IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

DOMAIN: IOT

TEAM ID: PNT2022TMID29654

TEAM MEMBERS:

- DHOSHINIE S -513119106016
- JAPAKUMAR M - 513119106031
- JAYAKUMAR B – 5131191106032
- GAUTAM VINAY S – 513119106302

INTRODUCTION

PROJECT OVREVIEW:

Crops in farms are many times ravaged by local animals like buffaloes, cows, goats, birds etc. this leads to huge losses for the farmers. It is not possible for farmers to barricade entire fields or stay on field 24 hours and guard it.so here we propose automatic 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 safety of 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

IDEATION PHASE

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:

- **Paper1: IOT Based Crop Protection System against Birds and Wild Animals Attacks**

Publication year: April 2020

Authors: P. Navaneetha , R. Ramiya Devi , S. Vennila , P.Manikandan,

Dr. S. Saravanan

Journal name: INTERNATIONAL OF INNOVATIVE RESEARCH IN

TECHNOLOGY (IJIRT 149166)

- **Paper2: Smart Irrigation and Crop Protection from Wild Animals**

Publication year: April 2020

Authors: N. Penchalaiah, D. Pavithra, P. Bhargavi, D.P. Madhuri, K. Eliyas Shaik, S.Md.Sohaib

Journal name: JOURNAL OF ENGINEERING SCIENCES (JES)

- **Paper3:Smart Crop Protection System for Wild Animals Using IOT**

Publication year: 2021 ICCICA

Author: IEEE

Journal name: IEEE

- **Paper4: Smart Crop Protection System Using IOT**

Publication year: April 2021

Author: Krunal Mahajan, Riya Parate, Ekta Zade, Shubham Khante, Shishir Bagal

Journal name: INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN TECHNOLOGY (IJIRT 151020)

- **Paper5: Smart Crop Protection System From Living Objects And Fire Using Arduino**

Publication year: September 2020

Authors: Dr. M. Chandra Mohan Reddy, Keerthi Raju, Kamakshi Kodi, Babitha Anapalli , Mounika Pulla

Journal name: Science, Technology and Development

- **Paper6: Smart Crop Protection System**

Publication year: July to August 2021

Authors: Mohit Korche, Sarthak Tokse, Shubham Shirbhate , Vaibhav Thakre, S. P. Jolhe

Journal name: International Journal of Latest Engineering Sciences (IJLES)

- **Paper7: Smart Crop Protection System from Wild Animals and Birds Using IOT**

Publication year: 2021

Authors: Sumana P.B, Sanjana. R, Sharanya.M, Harish N.J

Journal name: International Journal of Advance Research, Ideas And Innovations In Technology(IJARIIT)

- **Paper8: Smart Crop Protection Using Arduino**

Publication year: July 2021

Authors: Varshini B.M, Sushma A.V

Journal name: International Advanced Research Journal in Science, Engineering and Technology (IARJSET)

- **Paper9: Review paper on Smart Crop Protection System**

Publication year: Feb 2021

Authors: Krunal Mahajan, Riya Parate, Ekta Zade, Shubham Khante, Shishir Bagal

Journal name: International Research Journal of Engineering and Technology (IRJET)

- **Paper10: Implementation of Crop Protection System Against WildAnimals Attack**

Publication year: February 2019

Authors: Atchaya V, Kowsalya V, Dhivya Bharathi K.P, Arunkumar M

Journal name: International Journal of Advanced Technology in Engineering and Science (IJATES)

- **Paper11: IOT Based Crop Monitoring from Animals**

Publication year: March 2019

Authors: K.B Pavan Kumar, T. Bhavithra, S. Karishma, M. Pavithra, M. Prashanth Kumar

Journal name: Journal of Emerging Technologies and Innovative Research (JETIR)

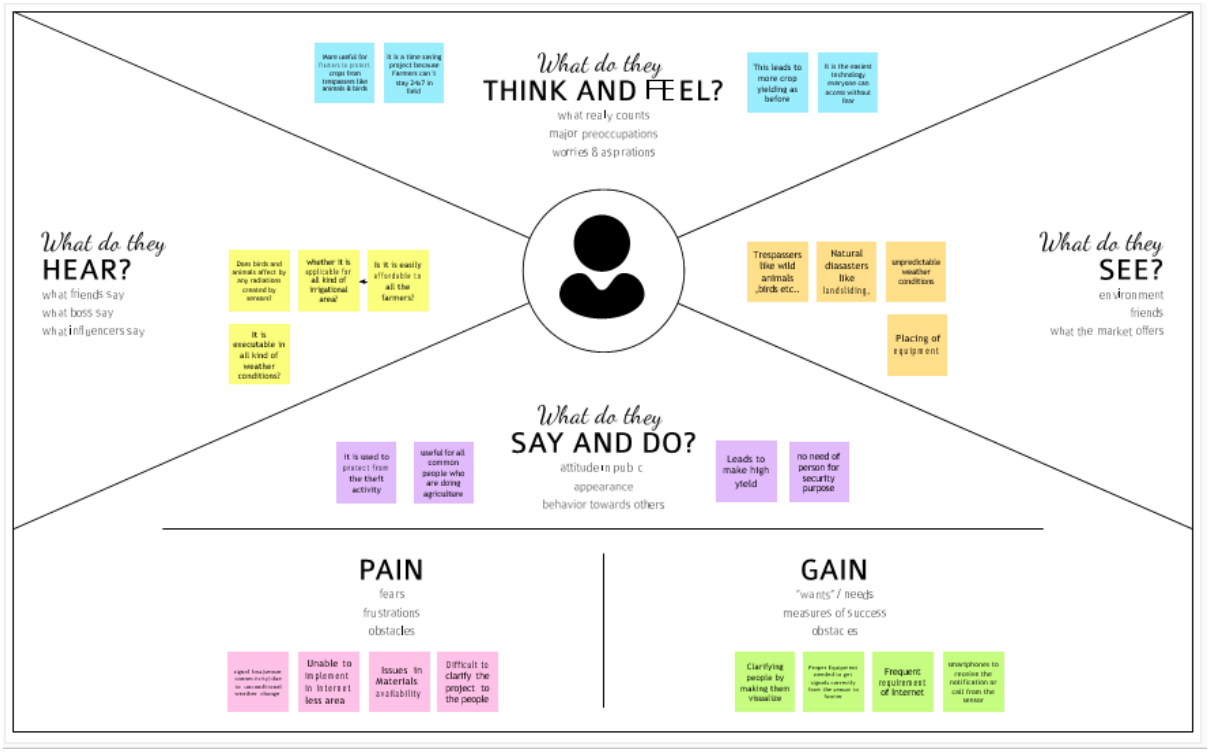
- **Paper12: Microcontroller Based Smart Crop Protection System To Detect Fire and Animals**

Publication year: April 2020

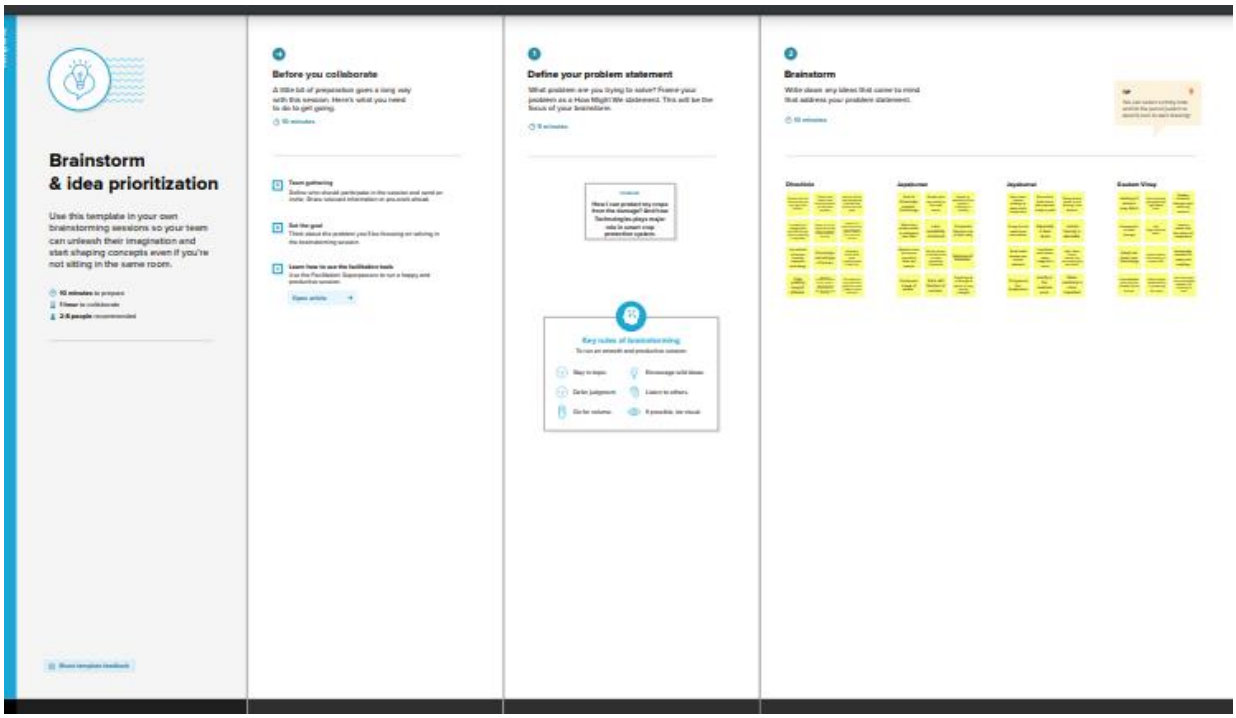
Authors: Premjyoti. G Patil, B.Pavan ,B.Siva Sai Reddy, B. Praveen kumar

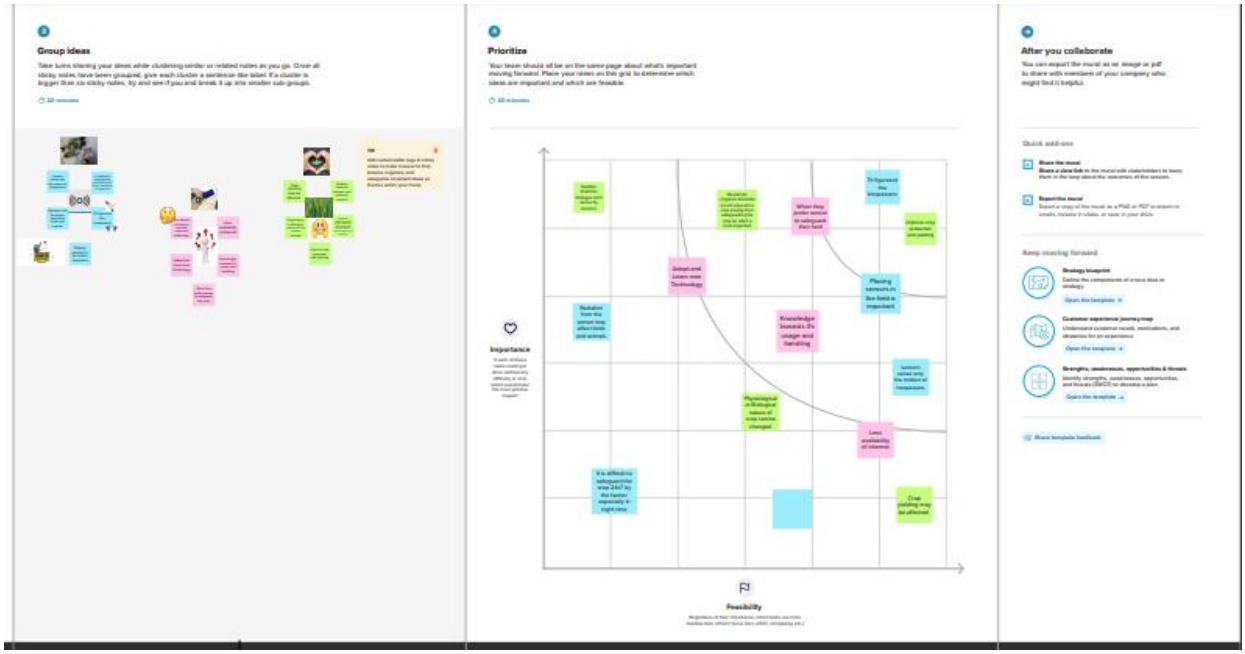
Journal name: International Journal of Innovative Science and Research Technology

EMPATHY MAP CANVAS

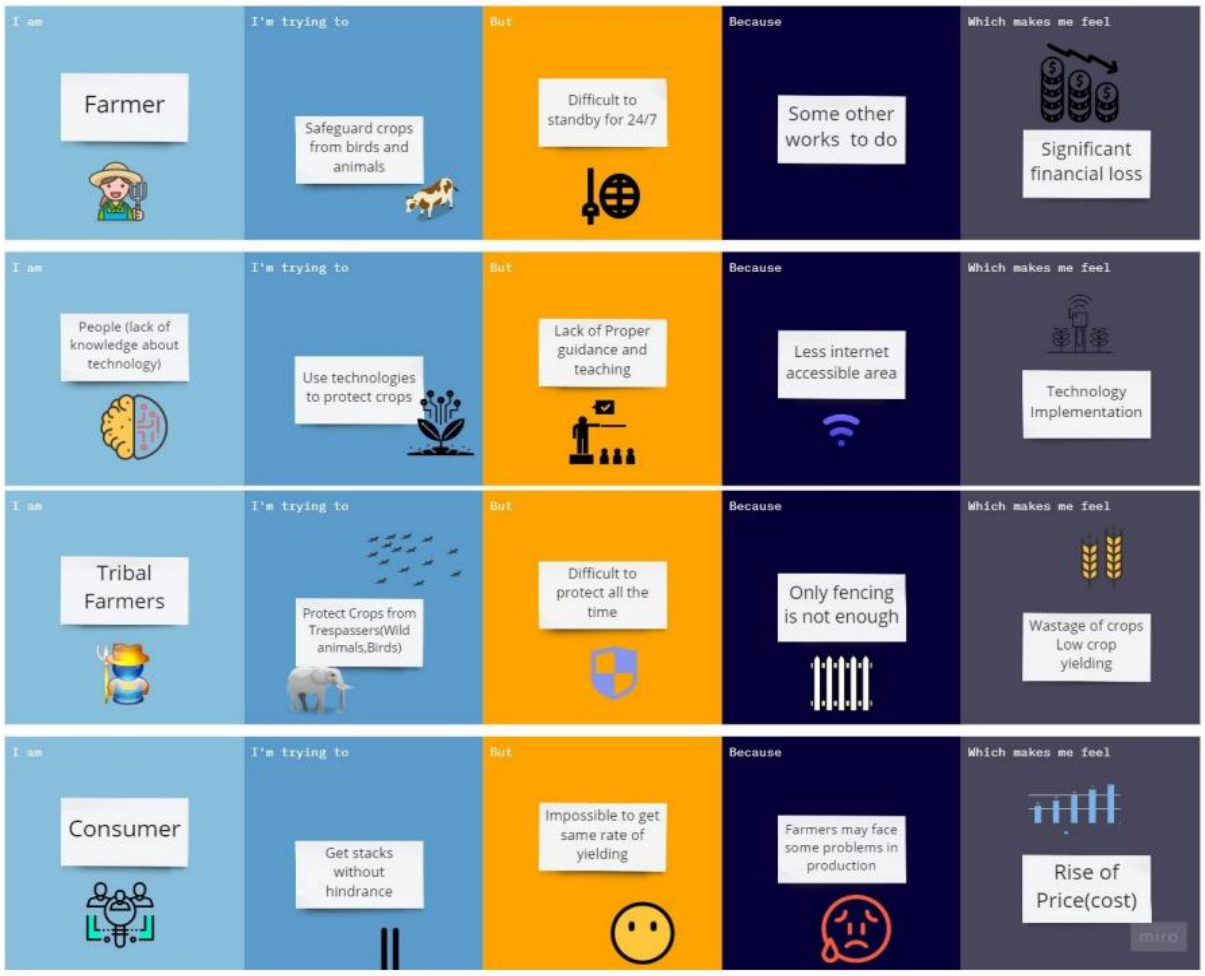


BRAINSTORMING





PROBLEM STATEMENT

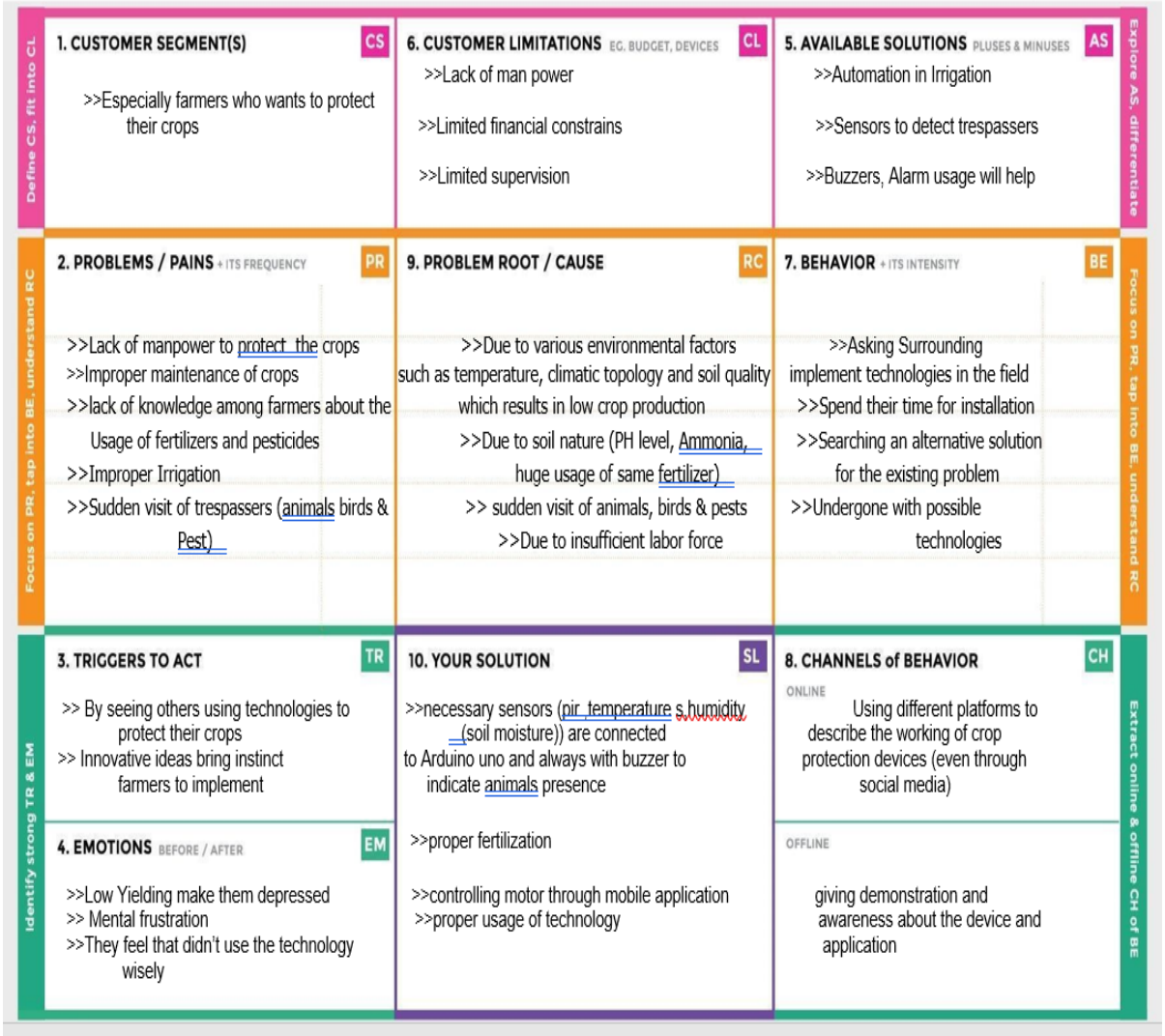


PROJECT DESIGN PHASE 1

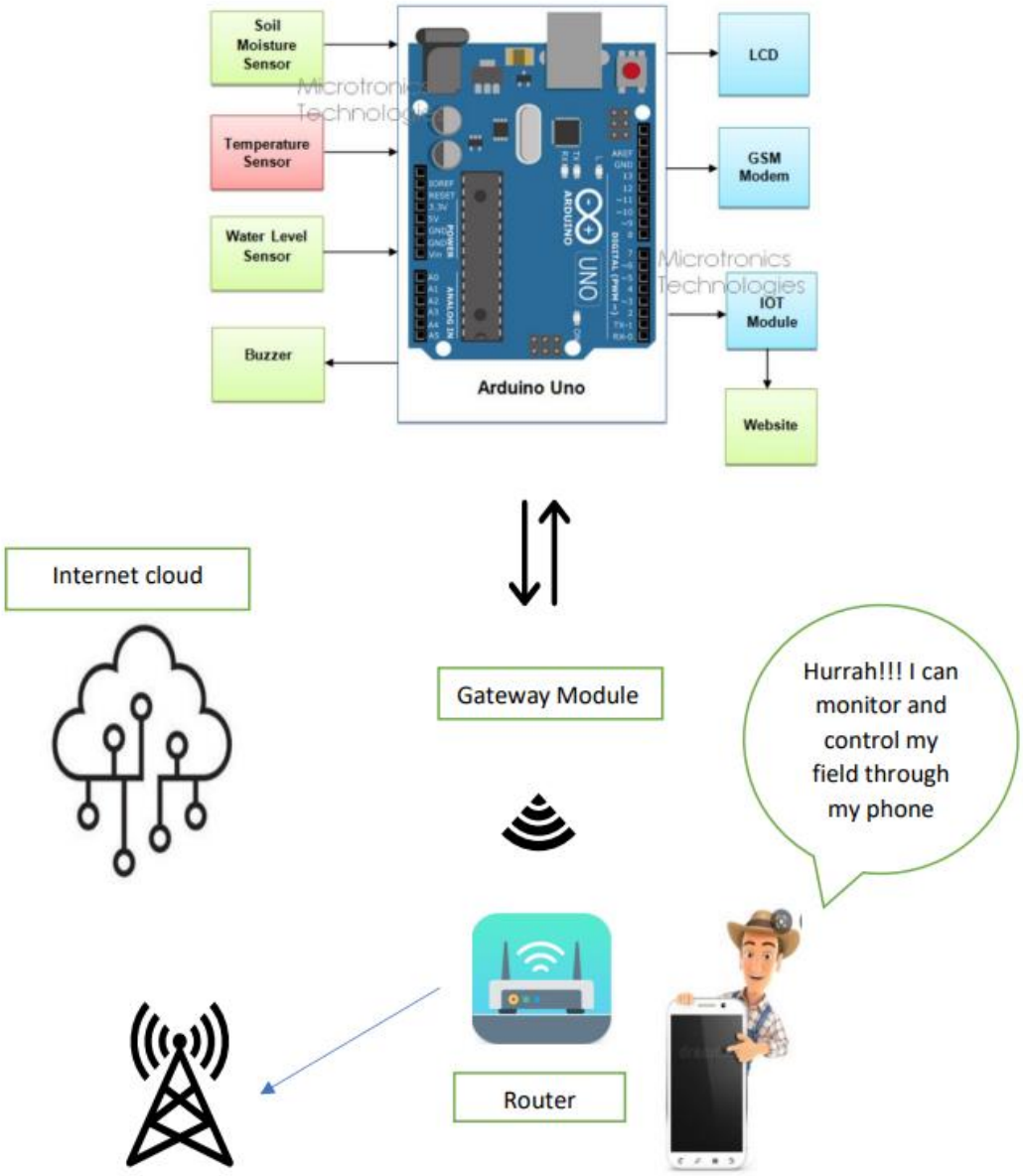
PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• Farmers can't stay in the field 24x7 in order to safeguard crops form animals (wild animals) and birds. • They don't know when the animals will attack the farm i.e., cause damage to the crop
2.	Idea / Solution description	• We are decided to protect crops from animals and birds by using SENSOR (PIR SENSOR), ARDUINO UNO, BUZZERS or ALARM...in the field. • This may help farmers to feel that their crops were safe and protected. This makes them to feel free.
3.	Novelty / Uniqueness	• In this project, apart from this sensors and Arduino. We decide to implement fencing with automatic door (opening & closing) by using ultrasonic sensor and servomotor.
4.	Social Impact / Customer Satisfaction	• Protecting crops from animals and birds especially in nights may become easier • They can do their other works without any fear about the crops.
5.	Business Model (Revenue Model)	• This will be one of the reasons for more crop yielding (i.e., Animals and birds presents may sense through sensors, therefore crops will be protected from damage). • Only Installation process is costlier, apart from that this will help farmers in great way.
6.	Scalability of the Solution	• To protect crops from animals and birds especially in night time is difficult. To avoid this discomfort, we will use this technology in our field to protect crops. • This will be FARMER'S FRIENDLY.

PROBLEM SOLUTIONFIT

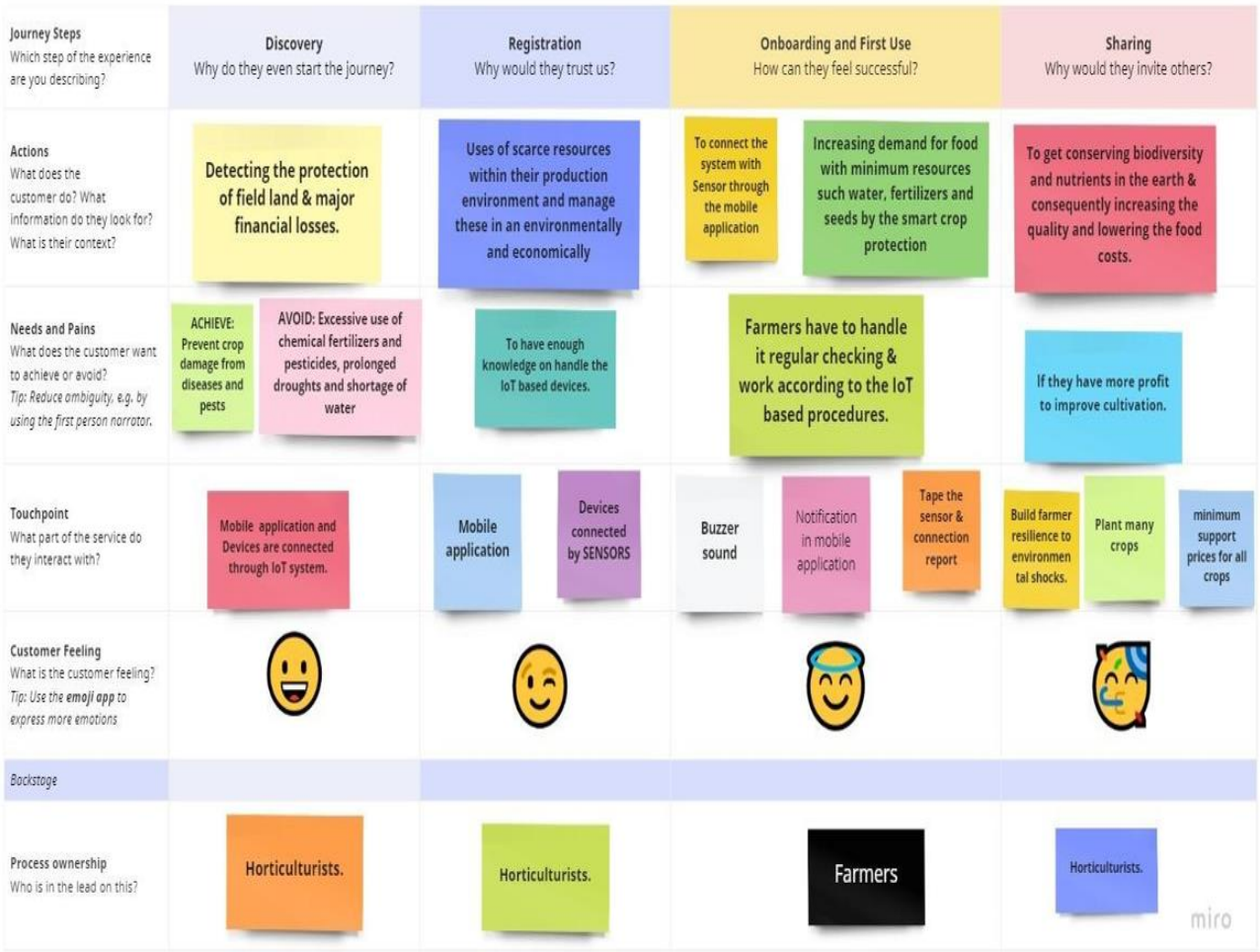


SOLUTION ARCHITECTURE



PROJECT DESIGN PHASE2

CUSTOMER JOURNEY



REQUIREMENT ANALYSIS

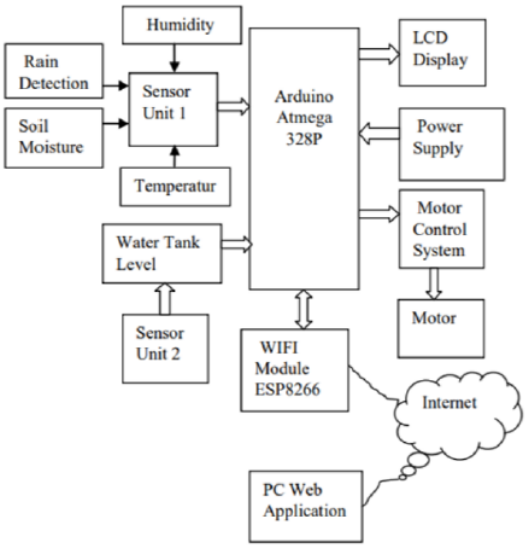
FUNCTIONAL REQUIREMENT:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	➢ Registration through Gmail
FR-2	User Confirmation	➢ Confirmation via Email ➢ Confirmation via OTP
FR-3	Log in	➢ Checking necessary Credentials
FR-4	Checking Weather Details	➢ Temperature Details ➢ Humidity details, Soil Moisture
FR-5	Management of motors and Sprinklers	➢ Farmers can operate motors and sprinklers through mobile application
FR-6	Logout	➢ Exit

NON FUNCTINAL REQUIREMENT:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	➢ Allows farmers to complete their day-to-day challenges
NFR-2	**Security**	➢ Is used to protect the farm from animals as well as unknown person
NFR-3	**Reliability**	➢ The use of smart IOT sensors can maintain these processes, increasing crop production
NFR-4	**Performance**	➢ Sensors helps to get instant warnings of soil salinity and moisture. Air and soil temperature system that allows farmers to schedule watering times and predict the chances of pests and also detect the motion of animals and birds.
NFR-5	**Availability**	➢ Equipment to auto adjust temperature, humidity etc and also to detect animals' and birds' motion
NFR-6	**Scalability**	➢ The biggest challenges faced by IOT in the agricultural sector are lack of information, high adoption costs and security concerns.

DATA FLOW DIAGRAM



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Farmer (In field)	Implementation	USN-1	Using Arduino (Microcontroller) to get input signals from sensors (motion detection sensor, temperature sensor, soil moisture sensor)	It can be accessed by them from their home and also by giving power supply to the components.	High	Sprint
		USN-2	Motion Detection sensor are used to detect the motion of the animals and birds as well as unknown persons in the field, if it detects it makes an Alarm.	Controlled by Arduino	High	Sprint
		USN-3	Soil Monitoring sensor is used to measure the humidity of the soil in the field and sprinkle water according to the soil moisture.	Controlled by Arduino	High	Sprint
		USN-4	Temperature Sensor is used to monitor the weather condition	Controlled by Arduino	High	Sprint
		USN-5	According to the soil moisture level motor gets on and off and also sprinklers will spray the water according to it. And it also controlled by mobile app.	Power supply is importance .it can be given by them from their home	High	Sprint
Farmer (mobile user)	Registration	USN-6	As a farmer he can register for the application by entering my email, password and confirming my password	It can be accessed from account/dashboard	High	Sprint
	Login	USN-7	As a user, he will receive confirmation email once he has registered for the application	They can receive confirmation email & click confirmation	High	Sprint
	Login Credential	USN-8	As a user, he can log into the application by entering email & password	It is ready to use the application	High	Sprint
	Using Application	USN-9	As a user, now he is ready to use the application	They can now operate the motor by ON/OFF the motor. They can also know the temperature and humidity level of the crops	High	Sprint

TECHNICAL STACK

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	Arduino	Arduino boards are able to read inputs-sensors	C, JavaScript
2.	Sensors	A device that detects and responds to some type of input from the physical environment	C, JavaScript
3.	Motion detection sensor	Detects motion of the animals and birds as well as unknown persons	C, JavaScript
4.	Soil Moisture sensor	Monitors the humidity and nature of the soil	C, JavaScript
5.	Temperature sensor	Detect the temperature of the soil as well as weather condition	C, JavaScript
6.	Database, Cloud Database	Database Service on Cloud	IBM Cloud, IBM Watson, NODE RED
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service
8.	Motors and sprinklers	Sprinkles the water when the motor is on	—
9.	LCD Display	Display when the motor gets on and off	—
10.	Mobile App	Used to on and off the motor through the mobile application	MIT Inventer
11.	Infrastructure (Server / Cloud)	Application Deployment on Cloud Cloud Server Configuration	Cloud Foundry

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Tinker cad, MIT Inverter
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Encryptionsss
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Technology used
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

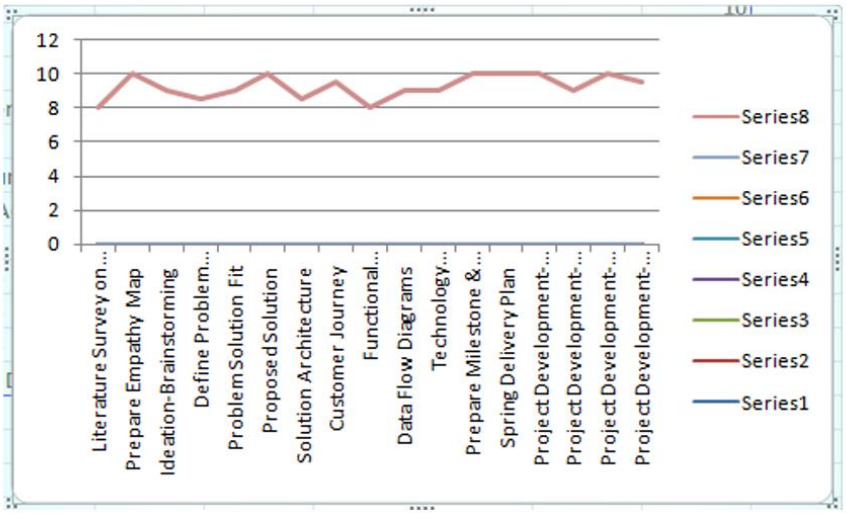
SPRINT PLAN DELIVERY

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$



CODING AND SOLUTIONING

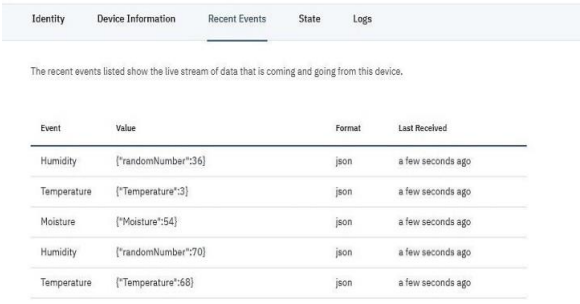
```
import random
import ibmiotf.application
import ibmiotf.device
from time import sleep
import sys
#IBM Watson Device Credentials.
organization = "ncgqpp"
deviceType = "raspberrypi"
deviceId = "123"
authMethod = "token"
authToken = "123456789"
def myCommandCallback(cmd):
 print("Command received: %s" % cmd.data['command'])
 status=cmd.data['command']
 if status=="sprinkler_on":
  print ("sprinkler is ON")
 else :
  print ("sprinkler is OFF")
#print(cmd)
try:
 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()
#Connecting to IBM watson.
deviceCli.connect()
while True:
#Getting 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 attack' : 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 attack %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 ("")

#Automation 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 fertilizer to crops.

if (PH_sensor > 7.5 or PH_sensor < 5.5):

 success = deviceCli.publishEvent("Alert2", "json",{ 'alert2': "Fertilizer PH level(%s) is not
safe,use other fertilizer"%PH_sensor } ,qos=0)
```

```
sleep(1)

if success:

 print('Published alert2 : ' , "Fertilizer PH level(%s) is not safe,use other fertilizer"
%PH_sensor,"to IBM Watson")

 print ("")

# To send alert message to farmer that animal attack on crops.

if (camera_reading == "Detected"):

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

 sleep(1)

if success:

 print('Published alert3 : ' , "Animal attack 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 action.

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")

 print("")

else:

 print("sprinkler-2 is OFF")

 print("")

#To send alert message if Moisture level is LOW and to Turn ON Motor-1 for irrigation.

if (moist_level < 20):

 print("Motor-1 is ON")

 success = deviceCli.publishEvent("Alert5", "json", { 'alert5': "Moisture level(%s) is low,
Irrigation started" %moist_level}, qos=0)
```

```
 sleep(1)
if success:
 print('Published alert5 : ' , "Moisture level(%s) is low, Irrigation started" %moist_level,"to IBM Watson" )
 print("")
else:
 print("Motor-1 is OFF")
 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("")
 else:
  print("Motor-2 of OFF")
  print("")
#command recived by farmer
deviceCli.commandCallback = myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```



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.

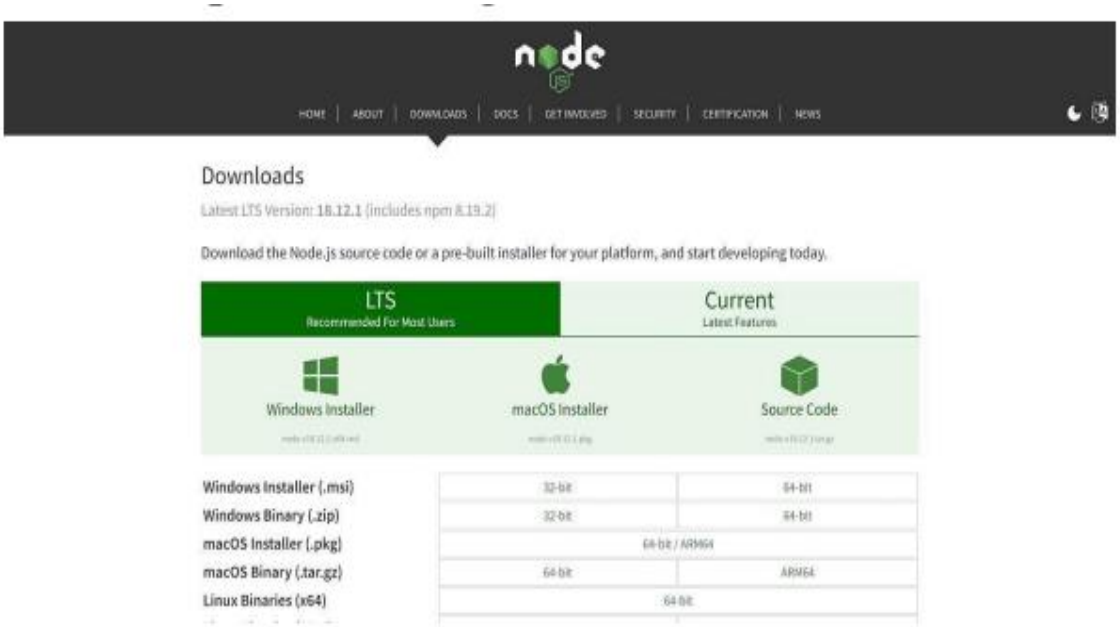
Event	Value	Format	Last Received
Humidity	{"randomNumber":36}	json	a few seconds ago
Temperature	{"Temperature":3}	json	a few seconds ago
Moisture	{"Moisture":54}	json	a few seconds ago
Humidity	{"randomNumber":70}	json	a few seconds ago
Temperature	{"Temperature":68}	json	a few seconds ago

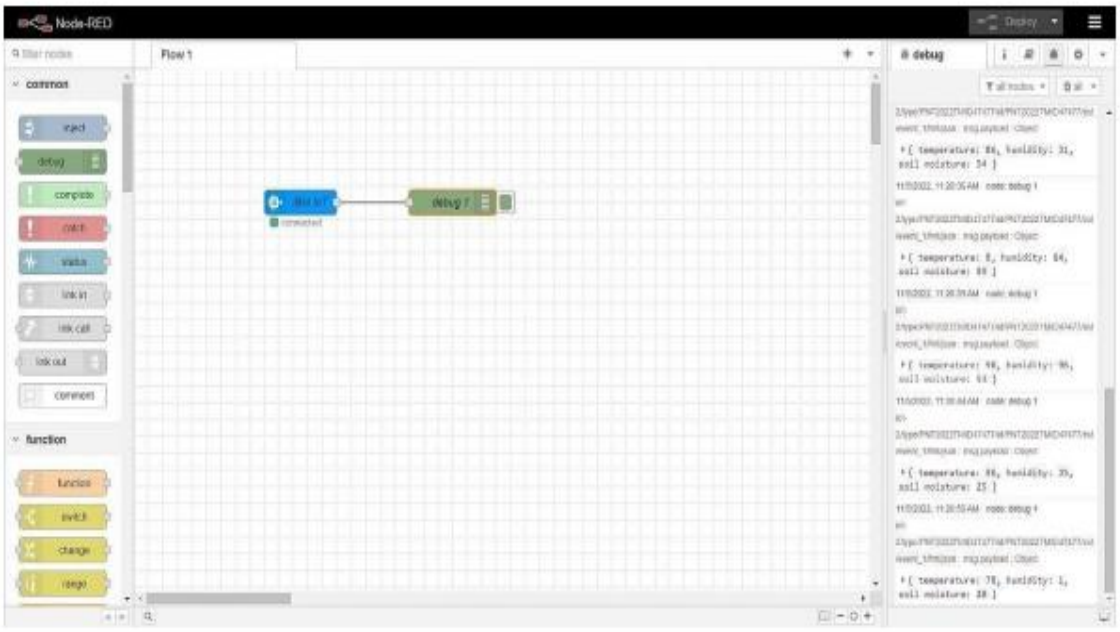
TESTING

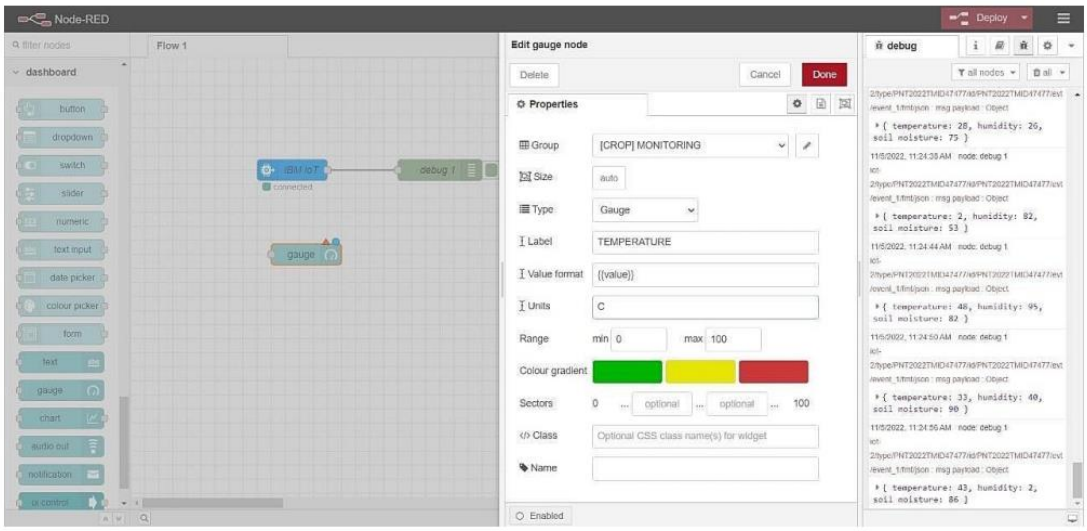
TEST CASES:

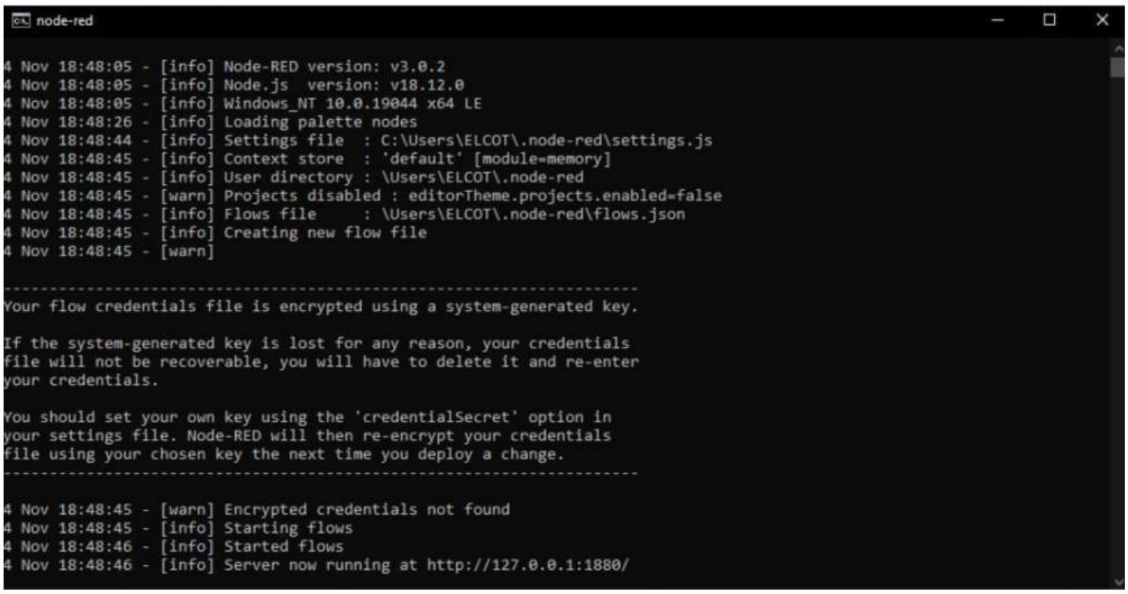
sno	parameter	Values	Screenshot
1	Model summary	-	
2	accuracy	Training accuracy-95% Validation accuracy-72%	
3	Confidence score	Class detected-80% Confidence score-80%	

USER ACCEPTANCE TESTING






Node-RED
Deploy
Flow 1
dashboard
button
dropdown
switch
slider
numeric
text input
date picker
colour picker
form
text
gauge
chart
audio out
notification
ui control
IBM IoT
connected
debug 1
gauge
Edit gauge node
Delete
Cancel
Done
Properties
Group
[CROP] MONITORING
Size
auto
Type
Gauge
Label
TEMPERATURE
Value format
{{value}}
Units
C
Range
min 0
max 100
Colour gradient
Sectors
0
optional
optional
100
Class
Optional CSS class name(s) for widget
Name
Enabled
debug
all nodes
{ temperature: 28, humidity: 26, soil moisture: 75 }
{ temperature: 2, humidity: 82, soil moisture: 53 }
{ temperature: 48, humidity: 95, soil moisture: 82 }
{ temperature: 33, humidity: 40, soil moisture: 90 }
{ temperature: 43, humidity: 2, soil moisture: 86 }


node-red
4 Nov 18:48:05 - [info] Node-RED version: v3.0.2
4 Nov 18:48:05 - [info] Node.js version: v18.12.0
4 Nov 18:48:05 - [info] Windows_NT 10.0.19044 x64 LE
4 Nov 18:48:26 - [info] Loading palette nodes
4 Nov 18:48:44 - [info] Settings file : C:\Users\ELCOT\.node-red\settings.js
4 Nov 18:48:45 - [info] Context store : 'default' [module=memory]
4 Nov 18:48:45 - [info] User directory : \Users\ELCOT\.node-red
4 Nov 18:48:45 - [warn] Projects disabled : editorTheme.projects.enabled=false
4 Nov 18:48:45 - [info] Flows file : \Users\ELCOT\.node-red\flows.json
4 Nov 18:48:45 - [info] Creating new flow file
4 Nov 18:48:45 - [warn]
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
4 Nov 18:48:45 - [warn] Encrypted credentials not found
4 Nov 18:48:45 - [info] Starting flows
4 Nov 18:48:46 - [info] Started flows
4 Nov 18:48:46 - [info] Server now running at http://127.0.0.1:1880/

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, Watson simulator, 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 time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "ncgqpp"
deviceType = "arduino"
deviceId = "12345"
authMethod = "token"
authToken = "1234567890"

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

    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroff":
        print ("motor is off")
```

```
    else :
        print ("please send proper command")


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()
time.sleep(2)


def myOnPublishCallback():
    print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %pulse,"SoilMoisture = %s %%" % soil,"to IBM Watson")
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=None)
    time.sleep(1)


while True:
        #Get Sensor Data from DHT11
        temp=random.randint(0,100)
        pulse=random.randint(0,100)
```

```
        soil=random.randint(0,100)
        data = { 'temp' : temp, 'pulse': pulse ,'soil':soil}

        myOnPublishCallback()

if not success:
    print("Not connected to IoTF")
    time.sleep(5)

deviceCli.commandCallback = myCommandCallback

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

NODE-RED FLOW:

```
{
 {
"id":"625574ead9839b34 ",
 "type":"ibmiotout",
 "z":"630c8601c5ac3295",
 "authentication":"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_button",
"z":"630c8601c5ac3295",
"name":"",
"group":"716e956.00eed6c",
"order":2,
"width":"0",
"height":"0",
"passthru":false,
"label":"MotorON",
"tooltip":"", "color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"motoron\"}",
"payloadType":"str",
"topic":"motoron",
"topicType":"s tr",
"x":360,
"y":160,
"wires":[["625574ead9839b34"]]},

```
 {
"id":"659589baceb4e0b0",
 "type":"ui_button",
"z":"630c8601c5ac3295",
 "name":"",
 "group":"716e956.00eed6c",
 "order":3,
 "width":"0",
 "height":"0",
 "passthru":true,
"label":"MotorOF F",
"tooltip":"",
 "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":"ibmiotin",
"z":"03acb6ae05a0c712",
"authentication":"apiKey",

"apiKey":"ef745d48e395ccc0",
"inputType":"evt",
"logicalInterface":"",
"ruleId":"",
"deviceId":"b827ebd607b5",
"applicationId":"",
"deviceType":"weather_monitor",
"eventType":"+",
"commandType":"",
"format":"json",
"name":"IBMIoT",
"service":"registered",
"allDevices":"",
"allApplications":"",
"allDeviceTypes":"",
"allLogicalInterfaces":"",
"allEvents":true,
"allCommands":"",
"allFormats ":"",
"qos":0,
"x":270,
"y":180,
"wires":[["50b13e02170d73fc",
"d7da6c2f5302ffaf",
"a949797028158f3f",
"a71f164bc3 78bcf1"]] },
{

"id":"50b13e02170d73fc ",

"type":"function",

"z":"03acb6ae05a0c712 ",

"name":"Soil Moisture",

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

"outputs":1,

"noerr": 0,

"initialize ":"",

"finalize":"",

"libs":[], "x":490,

"y":120,

"wires":[["a949797028158f3f","ba98e701f55f04fe"]] }

{

"id":"d7da6c2f5302ffaf",

"type":"function",

"z":"03acb6ae05a0c712",

"name":"Humidity",

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

"outputs":1,

"noerr": 0,

"initialize ":"",

"finalize":"",

"li bs ":[ ],

"x ": 48 0,

"y":260,

"wires":[["a949797028158f3f","70a5b076eeb80b70"]] },

```
 {
"id":"a949797028158f3f ",
"type":"debug",
"z":"03acb6ae05a0c712 ",
"name":"IBMo/p",
 "active":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",
 "title":"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":"function",

"z":"03acb6ae05a0c712",

"name":"Temperature",

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

"outputs":1,

"noerr": 0,

"initialize ":"",

"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",
"title":"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",
"title":"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":"httpin",
"z":"03acb6ae05a0c712 ",
"name":"",
"url":"/sensor",
"method":"ge t",
"upload":fals e,
"swaggerDoc" :"",
"x":370,
"y":500,
"wires":[["18a8cdbf7943d27a"]] },

```
{
"id":"18a8cdbf7943d27a",
"type":"function",
 "z":"03acb6ae05a0c712",
 "name":"httpfunction",
"func":"msg.payload{\"pulse\":global.get('p'),\"temp\":global.get('t'),\"soil\":glo
bal.get( 's')};\nreturn msg;",
"outputs":1,
 "noerr":0,
 "initialize":"",
 "finalize":"",
"li bs ":[ ],
 "x ": 63 0,
"y":500,
"wires":[["5c7996d53a445412"]] },
{
"id":"5c7996d53a445412 ",
 "type":"httpresponse",
 "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 LINK:

https://github.com/IBM-EPBL/IBM-Project-13699-1659526306