**IOT BASED SMART CROP**

**PROTECTION SYSTEMFOR AGRICULTURE**

**Team ID :** PNT2022TMID51203

**Team Members**

G.Dhivya

K.Rajeshwari

S.Neha

R.Esther binny vasantha

**INTRODUCTION**

**PROJECT OVREVIEW:**

Crops in farms are many times ravagedby local animalslike buffaloes, cows, goats, birds etc. this leads to huge losses for the farmers. It is not possible for farmers tobarricade entire fields or stay on field 24 hours and guard it.so here we proposeautomatic crop protection system from animals. This is a microcontroller based system using PIC family microcontroller. The microcontroller now sound an alarm to woo the animal away from the field as well as sends SMS to the farmer so that he may about the issue and come to the spot in case the animal don’t turn away by the alarm. This ensures complete safetyof crop from animals thus protecting farmers loss.

**PURPOSE:**

Our main purpose of the project is to develop intruder alert to the farm, to avoid losses due to animal and fire. These intruder alert protect the crop that damaging that indirectly increase yield of the crop. The develop system will not harmful and injurious to animal as well as human beings. Theme of project is to design a intelligent security system for farm protecting by using embedded system.

**LITERATURE SURVEY**

**EXISTING PROBLEM:**

The existing system mainly provide the surveillance functionality. Also these system don’t provide protection from wild animals, especially in such an application area. They also need to take actions based on the type of animal that tries to enter the area, as different methods are adopted to prevent different animals from entering restricted areas. The other commonly used method by farmer in order to prevent the crop vandalization by animals include building physical barriers, use of electric fences andmanual surveillance and various such exhaustive and dangerous method.

**REFERENCES:**

i. Mr.Pranav shitap, Mr.Jayesh redij, Mr.Shikhar Singh, Mr.Durvesh Zagade, Dr. Sharada Chougule. Department of ELECTRONICS AND TELECOMMUNICATION ENGINEERING, Finolex Academy of Management and technology, ratangiri, India.

ii. N.Penchalaiah, D.Pavithra, B.Bhargavi, D.P.Madhurai, K.EliyasShaik,S.Md.sohaib.Assitant Professor, Department of CSE,AITS, Rajampet,India UG Student, Department of CSE,AITS,Rajampet,India.

iii. Mr.P.Venkateswara Rao, Mr.Ch Shiva Krishna ,MR M Samba

SivaReddyLBRCE,LBRCE,LBRCE.

iv. Mohit Korche,Sarthak Tokse, ShubhamShirbhate, Vaibhav Thakre,S. P. Jolhe(HOD). Students , Final Year,Dept.of Electricalengineering,Government

College of engineering,Nagpur head of dept.,Electrical engineering,Government

College of engineering,Nagpur.

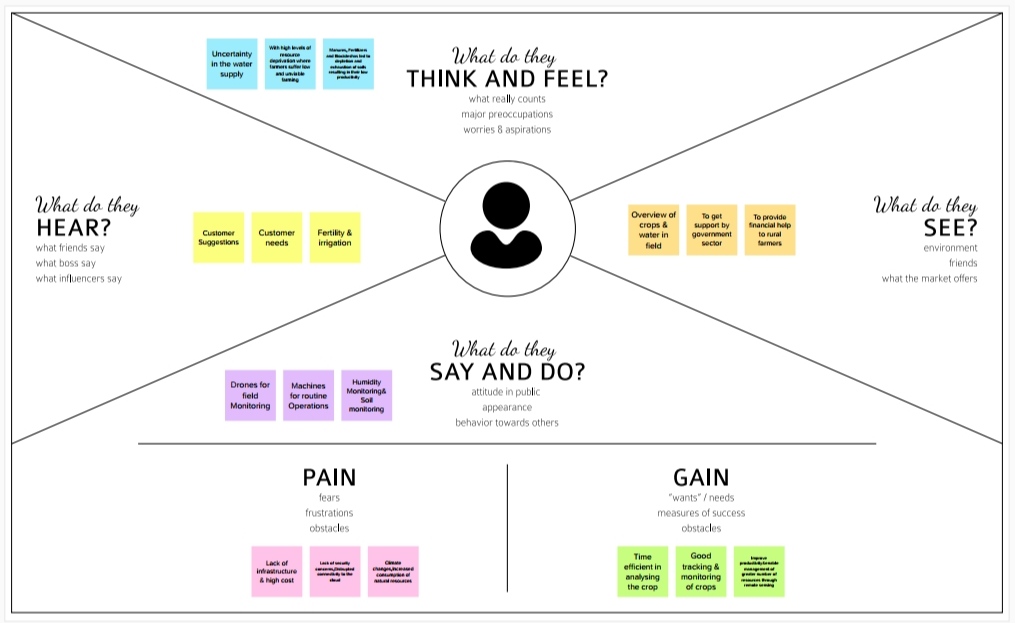
**PROBLEM STATEMENT DEFINITION STATEMENT:**

In the world economyof many Countrydependent upon the agriculture.

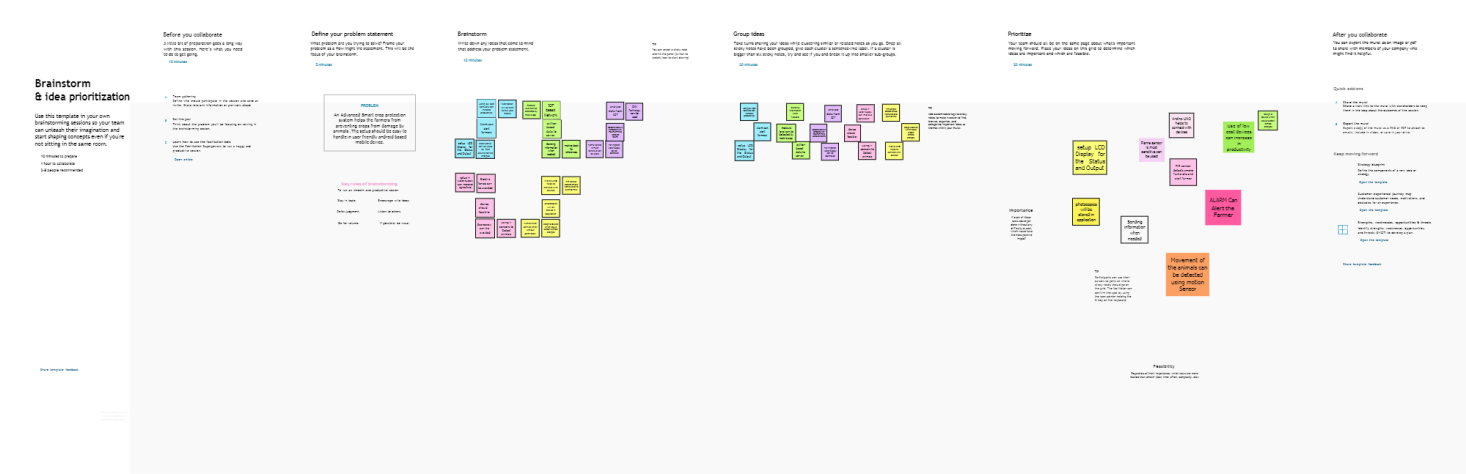
In spite of economic development agriculture is the backbone of the economy. Crops in forms are many times ravaged by local animals like buffaloes, cows, goats, birds and fire etc. this leads to huge loss for the farmers.it is not possible for farmers to blockade to entire fields or stay 24 hours and guard it. Agriculture meets food requirements of the people and produces several raw materialsfor industries. But because of animal interference and fire in agricultural lands, there will be huge loss of crops.Crops will be totally getting destroyed.

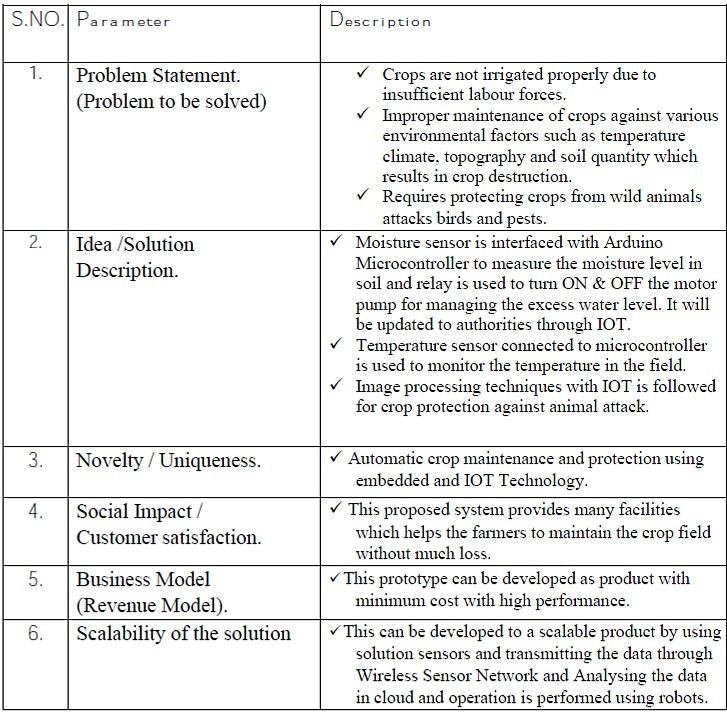
**IDEATION AND PROPOSED SOLUTION**

**EMPATHY M AP CANVAS:**

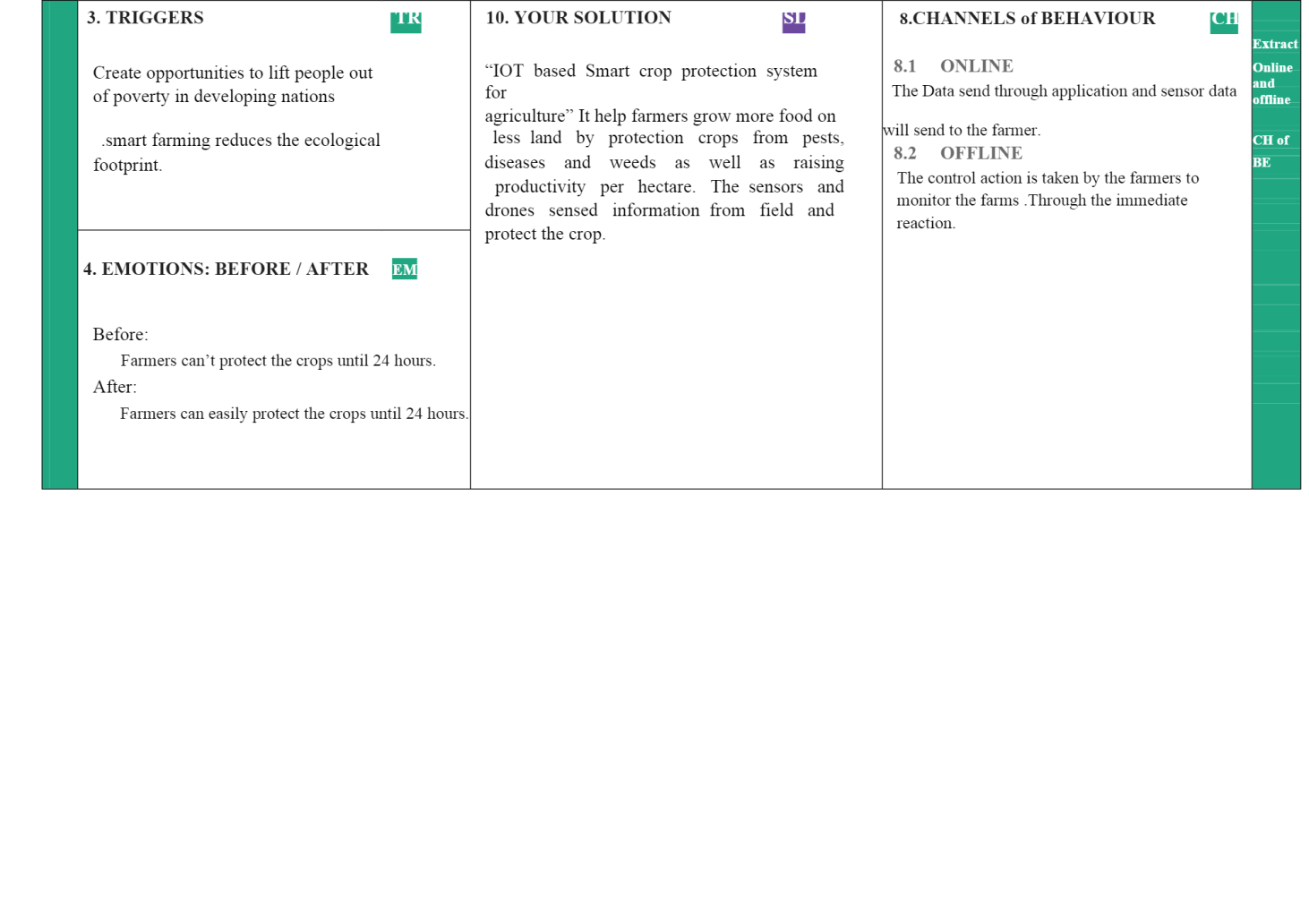


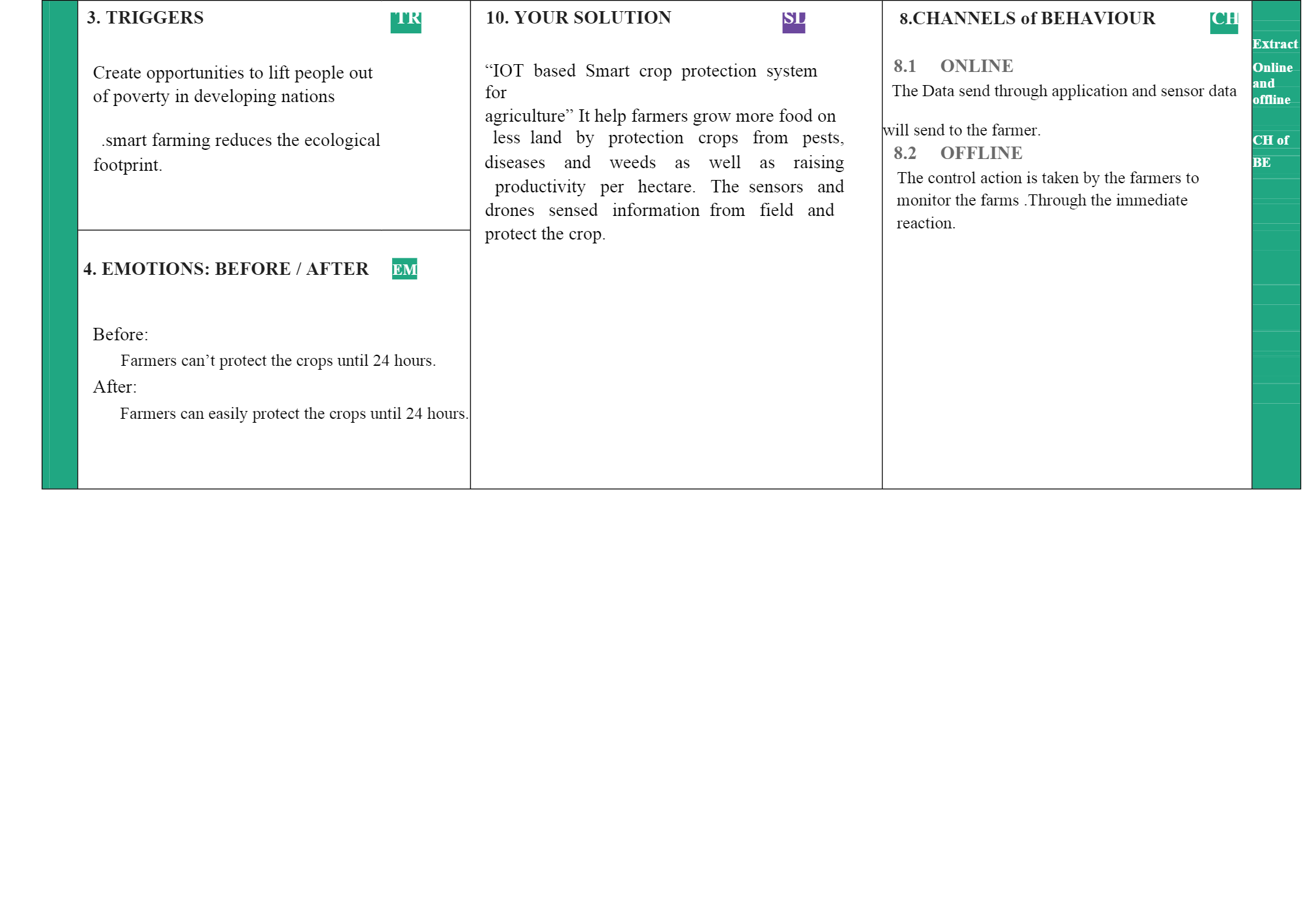
**IDEATION AND BRAINSTORMING:**

**PROPOSED SOLUTION:**



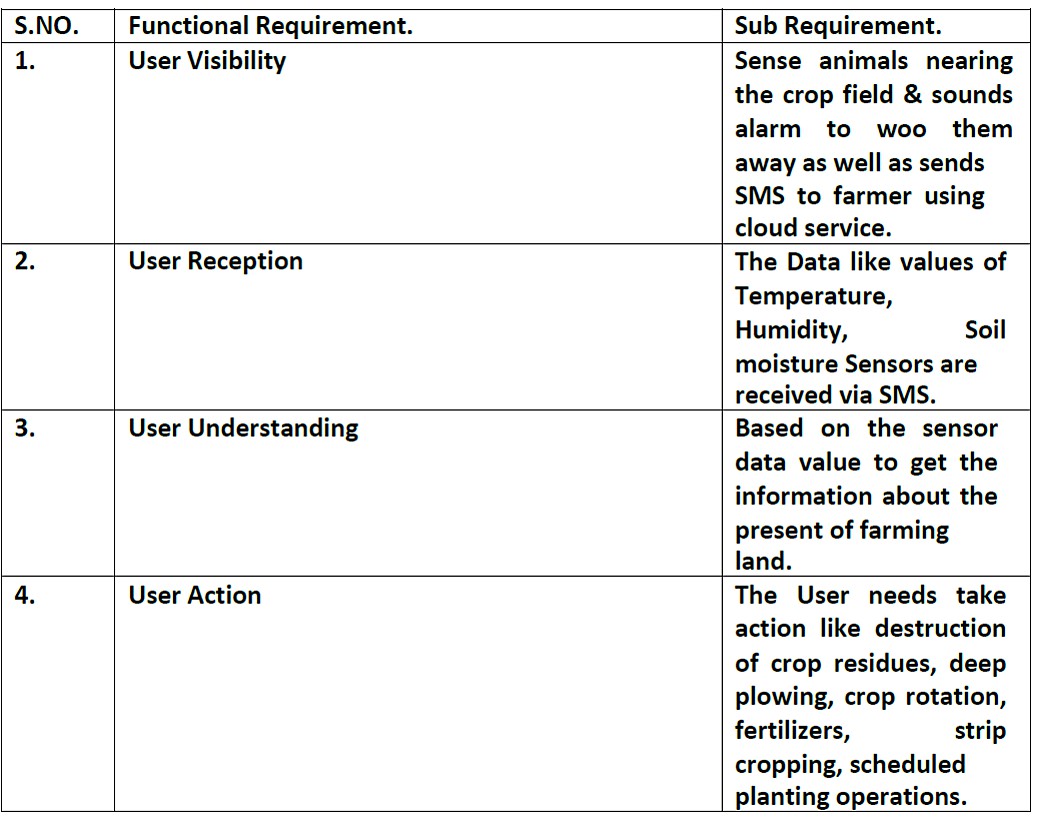
**PROBLEM SOLUTIONFIT:**



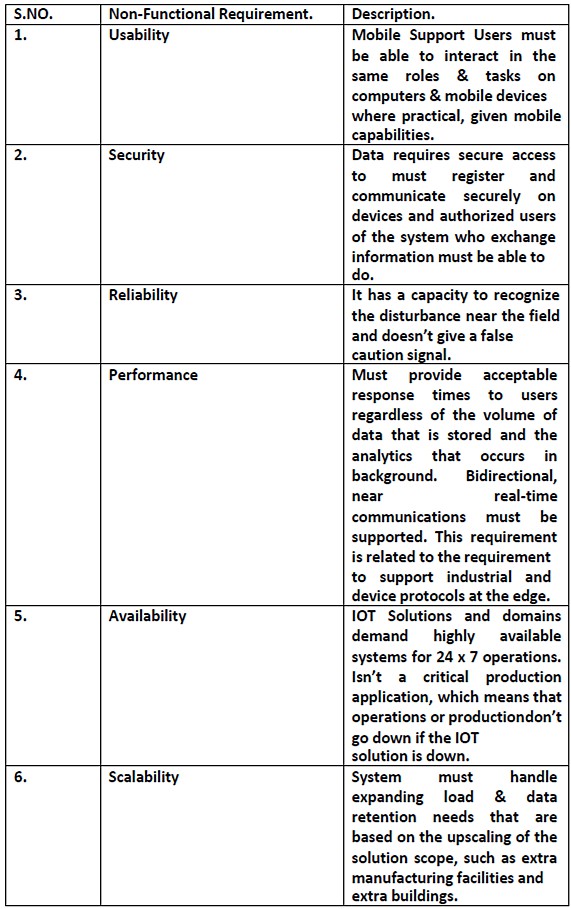


**REQUIREMENT ANALYSIS**

**FUNCTIONAL REQUIREMENT:**

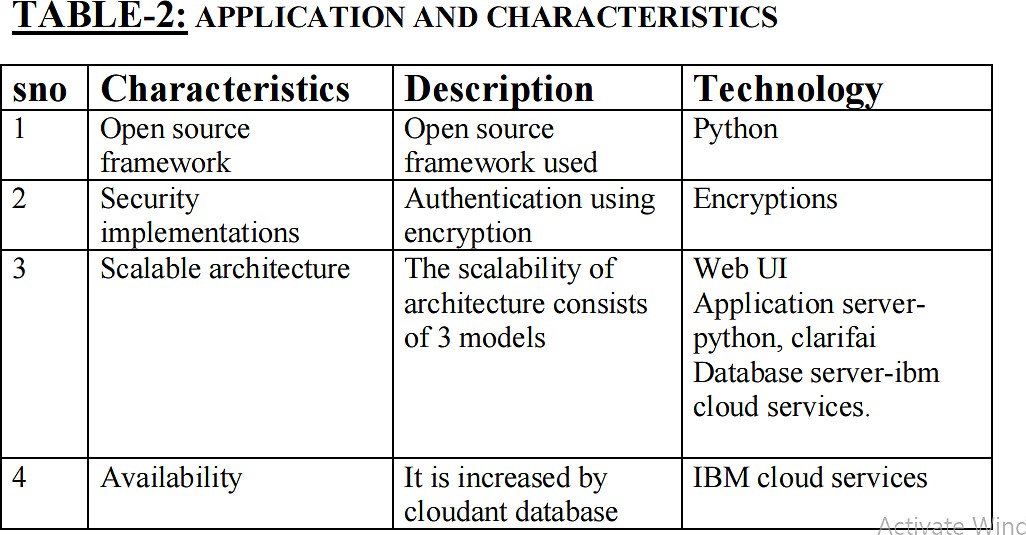
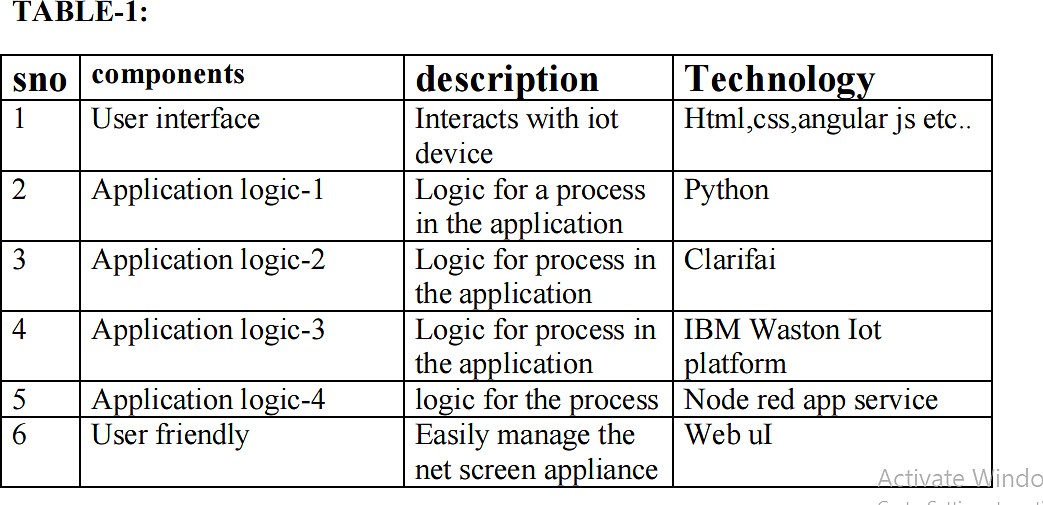
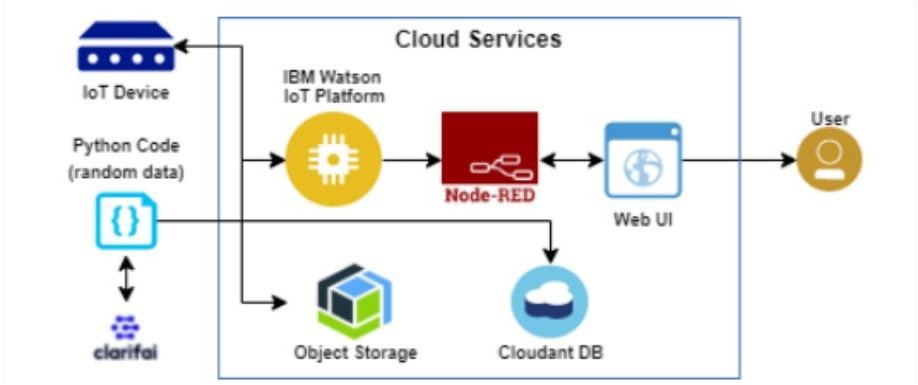
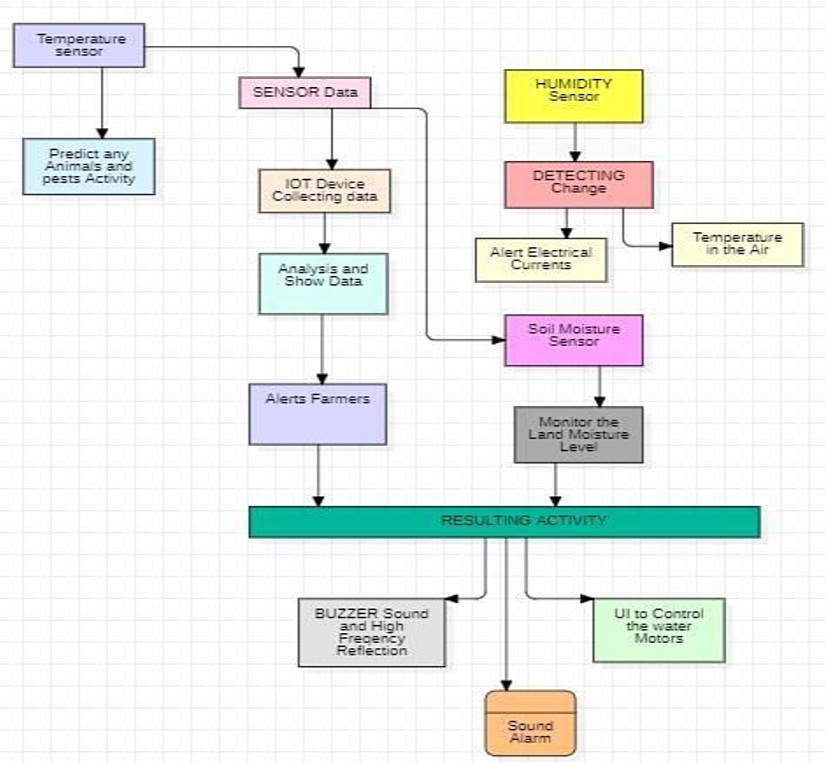


**NON FUNCTINAL REQUIREMENT:**



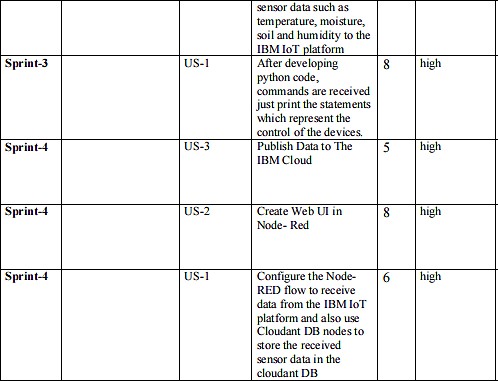
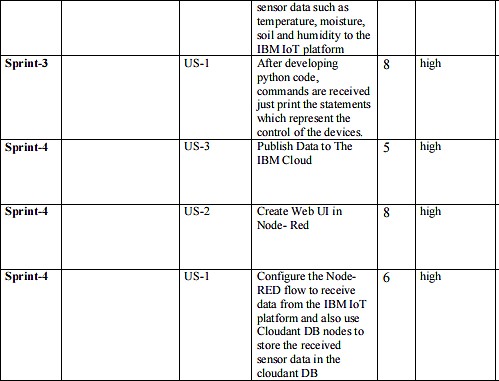
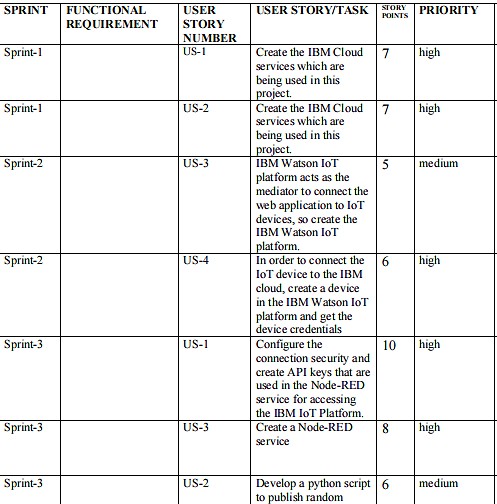
**PROJECT DESIGN**

**DATA FLOW DIAGRAM:**



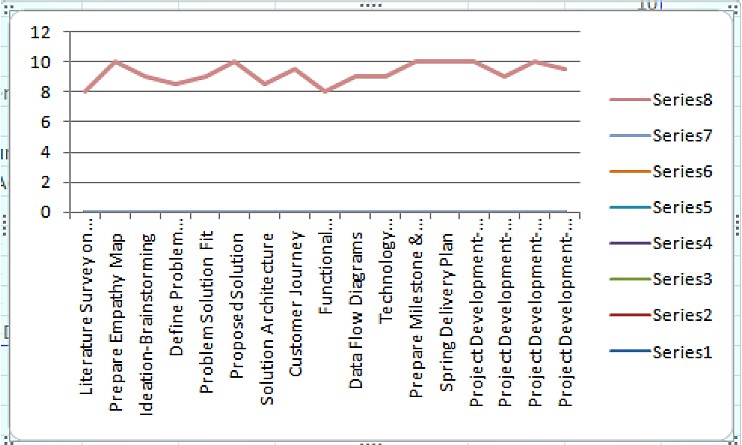
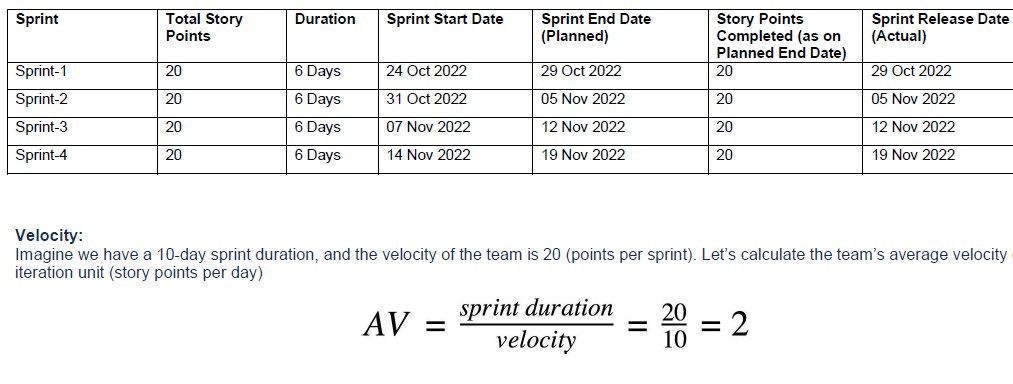
**SOLUTION AND TECHNICAL ARCHITECTURE:**

**USER STORIES:**



**PROJECT PLANNINGAND SCHEDULING**

**SPRINT PLANNINGAND ESTIMATION:**



**CODING AND SOLUTIONING**

**FEATURE-1**

import random

import ibmio .applica on import ibmio .device

from me import sleep import sys

#IBM Watson Device Creden als.

organiza on = "op701j" deviceType = "Lokesh" deviceId = "Lokesh89" authMethod = "token" authToken = "1223334444"

def myCommandCallback(cmd):

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

status=cmd.data['command']

if status=="sprinkler\_on":

print ("sprinkler is ON")

else :

print ("sprinkler is OFF")

#print(cmd)

try:

deviceOp ons = {"org": organiza on, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}

deviceCli = ibmio .device.Client(deviceOp ons)

except Excep on as e:

print("Caught excep on connec ng device: %s" % str(e))

sys.exit()

#Connec ng to IBM watson. deviceCli.connect()

while True:

#Ge ng values from sensors.

temp\_sensor = round( random.uniform(0,80),2) PH\_sensor = round(random.uniform(1,14),3)

camera = ["Detected","Not Detected","Not Detected","Not Detected","Not Detected","Not Detected",]

camera\_reading = random.choice(camera)

ﬂame = ["Detected","Not Detected","Not Detected","Not Detected","Not Detected","Not Detected",]

ﬂame\_reading = random.choice(ﬂame) moist\_level = round(random.uniform(0,100),2) water\_level = round(random.uniform(0,30),2)

#storing the sensor data to send in json format to cloud. temp\_data = { 'Temperature' : temp\_sensor }

PH\_data = { 'PH Level' : PH\_sensor }

camera\_data = { 'Animal a ack' : camera\_reading} ﬂame\_data = { 'Flame' : ﬂame\_reading } moist\_data = { 'Moisture Level' : moist\_level} water\_data = { 'Water Level' : water\_level}

# publishing Sensor data to IBM Watson for every 5-10 seconds.

success = deviceCli.publishEvent("Temperature sensor", "json", temp\_data, qos=0)

sleep(1)

if success:

print (" ............................publish ok............................. ")

print ("Published Temperature = %s C" % temp\_sensor, "to IBM Watson")

success = deviceCli.publishEvent("PH sensor", "json", PH\_data, qos=0)

sleep(1)

if success:

print ("Published PH Level = %s" % PH\_sensor, "to IBM Watson")

success = deviceCli.publishEvent("camera", "json", camera\_data, qos=0)

sleep(1)

if success:

print ("Published Animal a ack %s " % camera\_reading, "to IBM Watson") success = deviceCli.publishEvent("Flame sensor", "json", ﬂame\_data, qos=0) sleep(1)

if success:

print ("Published Flame %s " % ﬂame\_reading, "to IBM Watson")

success = deviceCli.publishEvent("Moisture sensor", "json", moist\_data, qos=0)

sleep(1)

if success:

print ("Published Moisture Level = %s " % moist\_level, "to IBM Watson")

success = deviceCli.publishEvent("Water sensor", "json", water\_data, qos=0)

sleep(1)

if success:

print ("Published Water Level = %s cm" % water\_level, "to IBM Watson")

print ("")

#Automa on to control sprinklers by present temperature an to send alert message to IBM Watson.

if (temp\_sensor > 35):

print("sprinkler-1 is ON")

success = deviceCli.publishEvent("Alert1", "json",{ 'alert1' : "Temperature(%s) is high, sprinkerlers are turned ON" %temp\_sensor }

, qos=0)

sleep(1)

if success:

print( 'Published alert1 : ', "Temperature(%s) is high, sprinkerlers are turned ON" %temp\_sensor,"to IBM Watson")

print("")

else:

print("sprinkler-1 is OFF")

print("")

#To send alert message if farmer uses the unsafe fer lizer to crops. if (PH\_sensor > 7.5 or PH\_sensor < 5.5):

success = deviceCli.publishEvent("Alert2", "json",{ 'alert2' : "Fer lizer PH level(%s) is not safe,use other fer lizer" %PH\_sensor } , qos=0)

sleep(1)

if success:

print('Published alert2 : ' , "Fer lizer PH level(%s) is not safe,use other fer lizer" %PH\_sensor,"to IBM Watson")

print("")

#To send alert message to farmer that animal a ack on crops. if (camera\_reading == "Detected"):

success = deviceCli.publishEvent("Alert3", "json", { 'alert3' : "Animal a ack on crops detected" }, qos=0)

sleep(1)

if success:

print('Published alert3 : ' , "Animal a ack on crops detected","to IBM Watson","to IBM Watson")

print("")

#To send alert message if ﬂame detected on crop land and turn ON the splinkers to take immediate ac on.

if (ﬂame\_reading == "Detected"):

print("sprinkler-2 is ON")

success = deviceCli.publishEvent("Alert4", "json", { 'alert4' : "Flame is detected crops are in danger,sprinklers turned ON" }, qos=0)

sleep(1)

if success:

print( 'Published alert4 : ' , "Flame is detected crops are in danger,sprinklers turned ON","to IBM Watson")

#To send alert message if Moisture level is LOW and to Turn ON Motor-1 for irriga on. if (moist\_level < 20):

print("Motor-1 is ON")

success = deviceCli.publishEvent("Alert5", "json", { 'alert5' : "Moisture level(%s) is low, Irriga on started" %moist\_level }, qos=0)

sleep(1)

if success:

print('Published alert5 : ' , "Moisture level(%s) is low, Irriga on started" %moist\_level,"to IBM Watson" )

print("")

#To send alert message if Water level is HIGH and to Turn ON Motor-2 to take water out.

if (water\_level > 20):

print("Motor-2 is ON")

success = deviceCli.publishEvent("Alert6", "json", { 'alert6' : "Water level(%s) is high, so motor is ON to take water out "

%water\_level }, qos=0)

sleep(1)

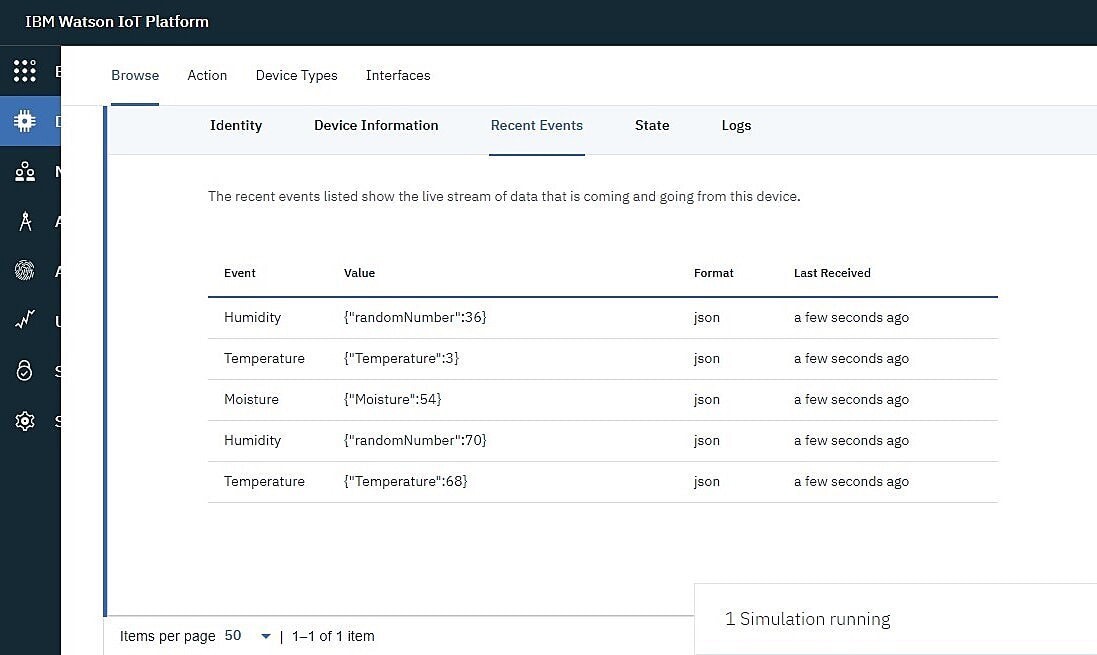
if success:

print('Published alert6 : ' , "water level(%s) is high, so motor is ON to take water out " %water\_level,"to IBM Watson" )

print("")

#command recived by farmer deviceCli.commandCallback = myCommandCallback

# Disconnect the device and applica on from the cloud deviceCli.disconnect()



**Features**

Output: Digital pulse high (3V) when triggered (mo on detected) digital low when idle (no mo on detected). Pulse lengths are determined by resistors and capacitors on the PCB and diﬀer from sensor to sensor. Power supply: 5V-12V input voltage for most modules (they have a

3.3V regulator),but 5V is ideal in case the regulator has diﬀerent specs.

**BUZZER**

Speciﬁca ons

● RatedVoltage : 6V DC

● Opera ng Voltage : 4 to 8V DC

● Rated Current\*: ≤30mA

● SoundOutput at 10cm\* : ≥85dB

● Resonant Frequency : 2300 ±300Hz

● Tone: Con nuous A buzzer is a loud noise maker.

Most modern ones are civil defense or air- raid sirens, tornado sirens, or the sirens on emergency service vehiclessuch as ambulances, police cars and ﬁre trucks. There are two general types, pneuma c and electronic.

**FEATURE-2:**

i. Goodsensi vity to Combus ble gas in wide range .

ii. Highsensi vity to LPG, Propane and Hydrogen .

iii. Longlife and low cost.

iv. Simpledrive circuit.

**TESTING**

**TEST CASES:**

sno parameter Values Screenshot

1 Model summary -

2 accuracy Training accuracy-

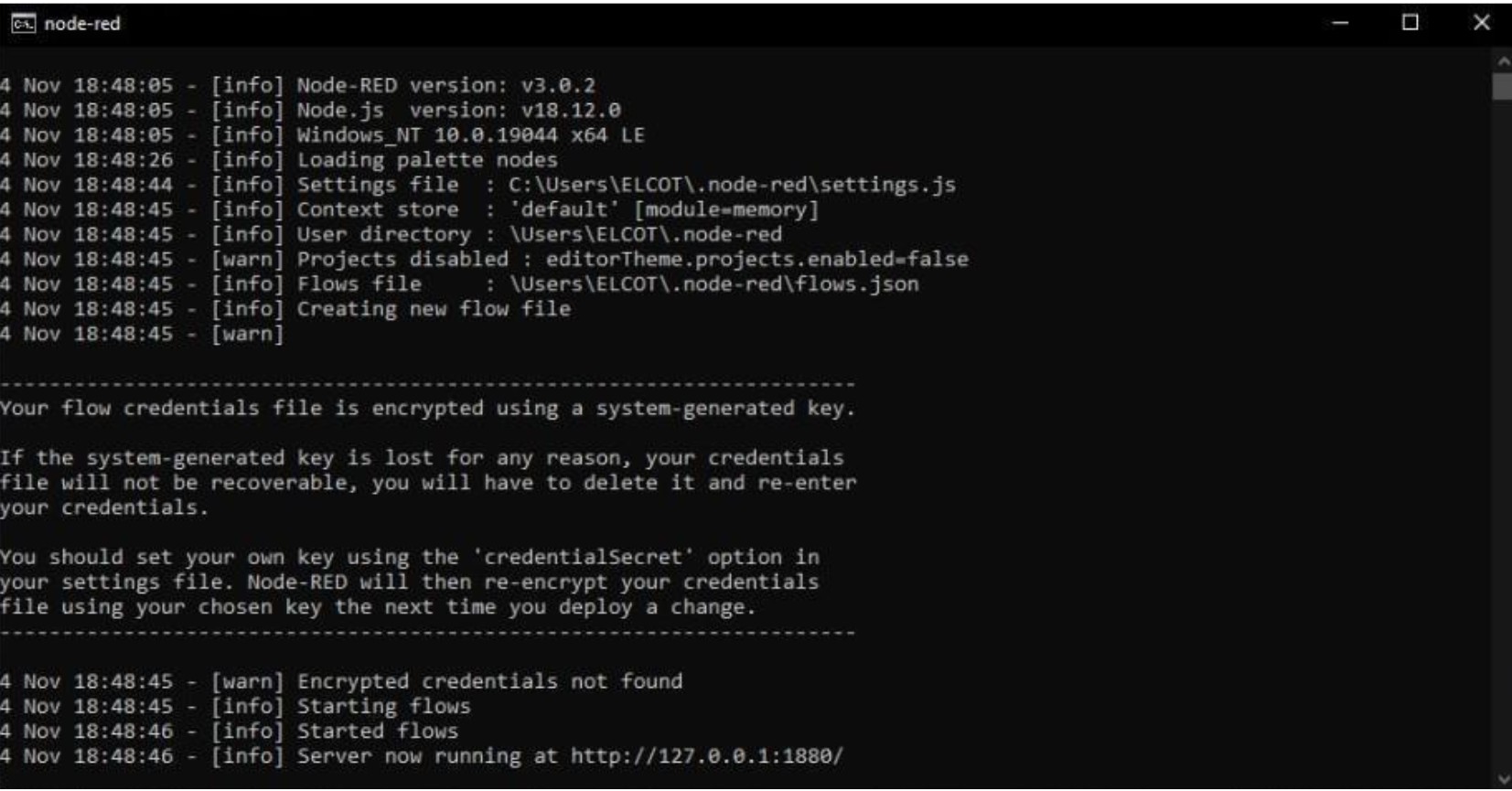
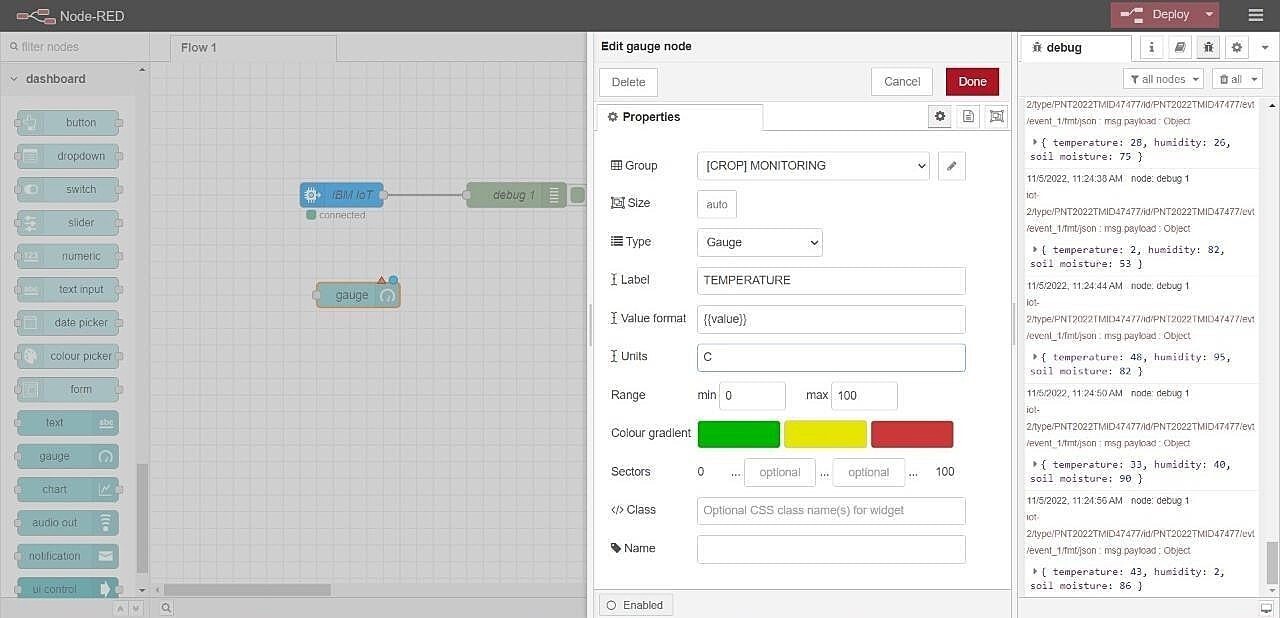
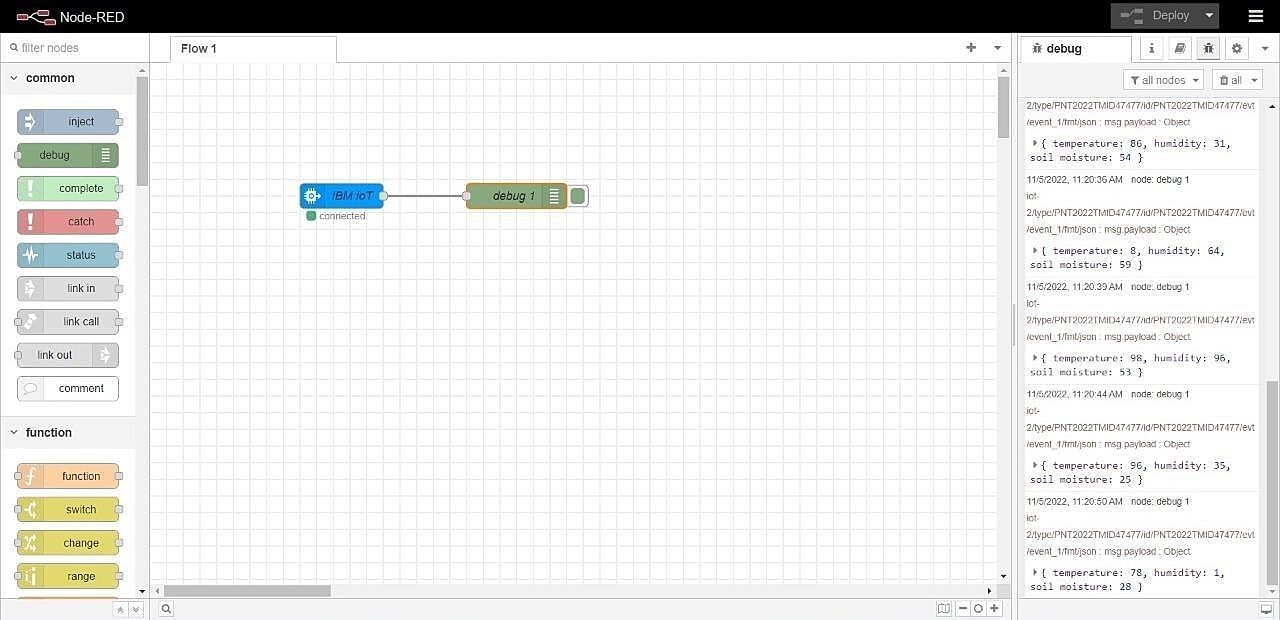
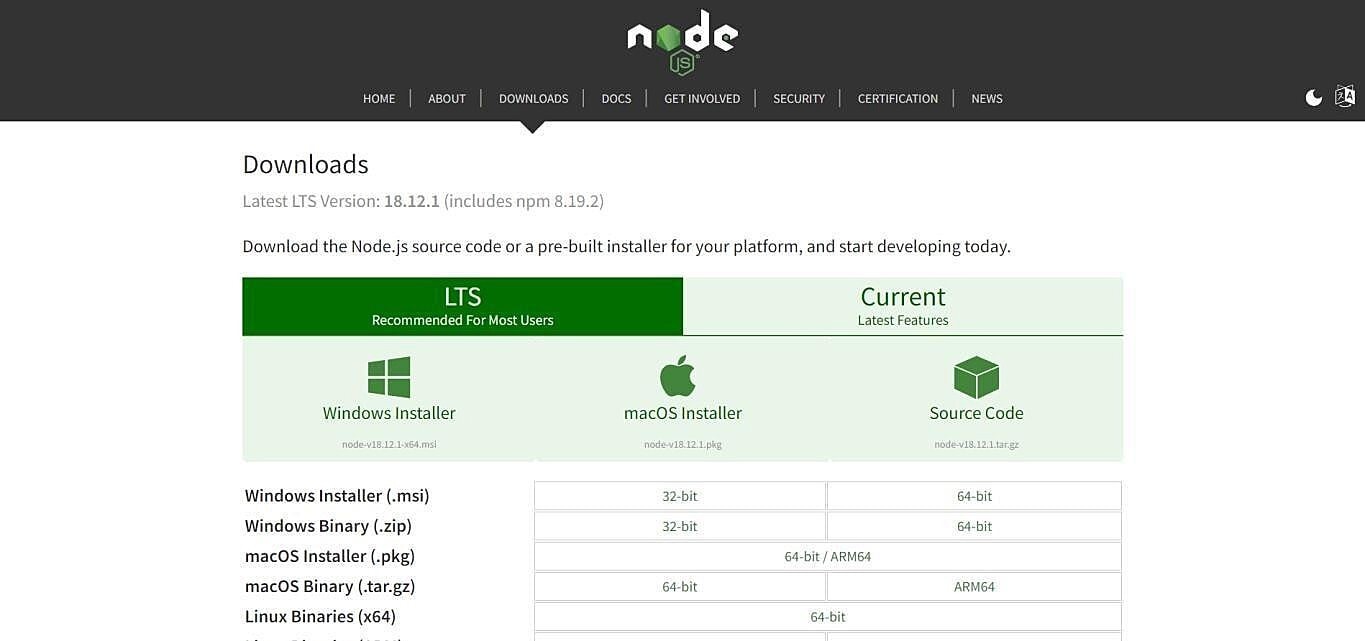
95% Validation accuracy-

72%

3 Confidence score Class detected-

80% Confidence score-80%

**User Acceptance Testing:**



**RESULTS**

The problem of crop vandalization by wild animals and fire has become a major social problem in current time.

It requires urgent attention as no effective solution exists till date for this problem. Thus this project carries a great social relevance as it aims to address this problem. This project willhelp farmers in protecting their orchards and fields and save them from significant financial losses and will save them from the unproductive efforts that they endure for the protection their fields. This will also help them in achievingbetter crop yields thus leading to theireconomic wellbeing.

**ADVANTAGES AND DISADVANTAGES**

**Advantage:**

Controllable food supply. you might have droughts or floods, but if you are growing the crops and breeding them to be hardier, you have a better chanceof not straving. It allows farmers to maximize yields using minimum resources such as water,fertilizers.

**Disadvantage:**

The main disadvantage is the time it can take to process the information.in order to keep feeding people as the population grows you have to radically change theenvironment of the planet

**CONCLUSION:**

A IoT Web Application is built for smart agricultural system using Watson IoT

platform, Watsonsimulator, IBM cloud and Node-RED

**FUTURE SCOPE**

In the future, there will be very large scope, this project can be made based on Image processing in which wild animaland fire can be detected by cameras and if it comes towards farmthen system will be directly activated through wireless networks. Wild animals can also be detected by using wireless networks such as laser wireless sensors and by sensing this laser or sensor’s security system will beactivated.

**APPENDIX**

**SOURCE CODE**

import me importsys import ibmio .applica on # toinstallpip install ibmio impor bmio .device

# Provide your IBM Watson Device Creden als organiza on = "8gyz7t" # replace the ORG ID deviceType = "weather\_monitor" #replace the Device type deviceId = "b827ebd607b5" # replace Device ID authMethod = "token" authToken = "LWVpQPaVQ166HWN48f" # Replace the authtoken

def myCommandCallback(cmd): # func on for Callbackif cm.data['command'] == 'motoron':

print("MOTOR ON IS RECEIVED")

elif cmd.data['command'] == 'motoroﬀ':print("MOTOR OFF IS RECEIVED")

if cmd.command == "setInterval":

else:

if 'interval' not in cmd.data:

print("Error - command is missing requiredinforma on: 'interval'")

interval = cmd.data['interval']

elif cmd.command == "print":

if 'message' not in cmd.data:

print("Error - commandis missing requiredinforma on: 'message'")

else:output = cmd.data['message']

print(output)

try:

deviceOp ons = {"org": organiza on, "type": deviceType, "id": deviceId,"authmethod":

authMethod,

"auth-token": authToken} deviceCli

= ibmio .device.Client(deviceOp ons)#

..............................................

exceptExcep on as e:

print("Caught excep on connec ng device: %s" % str(e))sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event oftype "gree ng"

10 mes deviceCli.connect()

while True:

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and applica on from the cloud deviceCli.disconnect()

**SENSOR.PY**

import me import sysimport

ibmio .applica on impor bmio .device

import random

# Provide your IBM Watson Device Creden als organiza on = "8gyz7t" # replace the ORG ID deviceType = "weather\_monitor" #replace the Device type deviceId = "b827ebd607b5" # replace Device ID authMethod = "token" authToken = "LWVpQPaVQ166HWN48f" # Replace the authtoken

def myCommandCallback(cmd):

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

print(cmd)

try:

deviceOp ons = {"org": organiza on, "type": deviceType, "id": deviceId,

"auth-method": authMethod, "auth-token": authToken}

deviceCli = ibmio .device.Client(deviceOp ons)

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

exceptExcep on as e:

print("Caught excep on connec ng device: %s" % str(e))sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event oftype "gree ng"

10 mes deviceCli.connect()

while True:

temp=random.randint(0,1

00)

pulse=random.randint(0,100)

soil=random.randint(0,100)

data = { 'temp' : temp, 'pulse': pulse ,'soil':soil}

#print data def myOnPublishCallback():

print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %pulse,"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") me.sleep(1)

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and applica on from the cloud deviceCli.disconnect()

**Node-RED FLOW :**

[

{ "id":"625574ead9839b34 ",

"type":"ibmiotout", "z":"630c8601c5ac3295",

"authen ca on":"apiKey", "apiKey":"ef745d48e395ccc0", "outputType":"cmd", "deviceId":"b827ebd607b5", "deviceType":"weather\_monitor", "eventCommandType":"data", "format":"json",

"data":"data",

"qos":0, "name":"IBM IoT", "service":"registere d","x":680,

"y":220,

"wires":[]

},

{

"id":"4cﬀ18c3274cccc4","type":"ui\_bu on", "z":"630c8601c5ac3295",

"name":"",

"group":"716e956.00eed6c", "order":2,

"width":"0",

"height":"0",

"passthru":false, "label":"MotorON", "tool p":"",

"color":"", "bgcolor":"", "className":"", "icon":"",

"payload":"{\"command\":\"motoron\"}",

"payloadType":"str", "topic":"motoron",

"topicType":"s tr","x":360,

"y":160, "wires":[["625574ead9839b34"]]},

{ "id":"659589baceb4e0b0",

"type":"ui\_bu on", "z":"630c8601c5ac3295",

"name":"", "group":"716e956.00eed6c", "order":3,

"width":"0",

"height":"0", "passthru":true, "label":"MotorOF F",

"tool p":"",

"color":"", "bgcolor":"", "className":"", "icon":"",

"payload":"{\"command\":\"motoroff\"}",

"payloadType":"str", "topic":"motoroﬀ",

"topicType":"s tr","x":350,

"y":220, "wires":[["625574ead9839b34"]]},

{"id":"ef745d48e395ccc0","type":"ibmiot", "name":"weather\_monitor","keepalive":"60", "serverName":"",

"cleansession":true, "appId":"", "shared":false},

{"id":"716e956.00eed6c", "type":"ui\_group", "name":"Form",

"tab":"7e62365e.b7e6b8

","order":1, "disp":true, "width":"6", "collapse":fal se},

{"id":"7e62365e.b7e6b8", "type":"ui\_tab",

"name":"contorl",

"icon":"dashboard ","order":1, "disabled":false, "hidden":false}

]

[

{

"id":"b42b5519fee73ee2", "type":"ibmio n", "z":"03acb6ae05a0c712",

"authen ca on":"apiKey", "apiKey":"ef745d48e395ccc0",

"inputType":"evt", "logicalInterface":"", "ruleId":"", "deviceId":"b827ebd607b5", "applica onId":"",

"deviceType":"weather\_monitor",

"eventType":"+", "commandType":"",

"format":"json", "name":"IBMIoT",

"service":"registered",

"allDevices":"", "allApplica ons":"", "allDeviceTypes":"", "allLogicalInterfaces":"", "allEvents":true, "allCommands":"", "allFormats

":"",

"qos":0, "x":270,

"y":180,

"wires":[["50b13e02170d73fc","d7da6c2f5302ffaf","a949797028158f3f","a71f164bc3 78bcf1"]]

},

{ "id":"50b13e02170d73fc ",

"type":"func on",

"z":"03acb6ae05a0c712 ","name":"Soil Moisture",

"func":"msg.payload = msg.payload.soil;\nglobal.set('s',msg.payload);\nreturn msg;",

"outputs":1, "noerr":

0,

"ini alize ":"", "ﬁnalize":"",

"libs":[],

"x":490, "y":120, "wires":[["a949797028158f3f","ba98e701f55f04fe"]]

},

{

"id":"d7da6c2f5302ﬀaf","type":"func on", "z":"03acb6ae05a0c712", "name":"Humidity",

"func":"msg.payload = msg.payload.pulse;\nglobal.set('p',msg.payload)\nreturn msg;",

"outputs":1, "noerr":

0,

"ini alize ":"", "ﬁnalize":"",

"li bs ":[

], "x ":

48

0,

"y":260, "wires":[["a949797028158f3f","70a5b076eeb80b70"]]

},

{ "id":"a949797028158f3f ",

"type":"debug",

"z":"03acb6ae05a0c712 ","name":"IBMo/p",

"ac ve":true, "tosidebar":true, "console":false, "tostatus":false, "complete":"payload", "targetType":"msg", "statusVal":"", "statusType":"auto", "x":780,

"y":180,

"wires":[]

},

{ "id":"70a5b076eeb80b70", "type":"ui\_gauge",

"z":"03acb6ae05a0c712",

"name":"", "group":"f4cb8513b95c98a4", "order":6,

"width":"0", "height":"0", "gtype":"gage",

" tle":"Humidity", "label":"Percentage(%)", "format":"{{value}}

","min":0,

"max":"100", "colors":["#00b500","#e6e600","#ca3838"], "seg1":"", "seg2":"",

"className

":"","x":86

0, "y":260,

"wires":[]

},

{

"id":"a71f164bc378bcf1","type":"func on", "z":"03acb6ae05a0c712", "name":"Temperature",

"func":"msg.payload=msg.payload.temp;\nglobal.set('t',msg.payload);\nreturn msg;","outputs":1,

"noerr":

0,

"ini alize ":"", "ﬁnalize":"", "li

bs

":[

],

"x

":

49

0, "y":360,

"wires":[["8e8b63b110c5ec2d","a949797028158f3f"]]

},

{ "id":"8e8b63b110c5ec2d", "type":"ui\_gauge",

"z":"03acb6ae05a0c712",

"name":"", "group":"f4cb8513b95c98a4", "order":11,

"width":"0",

"height":"0", "gtype":"gage",

" tle":"Temperature",

"label":"DegreeCelcius", "format":"{{value}}", "min":0,

"max":"100",

"colors":["#00b500","#e6e600","#ca3838"],"seg1":"", "seg2":"",

"className ":"", "x":790, "y":360,

"wires":[]

},

{ "id":"ba98e701f55f04fe",

"type":"ui\_gauge",

"z":"03acb6ae05a0c712", "name":"", "group":"f4cb8513b95c98a4", "order":1,

"width":"0", "height":"0",

"gtype":"gage",

" tle":"Soil Moisture", "label":"Percentage(%)", "format":"{{value}} ","min":0,

"max":"100", "colors":["#00b500","#e6e600","#ca3838"],"seg1":"", "seg2":"",

"className ":"", "x":790, "y":120,

"wires":[]

},

{ "id":"a259673baf5f0f98 ","type":"h pin",

"z":"03acb6ae05a0c712

","name":"", "url":"/sensor",

"method":"ge t", "upload":fals e, "swaggerDoc"

:"","x":370, "y":500,

"wires":[["18a8cdbf7943d27a"]]

},

{

"id":"18a8cdbf7943d27a","type":"func on", "z":"03acb6ae05a0c712",

"name":"h pfunc on", "func":"msg.payload{\"pulse\":global.get('p'),\"temp\":global.get('t'),\"soil\":global.get( 's')};\nreturn msg;",

"outputs":1, "noerr":0,

"ini alize":"", "ﬁnalize":"", "li

bs

":[

], "x ":

63

0,

"y":500, "wires":[["5c7996d53a445412"]]

},

{ "id":"5c7996d53a445412 ",

"type":"h presponse",

"z":"03acb6ae05a0c712 ","name":"", "statusCode":"",

"header s":{}, "x":870, "y":500,

"wires":[]

},

{ "id":"ef745d48e395ccc0", "type":"ibmiot",

"name":"weather\_monitor",

"keepalive":"60", "serverName":"", "cleansession":true, "appId":"", "shared":false},

{

"id":"f4cb8513b95c98a4","type":"ui\_group", "name":"monitor",

"tab":"1f4cb829.2fdee8 ","order":2,

"disp":

true, "width ":"6",

"collapse":f alse, "className ":""

},

{ "id":"1f4cb829.2fdee8", "type":"ui\_tab", "name":"Home", "icon":"dashboard

","order":3,

"disabled":false, "hidden":false }

**GitHub & Project Demo Link**

https://github.com/IBM-EPBL/IBM-Project-9126-1658981653