FINAL DELIVERABLES

PROJECT REPORT

TEAM ID	PNT2022TMID23722
PROJECT TITLE	IoT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

PROJECT OBJECTIVES:

- Gain information of Watson IoT Platform.
- linking IoT devices to the Watson IoT platform and
- exchanging the sensor data.
- Gain information on Cloudant DB
- Gain information on using the Clarifai service
- Gain information of storing images in IBM Object Storage and retrieving images
- Creating a Web Application through which the user interacts with the device.

LITERATURE SURVEY:

Abstract:

Most of the farmers are facing many problems nowadays due to many reasons. Our problem to solve is the invasion of various species such as birds and animals that harm the crops that are being cultivated. Various types of species such as birds and animals come to the cultivation field according to the crop that is being cultivated and also according to the season of cultivation. Some wild animals enter the field during night times when the field is near a forest region or when the farm cultivates some fruits and other crops that attract animals. Some animals cross the field in search of food and water and also the birds enter the field for food and they damage all the crops. When the animals enter the field they not only eat food but they also damage the entire field by walking upon the crops and also by spoiling the food crops. The birds, by entering the field they come to eat seeds of the crops and also they tend to drag the crops and ruin the entire field. Some birds enter the field to eat the insects and pests in the field.

Here to solve this situation we are proposing a solution using IOT(Internet of Things) where we use various types of sensors to monitor the entire field and using the help of the internet we tend to send the message to the farmer or the person who is responsible for solving the crisis that is currently occuring. The types of sensors we use will also give the information of the humidity level in the field, the temperature of the field, and detection of animals using their thermal radiation and also we process the information and give them in the form of graphs and images to the farmers for easy understanding.

PROJECT FLOW:

The device will sense the animals and birds using the Clarifai service.If any animal or bird is detected the image will be captured and stored in the IBM Cloud object storage.It also generates an alarm and avoid animals from destroying the crop.The image URL will be stored in the IBM Cloudant DB service. The device will also monitor the soil moisture levels,temperature, and humidity values and send them to the IBM IoT Platform.The image will be retrieved from Object storage and displayed in the web application. A web application is developed to visualize the soil moisture, temperature, and humidity values .Users can also control the motors through web applications.

Implementation:

Firstly, we should create codes for connecting the sensors to the Arduino and connecting the Arrduino to the Wifi module and connecting them to the Internet. Then we should create codes to monitor and intimate messages about humidity and temperature on a regular basis, and codes should be written for PIR sensor and UV sensor to make sure that the motion detection of animals is being intimidated and preventive measures are taken. The preventive measure for every problem should be given according to the problem that arose and the codes for every problem and their solution should be fed on the cloud to access and as a result if the person doesn't know what to do in this type of situation then they can refer to the solutions. Codes should be written to not to intimate humans and also there should be power backup for the system to function efficiently. The backup system is solar and all the products used should consume less power and function more efficiently. The system should be made in a way that it can function more effectively even when there is very low data rate. The program should be coded in such a way.

To Accomplish this,we have to complete all the activities and tasks listed below:

1.Create and configure IBM Cloud Services o Create IBM Watson IoT Platform o Create a device & configure the IBM IoT Platform
- o Create Node-RED service
- o Create a database in Cloudant DB to store location data
- o Create a cloud object storage service and create a bucket to store the images

2.Develop a python script to publish the sensor parameters like Temperature, Humidity, and Soil Moisture to the IBM IoT platform and detect the animals and birds in video streaming using Clarifai.

3.Develop a web Application using Node-RED Service.

4. Display the image in the Node-RED web UI and also display the temperature, humidity, and soil moisture levels. Integrate the buttons in the UI to control the Motors

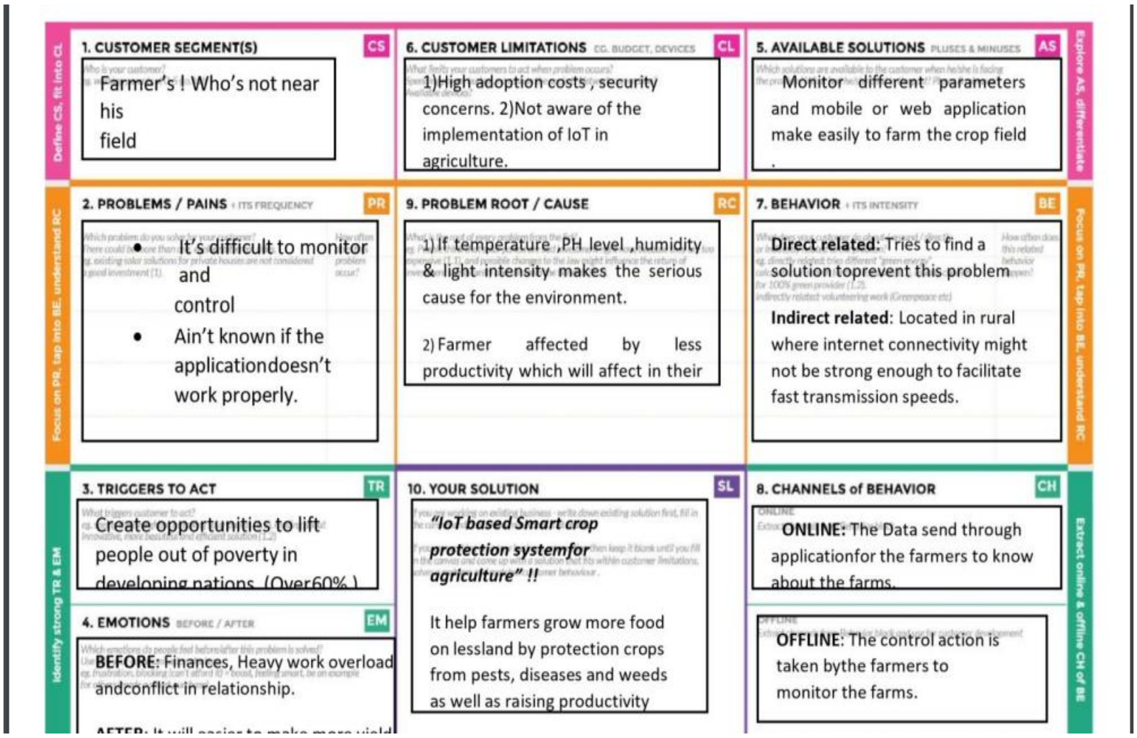
Conclusion:

AS a result of this system, we can detect the changes in the field easily and intimate the farmers about it and also we can take precautions and do remedies accordingly. Here we use very low power consuming highly efficient components that give us accurate results and also they perform at low data rate conditions without any lag and help in finding the remedies. This crop protection system helps in detection of all kinds of external dangers and it saves time and money to the farmers before any loss that may occur. With the help of this system the farmers can be in a peaceful environment at ease without any pressure.

PROBLEM STATEMENT:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Farmer	Monitor my crops	There are some disturbances	Of birds, animals & insects	Very frustrated and depressed about my field
PS-2	Farmer	Prevent animals from attacking my field	There is no easy and helpful technology	Of many kinds of birds & animals attack according to the type of cultivation	Unable to do anything many times

PROPOSED SOLUTION FIT:



REQUIRED SOFTWARE:

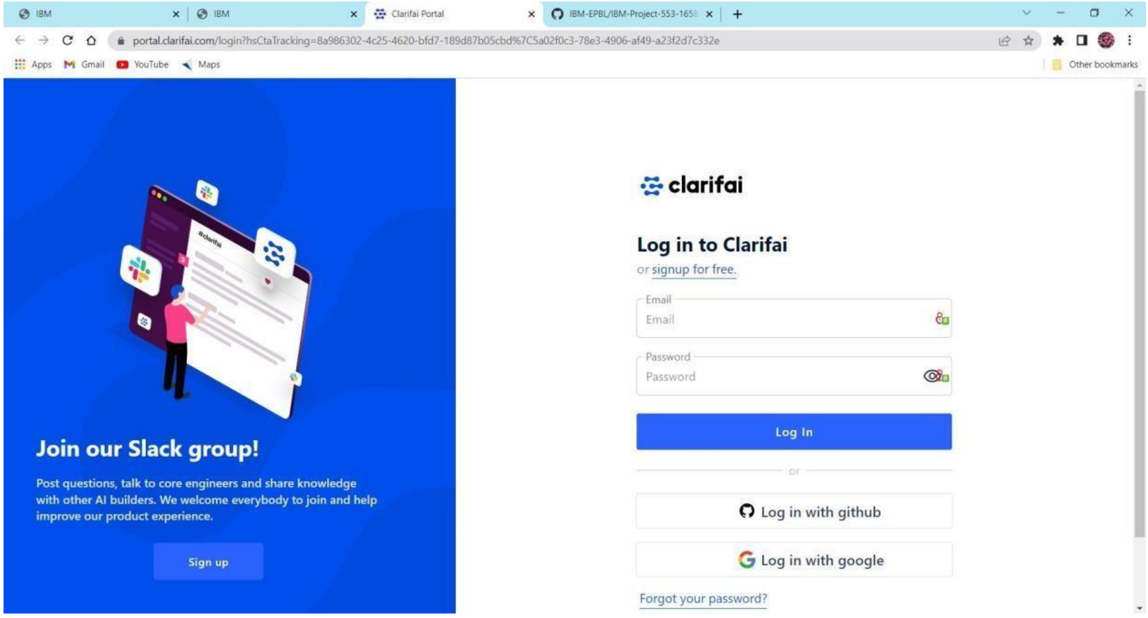
- **CLARIFAI**
- **IBM WATSON IOT PLATFORM**
- **PYTHON IDLE**
- **NODE RED**
- **MIT APP INVENTOR**

CLARIFAI:

Clarifai provides an end-to-end platform with the easiest to use UI and API in the market. Clarifai Inc. is an artificial intelligence (AI) company that specializes in computer vision and uses machine learning and deep neural networks to identify and analyse images and videos. The company offers its solution via API, mobile SDK, and on-premise solutions.

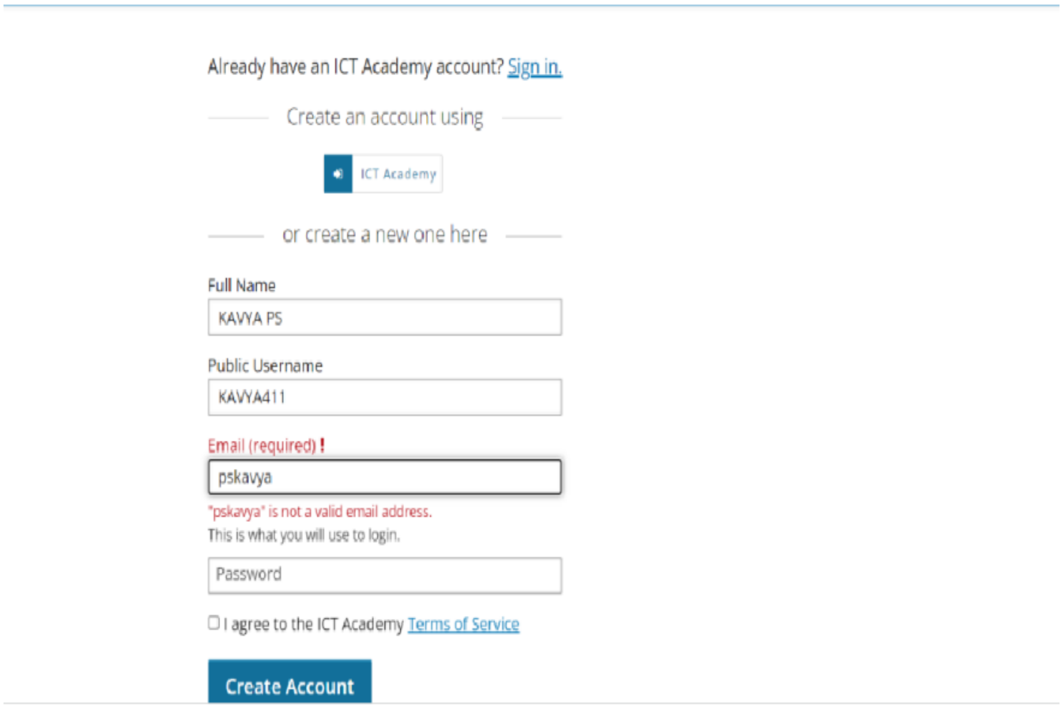
STEP 1:

- Open Clarifai portal in web browser.



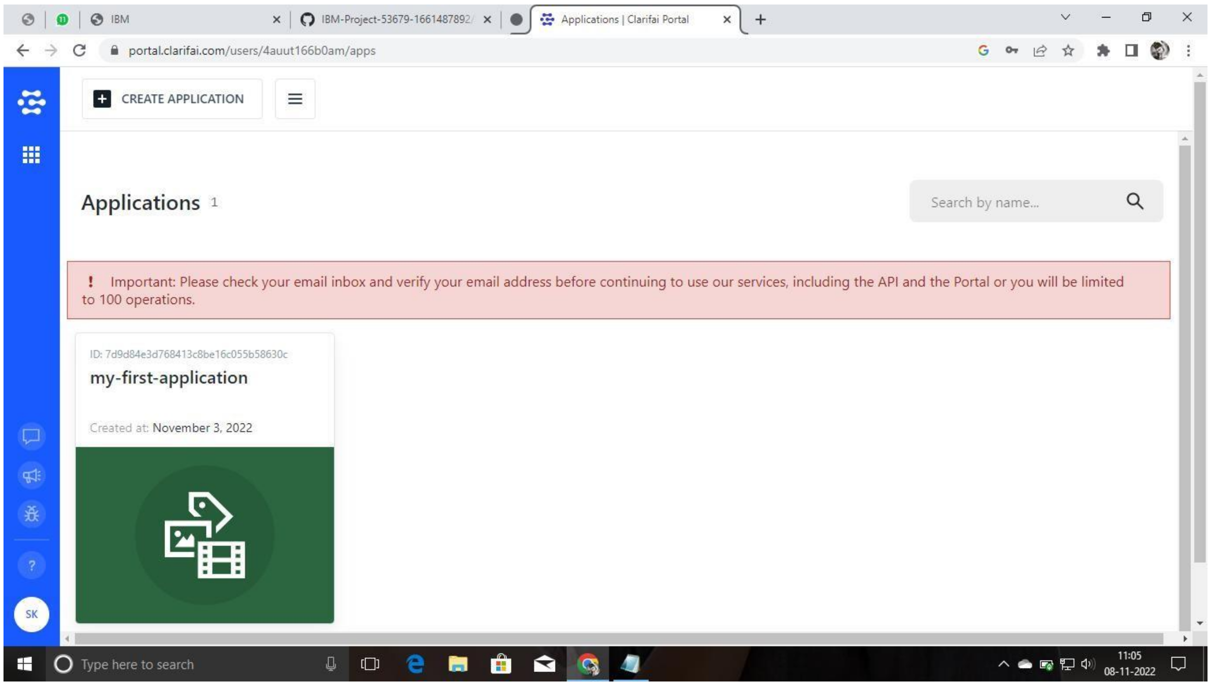
STEP 2:

- Signup using the required user mail and password



STEP 3:

Finally, Created an account



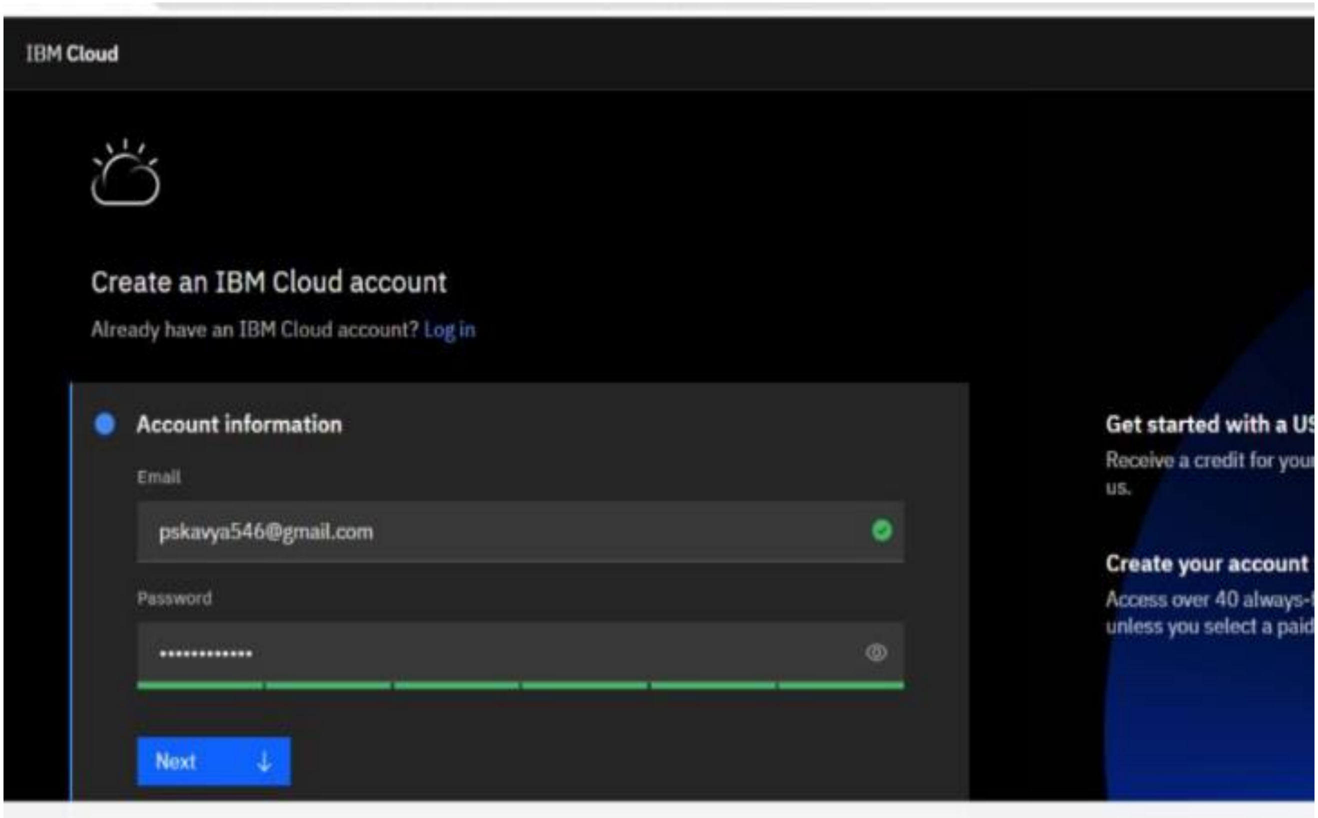
IBM WATSON IoT PLATFORM:

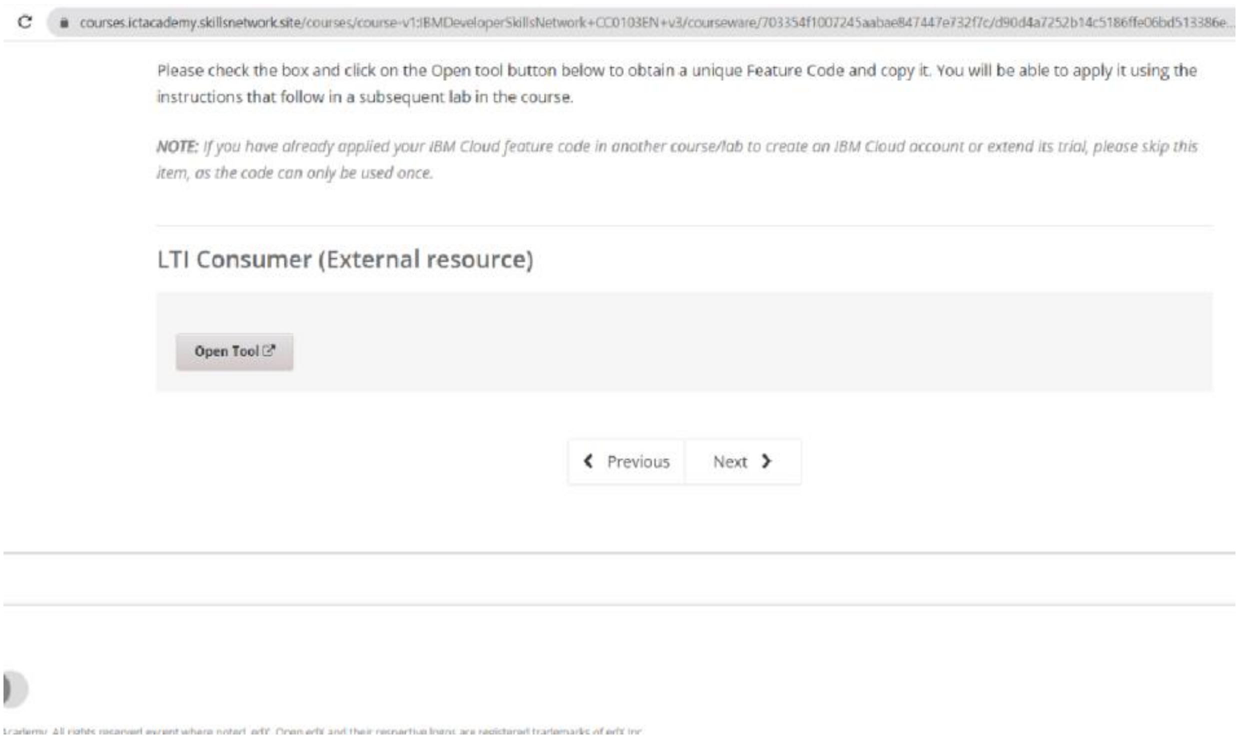
We need to have basic knowledge of the following cloud services:

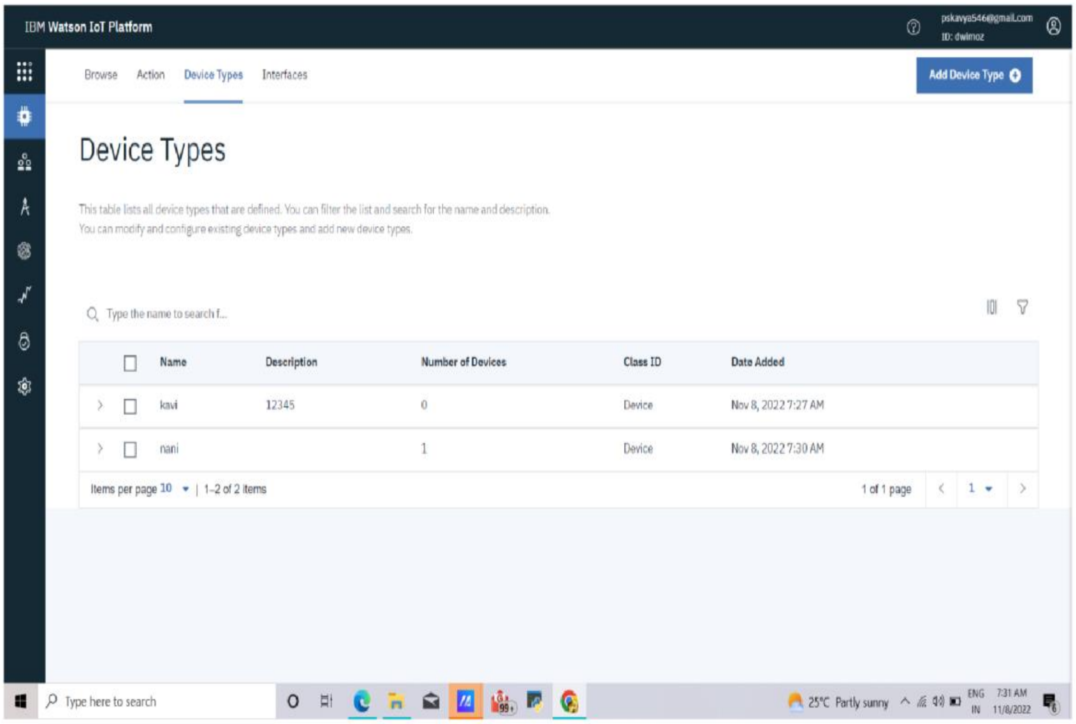
- IBM Watson IoT Platform
- Node-RED Service
- Cloudant DB

We need to create an IBM Cloud Account to complete this project.

LOGIN:





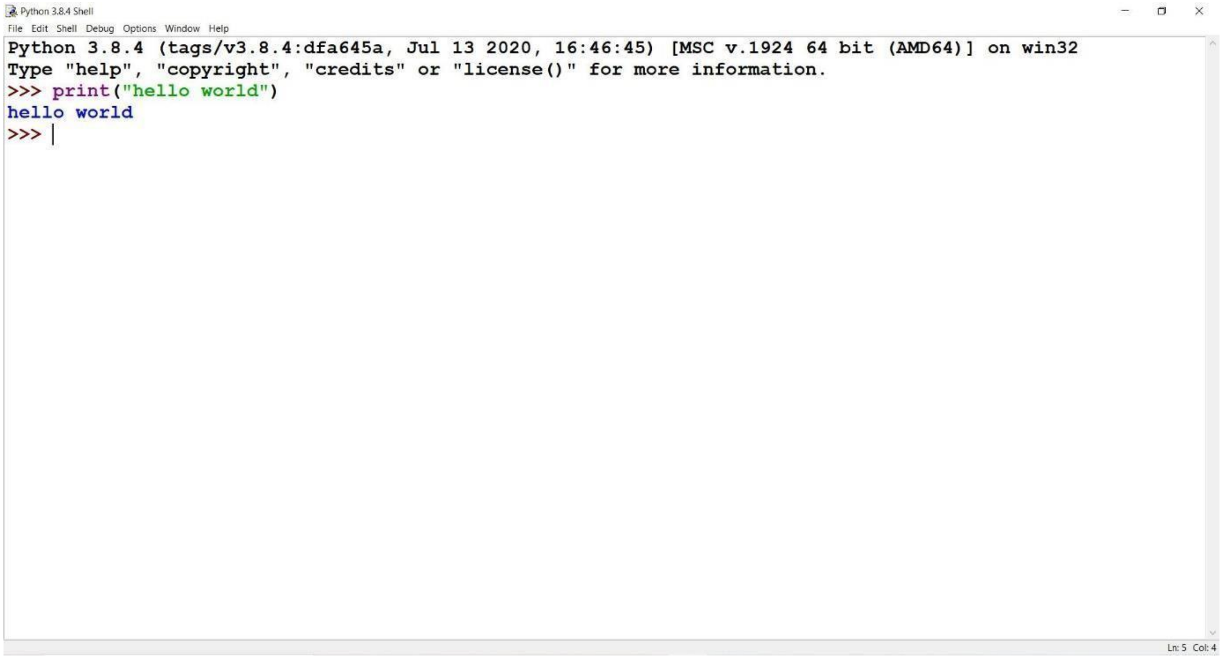


PYTHON IDLE INSTALLATION:

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

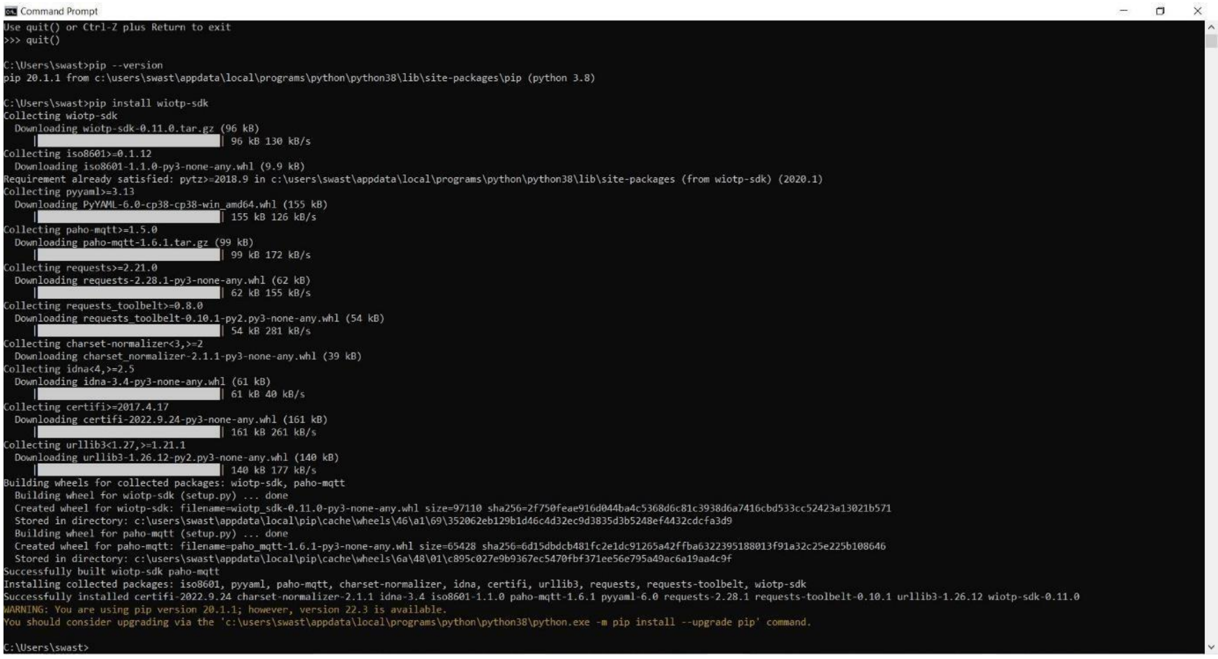
STEP 1:

- Python is installed successfully

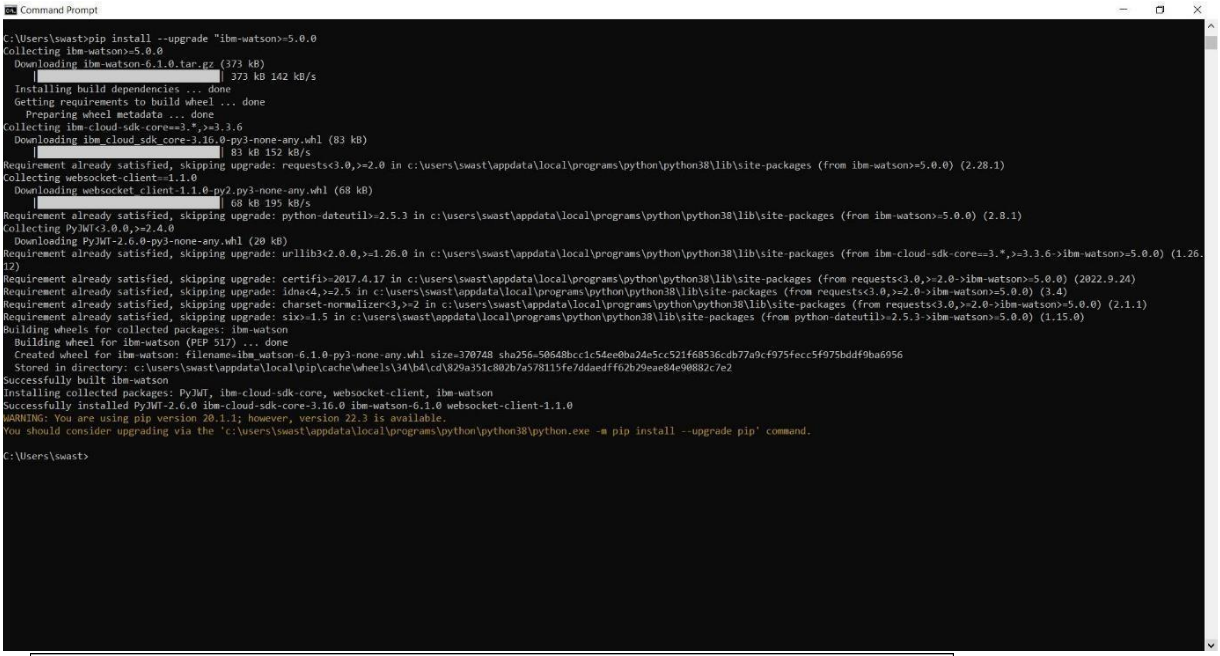


STEP 2:

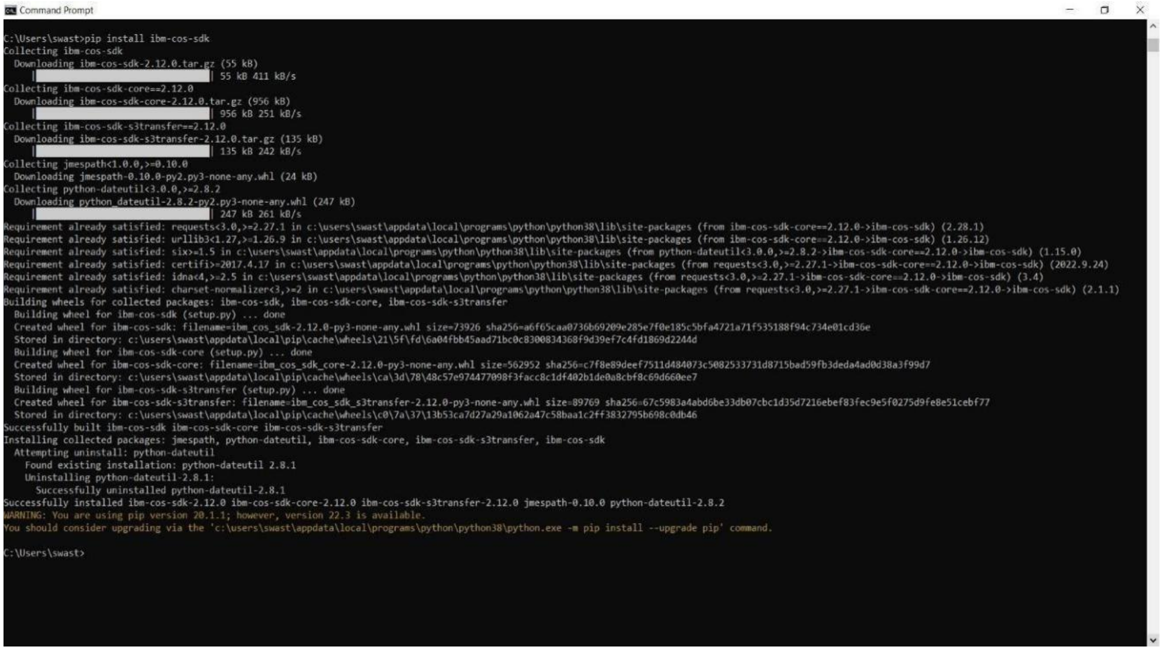
- The required python libraries are installed.
- Watson IoT Python SDK to connect to IBM Watson IoT Platform using python code is installed
- pip install wiotp-sdk



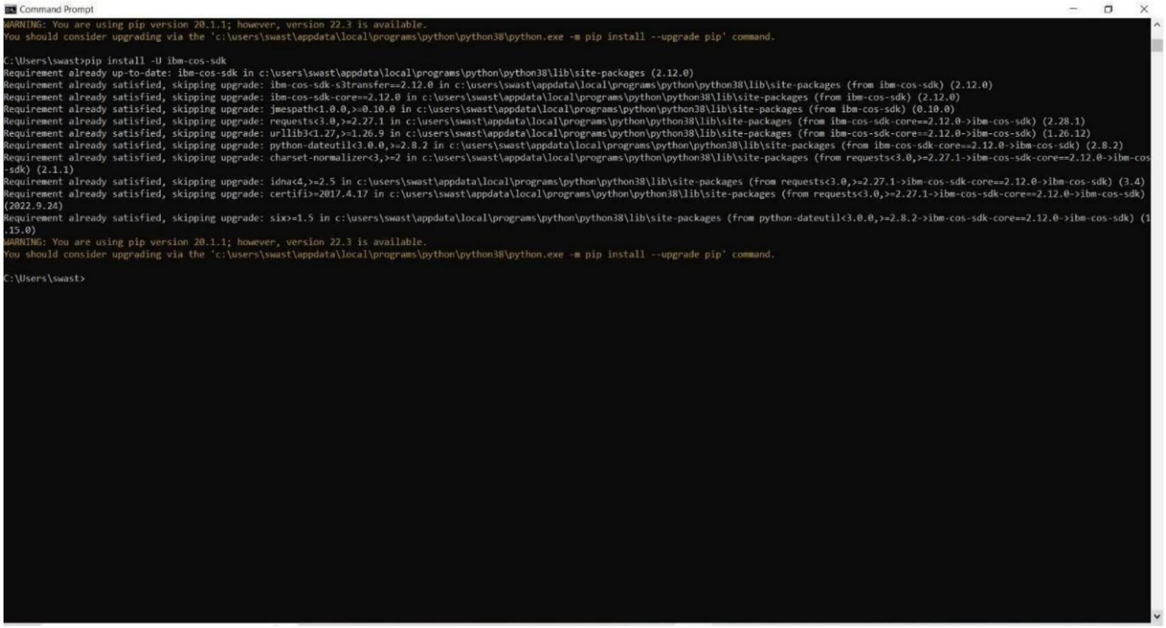
- Python client library for IBM Text to Speech is installed
- pip install --upgrade "ibm-watson>=5.0.0



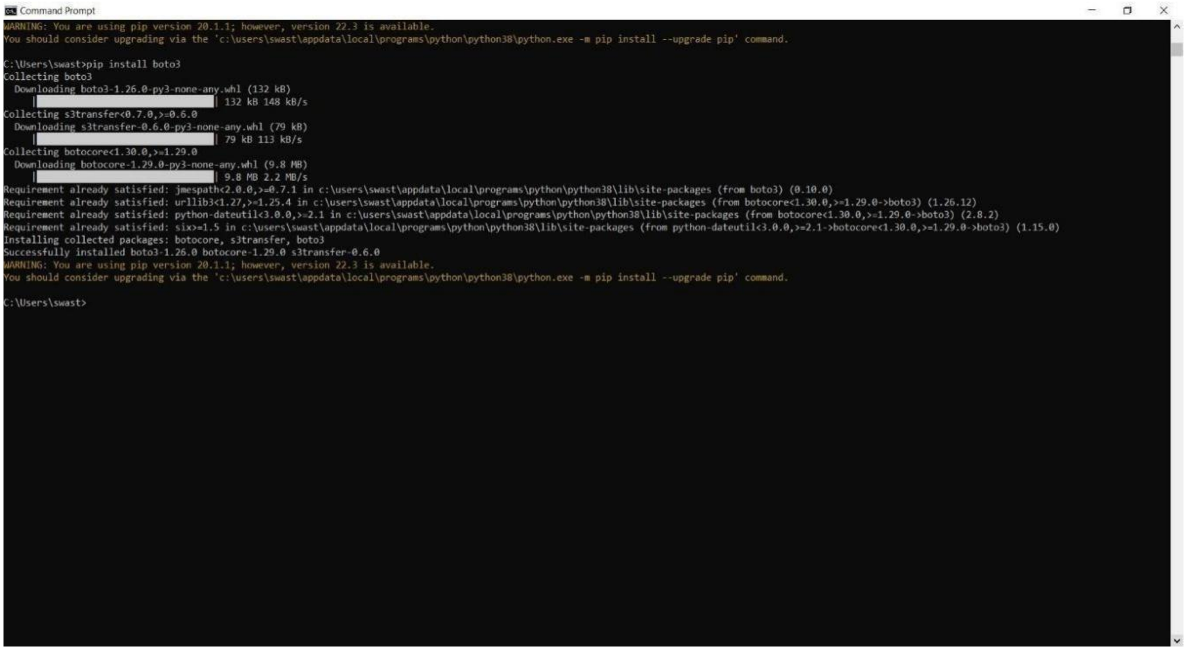
- Required Libraries for cloud object storage is installed.
- pip install ibm-cos-sdk



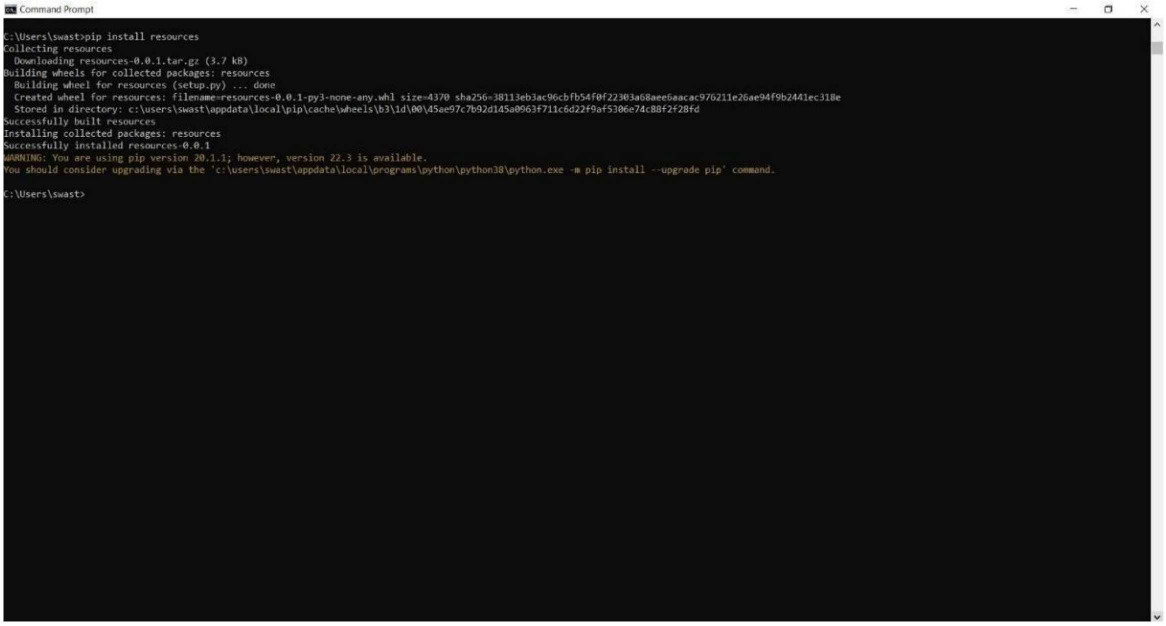
- pip install -U ibm-cos-sdk



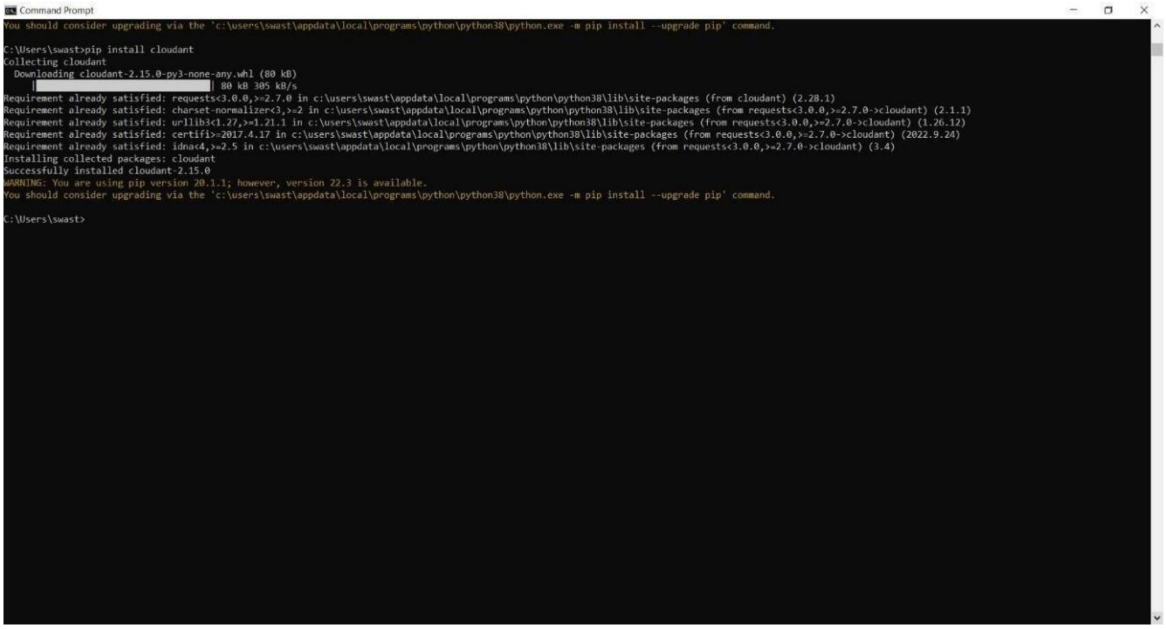
- pip install boto3



- pip install resources



- pip install cloudant



DATA FROM PYTHON TO IBM:

Python code to generate random data and pass it to IBM Watson IoT platform

Source Code:

```
import
time
import
sys
```

```
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 = "g5d4qO8EIgv4TWUCJj4hfEzgalqEjrDbE82AJDWlAOHo"
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-
W2njldnwtjO16V53LAVUCqPwc2aHTLmlj1xXvtdGKJBn",
"88cc5f47c1a28afbfb8ad16161583f5a", url="https://d6c89f97-cf91-48b7-b14b-
c99b2fe27c2f-bluemix.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 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')
    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.I
mage(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()
cv2.destroyAllWindows()
```

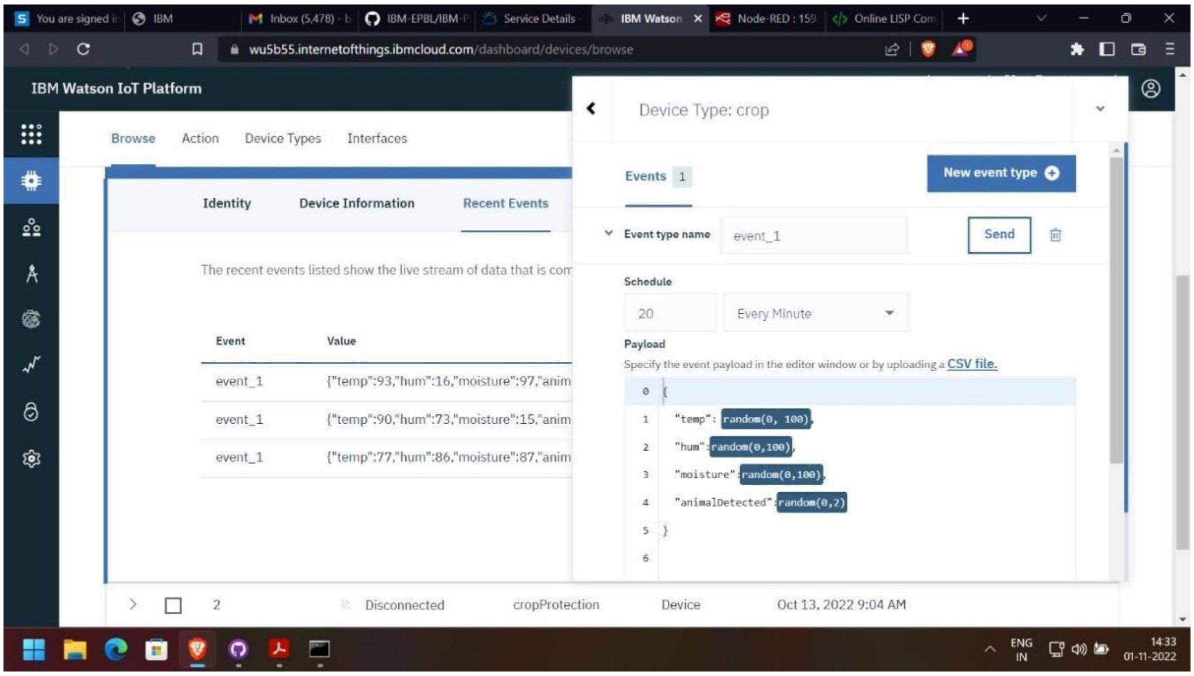
DATA GENERATION IOT PLATFORM:

Source code is deployed on IBM Watson IoT platform to generate sensor data.

Source Code:

```
{
        "temperature": random(0, 100),
        "humidity": random(0, 100),
        "moisture": random(0, 100),
        “animalDetected”:random(0,2)
}
```

Output:



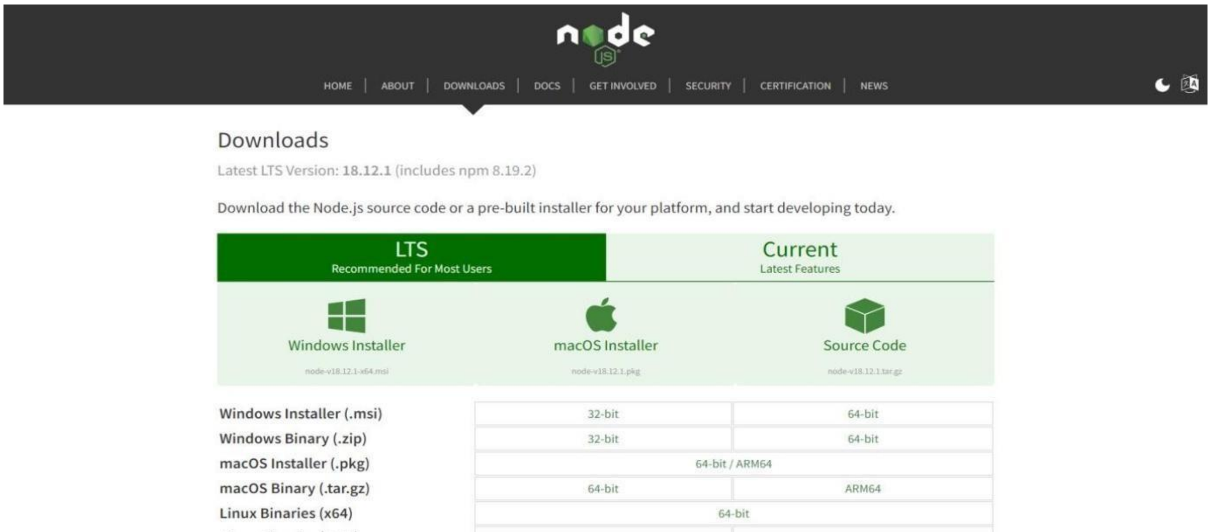
PYTHON CODE TO IBM:

```
import time
import sys

import ibmiotf.application
import ibmiotf.device import
random



#Provide your IBM Watson Device Credentials
organization = "wu5b55"

deviceType = "crop1"
deviceId = "1234"
authMethod = "token"
authToken = "1234567890"


# Initialize GPIO



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


except Exception as e:

        print("Caught exception connecting device: %s" % str(e))sys.exit()


# Connect and send a datapoint "hello" with value "world" into thecloud as
an event of type "greeting" 10 times

deviceCli.connect()
```

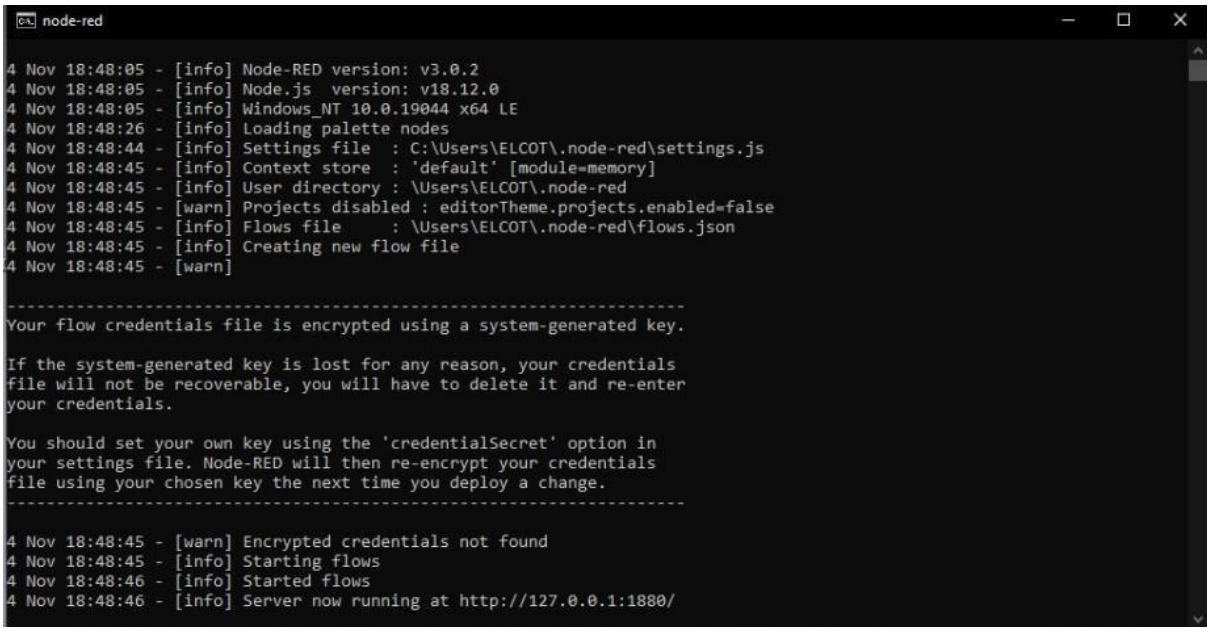
```
while True:

        #Get Sensor Data from DHT11


        temp=random.randint(0,100)
        Hum=random.randint(0,100)
        moisture=random.randint(0,100)


        data = { 'temperature' : temp, 'Humidity': Hum,
'Moisture':moisture }
#print data
        def myOnPublishCallback():
            print ("Temperature = " + str(temp)+" C Humidity = " +
str(hum)+ " moisture = " + str(moisture) + "to IBM Watson")


        success = deviceCli.publishEvent("IoTSensor", "json",data,qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(10)


        deviceCli.commandCallback = myCommandCallback


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

NODE-JS CONNECTION:

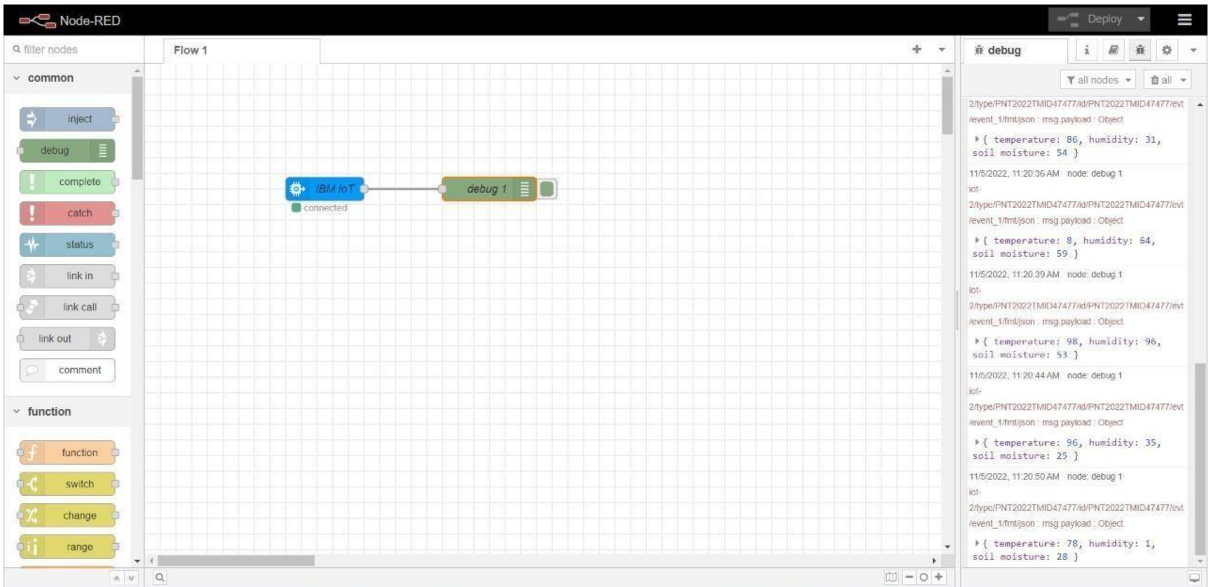
STEP1: Download and Install NODE JS.



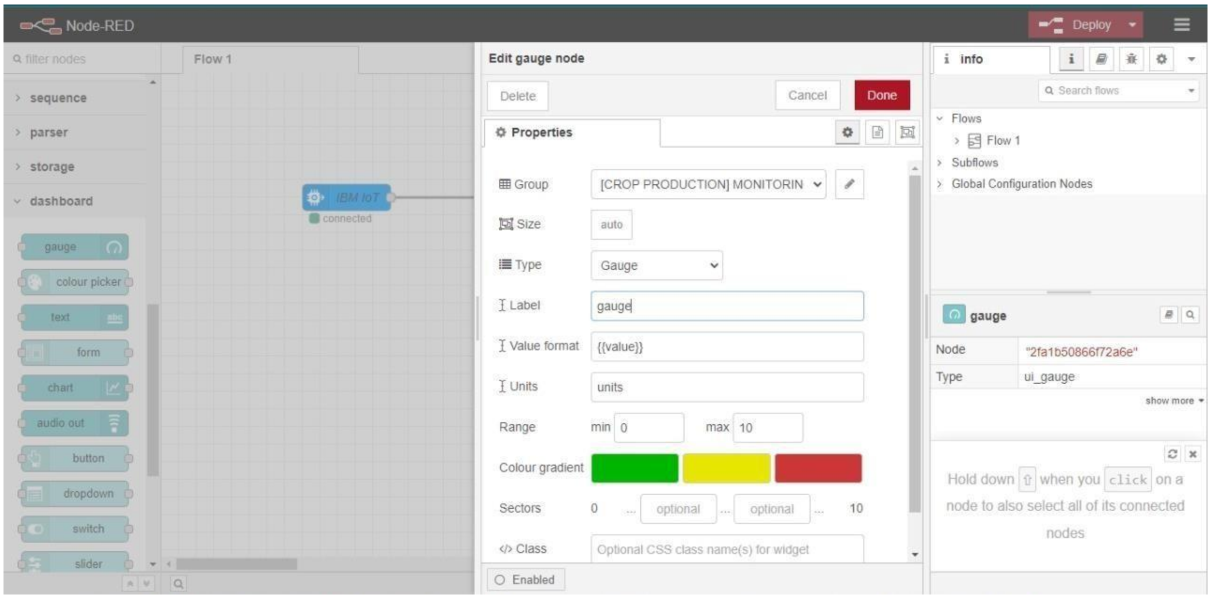
STEP2: Setup node.js and configure command prompt for error check .open node-red from the generated link.

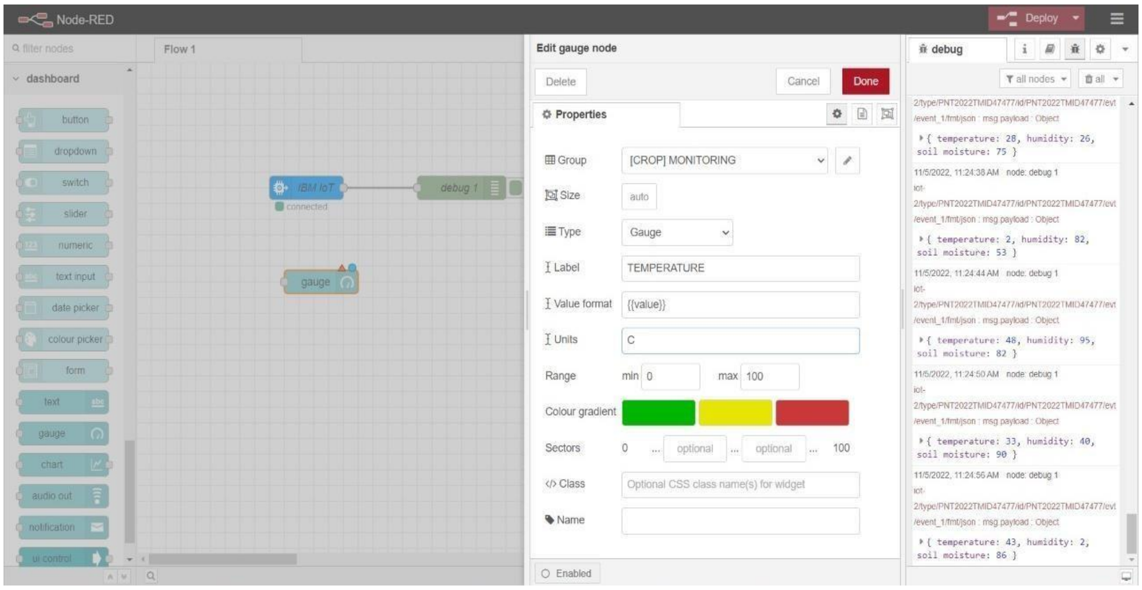


STEP3: Connect IBM IOT in and Debug 1 and Deploy.



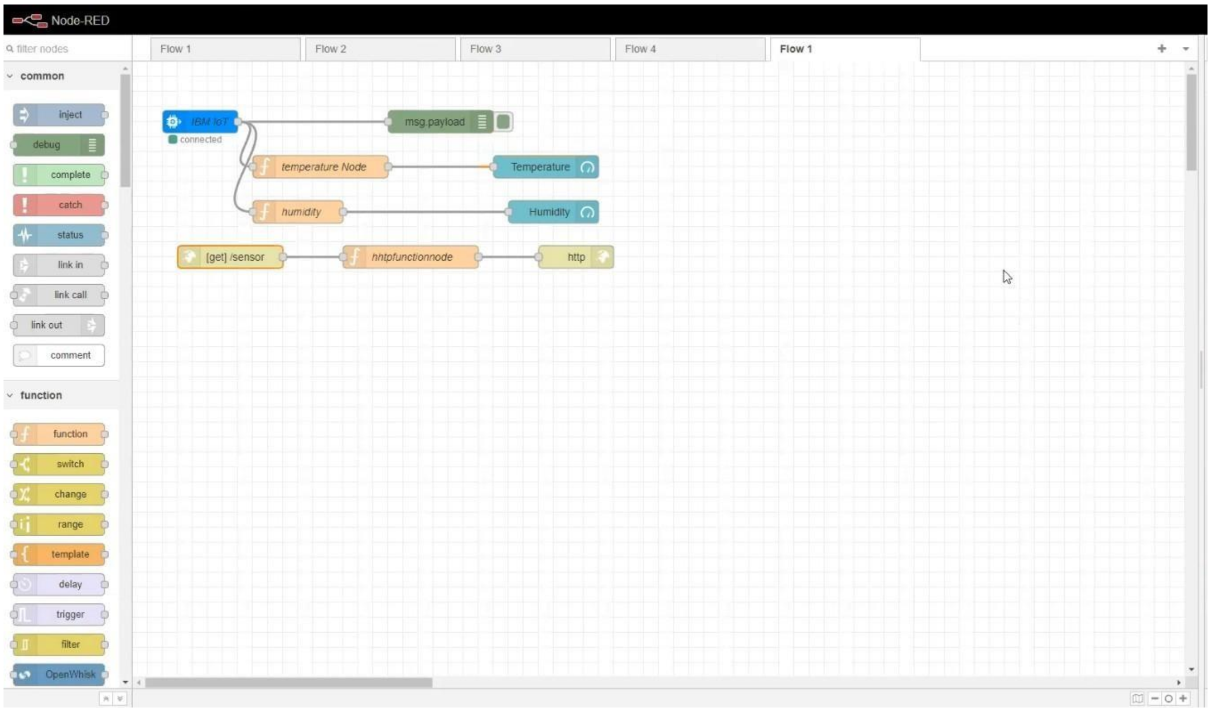
STEP4: Edit gauge node (Here the gauge nodes are named as Temperature, Humidity and Soilmoisture).



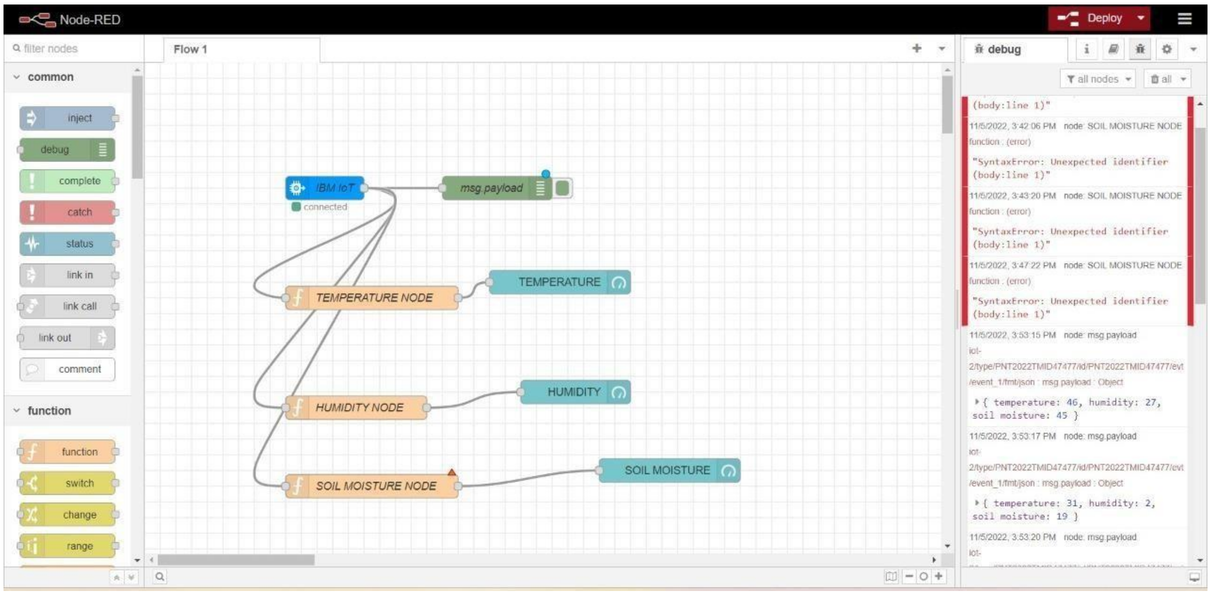


SIMULATION:

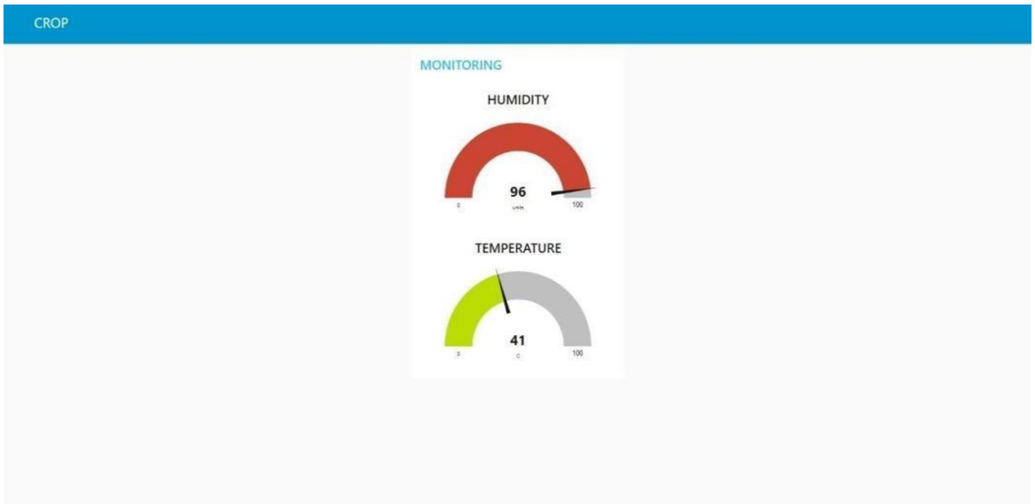
STEP1: Simulated program to get the random values

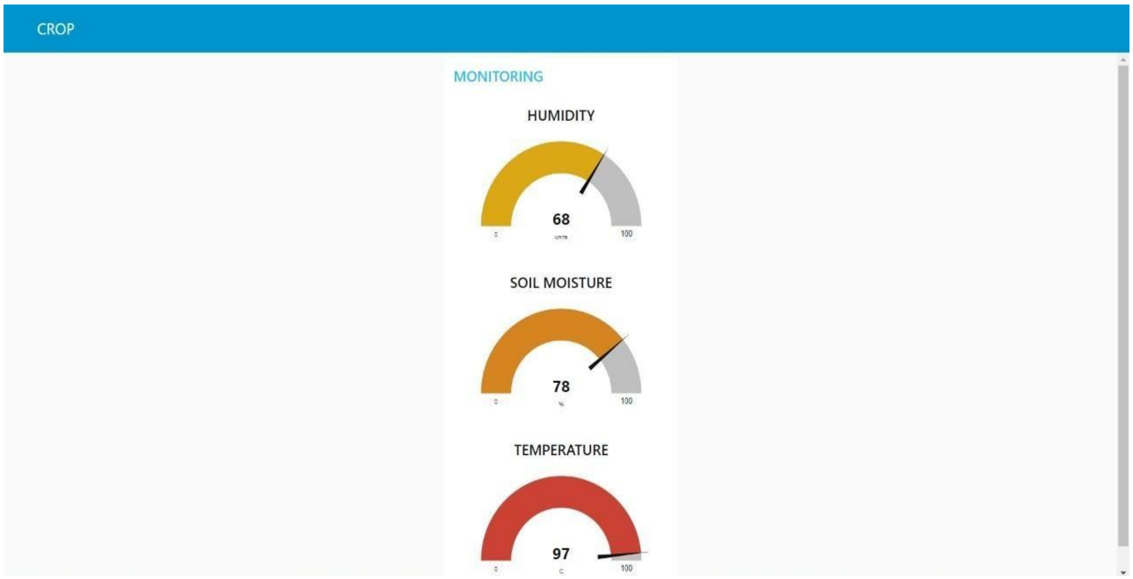


STEP2: Generate debug message from IBM Watson IoT Platform and connect the nodes.



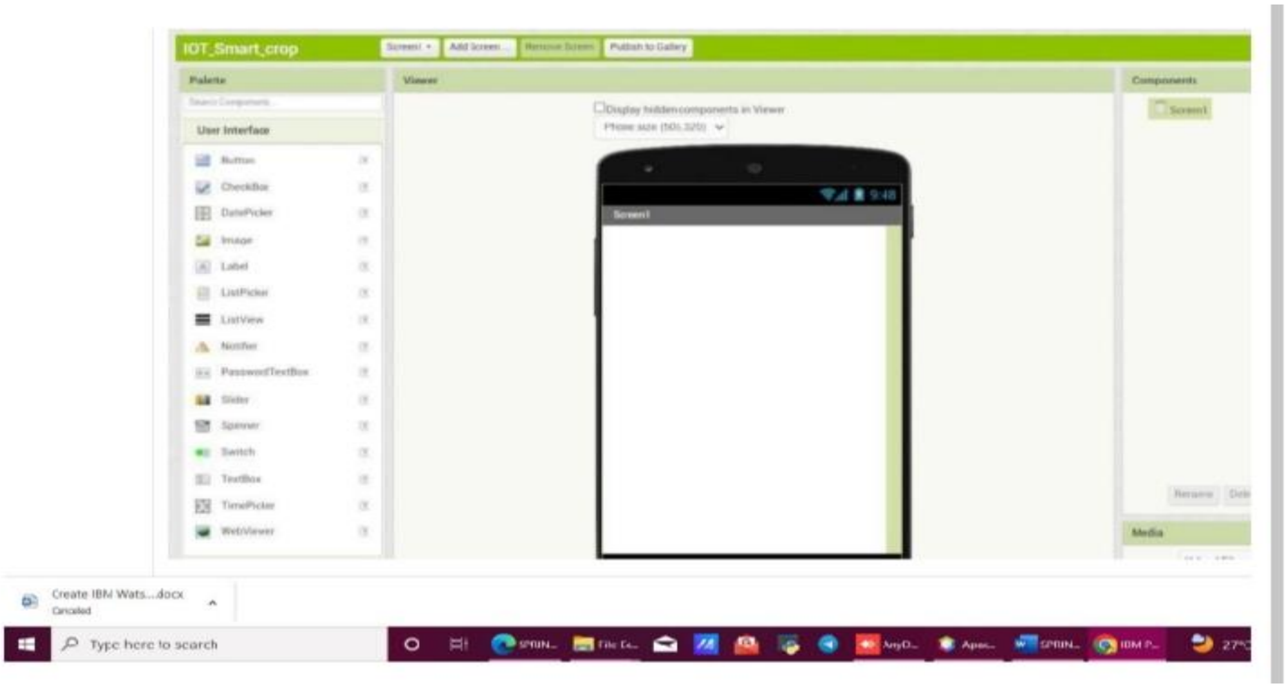
STEP3: Generate the some output from recent events.



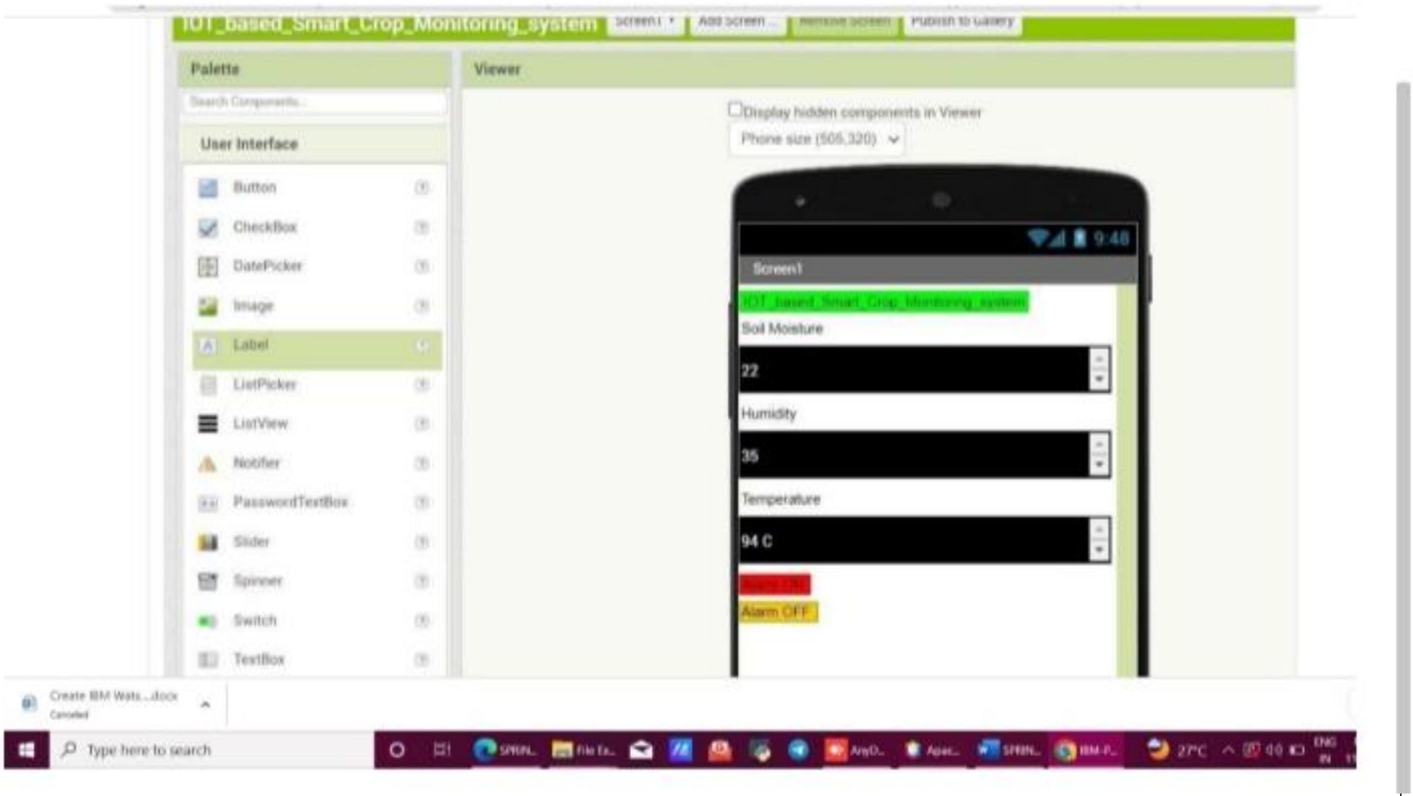


MIT APP INVENTOR:

STEP 1: MIT APP inventor to design the APP.



STEP 2: Customize the App interface to Display the Values.



ADVANTAGES:

- Farmers can monitor the health of farm animals closely, even if they are physically distant.
- Smart farming systems reduce waste, improve productivity and enable management of a greater number of resources through remote sensing.
- High reliance.
- Enhanced Security.

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.

APPLICATIONS:

- Monitoring the crop field with the help of sensors (light, humidity, temperature, soil moisture, etc.)
- Automating the irrigation system
- Soil Moisture Monitoring (including conductivity and pH)