Smart Farmer- IOT Enabled Smart Farming Application

PROJECT REPORT SUBMITTED BY

TEAM ID : PNT2022TMID33604

SELVAKAVIN S	**(922519106144)**
THANGAKAVIN D	**(922519106167)**
YOGESH KUMAR S	**(922519106185)**
THIRUMORTHI P	**(922519106170)**

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING IN

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

V.S.B ENGINEERING COLLEGE, KARUR

(Approved by AICTE & Affiliated by Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this mini project report titled **"Smart Farming"** is the bonafide record work by **Selvakavin S – 922519106144, Thangakavin D -**

922519106167,Yogesh Kumar S - 922519106185, Thirumoorthi P - 922519106170 for **IBM-NALAIYATHIRAN** in **VII** semester of **B.E.,** degree course **in Electronics and Communication Engineering** branch during the academic year of 2022-2023

Staff-In Charge **Evaluvator**
Mahesh Kumar .K **Mohanavel. G**

Head of the Department
Dr. P. S . Gomathi

ACKNOWLEDGEMENT

1. First and foremost, we express my thanks to our parents for providing us a very nice environment for doing this mini project. We wish to express our sincere thanks to our founder and Chairman **Shri.V.S.BALSAMY** for his endeavor in educating us in this premier institution.

2. We wish to express our appreciation and gratefulness to our principal, **Dr.V.NIRMAL KANNAN** and vice principal **Mr.T.S.KIRUBASANKAR** for their encouragement and sincere guidance.
3. We are grateful to our head of the department **Dr. P. S . Gomathi** . Our sincere thanks to all the teaching staff of V.S.B Engineering College and

our friends for their help in the successful completion of this IBM Nalaiyathiran project work.

4. Finally, we bow before God, the almighty who always had a better plan for us. We give our praise and glory to Almighty God for successful completion of this IBM Nalaiyathiran.

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

VSB ENGINEERING COLLEGE,KARUR

(Affiliated to Anna University, Chennai -600 025)

PROJECT REPORT

CHAPTER 1 - INTRODUCTION

1.1 Project Overview

Agriculture has always been the backbone of any economic development. To promote further growth of agriculture, it must be integrated with modern practices and technologies. With the wide spread acceptance of technology, it can be used in farming to make farmers perform their activity with ease. Electronics and IoT has found its application in many of the personal assistant devices. This can be extended to many vital fields like agriculture where their assistants can help solve many issues faced. Electronics can help devices get physically connected with their operational environment and analyze and collect data. IoT can help analyze and transfer the data to the user. The combination of these gives rise to an all-in-one device capable of carrying out a task.

1.2 Purpose

In recent times, the erratic weather and climatic changes have caused issues for farmers in predicting the perfect conditions to initiate farming. Though on a superficial scale it seems unpredictable, it can be determined with certain parameters with which crop planning can be done. Maintenance of farm fields during and after cultivation are also important. These can be performed by measuring soil moisture, humidity and temperature. Measurement of these parameters are performed using physical sensors. This system is in turn connected to IoT system which can provide a easy to access interface for farmers to read, analyze and take action based on the presented condition. Taking it a step ahead, the system can also gain access to motors and other electrical equipment used in farming and automate their operation. This can help with unsupervised operation ensuring accuracy and lesser response time.

CHAPTER 2 - LITERATURE SURVEY

2.1 Existing problem

There has been several attempts and solution to help farmers adopt technological practices. Few solutions restricted their performance with just suggestions and alerts. While few employed IoT independent electronics. Few of the cases of previous attempts and researches are described below.

a. “IoT based smart sensors agriculture stick for live temperature and moisture monitoring using Arduino, cloud computing & solar technology”. This work was performed using Cloud computing platform (Things Speak) for data acquisition. The circuit was designed using Arduino and DHT 11 sensors.

b. “Smart Farming using IoT, a solution for optimally monitoring farming conditions”. This work used ESP-32 based IoT platform and Blynk mobile application.

c. “Smart farming using IoT”. The automation and interface part made use of water pump and HTTP protocol for parameters monitoring using website.

The above stated prior works lacked one or two features, which when included could have enhanced the performance. In the first work, including a Raspberry Pi based controller in place of Arduino can help reduce the design area while also providing microcontroller with additional UI and IoT interfaces. In the second stated work, going with MIT app inventor instead of Blynk application can improve the possibility of feature expansion. Farmers or developers won’t need to go for a paid version of the app to include new features. In the third work, control of water pump can be enhanced with the use of servo-based water valves to direct and control the flow of water rather than using a bi-stated logic.

2.2 References

The following were the source of references:

1. ***Divya J., Divya M.,Janani V.”IoT based Smart Soil Monitoring System for Agricultural Production” 2017.***

2. ***Vaishali S, Suraj S, Vignesh G, Dhivya S and Udhayakumar S, ”Mobile Integrated Smart Irrigation Management and Monitoring System Using IOT”,2017.***

3. ***Shrihari M, ”A Smart Wireless System to Automate Produc on of Crops and Stop Intrusion Using Deep Learning” 2020.***

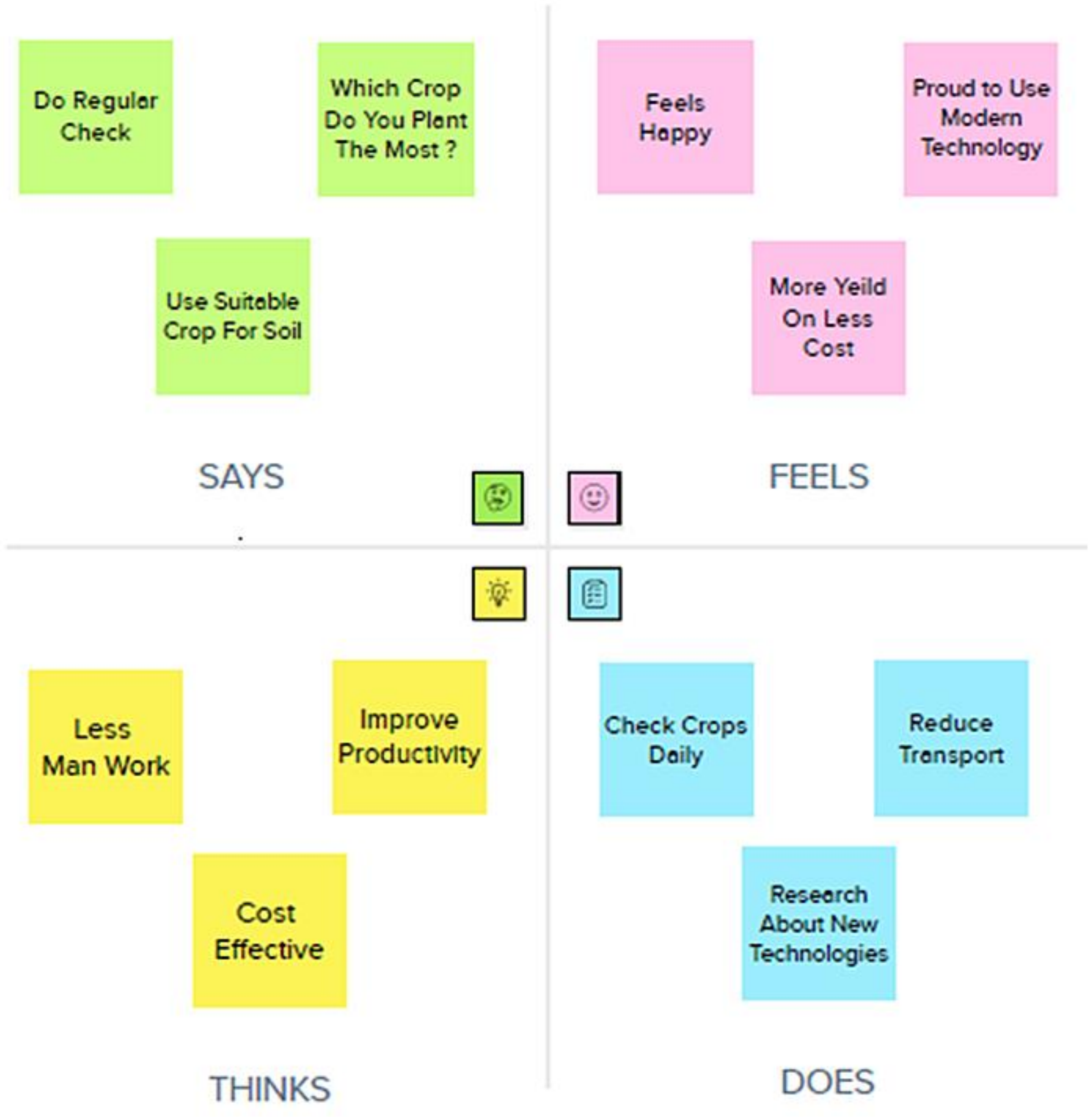
4. ***Zuraida Muhammad,Muhammad Azri Asyraf Mohd Hafez,Nor Adni Mat”Smart Agriculture Using Internet of Things with Raspberry Pi.” 2020.***

2.3 Problem Statement Definition

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you’ll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

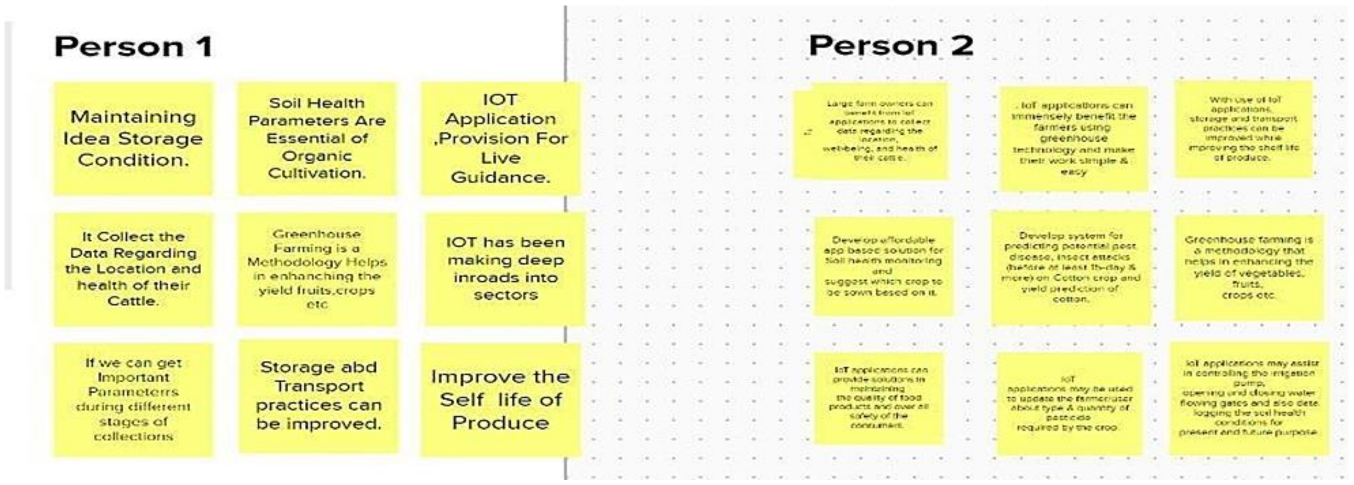
CHAPTER 3 - IDEATION AND PROPOSED SOLUTION



3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

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.



Person 1

Maintaining Idea Storage Condition.

Soil Health Parameters Are Essential of Organic Cultivation.

IOT Application ,Provision For Live Guidance.

It Collect the Data Regarding the Location and health of their Cattle.

Greenhouse Farming is a Methodology Helps in enhanching the yield fruits,crops etc

IOT has been making deep inroads into sectors

If we can get Important Parameterrs during different stages of collections

Storage abd Transport practices can be improved.

Improve the Self life of Produce

Person 2

Large farm owners can benefit from IoT applications to collect data regarding the location, well-being, and health of their cattle.

. IoT applications can immensely benefit the farmers using greenhouse technology and make their work simple & easy

With use of IoT applications, storage and transport practices can be improved while improving the shelf life of produce.

Develop affordable app based solution for Soil health monitoring and suggest which crop to be sown based on it.

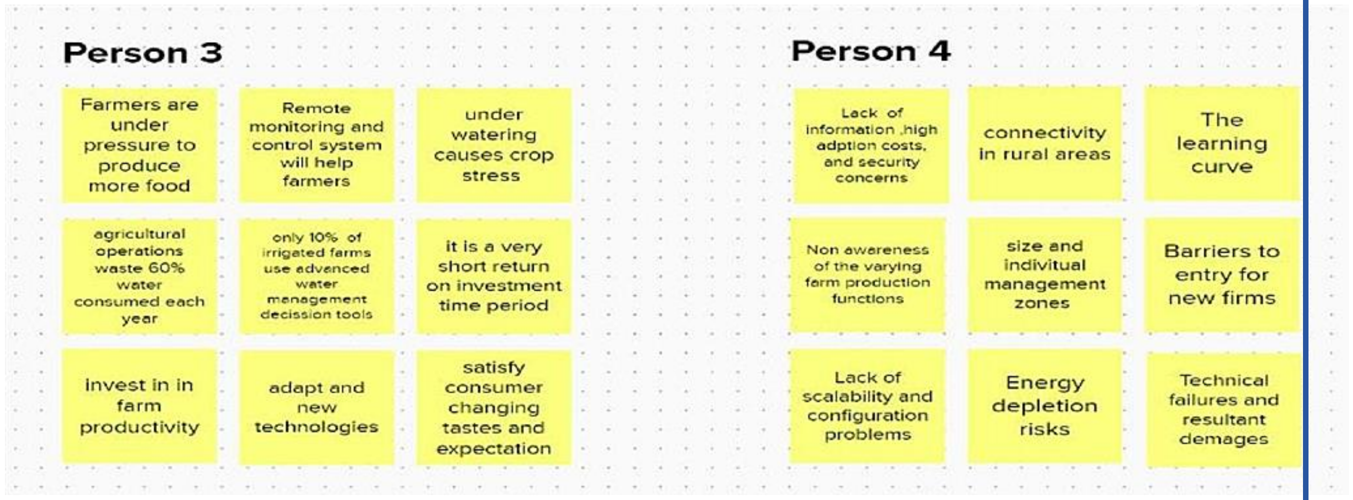
Develop system for predicting potential pest disease, insect attacks (before at least 15-day & more) on Cotton crop and yield prediction of cotton.

Greenhouse farming is a methodology that helps in enhancing the yield of vegetables, fruits, crops etc.

IoT applications can provide solutions in maintaining the quality of food products and over all safety of the consumers.

IoT applications may be used to update the farmer/user about type & quantity of pesticide required by the crop.

IoT applications may assist in controlling the irrigation pump, opening and closing water flowing gates and also data logging the soil health conditions for present and future purpose.



Person 3

Farmers are under pressure to produce more food

Remote monitoring and control system will help farmers

under watering causes crop stress

agricultural operations waste 60% water consumed each year

only 10% of irrigated farms use advanced water management decission tools

it is a very short return on investment time period

invest in in farm productivity

adapt and new technologies

satisfy consumer changing tastes and expectation

Person 4

Lack of information ,high adption costs, and security concerns

connectivity in rural areas

The learning curve

Non awareness of the varying farm production functions

size and indivitual management zones

Barriers to entry for new firms

Lack of scalability and configuration problems

Energy depletion risks

Technical failures and resultant demages

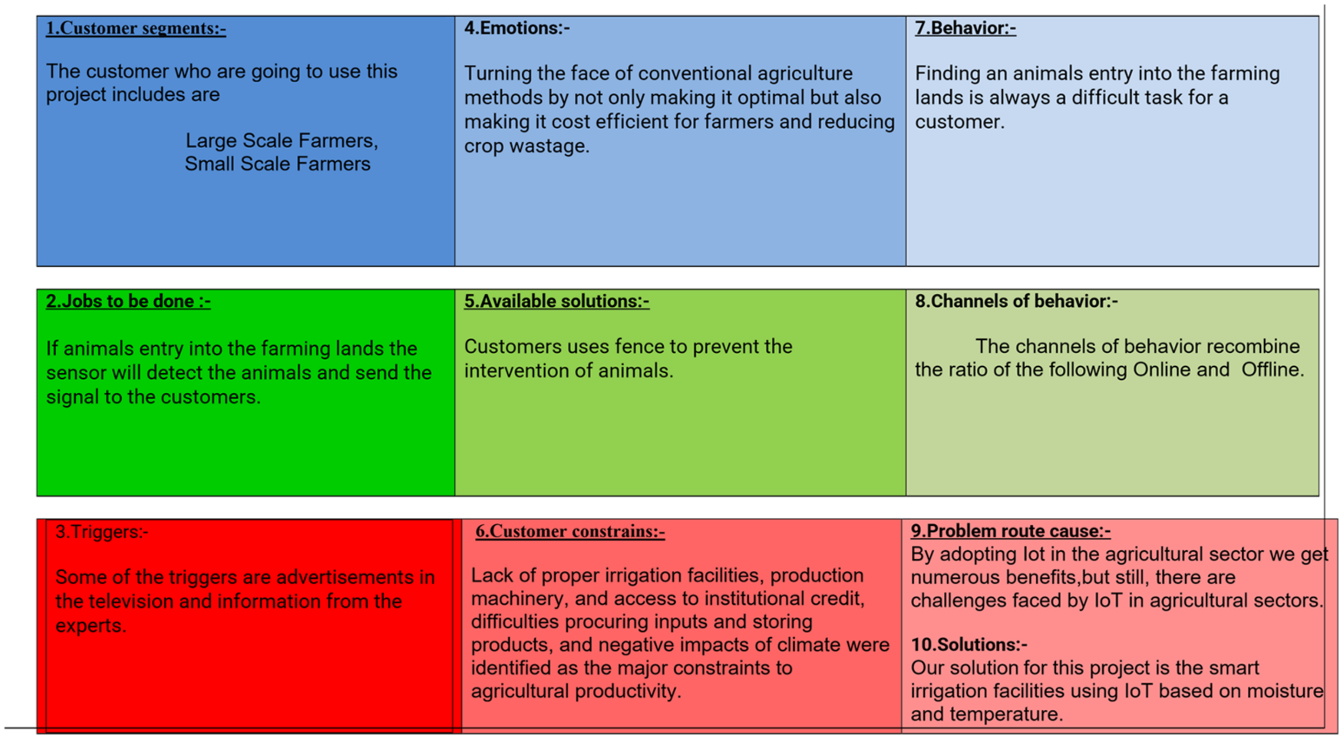
Proposed Solution Template:

S.NO:	PARAMETER	DESCRIPTION

1	Problem Statement (Problem to be solved)	Farmers are under pressure to produce more food and use less energy and water in the process. A remote monitoring and control system will help farmers deal effectively with these pressures. Irrigated farms typically deploy a single pump to irrigate 80 to 100 acres of land.
2	Idea /Solutiondescription	Smart farming is an emerging concept that refers to managing farms using technologies like IoT, robotics, drones and AI to increase the quantity and quality of products while optimizing the human labour required by production.
3	Novelty / Uniqueness	Unlike genetic resources found in the natural world, agricultural crops are truly a human mediated form of biodiversity. Through the process of domestication, human beings have for over 10,000 years been selecting and breeding plant species from the wild and creating new diversity adapted specifically for cultivation
4	Social Impact / Costomer Satisification	It determines how happy customers are with a company's products, services, and capabilities. Customer satisfaction information, including surveys and ratings, can help a company determine how to best improve or changes its products and services.
5	Business Model(Revenue Model)	The smart farming devices designed in such a way that should be profitable compared to traditional farming methods and the device should be reusable . The cost of the devices should be less compared to cost required for traditional farming. Hence the product must be profitable it does not make losses in any cases.

6	Scalability Of The Solution	The ability of the device's to increase or decrease in performance and cost in response to changes in application. The property of a device to handle a growing amount of works by adding resource to system.

3.4 Problem Solution fit



1.Customer segments:- The customer who are going to use this project includes are Large Scale Farmers, Small Scale Farmers	**4.Emotions:-** Turning the face of conventional agriculture methods by not only making it optimal but also making it cost efficient for farmers and reducing crop wastage.	**7.Behavior:-** Finding an animals entry into the farming lands is always a difficult task for a customer.
2.Jobs to be done :- If animals entry into the farming lands the sensor will detect the animals and send the signal to the customers.	**5.Available solutions:-** Customers uses fence to prevent the intervention of animals.	**8.Channels of behavior:-** The channels of behavior recombine the ratio of the following Online and Offline.
3.Triggers:- Some of the triggers are advertisements in the television and information from the experts.	**6.Customer constrains:-** Lack of proper irrigation facilities, production machinery, and access to institutional credit, difficulties procuring inputs and storing products, and negative impacts of climate were identified as the major constraints to agricultural productivity.	**9.Problem route cause:-** By adopting Iot in the agricultural sector we get numerous benefits,but still, there are challenges faced by IoT in agricultural sectors. **10.Solutions:-** Our solution for this project is the smart irrigation facilities using IoT based on moisture and temperature.

CHAPTER 4 - REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution. FR No.	Non-Functional Requirement	Description

NFR-1	Usability	Usability includes easy learn ability, efficiency in use, remember ability, lack of errors in operation and subjective pleasure.
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. The model uses dedicated and shared protection schemes to avoid farm service outages.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Log in to system	Check Credentials Check Roles of Access.

FR-4	Manage Modules	Manage System Admins Manage Roles of User Manage User permission
FR-5	Check whether details	Temperature details Humidity details
FR-6	Log out	Exit

Non-functional Requirements:

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

NFR-4	Performance	the idea of implementing integrated sensors with sensing soil and environmental or ambient parameters in farming will be more efficient for overall monitoring.
NFR-5	Availability	Automatic adjustment of farming equipment made possible by linking information like crops/weather and equipment to auto-adjust temperature, humidity, etc.
NFR-6	Scalability	Scalability is a major concern for IoT platforms. It has shown that different architectural choices of IoT platforms affect system scalability and that automatic real time decision-making is feasible in an environment composed of dozens of thousand.

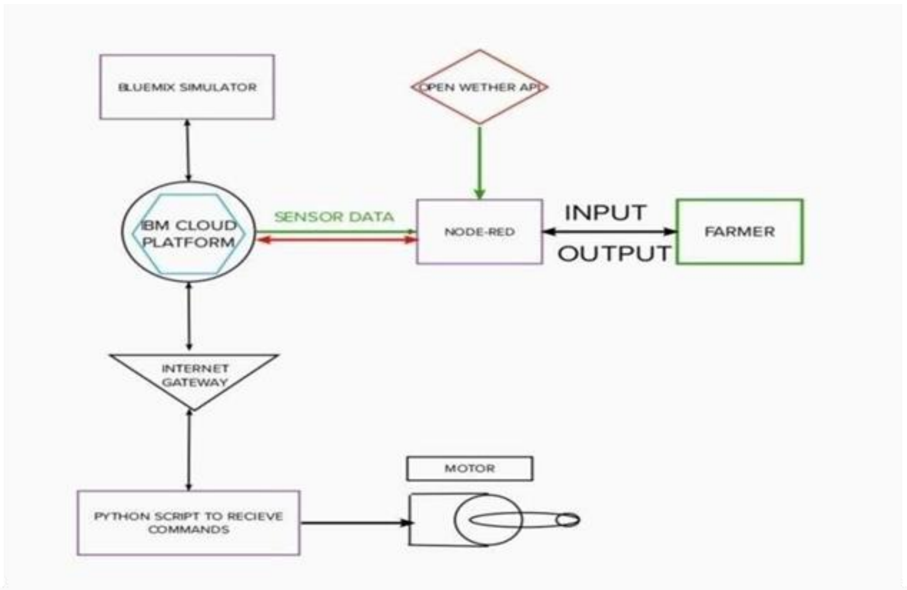
CHAPTER 5 - PROJECT DESIGN

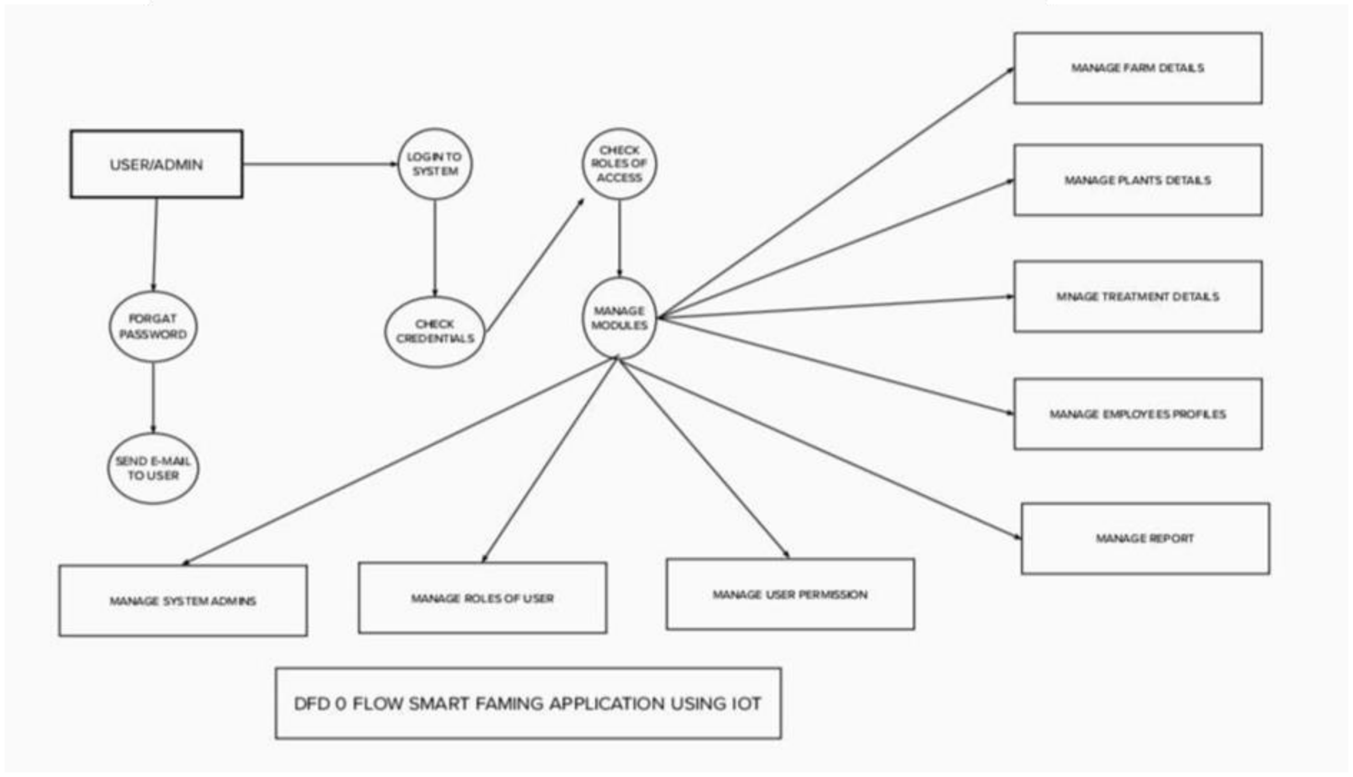
Data Flow Diagrams:

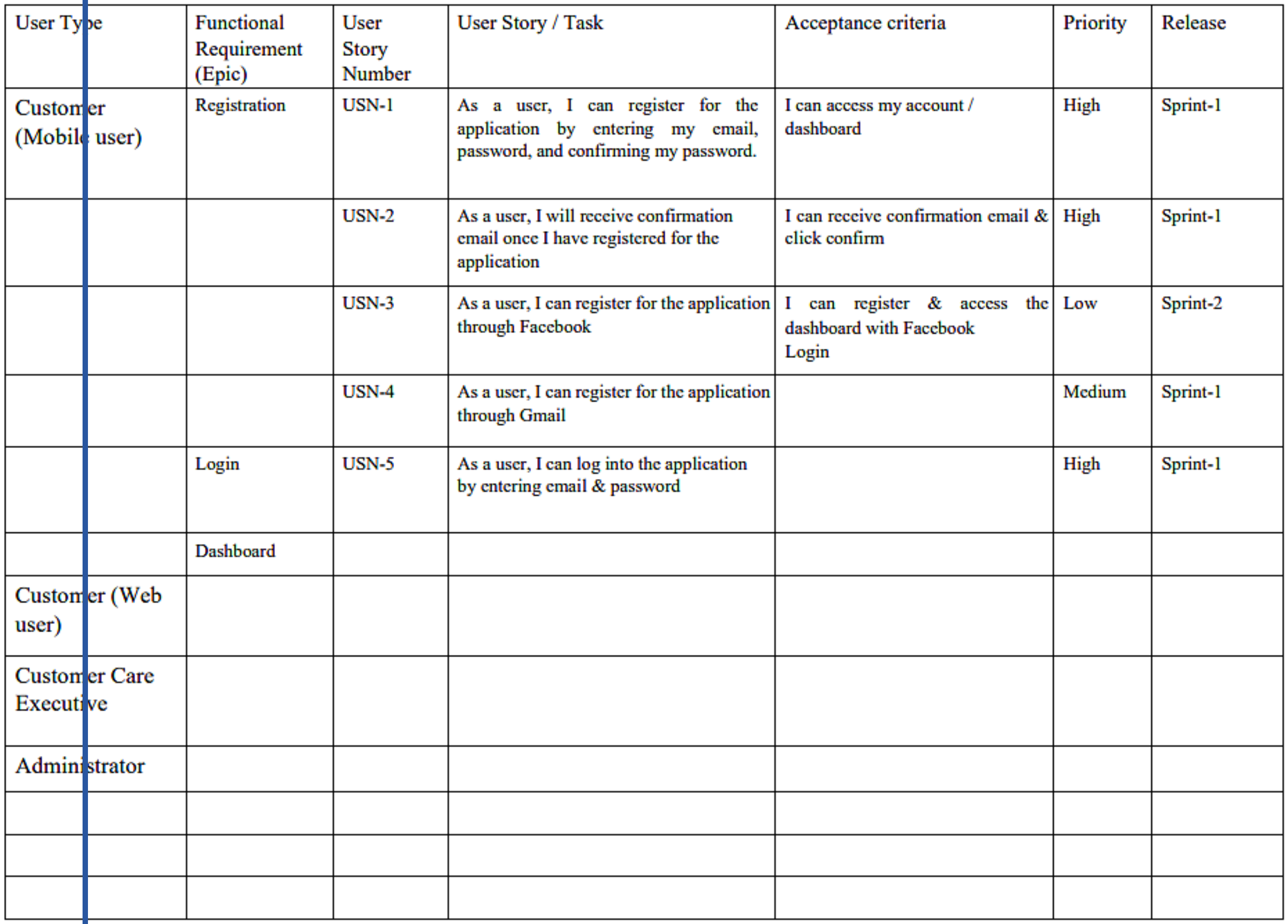
1. The different soil parameters temperature, soil moistures and then humidity are sensed using different sensors and obtained value is stored in the IBM cloud.

2. Arduino UNO is used as a processing Unit that process the data obtained from the sensors 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.


BLUEMIX SIMULATOR
OPEN WETHER API
IBM CLOUD PLATFORM
SENSOR DATA
NODE-RED
INPUT
OUTPUT
FARMER
INTERNET GATEWAY
MOTOR
PYTHON SCRIPT TO RECIEVE COMMANDS


USER/ADMIN
LOGIN TO SYSTEM
CHECK ROLES OF ACCESS
FORGAT PASSWORD
CHECK CREDENTIALS
MANAGE MODULES
SEND E-MAIL TO USER
MANAGE FARM DETAILS
MANAGE PLANTS DETAILS
MNAGE TREATMENT DETAILS
MANAGE EMPLOYEES PROFILES
MANAGE REPORT
MANAGE SYSTEM ADMINS
MANAGE ROLES OF USER
MANAGE USER PERMISSION
DFD 0 FLOW SMART FAMING APPLICATION USING IOT



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	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	Sprint-1
		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	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)						
Customer Care Executive						
Administrator						

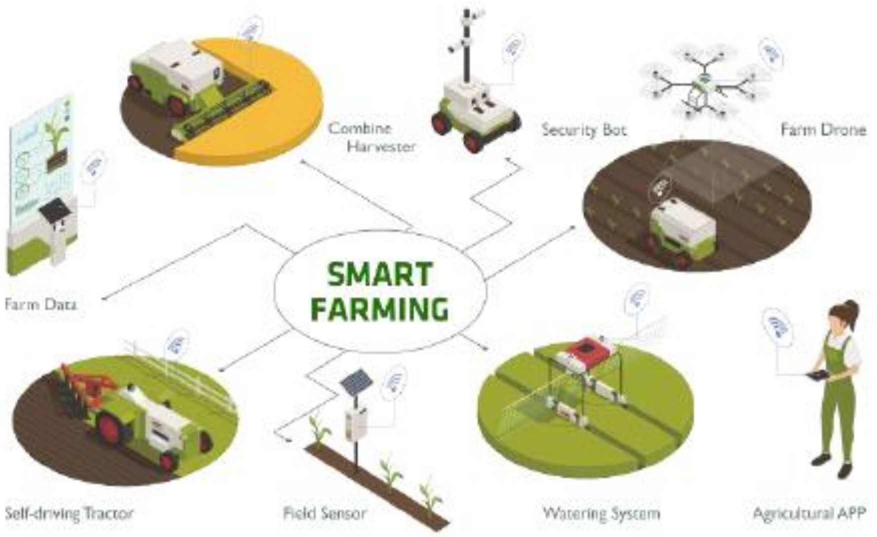
1. All the collected data are provided to the user through a mobile application that was developed using the MIT app inventor. The user could plan through an app, weather to water the crop or not depending upon the sensor values. By using the app they can remotely operate to the motor switch.

Solution and Technical Architecture:

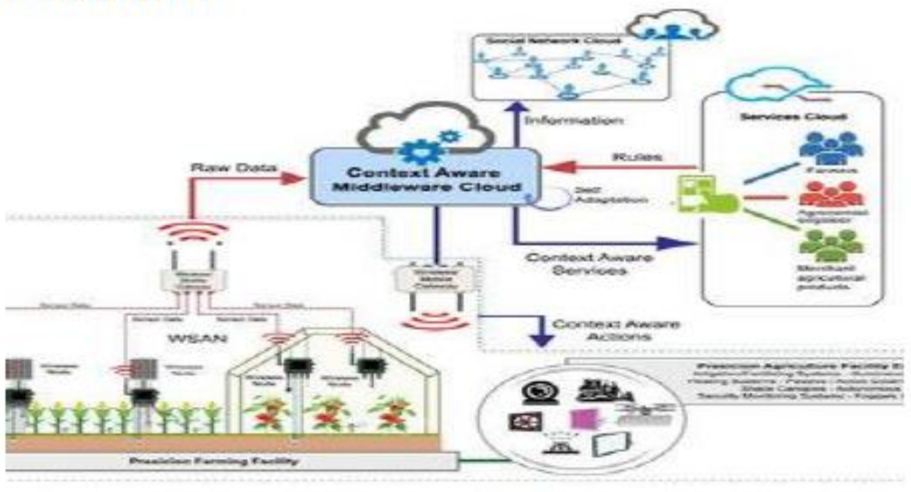
The technical architecture diagram is as follows:

Solution Architecture

Applications:

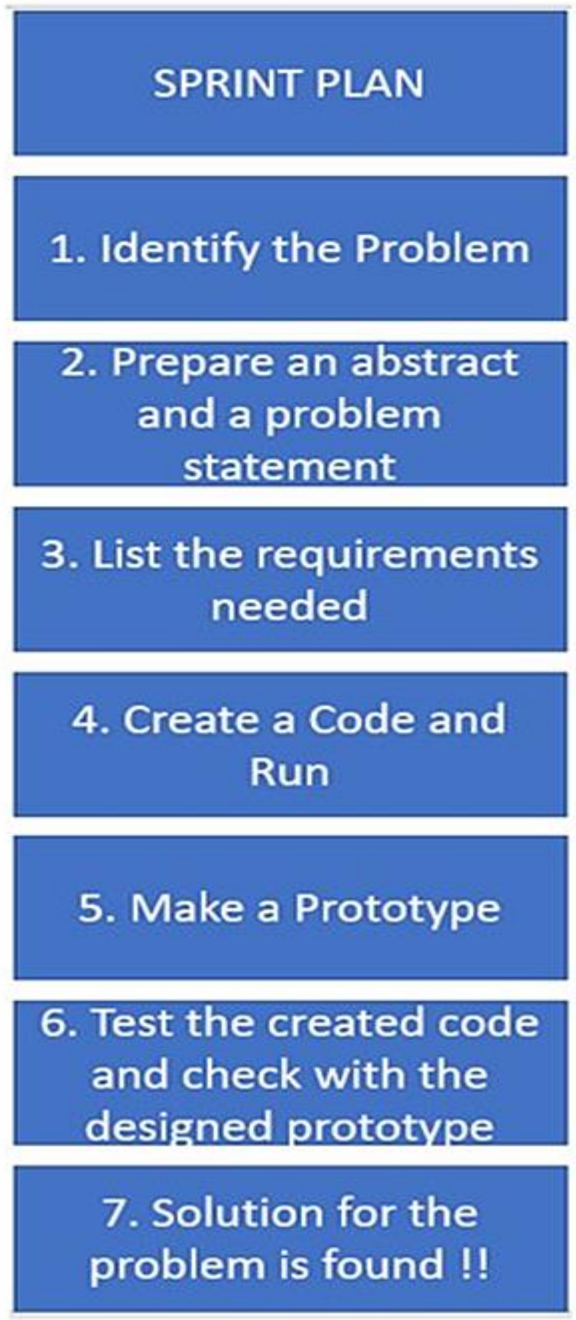


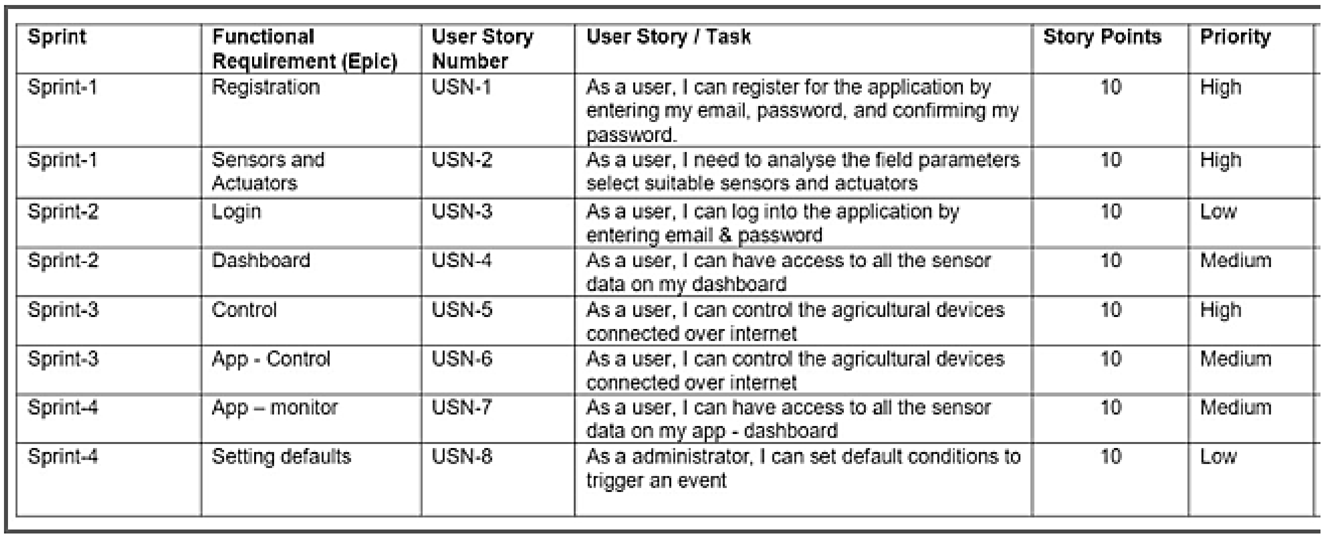
Middleware:



CHAPTER 6 - PROJECT PLANNING AND SCHEDULING

Sprint Planning and Estimation :


SPRINT PLAN
1. Identify the Problem
2. Prepare an abstract and a problem statement
3. List the requirements needed
4. Create a Code and Run
5. Make a Prototype
6. Test the created code and check with the designed prototype
7. Solution for the problem is found !!



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	10	High
Sprint-1	Sensors and Actuators	USN-2	As a user, I need to analyse the field parameters select suitable sensors and actuators	10	High
Sprint-2	Login	USN-3	As a user, I can log into the application by entering email & password	10	Low
Sprint-2	Dashboard	USN-4	As a user, I can have access to all the sensor data on my dashboard	10	Medium
Sprint-3	Control	USN-5	As a user, I can control the agricultural devices connected over internet	10	High
Sprint-3	App - Control	USN-6	As a user, I can control the agricultural devices connected over internet	10	Medium
Sprint-4	App – monitor	USN-7	As a user, I can have access to all the sensor data on my app - dashboard	10	Medium
Sprint-4	Setting defaults	USN-8	As a administrator, I can set default conditions to trigger an event	10	Low

Velocity:

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

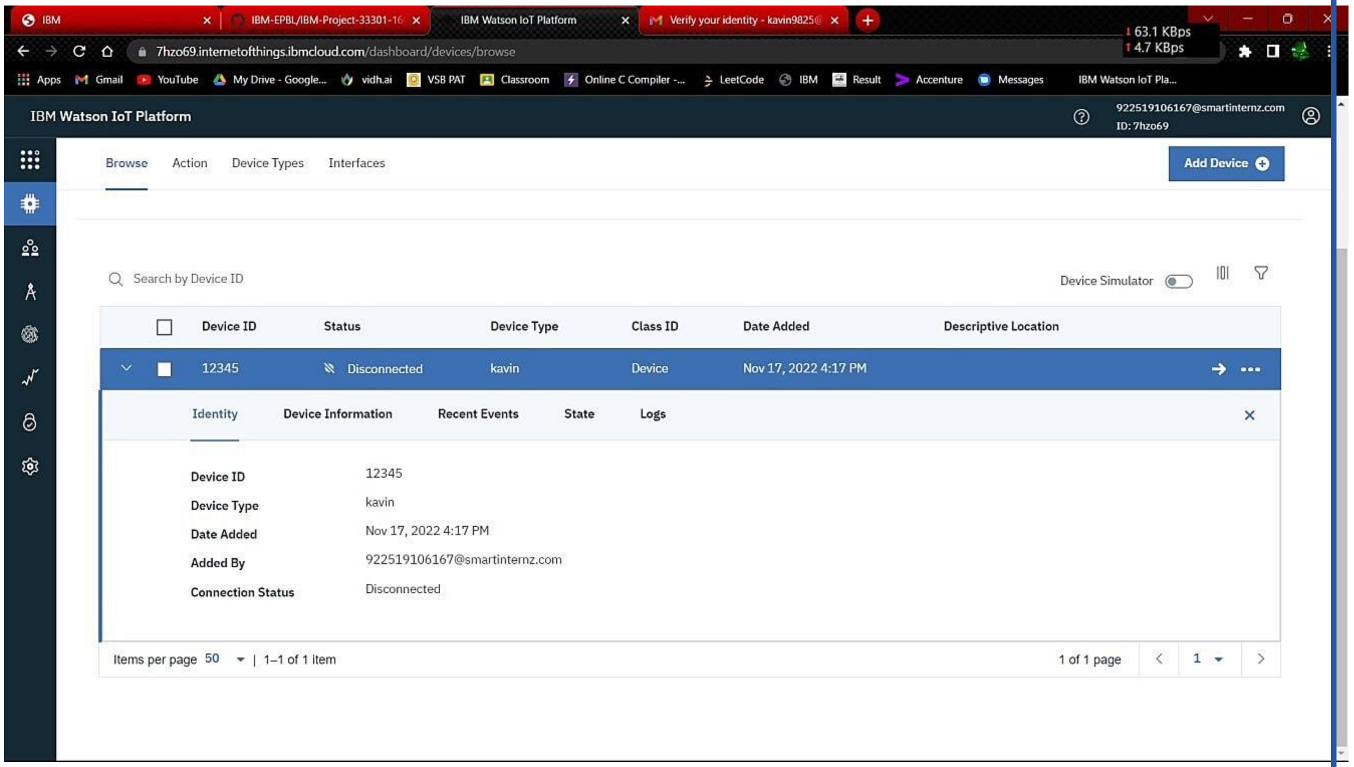
Burndown Chart:

A burndown chart is the graphical representation of work left to be done versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

CHAPTER 7 - CODING AND SOLUTIONING

Configuration of the IBM Watson IOT Platform and a device:

In the IBM Watson IOT Platform, under the catalog list, under the Internet of Things platform, a device has been created. From that the device credentials such as Device ID, Device Type, Organization ID,
Authentication token were obtained.



	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
	12345	Disconnected	kavin	Device	Nov 17, 2022 4:17 PM	

Identity | Device Information | Recent Events | State | Logs

Device ID	12345
Device Type	kavin
Date Added	Nov 17, 2022 4:17 PM
Added By	922519106167@smartinternz.com
Connection Status	Disconnected

Items per page 50 | 1–1 of 1 item   1 of 1 page

Development of Python Script to publish data to IBM Watson IOT platform:

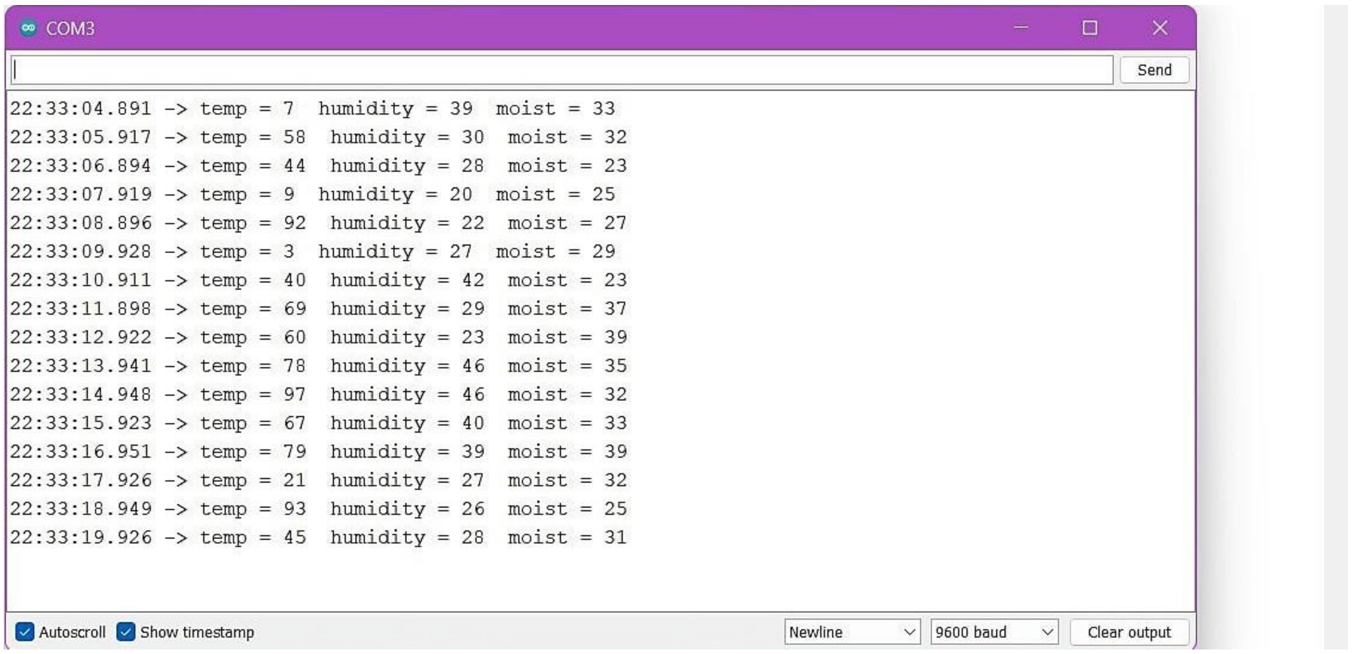
Code:

```
import wiotp.sdk.device

import time import os

import datetime

import random myConfig = {
```

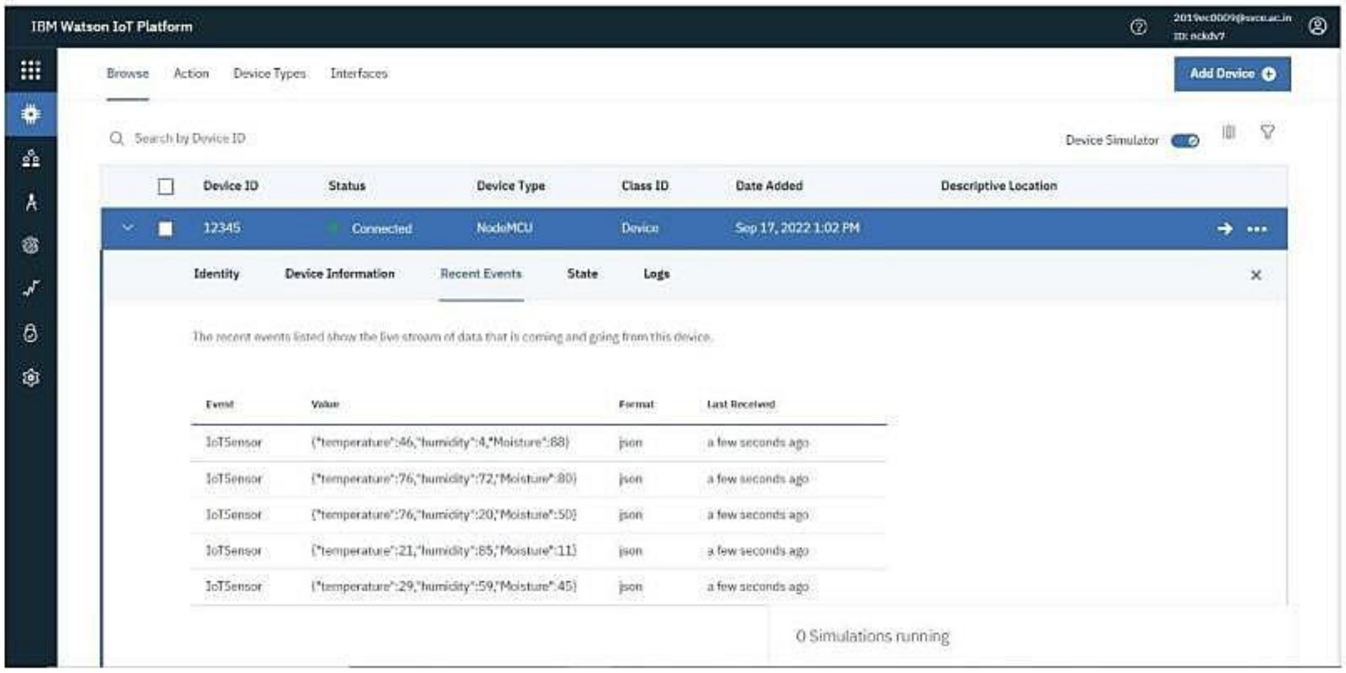
```
          "identity": {
          "orgId": "u9qhfi",
          "typeId": "Devicetypel",
          "deviceId": "DeviceID1"
          },
          "auth": {
         "token": ")hSb7_ZD+evl2fRhXi"
          } }
         client = wiotp.sdk.device.DeviceClient            (
config=myConfig,logHandlers=None)
  client.connect () def myCommandCallback (cmd) :
  print ("Message received from IBM IoT Platform: %s" % cmd.data['command'])
  m=cmd.data['command']
   if (m=="motoron"):
      print ("Motor is switched on")
  elif (m=="motoroff"):
      print ("Motor is switched OFF")
      print (" ")
 while True:
   soil=random.randint (0,100)
   temp=random.randint (-20, 125)
   hum=random.randint (0, 100)
   myData={'soil moisture': soil, 'temperature':temp, 'humidity':hum}
```

```
    client.publishEvent (eventId="status", msgFormat="json", data=myData, qos=0 ,
onPublish=None)
        print ("Published data Successfully: %s", myData)
        time.sleep (2)
client.commandCallback = myCommandCallback
client.disconnect ()
```

IBM Cloud after publishing data



```
22:33:04.891 -> temp = 7  humidity = 39  moist = 33
22:33:05.917 -> temp = 58  humidity = 30  moist = 32
22:33:06.894 -> temp = 44  humidity = 28  moist = 23
22:33:07.919 -> temp = 9  humidity = 20  moist = 25
22:33:08.896 -> temp = 92  humidity = 22  moist = 27
22:33:09.928 -> temp = 3  humidity = 27  moist = 29
22:33:10.911 -> temp = 40  humidity = 42  moist = 23
22:33:11.898 -> temp = 69  humidity = 29  moist = 37
22:33:12.922 -> temp = 60  humidity = 23  moist = 39
22:33:13.941 -> temp = 78  humidity = 46  moist = 35
22:33:14.948 -> temp = 97  humidity = 46  moist = 32
22:33:15.923 -> temp = 67  humidity = 40  moist = 33
22:33:16.951 -> temp = 79  humidity = 39  moist = 39
22:33:17.926 -> temp = 21  humidity = 27  moist = 32
22:33:18.949 -> temp = 93  humidity = 26  moist = 25
22:33:19.926 -> temp = 45  humidity = 28  moist = 31
```



	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
	12345	Connected	NodeMCU	Device	Sep 17, 2022 1:02 PM	

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

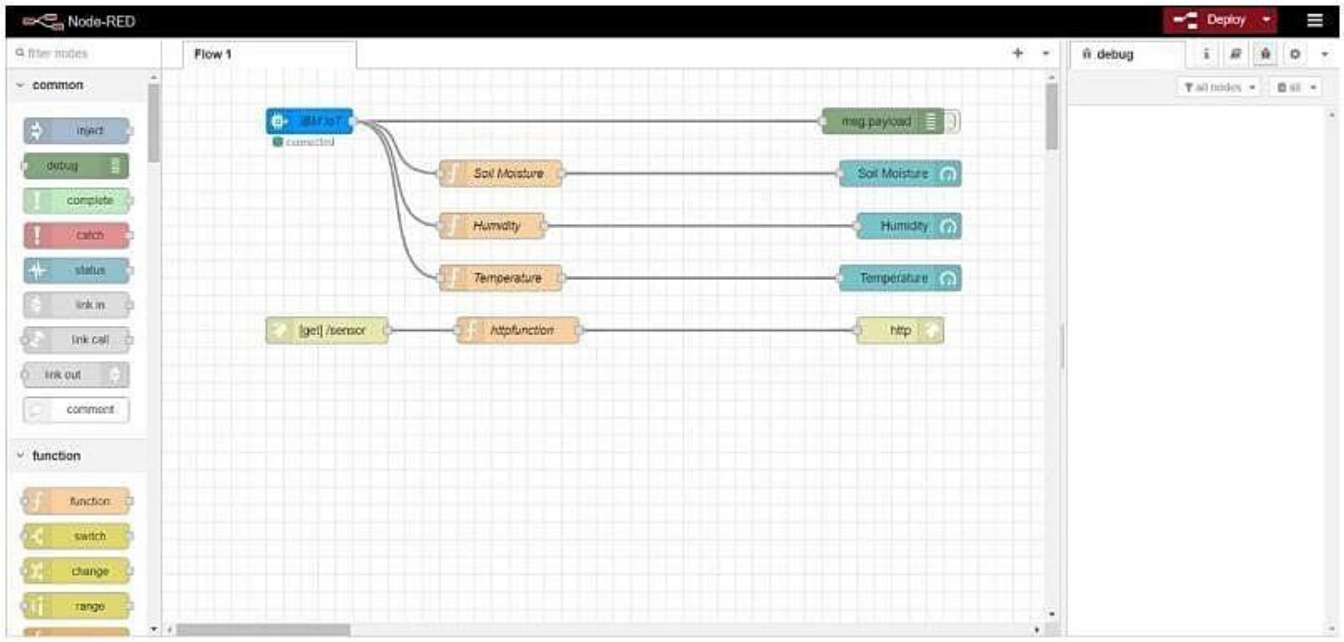
Event	Value	Format	Last Received
IoTSensor	{"temperature":46,"humidity":4,"Moisture":88}	json	a few seconds ago
IoTSensor	{"temperature":76,"humidity":72,"Moisture":80}	json	a few seconds ago
IoTSensor	{"temperature":76,"humidity":20,"Moisture":50}	json	a few seconds ago
IoTSensor	{"temperature":21,"humidity":85,"Moisture":11}	json	a few seconds ago
IoTSensor	{"temperature":29,"humidity":59,"Moisture":45}	json	a few seconds ago

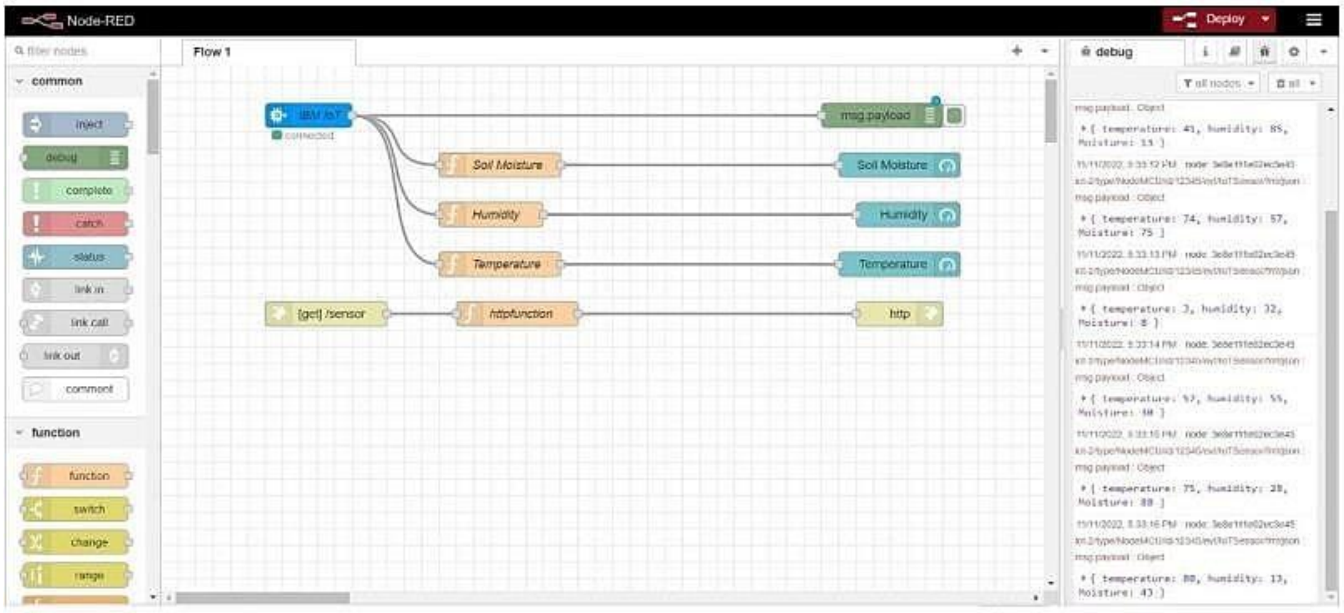
0 Simulations running

Creation of Node Red Service for device events:

In the IBM Watson IOT platform, under the catalog, under the Node Red app service, an application is deployed using cloud foundry. In the cloud foundry, a group has been created and using the ci pipeline, the app url is obtained. Using the URL, the Node red is launched. The IBM Watson IOT platform is connected to Node red using the IBM IoT palette. Using appropriate palettes, the data published in the IBM IoT platform is printed in the debug window of Node red

Code block for the function palette:





```
msg.payload : Object
▸{ temperature: 41, humidity: 85, Moisture: 15 }
11/11/2022, 8:33:12 PM  node: 3e8e11fe02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 74, humidity: 57, Moisture: 75 }
11/11/2022, 8:33:13 PM  node: 3e8e11fe02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 3, humidity: 32, Moisture: 8 }
11/11/2022, 8:33:14 PM  node: 3e8e11fe02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 57, humidity: 55, Moisture: 38 }
11/11/2022, 8:33:15 PM  node: 3e8e11fe02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 75, humidity: 28, Moisture: 88 }
11/11/2022, 8:33:16 PM  node: 3e8e11fe02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 80, humidity: 13, Moisture: 43 }
```

Soil moisture:

```
Soil = msg.payload.Moisture
msg.payload = "Soil Moisture : "

global.set('m',Soil)

msg.payload = Math.round(Soil)  return msg;
```

Humidity:

```
Humidity=msg.payload.humidity
msg.payload="Humidity:
"  global.set('h',Humidity)
msg.payload=Math.round(Humidity)  return msg;
```

Temperature:

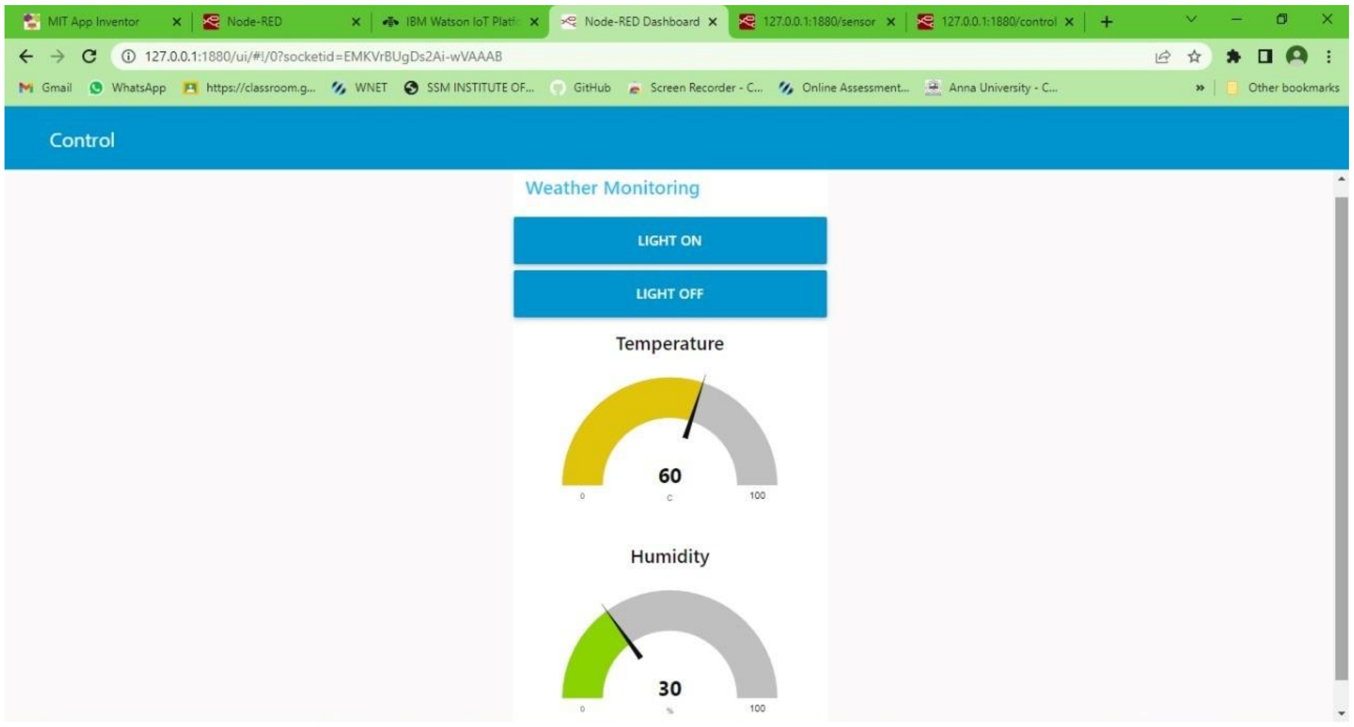
```
Temperature=msg.payload.temperature
msg.payload="Temperature:"
global.set('t',Temperature)
msg.payload=Math.round(Temperature)  return msg;
```

HTTP Function:

```
msg.payload = {"Temperature:":global.get('t'),"Humidity:":
global.get('h'),"Soil Moisture:":global.get('m')}
return msg;
```

1. **Creation of Website dashboard:**

A website dashboard has been created using the gauge palette. It can be accessed by adding "/ui" in the main url of Node red. This dashboard displays the gauge representation of the data published in the IBM IOplatform.



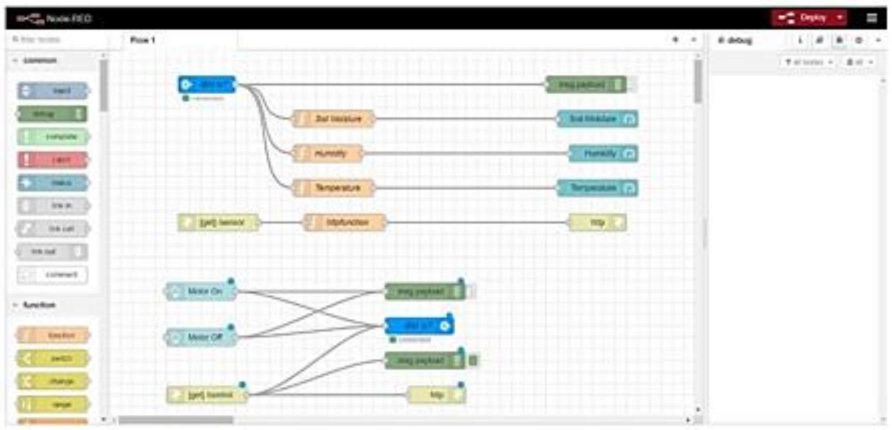
Python code used:

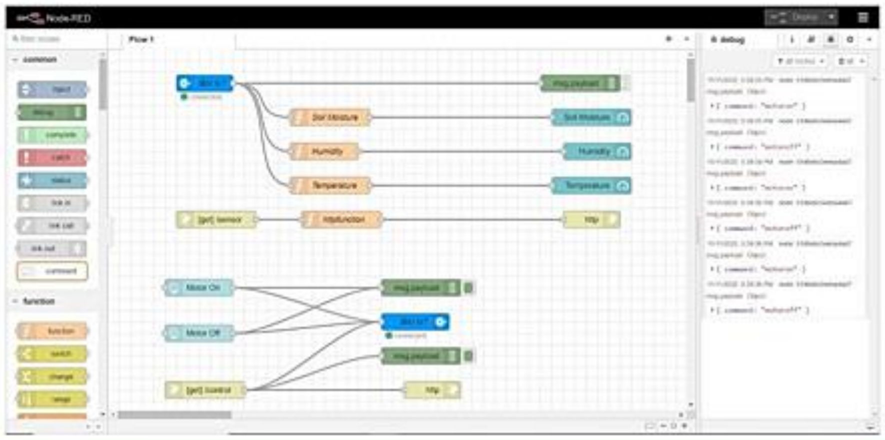
```
print("Caught exception
connecting device: %s" %
str(e))
sys.exit()
# Connect and send a
datapoint "hello" with
value "world" into the
cloud as an event of type
"greeting" 10 times
deviceCli.connect()
while True: #Get Sensor
Data from DHT11
```

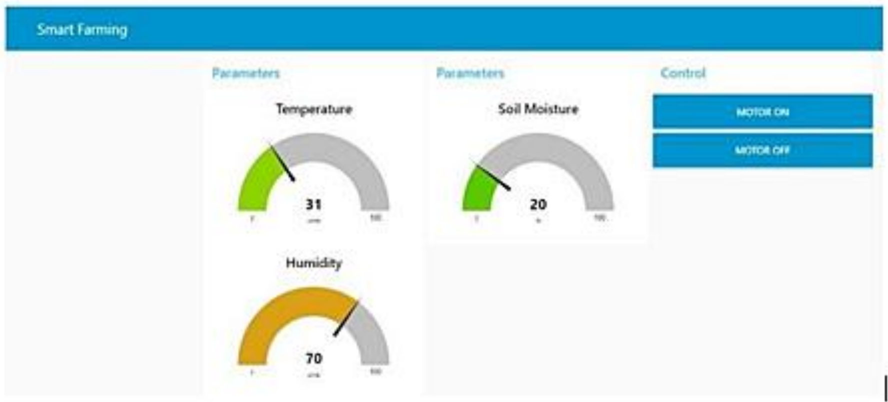
```
temp=random.randint(0,10
0)
pulse=random.randint(0,10
0)
moisture=
random.randint(0,100)
humidity=random.randint(0
,100);
lat = 17 lon = 18 data = {
'temperature' : temp,
'humidity' :
humidity, 'Moisture' :
moisture}
#print data
def
myOnPublishCallback():
print ("Published
Temperature = %s C" %
temp, "Humidity
= %s %%" % humidity,
"Soil Moisture = %s %%"
%
moisture,"to IBM Watson")
success =
deviceCli.publishEvent("Io
TSensor", "json", data,
qos=0,
                on_publish=myOnPublishC
                allback)
                if not success:
                print("Not connected to
                IoTF")
                time.sleep(1)
                deviceCli.commandCallbac
                k = myCommandCallback
                # Disconnect the device and
                application from the cloud
deviceCli.disconnect()
```

Creation of Node red service for device commands:

In addition to the palettes used in the Sprint-2, additional palettes such as buttons have been included to control devices by giving commands and the output is printed in the debug whenever a specific command is given.







Development of Python script to subscribe command from the IBM IOT platform:

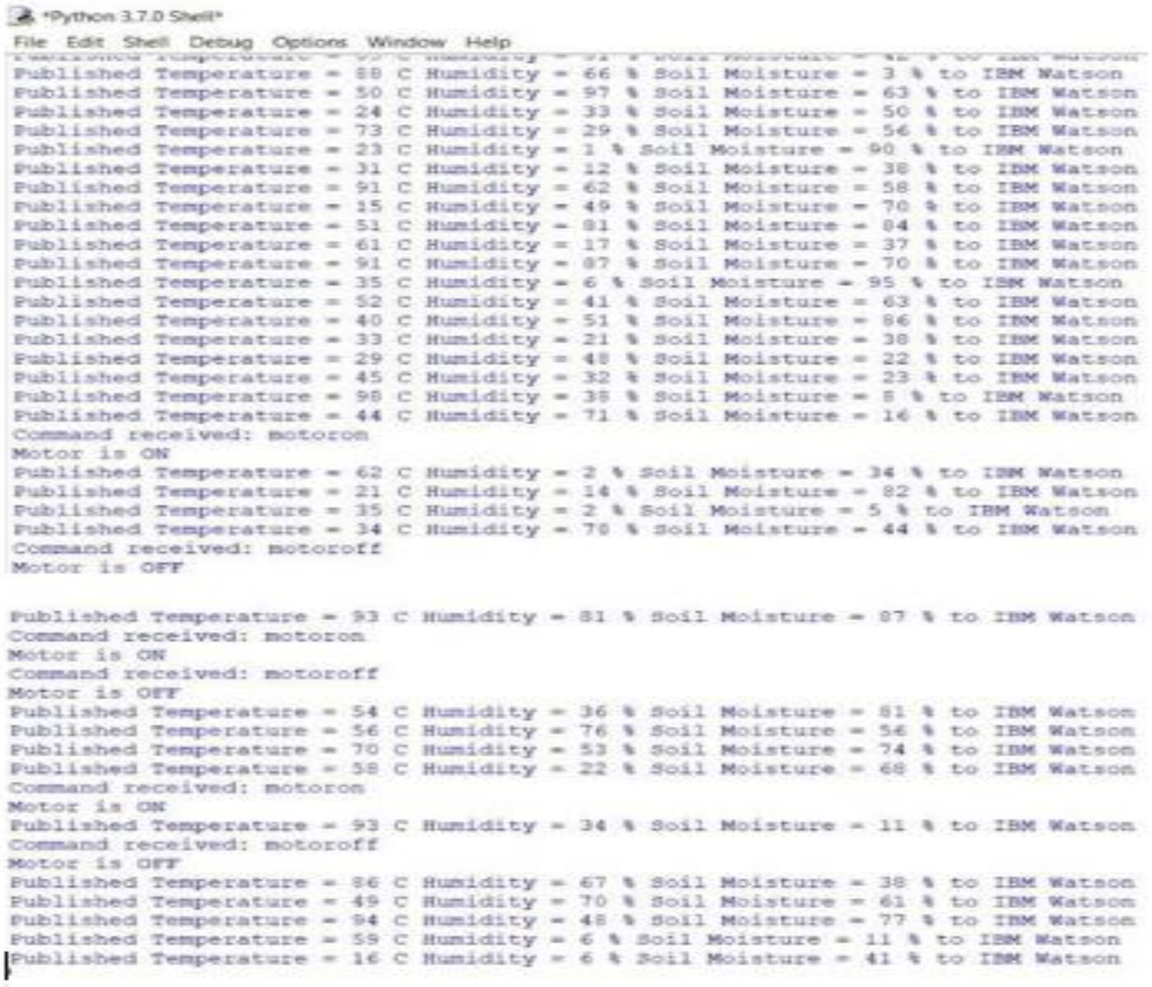
CODE:

import time

import sys

```
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "nckdv7"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"
# Initialize GPIO
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
status=cmd.data['command']
        if status=="motoron":
              print("Motor is ON")
        else:
            print("Motor is OFF")
#print(cmd)
 try:
      deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
"authmethod": 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)
      pulse=random.randint(0,100)
      moisture= random.randint(0,100)
      humidity=random.randint(0,100);
     lat = 17 lon = 18 data = { 'temperature' : temp, 'humidity' : humidity, 'Moisture'
moisture}
#print data

def myOnPublishCallback():
           print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %
humidity, "Soil Moisture = %s %%" % moisture,"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()
```



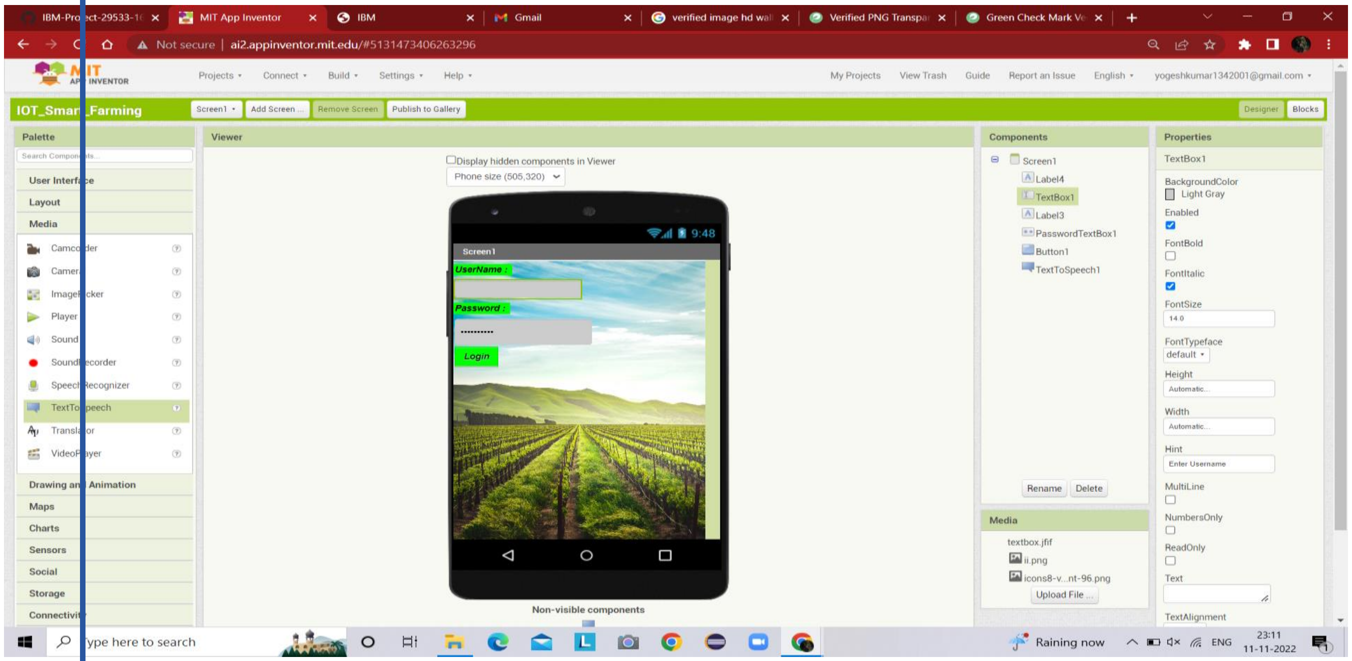
```
Published Temperature = 88 C Humidity = 66 % Soil Moisture = 3 % to IBM Watson
Published Temperature = 50 C Humidity = 97 % Soil Moisture = 63 % to IBM Watson
Published Temperature = 24 C Humidity = 33 % Soil Moisture = 50 % to IBM Watson
Published Temperature = 73 C Humidity = 29 % Soil Moisture = 56 % to IBM Watson
Published Temperature = 23 C Humidity = 1 % Soil Moisture = 90 % to IBM Watson
Published Temperature = 31 C Humidity = 12 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 91 C Humidity = 62 % Soil Moisture = 58 % to IBM Watson
Published Temperature = 15 C Humidity = 49 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 51 C Humidity = 81 % Soil Moisture = 84 % to IBM Watson
Published Temperature = 61 C Humidity = 17 % Soil Moisture = 37 % to IBM Watson
Published Temperature = 91 C Humidity = 87 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 35 C Humidity = 6 % Soil Moisture = 95 % to IBM Watson
Published Temperature = 52 C Humidity = 41 % Soil Moisture = 63 % to IBM Watson
Published Temperature = 40 C Humidity = 51 % Soil Moisture = 86 % to IBM Watson
Published Temperature = 33 C Humidity = 21 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 29 C Humidity = 48 % Soil Moisture = 22 % to IBM Watson
Published Temperature = 45 C Humidity = 32 % Soil Moisture = 23 % to IBM Watson
Published Temperature = 98 C Humidity = 38 % Soil Moisture = 8 % to IBM Watson
Published Temperature = 44 C Humidity = 71 % Soil Moisture = 16 % to IBM Watson
Command received: motoron
Motor is ON
Published Temperature = 62 C Humidity = 2 % Soil Moisture = 34 % to IBM Watson
Published Temperature = 21 C Humidity = 14 % Soil Moisture = 82 % to IBM Watson
Published Temperature = 35 C Humidity = 2 % Soil Moisture = 5 % to IBM Watson
Published Temperature = 34 C Humidity = 78 % Soil Moisture = 44 % to IBM Watson
Command received: motoroff
Motor is OFF

Published Temperature = 93 C Humidity = 81 % Soil Moisture = 87 % to IBM Watson
Command received: motoron
Motor is ON
Command received: motoroff
Motor is OFF
Published Temperature = 54 C Humidity = 36 % Soil Moisture = 81 % to IBM Watson
Published Temperature = 56 C Humidity = 76 % Soil Moisture = 56 % to IBM Watson
Published Temperature = 70 C Humidity = 53 % Soil Moisture = 74 % to IBM Watson
Published Temperature = 58 C Humidity = 22 % Soil Moisture = 68 % to IBM Watson
Command received: motoron
Motor is ON
Published Temperature = 93 C Humidity = 34 % Soil Moisture = 11 % to IBM Watson
Command received: motoroff
Motor is OFF
Published Temperature = 86 C Humidity = 67 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 49 C Humidity = 70 % Soil Moisture = 61 % to IBM Watson
Published Temperature = 94 C Humidity = 48 % Soil Moisture = 77 % to IBM Watson
Published Temperature = 59 C Humidity = 6 % Soil Moisture = 11 % to IBM Watson
Published Temperature = 16 C Humidity = 6 % Soil Moisture = 41 % to IBM Watson
```

<u>Output:</u>

- **Development of Mobile application using MIT App Inventor:**

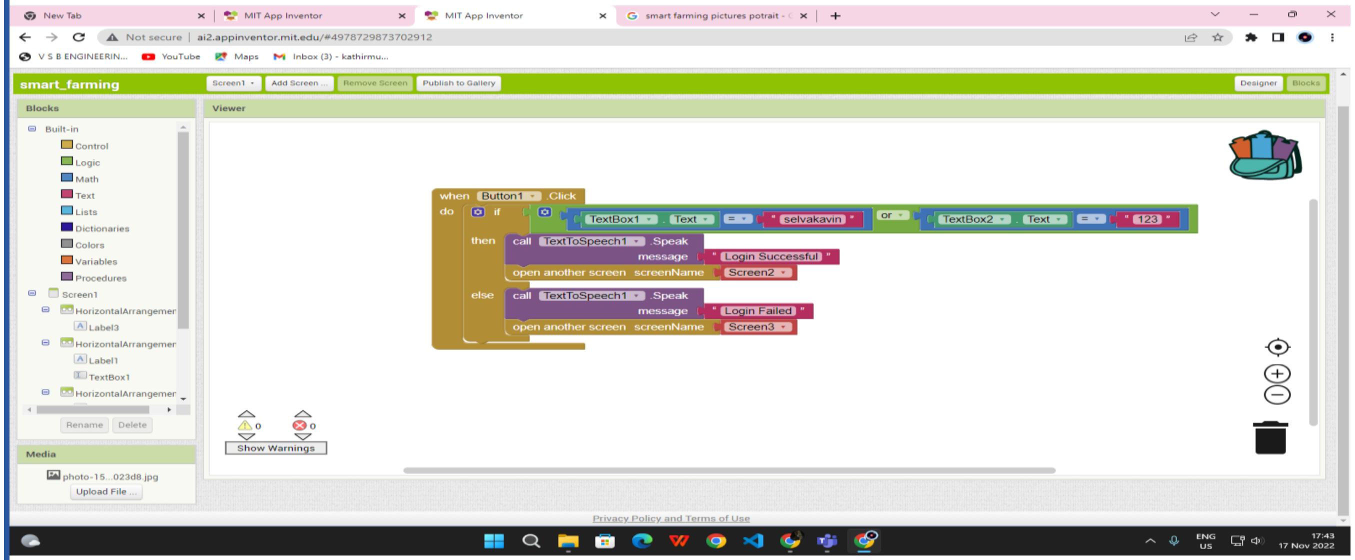
In the MIT App Inventor platform, an application is created which monitors the farmland parameters such as temperature, humidity, soil moisture and controls the actuators such as motors.

<u>**MIT App**</u>



<u>**Front End:**</u>

<u>**Backend:**</u>



```
when Button1 .Click
do  if  TextBox1 . Text = " selvakavin " or TextBox2 . Text = " 123 "
    then  call TextToSpeech1 .Speak
                        message " Login Successful "
          open another screen  screenName  Screen2
    else  call TextToSpeech1 .Speak
                        message " Login Failed "
          open another screen  screenName  Screen3
```

App working:

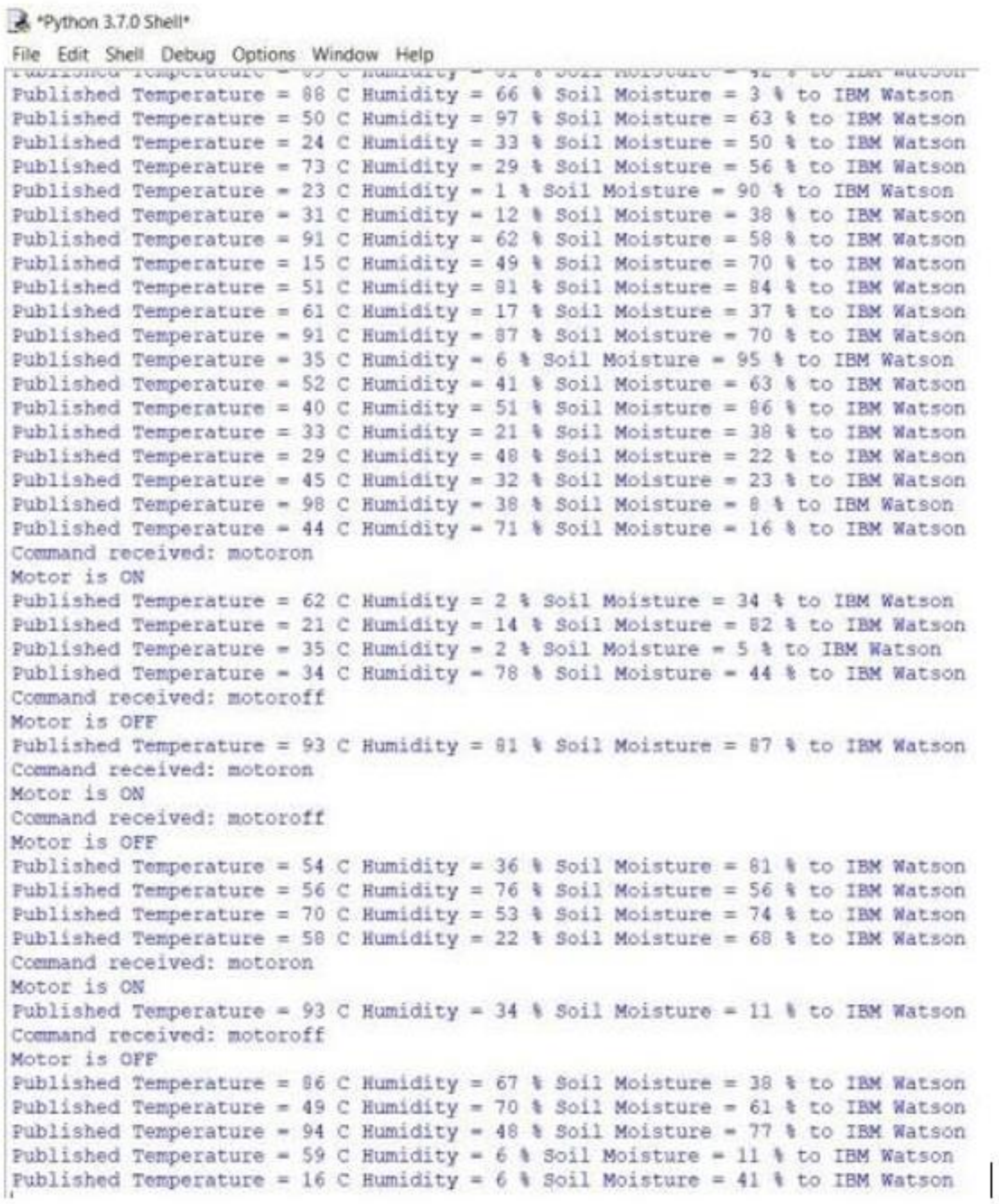
The app works based on HTTP protocol. The app uses HTTP GET method to parse the JSON data from the Node red website and displays the value in the UI. Using the HTTP POST method, the app sends command when a specific button is pressed. From where, the python code subscribes the command data from the cloud thereby notifying the command is received.

Python code:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "nckdv7"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "12345678" # Initialize GPIO
```

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

Output:



Python 3.7.0 Shell

```
Published Temperature = 88 C Humidity = 66 % Soil Moisture = 3 % to IBM Watson
Published Temperature = 50 C Humidity = 97 % Soil Moisture = 63 % to IBM Watson
Published Temperature = 24 C Humidity = 33 % Soil Moisture = 50 % to IBM Watson
Published Temperature = 73 C Humidity = 29 % Soil Moisture = 56 % to IBM Watson
Published Temperature = 23 C Humidity = 1 % Soil Moisture = 90 % to IBM Watson
Published Temperature = 31 C Humidity = 12 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 91 C Humidity = 62 % Soil Moisture = 58 % to IBM Watson
Published Temperature = 15 C Humidity = 49 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 51 C Humidity = 81 % Soil Moisture = 84 % to IBM Watson
Published Temperature = 61 C Humidity = 17 % Soil Moisture = 37 % to IBM Watson
Published Temperature = 91 C Humidity = 87 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 35 C Humidity = 6 % Soil Moisture = 95 % to IBM Watson
Published Temperature = 52 C Humidity = 41 % Soil Moisture = 63 % to IBM Watson
Published Temperature = 40 C Humidity = 51 % Soil Moisture = 86 % to IBM Watson
Published Temperature = 33 C Humidity = 21 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 29 C Humidity = 48 % Soil Moisture = 22 % to IBM Watson
Published Temperature = 45 C Humidity = 32 % Soil Moisture = 23 % to IBM Watson
Published Temperature = 98 C Humidity = 38 % Soil Moisture = 8 % to IBM Watson
Published Temperature = 44 C Humidity = 71 % Soil Moisture = 16 % to IBM Watson
Command received: motoron
Motor is ON
Published Temperature = 62 C Humidity = 2 % Soil Moisture = 34 % to IBM Watson
Published Temperature = 21 C Humidity = 14 % Soil Moisture = 82 % to IBM Watson
Published Temperature = 35 C Humidity = 2 % Soil Moisture = 5 % to IBM Watson
Published Temperature = 34 C Humidity = 78 % Soil Moisture = 44 % to IBM Watson
Command received: motoroff
Motor is OFF
Published Temperature = 93 C Humidity = 81 % Soil Moisture = 87 % to IBM Watson
Command received: motoron
Motor is ON
Command received: motoroff
Motor is OFF
Published Temperature = 54 C Humidity = 36 % Soil Moisture = 81 % to IBM Watson
Published Temperature = 56 C Humidity = 76 % Soil Moisture = 56 % to IBM Watson
Published Temperature = 70 C Humidity = 53 % Soil Moisture = 74 % to IBM Watson
Published Temperature = 58 C Humidity = 22 % Soil Moisture = 68 % to IBM Watson
Command received: motoron
Motor is ON
Published Temperature = 93 C Humidity = 34 % Soil Moisture = 11 % to IBM Watson
Command received: motoroff
Motor is OFF
Published Temperature = 86 C Humidity = 67 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 49 C Humidity = 70 % Soil Moisture = 61 % to IBM Watson
Published Temperature = 94 C Humidity = 48 % Soil Moisture = 77 % to IBM Watson
Published Temperature = 59 C Humidity = 6 % Soil Moisture = 11 % to IBM Watson
Published Temperature = 16 C Humidity = 6 % Soil Moisture = 41 % to IBM Watson
```

CHAPTER 8 - PERFORMANCE METRICS

S. No.	Name of the Phase	Tasks Performed	Performance Metrics
1.	Development of Problem Statement	The underlying problem analyzed and a rough idea of the solution was planned	The Problem statement was developed
2.	Ideation Phase	Extracting use and test cases	Empathy map, Ideation and Literature survey were formulated.
3.	Project Design Phase 1	Solution for the problem is formulated and architecture is designed	Problem solution fit was designed and the Proposed solution is finalized with the help of Solution architecture.
4.	Project Design Phase 2	In depth analysis of the solution is performed including requirements, tech stack, etc.	Solution Requirements, Overall Technology stack, Data flow diagrams, User stories were formulated.
5.	Project Planning Phase	Various sprints were designed as individual progressive steps.	Project Milestone and Sprint Plans were developed.

CHAPTER 9 - ADVANTAGES AND DISADVANTAGES

9.1 Advantages:

- By monitoring the soil parameters of the farm, the user can have a complete analysis of the field, in terms of numbers.
- Using the website and the application, an interactive experience can be achieved.
- As the data gets pushed to the cloud, one can access the data anywhere from this world. ○ Without human intervention, water pump can be controlled through the mobile application and it's flow can be customized using servo motors.
- By using Raspberry Pi MCU, scalability can be increased due to its high processing power and enough availability of GPIO pins

9.2 Disadvantages:

- Data transfer is through the internet. So data fetch and push might delay due to slow internet connection, depending on the location and other physical parameters.
- System can only monitor a certain area of the field. In order to sense and monitor an entire field, sensors should be placed in many places, which may increase the cost.
- Data accuracy may vary according to various physical parameters such as temperature, pressure, rain.
- Cost of the system is high due to usage of Raspberry Pi. ○ Rodent and insects may cause damage to the system.

CHAPTER 10 – CONCLUSION

The project thus monitors important parameters present in the field such as temperature, humidity, soil moisture etc., and controls important actuators such as motors etc. It is helpful for farmers to remotely monitor their fields even during adverse weather conditions and help them control farming equipments remotely using cloud.

CHAPTER 11 - FUTURE SCOPE

The project can be further extended by monitoring other parameters such as nutrient contents in the soil, soil texture etc. AI techniques integrated with cloud can be integrated to monitor any pest attacks present in the plant. The application can be made interactive which provides suggestions to farmers to improve their farmlands.

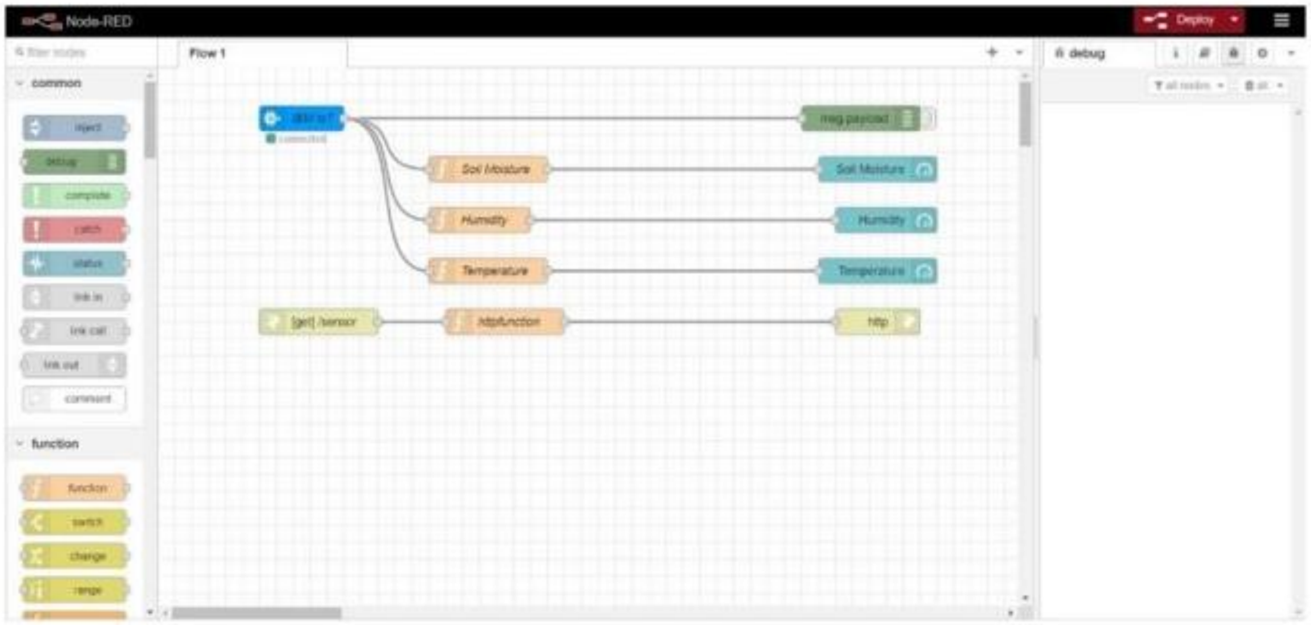
CHAPTER 12 – APPENDIX

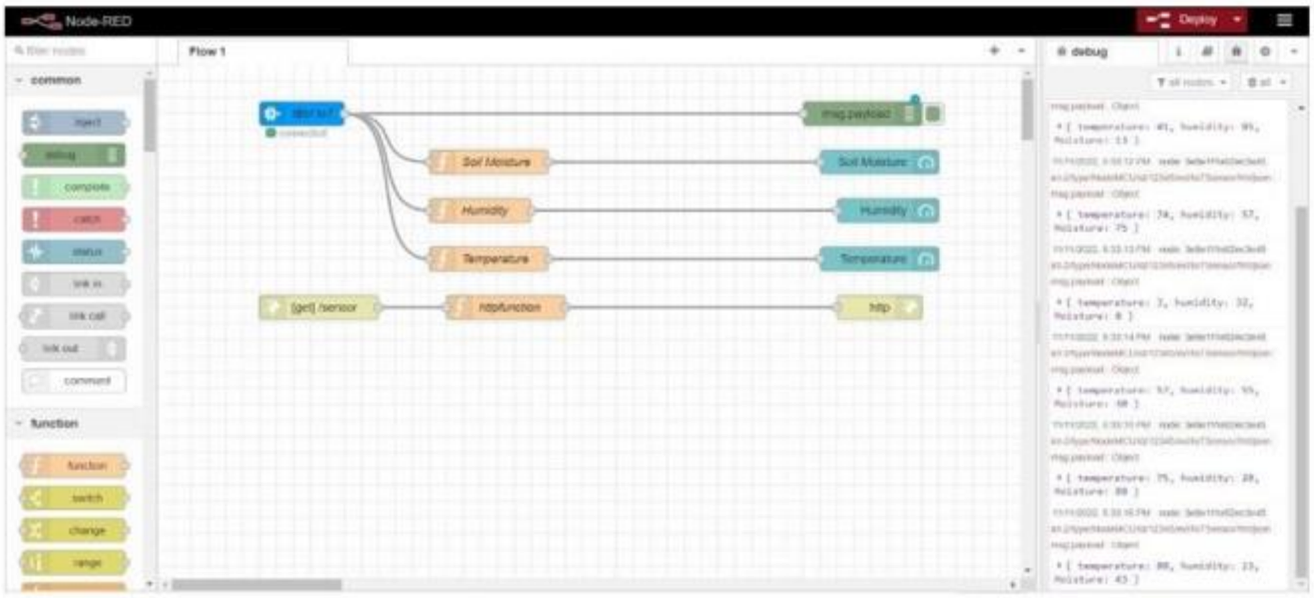
12.1 Source Code:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "nckdv7"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
```

```
authToken = "12345678"
# Initialize GPIO
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
status=cmd.data['command']
if status=="motoron":
print("Motor is ON")
else:
print("Motor is OFF")
#print(cmd)
try:
deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod,
"auth-token": authToken}
Page 40 of 43
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)
pulse=random.randint(0,100)
moisture= random.randint(0,100)
humidity=random.randint(0,100);
lat = 17
lon = 18
data = { 'temp' : temp, 'humidity' : humidity, 'Soil Moisture' : moisture}
#print data
def myOnPublishCallback():
print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % humidity, "Soil Moisture = %s %%"
% moisture,"to IBM Watson")
```

Node Red Service Creation:





```
msg.payload : Object
▸{ temperature: 41, humidity: 95, Moisture: 13 }
[illegible]
msg.payload : Object
▸{ temperature: 74, humidity: 57, Moisture: 75 }
[illegible]
msg.payload : Object
▸{ temperature: 3, humidity: 32, Moisture: 8 }
[illegible]
msg.payload : Object
▸{ temperature: 57, humidity: 55, Moisture: 68 }
[illegible]
msg.payload : Object
▸{ temperature: 75, humidity: 28, Moisture: 88 }
[illegible]
msg.payload : Object
▸{ temperature: 88, humidity: 23, Moisture: 43 }
```

Code block for the function palette:

1) Soil moisture:

```
Soil = msg.payload.Moisture
msg.payload = "Soil Moisture : "
global.set('m',Soil)
msg.payload = Math.round(Soil)
return msg;
```

2) Humidity:

```
Humidity = msg.payload.humidity
msg.payload = "Humidity : "
global.set('h',Humidity)
msg.payload = Math.round(Humidity )
return msg;
```

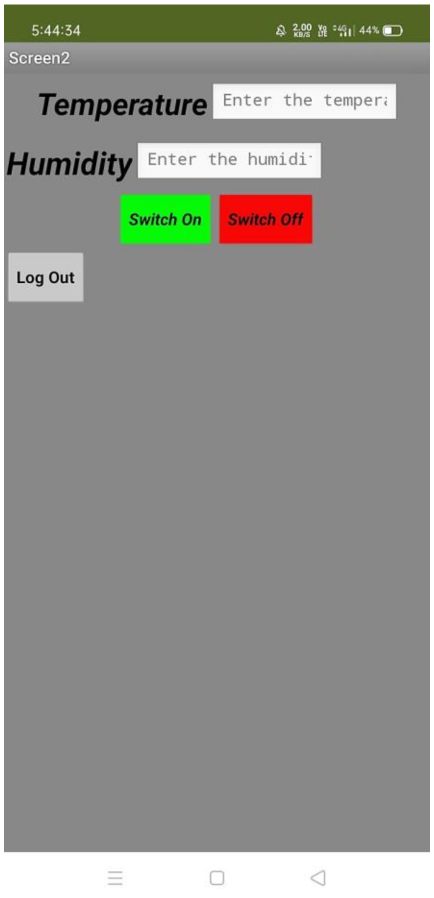
3) Temperature:

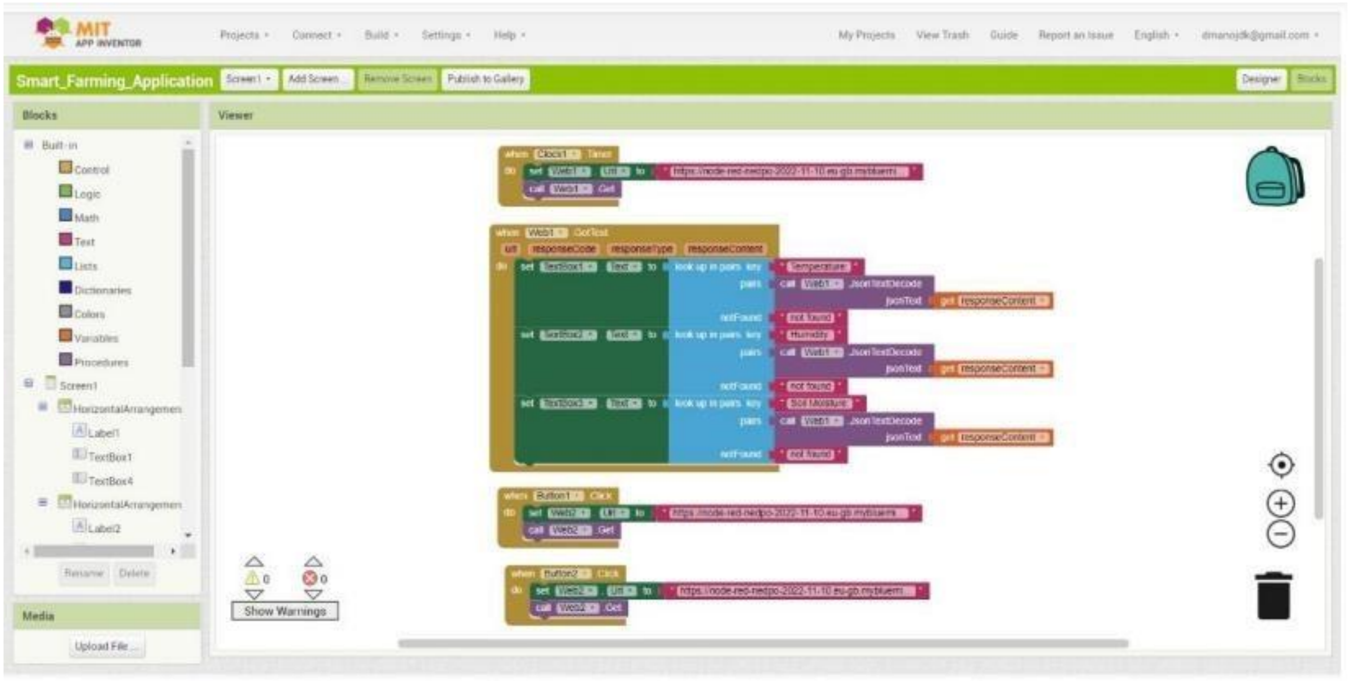
```
Temperature = msg.payload.temperature
msg.payload = "Temperature : "
global.set('t',Temperature)
msg.payload =Math.round(Temperature)
return msg;
```

4) HTTP Function:

```
msg.payload = {"Temperature:": global.get('t'),"Humidity:": global.get('h'),"Soil
Moisture:": global.get('m')}
return msg;
```

MIT App Front End:





```
when Clock1 . Timer
do  set Web1 . Url to "https://node-red-nedpo-2022-11-10.eu-gb.mybluemi..."
    call Web1 . Get

when Web1 . GotText
url  responseCode  responseType  responseContent
do  set TextBox1 . Text to  look up in pairs  key "Temperature"
                                              pairs  call Web1 . JsonTextDecode
                                                          jsonText  get responseContent
                                              notFound "not found"
    set TextBox2 . Text to  look up in pairs  key "Humidity"
                                              pairs  call Web1 . JsonTextDecode
                                                          jsonText  get responseContent
                                              notFound "not found"
    set TextBox3 . Text to  look up in pairs  key "Soil Moisture"
                                              pairs  call Web1 . JsonTextDecode
                                                          jsonText  get responseContent
                                              notFound "not found"

when Button1 . Click
do  set Web2 . Url to "https://node-red-nedpo-2022-11-10.eu-gb.mybluemi..."
    call Web2 . Get

when Button2 . Click
do  set Web2 . Url to "https://node-red-nedpo-2022-11-10.eu-gb.mybluemi..."
    call Web2 . Get
```

12.2 GitHub and Project Demo Link:

GitHub: https://github.com/IBM-EPBL/IBM-Project-33301-1660218261

ProjectDemo:**Link:https://drive.google.com/file/d/1O95EZsaBMfCV90EC08q9IszYW4JP8cGF/view?usp=drivesdk**