

IoT Based Smart Crop Protection System For Agriculture

TEAM ID : PNT2022TMID34170

TEAM LEADER : R.Praveena

TEAM MEMBER : A.Sheeja

TEAM MEMBER : A.Sreeja

TEAM MEMBER : S.P.Subithra

Table of Contents

No	Title	Page No
-----------	--------------	----------------

1	INTRODUCTION	
1.1	Project Overview	5
1.2	Purpose	5
2	LITERATURE SURVEY	
2.1	Existing problem	
2.2	References	6
2.3	Problem Statement Definition	6
3	IDEATION & PROPOSED SOLUTION	
3.1	Empathy Map Canvas	7
3.2	Ideation & Brainstorming	8
3.3	Proposed Solution	11
3.4	Problem Solution fit	12
4	REQUIREMENT ANALYSIS	
4.1	Functional requirement	13
4.2	Non-Functional requirements	14
5	PROJECT DESIGN	
5.1	Data Flow Diagrams	15
5.2	Solution & Technical Architecture	16
5.3	User Stories	17
6	PROJECT PLANNING & SCHEDULING	
6.1	Sprint Planning & Estimation	18
6.2	Sprint Delivery Schedule	19
6.3	Burndown Chart	20
7	CODING & SOLUTIONING	
7.1	Coding	20
7.2	Result	26
8	Appendix	
8.1	Source Code	27
8.2	Result	33
9	ADVANTAGES & DISADVANTAGES	41

10	CONCLUSION	42
11	FUTURE SCOPE	43
12	GitHub & Project Demo Link	44

List of Figures

Figure	Name of the Figure	Page number
1	Empathy Map Canvas	7
2	Ideation and Brainstorming	8
3	Proposed solution	11
4	Problem Solution Fit	12
5	Data Flow Diagrams	16
6	Technical Architecture	17
7	Burndown Chart	20

INTRODUCTION

1.1 Overview

Agriculture plays a vital role in the Indian economy. However, farmers suicides in India is worrying. The expressed reasons in order of importance behind farmer suicides were debt, environment, low produce prices, poor irrigation, increased cost of cultivation, use of chemical fertilizers and crop failure. A farmers decision about which crop to grow is generally clouded by his intuition and other irrelevant factors like making instant profits, lack of awareness about market demand, overestimating a soils potential to support a particular crop and so on. The need of the hour is to design a system that could provide predictive insights to the Indian farmers, thereby helping them make an informed decision about which crop to grow. This calls for the need of smart farming, which requires use of IoT. Application of IoT in agriculture could be a life changer for humanity and the whole planet. Sensor data analytics drives transparency into agricultural processes, as farmers get precious insights on the performance of their fields, greenhouses, etc. Farming powered by Machine Learning with its high-precision algorithms is a new concept emerging today. Aiming to increase the quantity and quality of products, this cutting-edge movement makes sustainable productivity growth for everyone working in the agriculture.

1.2 Purpose

With this in mind, we propose system for Smart Management of Crop Cultivation using IoT and Machine Learning a smart system that can assist farmers in crop management by considering sensed parameters (temperature, humidity) and other parameters (soil type, location of farm, rainfall) that predicts the most suitable crop to grow in that environment.

LITERATURE SURVEY

2.1 Existing Problem

As per Climate Change Crop Yield Assumptions report in [9], Crop Yields are projected to decline, with the larger declines to be expected in several developing economies which includes Southeast Asia (-5 percent) and India (-5 percent). [10] The variation in on-farm losses across regions may be partly explained by the range of reasons which include infrastructure and marketing challenges, unsuitable harvest timing, unexpected harsh climatic conditions and unable to predict suitable crops for farming in such climates.

2.2 References

[Smart Crop Prediction using IoT and Machine Learning – IJERT](https://www.ijert.org/iot-in-agricultural-crop-protection-and-power-generation)

<https://www.ijert.org/iot-in-agricultural-crop-protection-and-power-generation>

<https://www.freeprojectz.com/arduino-raspberry/iot-project-humidity>

https://www.researchgate.net/publication/356704301_IoT-Based...

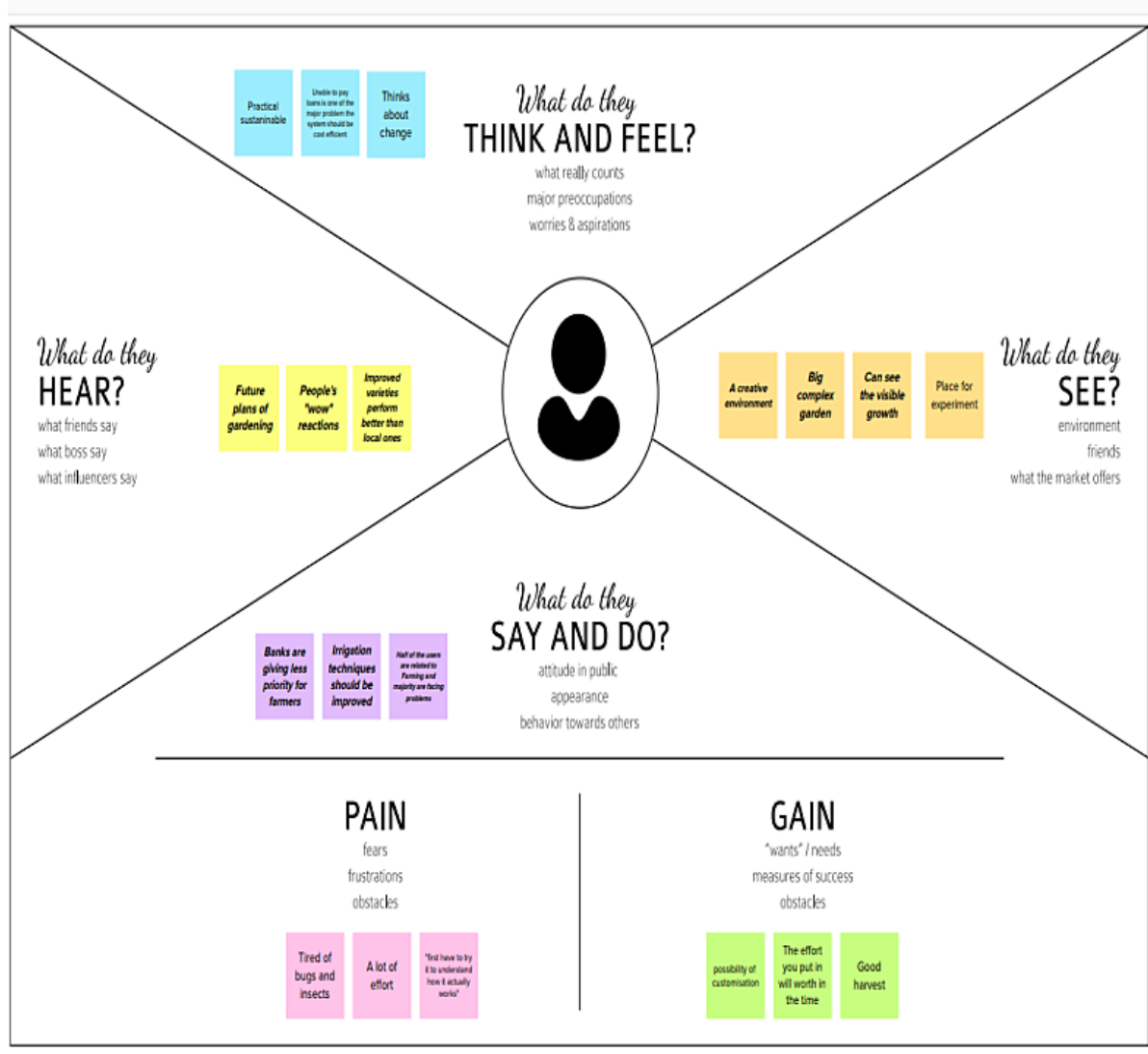
2.3 Problem Statement Definition

Crop protection **combines strategies, tools, and products that protect against various pests**. These include diseases, viruses, weeds, and insects. All of them can significantly lower or even kill plants. The best decision is to control the situation by reducing the risks rather than deal with the problem's consequences.

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

Empathy Map Canvas: An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

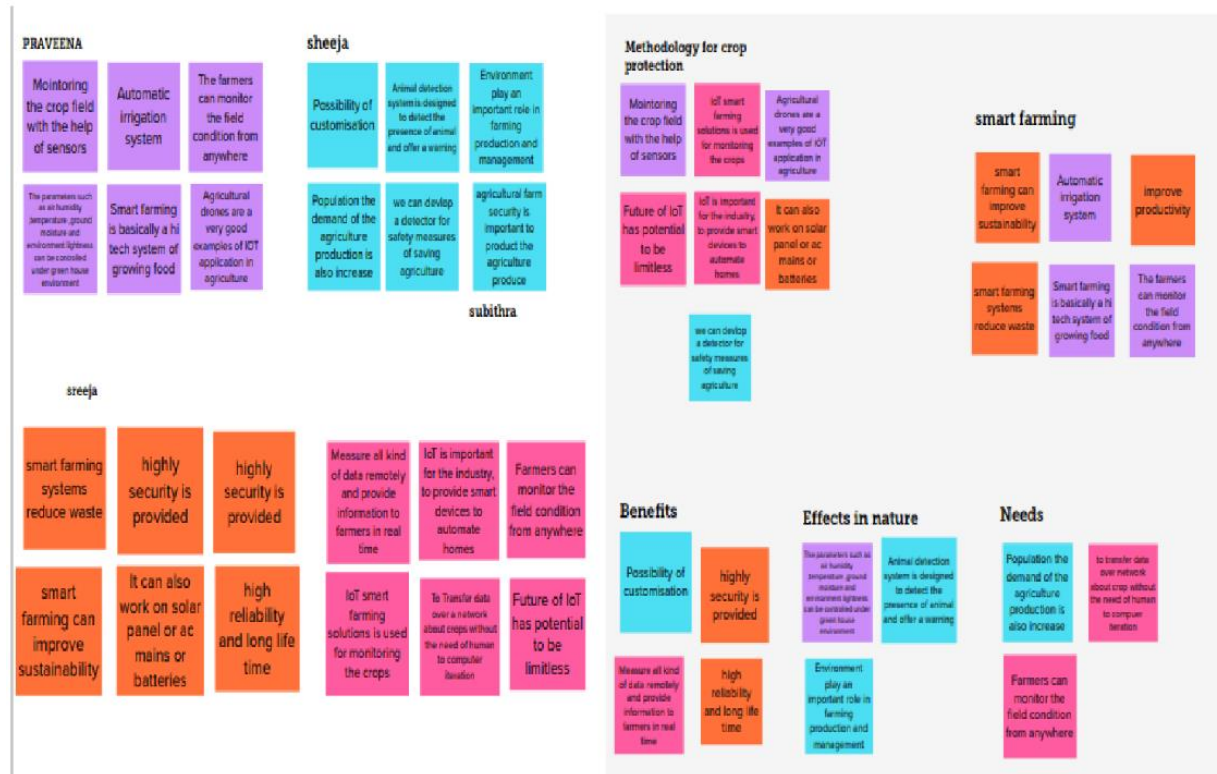


3.2 Ideation & Brainstorming

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

7

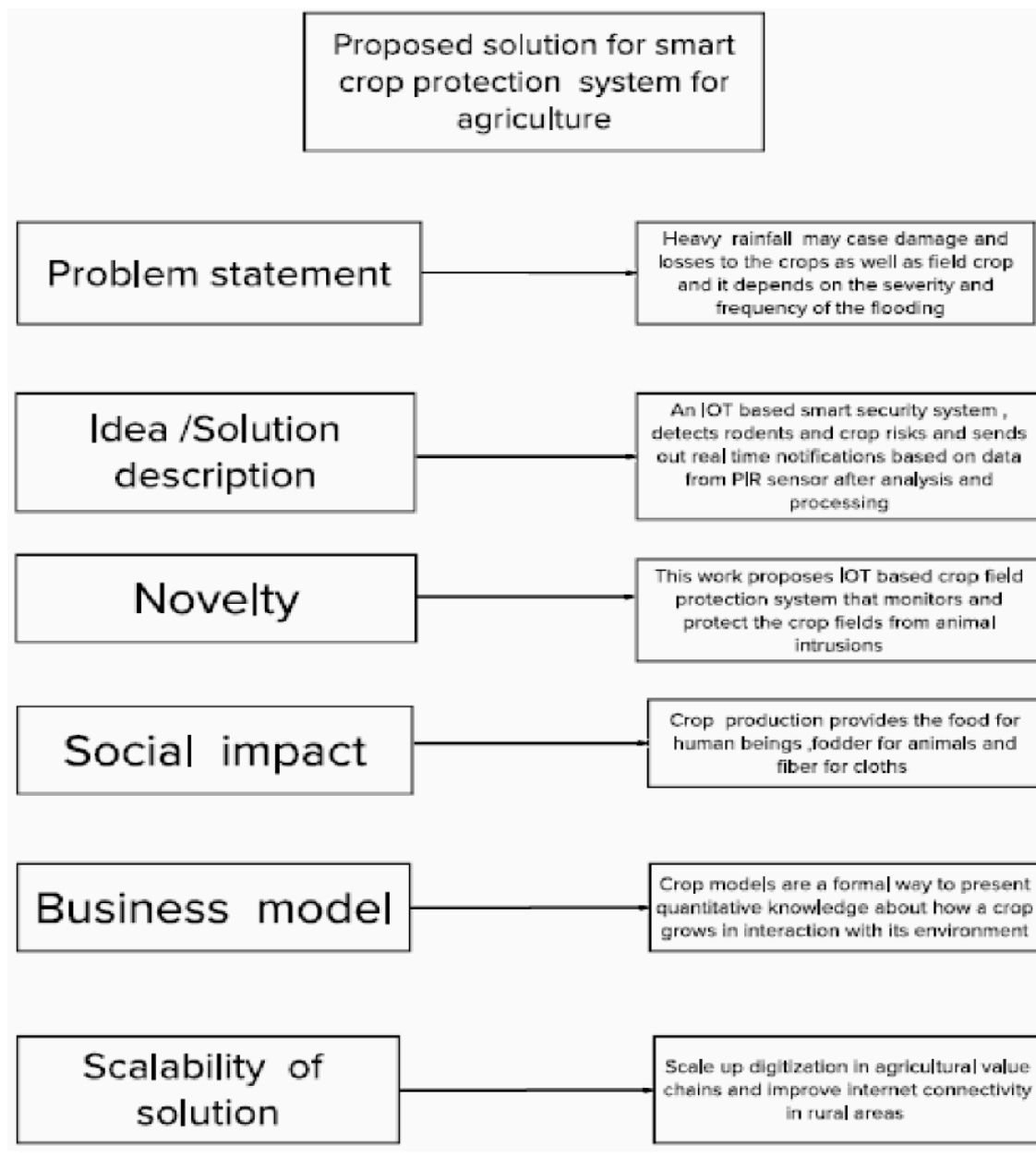
Step-2: Brainstorm, Idea Listing and Grouping



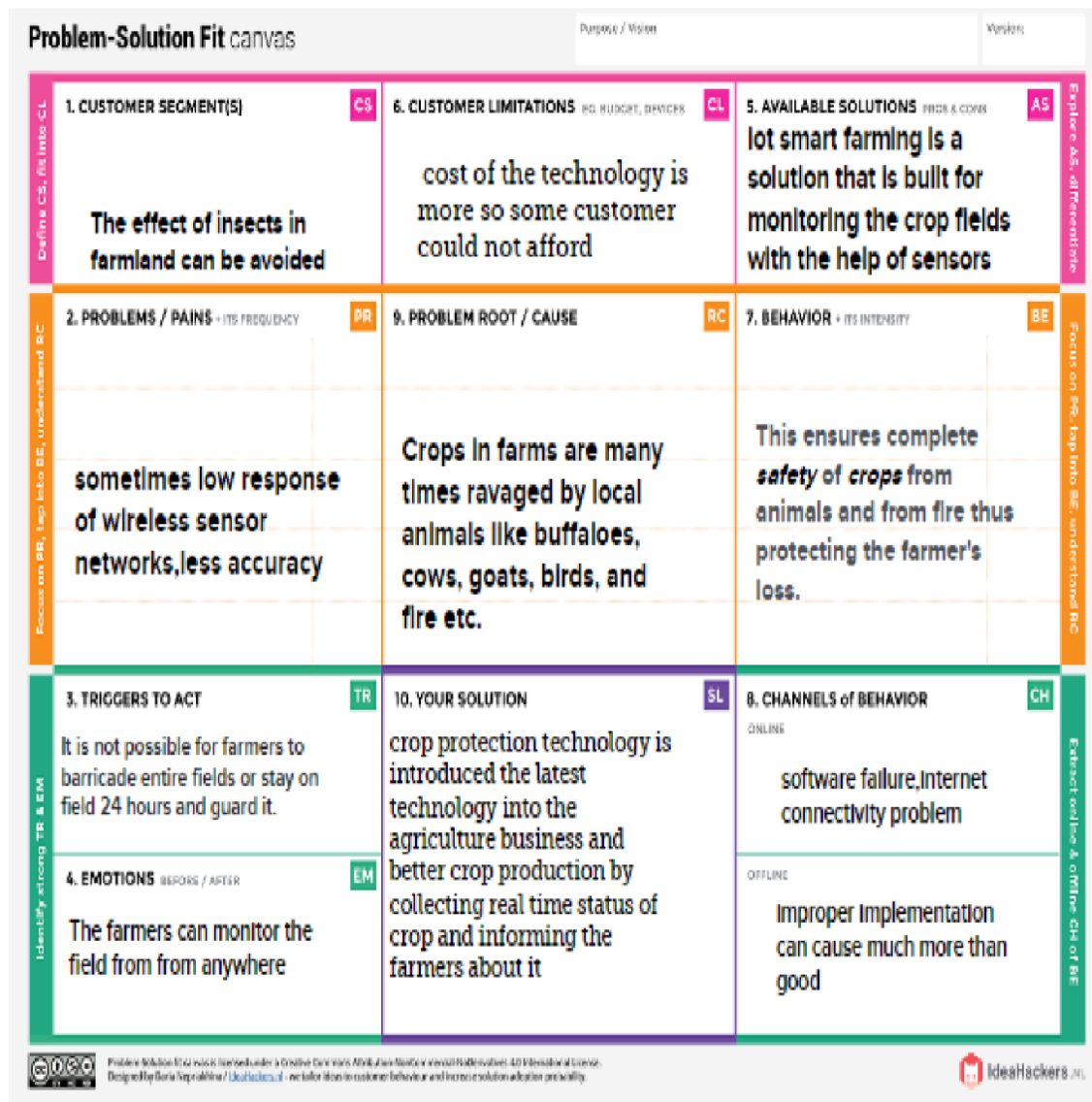
Step-3: Idea Prioritization



3.3 Proposed Solution



3.4 Problem Solution Fit



REQUIREMENT ANALYSIS

4.1 Functional Requirements

FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Visibility	Sensing animals nearing the crop field and sounds alarm to woo them away as well as sends SMS to farmer using cloud

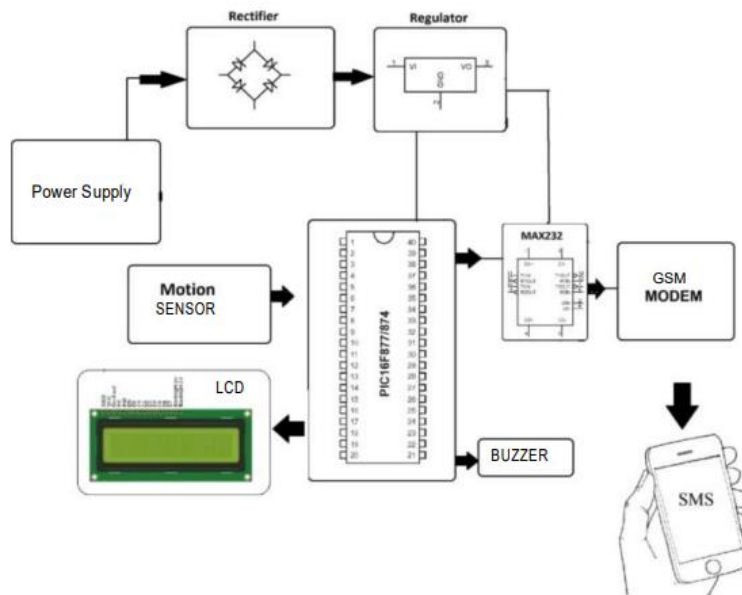
4.2 Non-functional Requirements

FR No	Non-Functional Requirement	Description
NFR-1	Usability	Mobile support. Users must be able to interact in the same roles & tasks on computers & mobile devices where practical, given mobile capabilities.
NFR-2	Security	Data requires secure access to must register and communicate securely on devices and authorized users of the system who exchange information must be able to do.
NFR-3	Reliability	It has a capacity to recognize the disturbance near the field and doesn't give a false caution signal.
NFR-4	Performance	Must provide acceptable response times to users regardless of the volume of data that is stored and the analytics that occurs in background. Bidirectional, near real-time communications must be supported. This requirement is related to the requirement to support industrial and device protocols at the edge.
NFR-5	Availability	IoT solutions and domains demand highly available systems for 24x7 operations. Isn't a <i>critical production</i> application, which means that operations or production don't go down if the IoT solution is down.
NFR-6	Scalability	System must handle expanding load and data retention needs that are based on the upscaling of the solution scope, such as extra manufacturing facilities and extra buildings.

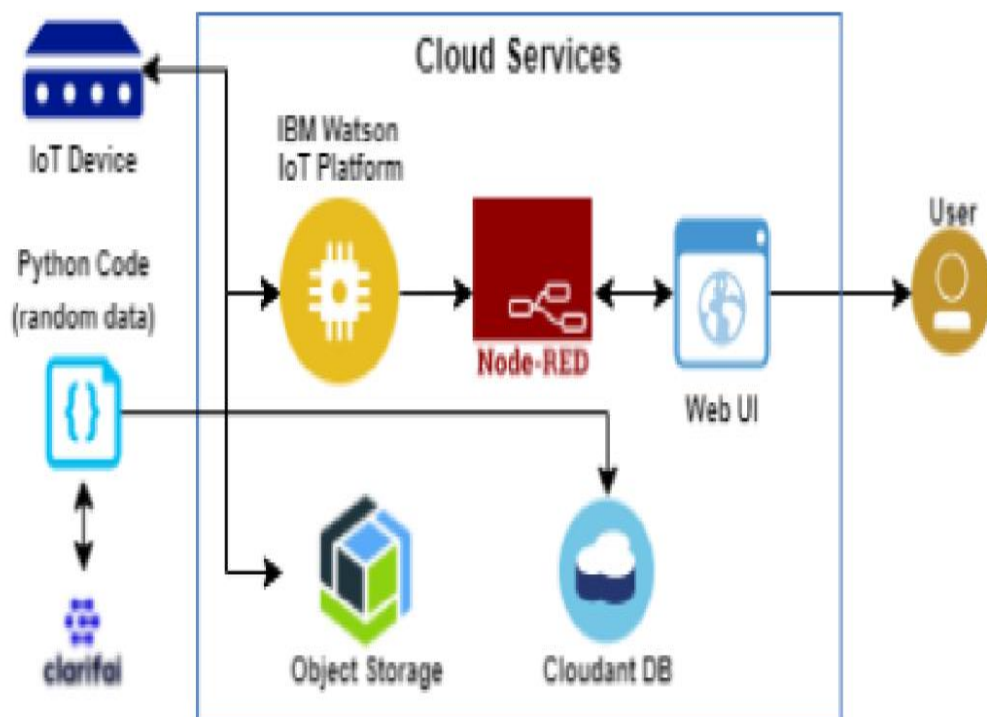
PROJECT DESIGN

5.1 Data Flow Diagrams

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



5.2 Technical Architecture:



5.3 USER STORIES

USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story/Task	Acceptance criteria	priority
Customer (Mobile user)	Download the database	USN-1	As a user I can register for the application by entering my email, password and confirming my password.	I can access my account/ dashboard	High
	Register	USN-2	As a user I can register for the application by entering my email, password and confirming my password.	I can receive confirmation email and click confirm	High
	Login	USN-3	As a user I will receive confirmation email once I have registered for the application.	I can register and access the dashboard with Facebook login	Low
	Upload the image	USN-4	As a user I must upload the image to identify the problem and works on it.		Medium
Customer (Web user)	The functional requirements are same as mobile user	Same as mobile user	Same as mobile user.	Same as mobile user	High when compare to mobile user

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story/Task	Story points	priority	Team members
Sprint -1	Download the database	USN-1	As a user I can register for the application by entering my email, password and confirming my password.	2	High	Praveena
Sprint -1	Register	USN-2	As a user I can register for the application by entering my email, password and confirming my password.	2	High	Sheeja
Sprint -2	Login	USN-3	As a user I will receive confirmation email once I have registered for the application.	1	Low	Sreeja
Sprint -1	Upload the image	USN-4	As a user I must upload the image to identify the problem and works on it.	1	Medium	Subithra

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart:

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

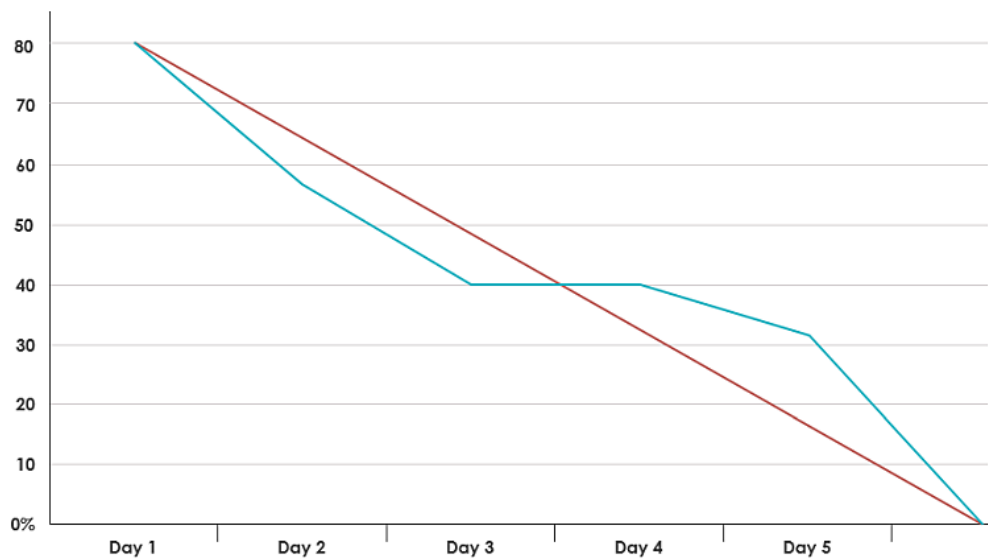
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	30	30 oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	49	6 nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	50	7 nov 2022

Velocity:

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

6.3 Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



CODING & SOLUTIONING

CODING

```
import random
import time
import sys

#IBM Watson Device Credentials.
organization = "zf801i"
deviceType = "bharathi"
deviceId = "bharathi123"
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 and 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, sprinklers are  
turned ON" %temp_sensor
```

```
    }, qos=0)
```

```
    sleep(1) if
```

```
    success:
```

```
        print(
```

```
        'Published
```

```
        alert1 : ',
```

```
        "Temperature(
```

```
        %s) is high,
```

```
        sprinklers
```

```
        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()

```


7.2 RESULT

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\my pc\Documents\nalalyathiran ibm\project development phase\sprint 1\Python script (IOT based smart crop protection s
ystem for agriculture).py
2022-10-30 15:23:08.539  ibmiotf.device.Client  INFO  Connected successfully: d:zf801i:bharathi:bharathi123
.....publish ok.....
Published Temperature = 41.7 C to IBM Watson
Published PH Level = 11.955 to IBM Watson
Published Animal attack Not Detected to IBM Watson
Published Flame Not Detected to IBM Watson
Published Moisture Level = 49.71 to IBM Watson
Published Water Level = 15.01 cm to IBM Watson

sprinkler-1 is ON
Published alert1 : Temperature(41.7) is high, sprinklerlers are turned ON to IBM Watson

Published alert2 : Fertilizer PH level(11.955) is not safe,use other fertilizer to IBM Watson

sprinkler-2 is OFF
Motor-1 is OFF
Motor-2 of OFF

.....publish ok.....
Published Temperature = 24.92 C to IBM Watson
Published PH Level = 3.948 to IBM Watson
Published Animal attack Detected to IBM Watson
Published Flame Not Detected to IBM Watson
Published Moisture Level = 65.01 to IBM Watson
Published Water Level = 11.14 cm to IBM Watson

sprinkler-1 is OFF
Published alert2 : Fertilizer PH level(3.948) is not safe,use other fertilizer to IBM Watson
Published alert3 : Animal attack on crops detected to IBM Watson to IBM Watson

sprinkler-2 is OFF
Motor-1 is OFF
Motor-2 of OFF
.....publish ok.....
```

8 APPENDIX

8.1 SOURCE CODE

```
import cv2

import numpy as np

import wiotsdk.device

import playsound

import random import
```

```

time import datetime

import ibm_boto3

from ibm_botocore.client import Config, ClientError

#CloudantDB

from cloudant.client import Cloudant

from cloudant.error import CloudantException from

cloudant.result import Result, ResultByKey from

clarifai_grpc.channel.clarifai_channel import ClarifaiChannel from

clarifai_grpc.grpc.api import service_pb2_grpc stub =

service_pb2_grpc.V2Stub(clarifaiChannel.get.grpc_channel()) from

clarifai_grpc.grpc.api import service_pb2, resource_pb2 from

clarifai_grpc.grpc.api.status import status_code_pb2

#This is how you authenticate

metadata = (('authorization', 'key 0620e202302b4508b90eab7efe7475e4'),)

COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

COS_API_KEY_ID = "g5d4q08Elgv4TWUCJj4hfEzgalqEjrDbE82AJDWIA0Ho"

COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"

COS_RESOURCE_CRN = "crn:v1:bluemix:public:cloud-

objectstorage:global:a/c2fa2836eaf3434bbc8b5b58fefff3f0:62e450fd-4c82-4153-ba41-

ccb53adb8111::"

clientdb = cloudant("apikey-W2njldnwtjO16V53LAVUCqPwc2aHTLmlj1xXvtdGKJBn",

```

```

"88cc5f47c1a28afbfb8ad16161583f5a", url="https://d6c89f97-cf91-48b7-b14b-
c99b2fe27c2fbluemix.cloudantnosqldb.appdomain.cloud")

clientdb.connect()

#Create resource

cos = ibm_boto3.resource("s3",

ibm_api_key_id=COS_API_KEY_ID,

ibm_service_instance_id=COS_RESOURCE_CRN,

ibm_auth_endpoint=COS_AUTH_ENDPOINT,

config=Config(signature_version="oauth"),

endpoint_url=COS_ENDPOINT

)

def = multi_part_upload(bucket_name, item_name, file_path):

try:

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

#set 5 MB chunks part_size =

1024 * 1024 * 5 #set threshold

to 15 MB file_threshold = 1024 *

1024 * 15 #set the transfer

threshold and chunk size

transfer_config =

ibm_boto3.s3.transfer.TransferC

onfig(

```

```

multipart_threshold=file_thresh
old,
multipart_chunksize=part_size
)

#the upload_fileobj method will automatically execute a multi-part upload
#in 5 MB chunks size with
open(file_path, "rb") as file_data:
cos.Object(bucket_name, item_name).upload_fileobj(
Fileobj=file_data,
Config=transfer_config
)

print("Transfer for {0} Complete!\n".format(item_name))
except ClientError as be: print("CLIENT ERROR:
{0}\n".format(be)) except Exception as e:
print("Unable to complete multi-part upload: {0}".format(e))

def myCommandCallback(cmd):
print("Command received: %s" % cmd.data)
command=cmd.data['command']
print(command) if(command=="lighton"):
print('lighton') elif(command=="lightoff"):
print('lightoff')
elif(command=="motoron"):

```

```

print('motoron')

elif(command=="motoroff"):

print('motoroff')

myConfig = {

"identity": {

"orgId": "chytun",

"typeId": "NodeMCU",

"deviceId": "12345"

},

"auth": {

"token": "12345678"

}

}

client = wiot.sdk.device.DeviceClient(config=myConfig, logHandlers=None) client.connect()

database_name = "sample"

my_database = clientdb.create_database(database_name) if

my_dtabase.exists():

print(f'"{database_name}" successfully created.')

cap=cv2.VideoCapture("garden.mp4")

if(cap.isOpened()==True): print('File opened')

else:

print('File not found')

```

```

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

    gray = cv3.cvtColor(frame, cv2.COLOR_BGR@GRAY)

    imS= cv2.resize(frame, (960,540))

    cv2.imwrite('ex.jpg',imS) with open("ex.jpg", "rb")
    as f:

    file_bytes = f.read()

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

    request = service_pb2.PostModeloutputsRequest(
        model_id='e9359dbe6ee44dbc8842ebe97247b201',
        inputs=[resources_pb2.Input(data=resources_pb2.Data(image=resources_pb2.Image(base64=
        file_bytes
        ))
        ))

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

    if response.status.code != status_code_pb2.SUCCESS:
        raise Exception("Request failed, status code: " + str(response.status.code))

    detect=False for concept in response.outputs[0].data.concepts: #print('%12s:
    %.f' % (concept.name, concept.value)) if(concept.value>0.98):

    #print(concept.name) if(concept.name=="animal"):

```

```

print("Alert! Alert! animal detected") playsound.playsound('alert.mp3')

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

cv2.imwrite(picname+'.jpg',frame) multi_part_upload('Dhakshesh',
picname+'.jpg', picname+'.jpg')

json_document={"link":COS_ENDPOINT+'/'+'+Dhakshesh'++'/'+picname+'.jpg'}
new_document = my_database.create_document(json_document)

if new_document.exists():

print(f"Document successfully created.")

time.sleep(5) detect=True moist=random.randint(0,100)

humidity=random.randint(0,100)

myData={'Animal':detect,'moisture':moist,'humidity':humidity}
print(myData)

if(humidity!=None):

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

print("Publish Ok..") client.commandCallback = myCommandCallback cv2.imshow('frame',imS)

if

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

break

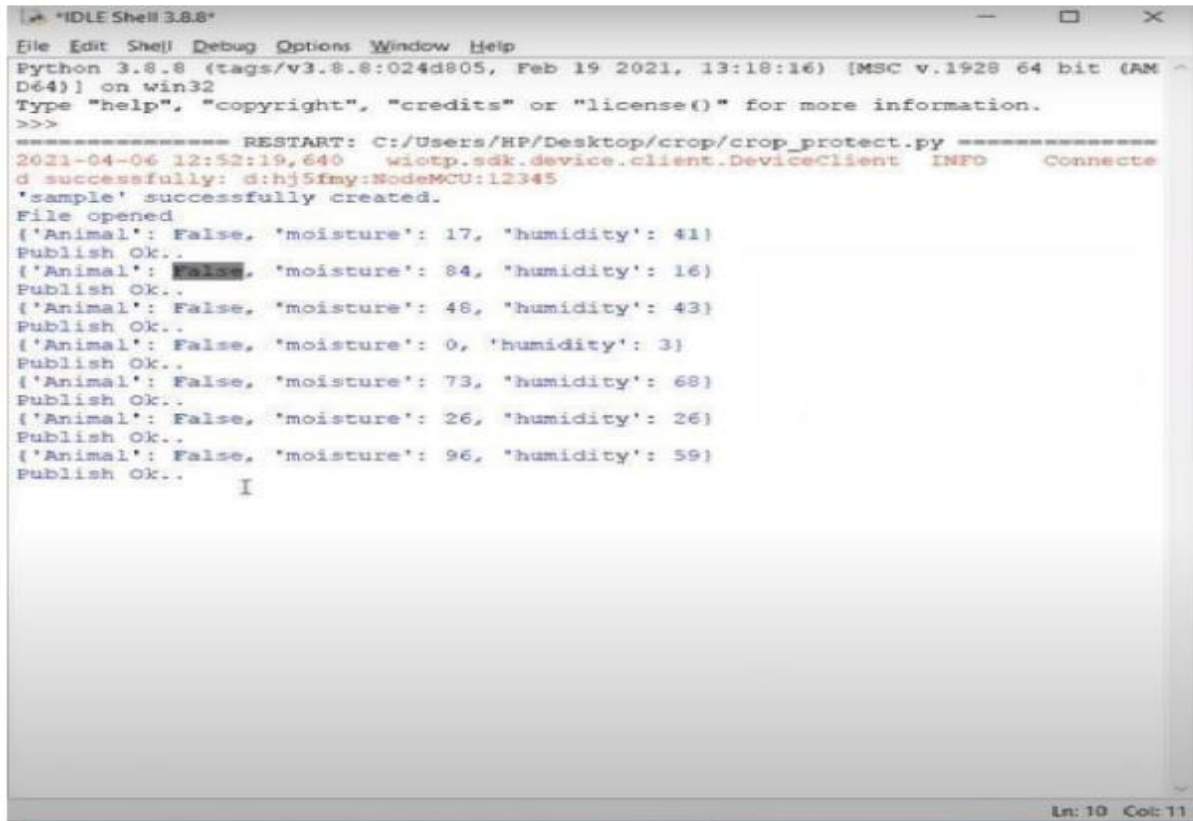
client.disconnect()

cap.release()

cv2.destroyAllWindows()

```

8.2 RESULT



```
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/Desktop/crop/crop_protect.py =====
2021-04-06 12:52:19,640 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:hj5fmy:NodeMCU:12345
'sample' successfully created.
File opened
{'Animal': False, 'moisture': 17, 'humidity': 41}
Publish OK..
{'Animal': False, 'moisture': 84, 'humidity': 16}
Publish OK..
{'Animal': False, 'moisture': 48, 'humidity': 43}
Publish OK..
{'Animal': False, 'moisture': 0, 'humidity': 3}
Publish OK..
{'Animal': False, 'moisture': 73, 'humidity': 68}
Publish OK..
{'Animal': False, 'moisture': 26, 'humidity': 26}
Publish OK..
{'Animal': False, 'moisture': 96, 'humidity': 59}
Publish OK..
I
```

9 ADVANTAGES & DISADVANTAGES

Advantages:

IoT crop monitoring systems help maintain optimal conditions to provide adequate crop quality. IoT-based weather monitoring systems in farming help calculate the required supply of chemicals, nutrients, and water to produce high-quality crop yields.

Disadvantages:

There could be wrong Analysis of Weather Conditions

There is no force which can change or control the weather conditions such as rain, sunlight, drought etc. Even when the smart systems are in place, the importance of natural occurrences can not be changed. The cost of maintenance becomes high whether there is a repair or not.

10 CONCLUSION

Internet of Things has enabled the agriculture crop monitoring easy and efficient to enhance the productivity of the crop and hence profits for the farmer. Wireless sensor network and sensors of different types are used to collect the information of crop conditions and environmental changes and this information is transmitted through network to the farmer/devices that initiates corrective actions. Farmers are connected and aware of the conditions of the agricultural field at anytime and anywhere in the world. Some disadvantages in communication must be overcome by advancing the technology to consume less energy and also by making user interface ease of use.

11 FUTURE SCOPE

The system can be enhanced further to add following functionality: Use of soil moisture sensors, environment sensors, pH sensors to increase the accuracy while predicting the crop. Locations market requirements can be considered, and neighbor farmers crop while suggesting the suitable crop.

12 GitHub & Project Demo Link:

GitHub Link:

[IBM-EPBL/IBM-Project-40270-1660626928: IoT Based Smart Crop Protection System for Agriculture \(github.com\)](https://github.com/IBM-EPBL/IBM-Project-40270-1660626928)

Project demo link:

https://drive.google.com/file/d/1lxJUgZK6_-7zjF3wTozjTimuK5Lpx6SC/view

