IOT BASED SMART CROP PROTECTION SYSTEMFOR AGRICULTURE

Team ID : PNT2022TMID28230

- Team Leader : FREDRICK PAUL A
- Team member : DHIYANESHWARAN R
- Team member : FARRY J
- Team member : HARIKRISHNAN V

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 tobarricadeentire 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 and manual surveillance andvarious such exhaustive and dangerous method.

REFERENCES:

i. Reshma S, Ramya J, Swathi S, Srinidhi R N, IJISRT International Journal of Innovative Science and Research Technology, April 2019.

ii. S. R. Chourey, P. A. Amale, N. B. Bhawarkar, International Journal of Electronics, Communication & Soft Computing Science and Engineering IJECSCSE 2017.

iii. Srikanth N, Aishwarya, Kavita H M, Rashmi Reddy K, Soumya D B. International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE) April 2019.

iv. Premjyoti G, Patil,B, IJISRT(International Journal of Innovative Science and Research Technology),April 2020.

PROBLEM STATEMENT DEFINITION STATEMENT:

IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

- Crop protection is the general method or the practice of protecting the crop yields from different agents including pests, weeds, plant diseases, and other
- Apart from crops, agricultural fields would have weeds, small animals like rats, mites, insects, pests, disease-causing pathogens and frequently raided by birds. All these factors are mainly responsible for the loss or damage to the crops. Thus to yield high crop production, farmers need to protect the crop from these pests. Hence crop protection management is important before, during and after the cultivation.

- The traditional agriculture and allied sector cannot meet the requirements of modern agriculture which requires high-yield, high quality and efficient output. Thus, it is very important to turn towards modernization of existing methods and using the information technology and data over a certain period to predict the best possible productivity and crop suitable on the very particular land.

CAUSES OF CROP DAMAGE:

- ✓ Several factors pose significant risk to farms leading to yield reduction when they are not correctly monitored and well managed. These factors can be grouped into three categories which are technological, biological and environmental .
- ✓ The pressure to increase crop production in many countries, has resulted in the expansion of land area dedicated to agriculture and the intensification of cropland management through practices such as irrigation, use of large quantities of inputs like inorganic fertilizers and synthetic chemicals for pest and weed control leads to **affect the crop fertilization.**
- ✓ Environmental factors that affect crop cultivation include light, temperature, water, humidity and nutrition. It's important to understand how these factors affect plant growth and development.

- ✓ Either directly or indirectly, most crop protection problems are caused by environmental stress. In some cases, poor environmental conditions (e.g., too little water) damage a plant directly. In other cases, environmental stress weakens a plant and makes it more susceptible to disease or insect attack.
- ✓ Crops in the farms are many times devastated by the wild as well as domestic animals and low productivity of crops is one of the reasons for this.

IMPACTS OF CROP DAMAGE:

- ✓ Crop damage from extreme heat leads to affect the plant growth.
- ✓ Increased infestations of insects,animals,pests and birds leads to affect the food productivity.
- ✓ It will directly effect the production of crops and hence food. The uprise of hunger will be the main effect.
- ✓ Loss of job of farmers and other food market owners will also be the result.
- ✓ Negative effect on economy of country will also be seen as the food will have to be bought from other countries. Sale of our country food will also diminish.
- ✓ Poverty will increase and hence the suicide rate of farmers.

IMPACTS OF CROP DAMAGE:

- ✓ Crop damage from extreme heat leads to affect the plant growth.
- ✓ Increased infestations of insects,animals,pests and birds leads to affect the food productivity.
- ✓ It will directly effect the production of crops and hence food. The uprise of hunger will be the main effect.

- ✓ Loss of job of farmers and other food market owners will also be theresult.
- ✓ Negative effect on economy of country will also be seen as the food willhave to be bought from other countries. Sale of our country food will also diminish.
- ✓ Poverty will increase and hence the suicide rate of farmers.

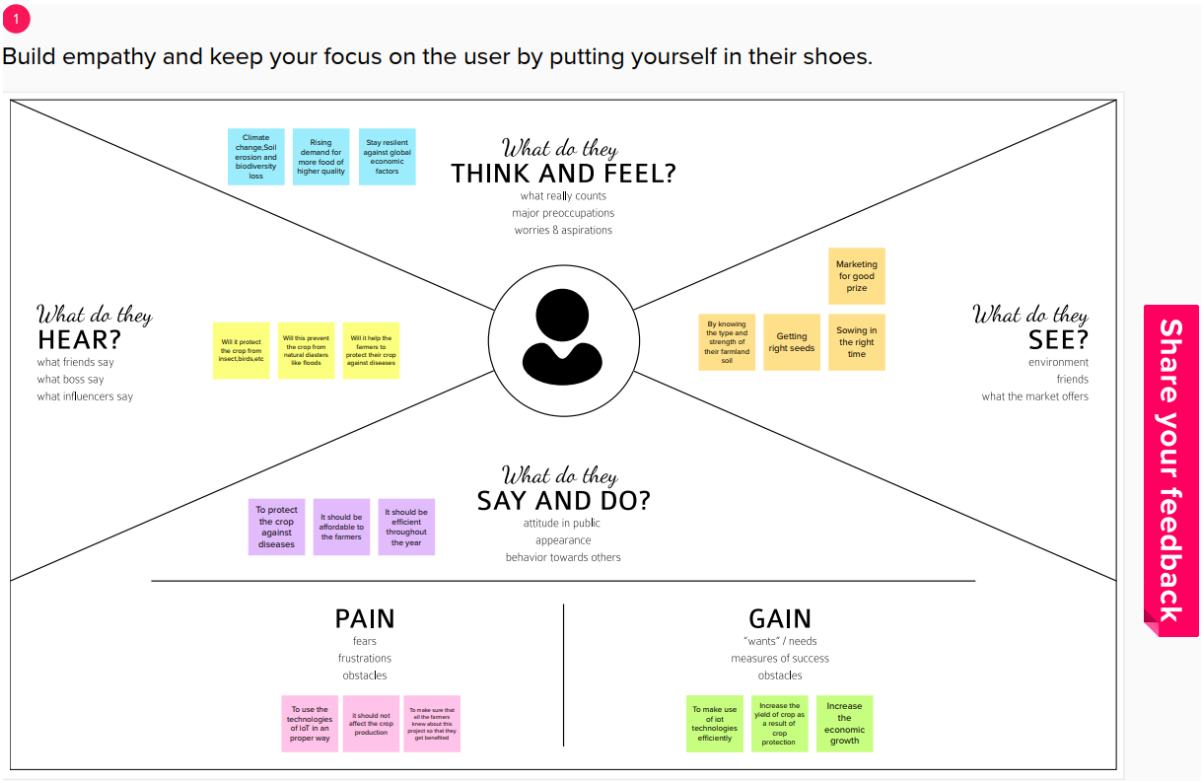
PREVENT THE CROP DAMAGE IN FARMING:

- ✓ Crop protection allows farmers to monitor climate change and noticethe appearance of dangerous weeds, pests, or diseases timely.
- ✓ Analysing different parameters using the available technology and practice the cultivation of crops depends on metric values obtained.
- ✓ Crop protection solutions use AI to collect and analyze large amounts of data. It provides farmers with detailed culture and soil conditions for plant protection planning. Crop Monitoring is an excellent example of the helpfulness of remote sensing for crop protection. The platform effectively takes care of the health of the soil, reducing the risk of plant diseases and pests. Moreover, it provides data on plant health, moisturelevels, and weather changes.
- ✓ Using different sensors and animal detection method can collect thedata of the field 24/7 and perform the crop cultivation manually or automatically when ever we want.This increase the productivity offarming
- ✓ High yield , enhanced monitoring of plants like maintaining the moisturecontent of plant growth, time consuming are achieved by modernized agriculture technology.

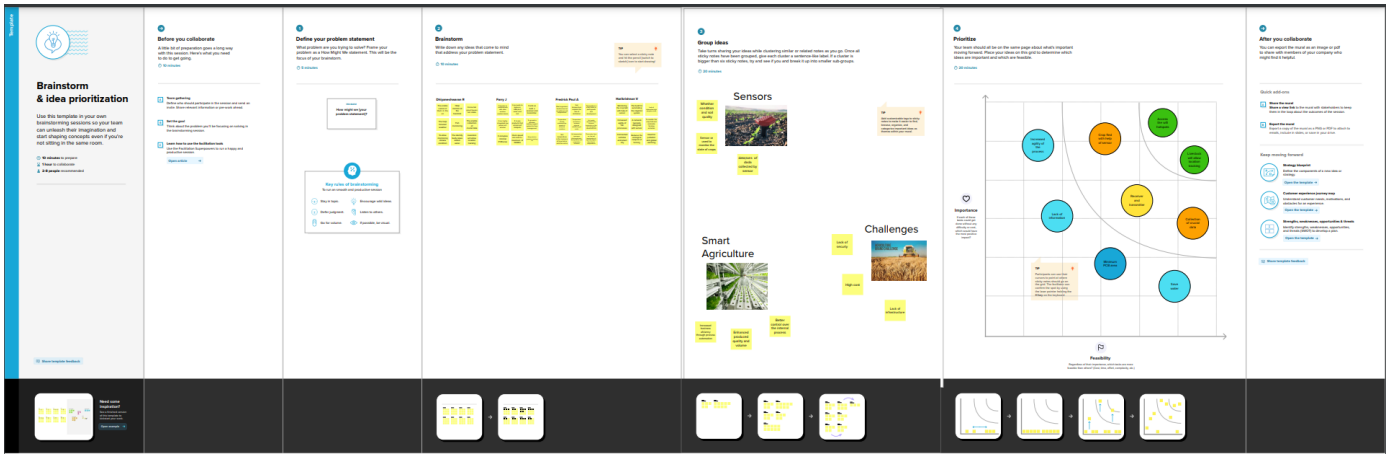
.

IDEATION AND PROPOSED SOLUTION

EMPATHY MAP CANVAS:



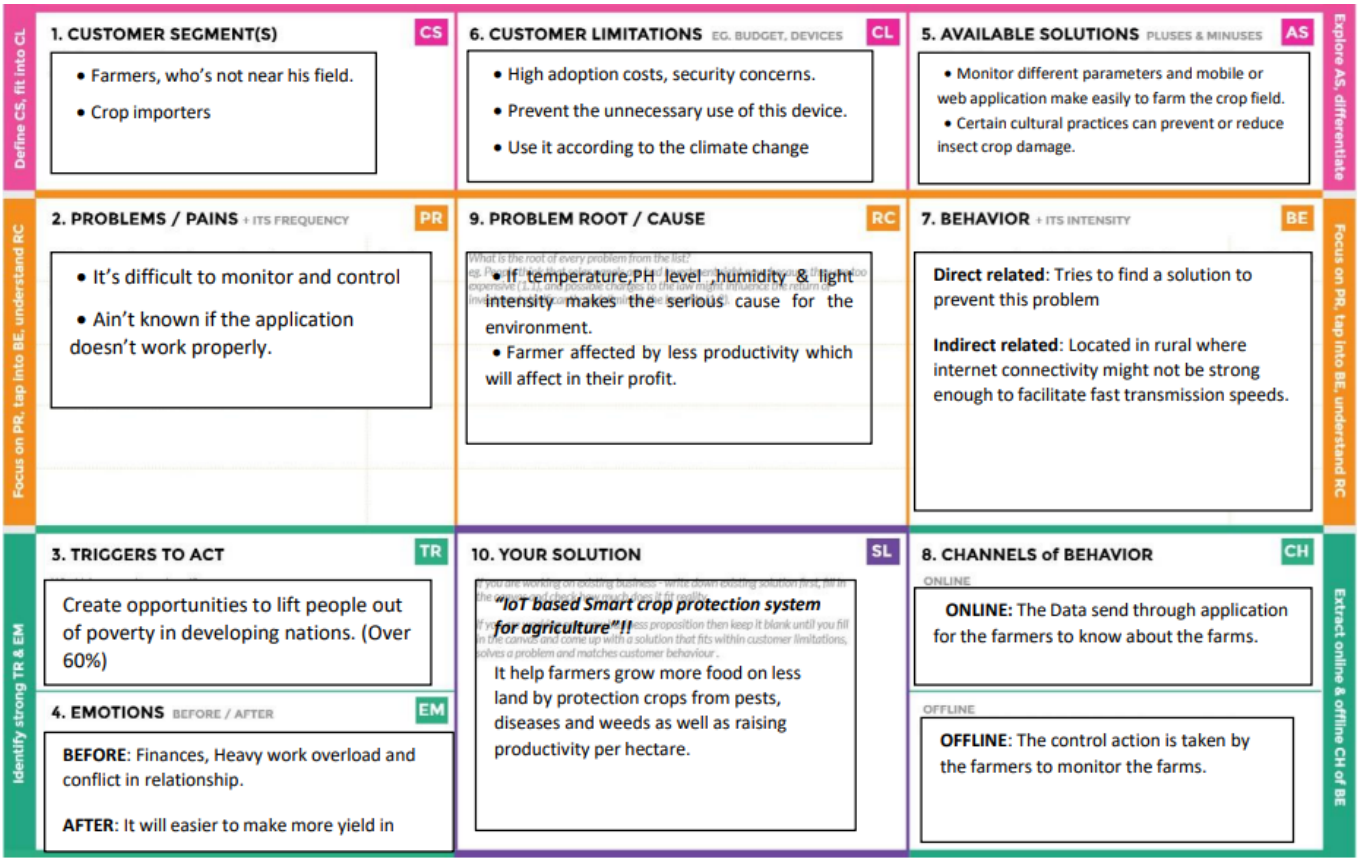
IDEATION AND BRAINSTORMING:



PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	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.
2.	Idea / Solution description	Crop protection solutions use CNN technique to collect and analyze large amounts of data. It provides farmers with detailed culture and soil conditions for plant protection planning using sensors.
3.	Novelty / Uniqueness	The design system will not be dangerous to animal and human being, and it protects farm.
4.	Social Impact / Customer Satisfaction	User friendly and cost effective. Not much effort is required for the farmer to protect the field, and for crop irrigation.
5.	Business Model (Revenue Model)	We provide 24x7 crop protection and automatic sprinkler system. This model increases the yield and productivity, thereby increase the profit.
6.	Scalability of the Solution	This system is capable of continuously monitoring the soil conditions, also helps in farmers to increase the crop yield. This system can be further improved, so that the entire irrigation system becomes automatic.

PROBLEM SOLUTIONFIT:



REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENT:

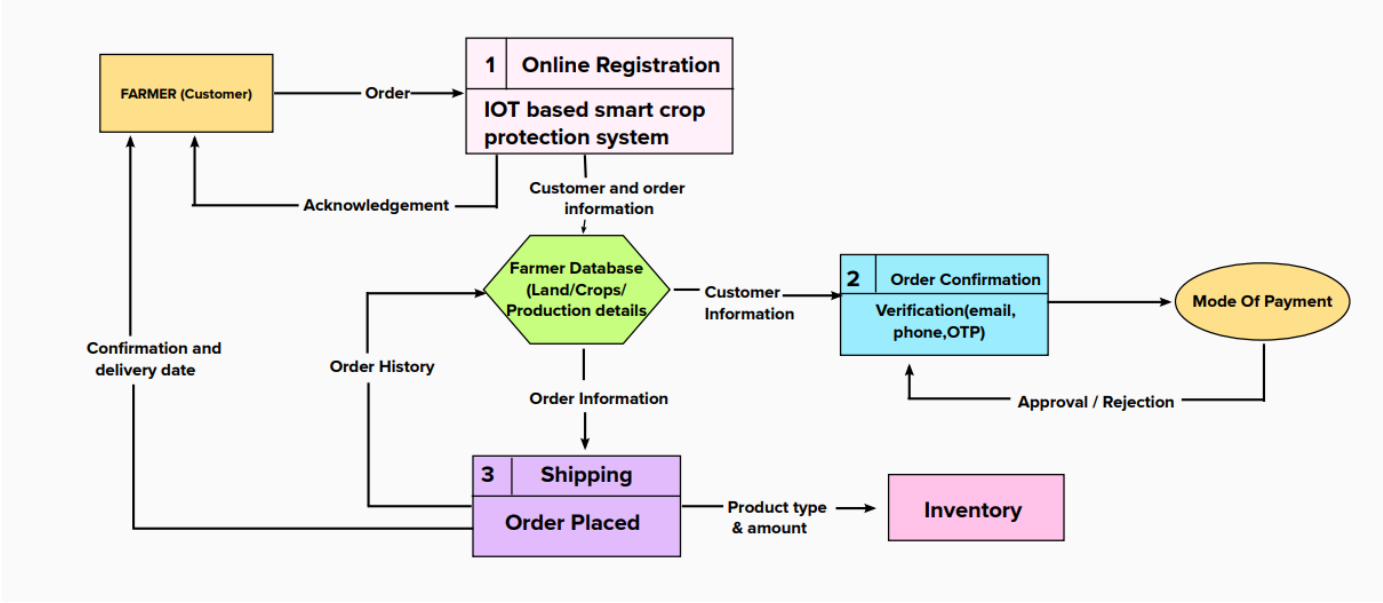
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User registration	Install the app signing up with Gmail Create a profile Observe the guidelines
FR-2	User Confirmation	Email confirmation required Reassurance via OTP
FR-3	Interface sensor	Connect the sensor and the application so that when animals enter the field, an alarm is generated.
FR-4	Accessing datasets	Sets of data are obtained from the cloudant DB.
FR-5	Mobile application	Mobile applications can be used to control field sprinklers and motors.

NON FUNCTINAL REQUIREMENT:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The project's contribution to farm protection is demonstrated through the smart protection system.
NFR-2	**Security**	This project was created to protect the crops from animals.
NFR-3	**Reliability**	With the help of this technology, farmers will be able to safeguard their lands and avoid suffering substantial financial losses. They will also benefit from higher crop yields, which will improve their economic situation.
NFR-4	**Performance**	When animals attempt to enter the field, IOT devices and sensors alert the farmer via message. We also utilise an SD card module that helps to store a specific sound to frighten the animals.
NFR-5	**Availability**	We can defend the crops against wild animals by creating and implementing resilient hardware and software.
NFR-6	**Scalability**	This system's integration of computer vision algorithms with IBM cloudant services makes it more efficient to retrieve photos at scale, enhancing scalability.

PROJECT DESIGN

DATA FLOW DIAGRAM:



SOLUTION AND TECHNICAL ARCHITECTURE:

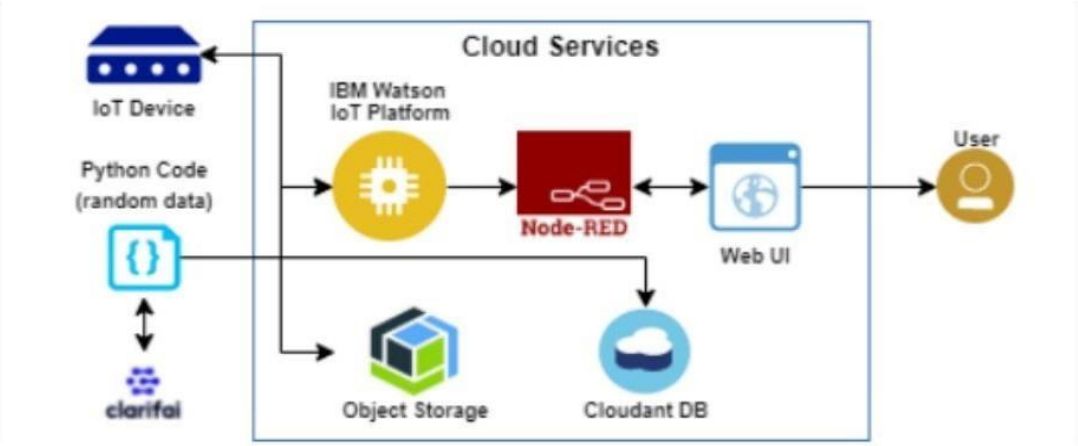


TABLE-1:

sno	components	description	Technology
1	User interface	Interacts with iot device	Html,css,angular js etc..
2	Application logic-1	Logic for a process in the application	Python
3	Application logic-2	Logic for process in the application	Clarifai
4	Application logic-3	Logic for process in the application	IBM Waston Iot platform
5	Application logic-4	logic for the process	Node red app service
6	User friendly	Easily manage the net screen appliance	Web uI

TABLE-2: APPLICATION AND CHARACTERISTICS

sno	Characteristics	Description	Technology
1	Open source framework	Open source framework used	Python
2	Security implementations	Authentication using encryption	Encryptions
3	Scalable architecture	The scalability of architecture consists of 3 models	Web UI Application server-python, clarifai Database server-ibm cloud services.
4	Availability	It is increased by cloudant database	IBM cloud services

USER STORIES:

SPRINT	FUNCTIONAL REQUIREMENT	USER STORY NUMBER	USER STORY/TASK	STORY POINTS	PRIORITY
Sprint-1		US-1	Create the IBM Cloud services which are being used in this project.	7	high
Sprint-1		US-2	Create the IBM Cloud services which are being used in this project.	7	high
Sprint-2		US-3	IBM Watson IoT platform acts as the mediator to connect the web application to IoT devices, so create the IBM Watson IoT platform.	5	medium
Sprint-2		US-4	In order to connect the IoT device to the IBM cloud, create a device in the IBM Watson IoT platform and get the device credentials	6	high
Sprint-3		US-1	Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.	10	high
Sprint-3		US-3	Create a Node-RED service	8	high
Sprint-3		US-2	Develop a python script to publish random sensor data such as temperature, moisture, soil and humidity to the IBM IoT platform	6	medium
Sprint-3		US-1	After developing python code, commands are received just print the statements which represent the control of the devices.	8	high
Sprint-4		US-3	Publish Data to The IBM Cloud	5	high
Sprint-4		US-2	Create Web UI in Node- Red	8	high
Sprint-4		US-1	Configure the Node-RED flow to receive data from the IBM IoT platform and also use Cloudant DB nodes to store the received sensor data in the cloudant DB	6	high

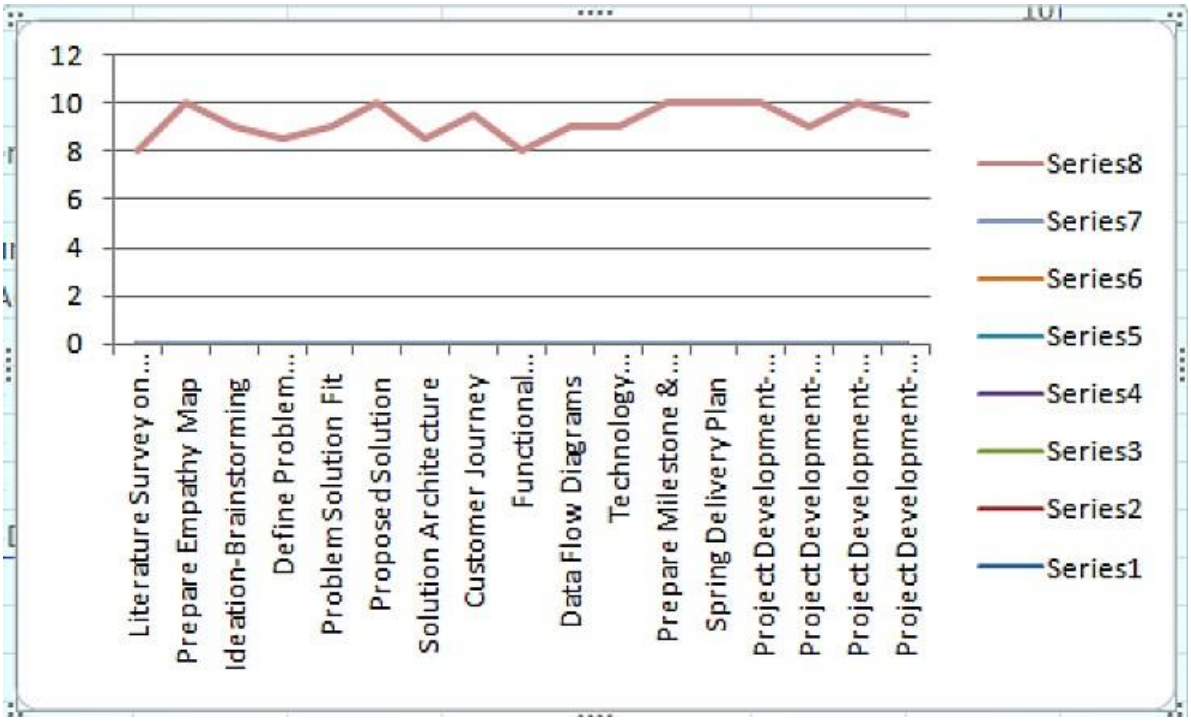
PROJECT PLANNING AND SCHEDULING

SPRINT PLANNINGAND ESTIMATION:

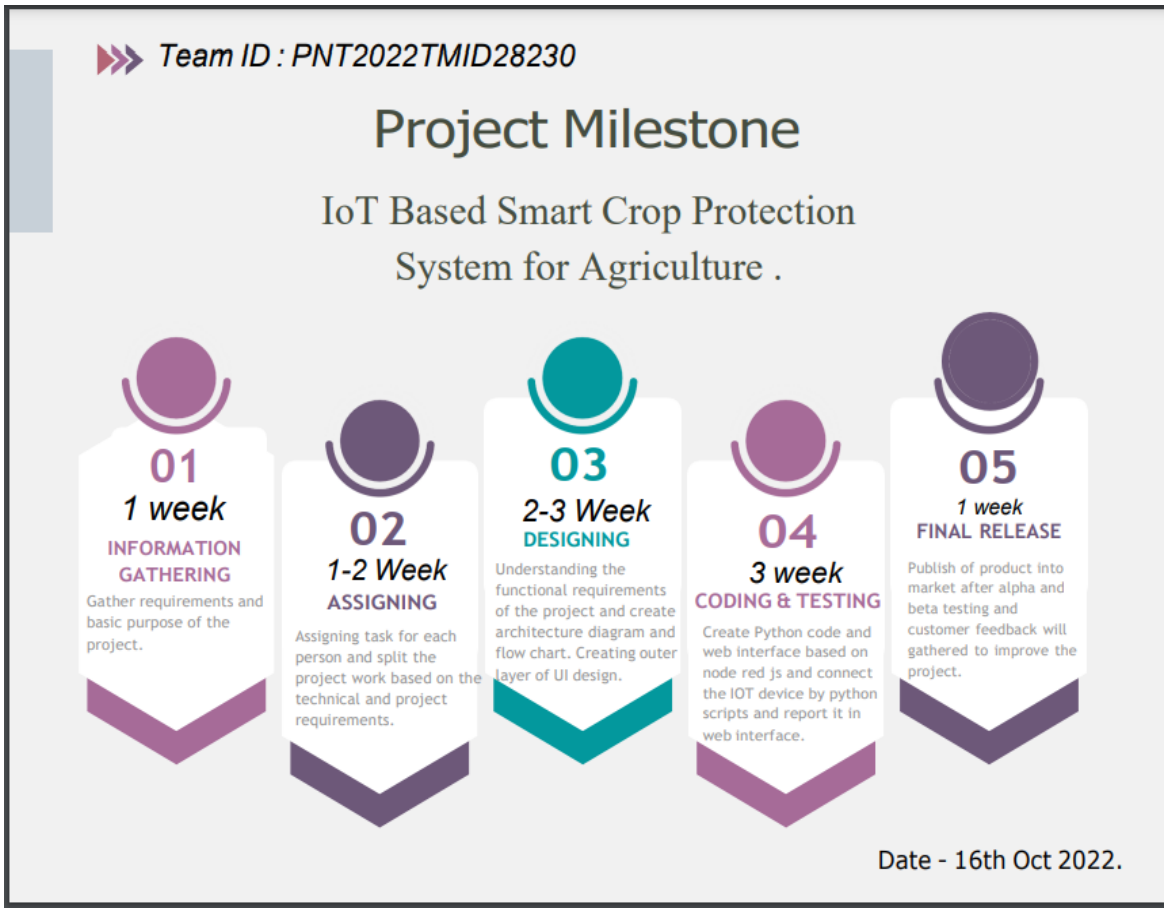
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022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

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity iteration unit (story points per day)

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



MILESTONE & ACTIVITY LIST AND SPRINT DELIVERY PLAN:



ACTIVITY LIST

S.No	Activity Title	Activity Description	Duration
1.	Understanding the Project Requirement	Assign the team members & create the repository in GithHub. Assign the task to each member and teach how to use and open access the GitHub and IBM Career Education.	1 Week
2.	Starting of Project	Advice student to attend classes of IBM portal create and develop an rough diagram based on the project description and gather information of IOT and IBM project.	1 week
3.	Attend classes	Team members & team lead must watch and learn from classes provided by IBM and Nalaya thiran and must gain access of MIT license for their project.	4 Week
4.	Budget and scope of the project	Budget & analyse the use of IOT in the project and discuss with the team for budget prediction to predict the favourability of the customer to buy the product for efficient use of the product among the environment.	1 week

IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE .

Sprint Plan

WEEK 1
22-27 AUG 2022

Preparation Phase

- Pre requisites .
- Environment Setup etc.

WEEK 2-4
29 AUG-17 SEP 2022

Ideation Phase

- Literature Survey .
- Empathy map .
- Defining Problem Statement .
- Ideation .

WEEK 5-6
19 SEP - 1 OCT 2022

Project Design Phase-I

- Proposed Solution.
- Problem Solution Fit.
- Solution Architecture.

WEEK 7-8
3 OCT-15 OCT 2022

Project Design Phase-II

- Requirement Analysis .
- Customer Journey.
- Dataflow diagram.
- Technology Architecture.

WEEK-9
17 OCT-22 OCT 2022

Project Planning Phase

- Milestones & Activity List.
- Sprint Delivery Plan

WEEK 10-13
24 OCT-19 NOV 2022

Project Development Phase

- Coding & Solution
- Acceptance Testing
- Performance Testing.

Team ID : PNT2022TMID28230

CODING AND SOLUTIONING

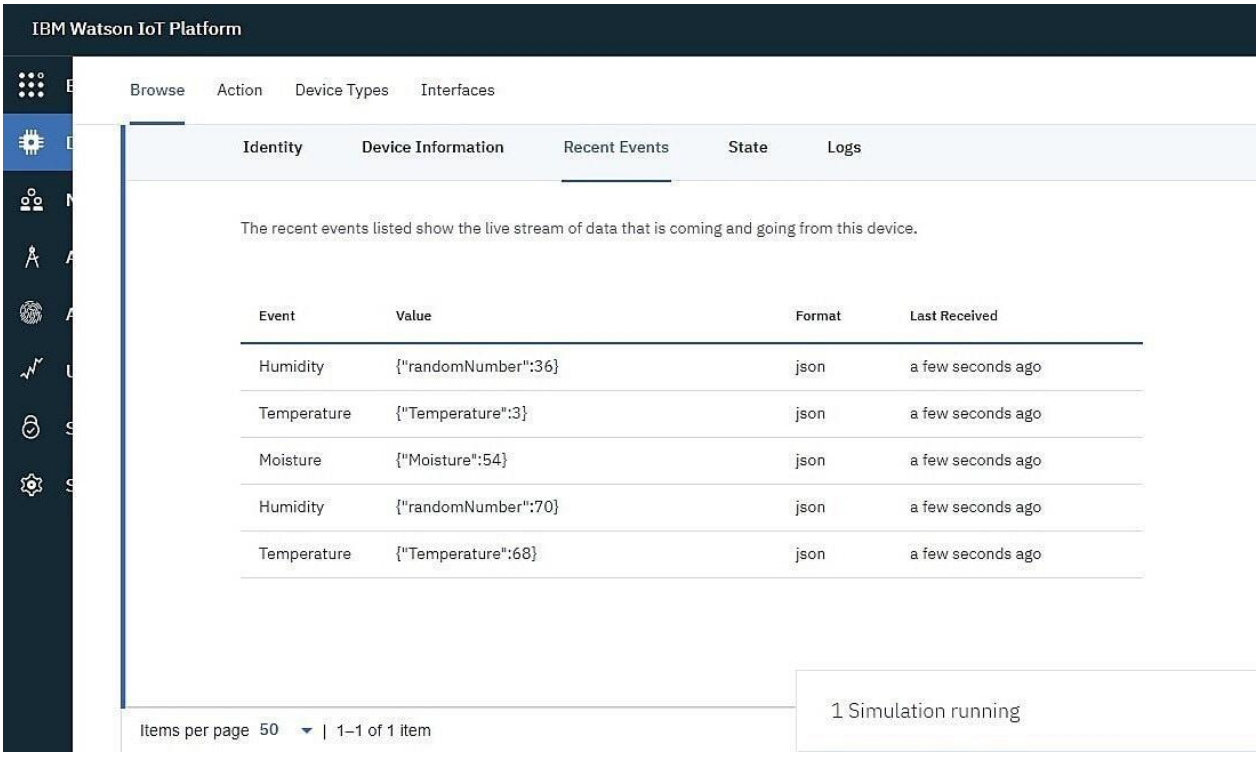
FEATURE-1

```
import random
import ibmiotf.application
import ibmiotf.device
from time import sleep
import sys
#IBM Watson Device Credentials.
organization = "op701j"
deviceType = "dhiyanesh"
deviceId = "dhiyanesh18"
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:
    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")

 #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("")
 #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 application from the cloud
deviceCli.disconnect()
```



Features

Output: Digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion 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 a3.3V regulator),but 5V is ideal in case the regulator has different specs.

BUZZER

Specifications

- RatedVoltage : 6V DC
- Operating Voltage : 4 to 8V DC

- Rated Current*: ≤30mA

- SoundOutput at 10cm* : ≥85dB
- Resonant Frequency : 2300 ±300Hz

- Tone: Continuous 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, pneumatic and electronic.

FEATURE-2:

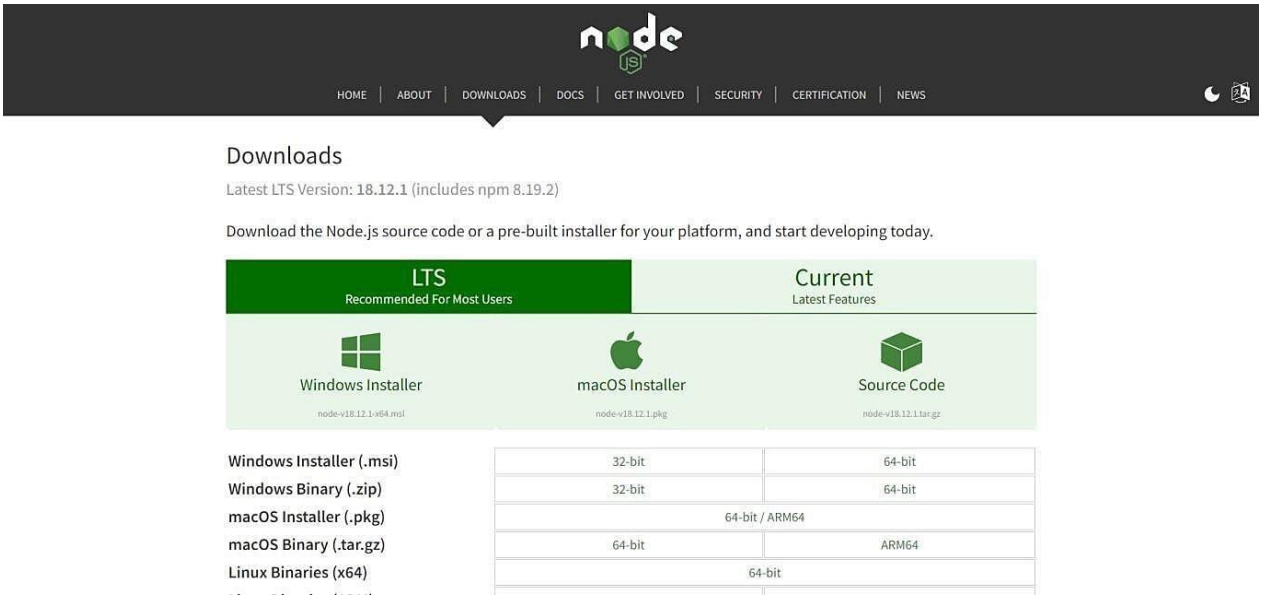
i. Goodsensitivity to Combustible gas in wide range .
ii. Highsensitivity to LPG, Propane and Hydrogen .
iii. Longlife and low cost.
iv. Simpledrive circuit.

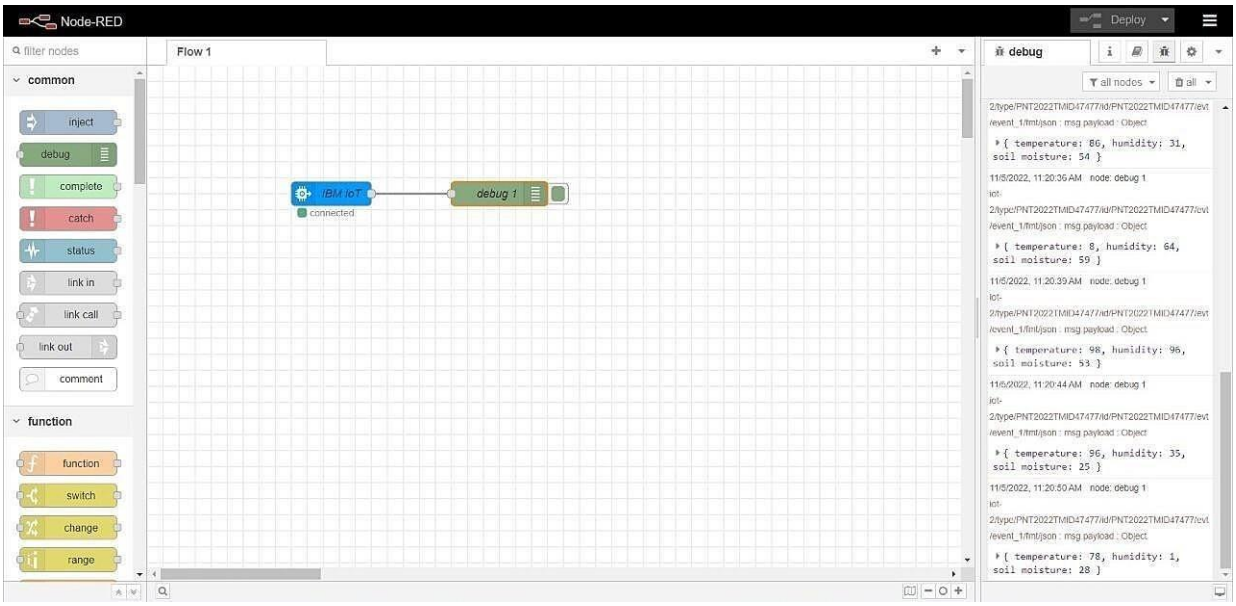
TESTING

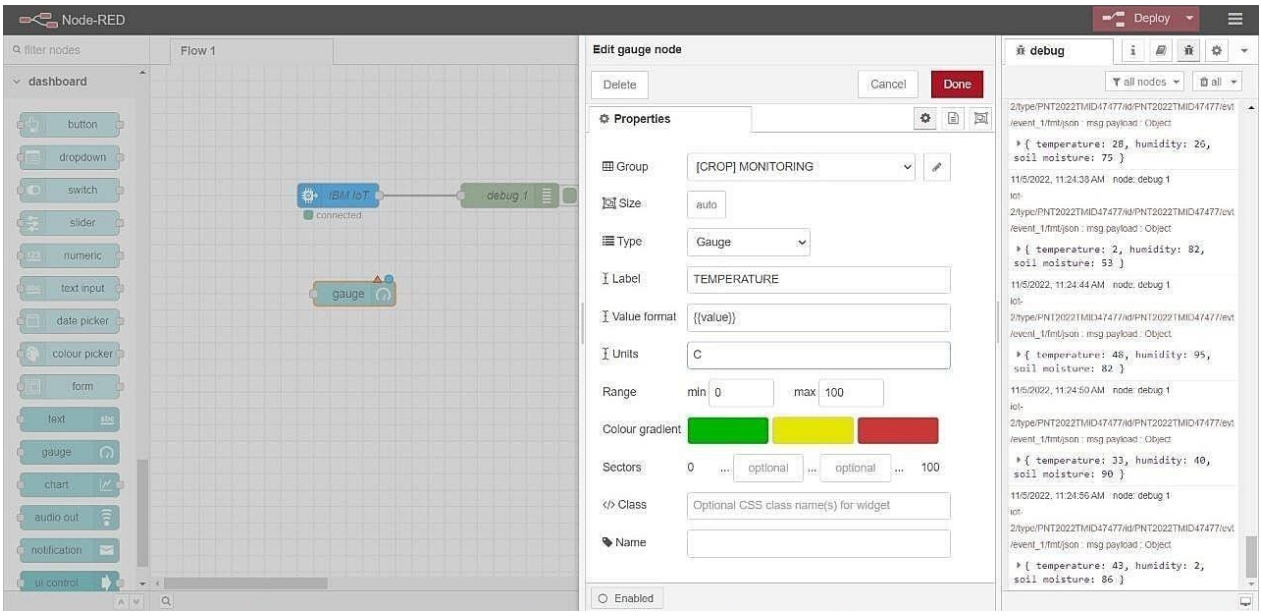
TEST CASES:

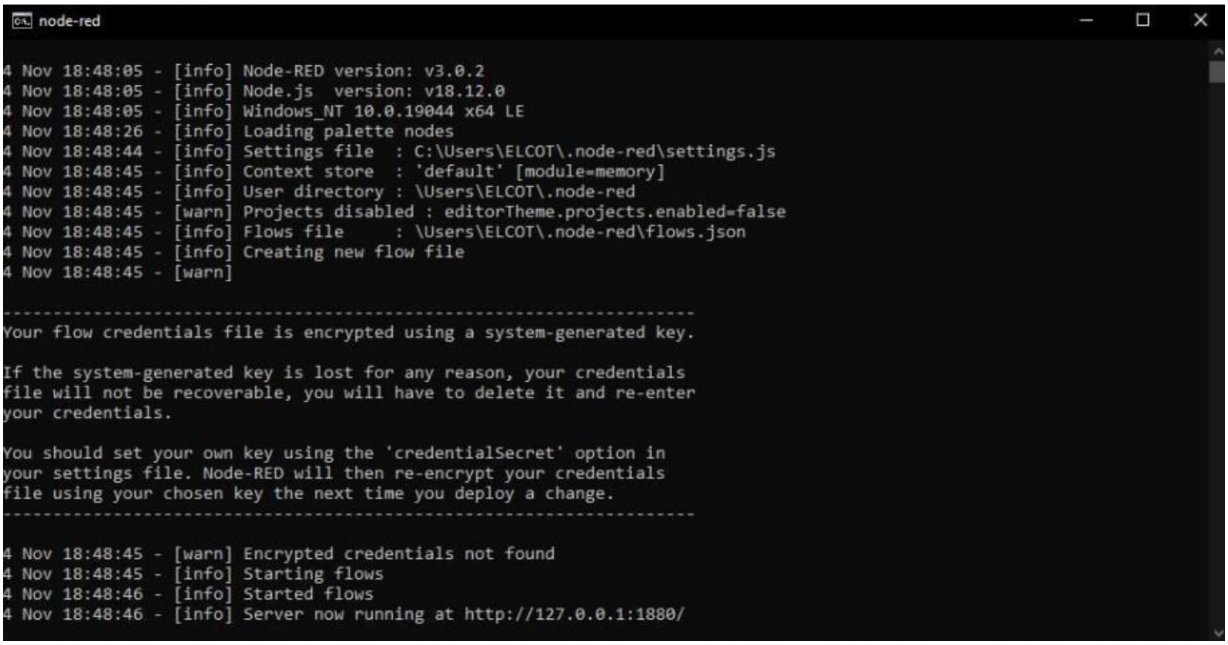
sno	parameter	Values	Screenshot
1	Model summary	-	
2	accuracy	Training accuracy-95% Validation accuracy-72%	
3	Confidence score	Class detected-80% Confidence score-80%	

User Acceptance Testing:






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
all
{ temperature: 28, humidity: 26, soil moisture: 75 }
11/5/2022, 11:24:38 AM node: debug 1
{ temperature: 2, humidity: 82, soil moisture: 53 }
11/5/2022, 11:24:44 AM node: debug 1
{ temperature: 48, humidity: 95, soil moisture: 82 }
11/5/2022, 11:24:50 AM node: debug 1
{ temperature: 33, humidity: 40, soil moisture: 90 }
11/5/2022, 11:24:56 AM node: debug 1
{ 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 will help 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 achieving better crop yields thus leading to their economic well being.

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 chance of not starving. 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 the environment 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 animal and fire can be detectedby cameras and if it comes towards farm then 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 be activated.

APPENDIX

SOURCE CODE

```
import time importsys import ibmiotf.application # toinstallpip
install ibmiotf importibmiotf.device


# Provide your IBM Watson Device Credentials organization = "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):  # function 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 requiredinformation: 'interval'")


    interval = cmd.data['interval']

elif cmd.command == "print":
if 'message' not in cmd.data:
            print("Error - commandis missing requiredinformation: 'message'")
            else:output = cmd.data['message']
            print(output)
```

```
try:

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


exceptException 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 oftype "greeting"
    10 times
deviceCli.connect()


while True:
        deviceCli.commandCallback = myCommandCallback


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

SENSOR.PY

```
    import time import
    sysimport
    ibmiotf.application
    importibmiotf.device
import random


    # Provide your IBM Watson Device Credentials organization = "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:
            deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
  "auth-method": authMethod, "auth-token": authToken}
  deviceCli = ibmiotf.device.Client(deviceOptions)
            #.....................................


exceptException 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 oftype "greeting"
  10 times
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")time.sleep(1)
```

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\":global.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 & Project Demo Link

https://github.com/IBM-EPBL/IBM-Project-23633-1659890374

https://github.com/IBM-EPBL/IBM-Project-23633-1659890374/tree/main/Final%20Deliverables/Demonstration%20video