**Project Report**

**Title:** IoT Based Smart Crop Protection System For Agriculture

**TEAM ID: PNT2022TMID43619**

**Presented By**

S.Senthilkumar

B.Shathika

A.Sneka

G.Akash

Table of Contents

1. **INTRODUCTION**
   1. Project Overview
   2. Purpose
2. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   1. Functional requirement
   2. Non-Functional requirements
5. **PROJECT DESIGN**
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
6. **PROJECT PLANNING & SCHEDULING**
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   1. Feature 1
   2. Feature 2
   3. Database Schema (if Applicable)
8. **TESTING**
9. **RESULTS**
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

GitHub & Project Demo Link

1)INTRODUCTION:

1.1 Project Overview

This is a IOT Based Smart Crop Protection System for Agriculture that can measure soil moisture, humidity and temperature conditions for agriculture using Watson IoT services. 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 moisture, Humidity and Temperature data obtained from sensors we make use of IBM IoT Simulator which can transmit these parameters as required.

* **Project Requirements** : Node-RED, IBM Cloud, IBM Watson IoT, Node.js, IBMDevice, 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 is grooming nowadays because it 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. Thus this approach towards Agriculture will help the farmers to get better yield at low cost and without much usage of Resources .

2) LITERATURE SURVEY:

2.1 Existing Problem

* Agriculture is a field which forms the basis of our economy. Yet it faces a lot of problems in terms of availability of resources, Irrigation, increasing rate of Pesticides, Climatic disasters, Insects which ruin the crops and makes a huge loss this sector.
* In agriculture water is needed for the crops for their growth. If the Soil gets dry it is necessary to supply water. But sometime if the farmer doesn't visit the field it is not possible to know the condition of soil.
* Sometimes over supply of water or less supply of water affects the growth of crops.
* Sometimes if the weather/temperature changes suddenly it is necessary to take certain actions.
* Specific crops grow better in specific conditions, they may get damaged due to bad weather.

2.2 References

[1] Hanshi Wang; Jingli Lu; Lizhen Liu; Wei Song; Zhaoxia Wang; “Community Alarm System Design Based On MCU And GSM” Year: 2015.

[2] Markus Borschbach; Navya Amin, “Quality Of Obstacle Distance Measurement Using Ultrasonic Sensor And Precision Of Two Computer Vision-Based Obstacle Detection Approaches” Year: 2015, 2015 International Conference on Smart Sensors and Systems (ICSSS).

[3] Mustapha, Baharuddin, AladinZayegh, and Rezaul K. Begg. “Ultrasonic and Infrared Sensors Performance in A Wireless Obstacle Detection System” Artificial Intelligence, Modeling and Simulation (AIMS), 2013 1st International Conference on. IEEE, 2013.

[4] Dr. Wilson, “Electric Fence” Handbook of Texas, Project report published by the Texas State Historical Association. August 4, 2011.

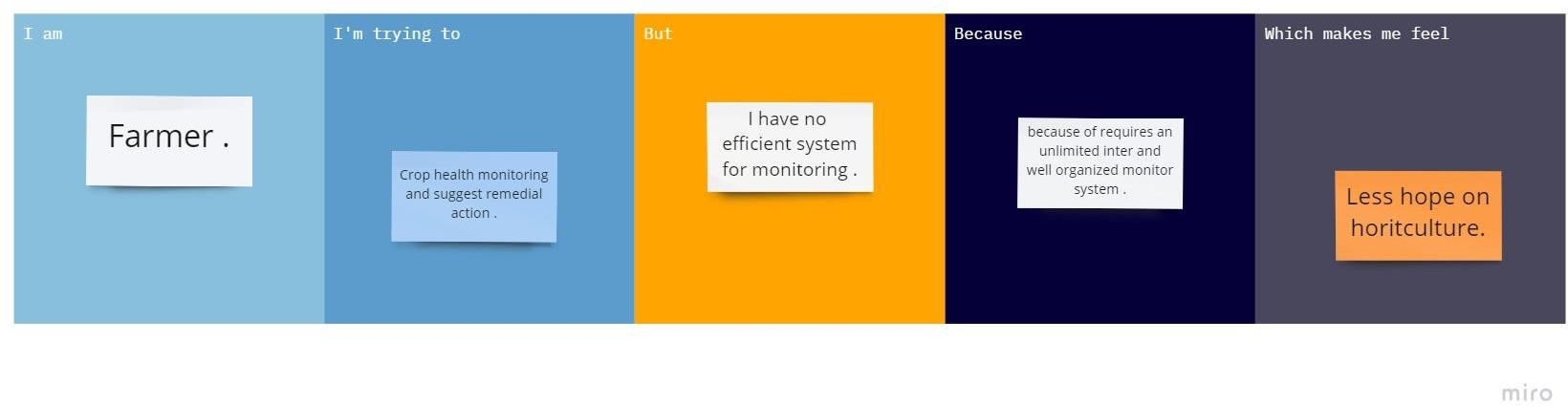
[5] T. Mohammad, “Using Ultrasonic and Infrared Sensors for Distance Measurement” World Academy of Science, Engineering and Technology, pp. 293-298, 2009.

[6] B. Hamrick, T. Campbell, B. Higginbotham, and S. Lapidge, “Managing an invasion: effective measures to control wild pigs,” 2011.

[7] A. R. Tiedemann, T. Quigley, L. White, W. Lauritzen, J. Thomas, and M. McInnis, “Electronic (fenceless) control of livestock,” US Department of Agriculture Forest Service Pacific Northwest Research Station PNW-RP-510, 1999.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement (PS)** | **I am (Customer)** | **I am trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | Farmer | Monitoring the growing condition | It involves risk on related equipment and understand the use of technology | Requires more knowledge and skills | Irritated |
| PS-2 | Farmer | Smart and precision irrigation | Climates change stop increased maintenance of channels | Purchasing and installing costs high | Suitable for mass crop protection |

2.3 Problem Statement Definition

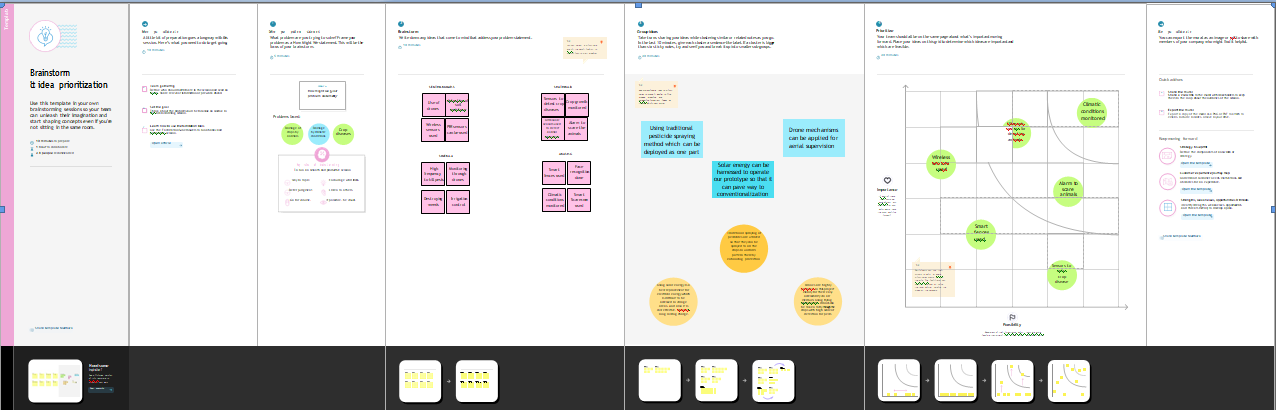


3) IDEATION & PROPOSED SOLUTION:

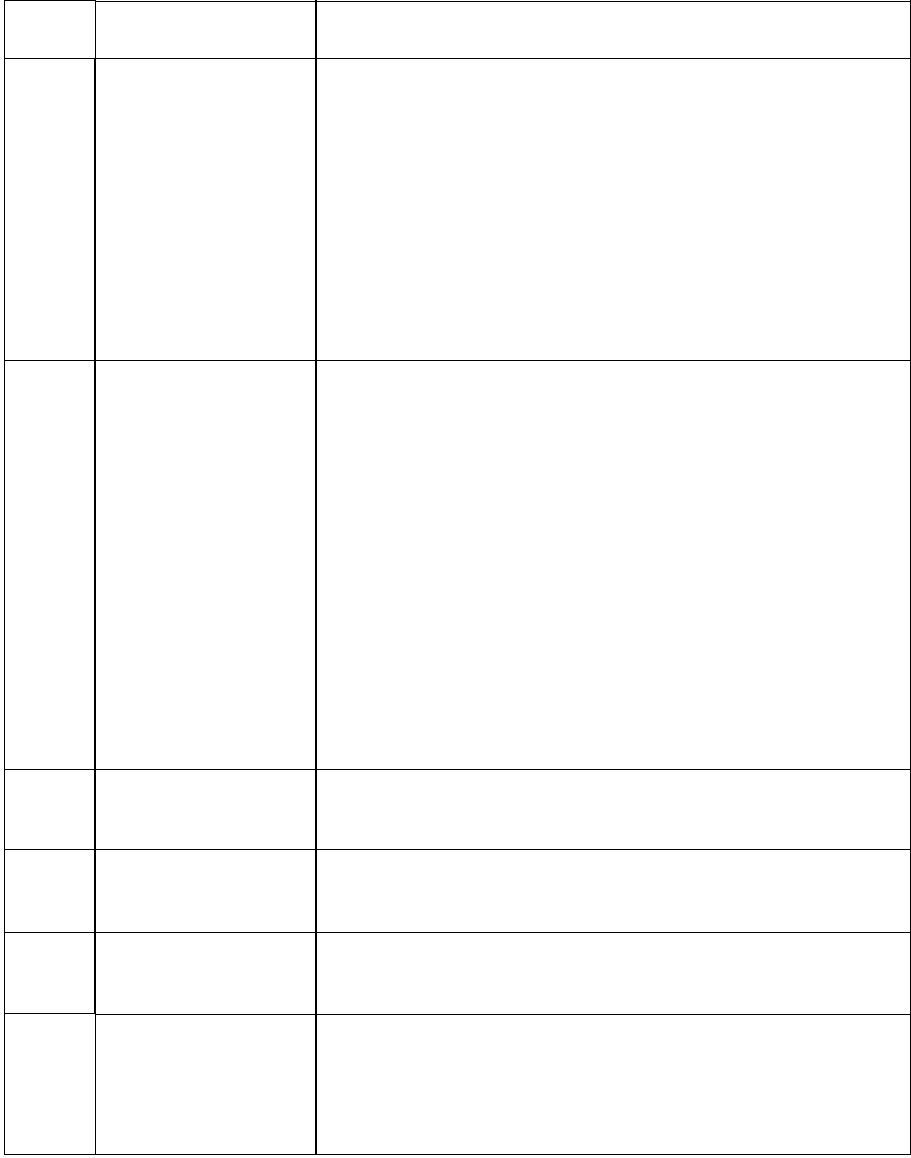
3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



3.3 Proposed Solution



**S.No. Parameter**  **Description**

* Crops are not irrigated properly due to insufficient labour forces.
* Improper maintenance of crops against various
* environmental factors such as temperature climate, topography and soil quality which results in crop
* destruction.
* Lack of knowledge among farmers in usage of fertilizers and hence crops are affected due to high ammonia, urea, potassium and high PH level fertilizers.
* Requires protecting crops from Wild animals attacks, birds and pests.

Problem Statement

1. (Problem to be solved

* Moisture sensor is interfaced with Arduino Microcontroller to measure the moisture level in soil and relay is used to turn ON and OFF the motor pump for managing the excess water level. It will be updated to authorities through IOT.
* Temperature sensor connected to microcontroller is used to monitor the temperature in the field. The optimum temperature required for crop cultivation is maintained using sprinklers.
* IOT based fertilizing methods are followed, to minimize the negative effects on growth of crops while using fertilizers.
* Image processing techniques with IOT is followed for crop protection against animal attacks.
* Requires protecting crops from Wild animals attacks, birds and pests.

2.

Idea / Solution description

Automatic crop maintenance and protection using embedded and IOT technology.

3. Novelty / Uniqueness

This proposed system provides many facilities which helps the farmers to maintain the crop field without much loss.

4. Social Impact /

Customer Satisfaction

This prototype can be developed as product with minimum cost with high performance .

5. Business Model

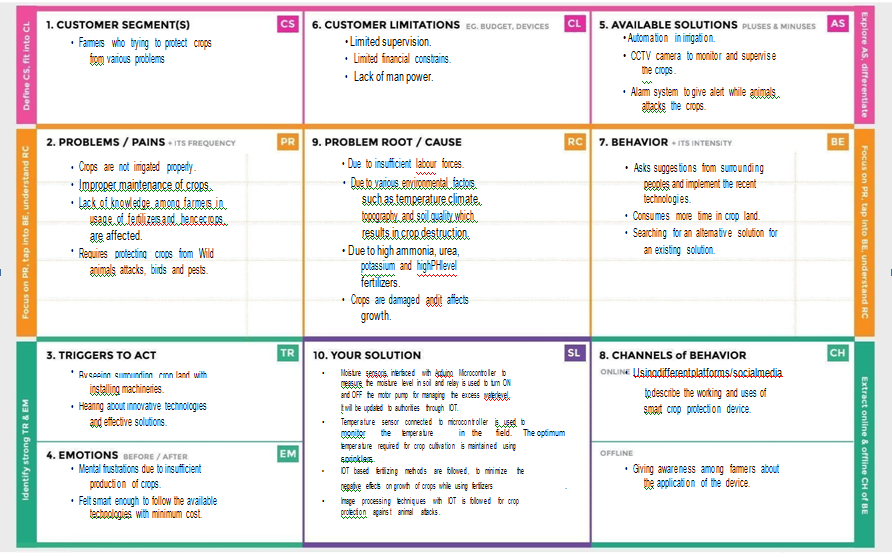
(Revenue Model)

This can be developed to a scalable product by using sensors and transmitting the data through Wireless Sensor Network and Analysing the data in cloud and operation is performed using robots.

Scalability of the Solution

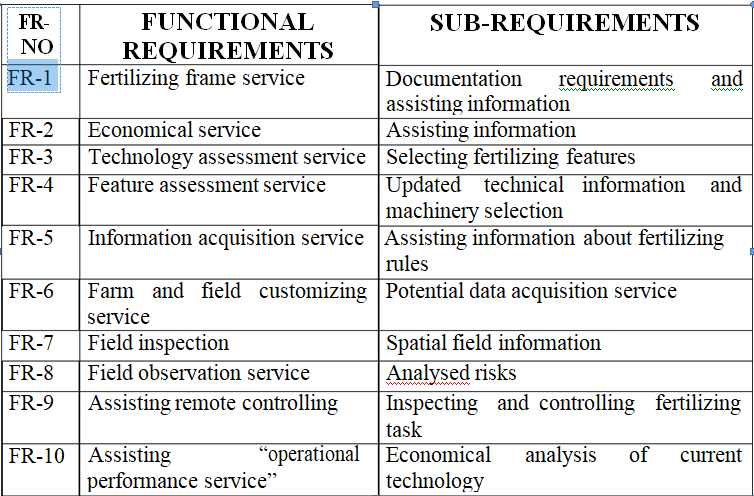
6.

3.4 Problem Solution Fit:

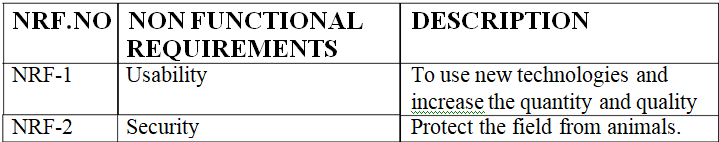


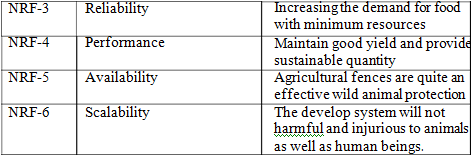
4) REQUIREMENT ANALYSIS:

4.1 Functional Requirement:



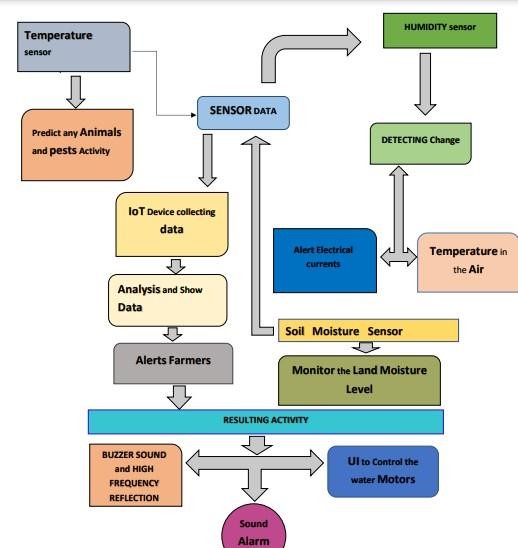
4.2 Non Functional Requirement:





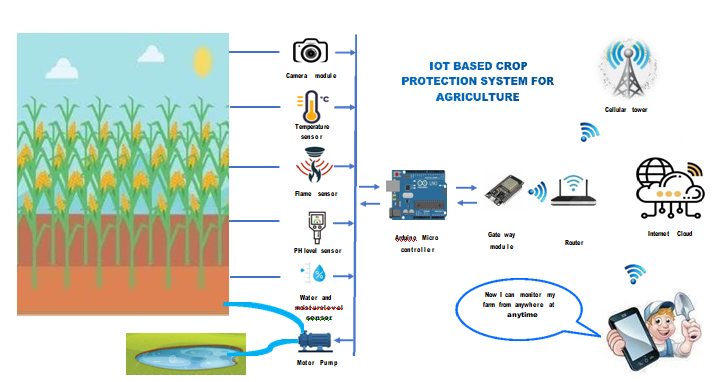
5.PROJECT DESIGN

5.1 Data Flow Diagrams

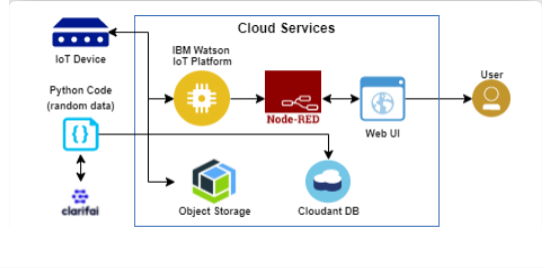


5.2 Solution & Technical Architecture:

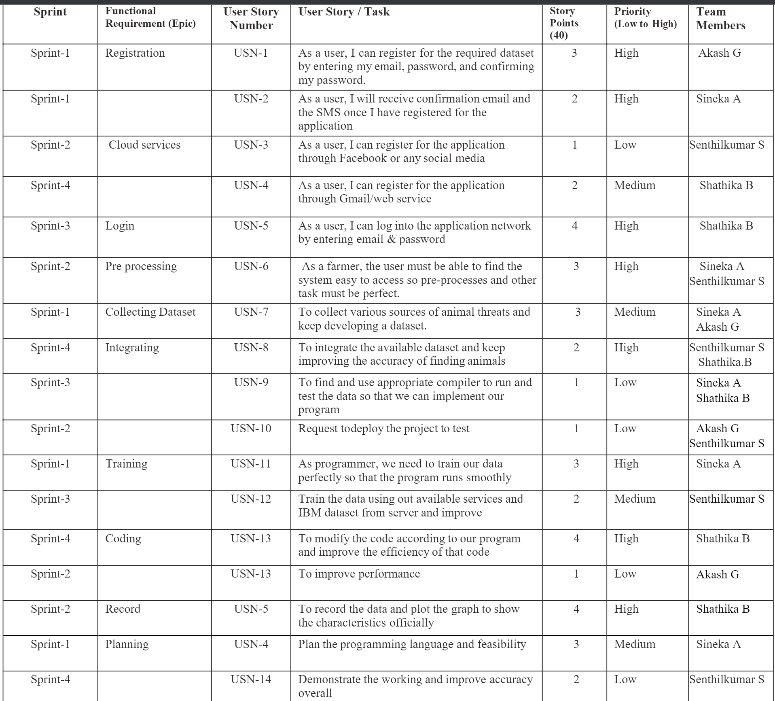
Solution Architecture



Technical Architecture

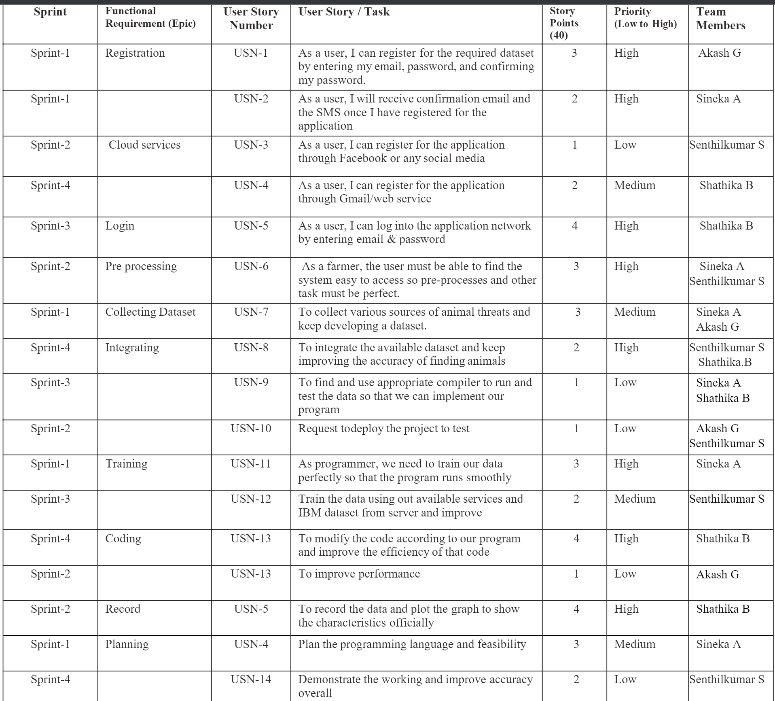


5.3 User Stories:

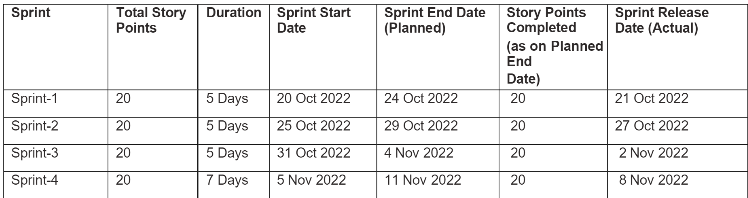


6.PROJECT PLANNING & SCHEDULING

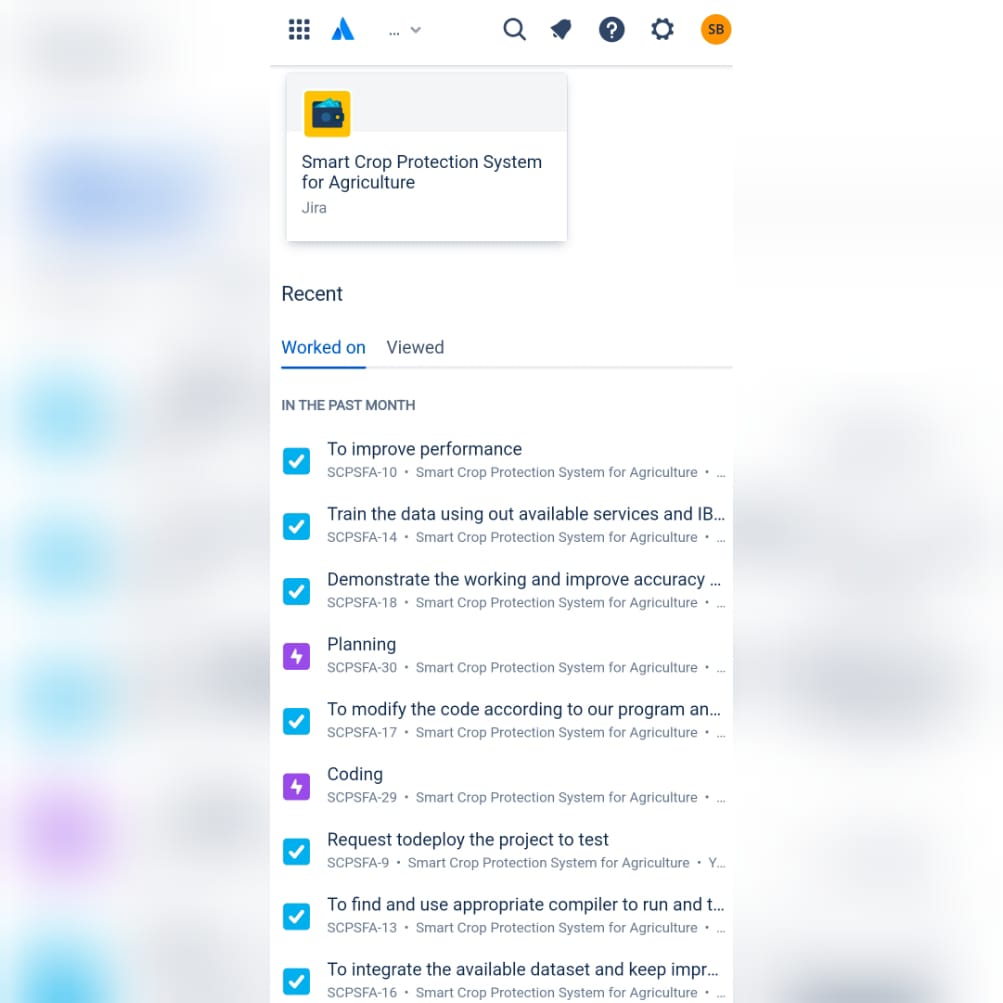
6.1 Sprint Planning & Estimation:

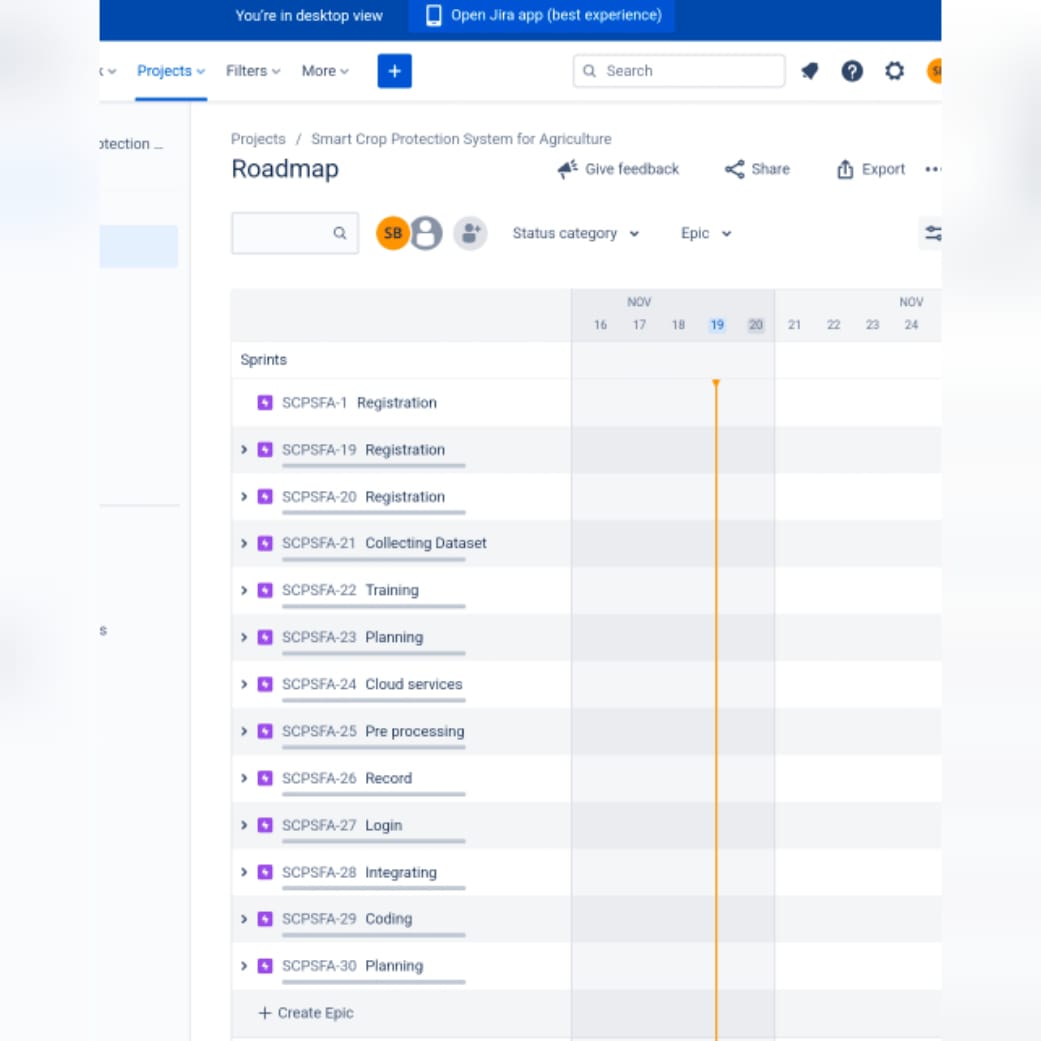


6.2 Sprint Delivery & Schedule:



6.3 Reports from JIRA:





7.CODING & SOLUTIONING:

7.1 Feature 1:

import cv2

import numpy as np import wiot.sdk.device import playsound import random

import time import datetime import ibm\_boto3

from ibm\_botocore.client import Config,

ClientError

#CloudantDB

from cloudant.client import Cloudant

from cloudant.error import CloudantException

from cloudant.result import Result, ResultByKey

from clarifai\_grpc.channel.clarifai\_channel import ClarifaiChannel

from clarifai\_grpc.grpc.api import service\_pb2\_grpc

stub = service\_pb2\_grpc.V2Stub(clarifaiChanne l.get.grpc\_channel())

from clarifai\_grpc.grpc.api import service\_pb2, resource\_pb2

from clarifai\_grpc.grpc.api.status import status\_code\_pb2

#This is how you authenticate metadata = (('authorization', 'key

0620e202302b4508b90eab7efe7475e4'),)

COS\_ENDPOINT = "https://s3.jp- tok.cloud-object- storage.appdomain.cloud"

COS\_API\_KEY\_ID =

"g5d4qO8EIgv4TWUCJj4hfEzgalqEjrDbE8 2AJDWlAOHo"

COS\_AUTH\_ENDPOINT =

"https://iam.cloud.ibm.com/identity/to ken"

COS\_RESOURCE\_CRN =

"crn:v1:bluemix:public:cloud-object- storage:global:a/c2fa2836eaf3434bbc8b 5b58fefff3f0:62e450fd-4c82-4153-ba41- ccb53adb8111::"

clientdb = cloudant("apikey- W2njldnwtjO16V53LAVUCqPwc2aHTLml j1xXvtdGKJBn", "88cc5f47c1a28afbfb8ad16161583f5a", url="https://d6c89f97-cf91-48b7-b14b- c99b2fe27c2f- bluemix.cloudantnosqldb.appdomain.cl oud")

clientdb.connect()

#Create resource

cos = ibm\_boto3.resource("s3", ibm\_api\_key\_id=COS\_API\_KEY\_ID,

ibm\_service\_instance\_id=COS\_RESOURC E\_CRN,

ibm\_auth\_endpoint=COS\_AUTH\_ENDPO INT,

config=Config(signature\_version="oauth "),

endpoint\_url=COS\_ENDPOINT)

def = multi\_part\_upload(bucket\_name, item\_name, file\_path):

try:

print("Starting file transfer for {0} to bucket: {1}\n".format(item\_name, bucket\_name))

#set 5 MB chunks part\_size = 1024 \* 1024 \* 5 #set threadhold to 15 MB

file\_threshold = 1024 \* 1024 \* 15

#set the transfer threshold and chunk size

transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(

multipart\_threshold=file\_threshold, multipart\_chunksize=part\_size)

#the upload\_fileobj method will automatically execute a multi-part upload

#in 5 MB chunks size with open(file\_path, "rb") as file\_data:

cos.Object(bucket\_name, item\_name).upload\_fileobj(

Fileobj=file\_data, Config=transfer\_config)

print("Transfer for {0} Complete!\n".format(item\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be)) except Exception as e:

print("Unable to complete multi-part upload: {0}".format(e))

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data)

command=cmd.data['command']

print(command) if(commamd=="lighton"): print('lighton') elif(command=="lightoff"): print('lightoff') elif(command=="motoron"): print('motoron') elif(command=="motoroff"): print('motoroff')

myConfig = { "identity": { "orgId": "chytun",

"typeId": "NodeMCU", "deviceId": "12345"

},

"auth": {

"token": "12345678"

}

}

client = wiot.sdk.device.DeviceClient(config=my Config, logHandlers=None)

client.connect()

database\_name = "sample"

my\_database = clientdb.create\_database(database\_nam e)

if my\_dtabase.exists():

print(f"'(database\_name)' successfully created.")

cap=cv2.VideoCapture("garden.mp4") if(cap.isOpened()==True):

print('File opened') else:

print('File not found')

while(cap.isOpened()): ret, frame = cap.read()

gray = cv3.cvtColor(frame, [cv2.COLOR\_BGR@GRAY](mailto:cv2.COLOR_BGR@GRAY))

imS= cv2.resize(frame, (960,540)) cv2.inwrite('ex.jpg',imS)

with open("ex.jpg", "rb") as f: file\_bytes = f.read()

#This is the model ID of a publicly available General model. You may use any other public or custom model ID.

request = service\_pb2.PostModeloutputsRequest(

model\_id='e9359dbe6ee44dbc8842ebe9 7247b201',

inputs=[resources\_pb2.Input(data=resou rces\_pb2.Data(image=resources\_pb2.Im age(base64=file\_bytes))

)])

response = stub.PostModelOutputs(request, metadata=metadata)

if response.status.code != status\_code\_pb2.SUCCESS:

raise Exception("Request failed, status code: " + str(response.status.code))

detect=False

for concept in response.outputs[0].data.concepts:

#print('%12s: %.f' % (concept.name, concept.value))

if(concept.value>0.98): #print(concept.name) if(concept.name=="animal"): print("Alert! Alert! animal detected") playsound.playsound('alert.mp3')

picname=datetime.datetime.now().strfti me("%y-%m-%d-%H-%M")

cv2.inwrite(picname+'.jpg',frame)

multi\_part\_upload('Dhakshesh', picname+'.jpg', picname+'.jpg')

json\_document={"link":COS\_ENDPOINT+ '/'+'Dhakshesh'+'/'+picname+'.jpg'}

new\_document = my\_database.create\_document(json\_do cument)

if new\_document.exists(): print(f"Document successfully created.") time.sleep(5)

detect=True

moist=random.randint(0,100)

humidity=random.randint(0,100)

myData={'Animal':detect,'moisture':moi st,'humidity':humidity}

print(myData) if(humidity!=None):

client.publishEvent(eventId="status",ms gFormat="json", daya=myData, qos=0, onPublish=None)

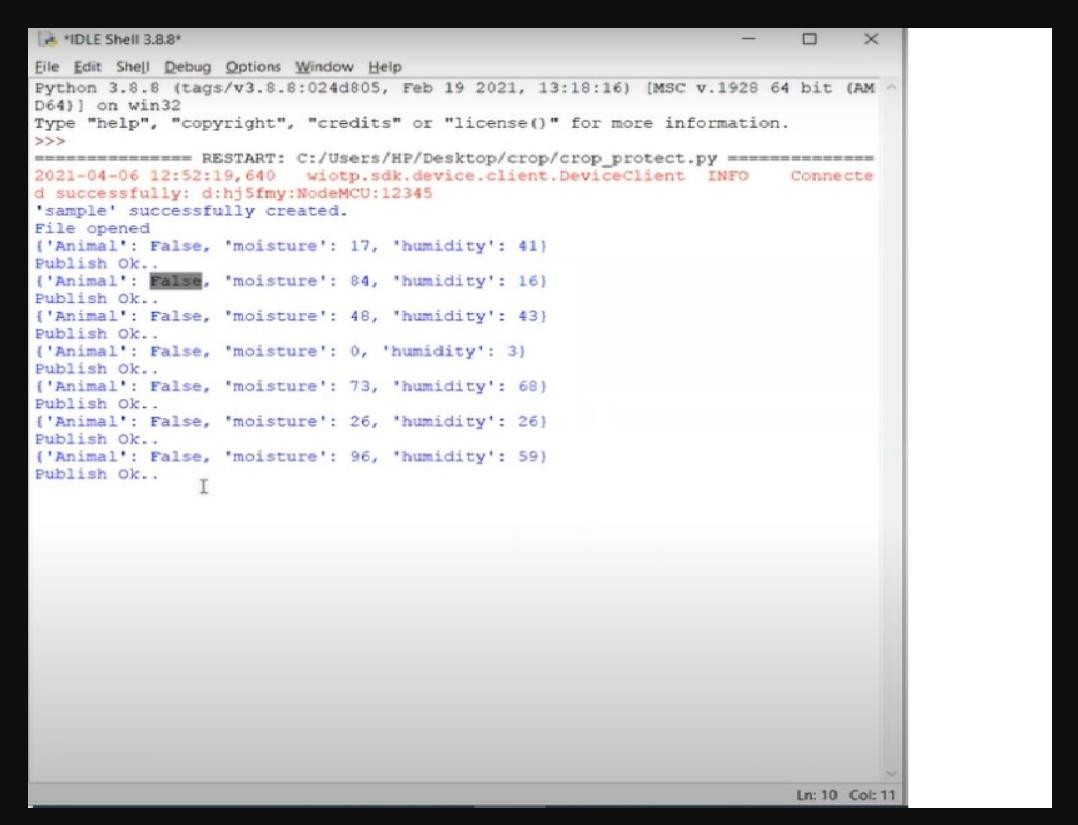
print("Publish Ok..")

client.commandCallback = myCommandCallback

cv2.imshow('frame',imS)

if cv2.waitKey(1) & 0xFF == ord('q'): break

client.disconnect() cap.release() cv2.destroyAllWindows()



7.2 Feature 2:

float x,y,z,temp; void setup()

{

pinMode(8, INPUT); pinMode(5, OUTPUT); pinMode(6, OUTPUT); pinMode(A5, INPUT); pinMode(A4, INPUT); Serial.begin(9600);

}

void loop()

{

x= digitalRead(8);

**y= analogRead(A5); z= analogRead(A4); Serial.println(x); Serial.println(y); Serial.println(z);**

**temp = (double)z / 1024; temp = temp \* 5;**

**temp = temp - 0.5; temp = temp \* 100; if ( (x>0) )**

**{**

**if ((y<550)&&(temp>30))**

**{**

**digitalWrite(5, HIGH); digitalWrite(6, HIGH);**

**}**

**else if((y<550)&&(temp<30))**

**{**

**digitalWrite(5, HIGH); digitalWrite(6, LOW);**

**}**

**else if((y>550)&&(temp>30))**

**{**

**digitalWrite(5, LOW); digitalWrite(6, HIGH);**

**}**

**else if((y>550)&&(temp<30))**

**{**

**digitalWrite(5, LOW); digitalWrite(6, LOW);**

**}**

**}**

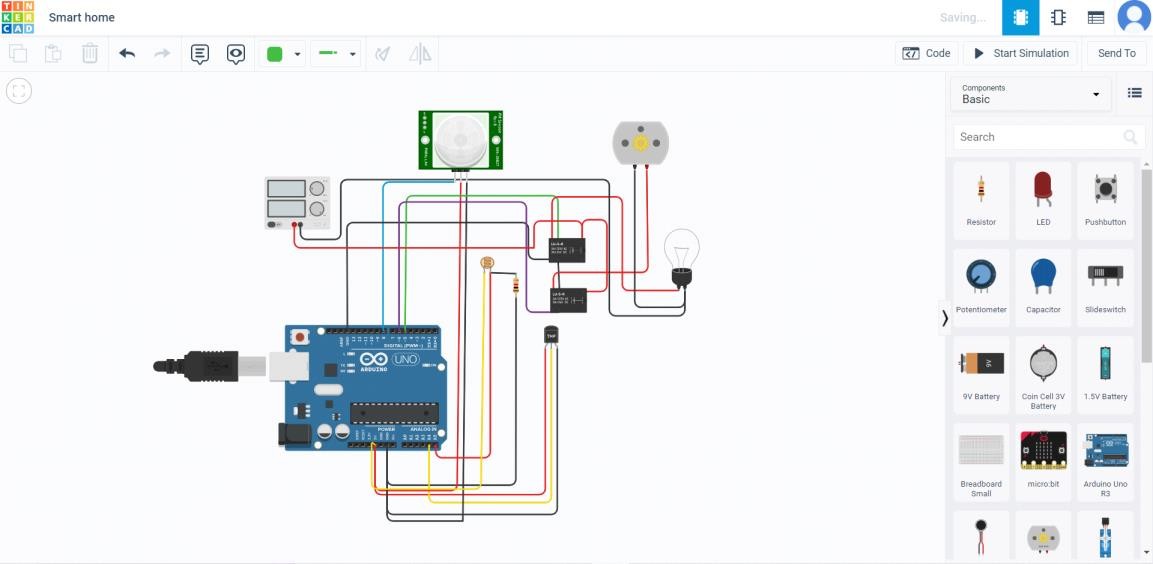
**else**

**{**

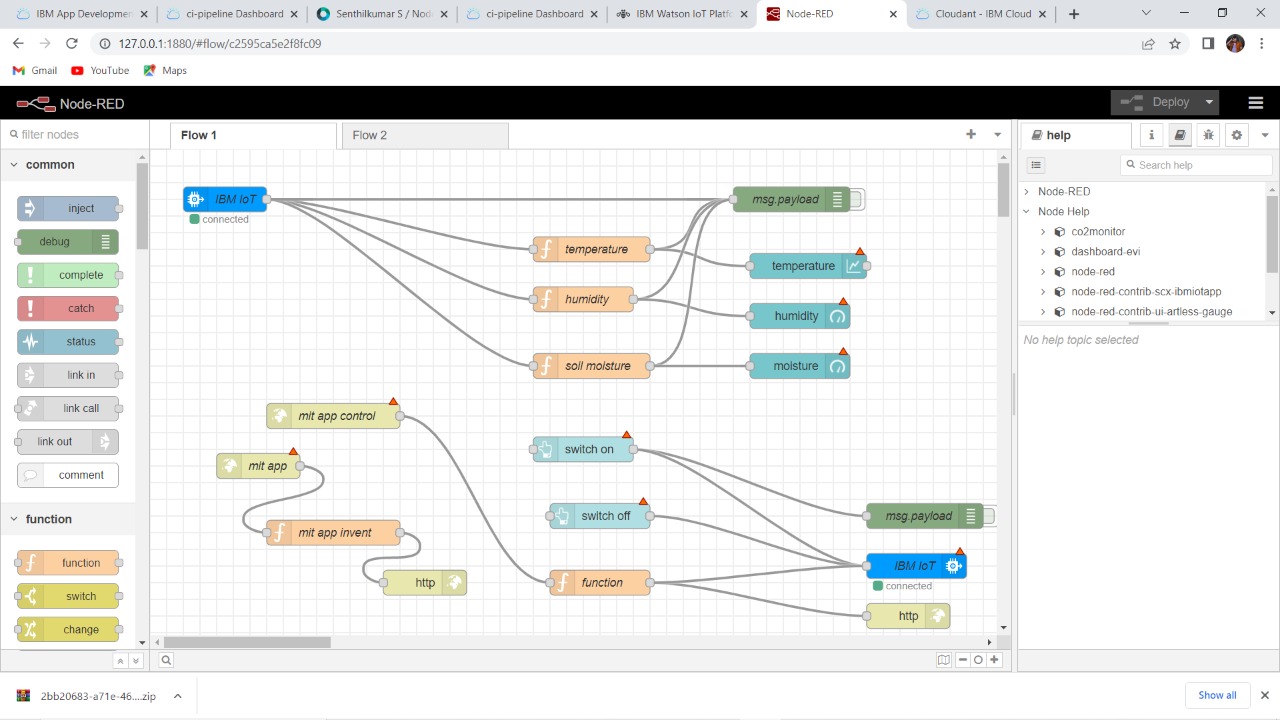
**digitalWrite(5, LOW); digitalWrite(6, LOW);**

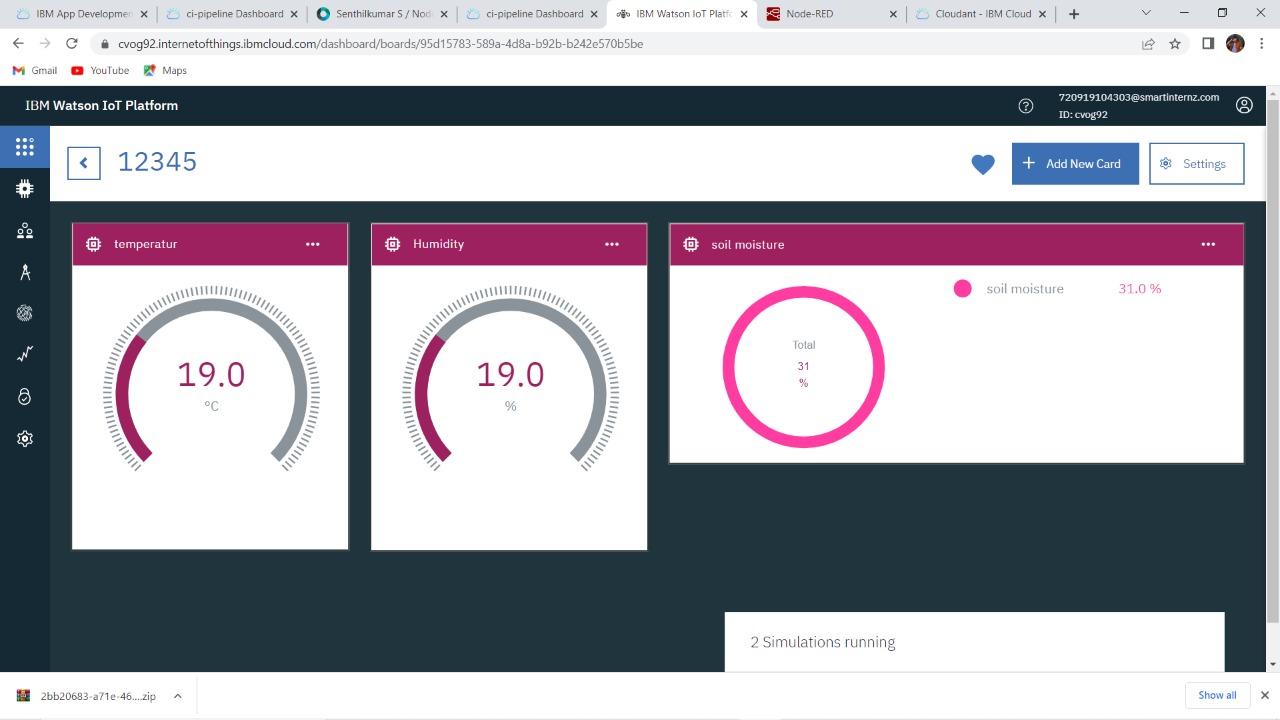
**}**

**}**



8.TESTING:





9) RESULT:

We have successfully built a web-based UI and integrated all the services using Node-RED

10) ADVANTAGES & DISADVANTAGES:

10.1 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.

10.2 Disadvantages

* Smart Agriculture requires internet connectivity continuously, but rural parts can not 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:

An IoT Web Application is built for smart agricultural system using Watson IoT platform, Watson simulator, IBM cloud and Node-RED.

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:

1. Source Code

import time

import sys

import ibmiotf.application # to install pip install ibmiotf

import ibmiotf.device

#Provide your IBM Watson Device Credentials

organization = "hrodmj" #replace the ORG ID

deviceType = "NODEMCU1"#replace the Device type wi

deviceId = "12345"#replace Device ID

authMethod = "token"

authToken = "abhi1234" #Replace the authtoken

def myCommandCallback(cmd): # function for Callback

print("Command received: %s" % cmd.data)

if cmd.data['command']=='motoron':

print("Motor On IS RECEIVED")

elif cmd.data['command']=='motoroff':

print("Motor Off IS RECEIVED")

if cmd.command == "setInterval":

if 'interval' not in cmd.data:

print("Error - command is missing required information: 'interval'")

else:

interval = cmd.data['interval']

elif cmd.command == "print":

if 'message' not in cmd.data:

print("Error - command is missing required information: 'message'")

else:

output=cmd.data['message']

print(output)

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

#..............................................

except Exception as e:

print("Caught exception connecting device: %s" % str(e))

sys.exit()

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

deviceCli.connect()

while True:

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()

1. GITHUB & PROJECT DEMO LINK:

https://github.com/IBM-EPBL/IBM-Project-8099-1658909

<https://drive.google.com/file/d/1PSbA7ijDnkQoL9NpFSb>

WUIs85YqHvObA/view?usp=drivesdk