SMART FARMER-IOT ENABLED SMART FARMING APPLICATION

PROJECT REPORT

IBM-Project-35958-1660290928

TEAM ID: PNT2022TMID26610

Submitted by

YOKESH M	**(212919104061)**
SACHIN RAJ G	**(212919104038)**
ANTONY AMOSE I	**(212919104004)**
ANIL SHEBIN S J	**(212919104002)**
VIGNESHWARAN S	**(212919104054)**

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ST.JOSEPH COLLEGE OF ENGINEERING

SRIPERUMBUDUR-602 117

PROJECT REPORT

1.INRODUCTION

1.1 Project Overview

IOT- Enabled Smart Farming agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature, humidity using some sensors. Farmer can monitor all the sensor parameters by using a web or mobile application even if the farmer is not near his field. Watering the crop is one of the Important task for the farmers. They can make the decision whether to water the crop or postpone it by monitoring the sensor parameters and control the motor pumps from the mobile application itself. All the sensor parameters are stored in the IBM Cloudant DB

IoT is network that connects physical objects or things embedded with electronics, software and sensors through network connectivity that collects and transfers data using cloud for communication. Data is transferred through internet without human to human or human to computer interaction. In this project we have not used any hardware. Instead of real soil and temperature conditions, sensors IBM IoT Simulator is used which can transmit soil moisture temperature as required..

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

Project Deliverables: Application for IoT based Smart Agriculture System

1.2 Purpose

IoT based farming improves the entire agriculture system by monitoring the field in real-time. With the help of IoT in agriculture not only saves the time but also reduces the extravagant use of resources such as water and electricity. Sometimes due to over or less supply of water in the agricultural field crops may not grow proper. Using IoT supply of water and growth of plants can be satisfied to a greater extent. The flow of water can be controlled from the application.

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

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

2. LITERATURE SURVEY

2.1 Existing Problem

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

2.2 References

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

[2] H.G.C.R.Laksiri, H.A.C.Dharmagunawardhana, J.V.Wijayakulasooriya ”Design and Optimization of loT Based Smart Irrigation System in Sri Lanka”2019 .

[3] Anushree Math, Layak Ali, Pruthviraj U ”Development of Smart Drip Irriga- tion System Using IoT”2018.

[4] Dweepayan Mishra1 ,Arzeena Khan2 Rajeev Tiwari3 , Shuchi Upadhay,”Automated Irrigation System-IoT Based Approach”,2018.

[5] R. Nageswara Rao, B.Sridhar,”IOT BASED SMART CROP-FIELD MONI- TORING AND AUTOMATION IRRIGATION SYSTEM”. 2018

[6] Shweta B. Saraf, Dhanashri H. Gawal,”IoT Based Smart Irrigation Monitoring And Controlling System”.2017

[7] Shrihari M, ”A Smart Wireless System to Automate Production of Crops and Stop Intrusion Using Deep Learning” 2020.

[8] G. Sushanth1, and S. Sujatha, ”IOT Based Smart Agriculture System”2018.

[9] Vaishali S, Suraj S, Vignesh G, Dhivya S and Udhayakumar S, ”Mobile Integrated Smart Irrigation Management and Monitoring System Using IOT”,2017

2.3 Problem Statement Definitions

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

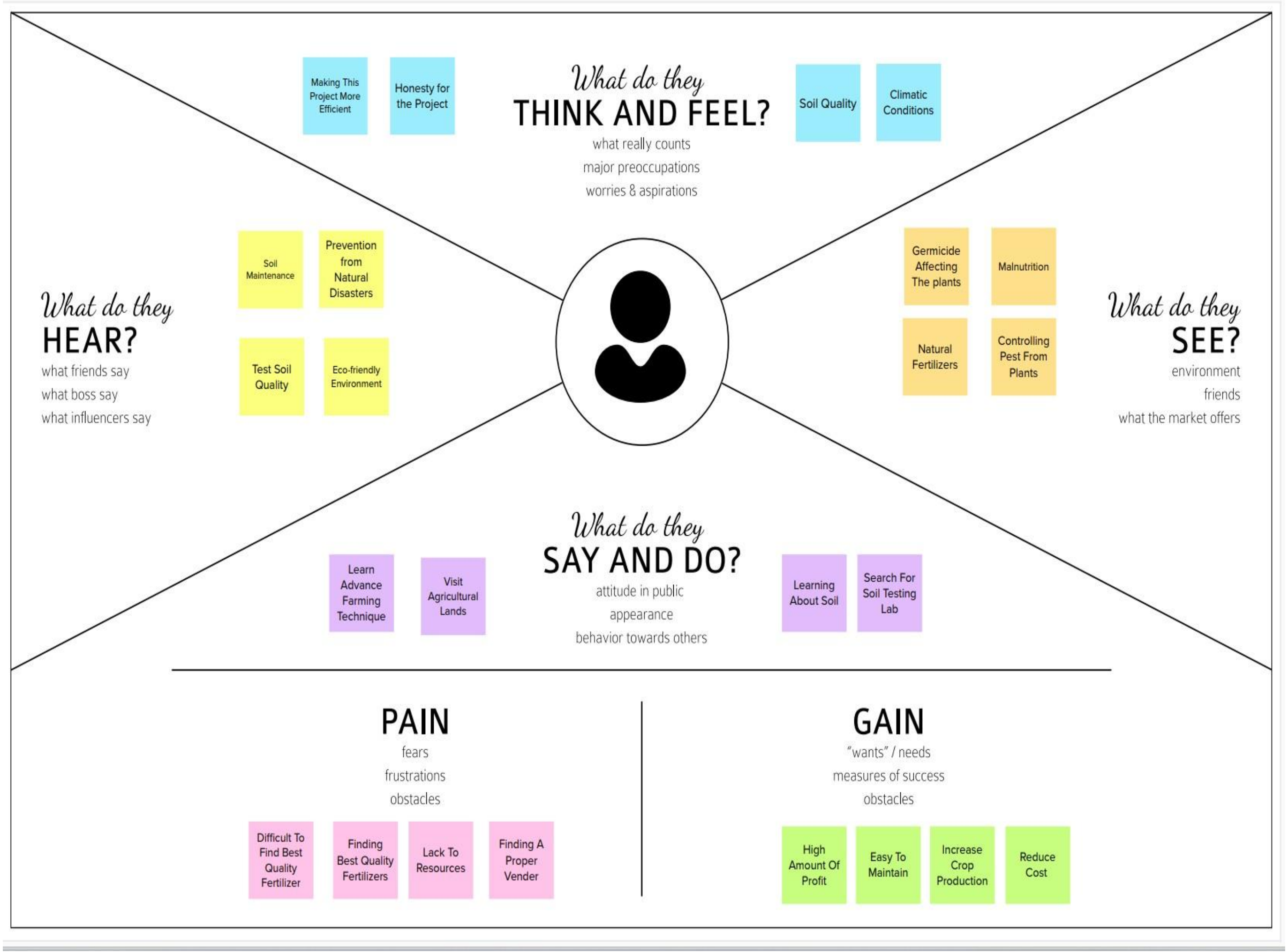
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

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

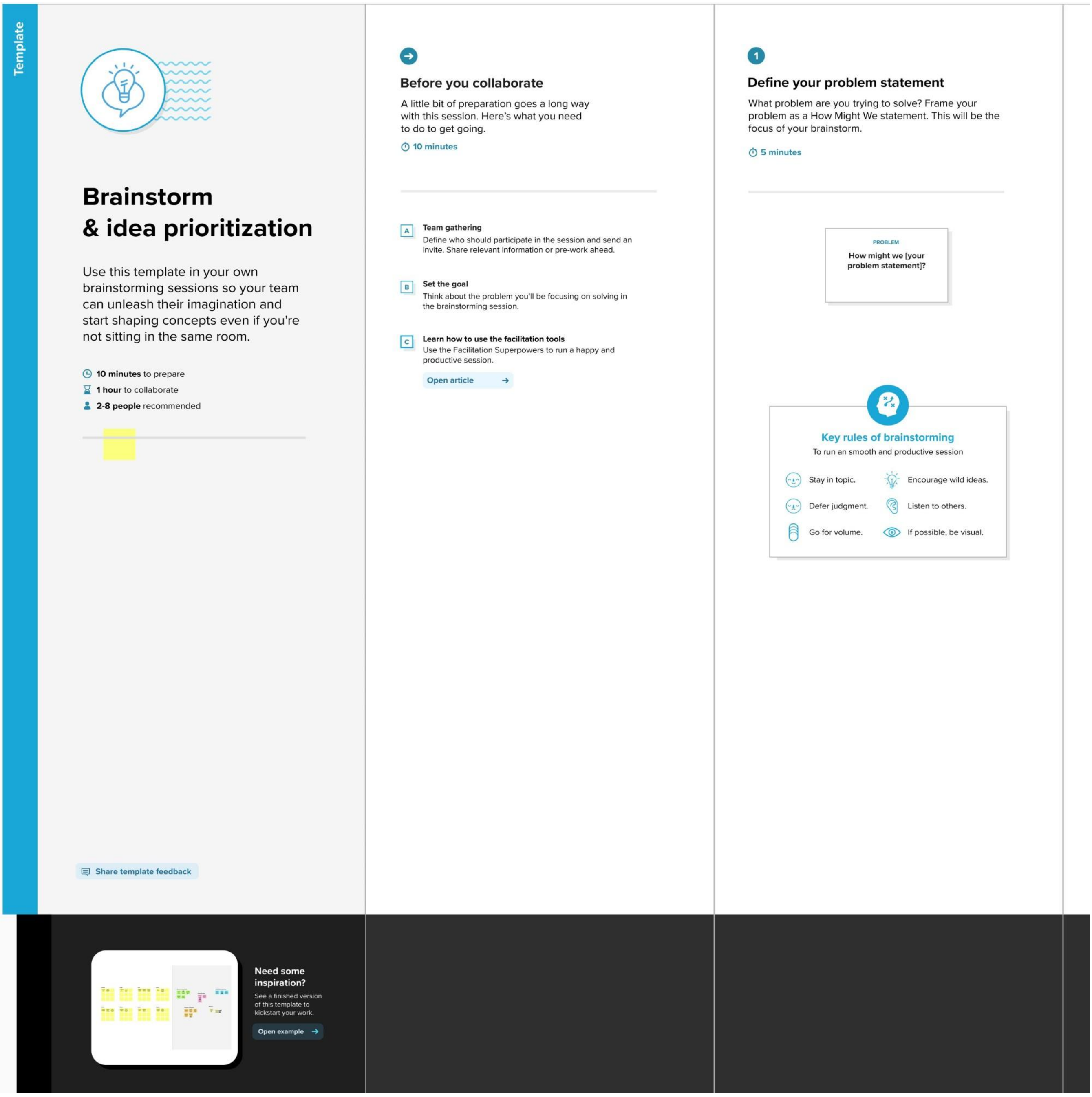
It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and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

Empathy Map

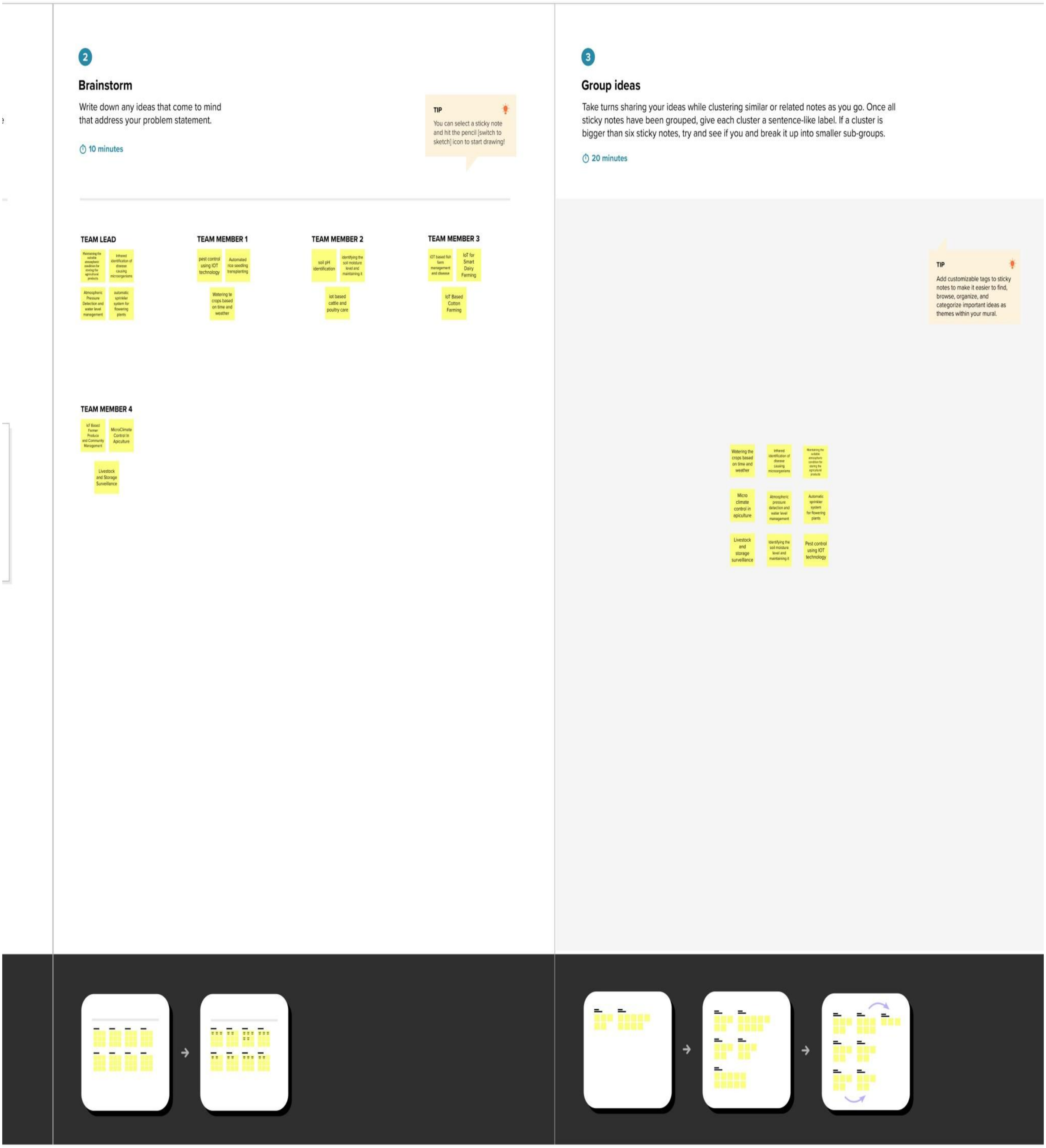


3.2 Ideation and Brainstorming

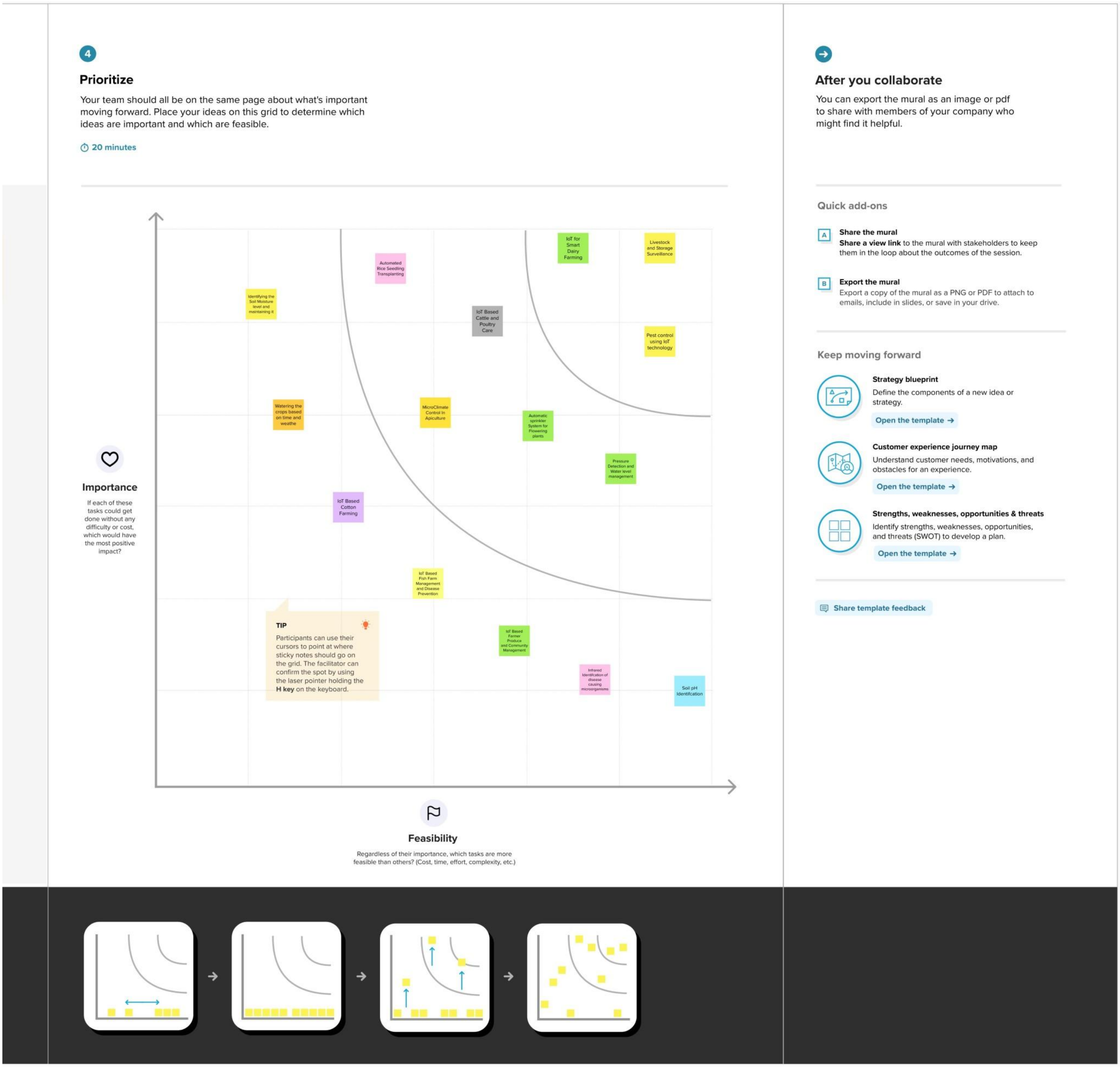
Step-1: Team Gathering, Collaboration and Select the Problem Statements:



Step-2: Brainstorm ,Idea Listing and Grouping



Step-3: Idea Prioritization

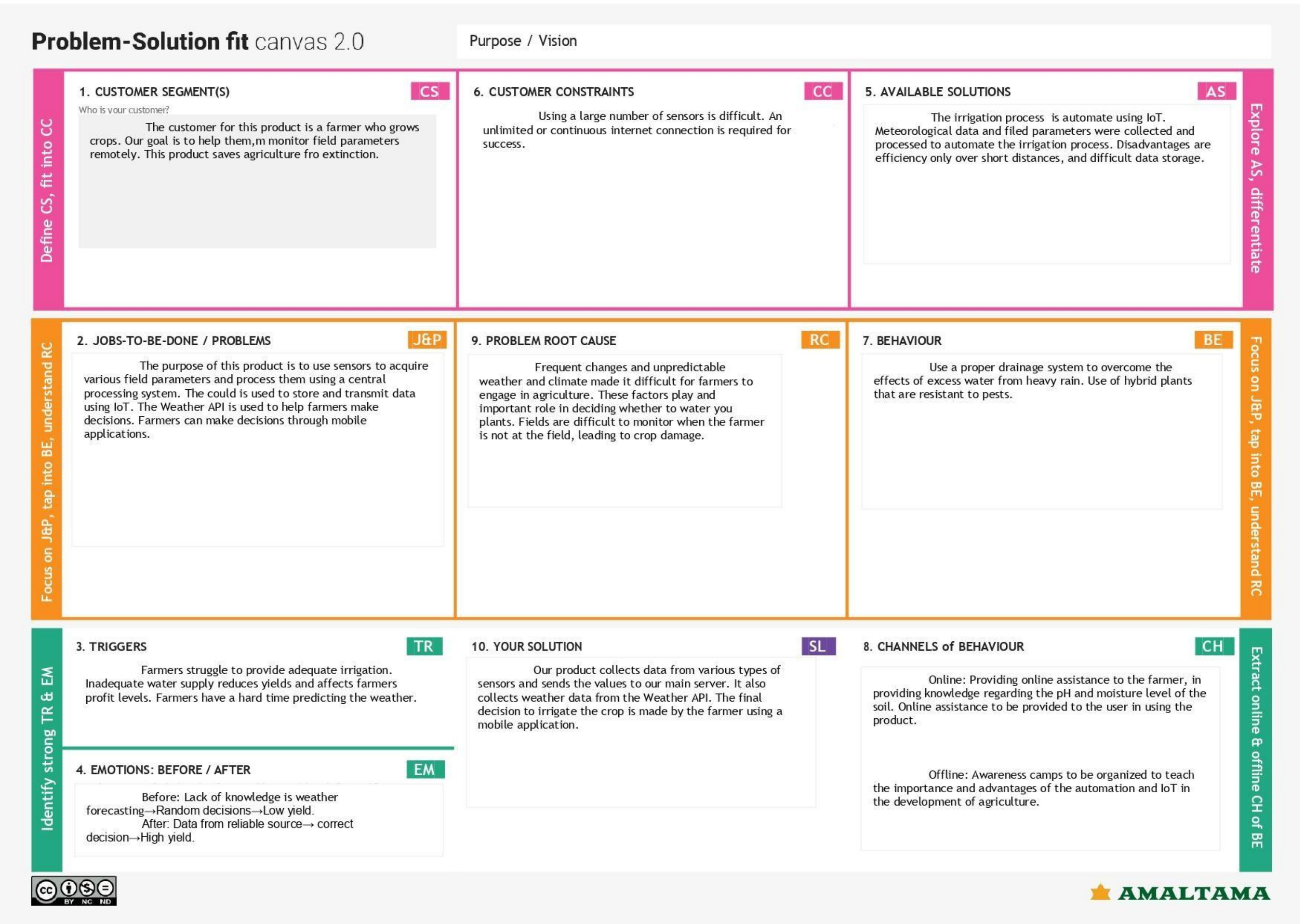


3.3 Proposed Solution:

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To make farming easier by choosing several constraints in agriculture and to overcome those constraints, to increase production quality and quantity using IoT.
2.	Idea / Solution description	Using smart techniques like monitoring farms climate, smart irrigation and soil analysis.
3.	Novelty / Uniqueness	Solar power smart irrigation system which helps you to monitor temperature, moisture humidity using smart sensors.
4.	Social Impact / Customer Satisfaction	It is better than the present modern irrigation system by using this method we can control soil erosion. There will be better production yields.

5.	Business Model (Revenue Model)	As the productivity increases customer satisfaction also increases and hence need for the application also Increases, which increases the revenue of the business.
6.	Scalability of the Solution	It is definitely scalable we can increase the constraints hen the problem arises.

3.4 Problem Solution fit



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

The customer for this product is a farmer who grows crops. Our goal is to help them,m monitor field parameters remotely. This product saves agriculture fro extinction.

6. CUSTOMER CONSTRAINTS CC

Using a large number of sensors is difficult. An unlimited or continuous internet connection is required for success.

5. AVAILABLE SOLUTIONS AS

The irrigation process is automate using IoT. Meteorological data and filed parameters were collected and processed to automate the irrigation process. Disadvantages are efficiency only over short distances, and difficult data storage.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

The purpose of this product is to use sensors to acquire various field parameters and process them using a central processing system. The could is used to store and transmit data using IoT. The Weather API is used to help farmers make decisions. Farmers can make decisions through mobile applications.

9. PROBLEM ROOT CAUSE RC

Frequent changes and unpredictable weather and climate made it difficult for farmers to engage in agriculture. These factors play and important role in deciding whether to water you plants. Fields are difficult to monitor when the farmer is not at the field, leading to crop damage.

7. BEHAVIOUR BE

Use a proper drainage system to overcome the effects of excess water from heavy rain. Use of hybrid plants that are resistant to pests.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

Farmers struggle to provide adequate irrigation. Inadequate water supply reduces yields and affects farmers profit levels. Farmers have a hard time predicting the weather.

4. EMOTIONS: BEFORE / AFTER EM

Before: Lack of knowledge is weather forecasting→Random decisions→Low yield.
After: Data from reliable source→ correct decision→High yield.

10. YOUR SOLUTION SL

Our product collects data from various types of sensors and sends the values to our main server. It also collects weather data from the Weather API. The final decision to irrigate the crop is made by the farmer using a mobile application.

8. CHANNELS of BEHAVIOUR CH

Online: Providing online assistance to the farmer, in providing knowledge regarding the pH and moisture level of the soil. Online assistance to be provided to the user in using the product.

Offline: Awareness camps to be organized to teach the importance and advantages of the automation and IoT in the development of agriculture.

Extract online & offline CH of BE

AMALTAMA

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	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 croup diseases
FR-4	Manage Modules	Manage Roles of Use 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 Mange the data of live stock conditions

4.2 Non-Functional Requirements:

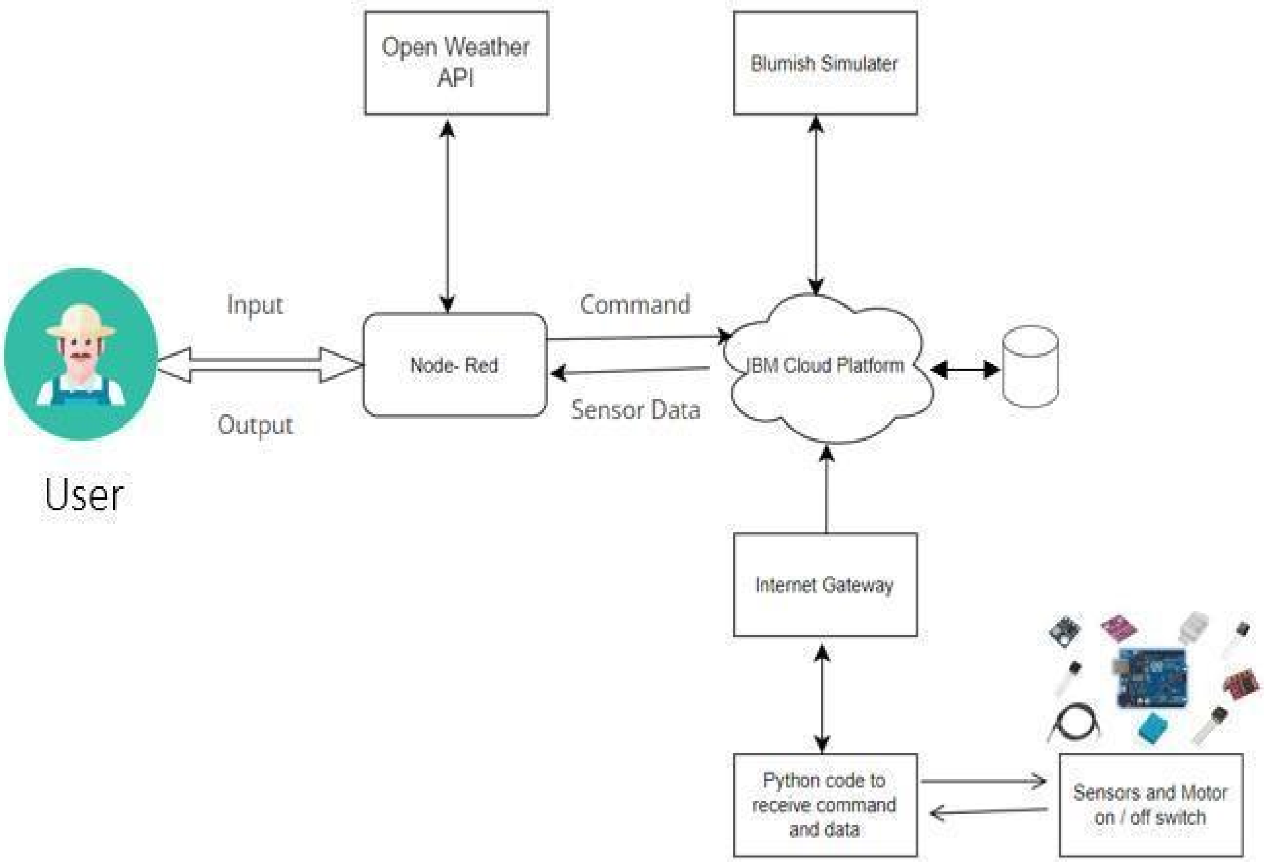
Following are the non-functional requirements of the proposed solution.

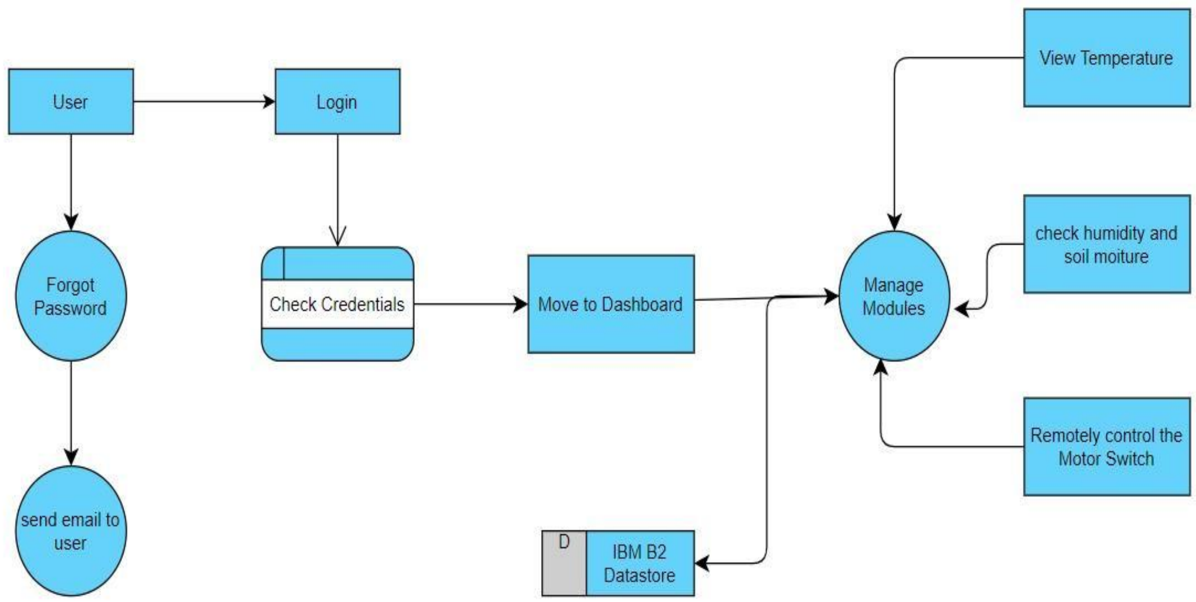
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	✓ User friendly guidelines for users to avail the features. ✓ Most simplistic user interface for ease of use.
NFR-2	**Security**	✓ All the details about the user are protected from unauthorized access. ✓ Detection and identification of any misfunctions of sensors.
NFR-3	**Reliability**	✓ Implementing Mesh IoT Networks. ✓ Building a Multi-layered defence for IoT Networks.
NFR-4	**Performance**	The use of modern technology solutions helps to achieve the maximum performances thus resulting in better quality and quantity yields.
NFR-5	**Availability**	This app is available for all platforms.
NFR-6	**Scalability**	Scalability refers to the ability to increase available resources and system capability without the need to go through a major system redesign or implementation

5.PROJECT DESIGN

5.1 Data Flow Diagrams:

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





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.

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

5.2 Solution & Technical Architecture

Solution Architecture:

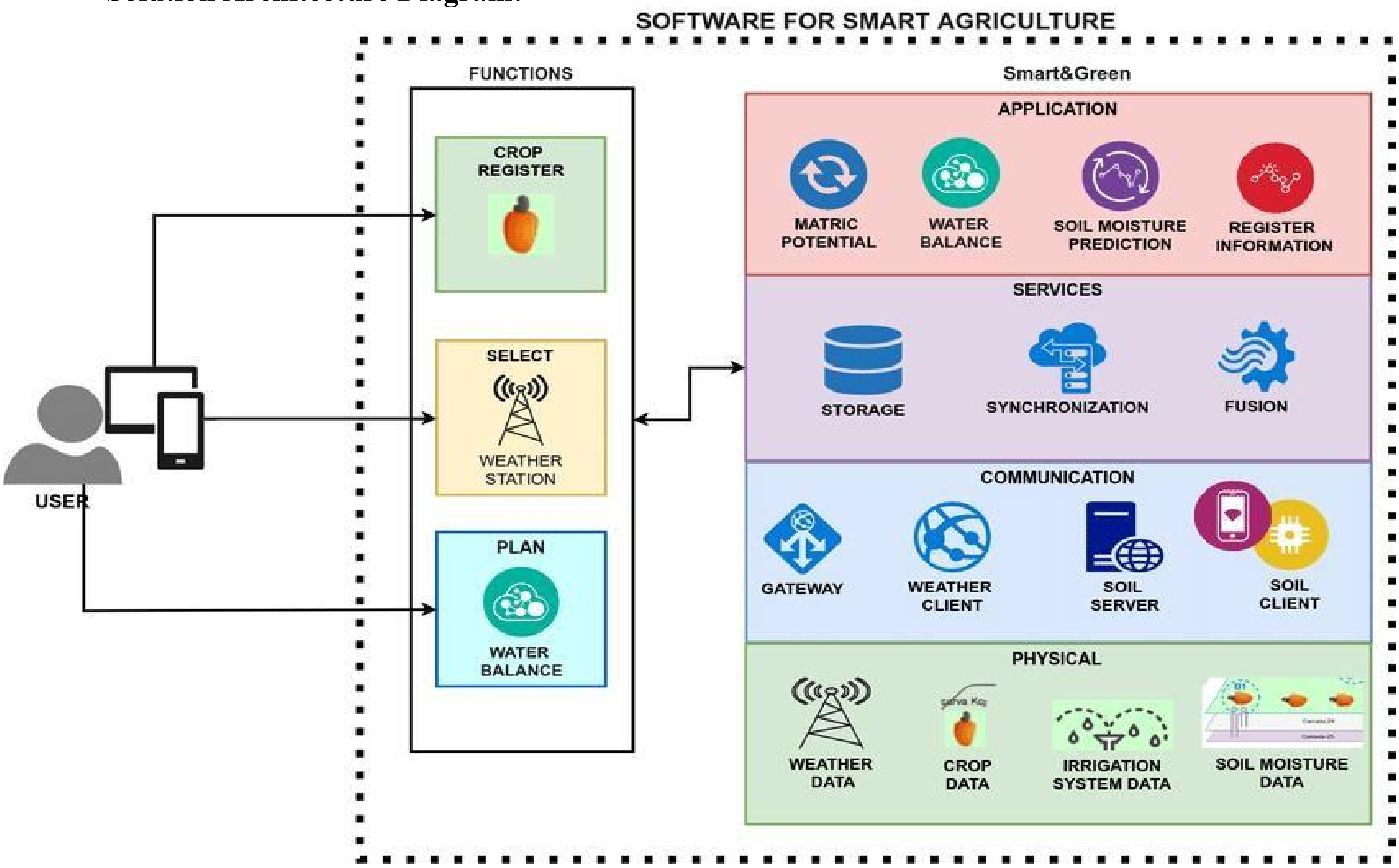
Solution architecture is a complex process

– with many sub-processes

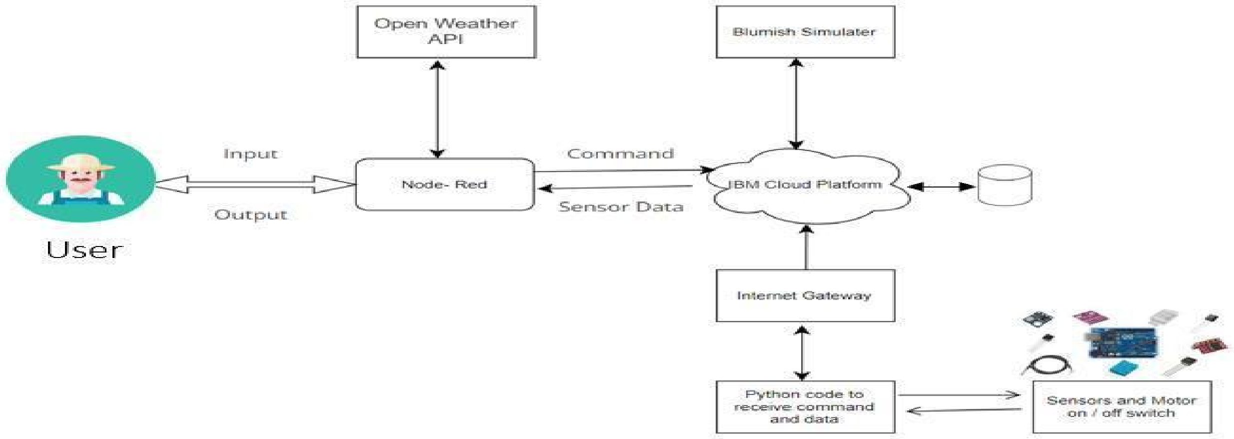
– that bridges the gap between business problems and technology solutions. Its goals are to:

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

Solution Architecture Diagram:



Technical Architecture



1. User Interface	How user interacts with application e.g. Web	MIT App Inventor
2. Application Logic-1	Logic for a process in the application	Python
3. Application Logic-2	Logic for a process in the application	IBM Watson IOT service
4. Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5. Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6. Cloud Database	Database Service on Cloud	IBM Cloud
7. File Storage	File storage requirements	IBM Block Storage or Other Storage
8. External API-1	Purpose of External API used in the application	Open Weather API
9. Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration:	Local, Cloud Foundry.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	Sensitive and private data must be protected from their production until the decision-making and storage stages.	Node-Red, Open weather App API, MIT app Inventor
3.	Scalable Architecture	Scalability is a major concern for IoT platforms. It has been shown that different architectural choices Of IoT platform affect system scalability and that automatic real time decision-making is feasible in an environment composed of dozens of thousand.	Technology used

5.3 User Stories

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Interfacing sensors and Motor Pump and IBM colud	USN-1	Develop a python Code to Interface Sensors and Motor Pump and IBM cloud.	20	High	Yokesh M (Leader)
Sprint-2	Node-Red	USN-2	Develop a web Application Using a NodeRed	20	High	Anil Shein S J (TM-3)
Sprint-3	Mobile Application	USN-3	Develop a mobile Application using MIT-App	20	High	Sachin Raj G (TM-1)
Sprint-4	Integration & Testing	USN-4	Integrating Python Script, Web application & Mobile App	20	High	Antony Amose I (TM-2) & Vigneshwaran s (TM-4)

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint planning & Estimation

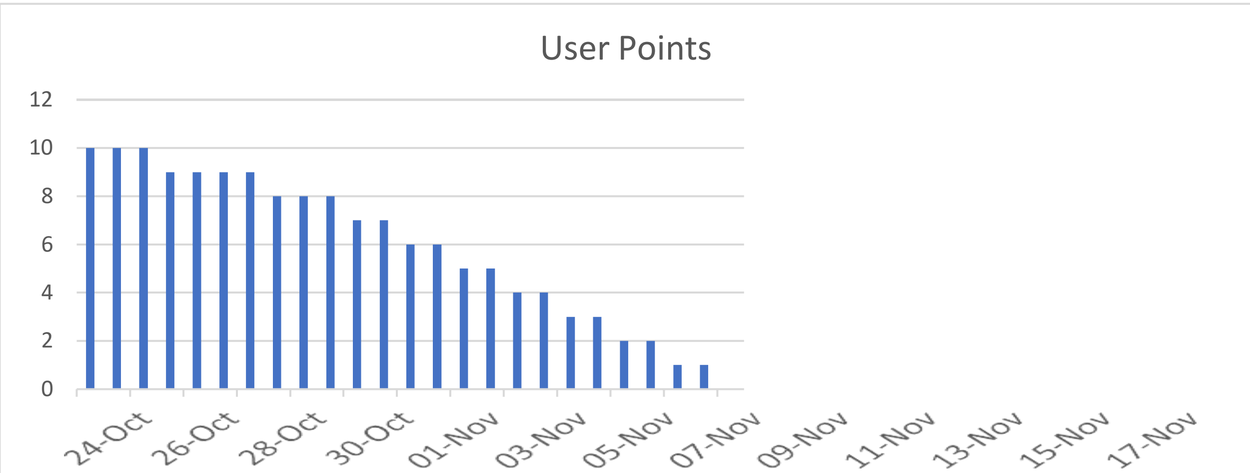
Title	Description	Date
Literature survey on the selected project & InformationGathering	Collect the relevant information on project use case, refer the existing solutions, technical papers, research publications etc.	10 Oct 2022
Prepare Empathy Map	Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements.	23 Sept 2022
Organizing the brainstorming session	Participate in Brainstorming & Ideation, list the ideas and shortlist the top 3 ideas.	17 Oct 2022
Proposed solution	proposed solution document, which includes the novelty, feasibility of idea, business model,social impact, scalability of solution, etc.	12 Oct 2022
ProblemSolution Fit	Prepare problem - solution fit document.	13 Oct 2022
SolutionArchitecture	Prepare Solution Architecture.	19 Oct 2022
Customer Journey	customer journey maps to understand the user interactions & experiences with the application (entry to exit).	14 Oct 2022
Technology Architecture	Prepare Technology Architecture.	15 Oct 2022
Milestone & Activity List	PrepareMilestone & Activity List.	17 Nov 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Organize a demonstration session for each sprint, review the code and share your inputs.	18 Nov 2022

ACTIVITY TITLE	ACTIVITY DESCRIPTION
Live Session	Attending the Live Session to Understand Project.
Assigning The Task	Splitting the work Equally to the Team Members & Assigning theTask.
Understanding The Project & Requirements	Discussing With Team Members About the Project.
Developing Code	Interfacing The sensors With Necessary Code and Storing The dataIn IBM Cloud.
Node Red	Connecting And Working on Node -Red.
MIT- App Inventor	Creating User Interface.
Report	Preparing Report for The Project Including All the Steps ToComplete

6.2 Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	11 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	17 Nov 2022

6.3 Report from JIRA



7. CODING & SOLUTIONING

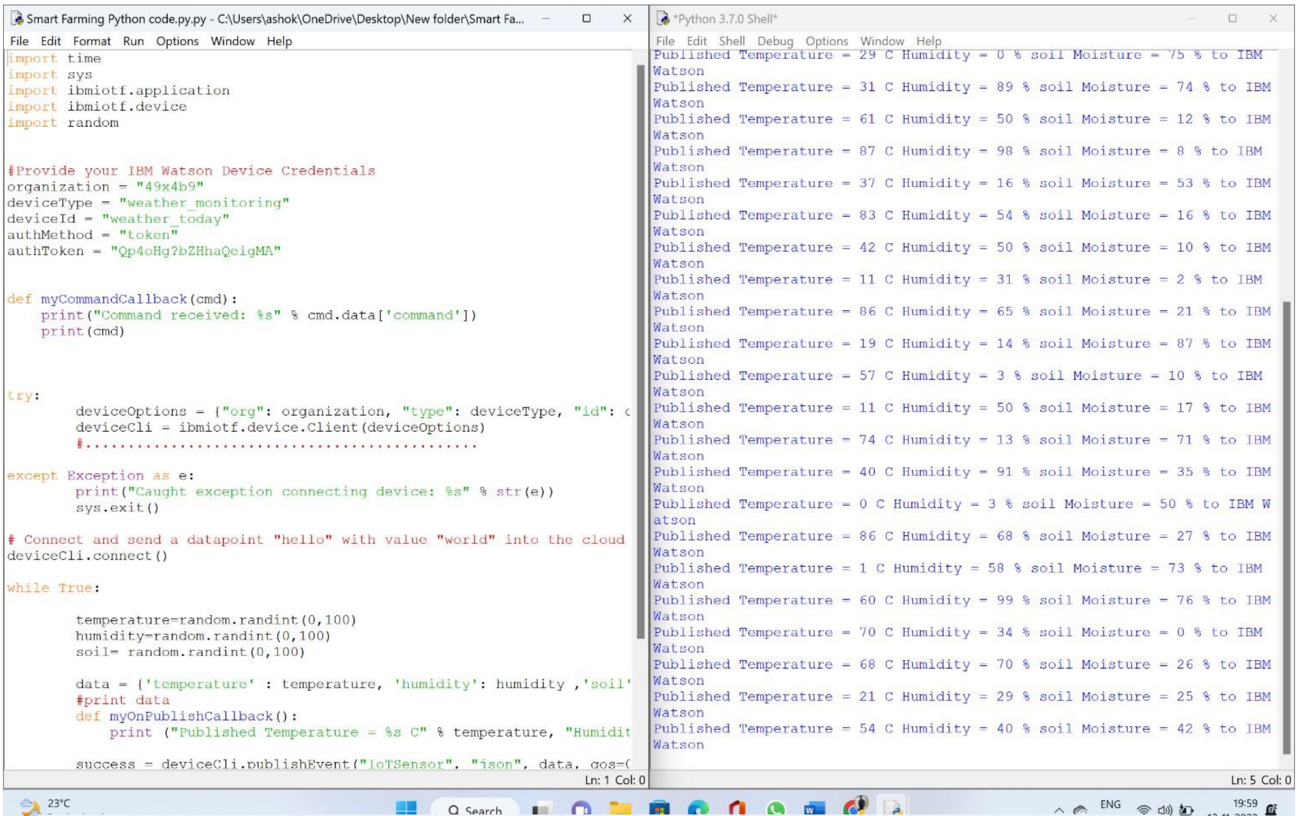
7.1 Feature 1

smartfarmer.py - C:\Users\Yokesh\Downloads\smartfarmer.py (3.7.4)

File Edit Format Run Options Window Help

```python
import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig ={
        "identity": {
                "orgId": "osr5tj",
                "typeId": "EPS07_Controller",
                "deviceId": "0707"
        },
        "auth":{
                "token": "@DOm9&k8Fehe_ntg5n"
        }
}
client = wiotp.sdk.device.DeviceClient (config=myConfig, logHandlers=None)
client.connet()


def myCommandCallback(cmd):
        print("Message received form IBM IoT Platform: %s" % cmd.data['command'])
        m=cd.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.randlint(-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()
```

Ln: 25 Col: 21



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

File Edit Format Run Options Window Help

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


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


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


try:
        deviceOptions = {"org": organization, "type": deviceType, "id": c
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #..............................................

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

# Connect and send a datapoint "hello" with value "world" into the cloud
deviceCli.connect()

while True:

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

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

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=(
```

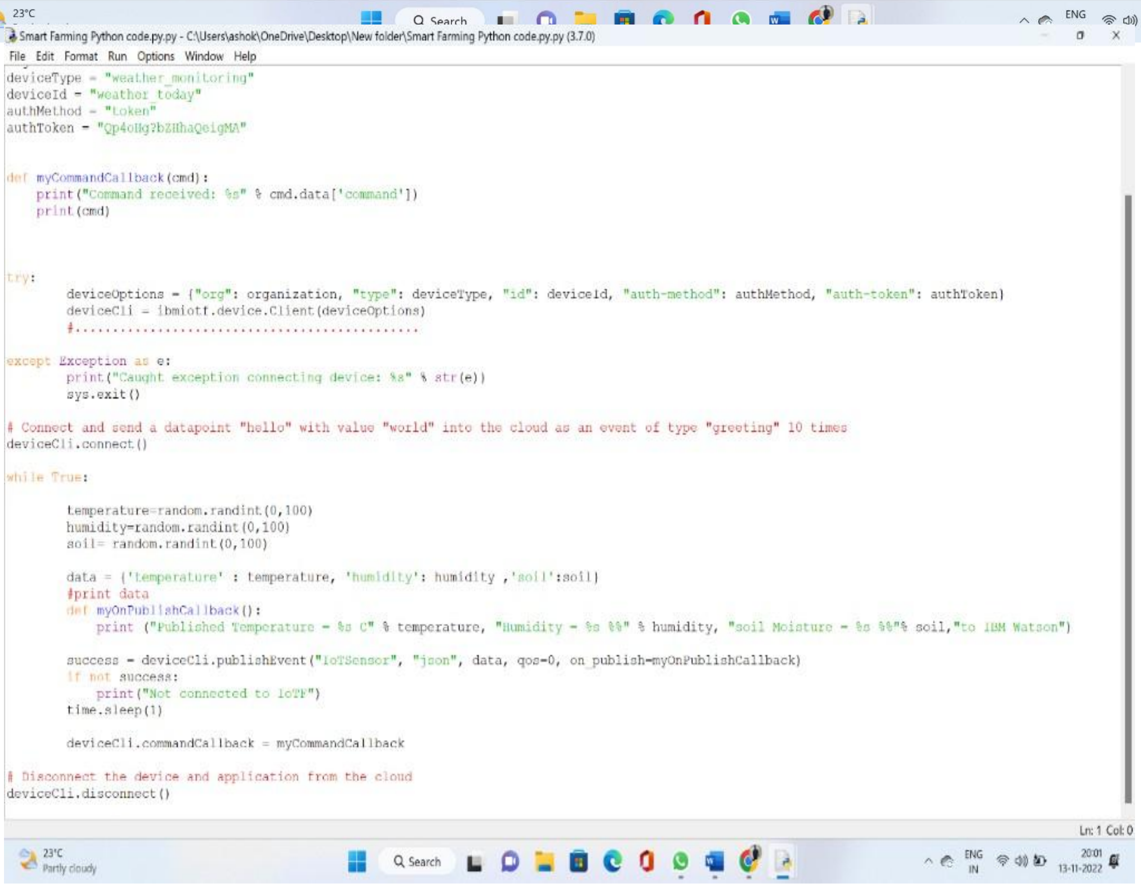
Ln: 1 Col: 0

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

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

Ln: 5 Col: 0



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

File Edit Format Run Options Window Help

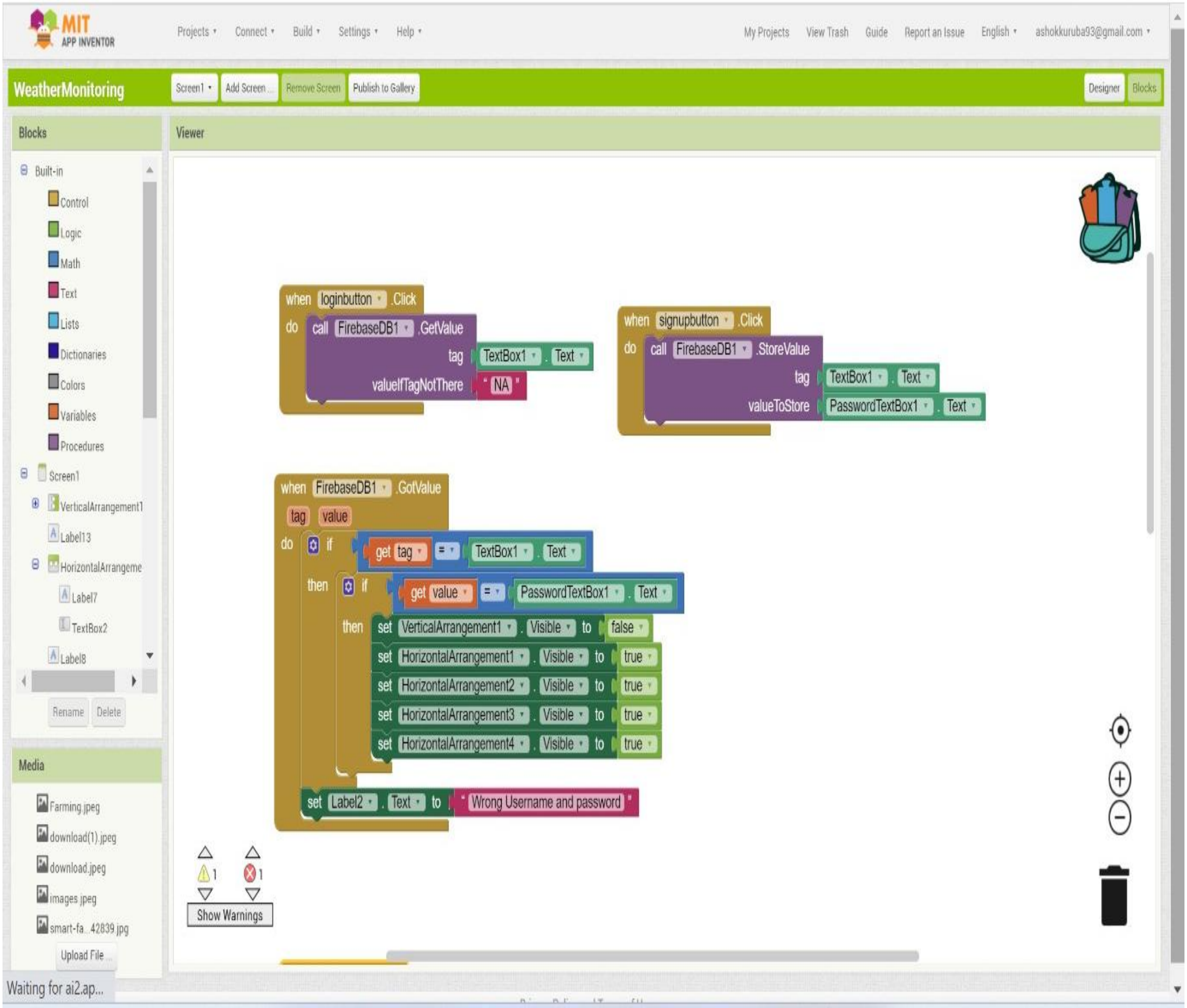
```
deviceType = "weather_monitoring"
deviceId = "weather_today"
authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"


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


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

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

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

while True:

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

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

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

        deviceCli.commandCallback = myCommandCallback

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

Ln: 1 Col: 0

7.2 Feature 2

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

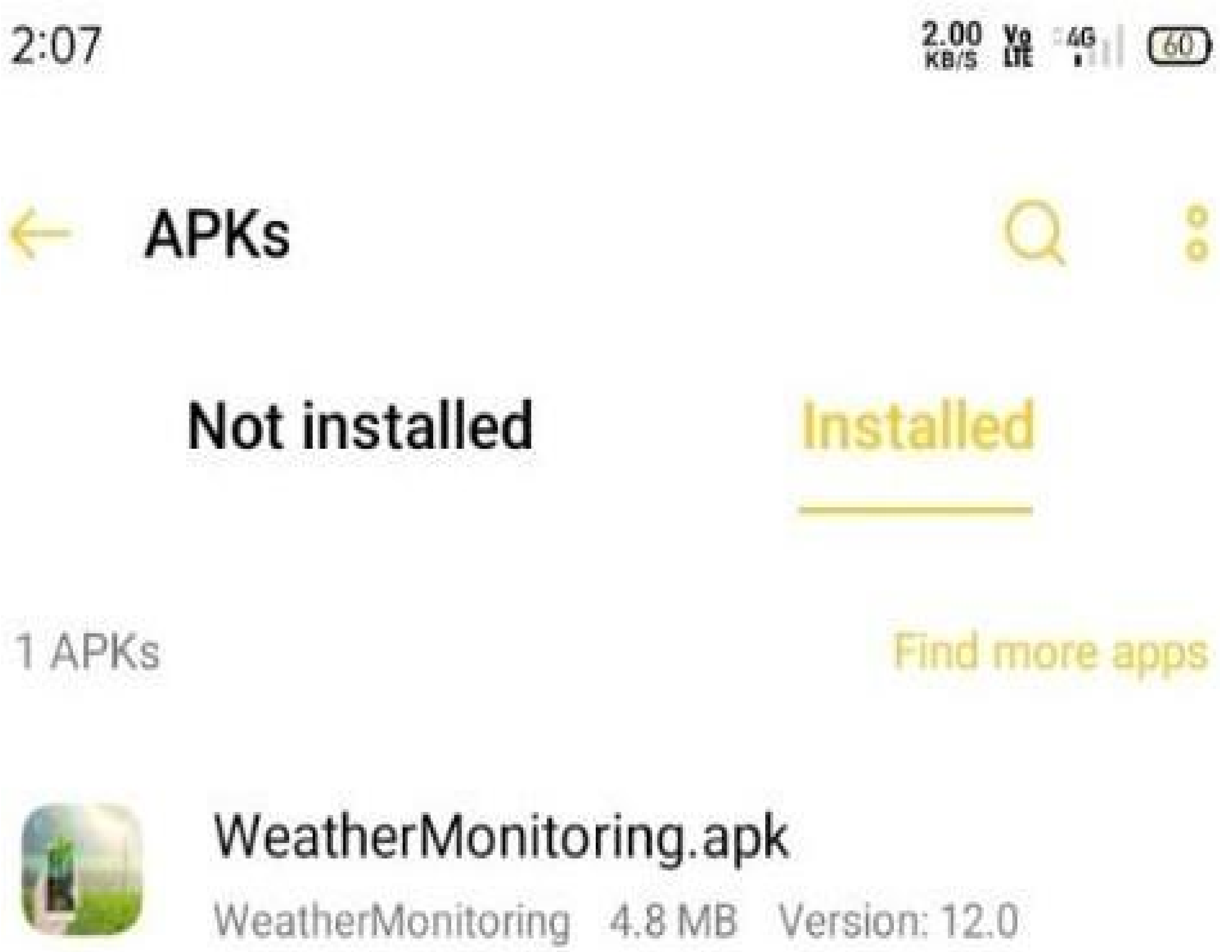
These are the blocks in the second page of the mobile application.



8. TESTING

8.1 Test Cases

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

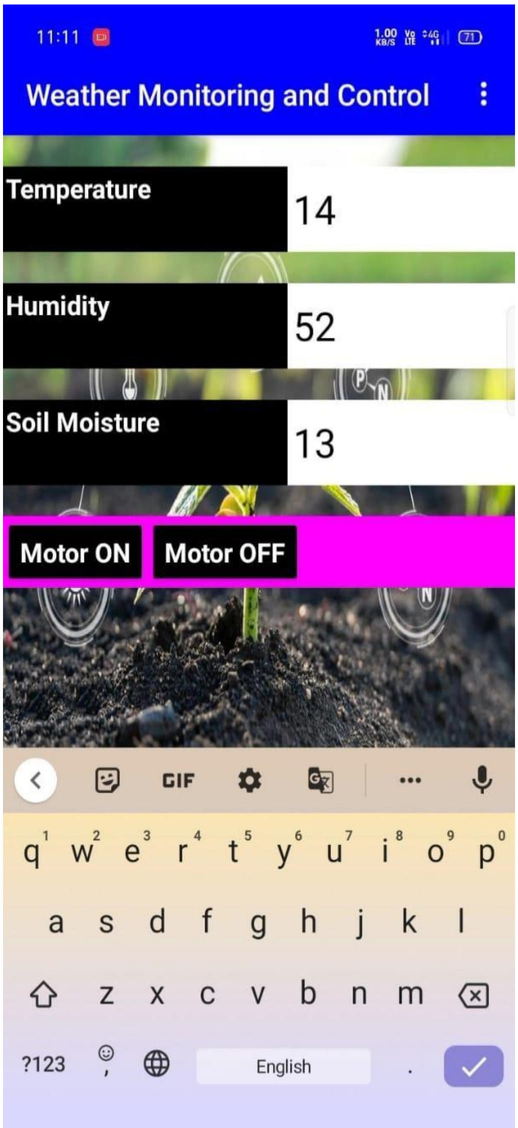


Step-2: After successful installation we can find app icon in our mobile as shown below.

Step-3: After clicking the app icon it ask the user need to create username and password.so give username and password and click the signup button. The user can see interface like these as shown below.

8.2 User Acceptance Testing

After successful login. The next page will be open. In that page we can see the real time temperature , humidity and soil moisture reading and motor ON and motor OFF control button also as shown below.



we are successfully created the IOT enabled smart farming application.

9. RESULTS

9.1 Performance Metrics

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

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

All the data like climatic conditions and changes in them, soil or crop conditions everything can be easily monitored.

Risk of crop damage can be lowered to a greater extent.

Many difficult challenges can be avoided making the process automated and the quality of crops can be maintained.

The process included in farming can be controlled using the web applications from anywhere, anytime.

DISADVANTAGES:

Smart Agriculture requires internet connectivity continuously, but rural parts cannot fulfill this requirement.

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

IoT devices need much money to implement.

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

12. FUTURE SCOPE: In future due to more demand of good and more farming in less time, for betterment of the crops and reducing the usage of extravagant resources like electricity and water IoT can be implemented in most of the places.

13. APPENDIX

Source Code:

```
import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig ={
  "identity": {
    "orgId": "osr5tj",
    "typeId": "EPS07_Controller",
    "deviceId": "0707"
  },
  "auth":{
    "token": "@DOm9&k8Fehe_ntg5n"
  }
}
client = wiotp.sdk.device.DeviceClient (config=myConfig, logHandlers=None)
client.connect()


def myCommandCallback(cmd):
  print("Message received form IBM IoT Platform: %s" % cmd.data['command'])
  m=cd.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.randlint(-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()
```

GitHub:

Name	GitHub (User Name)
Team Leader(Yokesh M)	Yokeshm007
Team Member(Sachin Raj G)	Rajsa321
Team Member(Antony Amose I)	Antonyamose
Team Member(Anil Shebin S J)	AnilShebin
Team Member(Vigneshwaran S)	Vigneshwaran022513

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-35958-1660290928

Project Demonstration Video Link:

https://youtu.be/uO_IbpGI0jI