Project Report

# 1. INTRODUCTION

## 1.1 Project Overview

The title of our project is **"IOT based Smart Crop Protection"**. The Overview of our Project is to Safeguard the farm from climatic changes like soil erosion, landslide, and birds, animals etc,.So, that we are making a IOT based project to protect the farm from climatic changes and haunting animals. For that we are just making a cloud based project and placing IOT based sensor. over which it will produce sounds and notification and provide results on IOT- MIT app. From which we can protect our farm. And it will provide better yield for us.

## 1.2 Purpose

- ⇨ The main purpose of our project is to protect the farm from climatic changes animals, birds , pests.
- ⇨ And to make the crop to grow better and provide better yield.

# 2. LITERATURE SURVEY

## 2.1 Existing problem

- ⇨ "Food" is the important thing, which is needed for everyone to survive in this world. For that farmers are doing their own part in a effective manner, during which they have to face some problems such as:
- ⇨ There are increasing pressures from climate change, soil erosion and biodiversity loss and from consumers' changing tastes in food and concerns about how it is produced.
- ⇨ And the natural world that farming works with – plants, pests and diseases – continue to pose their own challenges beyond that, they have to
- ⇨ Stay resilient against global economic factors.
- ⇨ Inspire young people to stay in rural areas and become future farmers
- ⇨ The effects of climate change affect farmers' ability to grow the food we all need. Increasingly volatile weather and more extreme events – like floods and droughts – change growing seasons, limit the availability of water, allow weeds, pests and fungi to thrive, and can reduce crop productivity.

## 2.2 References

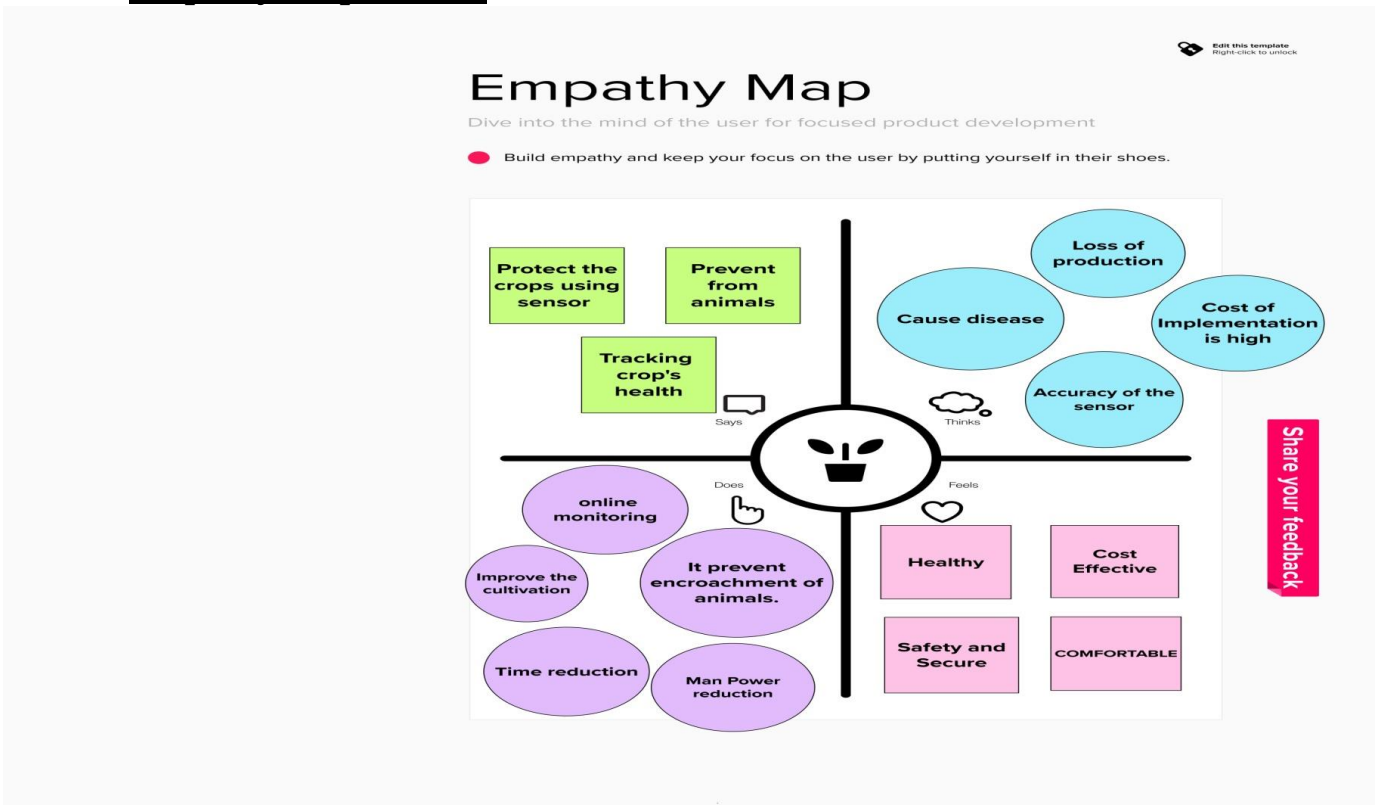

- https://smartinternz.com/assets/docs/Smart%20Home%20Automation%20using%20IBM%20cloud%20Service s%20(1).pdf
- https://smartinternz.com/assets/docs/Smart%20Home%20Automation%20using%20IBM%20cloud%20Service s%20(1).pdf
- https://openweathermap.org/
- https://smartinternz.com/assets/docs/Sending%20Http%20request%20to%20Open%20weather%20map%20web site%20to%20get%20the%20weather%20forecast.pdf
- https://www.youtube.com/watch?v=cicTw4SEdxk
- https://smartinternz.com/assets/docs/Smart%20Home%20Automation%20using%20IBM%20cloud%20Service s%20(1).pdf
- https://github.com/rachuriharish23/ibmsubscribe


## 2.3 Problem Statement Definition

- Agriculture is one of the Area which required urgent attention and advancement for high yield and efficient utilization of resources.
- In this paper an approach of smart crop monitoring is presented through Internet of things (IOT).
- A 4 Level framework is proposed namely sensing devices, sensor data level, base station level, edge computing and cloud data level for smart crop monitoring.
- In this project, farm is going to get protected from humidity, temperature, and animals. With the help of IOT cloud module.
- The agricultural form is been monitored with the help of MIT app and then, the data will be collected and stored it in cloud.
- It will monitor and sense the humidity level and movement of animals and will send the message as notification to the user.

# 3. IDEATION & PROPOSED SOLUTION

## 3.1 Empathy Map Canvas




## 3.2 Ideation & Brainstorming

### What do they think and feel?

As its name may imply, smart farming is the use of technology in animal agriculture, and it's something that's been around since the Industrial Revolution. The biggest difference between then and now, though? "Motorized devices are being replaced with IOT".

### What do they hear?

Smart farming is about using the new technologies which have arisen at the dawn of the Fourth Industrial Revolution in the areas of agriculture and cattle production to increase production quantity and quality, by making maximum use of resources and minimizing the environmental impact.

## What do they see?

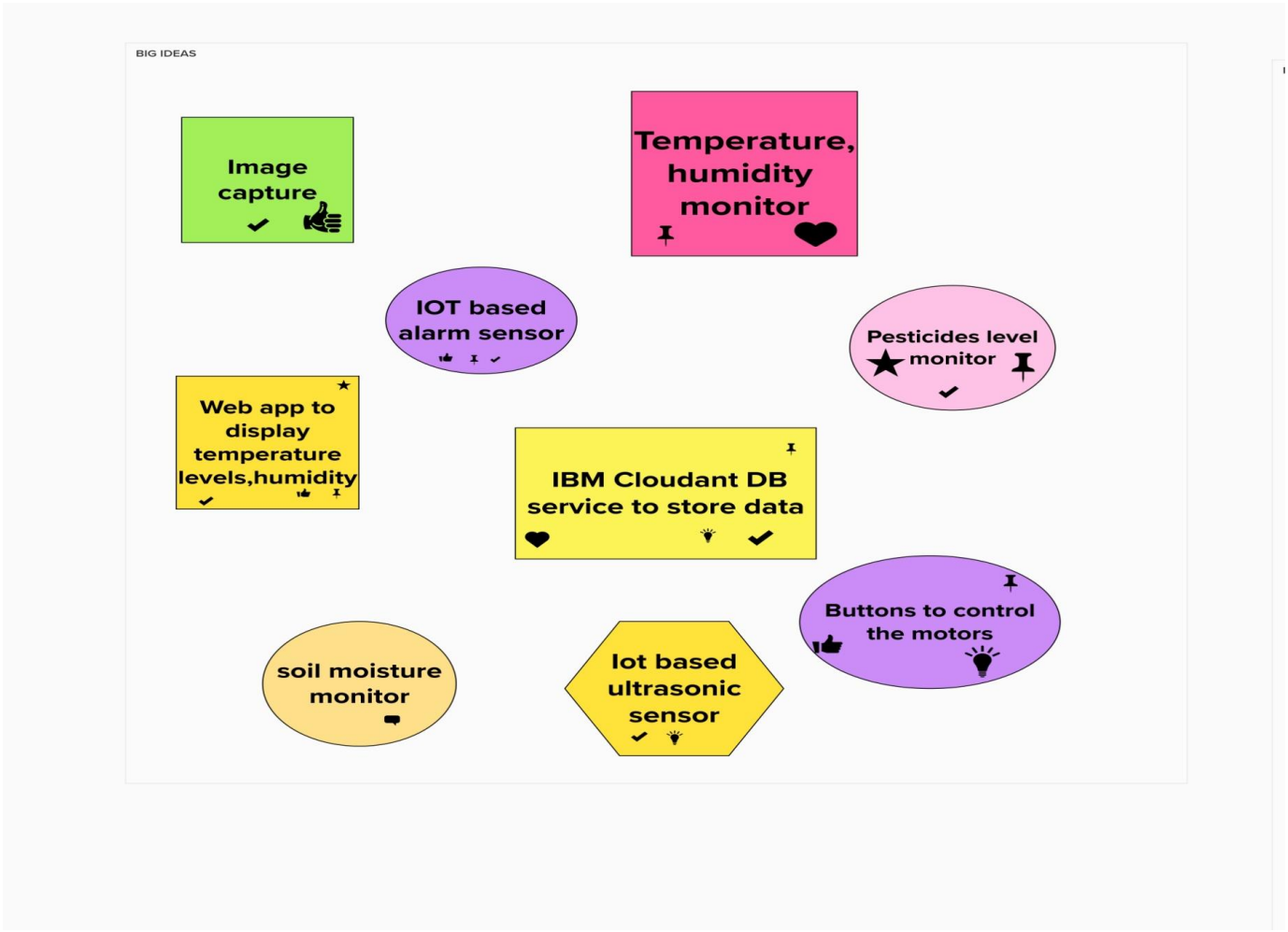
Smart farming is a management concept focused on providing the agricultural industry with the infrastructure to leverage advanced technology – including big data, the cloud and the internet of things (IoT) – for tracking, monitoring, automating and analyzing operations.

## What do they say and do?

- The aim of this technology is to **make the most of all the data** collected by various tools, by converting them into real **sources of information** in order to then define ways of simplifying agricultural work. It also allows for **accurate and predictive analysis** of all situations that may affect the farms, such as weather conditions (temperature, humidity, etc.) and sanitary or economic situations, for example. This makes it easier to organize the supply of energy, water, livestock feed and fertilizer.
- In its most advanced form, smart farming facilitates the exchange of information between different farms, creating a real network of connected farms accessible from a smartphone or a computer.

## BRAINSTORM:

## BRAINSTORM




## PRIORITIZATON:

PRIORITIZE


## 3.3 Proposed Solution

**Project Design Phase-I**
**Proposed Solution Template**

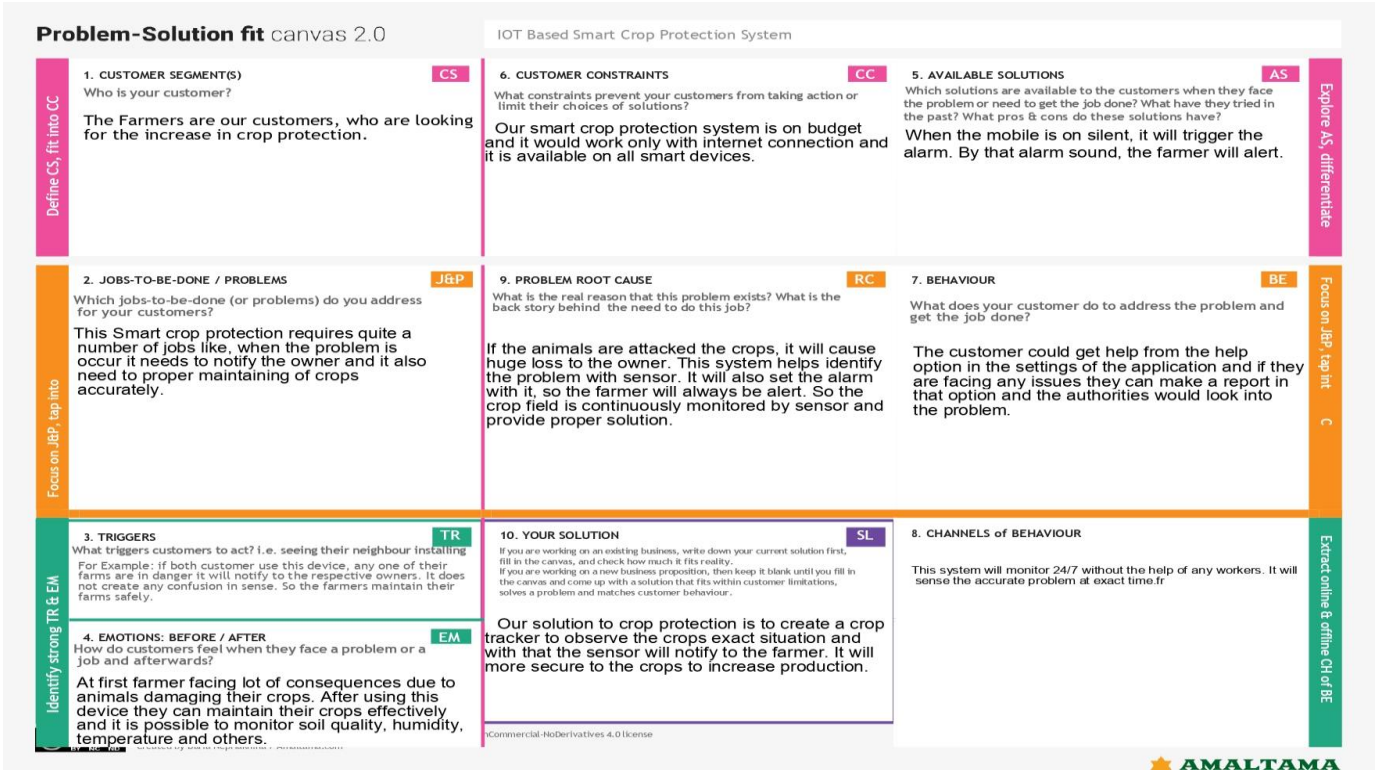
| | |
|---|---|
| Date | 19 September 2022 |
| Team ID | PNT2022TMID10974 |
| Project Name | Project – IoT Based Smart Crop Protection System |
| Maximum Marks | 2 Marks |

**Proposed Solution Template:**

Project team shall fill the following information in proposed solution template.

| 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, and fire, etc... This leads to huge losses for the farmers. So, here we propose automatic crop protection system from animals and fire. |
| 2. | Idea / Solution description | This System which assists the farmer in the irrigation process. We use IR Sensors to detect the Wild animals, Soil moisture and connected to Arduino microcontrollers. |
| 3. | Novelty / Uniqueness | The smart farm protection system gives reliable security and safety to crops by monitoring 24/7. |
| 4. | Social Impact / Customer Satisfaction | It increases the crop productivity and soil fertility. Peoples are live a healthy life. |
| 5. | Business Model (Revenue Model) | It gives more protection and increase productivity of the crops at low cost. |
| 6. | Scalability of the Solution | It increases the Protection of the crops in the field by indicating the Farmer through the Sensors. |

## 3.3 Problem Solution fit




# 4 REQUIREMENT ANALYSIS

## 3.4 Functional requirement

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | **User Requirements** | Crop Protection<br>Automatic Sprinkler System<br>Monitors Soil Moisture ,Humidity and Temperature |
| FR-2 | **User Registration** | Manual Registration<br>Registration through webpage<br>Registration through Form<br>Registration through Gmail |
| FR-3 | **User Confirmation** | Confirmation via Phone<br>Confirmation via Email<br>Confirmation via OTP |
| FR-4 | **Payment Options** | Cash on Delivery<br>Net Banking/UPI<br>Credit/Debit/ATM Card |
| FR-5 | **Product Delivery and Installation** | Door Step delivery<br>Take away<br>Free Installation and 1 year Warranty |
| FR-6 | **Product Feedback** | Through Webpage<br>Through Phone calls<br>Through Google forms |

## 3.5 Non-Functional requirements

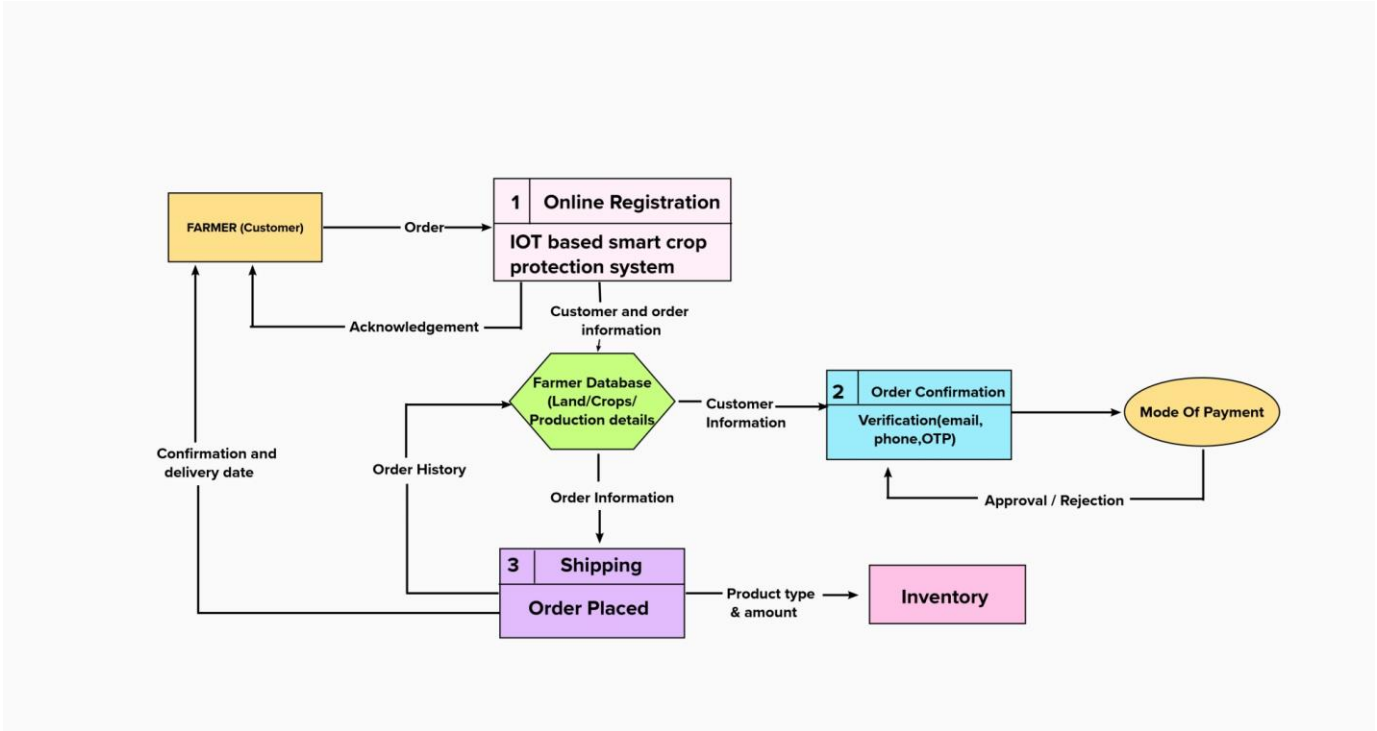
**Non-functional Requirements:**

Following are the non-functional requirements of the proposed solution.

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | **Usability** | Have a clear and self-explanatory manual.<br>Easier to use<br>Even an illiterate farmer have to use the product without any difficulties |
| NFR-2 | **Security** | Application has to be secured with 2 step authorisation<br>Passwords and passkeys will be assigned as per the users need. |
| NFR-3 | **Reliability** | Hardware requires a regular checking and service<br>Software may be updated periodically<br>Immediate alert is provided in case of any system failure |
| NFR-4 | **Performance** | The application must have a good user interface It should have a minimal energy requirement It has to save water and energy |
| NFR-5 | **Availability** | All the features will be available when the user requires. It depends on the need of the farmer and the customization the user has done. |
| NFR-6 | **Scalability** | The product has to cover all the space of land irrespective of the size or area of a farm field. |

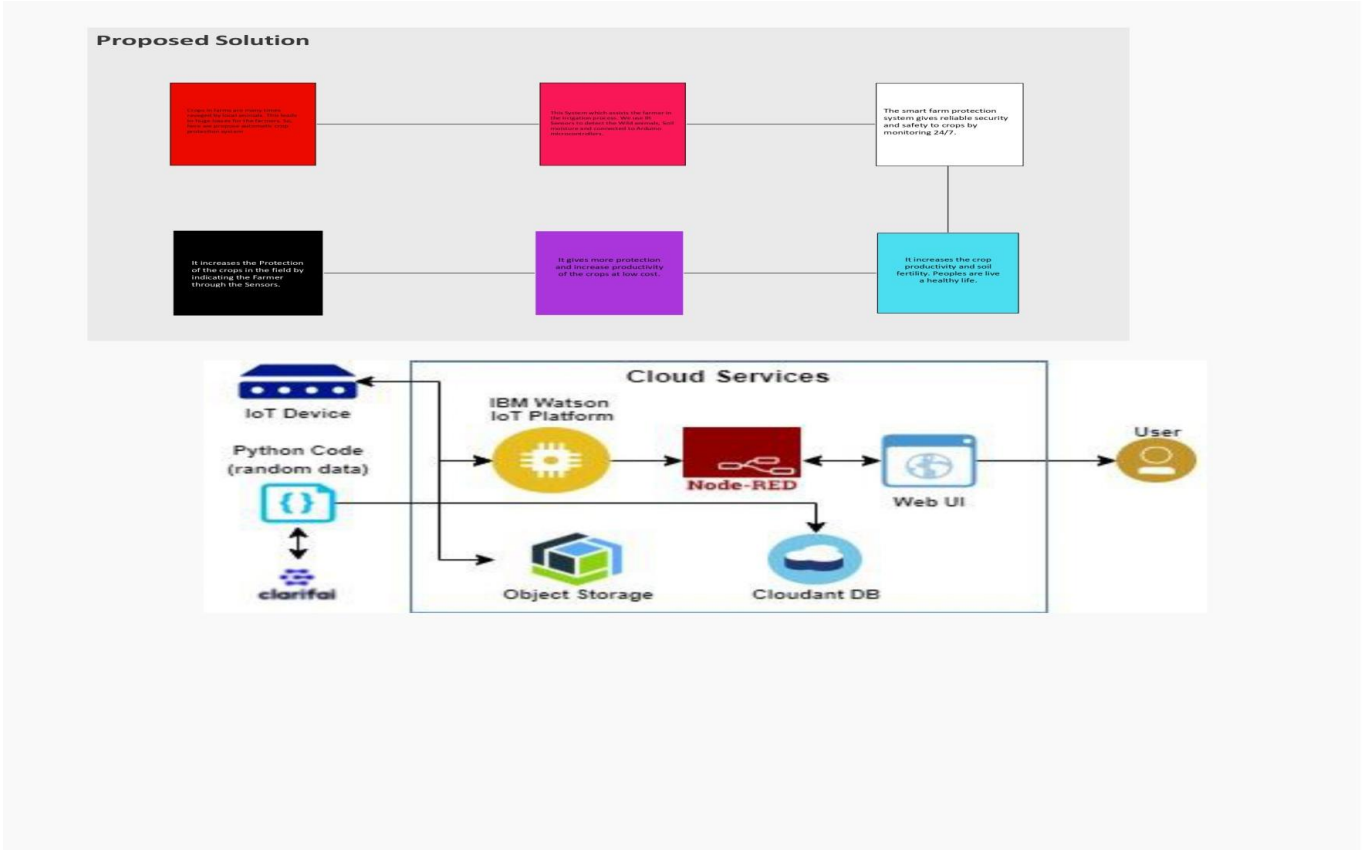
# 4. PROJECT DESIGN

## 4.1 Data Flow Diagrams



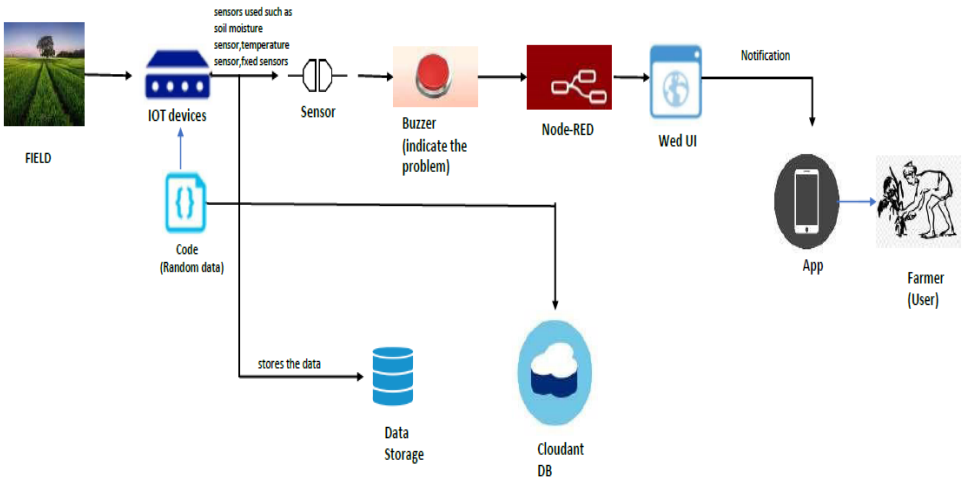
## 4.2 Solution & Technical Architecture

### SOLUTION ARCHITECTURE:




### TECHNICAL ARCHITECTURE:

Solution Architecture Diagram:




## 4.3 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 | Same as mobile user | Same as mobile user. | Same as mobile user | High when compare |

# 5. PROJECT PLANNING & SCHEDULING

**Project Planning Phase**
**Project Planning Template**

| | |
|---|---|
| Date | 5 November 2022 |
| Team ID | PNT2022TMID10974 |
| Project Name | Project –IoT Based Smart Crop Protection For Agriculture |
| Maximum Marks | 8 Marks |

**Product Backlog, Sprint Schedule, and Estimation (4 Marks)**
Use the below template to create product backlog and sprint schedule

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | IBM Cloud Services | USN-1 | Create a Cloud Account in IBM | 10 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-1 | Software | USN-2 | Install the Python IDE | 5 | Medium | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-1 | Clarifai | USN-3 | Create an Account in Clarifai (To detect the animals and birds we are using an open-source platform Clarifai.) | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-2 | IBM Watson Platform | USN-4 | Create IBM Watson IoT Platform and Device (It acts as the mediator to connect the web application to IoT device) | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-2 | Node Red Services | USN-5 | Create Node Red Services (To Create a Web Application) | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |

| | | | | | | |
|---|---|---|---|---|---|---|
| Sprint-2 | Cloudant DB | USN-6 | Create a Database in Cloudant DB (To Store the Image URL, Launch the Cloudant DB) | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-3 | Cloud Object Storage | USN-7 | Create a Cloud Object Storage Service | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-4 | Python Code | USN-8 | Develop a Python Script | 10 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |

**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 | 15 | 7 Days | 24 Oct 2022 | 31 Oct 2022 | 15 | 31 Oct 2022 |
| Sprint-2 | 15 | 7 Days | 01 Nov 2022 | 07 Nov 2022 | 15 | 07 Nov 2022 |
| Sprint-3 | 10 | 6 Days | 08 Nov 2022 | 13 Nov 2022 | 10 | 13 Nov 2022 |
| Sprint-4 | 10 | 6 Days | 14 Nov 2022 | 20 Nov 2022 | 10 | 20 Nov 2022 |

**6.**

## 6.1 Sprint Planning & Estimation

**Product Backlog, Sprint Schedule, and Estimation (4 Marks)**
Use the below template to create product backlog and sprint schedule

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | IBM Cloud Services | USN-1 | Create a Cloud Account in IBM | 10 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-1 | Software | USN-2 | Install the Python IDE | 5 | Medium | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-1 | Clarifai | USN-3 | Create an Account in Clarifai (To detect the animals and birds we are using an open-source platform Clarifai.) | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-2 | IBM Watson Platform | USN-4 | Create IBM Watson IoT Platform and Device (It acts as the mediator to connect the web application to IoT device) | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-2 | Node Red Services | USN-5 | Create Node Red Services (To Create a Web Application) | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-2 | Cloudant DB | USN-6 | Create a Database in Cloudant DB (To Store the Image URL, Launch the Cloudant DB) | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-3 | Cloud Object Storage | USN-7 | Create a Cloud Object Storage Service | 5 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |
| Sprint-4 | Python Code | USN-8 | Develop a Python Script | 10 | High | Bhrintha P<br>Divya A<br>Janavigha P<br>Jayasurya S |

## 6.2 Sprint Delivery Schedule

**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 | 15 | 7 Days | 24 Oct 2022 | 31 Oct 2022 | 15 | 31 Oct 2022 |
| Sprint-2 | 15 | 7 Days | 01 Nov 2022 | 07 Nov 2022 | 15 | 07 Nov 2022 |
| Sprint-3 | 10 | 6 Days | 08 Nov 2022 | 13 Nov 2022 | 10 | 13 Nov 2022 |
| Sprint-4 | 10 | 6 Days | 14 Nov 2022 | 20 Nov 2022 | 10 | 20 Nov 2022 |

**Velocity:**

Sprint-1 and Sprint-2

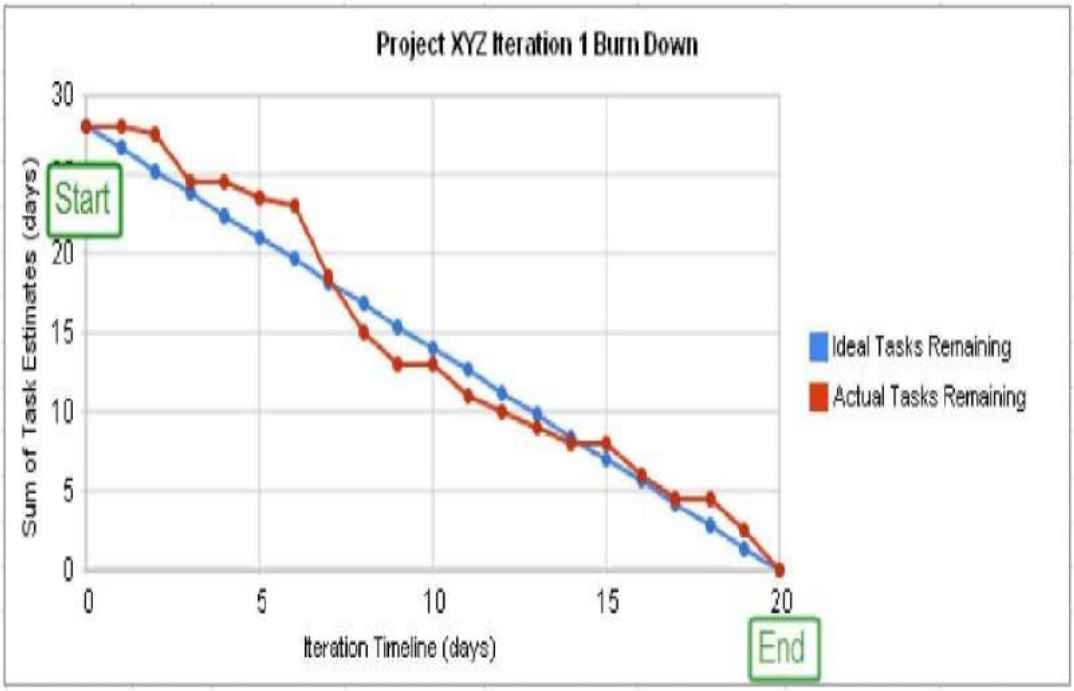
AV = Sprint Duration / Velocity= 15/7=2.14

Sprint-3 and Sprint-4

AV=Sprint Duration / Velocity = 10/6 = 1.6

## BURNDOWN CHART:

# Burndown Chart



# 7. CODING & SOLUTIONING

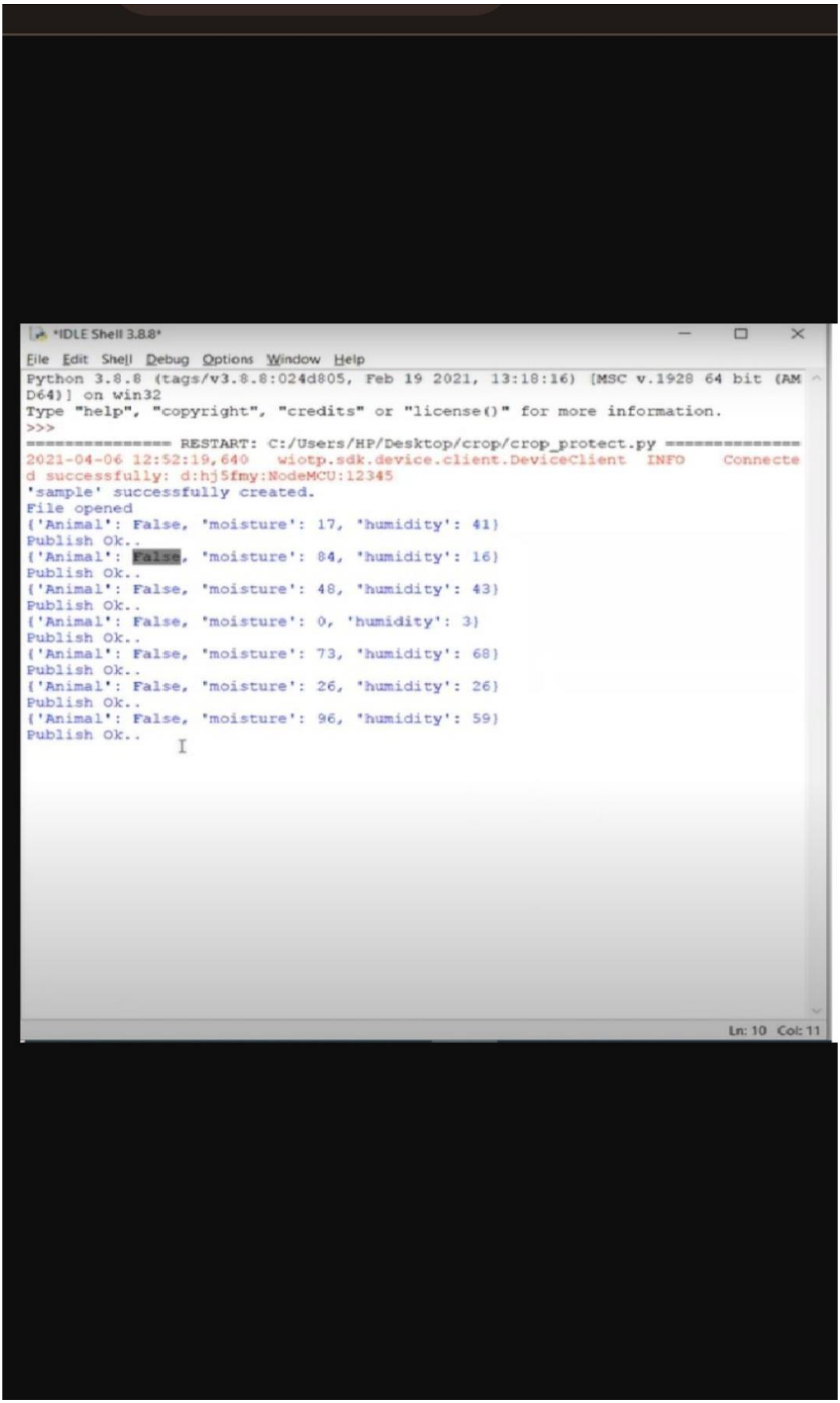
## 7.1 Feature 1

```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
import cv2
import numpy as np
import wiot.sdk.device
import playsound
import random
import time
import datetime
import ibm_boto3
from ibm_botocore.client import Config, ClientError

#CloudantDB
from cloudant.client import Cloudant
from cloudant.error import CloudantException
from cloudant.result import Result, ResultByKey
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import service_pb2_grpc
stub = service_pb2_grpc.V2Stub(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 = "g5d4qO8EIgv4TWUCJj4hfEzgalqEjrDbE82AJDW1AOHo"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/c2fa2836eaf3434bbc8b5b58fefff3f0:62e450fd-4c82-4153-ba41-ccb53adb8111::"
clientdb = cloudant("apikey-W2njldnwtjO16V53LAVUCqPwc2aHTLm1j1xXvtdGKJBn", "88cc5f47c1a28afbfb8ad16161583f5a", url="https://d6c89f97-cf91-48b7-b14b-c99b2fe27c2f-bluemix.clouda
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"),
```

```
                        config=Config(signature_version="oauth"),
                        endpoint_url=COS_ENDPOINT
                        )
def = multi_part_upload(bucket_name, item_name, file_path):
    try:
        print("Starting file transfer for {0} to bucket: {1}\n".format(item_name, bucket_name))
        #set 5 MB chunks
        part_size = 1024 * 1024 * 5
        #set threadhold to 15 MB
        file_threshold = 1024 * 1024 * 15
        #set the transfer threshold and chunk size
        transfer_config = ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,
            multipart_chunksize=part_size
            )
        #the upload_fileobj method will automatically execute a multi-part upload
        #in 5 MB chunks size
        with open(file_path, "rb") as file_data:
            cos.Object(bucket_name, item_name).upload_fileobj(
                Fileobj=file_data,
                Config=transfer_config
                )
        print("Transfer for {0} Complete!\n".format(item_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to complete multi-part upload: {0}".format(e))

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

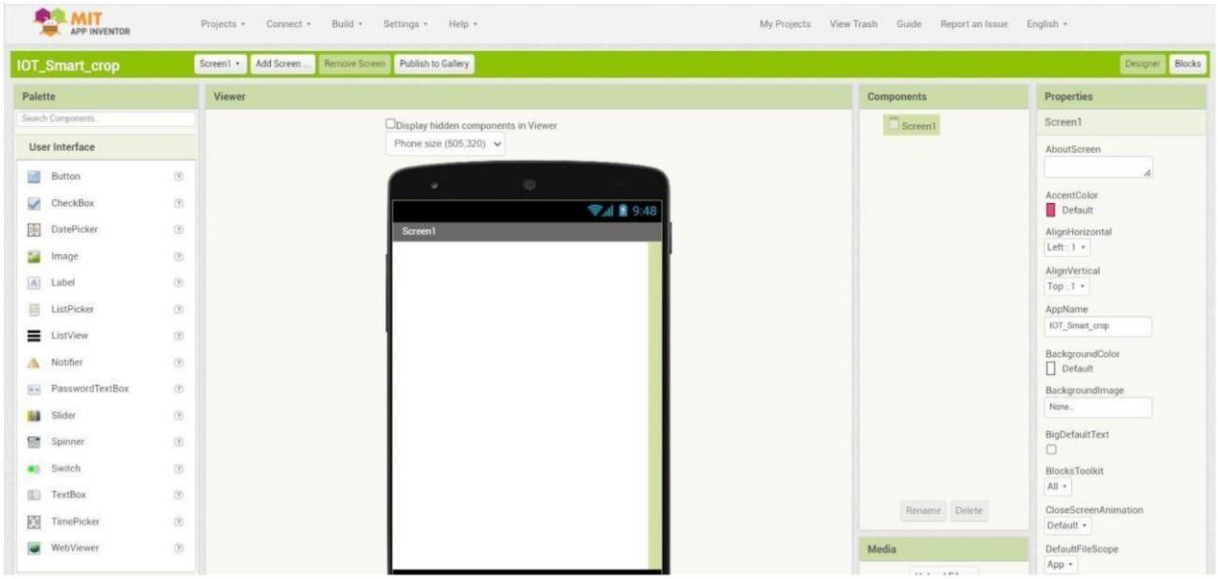
```
        print('motoron')
    elif(command=="motoroff"):
        print('motoroff')
myConfig = {
    "identity": {
        "orgId": "chytun",
        "typeId": "NodeMCU",
        "deviceId": "12345"
        },
    "auth": {
        "token": "12345678"
        }
    }
client = wiot.sdk.device.DeviceClient(config=myConfig, 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.inwrite('ex.jpg',imS)
    with open("ex.jpg", "rb") as f:
        file_bytes = f.read()
    #This is the model ID of a publicly available General model. You may use any other public or custom model ID.
    request = service_pb2.PostModeloutputsRequest(
        model_id='e9359dbe6ee44dbc8842ebe97247b201',
...         inputs=[resources_pb2.Input(data=resources_pb2.Data(image=resources_pb2.Image(base64=file_bytes))
```

```
...         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.inwrite(picname+'.jpg',frame)
...                 multi_part_upload('Dhakshesh', picname+'.jpg', picname+'.jpg')
...                 json_document={"link":COS_ENDPOINT+'/'+'Dhakshesh'+'/'+picname+'.jpg'}
...                 new_document = my_database.create_document(json_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()
```

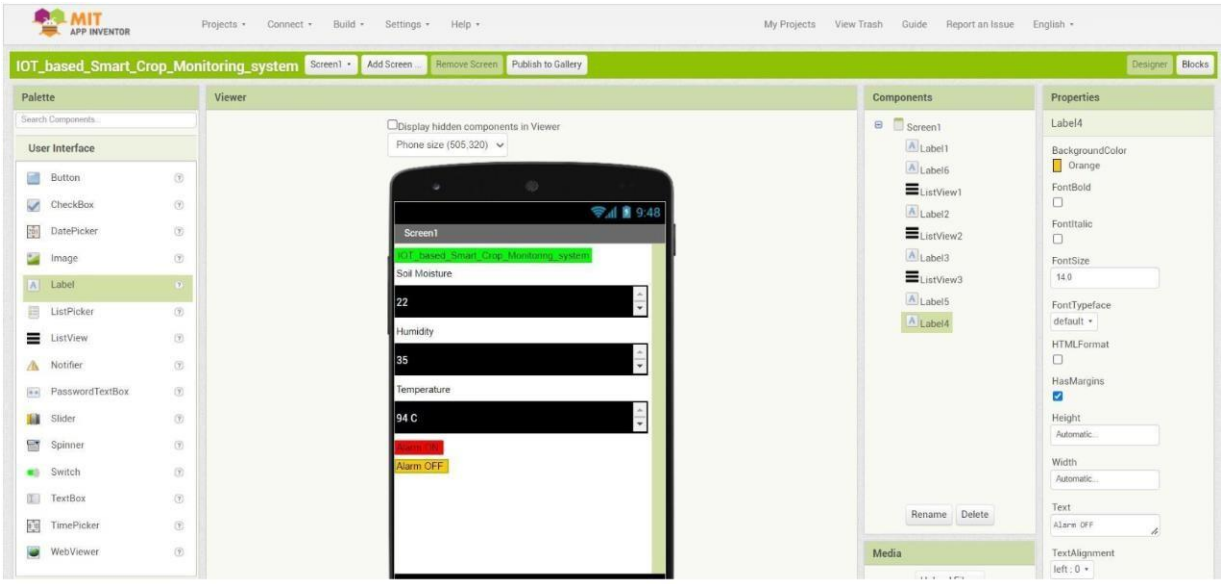

*IDLE Shell 3.8.8*
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/HP/Desktop/crop/crop_protect.py ==========
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..
Ln: 10 Col: 11

## 7.2 Feature 2

## MIT app inventor to design the app:




## Customizing the app interface to display the values:



# 8 TESTING

## 8.2Test Cases

### Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

| Resolution | Severity 1 | Severity 2 | Severity 3 | Severity 4 | Subtotal |
|---|---|---|---|---|---|
| By Design | 11 | 4 | 2 | 2 | 19 |
| Duplicate | 1 | 1 | 2 | 0 | 4 |
| External | 2 | 3 | 0 | 1 | 6 |
| Fixed | 10 | 2 | 3 | 20 | 35 |
| Not Reproduced | 0 | 0 | 2 | 0 | 2 |
| Skipped | 0 | 0 | 2 | 1 | 3 |
| Won't Fix | 0 | 5 | 2 | 1 | 8 |
| Totals | 24 | 15 | 13 | 25 | 77 |

### Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

| Section | Total Cases | Not Tested | Fail | Pass |
|---|---|---|---|---|
| Print Engine | 5 | 0 | 1 | 4 |
| Client Application | 47 | 0 | 2 | 45 |
| Security | 3 | 0 | 0 | 3 |
| Outsource Shipping | 2 | 0 | 0 | 2 |
| Exception Reporting | 11 | 0 | 2 | 9 |
| Final Report Output | 5 | 0 | 0 | 5 |
| Version Control | 3 | 0 | 1 | 2 |

## 9 RESULTS

Thus the IOT based Smart Crop Protection has been build successfully with the help of MIT app, Node.Js, and node red. And the output has been tested and verified using MIT app.

## ADVANTAGES :

- ⇨ Sensors in Agriculture invented to meet the increasing demand for food with minimum resources such water, fertilizers and seeds.
- ⇨ They are easy to operate and use and easy to maintain.
- ⇨ Sensors are cheaper in price and best in quality.
- ⇨ They can used for measuring pollution and global warming for their fields and crops.

## DISADVANTAGES:

- ⇨ Farms are located in remote areas and are far from access to the internet.
- ⇨ A farmer needs to have access to crop data reliably at any time from any location, so connection issues would cause an advanced monitoring system to be useless.
- ⇨ High Cost: Equipment needed to implement IoT in agriculture is expensive.

## 10 CONCLUSION:

Smart farming reduces the ecological footprint of farming. Minimized or site-specific application of inputs, such as fertilizers and pesticides, in precision agriculture systems will mitigate leaching problems as well as the emission of greenhouse gases

## 11 FUTURE SCOPE

IoT smart agriculture products are designed to help monitor crop fields using sensors and by automating irrigation systems. As a result, farmers and associated brands can easily monitor the field conditions from anywhere without any hassle.

## 12 APPENDIX

➔ **GitHub Link**

**https://github.com/IBM-EPBL/IBM-Project-6209-1658824607**

➔ **Source Code**

**https://github.com/IBM-EPBL/IBM-Project-6209-1658824607/blob/main/Final%20Deliverables/Final%20Code/Source%20code**

Prepared by:


- P.Bhrintha
- S.Jayasurya
- P.Janavigha
- A.Divya

**THANK YOU!**