly Navigation to Access	Easily Navigated through Pages	High
Customer Care Executive	Helpline to access and report the data error, if in case	USN-8	Provided Stability Support to Recover Issues	If the Data is not Shown or Sensors Not Sensing the Value	Low

CHAPTER 6

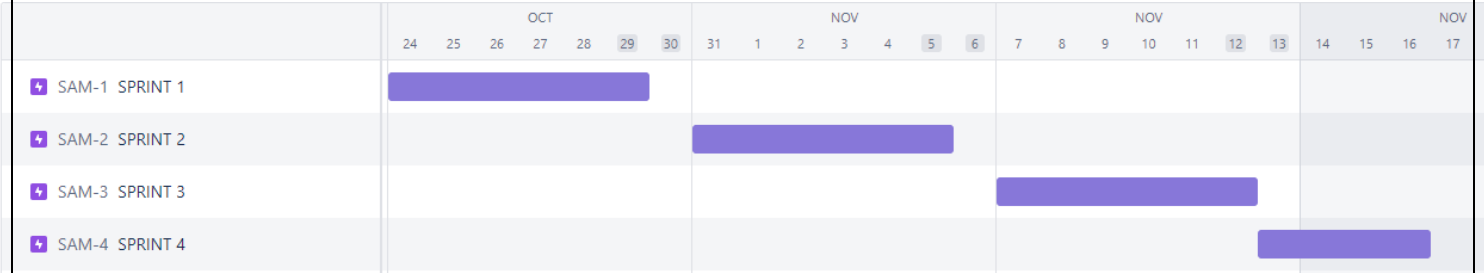
PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature review on the chosen project and information gathering using references from IEEE Papers.	25 September 2022
Prepare Empathy Map	Get an Empathy Map Canvas ready to record the user's gains and pains and also prepare list of problem statements	26 September 2022
Ideation	Create a list of them by arranging the brainstorming session, then rank the top three concepts according to their viability and significance.	27 September 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, workability of idea, business pattern, social clash and so on.	5 October 2022
Problem Solution Fit	Prepare problem - solution fit document.	8 September 2022

Solution Architecture	Prepare solution architecture document.	11 September 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application	14 October 2022
Functional Requirement	Create the project's functional requirements.	17 October 2022
Data Flow Diagrams	Create the data flow diagrams, then submit them for evaluation.	17 October 2022
Technology Architecture	Prepare the technology By using the architecture diagram.	18 October 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	22 October 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	Towards Progress

6.2 Reports from JIRA



CHAPTER 7

CODING & SOLUTIONING

7.1 Features - Development of Sprint-1

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

#IBM
organization = "3nc6qc"
deviceType = "node"
deviceId = "008"
authMethod = "use-token-auth"
authToken = "6383637992"

#Gpio
def mycommandCallback(cmd):
    print("Command Received: %s" %cmd.data['command'])
    status = cmd.data['command']
    if status=="lighton":
        print("LED is ON")
    elif status=="lightoff":
        print("LED is OFF")
    try:
        else:
            print("please send proper command")
    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()
    #CONNECCT
    deviceCli.connect()
    while True:
        temp=random.randint(0,100)
        hum=random.randint(0,100)

        data={'temp':temp,'hum':hum}
        def myOnPublishCallback():
            print("Published Temperature = %s C"%temp,"Humidity = %s %" %hum, "to IBM Watson")
            success = deviceCli.publishEvent("IoTSensor","json",data,qos=0,
            on_publish=myOnPublishCallback)
            if not success:
                print("Not connected to IoT")
                time.sleep(10)
            deviceCli.commandCallback = mycommandCallback
            #Disconnect
            deviceCli.disconnect()
```



```
ibmconnectTest) Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\india\Desktop\IBM-EBPL\pgms\ibmconnectTest.py =====
2022-11-08 22:25:37,567 ibmiotf.device.Client INFO Connected successfully: d:r5gral:Dora:30
Published Temperature = 19 C Humidity = 77 % to IBM Watson
Published Temperature = 69 C Humidity = 0 % to IBM Watson
Published Temperature = 18 C Humidity = 18 % to IBM Watson
Published Temperature = 26 C Humidity = 3 % to IBM Watson
Published Temperature = 66 C Humidity = 61 % to IBM Watson
Published Temperature = 81 C Humidity = 41 % to IBM Watson
Published Temperature = 38 C Humidity = 88 % to IBM Watson
Published Temperature = 53 C Humidity = 18 % to IBM Watson
Published Temperature = 26 C Humidity = 37 % to IBM Watson
Published Temperature = 96 C Humidity = 26 % to IBM Watson
#vishnu
organiza
deviceTy
deviceId
authMeth
authToke
#Gpio
def myco
prin
stat
if s
elif
else
try:
```

Browse Action Device Types Interfaces Add Device +

<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	
<input checked="" type="checkbox"/>	008	Disconnected	node	Device	10 Nov 2022 10:04		→ ...
Identity Device Information Recent Events State Logs X							
The recent events listed show the live stream of data that is coming and going from this device.							
Event	Value	Format	Last Received				
event2	{"Alert!! Alert!! Distance":65.03}	json	a few seconds ago				
event2	{"Alert!! Alert!! Distance":64.96}	json	a few seconds ago				
event2	{"Alert!! Alert!! Distance":64.96}	json	a few seconds ago				
event2	{"Alert!! Alert!! Distance":64.96}	json	a few seconds ago				
event2	{"Alert!! Alert!! Distance":64.96}	json	a few seconds ago				

Development of Sprint-2

Connecting IoT Simulator to IBM Watson IoT Platform

Open link provided in above section

Give the credentials of your device in IBM Watson IoT Platform

Click on connect

My credentials given to simulator are: OrgID: x93v8c

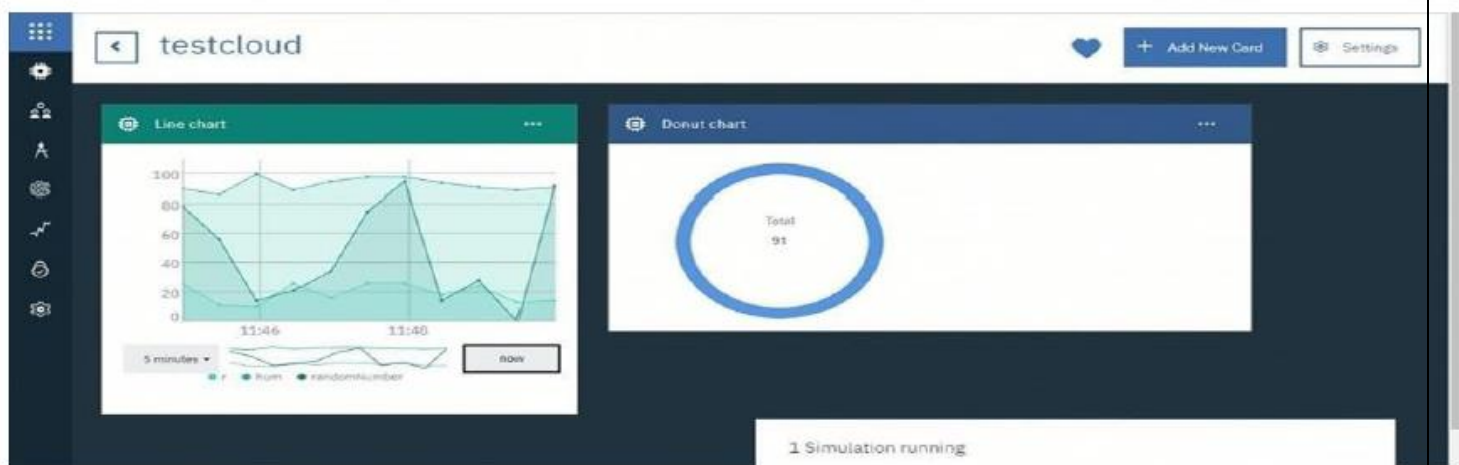
api: a-157uf3f5rg4qxpd3

Device type: NodeMcu

token: 6ogMaaQHNWFEGOD8R?

Device ID : 12345

Device Token : 12345678



You can see the received data in graphs by creating cards in Boards tab

- You will receive the simulator data in cloud
- You can see the received data in Recent Events under your device
- Data received in this format(json)

```
{  
  "d": {  
    "name": "abcd",  
    "temperature": 17,  
    "humidity": 76,  
    "Moisture ": 25  
  }  
}
```

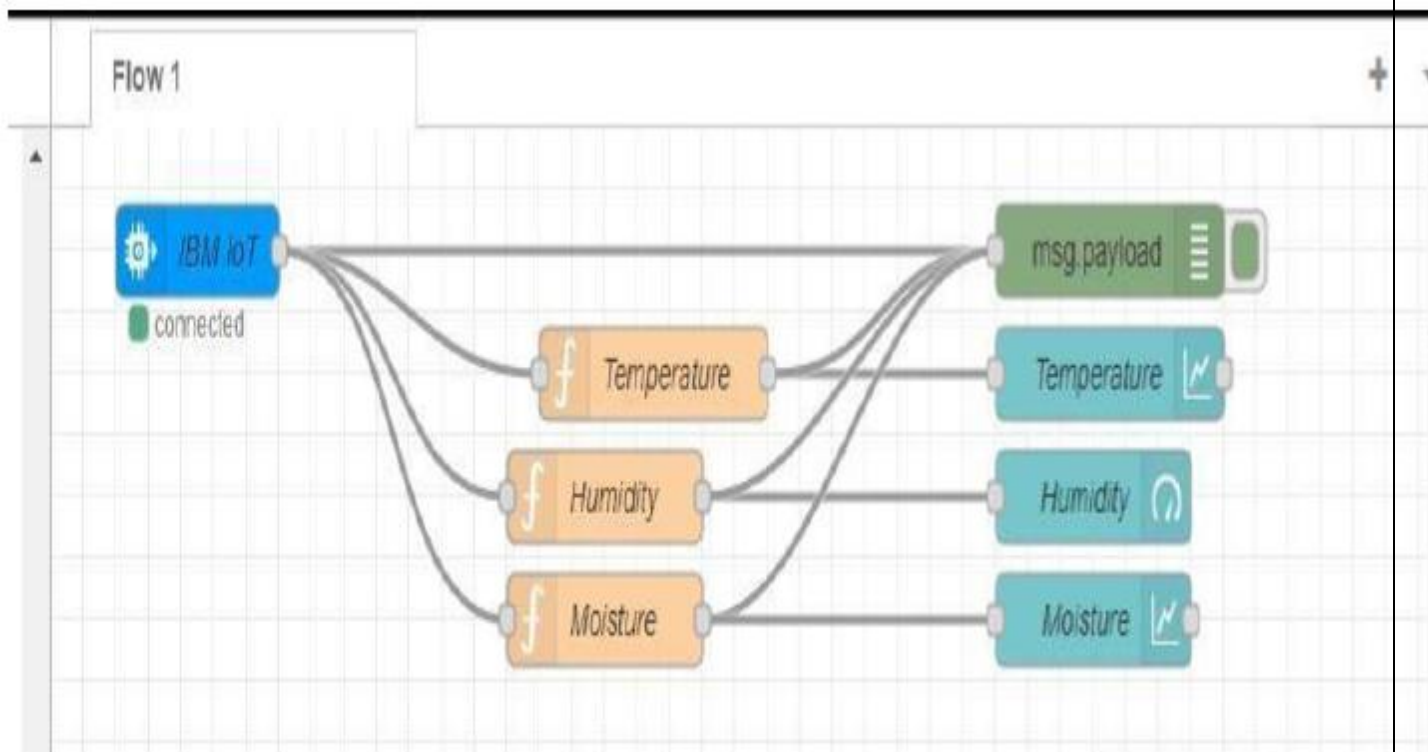

Display the data using debug node for verification

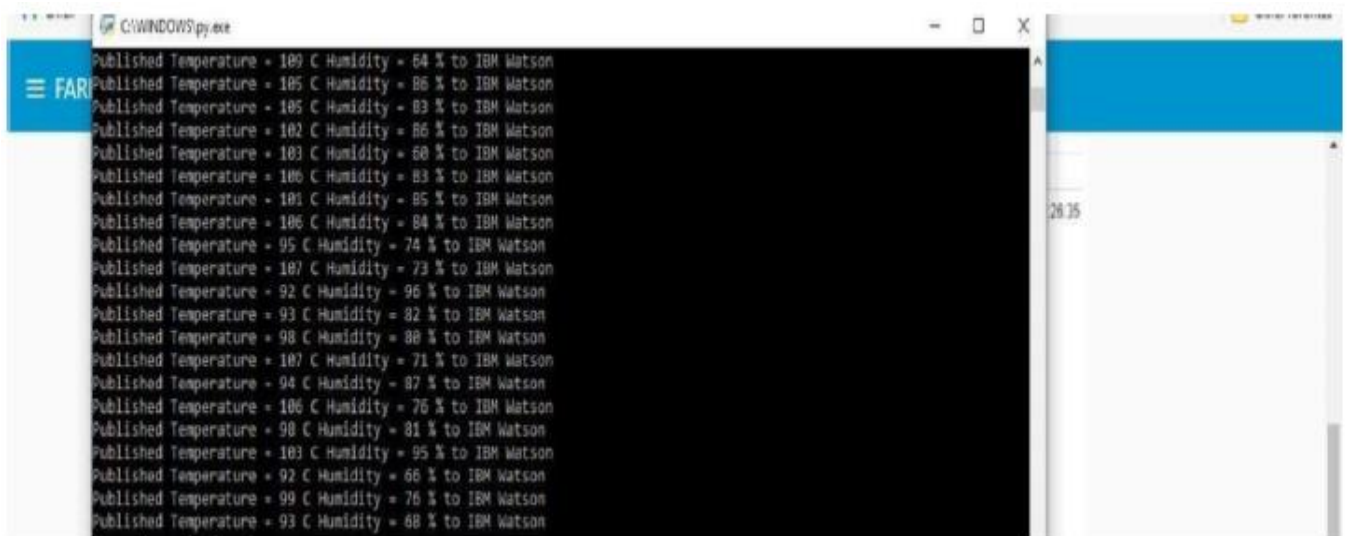
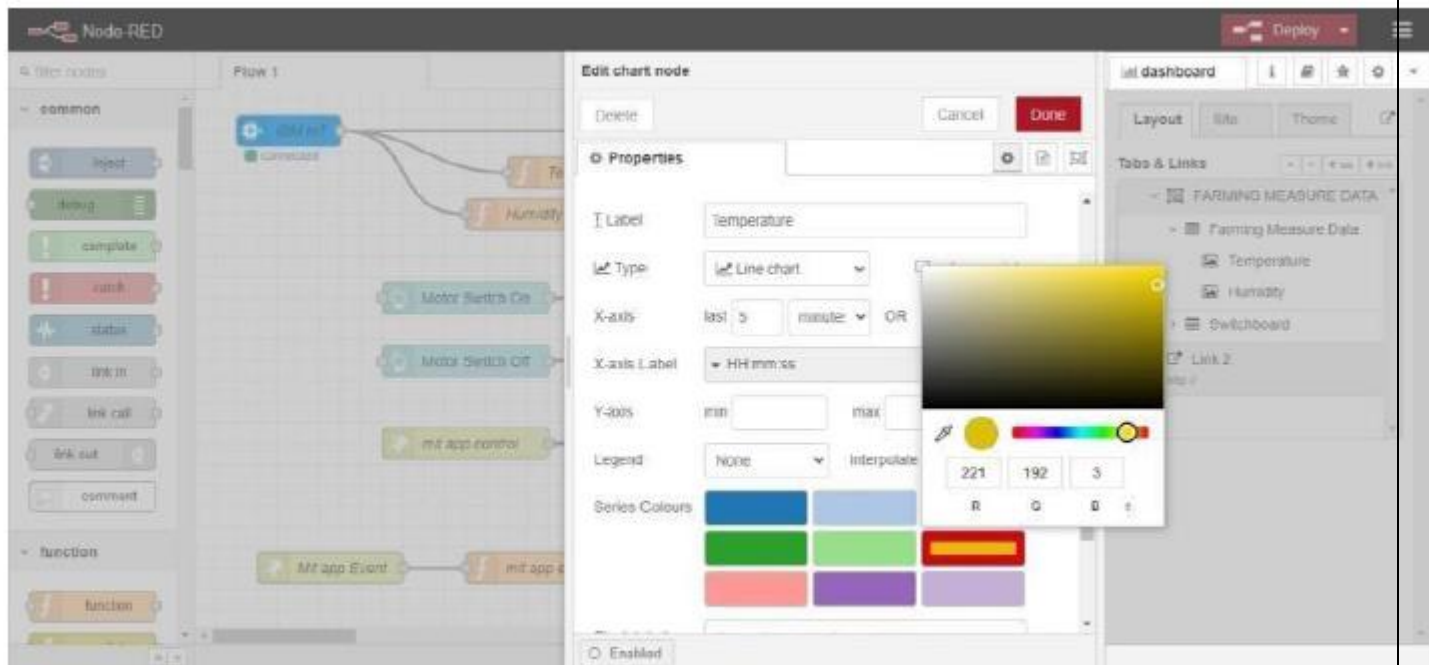
Connect function node and write the Java script code to get each reading separately.

The Java script code for the function node is:

```
msg.payload=msg.payload.d.temperature return msg;
```

Finally connect Gauge nodes from dashboard to see the data in UI





Configuration of Node-Red to collect data from OpenWeather

The Node-Red also receive data from the OpenWeather API by HTTP GET request. An inject trigger is added to perform HTTP request for every certain interval. HTTP request node is configured with URL we saved before in section 4.4 The data we receive from OpenWeather after request is in below JSON

```
format:{"coord":{"lon":79.85,"lat":14.13},"weather":[{"id":803,"main":"Clouds",
"description":"brokenclouds","icon":"04n"}],"base":"stations","main":{"temp":307
59,"feels_like":305.5,"temp_min":307.59,"temp_max":307.59,"pressure":1002,"h
umidity":35,"sea_level":1002,"grnd_level":1000},"wind":{"speed":6.23,"deg":170
```

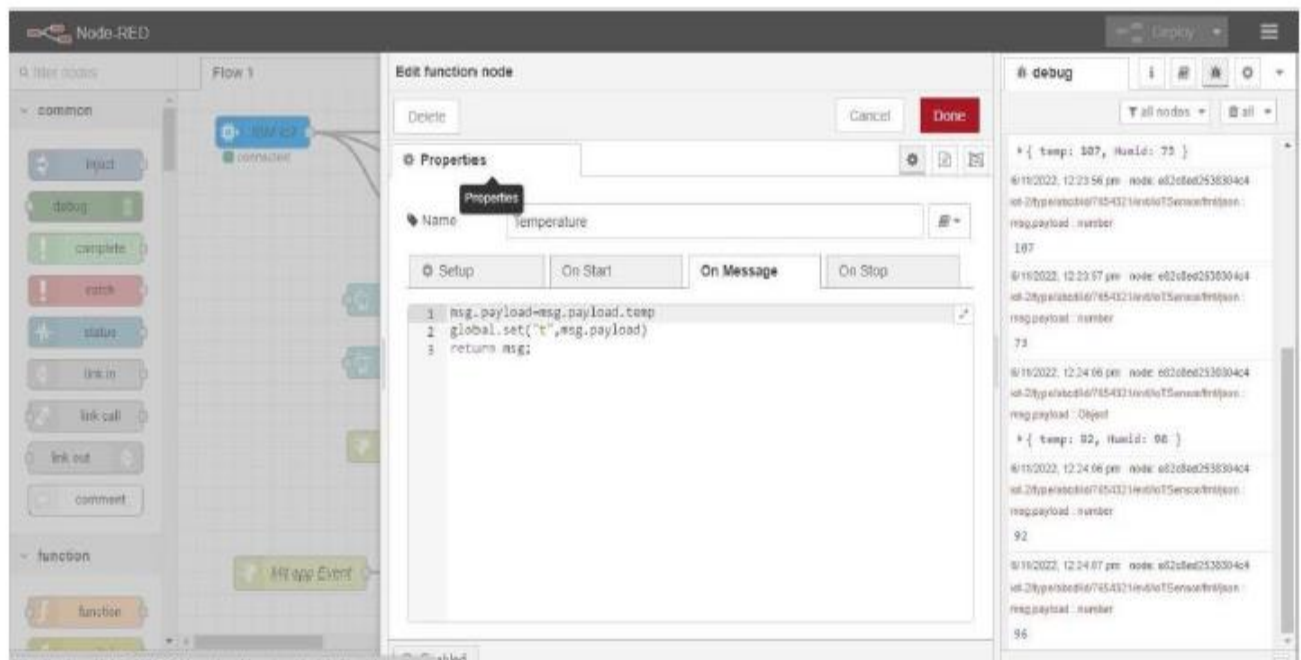
```
}  
,"clouds":{"all":68},"dt":1589991979,"sys":{"country":"IN","sunrise":1589933553,  
"sunset":1589979720},"timezone":19800,"id":1270791,"name":"Gūdūr","cod":20  
0}
```

In order to parse the JSON string we use Java script functions and get each parameters

```
Var temperature = msg.payload.main.temp; temperature = temperature-273.15;  
return {payload : temperature.toFixed(2)};
```

In the above Java script code we take temperature parameter into a new variable and convert it from kelvin to Celsius

Then we add Gauge and text nodes to represent data visually in UI



Development of Sprint-3

```
import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device
#Provide your IBM Watson Device Credentials
organization = "0lsrz8" # repalce it with organization ID
deviceType = "SMART FARMER #replace it with device type
deviceId = "Device_1" #repalce with device id
authMethod = "token"
authToken = "SMARTFarmer@123"#repalce with token
def myCommandCallback(cmd): # function for Callback
print("Command received: %s" % cmd.data)
if cmd.data['command']=='motoron':
print("Turn Motor ON")
elif cmd.data['command']=='motoroff':
print("Turn Motor OFF")
elif cmd.data['command']=='lighton':
print("Turn Light ON")
elif cmd.data['command']=='lightoff':
print("Turn Light OFF")
if cmd.data['command']=='ACTIVATE IRRIGATION':
print("TurnON")
elif cmd.data['command']=='DEACTIVATE IRRIGATION':
print("TurnOFF")

        elif cmd.data['command']=='HIGH TEMPERATURE':
print("TurnON")
elif cmd.data['command']=='LOW TEMPERATURE':
print("TurnOFF")
if cmd.data['command']=='BAD WEATHER':
print("TurnON")
elif cmd.data['command']=='GOOD WEATHER':
print("TurnOFF")
```

```

elif cmd.data['command']=='HUMIDITY HIGH':
    print("TurnON")
elif cmd.data['command']=='HUMIDITY LOW':
    print("TurnOFF")
if cmd.command == "setInterval":
    if 'interval' not in cmd.data:
        print("Error - command is missing required
information: 'interval'")
    else:
        interval = cmd.data['interval']
elif cmd.command == "print":
    if 'message' not in cmd.data:
        print("Error - command is missing required
information: 'message'")
    else:
        output=cmd.data['message']
        print(output)
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:
    deviceCli.commandCallback = myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()

```


python code.py - C:/Users/Sriiram V G/AppData/Local/Programs/Python/Python39/python code.py (3.9.8)

File Edit Format Run Options Window Help

```
import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device
#Provide your IBM Watson Device Credentials
organization = "01erz8" # replace it with organization ID
deviceType = "SMART_FARMER" #replace it with device type
deviceId = "Device_1" #replace with device id
authMethod = "token"
authToken = "SMARTFarmer8123" #replace with token
def myCommandCallback(cmd): # function for Callback
    print("Command received: %s" % cmd.data)
    if cmd.data['command'] == 'motoron':
        print("Turn Motor ON")

    elif cmd.data['command'] == 'motoroff':
        print("Turn Motor OFF")
    elif cmd.data['command'] == 'lighton':
        print("Turn Light ON")
    elif cmd.data['command'] == 'lightoff':
        print("Turn Light OFF")
    if cmd.data['command'] == 'ACTIVATE IRRIGATION':
        print("TurnON")

    elif cmd.data['command'] == 'DEACTIVATE IRRIGATION':
        print("TurnOFF")
    elif cmd.data['command'] == 'HIGH TEMPERATURE':
        print("TurnON")
    elif cmd.data['command'] == 'LOW TEMPERATURE':
        print("TurnOFF")
    if cmd.data['command'] == 'BAD WEATHER':
        print("TurnON")

    elif cmd.data['command'] == 'GOOD WEATHER':
        print("TurnOFF")
    elif cmd.data['command'] == 'HUMIDITY HIGH':
        print("TurnON")
    elif cmd.data['command'] == 'HUMIDITY LOW':
        print("TurnOFF")
```

python code.py - C:/Users/Sriiram V G/AppData/Local/Programs/Python/Python39/python code.py (3.9.8)

File Edit Format Run Options Window Help

```
elif cmd.data['command'] == 'HUMIDITY HIGH':
    print("TurnON")
elif cmd.data['command'] == 'HUMIDITY LOW':
    print("TurnOFF")

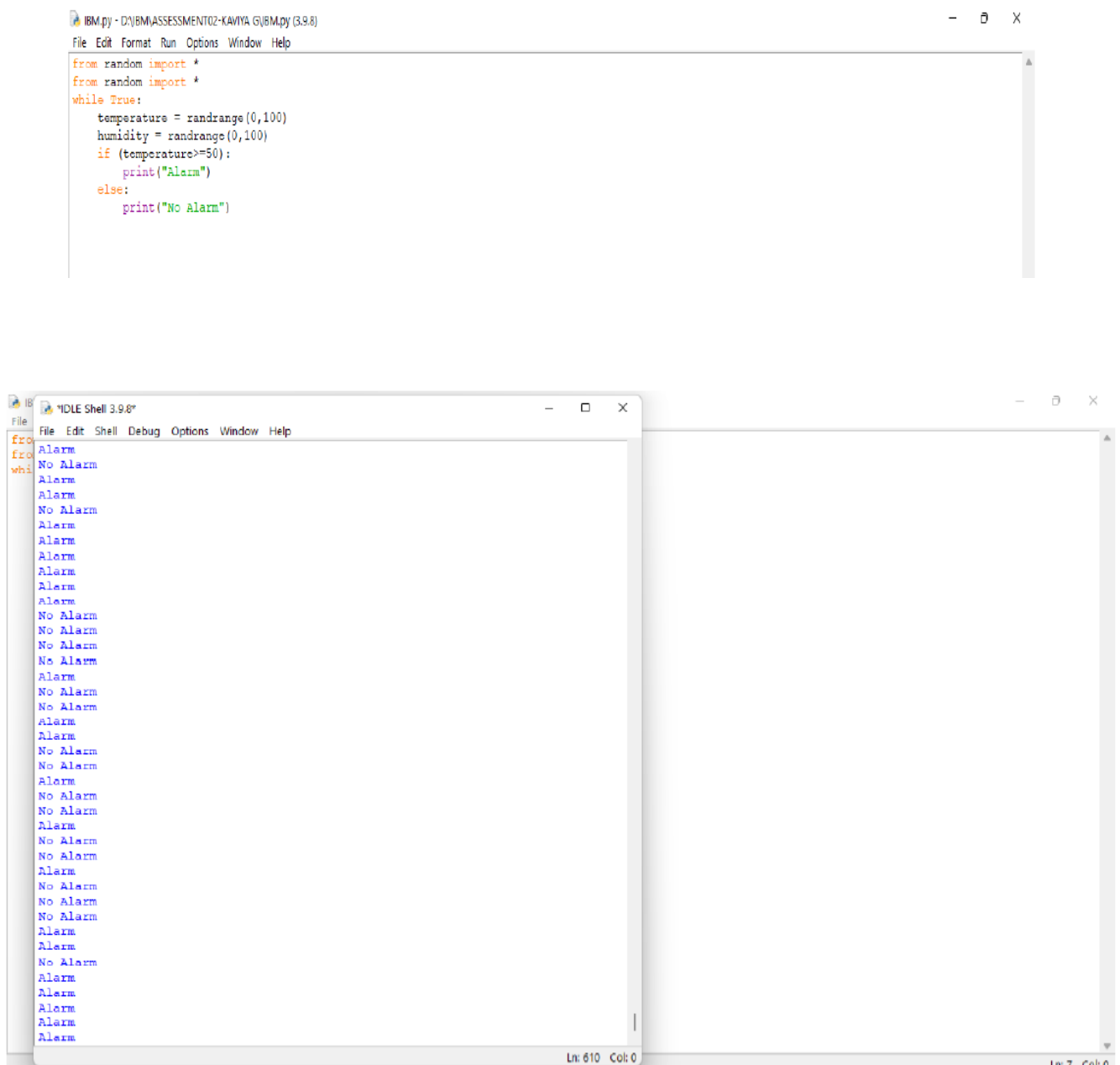
if cmd.command == "setInterval":
    if 'interval' not in cmd.data:
        print("Error - command is missing required
information: 'interval'")
    else:
        interval = cmd.data['interval']
        elif cmd.command == "print":
            if 'message' not in cmd.data:
                print("Error - command is missing required
information: 'message'")
            else:
                output=cmd.data['message']
                print(output)
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:

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

Ln: 57 Col: 4

PYTHON CODE FOR TEMPERATURE:

```
from random import *
from random import *
while True:
    temperature = randrange(0,100)
    humidity = randrange(0,100)
    if (temperature>=50):
        print("Alarm")
    else:
        print("No Alarm")
```



The top screenshot shows a Python IDE window titled 'IBM.py - D:\BM\ASSESSMENT02-KAVIYA G\IBM.py (3.9.8)'. The code in the editor is as follows:

```
from random import *
from random import *
while True:
    temperature = randrange(0,100)
    humidity = randrange(0,100)
    if (temperature>=50):
        print("Alarm")
    else:
        print("No Alarm")
```

The bottom screenshot shows the same IDE window with the program executed. The output is a list of 'Alarm' and 'No Alarm' messages. The status bar at the bottom indicates 'Ln: 610 Col: 0'.

```
import random as rand
print("WELCOME SMART FARMER")
temperature = float(rand.uniform(15,50))
if(temperature>22 and temperature<40):
    humidity = int(rand.randint(45,65))
elif(temperature<22):
    humidity = int(rand.randint(60,70))
elif(temperature>40):
    humidity = int(rand.randint(25,35))
moisture = int(rand.randint(00,70))
print("temperature:",temperature,"C","\n","humidity:",humidity,"\n","moisture:",moisture)
if(temperature>35 or moisture<20 ):
    print("Irrigation required")
    print("Activate irrigation ?")
    decision = input()
    if(decision == 'yes'):
        print("Irrigation activated")
    else:
        print('Irrigation not activated')
    else:
        print("Irrigation not required")
```

```
SPRINT 1.py - C:/Users/Sriram V G/AppData/Local/Programs/Python/Python39/SPRINT 1.py (3.9.8)
File Edit Format Run Options Window Help
import random as rand

print("WELCOME SMART FARMER")
temperature = float(rand.uniform(15,50))
if(temperature>22 and temperature<40):
    humidity = int(rand.randint(45,65))
elif(temperature<22):
    humidity = int(rand.randint(60,70))
elif(temperature>40):
    humidity = int(rand.randint(25,35))
moisture = int(rand.randint(00,70))
print("temperature:",temperature,"C","\n","humidity:",humidity,"\n","moisture:",moisture)
if(temperature>35 or moisture<20 ):
    print("Irrigation required")
    print("Activate irrigation ?")
    decision = input()
    if(decision == 'yes'):
        print("Irrigation activated")
    else:
        print('Irrigation not activated')
else:
    print("Irrigation not required")
```

```
"IDLE Shell 3.9.8"
File Edit Shell Debug Options Window Help
Python 3.9.8 (tags/v3.9.8:bb3fdecf, Nov 5 2021, 20:48:33) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Sriram V G/AppData/Local/Programs/Python/Python39/SPRINT 1.
PY
WELCOME SMART FARMER
temperature: 39.07509771201782 c
humidity: 54
moisture: 28
Irrigation required
Activate irrigation ?
>>>
```

Development of Sprint-4

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

#Provide your IBM Watson Device Credentials
organization = "157uf3"
deviceType = "abcd"
deviceId = "7654321"
authMethod = "token"
authToken = "87654321"

# 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:
    #Get Sensor Data from DHT11
    temp=random.randint(90,110)
    Humid=random.randint(60,100)
    Mois=random.randint(20,120)
    data = { 'temp': temp , 'Humid': Humid , 'Mois': Mois}

    #print data
    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s %" % Humid, "Moisture =%s deg c" % Mois "to IBM Watson")
        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoT")
            time.sleep(10)
```

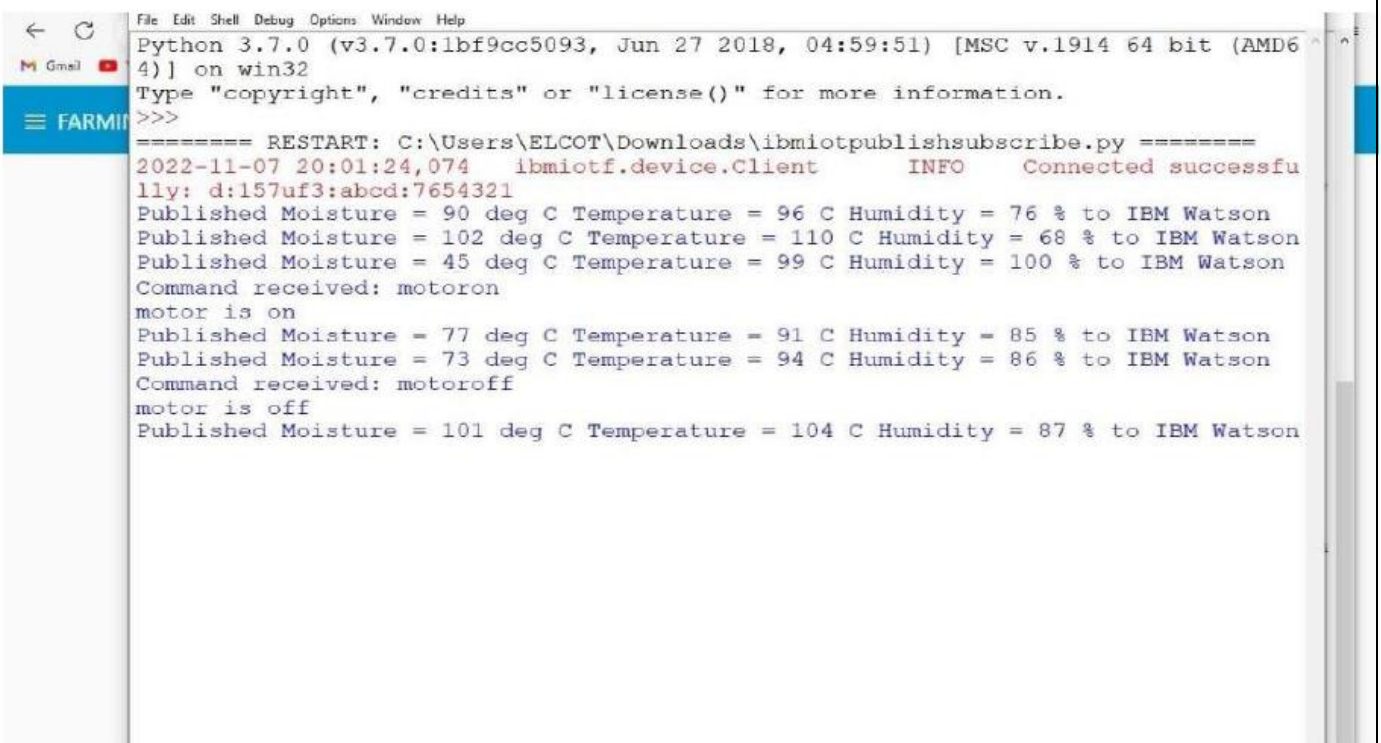
deviceCli.commandCallback = myCommandCallback # Disconnect the device
and application from the cloud deviceCli.disconnect()

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

#Provide your IBM Watson Device Credentials
organization = "157uf3"
deviceType = "abcd"
deviceId = "7654321"
authMethod = "token"
authToken = "97654321"

# 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": authMe
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....
```



The screenshot shows a Python 3.7.0 terminal window. The script connects to IBM Watson IoT, publishes moisture, temperature, and humidity data, and responds to 'motoron' and 'motoroff' commands. The output shows successful connection and data publishing.

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ELCOT\Downloads\ibmiotpublishsubscribe.py =====
2022-11-07 20:01:24,074 ibmiotf.device.Client INFO Connected successfully: d:157uf3:abcd:7654321
Published Moisture = 90 deg C Temperature = 96 C Humidity = 76 % to IBM Watson
Published Moisture = 102 deg C Temperature = 110 C Humidity = 68 % to IBM Watson
Published Moisture = 45 deg C Temperature = 99 C Humidity = 100 % to IBM Watson
Command received: motoron
motor is on
Published Moisture = 77 deg C Temperature = 91 C Humidity = 85 % to IBM Watson
Published Moisture = 73 deg C Temperature = 94 C Humidity = 86 % to IBM Watson
Command received: motoroff
motor is off
Published Moisture = 101 deg C Temperature = 104 C Humidity = 87 % to IBM Watson
```


CHAPTER 8

TESTING

8.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff.

Unit testing is a type of testing in which individual units or functions of software testing. Its primary purpose is to test each unit or function. A unit is the smallest testable part of an application. It mainly has one or a few inputs and produces a single output.

8.2 Integration Testing

Integration testing is also known as integration and testing (I&T) , is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers.

Integration Testing is a type of software testing, which is performed on software to determine the flow between two or more modules by combining them. Integration testing makes sure that the interactions between different components of the software is completed smoothly without any complication.

The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

8.3 Test Cases

Table 8.1

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Temperature Detection	Username and Password	60	60	PASS

Table 8.2

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Humidity Detection	Username and Password	48	48	PASS

Table 8.3

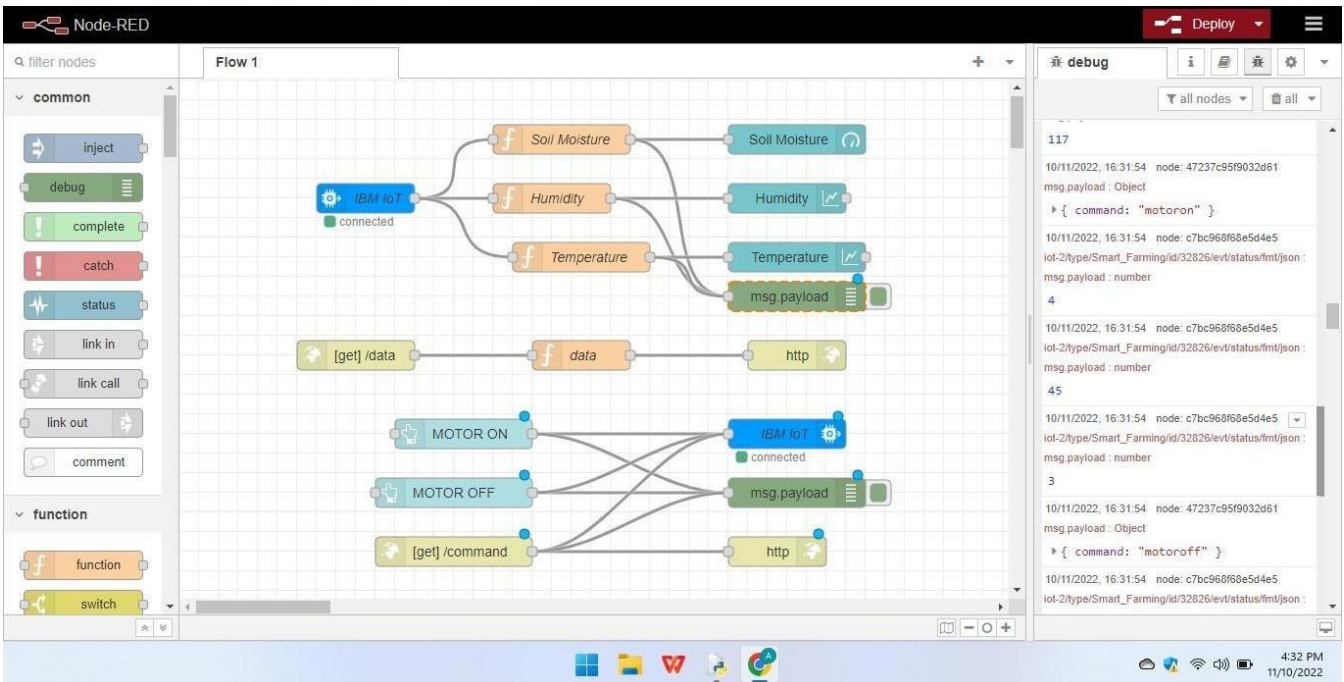
S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Moisture Detection	Username and Password	17	17	PASS

*Note : The Output Values may vary accordingly.

CHAPTER 9

RESULTS

A screenshot of a Python 3.7.0 Shell terminal window. The window title is "Python 3.7.0 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The terminal output shows a series of "Published data successfully" messages, each containing a JSON object with "soil_moisture", "temperature", and "humidity" values. These are followed by two "Message received from IBM IoT Platform" messages: "motoron" (with "Motor is switched ON") and "motoroff" (with "Motor is switched OFF"). The terminal ends with three more "Published data successfully" messages. The taskbar at the bottom shows the Windows Start button, task view, and several application icons (Word, File Explorer, etc.). The system clock in the bottom right corner shows "4:31 PM" and "11/10/2022".





SMART FARM

Temperature : 60

Humidity : 48

Moisture : 17

MOTOR ON

MOTOR OFF

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

Advantages:

- ❖ As it is a mobile friendly application one can access all the metrics in one touch.
- ❖ It has clean User interface so that user have smooth control over the application.
- ❖ The consumption of electric power is less as compared to other application.
- ❖ The moisture level and the temperature levels are monitored at regular intervals.
- ❖ It can run on all android versions.
- ❖ The application requires less memory and storage space.

Disadvantages:

- ❖ When the network connectivity is poor the performance of the application will be affected
- ❖ As it is platform dependent it cannot run on all devices.
- ❖ The application will produce inaccurate values when there is a fault or any change in API.
- ❖ The user should be more aware on the results produced.

CHAPTER 11

CONCLUSION

In this work, we successfully develop a system that can help in an automated irrigation system by analyzing the moisture level of the ground.

The smart irrigation system proves to be a useful system as it automates and regulates the watering without any manual intervention. The primary applications for this project are for farmers and gardeners who do not have enough time to water crops/plants.

The farmers are facing major problems in watering their agriculture fields. So that the Farmers can Watering their plant Smart.

CHAPTER 12

FUTURE SCOPE

- It helps in automatic irrigation for crops and also helps to maintain the water level in field.
- The system will notify on the critical conditions.
- As this is an automated device it can work even in the absence of farmer.

