

# **SMARTFARMER - IOT ENABLED SMART FARMING APPLICATION**

**IBM PROJECT REPORT  
TEAM ID : PNT2022TMID30597**

*Submitted by*

**SUBITHRA R - 613019104083**

**SWARNASHRI S - 613019104086**

**SNEGA M - 613019104075**

**TAMILSELVI N - 610919104062**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

**VIVEKANANDHA COLLEGE OF TECHNOLOGY FOR**

**WOMEN,TIRUCHENGODE,NAMAKKAL**

**ANNA UNIVERSITY : CHENNAI – 600 025**

## TABLE OF CONTENTS

CHAPTER	CHAPTER NAME	PG.NO
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Project Overview	1
	1.2 Purpose	1
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>3</b>
	2.1 Existing Problem	3
	2.2 References	4
	2.3 Problem Statement Definition	4
<b>3</b>	<b>IDEATION AND PROPOSED SOLUTION</b>	<b>5</b>
	3.1 Empathy Map Canvas	5
	3.2 Ideation & Brainstorming	6
	3.3 Proposed Solution	10
	3.4 Problem Solution Fit	10
<b>4</b>	<b>REQUIRMENTS ANALYSIS</b>	<b>11</b>
	4.1 Functional Requirements	11
	4.2 Non - Functional Requirements	11
<b>5</b>	<b>PROJECT DESIGN</b>	<b>12</b>
	5.1 Data Flow Diagram	12
	5.2 Solution & Technical Architecture	12
	5.3 User Stories	14
<b>6</b>	<b>PROJECT PLANNING &amp; SCHEDULING</b>	<b>15</b>
	6.1 Sprint Planning & Estimation	15
	6.2 Sprint Delivery Schedule	16

	6.3 Report from JIRA	17
<b>7</b>	<b>CODING &amp; SOLUTIONING</b>	<b>18</b>
	7.1 Features	18
<b>8</b>	<b>TESTING</b>	<b>36</b>
	8.1 Testing	36
	8.2 Integration Testing	36
	8.3 Test Cases	37
<b>9</b>	<b>RESULTS</b>	<b>38</b>
<b>10</b>	<b>ADVANTAGES &amp; DISADVANTAGES</b>	<b>40</b>
<b>11</b>	<b>CONCLUTION</b>	<b>41</b>
<b>12</b>	<b>FUTURE SCOPE</b>	<b>42</b>
<b>13</b>	<b>APPENDIX</b>	<b>43</b>
	13.1 Source Code	43
	13.2 GitHub and Project Demo Link	43

# 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.

### 1.1 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. This need is anticipated to rise in the coming years due to population growth. 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 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.

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. 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.

## 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 are developed, implemented and tested. There is a need for an 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 used for automation purpose.
2.	IoT Based Smart Irrigation Monitoring And Controlling System	Shweta B. Saraf et.al, 15 January 2018	In this paper, the proposed system is based on IoT that uses real-time input data. Smart farm irrigation system uses an android phone for remote monitoring and controlling of drips through a 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 a web-based java graphical user interface.

3	Smart waste collection monitoring and alert system via IoT	<u>Zainal Hisham</u> <u>Che Soh et.al, 24</u> June 2019	<p>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.</p>
---	--	---	---

## 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

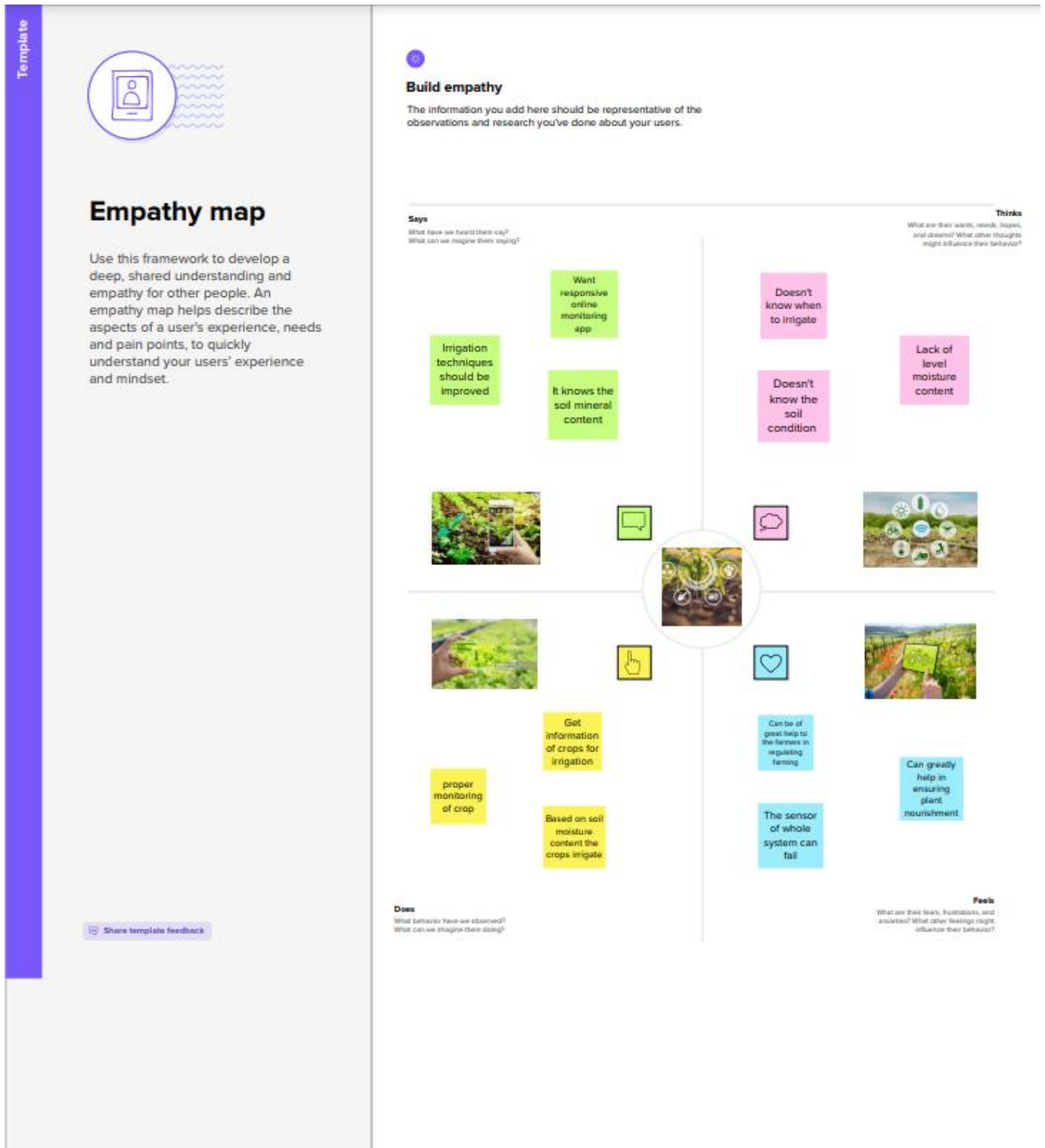
## 2.3 Problem Statement Definition

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. Ideally, Each field should get just the right amount of water at just the right time. Most of the rural area people can't implement the IOT devices because of they don't know about the device to use. 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  
3 people recommended



Show 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

#### 1 Team gathering

Define who should participate in the session and send an invite. Share relevant information as pre-work ahead.

#### 2 Set the goal

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

#### 3 Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

Open article

#### PROBLEM STATEMENT

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 ideally. Each field should get just the right amount of water at just the right time. Most of the rural area people can't implement the IoT devices because of they don't know about the device to use it.

## Step-2: Brainstorm, Idea Listing and Grouping

2

### Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

**TIP**  
You can select a sticky note and hit the pencil (which is visible) icon to start drawing!

#### SUBITHRA.R

It control various parameter on thier field

By educating and creating awarness to farmers about iot device

It can show the fertility of the land.It so farmer can used the require fertilizer

Monitoring climate conditions

#### SWARNASHRILS

It can notify the problems that accour in the feild

Suitable for all types of soil

Real-time Analysis of Soil Demand

Soil Testing & its Quality.

#### TAMIL SELVI.N

It will automatically monitor

It Provides a Source of Income in Rural Areas

Intelligent Irrigation System

Easy to operate and maintenance

#### SNEGAM

Greenhouses automation

Reduction of risk

Saving the cost of labour

Predictive Analysis for Crops

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

It Provides a Source of Income in Rural Areas

Predictive Analysis for Crops

Intelligent Irrigation System

Soil Testing & its Quality.

Monitoring climate conditions

Suitable for all types of soil

Real-time Analysis of Soil Demand

Intelligent Irrigation System



It can show the fertility of the land.It so farmer can used the require fertilizer

By educating and creating awarness to farmers about iot device

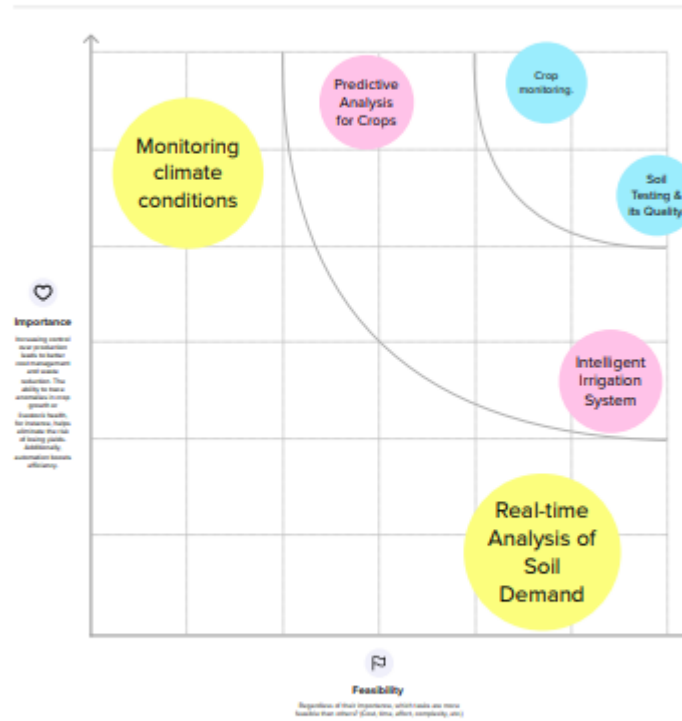
### Step-3: Idea Prioritization

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



### 3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<p>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</p> <p>Soil erosion and soil nutrient loss, product satisfaction, customers' adaptation to climate, usage of harmful fertilizer and manure and pesticides are major problems faced by farmers</p> <p>For taking agriculture to the next generation, only fresh water can be used for farming. Farmers are more affected and annoyed by the above factors.</p>
2.	Idea / Solution description	<p>The IoT-based agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors. Farmers can monitor all the sensor parameters by using a web or mobile application even if the farmer is not near his field. Watering the crop is one of the important tasks for the farmers. They can make the decision whether to water the crop or postpone it by monitoring the sensor parameters and controlling the motor pumps from the mobile application itself.</p>
3.	Novelty / Uniqueness	<p>Building a community of farmers around the globe will definitely do great, where they can find lots and lots of information about crops and plantations directly from the other farmers and the agricultural experts around the globe.</p>

4.	Social Impact / Customer Satisfaction	The good thing is that IoT instructs or alerts them to do the amount of work at the right time. So that the yielding will be more and good, it also reduces the attention and time required to the field, which makes the Customer Satisfaction. On the other hand the main disadvantage is that IoT reduces the number of laborers and their wages, as a result many people may lose their work.
5.	Business Model (Revenue Model)	Many agricultural products like fertilizers, pesticides, manure, and field equipment can also be promoted in the form of ads. A small amount of subscription fee can also be collected from the farmers.
6.	Scalability of the Solution	The scalability of the above proposed solution is not limited. Here a lot of sensors and analyzing tools and algorithms can be integrated to provide the best experience.

## 3.4 Problem Solution Fit

<p><b>Define CS, fit into CC</b></p> <p><b>1. CUSTOMER SEGMENTS</b> Who is your customer? i.e. working parents of 0-5 y.o.</p> <p><b>The customer for this product is a farmer who grows crops. Our goal is to help them, monitor field parameters remotely. This product saves agriculture from extinction.</b></p>	<p><b>6. CUSTOMER</b></p> <p>What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.</p> <p><b>The Crops are being destroyed because of not getting the exact data about the crops. It consumes high data storage</b></p>	<p><b>5. AVAILABLE SOLUTIONS</b></p> <p>Which solutions are available to the customers when they face the problem?</p> <p>or need to get the job done? What have they tried in the past? What pros &amp; cons do these solutions have? i.e. pen and</p> <p><b>Automated irrigation using IoT. The Meteorological data and field parameters are being collected and processed and those data's are automatically updated to the device.</b></p> <p><b>Explore AS, differentiate</b></p>
<p><b>Focus on J&amp;P, tap into BE, understand RC</b></p> <p><b>2. JOBS-TO-BE-DONE / PROBLEMS</b></p> <p>Which jobs-to-be-done (or problems) do you address for your customers? there could be more than one; explore different sides.</p> <p><b>The main objective of this project is to obtain a automated device which stores the data of crops and the use sensors to acquire various field parameters and process them using a central processing system. The cloud is used to store and transmit data using</b></p>	<p><b>9. PROBLEM ROOT CAUSE</b></p> <p>What is the real reason that this problem exists? What is the back story behind the need to do this job?</p> <p><b>The Problem is raised due to Frequent change in climate and unpredictable weather and climate made it difficult for farmers to engage in agriculture. These factors play an important role in deciding whether to water your plants. Fields are difficult to monitor when the farmer is not at the field, leading to crop damage.</b></p>	<p><b>7. BEHAVIOUR</b></p> <p>What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)</p> <p><b>It is best to use a proper drainage system to overcome the effects of excess water from heavy rain. Use of hybrid plants that are resistant to pests.</b></p> <p><b>Focus on J&amp;P, tap into BE, understand RC</b></p>

## 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	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Sensor Function for framing System	Measure the Temperature and Humidity Measure the Soil Monitoring Check the cropdiseases
FR-4	Manage Modules	Manage Roles of User Manage User permission
FR-5	Check whether details	Temperature details Humidity details
FR-6	Data Management	Manage the data of weather conditions Manage the data of crop conditions Manage the data of live stock conditions

## Non-functional Requirements:

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

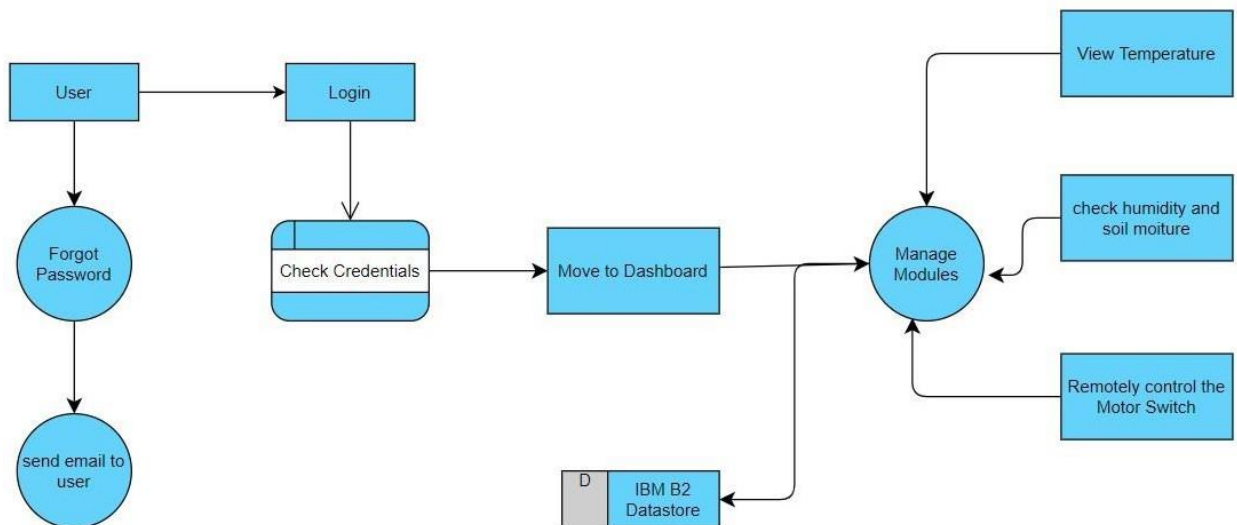
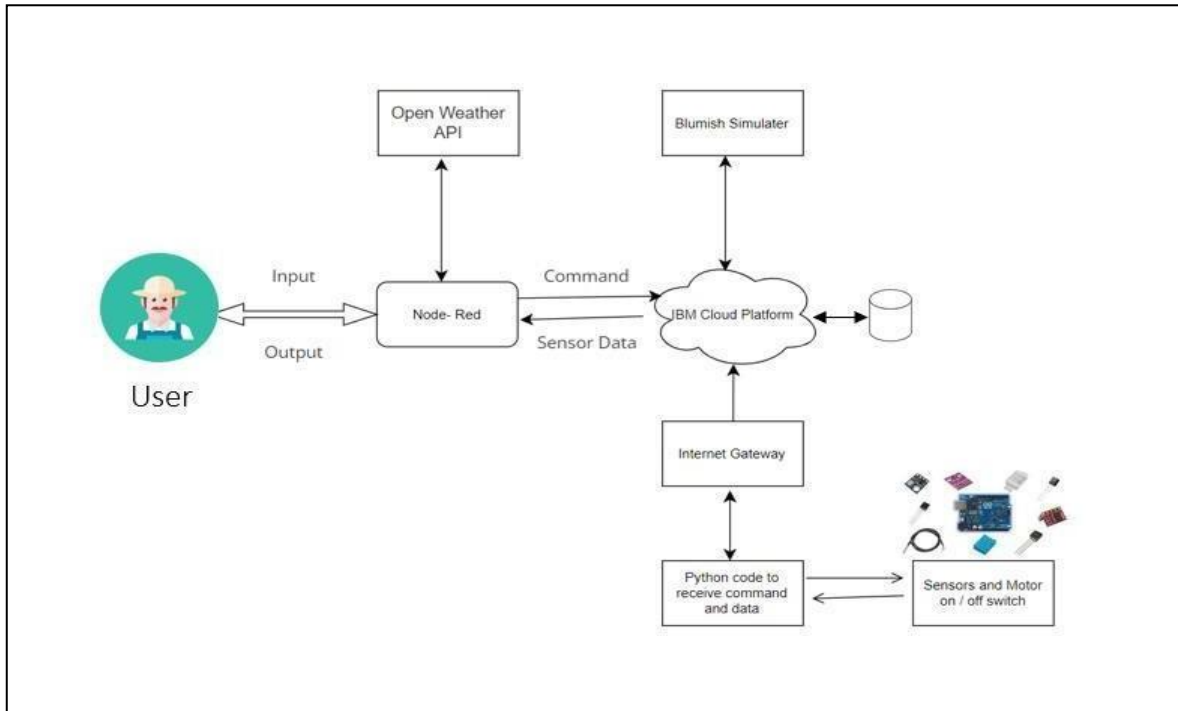
S No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	User friendly guidelines for users to avail the features. Most simplistic user interface for ease of use and it includes easy learn ability, efficiency in use, remember ability, lack of errors in operation and subjective pleasure
NFR-2	<b>Security</b>	All the details about the user are protected from unauthorized access. Detection and identification of any misfunctions of sensors. Sensitive and private data must be protected from their production until the decision-making and storage stages
NFR-3	<b>Reliability</b>	Implementing Mesh IoT Networks Building a Multi-layered defence for IoT Networks and This model uses dedicated and shared protection schemes to avoid farm service outages
NFR-4	<b>Performance</b>	The use of modern technology solutions helps to achieve the maximum performances thus resulting in better quality and quantity yields. the idea of implementing integrated sensors with sensing soil and environmental in farming will be more efficient for overall monitoring.
NFR-5	<b>Availability</b>	This app is available for all platforms Then Automatic adjustment of farming equipment made possible by linking information like crops/weather and equipment to auto-adjust temperature, humidity
NFR-6	<b>Scalability</b>	Scalability refers to the ability to increase available resources and system capability without the need to go through a major system redesign or implementation.



# CHAPTER 5

## PROJECT DESIGN

### 5.1 Data Flow Diagram



## 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.

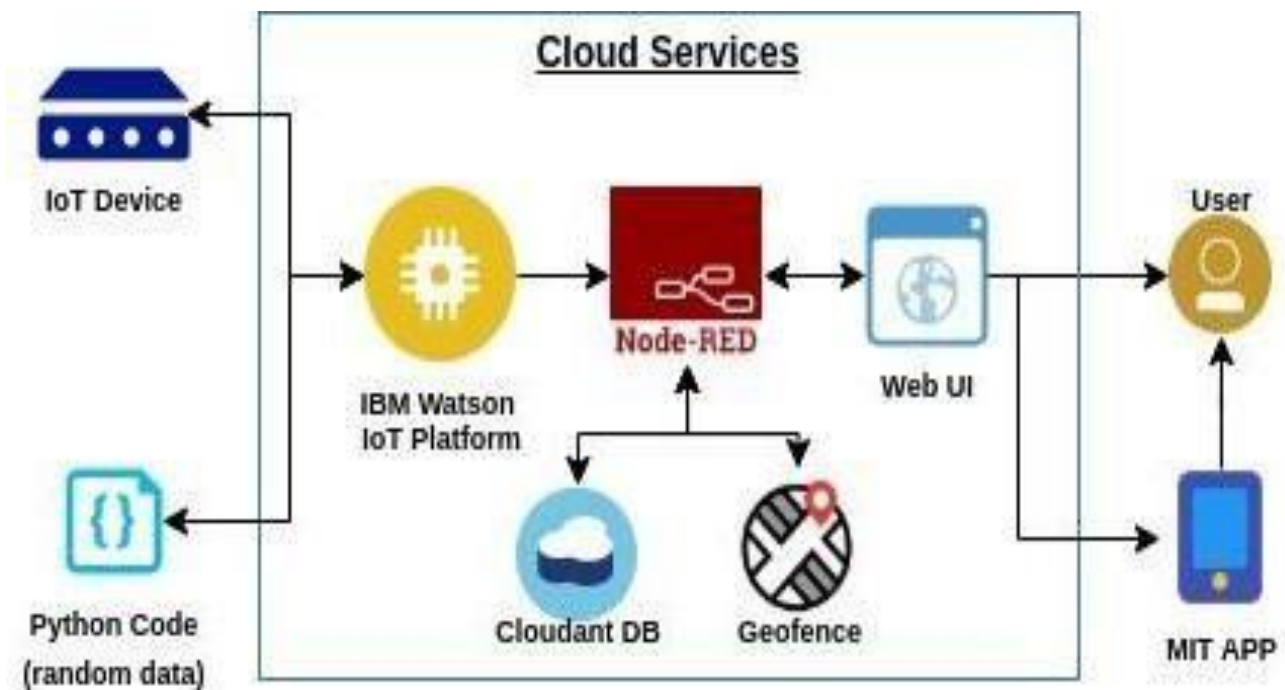


Figure 1: Architecture and data flow of the smart farming application

**Table-1 : Components & Technologies:**

Description	Technology
How user interacts with applicatione.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript
Logic for a process in the application	Python
Logic for a process in the application	IBM Watson service
Logic for a process in the application	IBM Watson Assistant
Data Type, Configurations etc.	MySQL
Database Service on Cloud	IBM DB2, IBM Cloudant etc.
File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
Purpose of External API used in the application	IBM Weather API, etc.
Application Deployment on LocalSystem / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry,Kubernetes, etc. <b>Table-2: Application Characteristics:</b>

**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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Custoer (Mobile user)	Registration	USN-1	As a user, I can register for the applicationby 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 confirmationemail 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 Gmail		Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password		High	Sprint-1
Customer (Web user)	Dashboard	USN-5	As a User can view the dashboard, and this dashboard include the check roles of access and then move to the manage modules.	I can view the dashboardin this smart farming application system.	High	Sprint 2
		USN-6	User can remotely access the motor switch	In the smart farming app	High	Sprint 3
Administrat or			As a user once view the manage modules this describes the Manage system Admins and Manage Roles of User			Sprint 2

## 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

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

## 6.2 Sprint Delivery Schedule

### Product Backlog, Sprint Schedule, and Estimation:

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story /Task</b>	<b>Story Points</b>	<b>Priority</b>	<b>Team Member</b>
<b>Sprint-1</b>	Registration (Farmer Mobile User)	UNS-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Subithra.R (Leader)
<b>Sprint-1</b>	Login	UNS-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Swarna shri (Member 1)

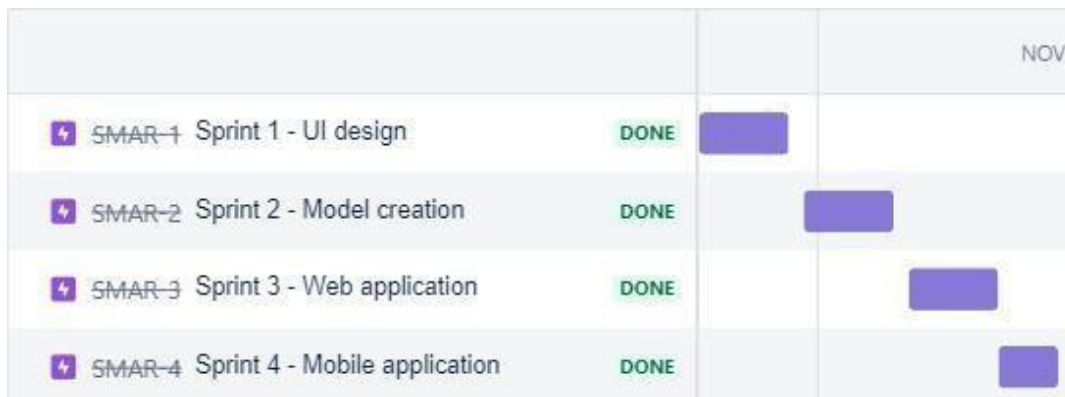
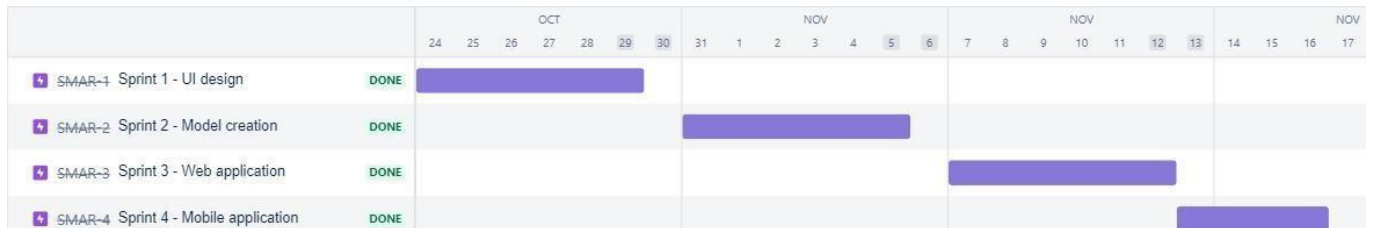
<b>Sprint-2</b>	User Interface	UN S-3	As a user, I can register for the application through Facebook	3	Low	Tamil selvi . N (Member 2)
<b>Sprint-1</b>	Data Visualization	UN S-4	As a user, I can register for the application through GMAIL	2	Medium	Snega . m (Member 3)
<b>Sprint-3</b>	Registration(Farmer Web User)	US N - 1	As a user, I can log into the application by entering email and password	3	High	Subithra.R (leader)
<b>Sprint 2</b>	Login	US N - 2	As a registered user, I need to easily login log into my registered account via the web page in minimum time	3	High	Swarna shri
<b>Sprint 4</b>	Web UI	US N - 3	As a user, I need to have a friendly user interface to easily view and access the resources	3	Medium	Tamil selvi .N



<b>Sprint - 1</b>	Registration (Chemical Manufacturer - Web user)	US N - 1	As a new user, I want to first register using my organization email and create a password for the account.	2	High	Snega.M
<b>Sprint - 4</b>	Login	USN - 2	As a registered user, I need to easily log in using the registered account via the web page.	3	High	Subithra. R
<b>Sprint - 3</b>	Web UI	USN - 3	As a user, I need to have a user-friendly interface to easily view and access the resources.	3	Medium	Swarna shri
<b>Sprint - 1</b>	Registration (Chemical Manufacturer - Mobile User)	USN - 1	As a user, I want to first register using my email and create a password for the account.	1	High	Tamil Selvi . N

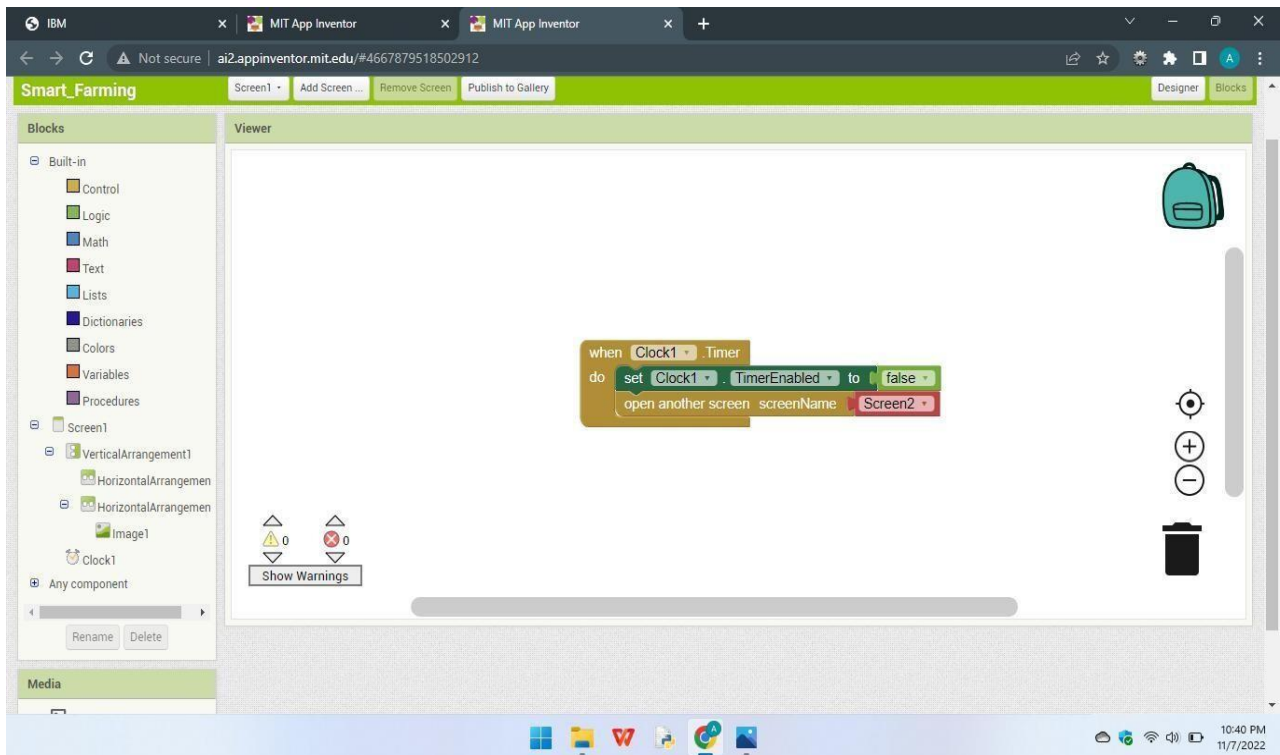
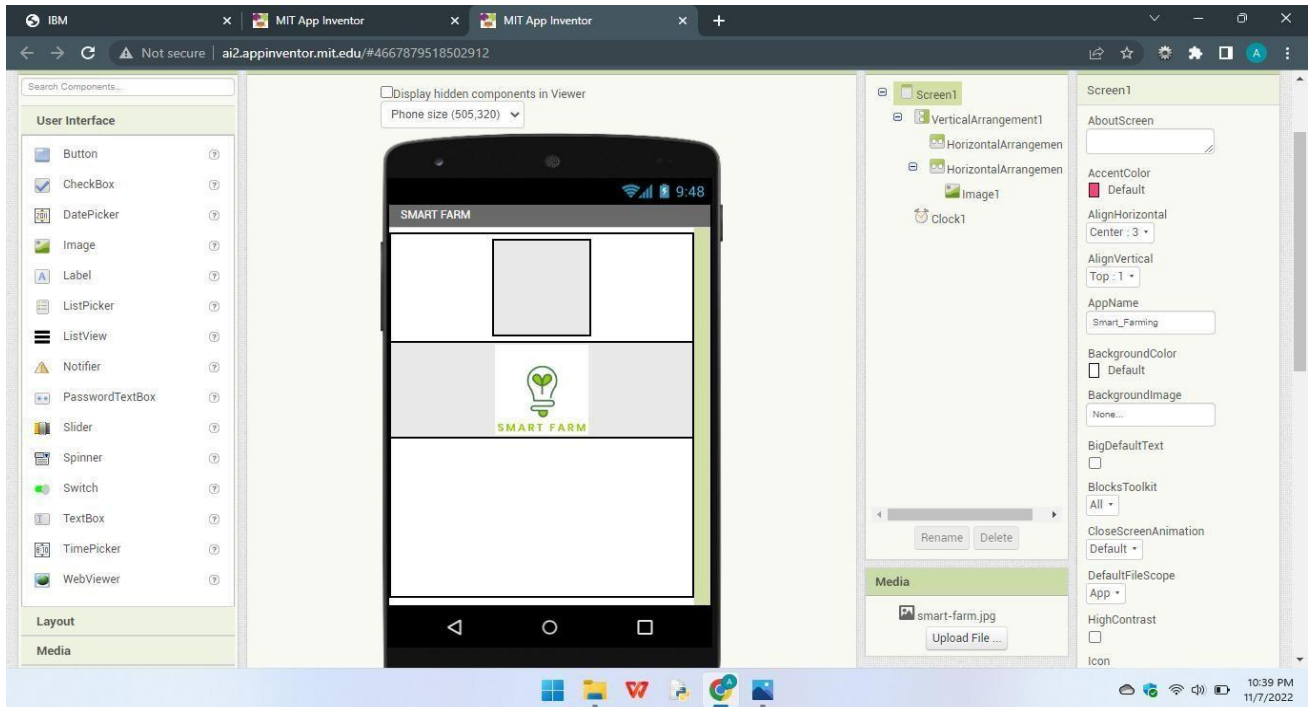
<b>Sprint - 1</b>	Login	USN - 2	As a registered user, I need to easily log in to the application.	2	Low	Snega . M
-------------------	-------	------------	---	---	-----	-----------

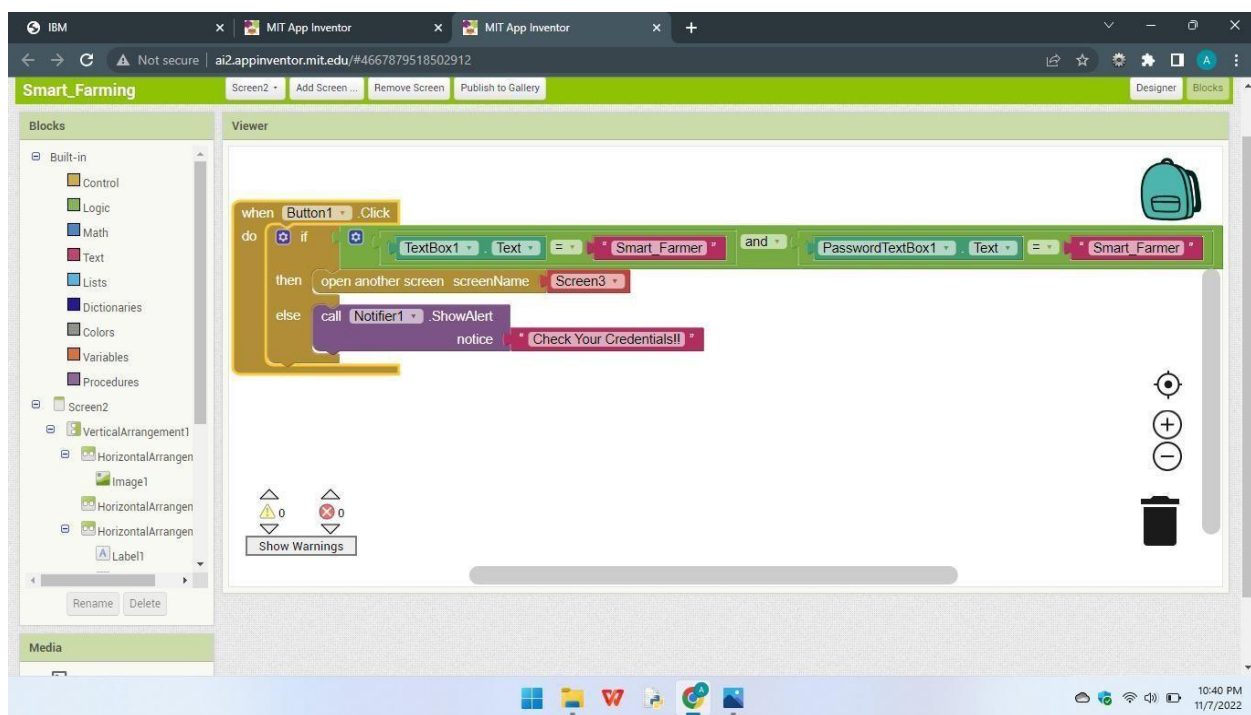
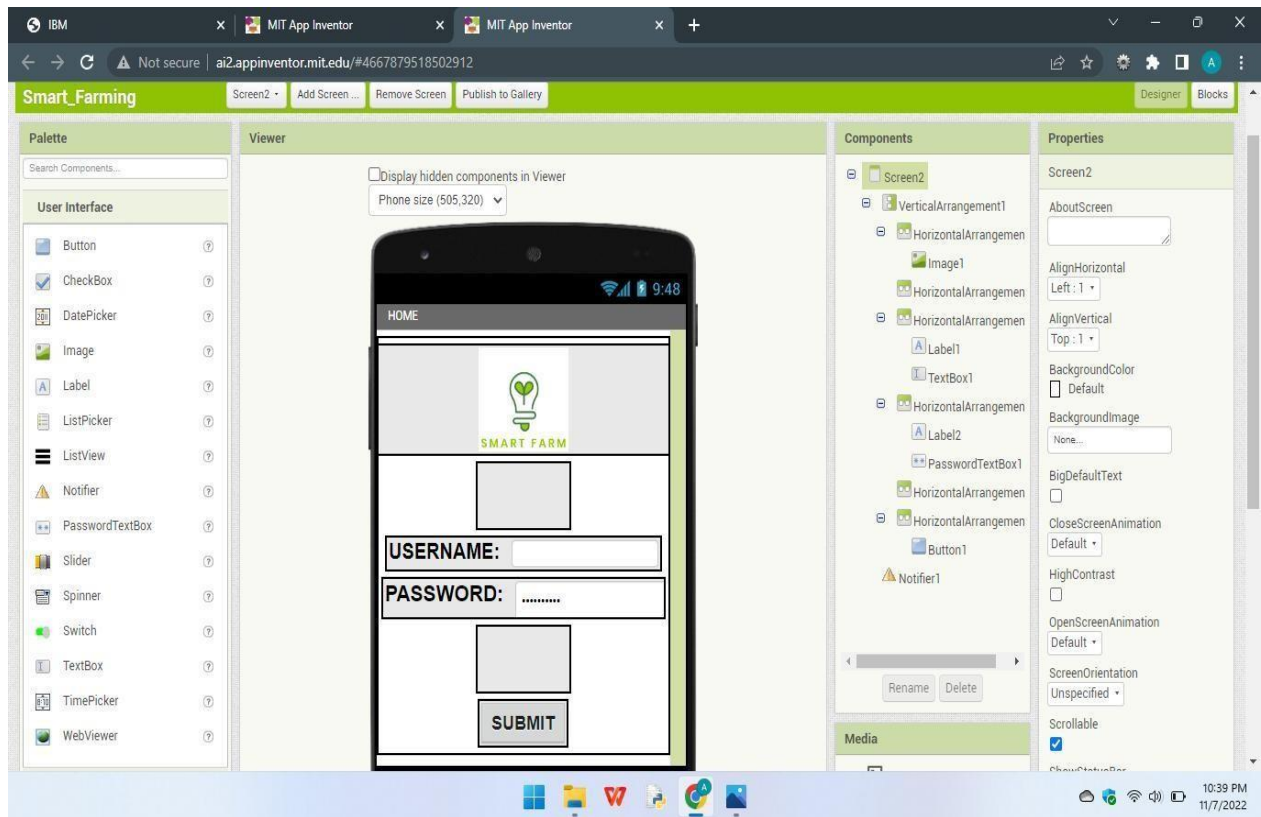
## 6.3 Reports from JIRA

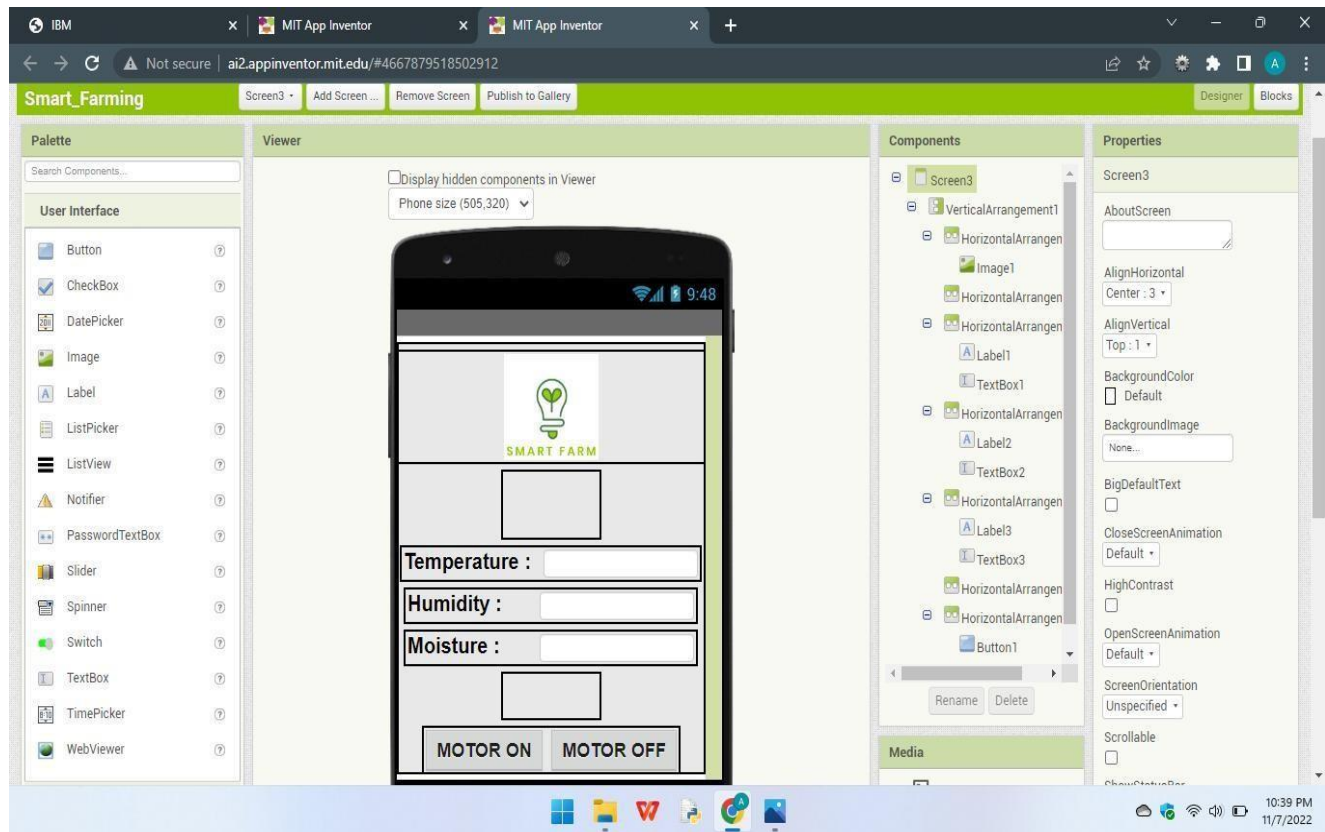


## CHAPTER 7 CODING & SOLUTIONING

### 7.1 Features - Development of Sprint-1 (User Interface Design)









USERNAME: Smart\_Farmer

PASSWORD: .....

SUBMIT



Temperature :

Humidity :

Moisture :

MOTOR ON

MOTOR OFF

## Testing - Compatibility Testing :

- Compatibility Testing is a type of Software testing to check whether your software is capable of running on different hardware, operating systems, applications, network environments or Mobile devices.
- Compatibility testing is a type of non-functional testing.
- Herewith, our test report is attached with log and output.

## Test Report for Sprint 1 :

PROJECT NAME : SMART FARMER – IOT ENABLED SMART FARMING APPLICATION
---

TEAM ID	PNT2022TMID30597
MODULE DESCRIPTION	Here we tested the compatibility of our Registration, Login and Dashboard Module with high Authentication
TYPE	Testing/verification



## Development of Sprint-2 (Python Code for Publish & Subscribe)

```
Smart_Farming.py - C:\Users\A S ABISHEK\AppData\Local\Programs\Python\Python37\Smart_Farming.py (3.7.0)
File Edit Format Run Options Window Help

import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig={
    "identity":{
        "orgId":"uq23sr",
        "typeId":"Smart_Farming",
        "deviceId":"32826"
    },
    "auth": {
        "token":"3wNLT001g8VpEJEpsq"
    }
}
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()
```

```
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\A S ABISHEK\AppData\Local\Programs\Python\Python37\Smart_Farming.py
2022-11-06 01:11:23,500 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:uq23sr:Smart_Farming:32826Published
data successfully: %s
{'soil_moisture': 83, 'temperature': 29, 'humidity': 36}
Published data successfully: %s {'soil_moisture': 47, 'temperature': 96, 'humidity': 25}
Published data successfully: %s {'soil_moisture': 45, 'temperature': 5, 'humidity': 49}
Published data successfully: %s {'soil_moisture': 82, 'temperature': 29, 'humidity': 80}
Published data successfully: %s {'soil_moisture': 92, 'temperature': 49, 'humidity': 86}
Published data successfully: %s {'soil_moisture': 60, 'temperature': 11, 'humidity': 98}
Published data successfully: %s {'soil_moisture': 40, 'temperature': -18, 'humidity': 100}
Published data successfully: %s {'soil_moisture': 17, 'temperature': 80, 'humidity': 33}
Published data successfully: %s {'soil_moisture': 60, 'temperature': 93, 'humidity': 95}
Published data successfully: %s {'soil_moisture': 22, 'temperature': 15, 'humidity': 70}
Published data successfully: %s {'soil_moisture': 78, 'temperature': 72, 'humidity': 15}
Published data successfully: %s {'soil_moisture': 42, 'temperature': 41, 'humidity': 63}
Published data successfully: %s {'soil_moisture': 21, 'temperature': 61, 'humidity': 50}
Published data successfully: %s {'soil_moisture': 36, 'temperature': 95, 'humidity': 56}
Published data successfully: %s {'soil_moisture': 14, 'temperature': 102, 'humidity': 74}
Published data successfully: %s {'soil_moisture': 34, 'temperature': 31, 'humidity': 44}
Published data successfully: %s {'soil_moisture': 95, 'temperature': 1, 'humidity': 37}
Published data successfully: %s {'soil_moisture': 58, 'temperature': 61, 'humidity': 97}
Published data successfully: %s {'soil_moisture': 94, 'temperature': 86, 'humidity': 14}
Published data successfully: %s {'soil_moisture': 56, 'temperature': 89, 'humidity': 82}
Published data successfully: %s {'soil_moisture': 100, 'temperature': -12, 'humidity': 61}
Published data successfully: %s {'soil_moisture': 82, 'temperature': 63, 'humidity': 41}
Published data successfully: %s {'soil_moisture': 30, 'temperature': 53, 'humidity': 84}
Published data successfully: %s {'soil_moisture': 42, 'temperature': 83, 'humidity': 71}
Published data successfully: %s {'soil_moisture': 80, 'temperature': 107, 'humidity': 38}
Published data successfully: %s {'soil_moisture': 64, 'temperature': 14, 'humidity': 73}
Published data successfully: %s {'soil_moisture': 95, 'temperature': 0, 'humidity': 100}
Published data successfully: %s {'soil_moisture': 67, 'temperature': 124, 'humidity': 21}
Published data successfully: %s {'soil_moisture': 51, 'temperature': 109, 'humidity': 5}
Published data successfully: %s {'soil_moisture': 25, 'temperature': 35, 'humidity': 58}
```

Browse

Action

Device Types

Interfaces

Add Device

32826

Connected

Smart\_Farming

Device

Oct 28, 2022 11:29 PM

Identity

Device Information

Recent Events

State

Logs

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

Event	Value	Format	Last Received
status	{"soil_moisture":89,"temperature":80,"humidity...	json	a few seconds ago
status	{"soil_moisture":44,"temperature":93,"humidity...	json	a few seconds ago
status	{"soil_moisture":34,"temperature":59,"humidity...	json	a few seconds ago
status	{"soil_moisture":18,"temperature":76,"humidity...	json	a few seconds ago
status	{"soil_moisture":71,"temperature":123,"humidut...	json	a few seconds ago

0 Simulations running

1:13 AM

11/6/2022

## Development of Sprint-3 (Web Application Using Node-Red)

```
Smart_Farming.py - C:\Users\A S ABISHEK\AppData\Local\Programs\Python\Python37\Smart_Farming.py (3.7.0)
File Edit Format Run Options Window Help

import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig={
    "identity":{
        "orgId":"ug23sr",
        "typeId":"Smart_Farming",
        "deviceId":"32826"
    },
    "auth": {
        "token":"3wNLT00lg8VpEJEpsq"
    }
}
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("\n")
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",myData)
    time.sleep(2)
    client.commandCallback=myCommandCallback
client.disconnect()
```

```
"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\A S ABISHEK\AppData\Local\Programs\Python\Python37\Smart_Farming.py
2022-11-10 16:04:42,463 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:ug23sr:Smart_Farming:32826Published
data successfully
{'soil_moisture': 37, 'temperature': 1, 'humidity': 35}
Published data successfully {'soil_moisture': 89, 'temperature': 94, 'humidity': 24}
Published data successfully {'soil_moisture': 57, 'temperature': 28, 'humidity': 90}
Published data successfully {'soil_moisture': 65, 'temperature': -18, 'humidity': 4}
Published data successfully {'soil_moisture': 87, 'temperature': 81, 'humidity': 92}
Published data successfully {'soil_moisture': 62, 'temperature': -16, 'humidity': 33}
Published data successfully {'soil_moisture': 99, 'temperature': 105, 'humidity': 62}
Published data successfully {'soil_moisture': 41, 'temperature': 114, 'humidity': 78}
Published data successfully {'soil_moisture': 26, 'temperature': -15, 'humidity': 49}
Published data successfully {'soil_moisture': 55, 'temperature': 84, 'humidity': 87}
```

Service Details - IBM Cloud x IBM Watson IoT Platform x +

y13urg.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

revisub73@gmail.com  
ID: y13urg

Browse Action Device Types Interfaces

Add Device +

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator ☐

<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added
> <input type="checkbox"/>	12345	Disconnected	Arduino	Device	Nov 3, 2022 2:15 PM
> <input type="checkbox"/>	Arduino_1	Disconnected	Arduin	Device	Nov 3, 2022 2:16 PM
> <input type="checkbox"/>	TEST	Connected	ESP8266	Device	Nov 19, 2022 8:53 PM

Items per page 50 | 1-3 of 3 items

1 of 1 page

Type here to search

22°C Haze

21:28  
19-11-2022

Browse Action Device Types Interfaces

Add Device +

<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added
▼ <input type="checkbox"/>	32826	Connected	Smart_Farming	Device	Oct 28, 2022 11:29 PM

Identity Device Information Recent Events State Logs

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

Event	Value	Format	Last Received
status	{"soil_moisture":29,"temperature":99,"humidity"...	json	a few seconds ago
status	{"soil_moisture":2,"temperature":42,"humidity"...	json	a few seconds ago
status	{"soil_moisture":2,"temperature":37,"humidity"...	json	a few seconds ago
status	{"soil_moisture":94,"temperature":106,"humidit...	json	a few seconds ago
status	{"soil_moisture":71,"temperature":-6,"humidity"...		

0 Simulations running

4:07 PM  
11/10/2022



Service Details - IBM Cloud x IBM Watson IoT Platform x Node-RED: node-red-zncs- x Getting Started with MIT App Inventor x MIT App Inventor

node-red-zncs-2022-11-04.au-syd.mybluemix.net/red/#flow/f23f5cad061e8487

Node-RED

Flow 1

common

- inject
- debug
- complete
- catch
- status
- link in
- link call
- link out
- comment

function

- function
- switch

Flow 1 Diagram:

```
graph LR
    IoT[IBM IoT] --> SM[Soil Moisture]
    IoT --> Hum[Humidity]
    IoT --> Temp[Temperature]
    SM --> SM_Monitor[Soil Moisture]
    Hum --> Hum_Monitor[Humidity]
    Temp --> Temp_Monitor[Temperature]
    Temp --> MP1[msg payload]
    MP1 --> MP1_Monitor[msg payload]
    GET1[/data] --> Data[data]
    Data --> HTTP1[http]
    MOT_ON[MOTOR ON] --> IoT2[IBM IoT]
    MOT_OFF[MOTOR OFF] --> IoT2
    MOT_ON --> MP2[msg payload]
    MOT_OFF --> MP2
    MP2 --> HTTP2[http]
    GET2[/command] --> HTTP2
```

debug

msg payload : number

95

10/11/2022, 16:09:05 node: c7bc968f68e5d4e5  
iot-2/type/Smart\_Farming/id/32826/evt/status/fmt/json :  
msg.payload : number

71

10/11/2022, 16:09:05 node: c7bc968f68e5d4e5  
iot-2/type/Smart\_Farming/id/32826/evt/status/fmt/json :  
msg.payload : number

93

10/11/2022, 16:09:07 node: c7bc968f68e5d4e5  
iot-2/type/Smart\_Farming/id/32826/evt/status/fmt/json :  
msg.payload : number

16

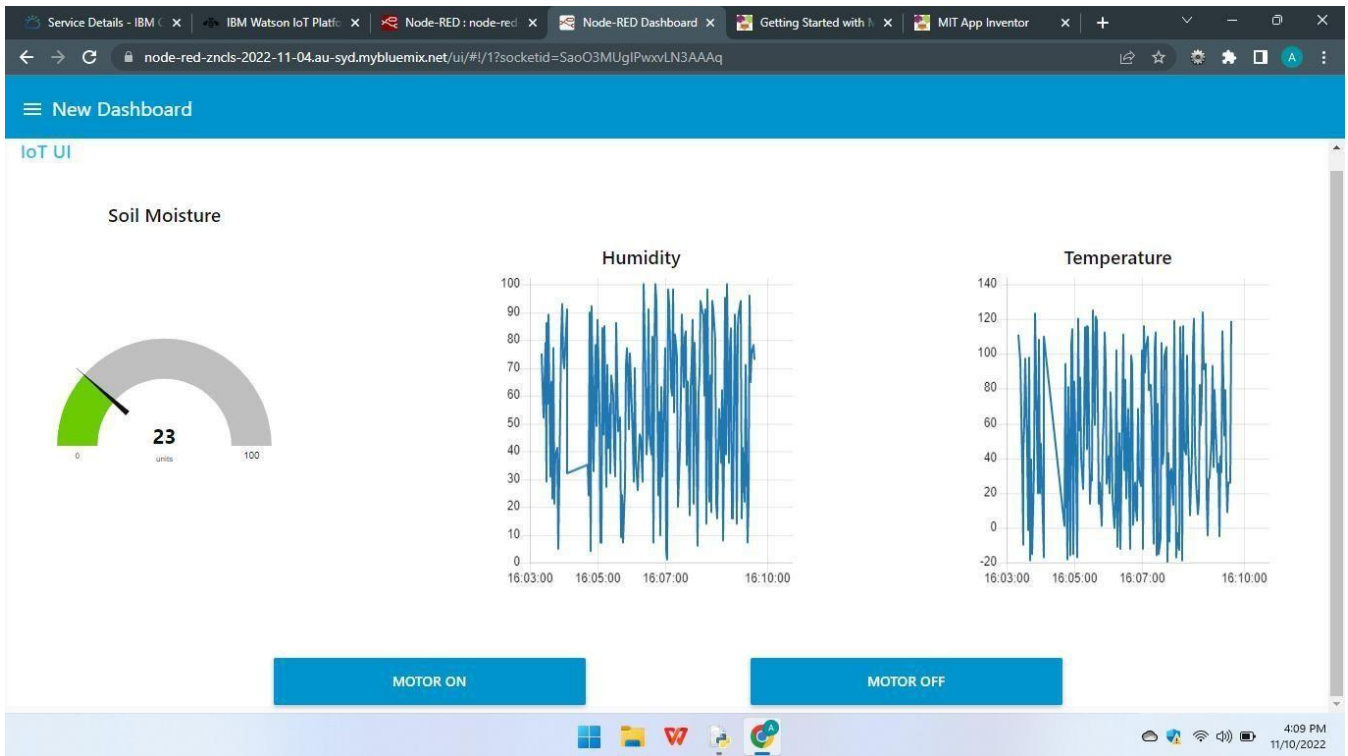
10/11/2022, 16:09:07 node: c7bc968f68e5d4e5  
iot-2/type/Smart\_Farming/id/32826/evt/status/fmt/json :  
msg.payload : number

14

10/11/2022, 16:09:07 node: c7bc968f68e5d4e5  
iot-2/type/Smart\_Farming/id/32826/evt/status/fmt/json :  
msg.payload : number

35

4:09 PM  
11/10/2022



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published data successfully {'soil_moisture': 50, 'temperature': 112, 'humidity': 79}
Published data successfully {'soil_moisture': 54, 'temperature': 112, 'humidity': 79}
Published data successfully {'soil_moisture': 37, 'temperature': -6, 'humidity': 74}
Published data successfully {'soil_moisture': 69, 'temperature': 96, 'humidity': 83}
Published data successfully {'soil_moisture': 57, 'temperature': 88, 'humidity': 84}
Published data successfully {'soil_moisture': 25, 'temperature': 54, 'humidity': 99}
Published data successfully {'soil_moisture': 27, 'temperature': 16, 'humidity': 7}
Published data successfully {'soil_moisture': 28, 'temperature': -1, 'humidity': 100}
Published data successfully {'soil_moisture': 64, 'temperature': 69, 'humidity': 19}
Published data successfully {'soil_moisture': 9, 'temperature': 39, 'humidity': 86}
Published data successfully {'soil_moisture': 8, 'temperature': -3, 'humidity': 61}
Published data successfully {'soil_moisture': 41, 'temperature': 61, 'humidity': 49}
Published data successfully {'soil_moisture': 87, 'temperature': 92, 'humidity': 8}
Published data successfully {'soil_moisture': 84, 'temperature': 92, 'humidity': 84}
Message received from IBM IoT Platform: motoron
Motor is switched ON

Published data successfully {'soil_moisture': 80, 'temperature': 26, 'humidity': 99}
Message received from IBM IoT Platform: motoroff
Motor is switched OFF

Published data successfully {'soil_moisture': 31, 'temperature': 108, 'humidity': 46}
Published data successfully {'soil_moisture': 36, 'temperature': 86, 'humidity': 69}
Published data successfully {'soil_moisture': 49, 'temperature': 99, 'humidity': 34}
Published data successfully {'soil_moisture': 91, 'temperature': 90, 'humidity': 15}
Published data successfully {'soil_moisture': 99, 'temperature': 75, 'humidity': 2}
Published data successfully {'soil_moisture': 25, 'temperature': 2, 'humidity': 99}
Published data successfully {'soil_moisture': 61, 'temperature': 7, 'humidity': 61}
Published data successfully {'soil_moisture': 17, 'temperature': 39, 'humidity': 85}
Published data successfully {'soil_moisture': 89, 'temperature': 51, 'humidity': 61}
Published data successfully {'soil_moisture': 72, 'temperature': 18, 'humidity': 7}
Published data successfully {'soil_moisture': 7, 'temperature': 42, 'humidity': 36}
Published data successfully {'soil_moisture': 67, 'temperature': -4, 'humidity': 94}
Published data successfully {'soil_moisture': 21, 'temperature': 41, 'humidity': 74}
Published data successfully {'soil_moisture': 26, 'temperature': 114, 'humidity': 71}
Published data successfully {'soil_moisture': 89, 'temperature': -2, 'humidity': 48}
Published data successfully {'soil_moisture': 10, 'temperature': -12, 'humidity': 2}

Ln: 406 Col: 0
4:18 PM 11/10/2022
```

Service Details - IBM | IBM Watson IoT Platform | Node-RED: node-red | Node-RED Dashboard | Getting Started with | MIT App Inventor

node-red-zncds-2022-11-04.au-syd.mybluemix.net/red/#flow/f23f5cad061e8487

Node-RED

Flow 1

debug

common

- inject
- debug
- complete
- catch
- status
- link in
- link call
- link out
- comment

function

- function
- switch

Flow 1 Diagram:

```
graph LR
    IoT[IBM IoT] --> SM[Soil Moisture]
    IoT --> Hum[Humidity]
    IoT --> Temp[Temperature]
    SM --> SM_Widget[Soil Moisture]
    Hum --> Hum_Widget[Humidity]
    Temp --> Temp_Widget[Temperature]
    Temp --> MP[msg.payload]
    MP --> MP_Widget[msg.payload]
    GET_DATA[GET /data] --> DATA[data]
    DATA --> HTTP[http]
    MOTOR_ON[MOTOR ON] --> IoT
    MOTOR_ON --> MP_Widget
    MOTOR_OFF[MOTOR OFF] --> IoT
    MOTOR_OFF --> MP_Widget
    GET_CMD[GET /command] --> MP_Widget
    GET_CMD --> HTTP
```

debug console:

```
msg.payload : number
92
10/11/2022, 16:15:47 node: 47237c95f9032d61
msg.payload : Object
{ command: "motoron" }
10/11/2022, 16:15:48 node: c7bc968f68e5d4e5
iot-2/type/Smart_Farming/id/32826/evt/status/fmt/json :
msg.payload : number
80
10/11/2022, 16:15:49 node: c7bc968f68e5d4e5
iot-2/type/Smart_Farming/id/32826/evt/status/fmt/json :
msg.payload : number
99
10/11/2022, 16:15:50 node: c7bc968f68e5d4e5
iot-2/type/Smart_Farming/id/32826/evt/status/fmt/json :
msg.payload : number
26
10/11/2022, 16:15:50 node: 47237c95f9032d61
msg.payload : Object
{ command: "motoroff" }
10/11/2022, 16:15:51 node: c7bc968f68e5d4e5
```

4:16 PM 11/10/2022

## Development of Sprint-4 (MIT Application and its Execution)

Smart\_Farming.py - C:\Users\A S ABISHEK\AppData\Local\Programs\Python\Python37\Smart\_Farming.py (3.7.0)

File Edit Format Run Options Window Help

```
import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig={
    "identity":{
        "orgId":"uq23sr",
        "typeId":"Smart_Farming",
        "deviceId":"32826"
    },
    "auth": {
        "token":"3wNLT001g8VpEJEpsq"
    }
}
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",myData)
    time.sleep(2)
    client.commandCallback=myCommandCallback
client.disconnect()
```

Ln: 32 Col: 38



4:04 PM  
11/10/2022

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.

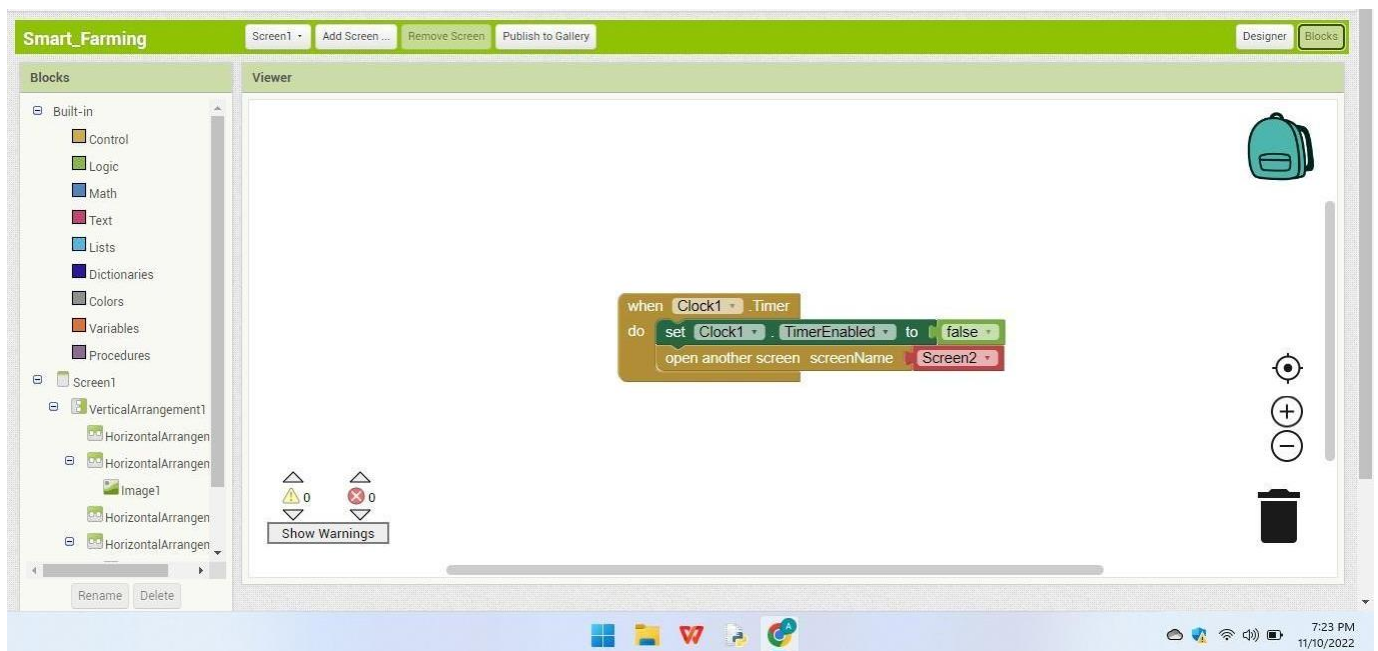
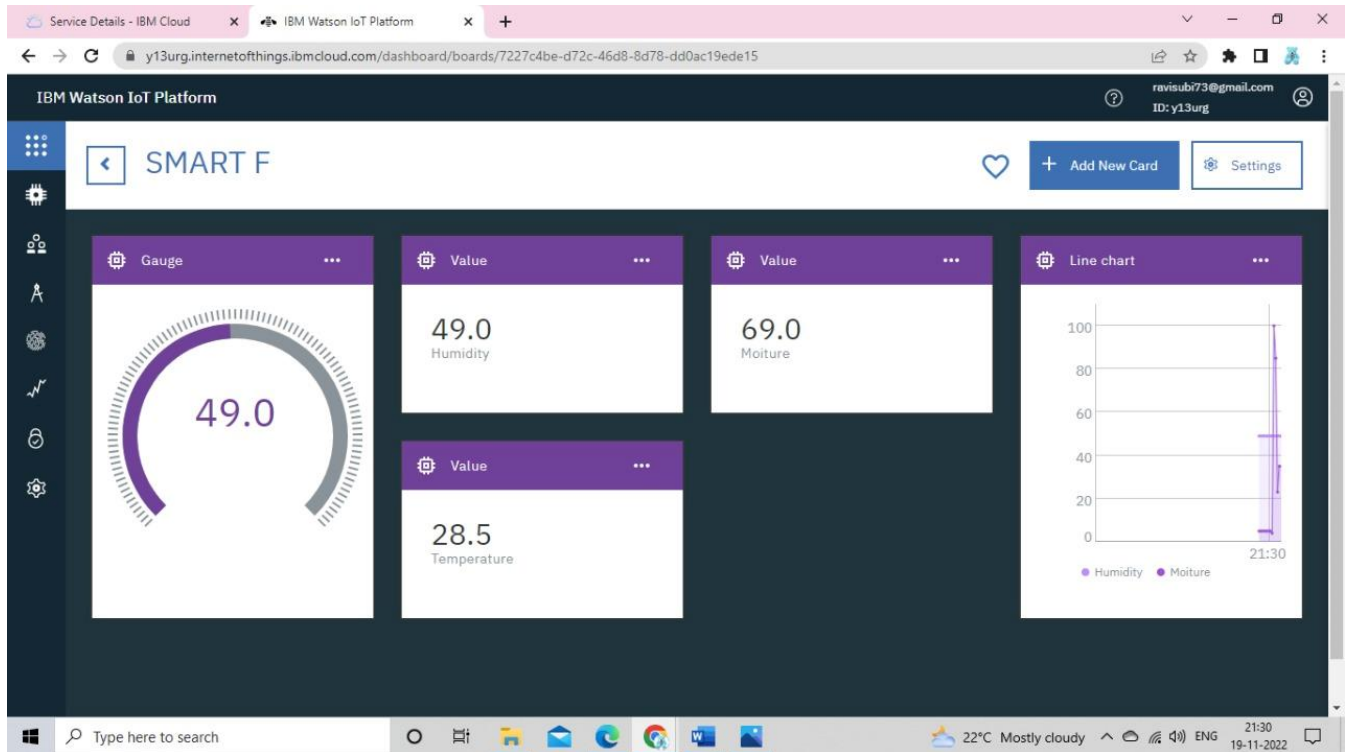
```
>>>
RESTART: C:\Users\A S ABISHEK\AppData\Local\Programs\Python\Python37\Smart_Farming.py
2022-11-10 16:04:42.463 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:uq23sr:Smart_Farming:32826Published
data successfully
{'soil_moisture': 37, 'temperature': 1, 'humidity': 35}
Published data successfully {'soil_moisture': 89, 'temperature': 94, 'humidity': 24}
Published data successfully {'soil_moisture': 57, 'temperature': 28, 'humidity': 90}
Published data successfully {'soil_moisture': 65, 'temperature': -18, 'humidity': 4}
Published data successfully {'soil_moisture': 87, 'temperature': 81, 'humidity': 92}
Published data successfully {'soil_moisture': 62, 'temperature': -16, 'humidity': 33}
Published data successfully {'soil_moisture': 99, 'temperature': 105, 'humidity': 62}
Published data successfully {'soil_moisture': 41, 'temperature': 114, 'humidity': 78}
Published data successfully {'soil_moisture': 26, 'temperature': -15, 'humidity': 49}
Published data successfully {'soil_moisture': 55, 'temperature': 84, 'humidity': 87}
```

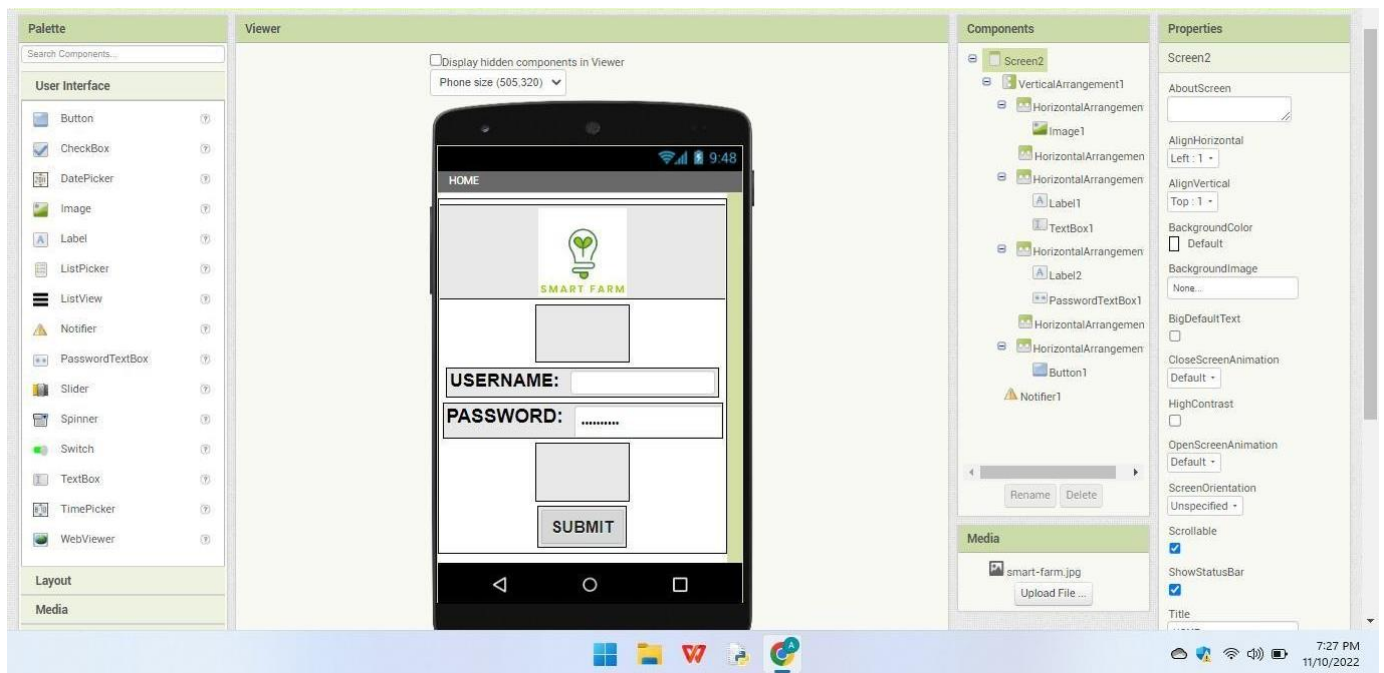
Ln: 5 Col: 0

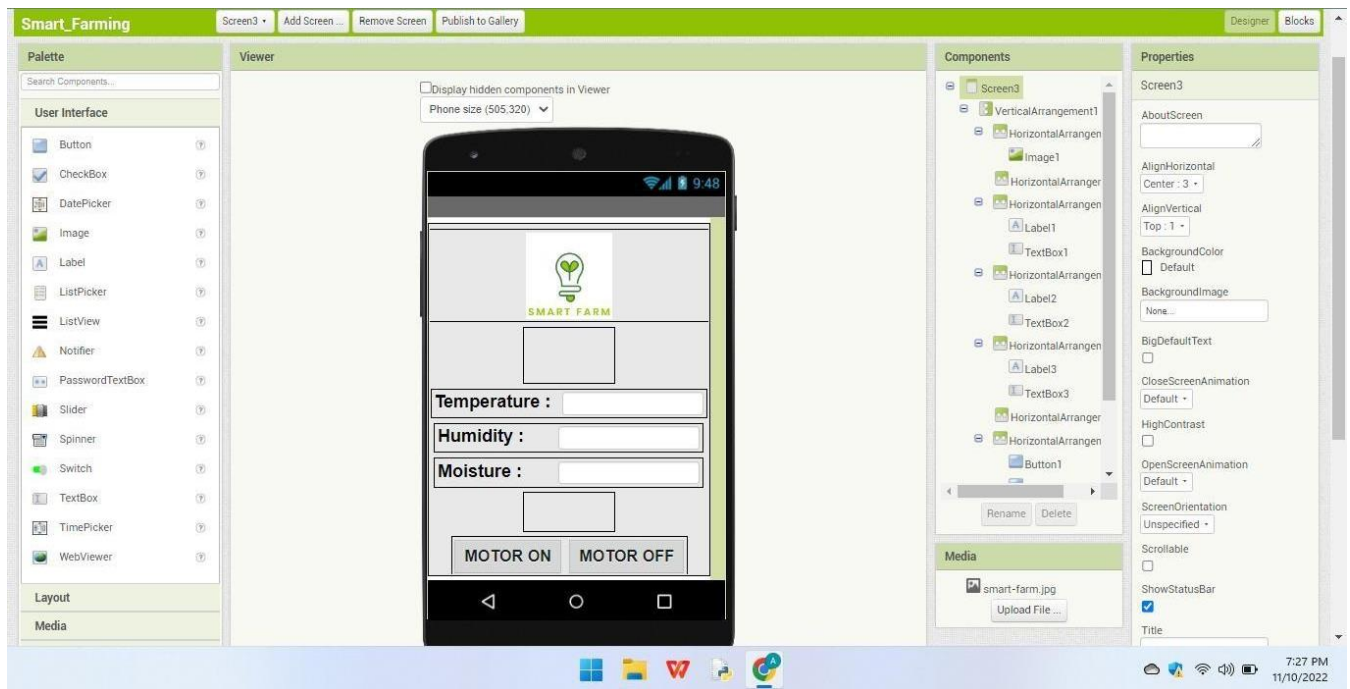


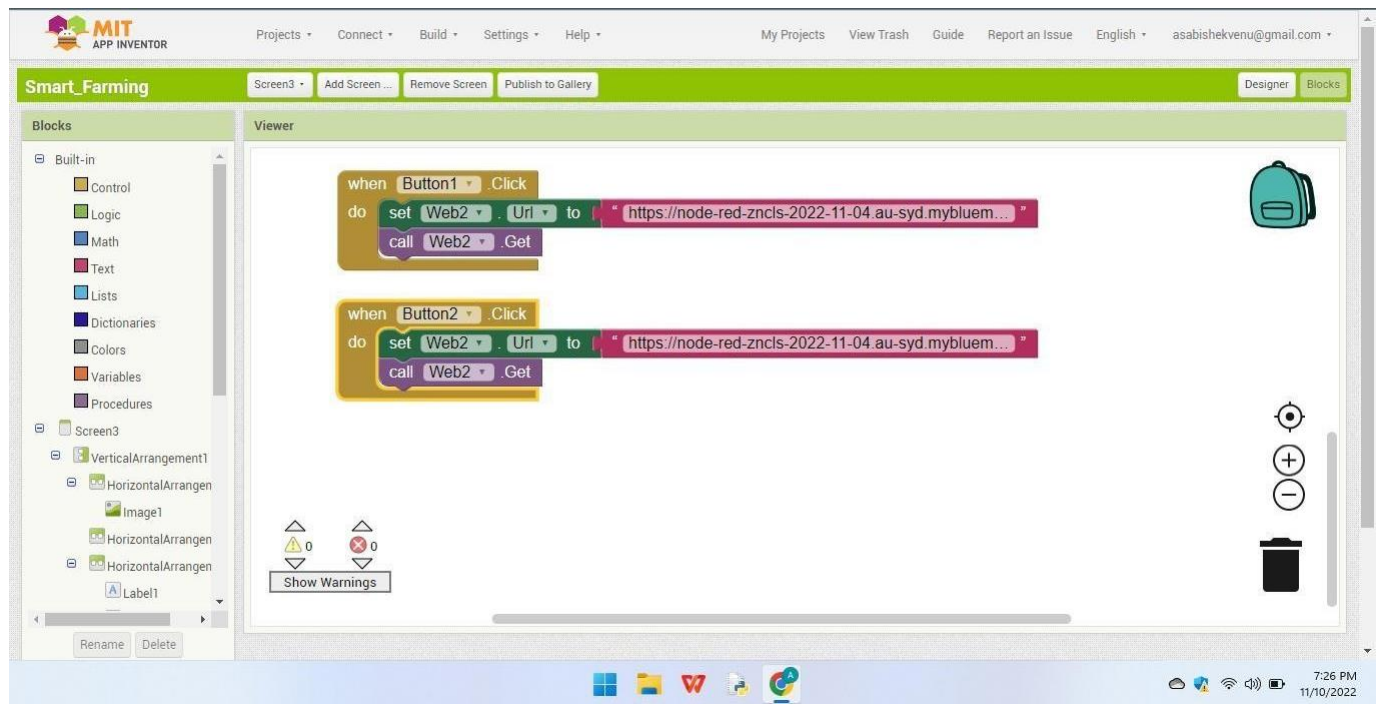
4:05 PM  
11/10/2022













**USERNAME:**

**PASSWORD:**

**SUBMIT**



## SMART FARM

**Temperature :**

**Humidity :**

**Moisture :**

**MOTOR ON**

**MOTOR OFF**

## **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.



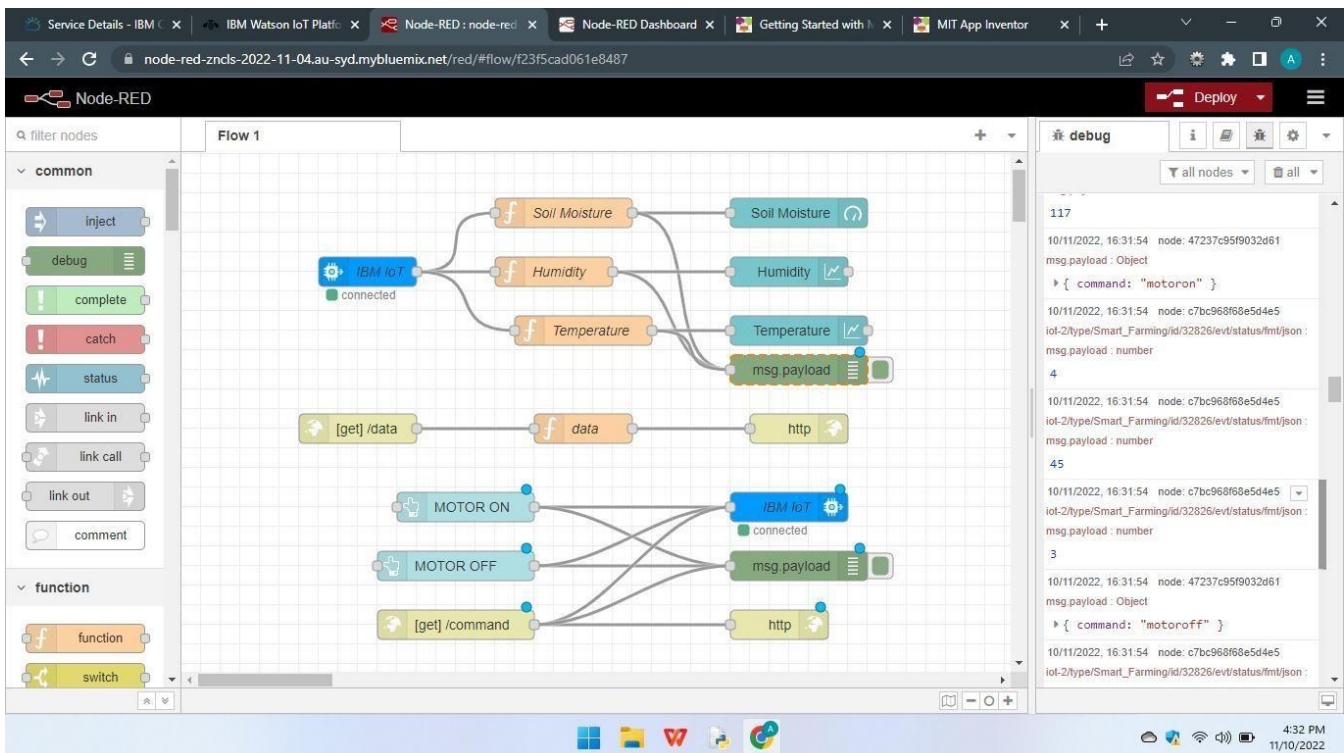
# CHAPTER 9

## RESULTS

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published data successfully {'soil_moisture': 81, 'temperature': 120, 'humidity': 95}
Published data successfully {'soil_moisture': 52, 'temperature': 124, 'humidity': 33}
Published data successfully {'soil_moisture': 70, 'temperature': 123, 'humidity': 1}
Published data successfully {'soil_moisture': 17, 'temperature': 60, 'humidity': 48}
Published data successfully {'soil_moisture': 4, 'temperature': 103, 'humidity': 19}
Published data successfully {'soil_moisture': 74, 'temperature': -2, 'humidity': 89}
Published data successfully {'soil_moisture': 62, 'temperature': 92, 'humidity': 21}
Published data successfully {'soil_moisture': 11, 'temperature': 74, 'humidity': 56}
Published data successfully {'soil_moisture': 38, 'temperature': 50, 'humidity': 86}
Published data successfully {'soil_moisture': 14, 'temperature': 116, 'humidity': 54}
Published data successfully {'soil_moisture': 60, 'temperature': 100, 'humidity': 67}
Published data successfully {'soil_moisture': 71, 'temperature': 101, 'humidity': 78}
Published data successfully {'soil_moisture': 41, 'temperature': 121, 'humidity': 56}
Published data successfully {'soil_moisture': 78, 'temperature': 98, 'humidity': 49}
Published data successfully {'soil_moisture': 39, 'temperature': 73, 'humidity': 48}
Published data successfully {'soil_moisture': 61, 'temperature': 55, 'humidity': 89}
Published data successfully {'soil_moisture': 80, 'temperature': 85, 'humidity': 69}
Published data successfully {'soil_moisture': 21, 'temperature': 106, 'humidity': 62}
Published data successfully {'soil_moisture': 16, 'temperature': -4, 'humidity': 15}
Published data successfully {'soil_moisture': 18, 'temperature': 111, 'humidity': 6}
Published data successfully {'soil_moisture': 86, 'temperature': -2, 'humidity': 79}
Published data successfully {'soil_moisture': 91, 'temperature': 39, 'humidity': 52}
Published data successfully {'soil_moisture': 5, 'temperature': 55, 'humidity': 25}
Published data successfully {'soil_moisture': 41, 'temperature': -20, 'humidity': 90}
Published data successfully {'soil_moisture': 99, 'temperature': 117, 'humidity': 79}
Message received from IBM IoT Platform: motoron
Motor is switched ON

Published data successfully {'soil_moisture': 4, 'temperature': 3, 'humidity': 45}
Message received from IBM IoT Platform: motoroff
Motor is switched OFF

Published data successfully {'soil_moisture': 56, 'temperature': 113, 'humidity': 48}
Published data successfully {'soil_moisture': 45, 'temperature': 25, 'humidity': 99}
Published data successfully {'soil_moisture': 72, 'temperature': -11, 'humidity': 55}
```





## 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.

## CHAPTER 13

### APPENDIX

#### 13.2 Source Code :

```
#include <ESP8266WiFi.h>

#include <WiFiClient.h> #include
<PubSubClient.h>#include "DHT.h"


const char* ssid = "SMART-G";
const char* password = "10112019";


#define DHTPIN D6#define G
D0
#define DHTTYPE DHT11 DHT
dht(DHTPIN, DHTTYPE);


#define ID "y13urg"
#define DEVICE_TYPE "ESP8266"
#define DEVICE_ID "TEST" #define
TOKEN "TEST-12345"

char  server[]  =  ID  ".messaging.internetofthings.ibmcloud.com";  char
publish_Topic1[] = "iot-2/evt/Data1/fmt/json";

char  publish_Topic2[]  =  "iot-2/evt/Data2/fmt/json";  char
publish_Topic3[] = "iot-2/evt/Data2/fmt/json";char publish_Topic4[]
= "iot-2/evt/Data2/fmt/json";char authMethod[] = "use-token-auth";
char token[] = TOKEN;

char clientId[] = "d:" ID ":" DEVICE_TYPE ":" DEVICE_ID;


WiFiClient wifiClient;
```

```

PubSubClient client(server, 1883, NULL, wifiClient);

void setup() {
  pinMode(D0,OUTPUT);
  digitalWrite(D0,HIGH);
  Serial.begin(115200); dht.begin();
  Serial.println();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println(WiFi.localIP());

  if (!client.connected()) { Serial.print("Reconnecting
    client to ");Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) { Serial.print(".");
      delay(500);
    }
    Serial.println("Connected TO IBM IoT cloud!");
  }
}

long previous_message = 0;void loop() {
  client.loop();
  long current = millis();
  if (current - previous_message > 3000) {
    previous_message = current;
    float hum = dht.readHumidity(); float temp =

```

```

    dht.readTemperature();
    float MOI = map(analogRead(A0), 0, 1023, 100, 0);
    float bi = map(digitalRead(D1), 0, 1, 100, 0);if
    (isnan(hum) || isnan(temp) ){
    Serial.println(F("Failed to read from DHT sensor!"));return;
}

Serial.print("Temperature: ");
Serial.print(temp); Serial.print("°C");
Serial.print(" Humidity: ");
Serial.print(hum); Serial.print("%");
Serial.print("SOIL MOITURE: ");
Serial.print(MOI); Serial.print("ANIMAL AND
BIRD: ");Serial.print(bi);
if(MOI<=10)
{
    digitalWrite(D0,LOW);
    delay(100);
    digitalWrite(D0,HIGH);
}
else
{
    digitalWrite(D0,HIGH);
}

String payload = "{\"d\":{\"Name\":\"\" DEVICE_ID \"\"\";payload +=
    \",\"Temperature\":\"\";
    payload += temp;payload +=
    \"}\"}";

```



```

Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publish_Topic1, (char*) payload.c_str())) {Serial.println("Published
    successfully");
} else {
    Serial.println("Failed");
}
String payload1 = "{\"d\":{\"Name\":\"\" DEVICE_ID \"\"";payload1 +=
    "\",\"Humidity\":\"";
    payload1 += hum;payload1
    += "}}";
    Serial.print("Sending payload: ");
    Serial.println(payload1); Serial.println('\n');

if (client.publish(publish_Topic2, (char*) payload1.c_str())) {
    Serial.println("Published successfully");
} else {
    Serial.println("Failed");
}

String payload3 = "{\"d\":{\"Name\":\"\" DEVICE_ID \"\"";payload3 +=
    "\",\"Moiture\":\"";
    payload3 += MOI;payload3
    += "}}";

Serial.print("Sending payload: ");
Serial.println(payload3);

```

```

if (client.publish(publish_Topic3, (char*) payload3.c_str())) {
    Serial.println("Published successfully");
} else {
    Serial.println("Failed");
}

```

```

String payload4 = "{\"d\":{\"Name\":\"\" DEVICE_ID \"\"";payload4 +=
    "\",\"Animal&Bird\":\"";
    payload4 += bi; payload4 +=
    "\"}"}";

```

```

Serial.print("Sending payload: ");
Serial.println(payload4);

```

```

if (client.publish(publish_Topic4, (char*) payload4.c_str())) {
    Serial.println("Published successfully");
} else {
    Serial.println("Failed");
}

```

```

}
}

```

## 13.2 GITHUB LINK

<https://github.com/IBM-EPBL/IBM-Project-15427-1659598581.git>

## 13.2 DEMO LINK:

<https://youtu.be/iOVOSv8wrt4>