



IoT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE



PROJECT REPORT

Submitted By

TEAM ID PNT2022TMID32078

JEYA SURYA R (TEAM LEAD) (731619205019)

DAMODHARAN M (731619205008)

VIGNESHWARAN D (731619205059)

LOGAMAGESH E (731619205030)

in partial fulfilment for the award of the

degree of

BACHELOR OF TECHNOLOGY

IN

K S R INSTITUTE FOR ENGINEERING AND TECHNOLOGY,

TIRUCHENGODE

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|------------------------|--|-----------------|
| 1 | INTRODUCTION | 4 |
| | 1.1 PROJECT OVERVIEW | 4 |
| | 1.2 PURPOSE | 4 |
| 2 | LITERATURE SURVEY | 5 |
| | 2.1 EXISTING PROBLEM | 5 |
| | 2.2 REFERENCES | 5 |
| | 2.3 PROBLEM STATEMENT DEFINITION | 6 |
| 3 | IDEATION AND PROPOSED SOLUTION | 7 |
| | 3.1 EMPATHY MAP CANVAS | 7 |
| | 3.2 IDEATION AND BRAINSTORMING | 7 |
| | 3.3 PROPOSED SOLUTION | 8 |
| | 3.4 PROBLEM-SOLUTION FIT | 9 |
| 4 | REQUIREMENT ANALYSIS | 10 |
| | 4.1 FUNCTIONAL REQUIREMENT | 10 |
| | 4.2 NON- FUNCTIONAL REQUIREMENT | 10 |
| 5 | PROJECT DESIGN | 11 |
| | 5.1 DATA FLOW DIAGRAM | 11 |
| | 5.2 SOLUTION AND TECHNOLOGY ARCHITECTURE | 11 |
| | 5.3 USER-STORIES | 13 |

| | | |
|-----------|--|-----------|
| 6 | PROJECT PLANNING AND SCHEDULING | 14 |
| | 6.1 SPRINT PLANNING AND ESTIMATION | 14 |
| | 6.2 SPRINT DELIVERY SCHEDULE | 15 |
| | 6.3 REPORT FROM JIRA | 15 |
| 7 | CODING AND SOLUTIONS | 25 |
| | 7.1 FEATURE 1 | 25 |
| | 7.2 FEATURE 2 | 26 |
| | 7.3 DATABASE SCHEMA | 27 |
| 8 | TESTING | 29 |
| | 8.1 TEST CASES | 29 |
| | 8.2 USER ACCEPTANCE TESTING | 29 |
| 9 | RESULT | 30 |
| | 9.1 PERFORMANCE METRICS | 30 |
| 10 | ADVANTAGES AND DISADVANTAGES | 33 |
| 11 | CONCLUSION | 34 |
| 12 | FUTURE SCOPE | 34 |
| 13 | APPENDIX | 35 |
| | 13.1 SOURCE CODE | 35 |
| | 13.2 GITHUB & PROJECT DEMO LINK | 35 |
| 14 | REFERENCES | 36 |

CHAPTER 1

INTRODUCTION

1.1 Project Overview

- The device will detect 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 application.

1.2 Purpose

An intelligent crop protection system helps the farmers in protecting the crop from the animals and birds which destroy the crop. This system also helps farmers to monitor the soil moisture levels in the field and also the temperature and humidity values near the field. The motors and sprinklers in the field can be controlled using the mobile application. 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 occurring. 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.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing Problem

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.

2.2 REFERENCES

Shishir Bagal , Krunal Mahajan , Riya Parate , Ekta Zade , Shubham Khante (2021) have investigated the title of “Smart Crop Protection System Using IOT” . The Smart protection system defines that this project help to farmer for the protection of a farm. We have designed this project for the only secure from animals but we this project have the provision to secure from the human begins also. This can achieve by the help of IOT device that we are discuss in this paper. The SCPS work on the battery so that this project can be easily portable and also we are add solar panels and converter modules this can help the battery to charge from solar energy. The IOT device is used to indicate the farmer by a message while someone enter into the farm and we are used SD card module that helps to store a specified sound to fear the animals.

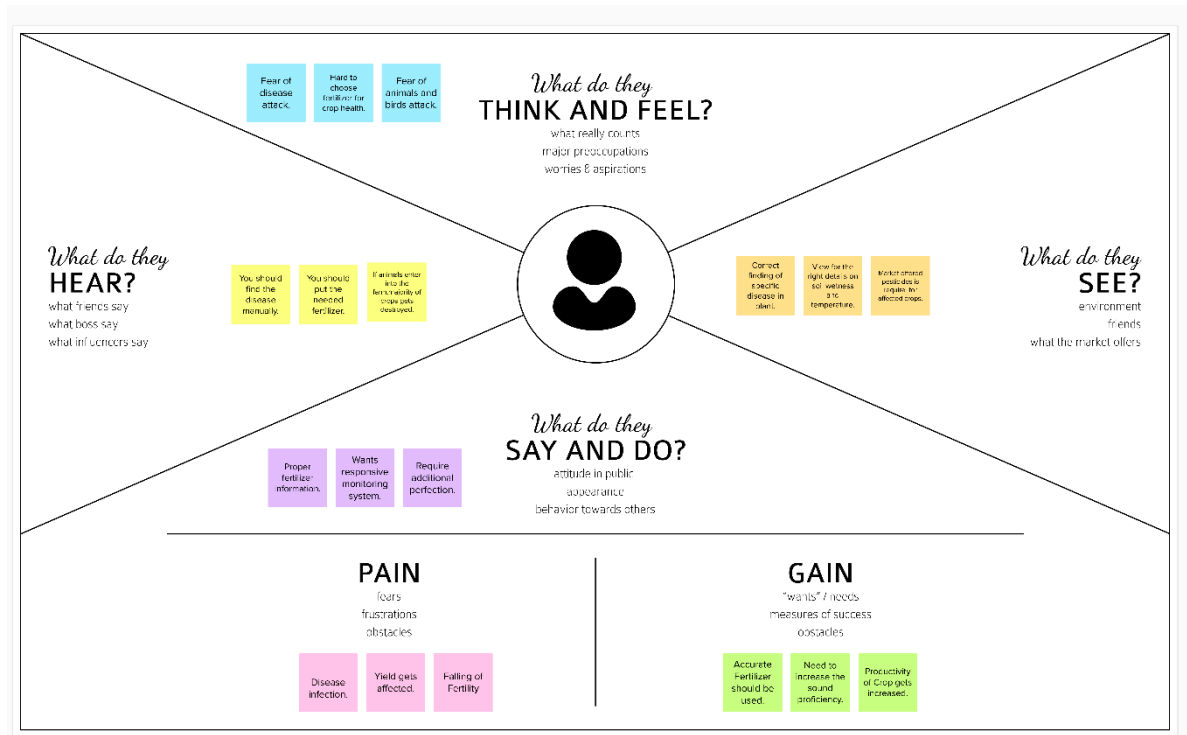
2.3 Problem Statement Definition

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.

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

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. It is not possible for farmers to barricade entire fields or stay on field 24 hours and guard it. So here we propose automatic crop protection system from animals and fire. This is aarduino Uno based system using microcontroller. This system uses a motion sensor to detect wild animals approaching near the field and smoke sensor to detect the fire. In such a case the sensor signals the microcontroller to take action. If there is a smoke, it immediately turns ON the motor. This ensures complete safety of crops from animals and from fire thus protecting the farmer's loss. This is aarduino. Uno based system using microcontroller. This system uses a motion sensor to detect wild animals approaching near the field and smoke sensor to detect the fire. In such a case the sensor signals the microcontroller to take action.



3.3 Proposed Solution

Moisture sensor is interfaced with Arduino Microcontroller to measure the moisture level in soil and relay is used to turn ON and OFF the motor pump for managing the excess water level. It will be updated to authorities through IOT. Temperature sensor connected to microcontroller is used to monitor the temperature in the field. The optimum temperature required for crop cultivation is maintained using sprinklers. IOT based fertilizing methods are followed, to minimize the negative effects on growth of crops while using fertilizers.

The PIR sensor and UV sensors detect the motion of animals and birds for a particular arrange. The thermal radiation temperature of humans at different ages is fed to the system so there won't be any false alarm. If any invasion of animals is found, the camera focuses on the region and the processed image is sent to the farmer. After seeing the image of the animal that entered, they can decide to take any actions. A fence is built around the field to prevent large animals from entering where the sensors are placed at all the corners of the field fully covering the entire region.

3.4 PROBLEM-SOLUTION FIT

| | | |
|--|--|--|
| 1. CUSTOMER SEGMENT(S) CS • Farmers who trying to protect Crops from various problems | 6. CUSTOMER LIMITATIONS CL <small>EG. BUDGET, DEVICES</small> • Limited supervision. • Limited financial Constrains. • Lack of man power. | 5. AVAILABLE SOLUTIONS AS <small>PLUSES & MINUSES</small> • Automation in irrigation. • CCTV Camera to monitor and supervise the crops. • Alarm system to give alert while animals attacks the crops. |
| 2. PROBLEMS / PAINS PR <small>+ ITS FREQUENCY</small> • Crops are not irrigated properly. • Improper maintenance of crops. • Lack of knowledge among farmers in usage of fertilizers and hence crops are affected. • Requires protecting Crops from Wild animals attacks, birds and pests. | 9. PROBLEM ROOT / CAUSE RC • Due to insufficient labour forces. • Due to various environmental factors such as temperature climate, topography and soil quality which results in crop destruction. • Due to high ammonia, urea, potassium and high PH level fertilizers. • Crops are damaged and it affects growth. | 7. BEHAVIOR BE <small>+ ITS INTENSITY</small> • Asks suggestions from surrounding peoples and implement the recent technologies. • Consumes more time in crop land. • Searching for an alternative solution for an existing solution. |
| 3. TRIGGERS TO ACT TR • By seeing surrounding Crop land with installing machineries. • Hearing about innovative technologies and effective solutions. 4. EMOTIONS EM <small>BEFORE / AFTER</small> • Mental frustrations due to insufficient production of crops. • Felt smart enough to follow the available technologies with minimum cost. | 10. YOUR SOLUTION SL • Moisture sensor is interfaced with Arduino Microcontroller to measure the moisture level in soil and relay is used to turn ON and OFF the motor pump for managing the excess water level. It will be updated to authorities through IoT. • Temperature sensor connected to microcontroller is used to monitor the temperature in the field. The optimum temperature required for crop cultivation is maintained using sprinklers. • IoT based fertilizing methods are followed, to minimize the negative effects on growth of crops while using fertilizers. • Image processing techniques with IoT is followed for crop protection against animal attacks. | 8. CHANNELS of BEHAVIOR CH ONLINE Using different platforms /social media to describe the working and uses of smart Crop protection device. OFFLINE • Giving awareness among farmers about the application of the device. |

CHAPTER 4

REQUIREMENT ANALYSIS

4.1Functional Requirement

Following are the functional requirements of the proposed solution.

- User Registration ,Registration through Form Registration through Gmail Registration through LinkedIN
- User Confirmation ,Confirmation via Email Confirmation via OTP
- Tracking Expense Helpful insights about money management
- Alert Message Give alert mail if the amount exceeds the budget limit Category This application shall allow users to add categories of their expenses

4.2Non Functional requirement

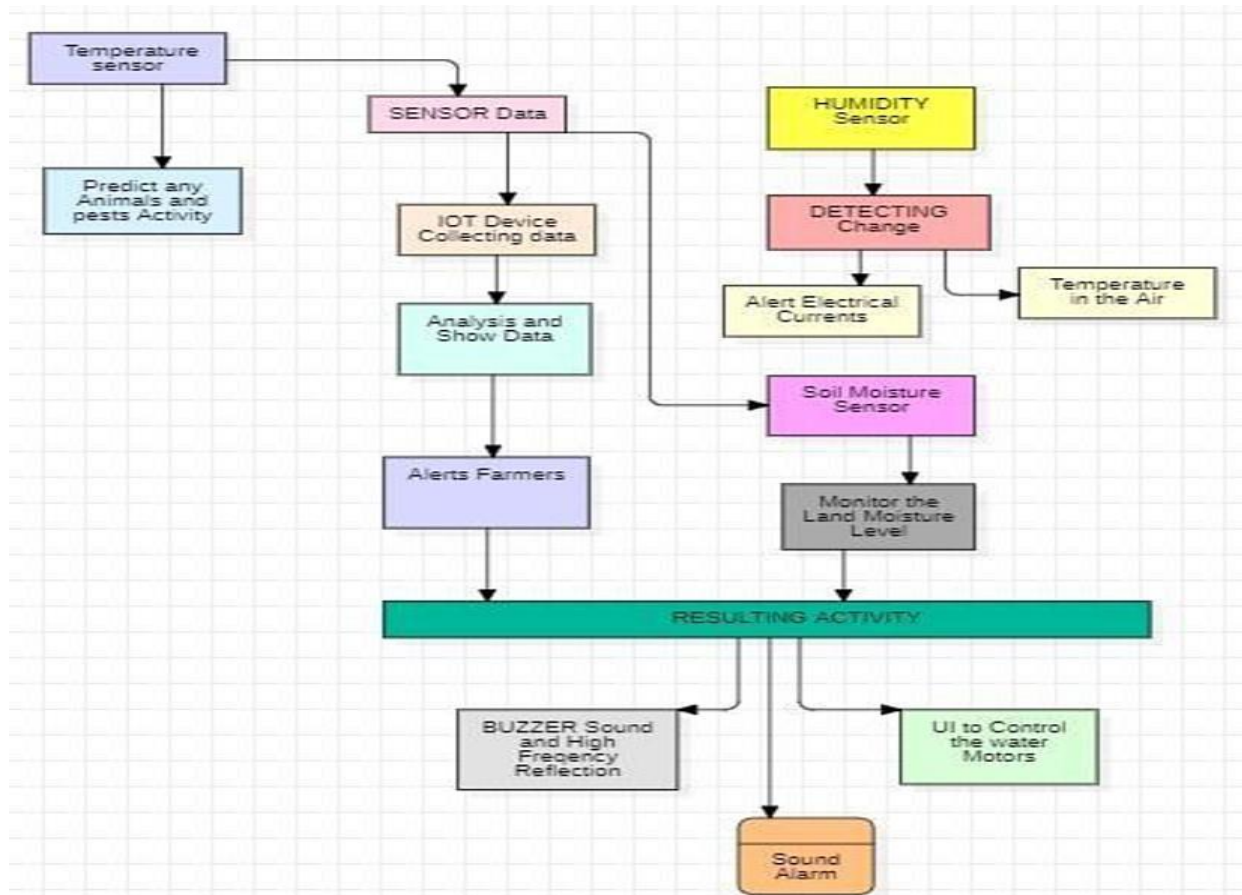
Following are the non-functional requirements of the proposed solution.

- Usability You will able to allocate money to different priorities and also help you to cut down on unnecessary spending
- Security More security of the customer data and bank account details.
- Reliability Used to manage his/her expense so that the user is the path of financial stability. It is categorized by week, month, and year and also helps to see more expenses made. Helps to define their own categories.
- NFR-4 Performance The types of expense are categories along with an option .Throughput of the system is increased due to light weight database support.
- NFR-5 Availability Able to track business expense and monitor important for maintaining healthy cash flow. NFR-6 Scalability The ability to appropriately handle increasing demands.

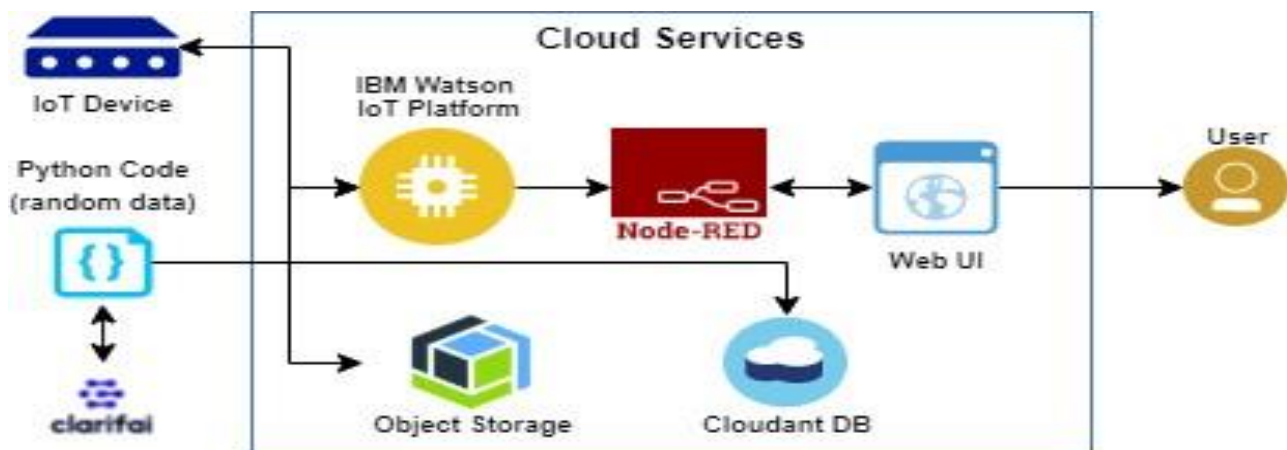
CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAM



5.2 Technical Architecture



5.3 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 | 6 | medium |

| | | | | | |
|----------|--|------|---|---|------|
| | | | sensor data such as temperature, moisture, soil and humidity to the IBM IoT platform | | |
| 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 |

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---------|-------------------------------|-------------------|---|--------------|----------|----------------------------|
| Sprint1 | SensorData (python script) | USN-1 | The Data of sensor which are feed to the Raspberrypi. Here we are using python script to generate a random sensor data. | 3 | High | Jeya Surya (Teamleader) |
| Sprint1 | Automation (python script) | USN-2 | Some activities are made to automation to overcome insufficient of labour force in the field. Hence that also included in python script to implement automation . | 5 | High | Vigneshwaran (Team Member) |
| Sprint2 | IBM IOT platform | USN-3 | To send the raspberrypi data to IOT platform, we create an IBM IOT platform and connect the raspberrypi to the device created in IBM IOT. | 5 | High | Logamagesh (Team Member) |
| Sprint3 | Node RED service | USN-4 | To access the IBM IOT platform from external application or from external UI Node red service is established. | 5 | High | Vigneshwaran (Team Member) |

| | | | | | | |
|---------|------------------|-------|--|---|------|--|
| Sprint3 | API Key | USN-5 | To protect the IBM IOT platform creating an API Key. | | High | Damodharan (Team Member) |
| Sprint4 | User Application | USN-6 | To monitor and control the field sensors the User is provided with an User application created by MIT app inventor | 8 | High | Jeya Surya (Team Leader) Logamagesh (Team Member) |

6.2 SPRINT DELIVERY SCHEDULE

| 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 | 8 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 8 | 29 Oct 2022 |
| Sprint-2 | 5 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 5 | 05 Nov 2022 |
| Sprint-3 | 8 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 8 | 12 Nov 2022 |
| Sprint-4 | 8 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 8 | 19 Nov 2022 |

6.3 REPORT FROM JIRA

REQUIRED SOFTWARE

- CLARIFAI
- IBM WATSON IOT PLATFORM
- PYTHON IDLE

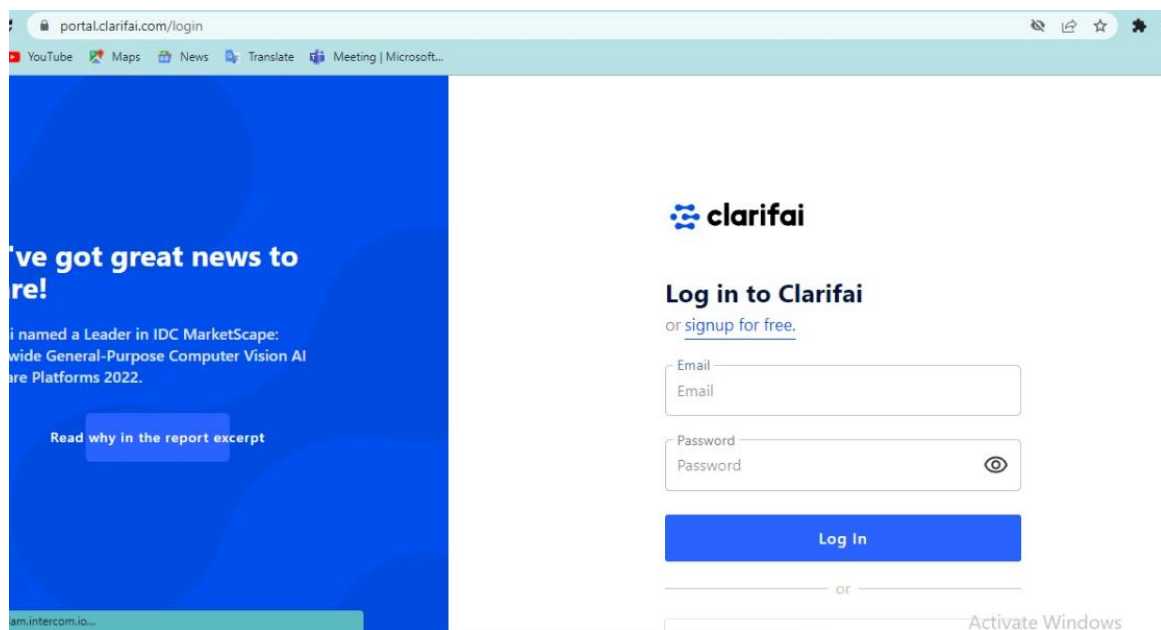
- **NODERED**
- **MITAPPINVENTOR**

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.

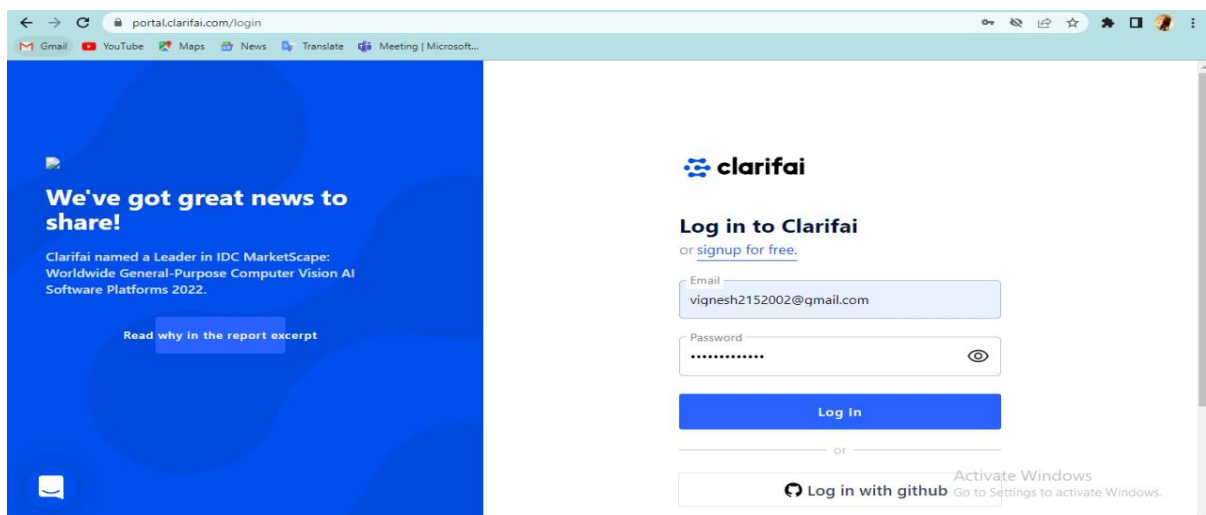
STEP 1:

- Open Clarifai portal in web browser.



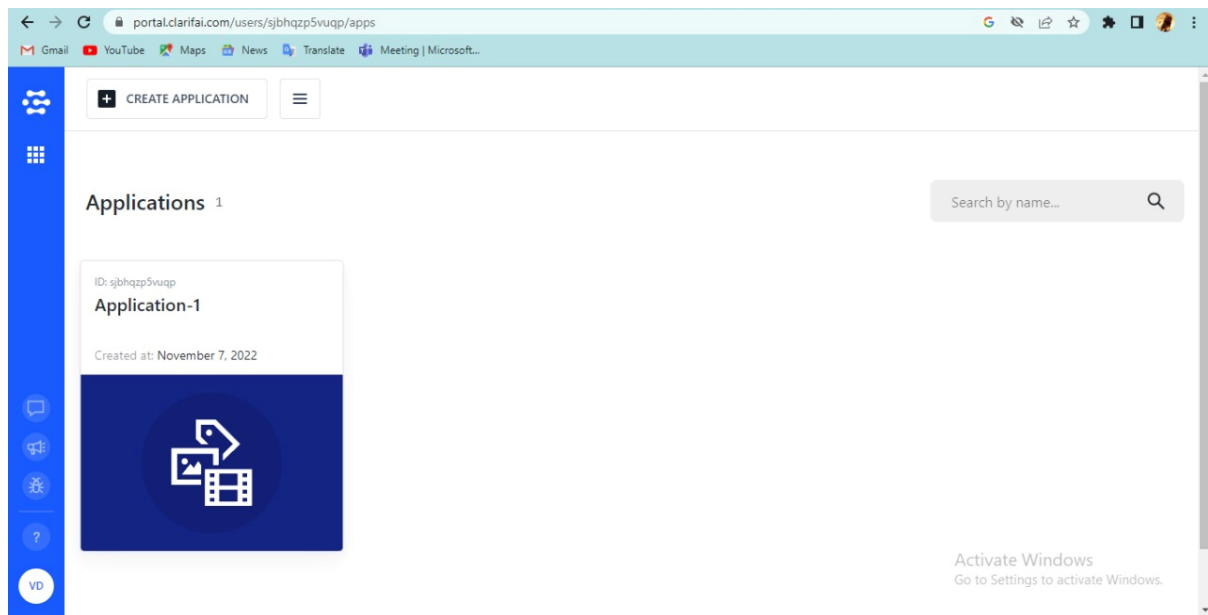
STEP 2:

- Signup using the required user mail and password.



STEP 3:

- Finally, Created an account



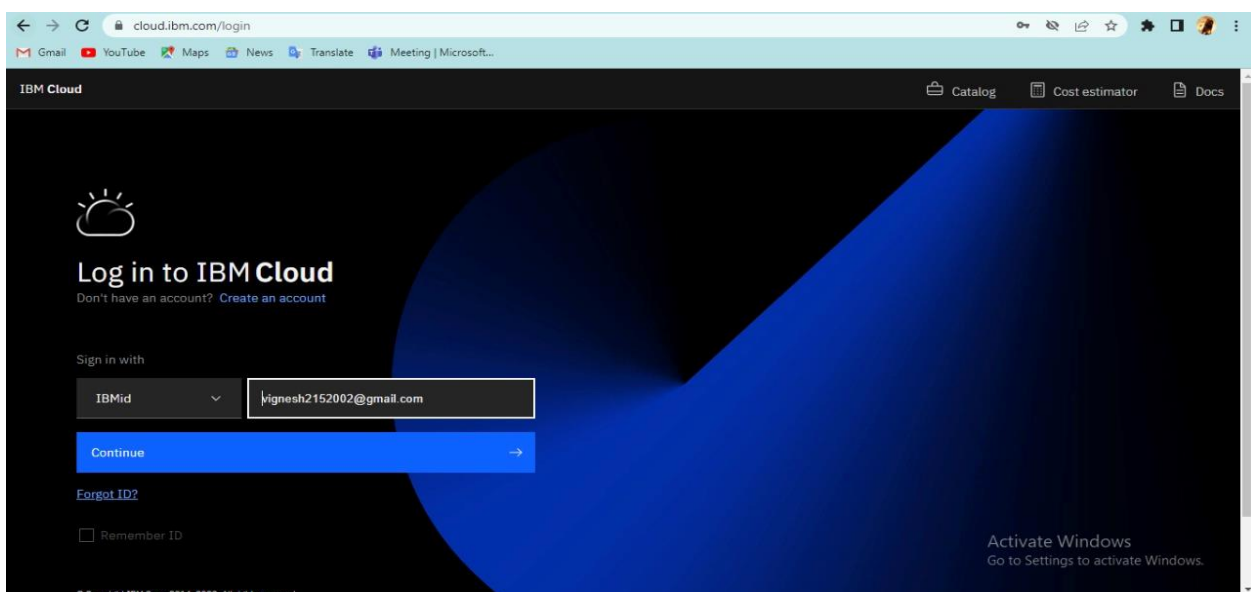
IBM WATSON IoT PLATFORM:

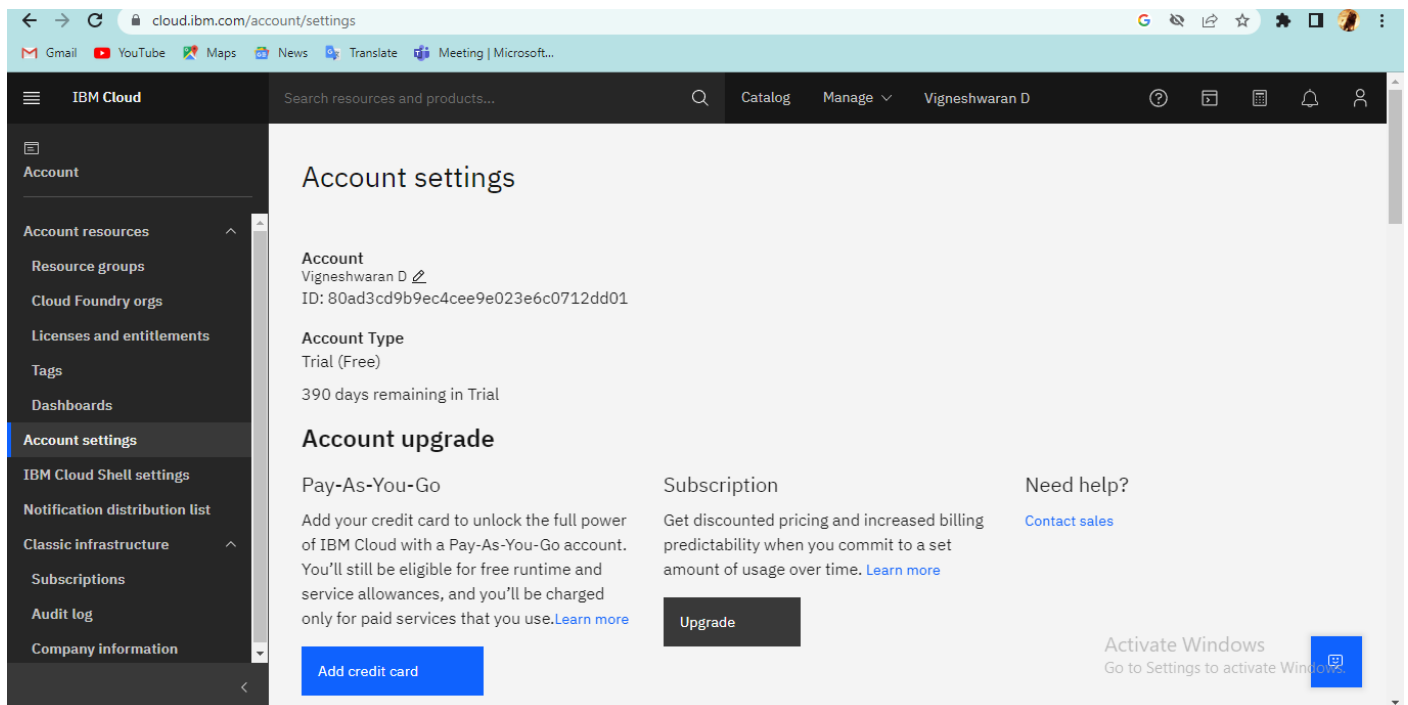
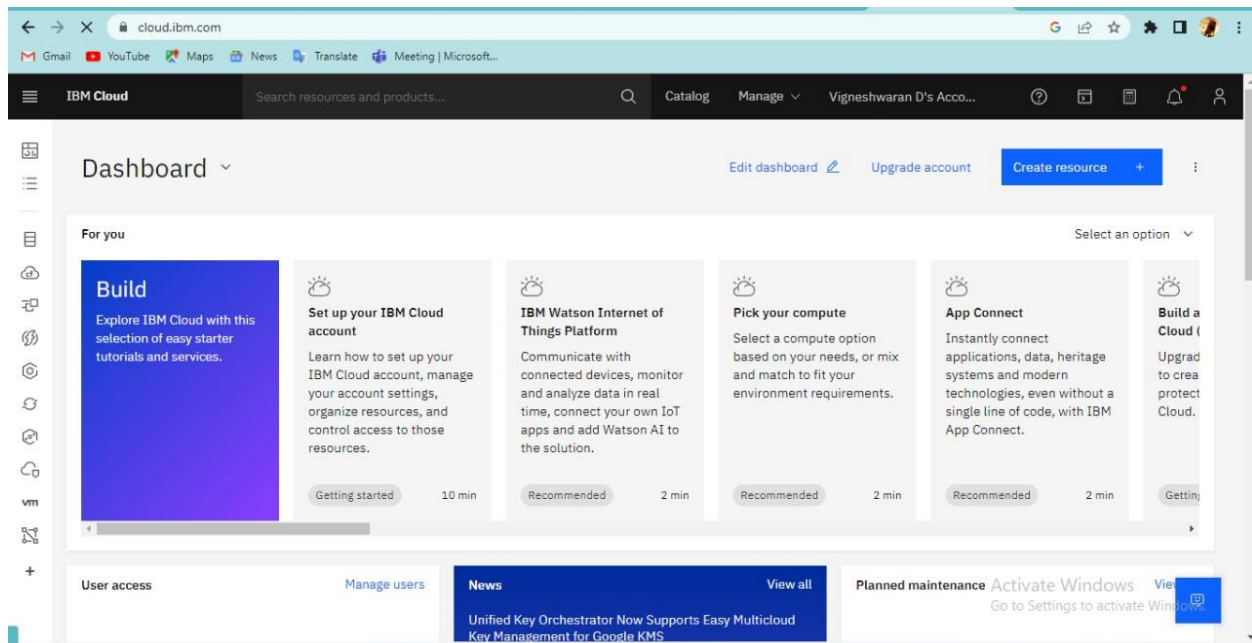
We need to have basic knowledge of the following cloud services:

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

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

LOGIN:



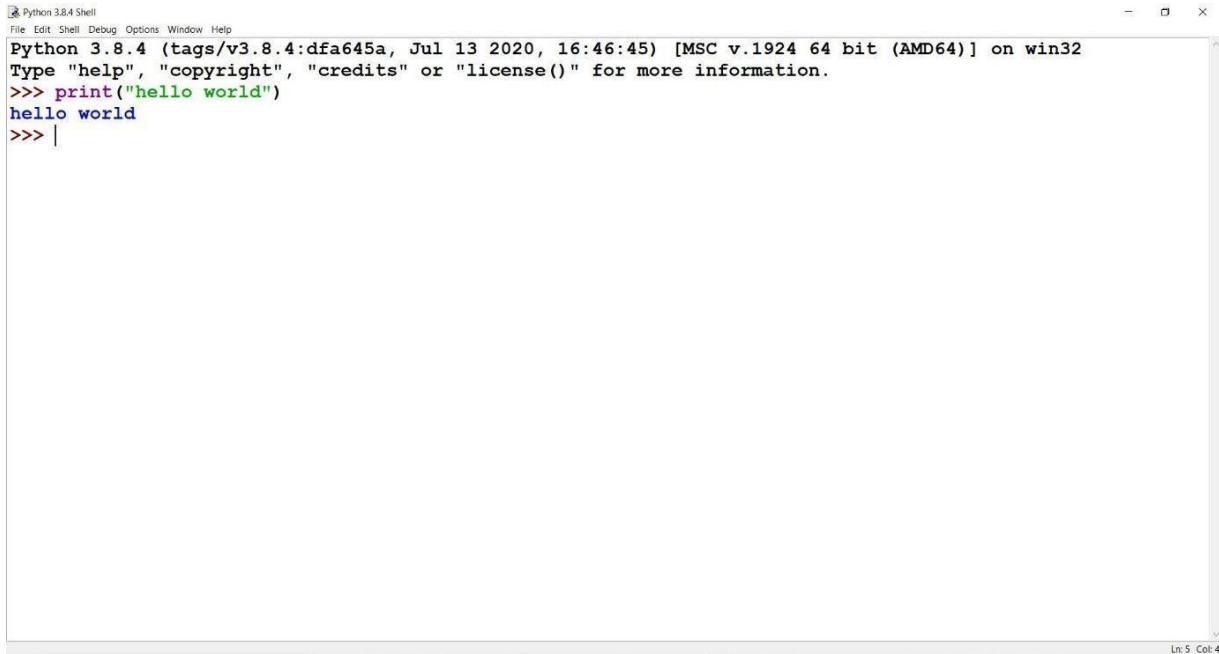


PYTHON IDLE INSTALISATION

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

STEP 1:

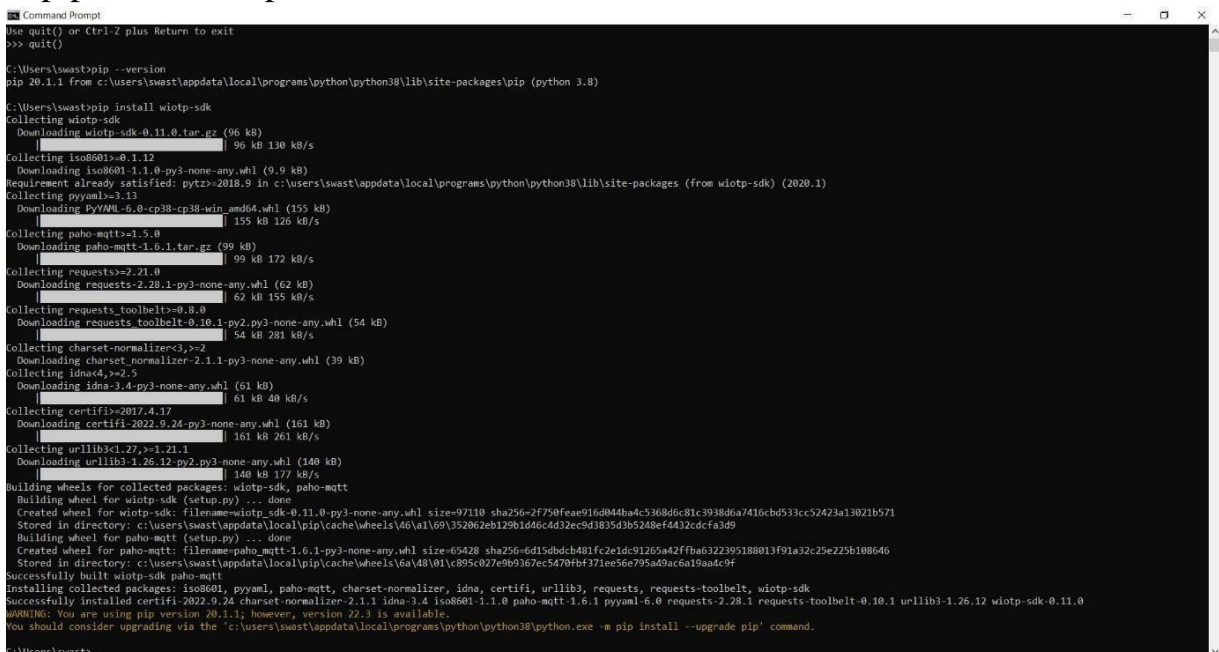
- Python is installed successfully



```
Python 3.8.4 Shell
File Edit Shell Debug Options Window Help
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:46:45) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("hello world")
hello world
>>> |
```

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



```
Command Prompt
Use quit() or Ctrl-Z plus Return to exit
>>> quit()

C:\Users\swast>pip --version
pip 20.1.1 from c:\users\swast\appdata\local\programs\python\python38\lib\site-packages\pip (python 3.8)

C:\Users\swast>pip install wiotp-sdk
Collecting wiotp-sdk
  Downloading wiotp-sdk-0.11.0.tar.gz (96 kB)
    |#####| 96 kB 130 kB/s
Collecting iso8601>=0.1.12
  Downloading iso8601-1.1.0-py3-none-any.whl (9.9 kB)
Requirement already satisfied: pytz>=2018.9 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from wiotp-sdk) (2020.1)
Collecting pyyaml>=3.13
  Downloading PyYAML-6.0-cp38-win_amd64.whl (155 kB)
    |#####| 155 kB 126 kB/s
Collecting paho-mqtt>=1.5.0
  Downloading paho-mqtt-1.6.1.tar.gz (99 kB)
    |#####| 99 kB 172 kB/s
Collecting requests>=2.21.0
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
    |#####| 62 kB 155 kB/s
Collecting requests-toolbelt>=0.8.0
  Downloading requests_toolbelt-0.10.1-py2.py3-none-any.whl (54 kB)
    |#####| 54 kB 201 kB/s
Collecting charset-normalizer<3,>=2
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
    |#####| 61 kB 40 kB/s
Collecting certifi>=2017.4.17
  Downloading certifi-2022.9.24-py3-none-any.whl (161 kB)
    |#####| 161 kB 261 kB/s
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.12-py2.py3-none-any.whl (140 kB)
    |#####| 140 kB 177 kB/s
Building wheels for collected packages: wiotp-sdk, paho-mqtt
  Building wheel for wiotp-sdk (setup.py) ... done
  Created wheel for wiotp-sdk: filename=wiotp_sdk-0.11.0-py3-none-any.whl size=97110 sha256=2f759fae916d044ba4c5368d6c81c3938d6a7416cbd533cc52423a13021b571
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\46\val\09\352062eb129b1d66cad32ec9d3835d3b5248ef4432cdcf3d9
  Building wheel for paho-mqtt (setup.py) ... done
  Created wheel for paho-mqtt: filename=paho_mqtt-1.6.1-py3-none-any.whl size=65428 sha256=6d15dbdc481fc2e1dc91265a42ffba6322395188013f91a32c25e225b108646
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\6a\d8\01\c895c027e9b9367ec5470fbf371ee56e795ad9ac6a19aadcf9f
Successfully built wiotp-sdk paho-mqtt
Installing collected packages: iso8601, pyyaml, paho-mqtt, charset-normalizer, idna, certifi, urllib3, requests, requests-toolbelt, wiotp-sdk
Successfully installed certifi-2022.9.24 charset-normalizer-2.1.1 idna-3.4 iso8601-1.1.0 paho-mqtt-1.6.1 pyyaml-6.0 requests-2.28.1 requests-toolbelt-0.10.1 urllib3-1.26.12 wiotp-sdk-0.11.0
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

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

```
Command Prompt
C:\Users\swast>pip install --upgrade "ibm-watson>=5.0.0
Collecting ibm-watson>=5.0.0
  Downloading ibm-watson-6.1.0.tar.gz (373 kB)
    Installing build dependencies ... done
    Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Collecting ibm-cloud-sdk-core==3.*>=3.3.6
  Downloading ibm-cloud-sdk-core-3.16.0-py3-none-any.whl (83 kB)
Requirement already satisfied, skipping upgrade: requests<3.0,>=2.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-watson>=5.0.0) (2.28.1)
Collecting websocket-client==1.1.0
  Downloading websocket-client-1.1.0-py2.py3-none-any.whl (68 kB)
Requirement already satisfied, skipping upgrade: python-dateutil<=2.5.3 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-watson>=5.0.0) (2.8.1)
Collecting PyJWT<3.0.0,>=2.4.0
  Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB)
Requirement already satisfied, skipping upgrade: six>=1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cloud-sdk-core==3.*>=3.3.6>ibm-watson>=5.0.0) (1.26.0)
Requirement already satisfied, skipping upgrade: certifi<=2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.0>ibm-watson>=5.0.0) (2022.9.24)
Requirement already satisfied, skipping upgrade: idna<=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.0>ibm-watson>=5.0.0) (3.4)
Requirement already satisfied, skipping upgrade: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.0>ibm-watson>=5.0.0) (2.1.1)
Requirement already satisfied, skipping upgrade: urllib3<2.0.0,>=1.26.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<=2.5.3>ibm-watson>=5.0.0) (1.26.0)
Building wheels for collected packages: ibm-watson
  Building wheel for ibm-watson (PEP 517) ... done
  Created wheel for ibm-watson: filename=ibm_watson-6.1.0-py3-none-any.whl size=376748 sha256=56648b1c154ee0ba2a5cc521f68536cd77a9cf975fccc5f975bdf9ba6956
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\34\bd\cd\829a351c802b7a578115f67daedf62b9eae84e9882c7e2
Successfully built ibm-watson
Installing collected packages: PyJWT, ibm-cloud-sdk-core, websocket-client, ibm-watson
Successfully installed PyJWT-2.6.0 ibm-cloud-sdk-core-3.16.0 ibm-watson-6.1.0 websocket-client-1.1.0
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

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

```
Command Prompt
C:\Users\swast>pip install ibm-cos-sdk
Collecting ibm-cos-sdk
  Downloading ibm-cos-sdk-2.12.0.tar.gz (55 kB)
    Collecting ibm-cos-sdk-core==2.12.0
      Downloading ibm-cos-sdk-core-2.12.0.tar.gz (956 kB)
        Collecting ibm-cos-sdk-s3transfer==2.12.0
          Downloading ibm-cos-sdk-s3transfer-2.12.0.tar.gz (135 kB)
            Collecting jmespath<1.0.0,>=0.10.0
              Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
            Collecting python-dateutil<3.0.0,>=2.8.2
              Downloading python-dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Requirement already satisfied: requests<3.0,>=2.27.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (2.28.1)
Requirement already satisfied: urllib3<1.27,>=1.26.9 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (1.26.12)
Requirement already satisfied: six>=1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<3.0.0,>=2.8.2>ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (1.15.0)
Requirement already satisfied: certifi<=2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1>ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (2022.9.24)
Requirement already satisfied, skipping upgrade: idna<=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1>ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (3.4)
Requirement already satisfied, skipping upgrade: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1>ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (2.1.1)
Building wheels for collected packages: ibm-cos-sdk-core, ibm-cos-sdk-s3transfer
  Building wheel for ibm-cos-sdk-core (setup.py) ... done
  Created wheel for ibm-cos-sdk-core: filename=ibm_cos_sdk_core-2.12.0-py3-none-any.whl size=73926 sha256=a6f6caad736b09209e285e7f6e185c5bfa4721a71f35188f94c734e81cd36e
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\21\5f\fd\6a04bb45aad71bc0c8300814368f9d39ef7c4fd1869d2d44d
  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ... done
  Created wheel for ibm-cos-sdk-s3transfer: filename=ibm_cos_sdk_s3transfer-2.12.0-py3-none-any.whl size=52992 sha256=c7f8e89dee7511d484073c5082533731d8715bad59fb3deda4ad0d38a1f99d7
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\ca\3d\78\48c57e574477898f3acc81df480b1de0a8cbfc69d56bee7
  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ... done
  Created wheel for ibm-cos-sdk-s3transfer: filename=ibm_cos_sdk_s3transfer-2.12.0-py3-none-any.whl size=89769 sha256=67c5983e4abdb0b13db07c81d35d7216ebef83fec9e5f0275d9fe8e51ceb77
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\c0\7a\37\13b5ca7427a29a1062a47c58baa1c2ff3832795b68c8bd4d6
Successfully built ibm-cos-sdk-core ibm-cos-sdk-s3transfer
Installing collected packages: jmespath, python-dateutil, ibm-cos-sdk-core, ibm-cos-sdk-s3transfer, ibm-cos-sdk
  Attempting uninstall: python-dateutil
    Found existing installation: python-dateutil 2.8.1
    Uninstalling python-dateutil-2.8.1:
      Successfully uninstalled python-dateutil-2.8.1
  Successfully installed ibm-cos-sdk-2.12.0 ibm-cos-sdk-core-2.12.0 ibm-cos-sdk-s3transfer-2.12.0 jmespath-0.10.0 python-dateutil-2.8.2
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- pip install -U ibm-cos-sdk

```
Command Prompt
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>pip install -U ibm-cos-sdk
Requirement already up-to-date: ibm-cos-sdk in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (2.12.0)
Requirement already satisfied, skipping upgrade: ibm-cos-sdk-s3transfer==2.12.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (2.12.0)
Requirement already satisfied, skipping upgrade: ibm-cos-sdk-core==2.12.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (2.12.0)
Requirement already satisfied, skipping upgrade: jmespath<1.0.0,>=0.10.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk) (0.10.0)
Requirement already satisfied, skipping upgrade: requests<3.0,>=2.27.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (2.28.1)
Requirement already satisfied, skipping upgrade: urllib3<1.27,>=1.26.9 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (1.26.12)
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.8.2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (2.8.2)
Requirement already satisfied, skipping upgrade: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1>ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (2.1.1)
Requirement already satisfied, skipping upgrade: idna<=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1>ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (3.4)
Requirement already satisfied, skipping upgrade: certifi<=2017.4.17 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0,>=2.27.1>ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (2022.9.24)
Requirement already satisfied, skipping upgrade: six>=1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<3.0.0,>=2.8.2>ibm-cos-sdk-core==2.12.0>ibm-cos-sdk) (1.15.0)
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- pip install boto3

```
Command Prompt
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>pip install boto3
Collecting boto3
  Downloading boto3-1.26.0-py3-none-any.whl (132 kB)
    | 132 kB 148 kB/s
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)
    | 79 kB 113 kB/s
Collecting botocore<1.30.0,>=1.29.0
  Downloading botocore-1.29.0-py3-none-any.whl (9.8 MB)
    | 9.8 MB 2.2 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from boto3) (0.10.0)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from botocore<1.30.0,>=1.29.0->boto3) (1.26.12)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from botocore<1.30.0,>=1.29.0->boto3) (2.8.2)
Requirement already satisfied: six<1.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.30.0,>=1.29.0->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.26.0 botocore-1.29.0 s3transfer-0.6.0
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- pip install resources

```
Command Prompt
C:\Users\swast>pip install resources
Collecting resources
  Downloading resources-0.0.1.tar.gz (3.7 kB)
Building wheels for collected packages: resources
  Building wheel for resources (setup.py) ... done
  Created wheel for resources: filename=resources-0.0.1-py3-none-any.whl size=4370 sha256=38113eb3ac96cb54f0f22303a68ae65aac976211e26ae94f9b2441ec31be
  Stored in directory: c:\users\swast\appdata\local\pip\cache\wheels\b3\1d\00\45ae97c7b92d145a0963f711c6d22f9af530e74c8bf2f28fd
Successfully built resources
Installing collected packages: resources
Successfully installed resources-0.0.1
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```

- pip install cloudant

```
Command Prompt
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>pip install cloudant
Collecting cloudant
  Downloading cloudant-2.15.0-py3-none-any.whl (80 kB)
    | 80 kB 305 kB/s
Requirement already satisfied: requests<3.0.0,>=2.7.0 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from cloudant) (2.28.1)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0.0,>=2.7.0->cloudant) (2.1.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0.0,>=2.7.0->cloudant) (1.26.12)
Requirement already satisfied: certifi<=2017.4.7 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0.0,>=2.7.0->cloudant) (2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in c:\users\swast\appdata\local\programs\python\python38\lib\site-packages (from requests<3.0.0,>=2.7.0->cloudant) (3.4)
Installing collected packages: cloudant
Successfully installed cloudant-2.15.0
WARNING: You are using pip version 20.1.1; however, version 22.3 is available.
You should consider upgrading via the 'c:\users\swast\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\Users\swast>
```


PROJECT DEVELOPMENT

STEP 1: Write a python code for randomize Soil Moisture ,Temperature, Humidity and Animal detection.

```
ooo.py - C:/Users/DELL/AppData/Local/Programs/Python/Python37/ooo.py (3.7.0)
File Edit Format Run Options Window Help

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "iirt5j7"
deviceType = "abcd"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    elif status == "lightoff":
        print ("led is off")
    else :
        print ("please send proper command")

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

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

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times
deviceCli.connect()

while True:
    #Get Sensor Data from DHT11
```

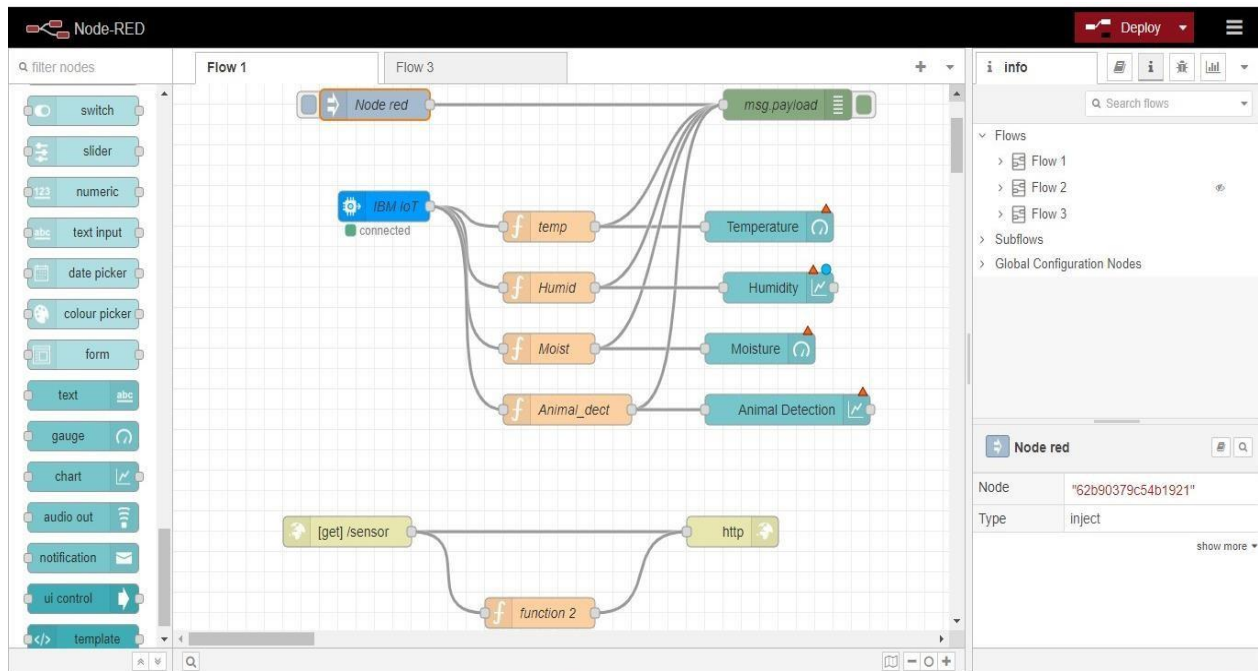
STEP 2: Run the python code it send data to IBM IoT Watson Platform.

The screenshot displays the IBM Watson IoT Platform interface. On the left, a sidebar contains navigation icons. The main area shows a device configuration page for 'Device Type: abcd' with ID '12345'. The 'Recent Events' tab is active, showing a table of events. A modal window is open for configuring an event type named 'event_1'. The 'Schedule' is set to 'Every Minute' with a delay of '20'. The 'Payload' is defined as a JSON object with random values for temperature, humidity, moisture, and animal detection.

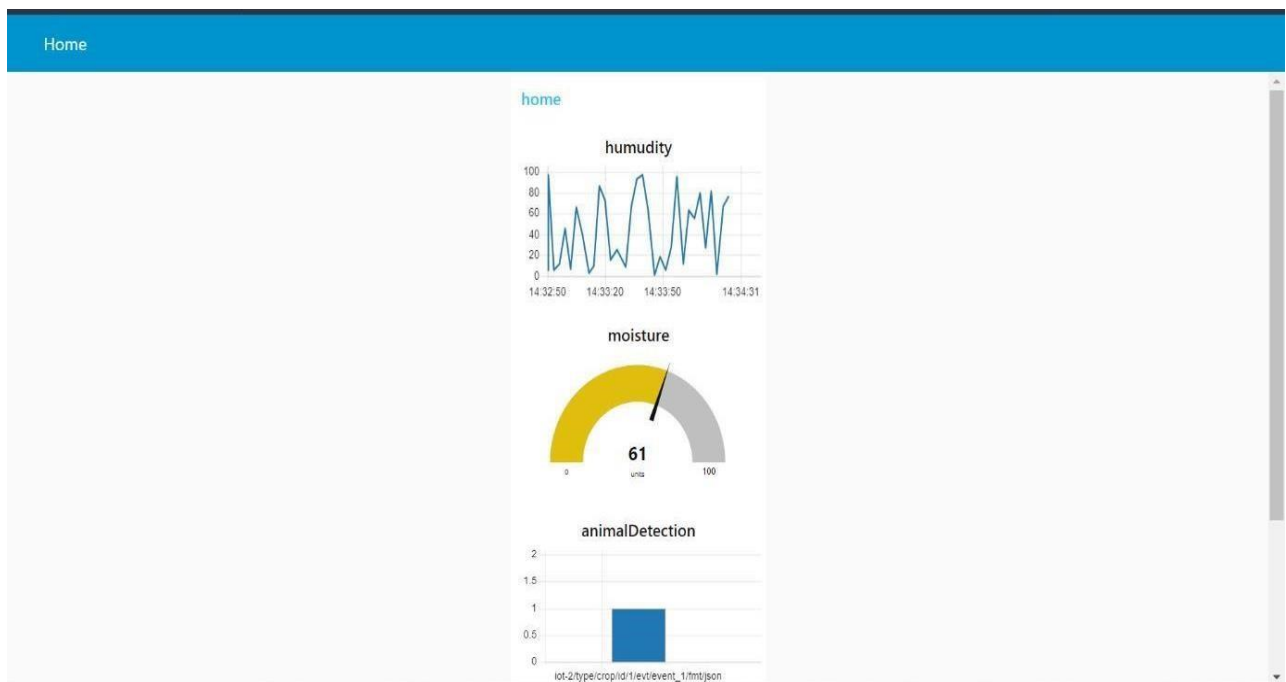
| Event | Value |
|-----------|---|
| IoTSensor | {"temp":91,"Humid":67,"Moist":25,"Animal_dect..."} |
| IoTSensor | {"temp":102,"Humid":78,"Moist":92,"Animal_dect..."} |
| IoTSensor | {"temp":106,"Humid":69,"Moist":25,"Animal_dect..."} |
| IoTSensor | {"temp":92,"Humid":79,"Moist":82,"Animal_dect..."} |

```
{
  "temp": random(90,110)
  "Humid": random(60,100)
  "Moist": random(0,100)
  "Animal_dect": random(0,2)
}
```

STEP 3: Open Node-RED flow dashboard.



STEP 4: Open Node-RED user interface to show the Soil Moisture, Humidity and Temperature value in gauge.



CHAPTER 7

CODING AND SOLUTIONS

7.1 FEATURE

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

Source Code:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device
Credentialsorganization = "wu5b55"
deviceType = "crop1"
deviceId = "1234"
authMethod =
"token"
authToken = "1234567890"

# Initialize GPIOtry:

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

except Exception as e:

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

# Connect and send a datapoint "hello" with value "world" into the cloud as
an event of type "greeting" 10 times
deviceCli.connect()
while True:

        temp=random.randint(0,
100)
        Hum=random.randint(0,1
00)
        moisture=random.randint
```



```

(0,100)
data = { 'temperature' : temp, 'Humidity': Hum, 'Moisture':moisture }
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 IoT")time.sleep(10)
    deviceCli.commandCallback = myCommandCallback
# Disconnect the device and application from the
clouddeviceCli.disconnect()

```

7.2 FEATURE 2

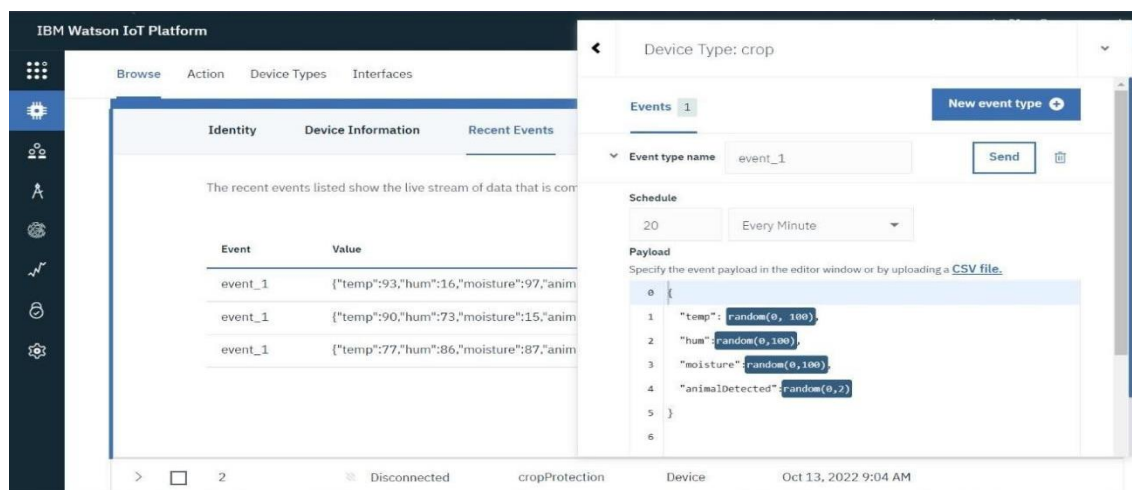
Source code is deployed on IBM Watson IoT platform to generate sensor data.SourceCode:

```

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

```

Output:



7.3 DATABASE SCHEMA

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()
```

CHAPTER 8 TESTING

8.1 TEST CASES

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 | 10 | 4 | 2 | 3 | 20 |
| Duplicate | 1 | 0 | 3 | 0 | 4 |
| External | 2 | 3 | 0 | 1 | 6 |
| Fixed | 11 | 2 | 4 | 20 | 37 |
| Not Reproduced | 0 | 0 | 1 | 0 | 1 |
| Skipped | 0 | 0 | 1 | 1 | 2 |
| Won't Fix | 0 | 5 | 2 | 1 | 8 |
| Totals | 24 | 14 | 13 | 26 | 77 |

8.2 USER ACCEPTANCE TESTING

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

| Section | Total Cases | Not Tested | Fail | Pass |
|---------------------|-------------|------------|------|------|
| Print Engine | 7 | 0 | 0 | 7 |
| Client Application | 51 | 0 | 0 | 51 |
| Security | 2 | 0 | 0 | 2 |
| Outsource Shipping | 3 | 0 | 0 | 3 |
| Exception Reporting | 9 | 0 | 0 | 9 |
| Final Report Output | 4 | 0 | 0 | 4 |
| Version Control | 2 | 0 | 0 | 2 |

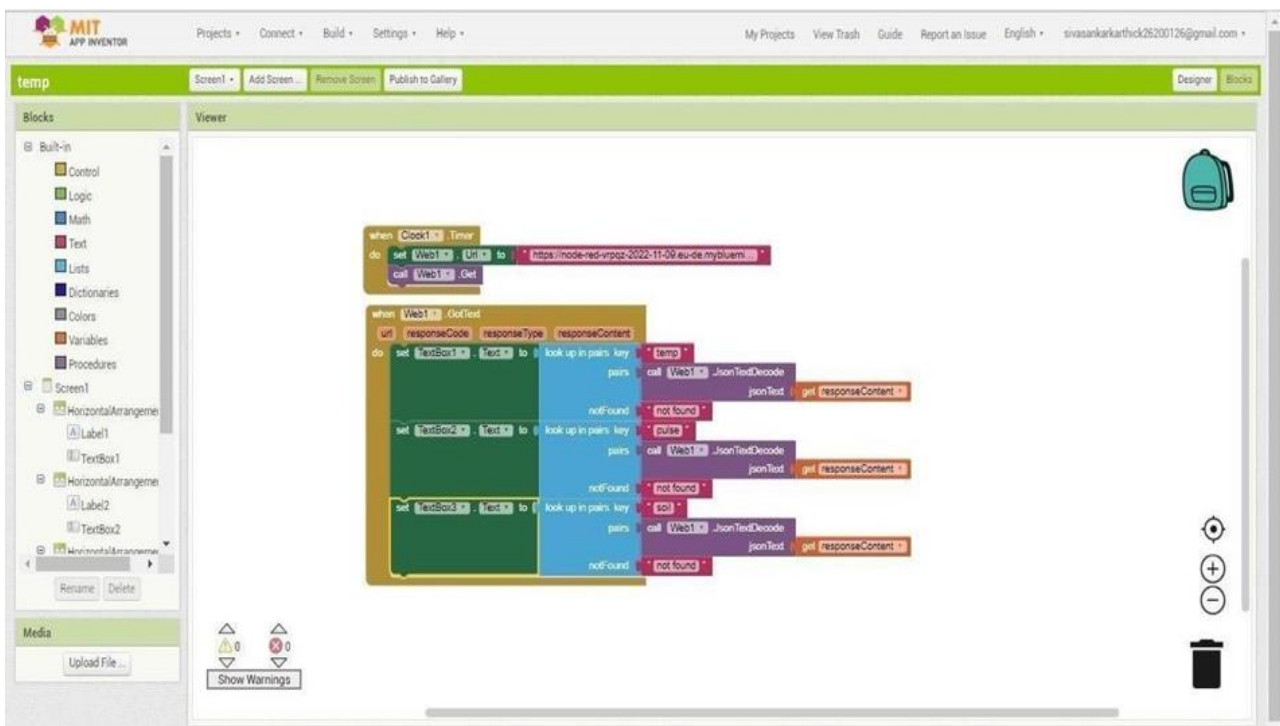
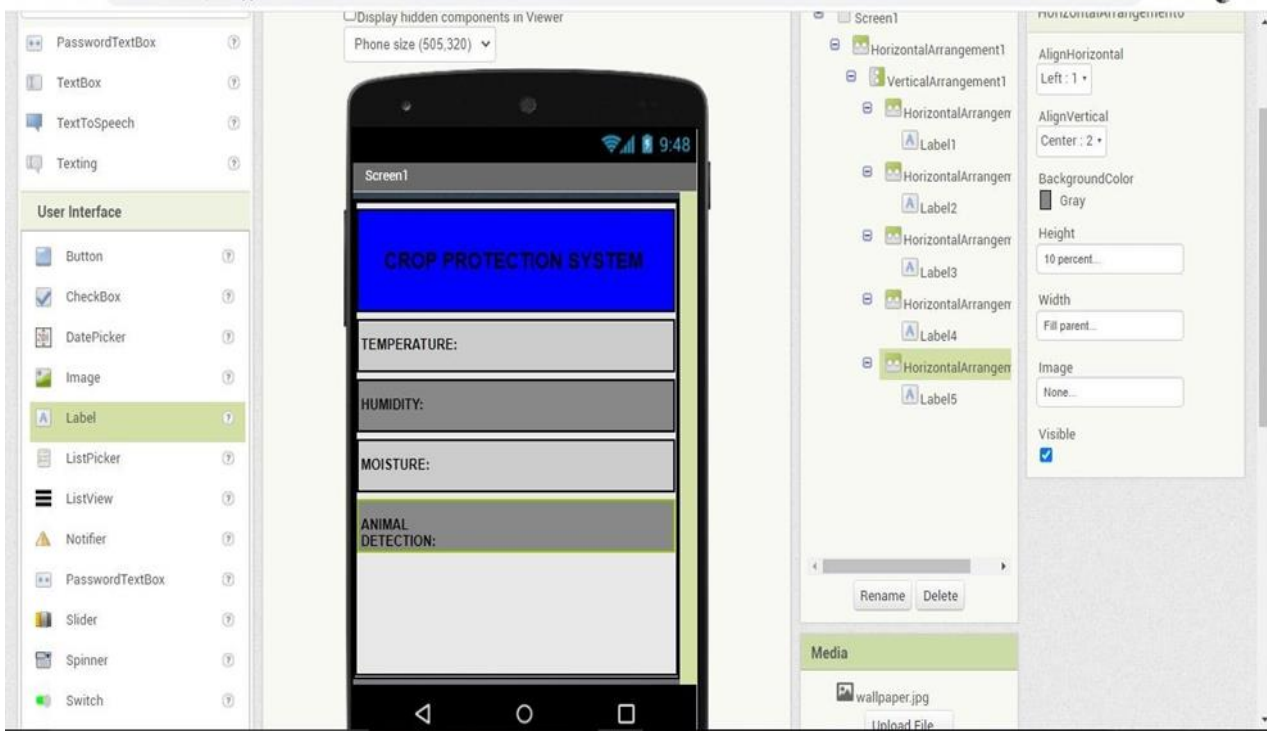
CHAPTER 9

RESULT

9.1 PERFORMANCE METRICS

MIT APP INVENTOR:

STEP 1: MIT APP inventor to design the APP.



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



CHAPTER 10

ADVANTAGES AND DISADVANTAGES

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)

CHAPTER 11

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.

CHAPTER 12

FUTURE SCOPE

Study and analysis of the developed Crop protection systems for its costeffectiveness with the development of Arduino based variable frequency Ultrasonic birddeterrent circuit. outline of the crop damage caused by a particular Wild animal if thebehavioral features of the With the reduced cost in the smart phones.

CHAPTER 13

APPENDIX

13.1SOURCE CODE

The source code has been uploaded in github. To refer the final source code click [“SOURCE CODE”](#)

13.2 GITHUB & PROJECT DEMO LINK

GITHUB LINK

The github link :["GITHUB LINK"](#)

PROJECT DEMO LINK

The Project Demo link: ["DEMO LINK"](#)

CHAPTER 14

REFERENCE

[1]Priyanka Deotale, Prasad Lokulwar (2021) have presented the paper titled “Smart Crop Protection System from Wild animals Using IoT” . Crops in the agricultural land are destroyed by the domestic animals and wild animals ,it is one of the reason for low productivity . Farmers can't be there for entire 2 hours so we have make use of IOT to control the animals destroying the field . Once the animal is detected the system will larm and start lightning in the corner of the farm . It will not harm any animals and we can also protect the crops.

[2] N.S. Gogul Dev , K.S. Sreenesh , P.K. Binu (2019) has presented a paper titled “IoT Based Automated Crop Protection System” . Low productivity of crops is one of the main problems faced by the farmers in our country. This can be because of two main reasons. Crops destroyed by wild animals and because of bad weather condition. This paper provides a solution to the destruction of crops by animals. This system will provide a complete technical solution using the Internet of things (IOT) to the farmers to prevent their crops from wild animals and provide information to the farmers to maximize their production. Animals are detected using PIR sensors and cameras where animals are identified using TensorFlow image processing Techniques. Raspberry PI is used as the processing unit of the system and sound buzzers are used to emit the ultrasound frequencies