Project Report Format

1. **INTRODUCTION**
    - 1.1 Project Overview
    - 1.2 Purpose
2. **LITERATURE SURVEY**
    - 2.1 Existing problem
    - 2.2 References
    - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    - 3.1 Empathy Map Canvas
    - 3.2 Ideation & Brainstorming
    - 3.3 Proposed Solution
    - 3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    - 4.1 Functional requirement
    - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
    - 5.1 Data Flow Diagrams
    - 5.2 Solution & Technical Architecture
    - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
    - 6.1 Sprint Planning & Estimation
    - 6.2 Sprint Delivery Schedule
    - 6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    - 7.1 Feature 1
    - 7.2 Feature 2
    - 7.3 Database Schema (if Applicable)
8. **TESTING**
    - 8.1 Test Cases
    - 8.2 User Acceptance Testing
9. **RESULTS**
    - 9.1 Performance Metrics
10. **ADVANTAGES &DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**
    - 13.1 Source Code
    - 13.2 GitHub & Project Demo Link

1. INTRODUCTION

1.1 PROJECT OVERVIEW

In many households, people carelessly leave the appliances and equipment on for too long when not in use. Also, electronic devices are poorly maintained.This could cause a fire. Sometimes it is not possible for the fire department to reach the destination on time. So here we propose an Industry-Specific Fire Management System that is used to detect fires in households.

1.2 PURPOSE

In our current generation people are much more careless. So, fire incidentshave become a common happening. Our project ensures that the fire is put offimmediately when a fire is detected. In case it fails to put off the fire, it would send a alert.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

It’s a huge challenge for the Fire department to reach on time to save the people. The fire Management didn’t have awareness to tackle all these stuffsand didn’t have the knowledge of current technologies also and cost of the application which is existing now was not at nominal cost.

2.2 REFERENCE

1) Paper 1: Fire Detection, Monitoring and AlertingSystem based on IOTPublished year: 2019

Author name: Shreya Gosrani, AbhishekJadhav, Krutika LekhakD ChhedaJ

Journal name: International journal of engineering applied science andtechnology

2) Paper 2: Fire Monitoring and Controlling System based on IOTPublished year: 2020

Authors: Nitin Galugade,Mahesh Jakkar, DevikaNair, Madhur Gawas

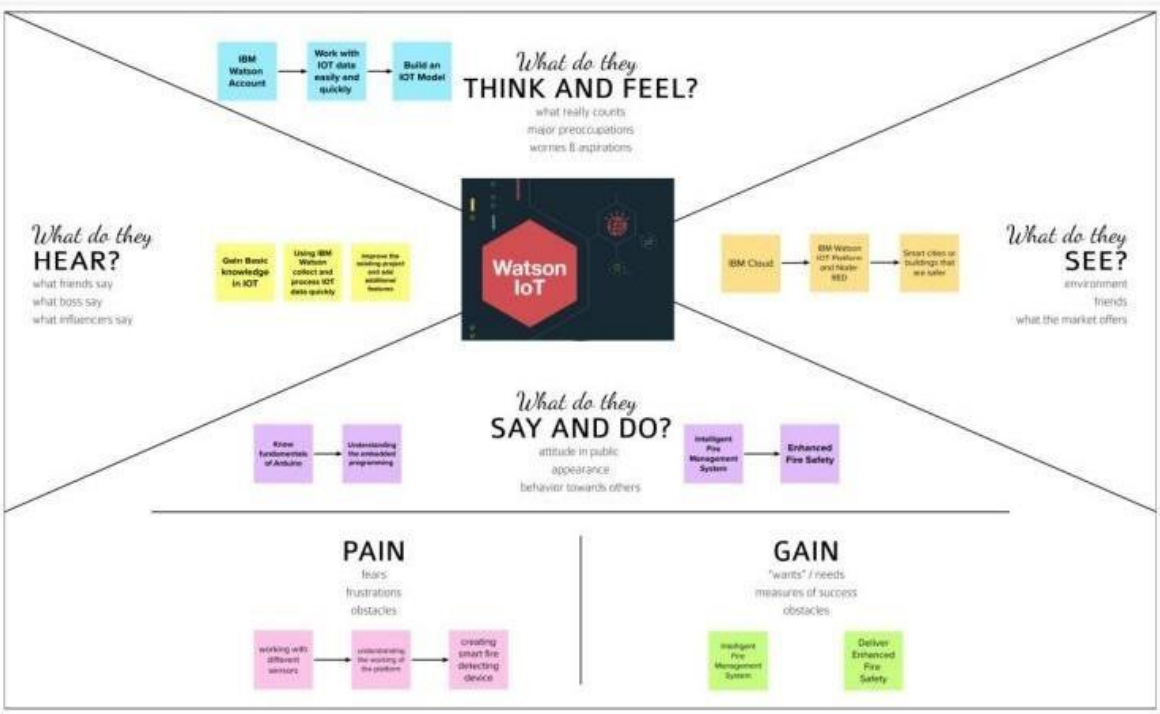
Journal name: International Journal of Engineering Research& Technology(IJERT)

2.3 PROBLEM STATEMENT DEFINITION

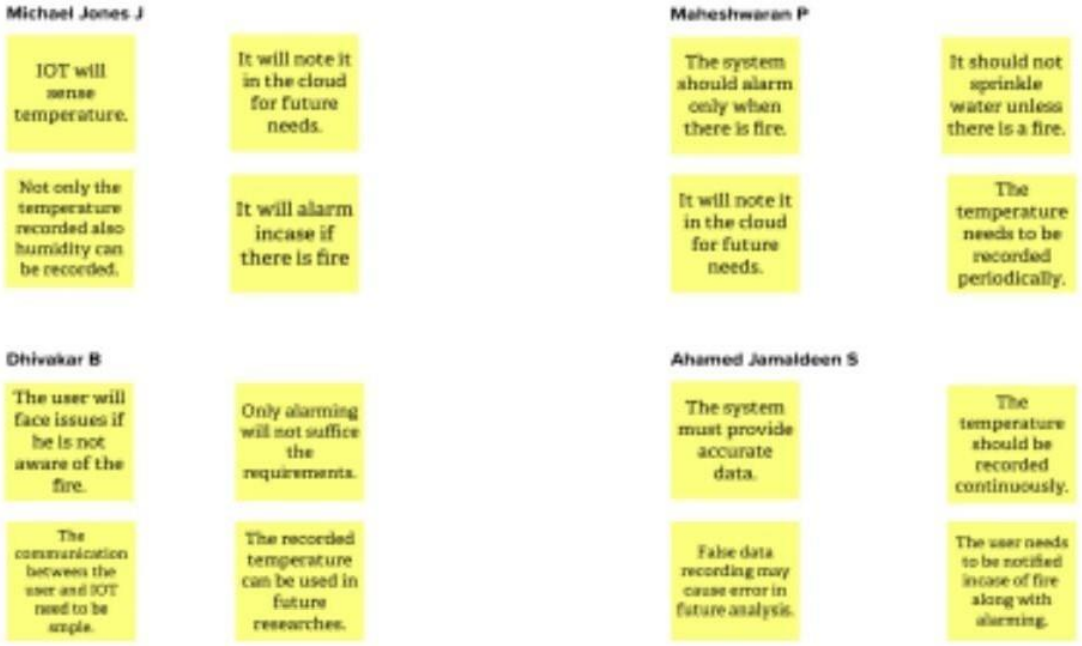
The smart fire management system includes a humidity sensor and temperature sensors, smoke sensor and flame sensor to detect any changes in the environment. Based on the temperature readings is high and humidity is low, the alarm and the sprinklerswill be switched on manually through web.

3. IDEATION & PROPOSEDSOLUTION

3.1 EMPATHY MAP CANVAS



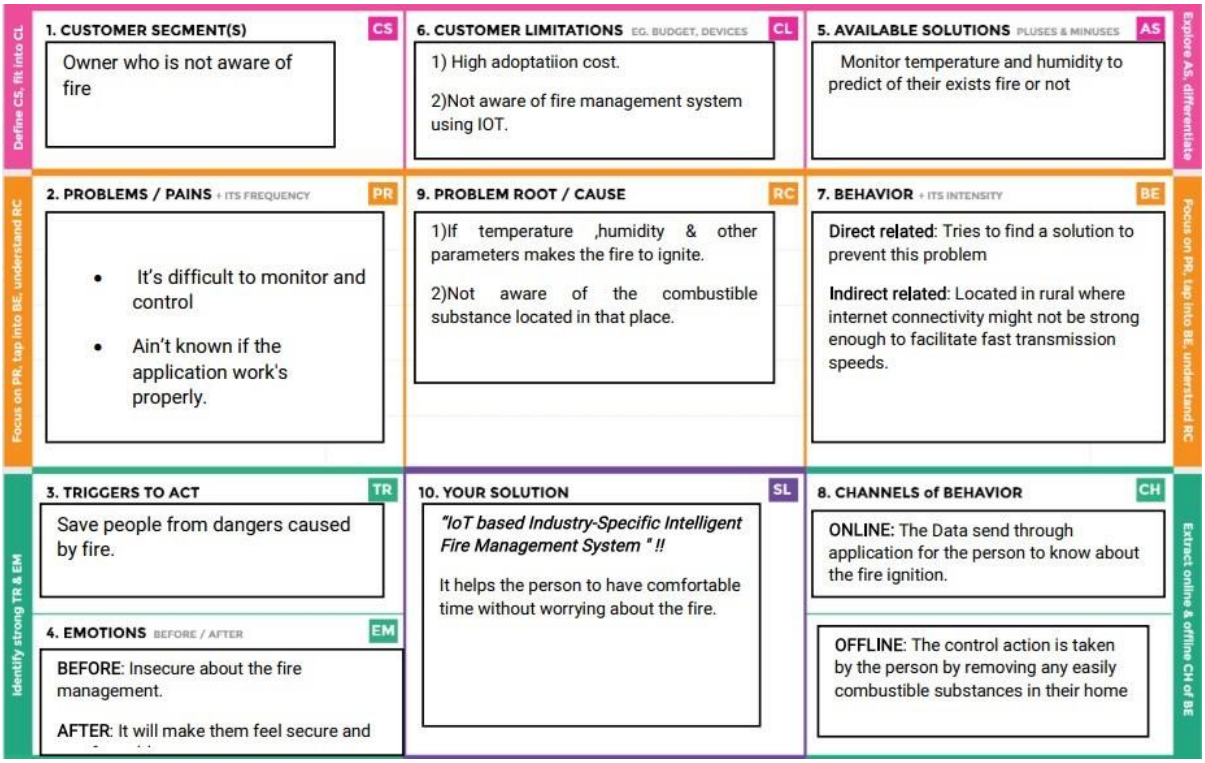
3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Industry-Specific Intelligent Fire Management System
2.	Idea / Solution description	Sense the temperature and humidity in the room periodically. If it exceeds the threshold ring alarm and sprinkle water and notify the user.
3.	Novelty / Uniqueness	It will notify the user about the ignition.
4.	Social Impact / Customer Satisfaction	It will help in people to go out without worrying about the cause of fire because if occurred it will notify them.
5.	Business Model (Revenue Model)	The data collected can be used for analysis in the climatic researches.
6.	Scalability of the Solution	This model can able to handle many numbers of inputs and provide the respective outputs.

3.4 PROBLEM SOLUTION FIT



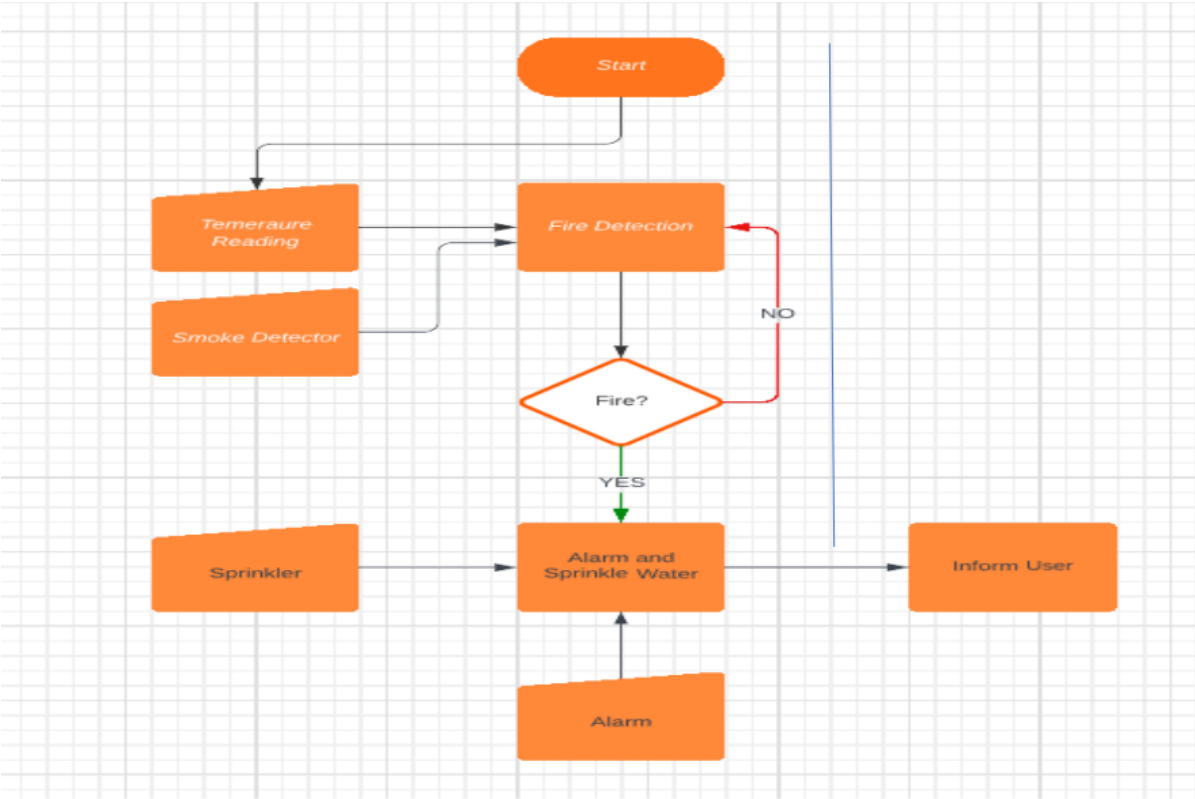
4.REQUIREMENT ANALYSIS
4.1 FUNCTIONAL REQUIREMENTS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail.
FR-2	User Visibility	Through notifications.
FR-3	User Reception	The data like temperature, smoke contented gas level is shown.
FR-4	User Understanding	Based on the data, the user understands that if any of the data is above the threshold value, then the place has caught in fire.
FR-5	User action	Incase of fire the sprinklers an alarm will be on. The user needs to escape and indicate the fire department.

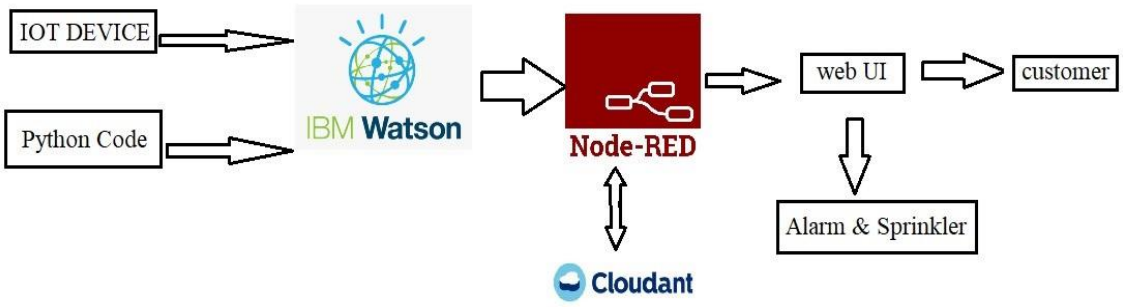
4.2 NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It needs to observe all time and give alert incase of fire.
NFR-2	**Security**	It helps user, incase of fire by notifying it.
NFR-3	**Reliability**	It does not give false alarm.
NFR-4	**Performance**	It will sprinkler water to smoother the fire in early stage.
NFR-5	**Availability**	It will always be checking if there is fire or not.
NFR-6	**Scalability**	The scalability depends on the options to change as needed in the board and sensors.

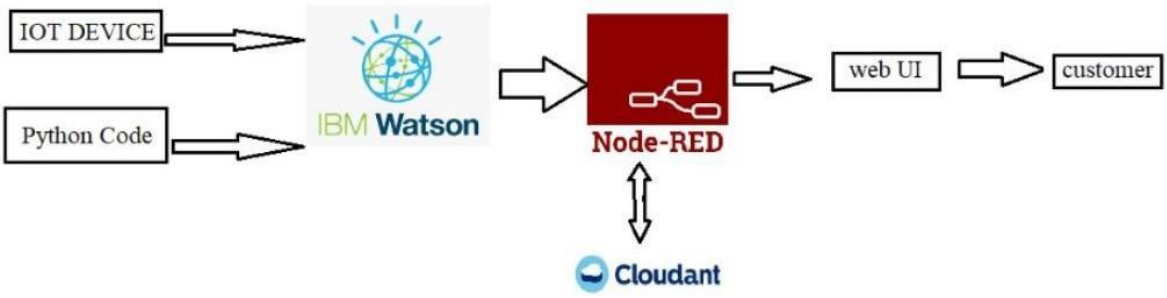
5.PROJECT DESIGN
5.1 DATA FLOW DIAGRAM



5.2 SOLUTION AND TECHNICAL ARCHITECTURESOLUTION ARCHITECTURE



TECHNICAL ARCHITECTURE



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can download the application.	I can view the data sent by the hardware.	High	Sprint-1
		USN-2	As a user, I can register to application with my e-mail.	I can access my profile.	High	Sprint-1
		USN-3	As a user, I will receive confirmation email or OTP to SMS once I have registered for the application.	I can receive confirmation email, click confirm.	High	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password.	I can access my profile and dashboard.	High	Sprint-2
Customer Care Executive	Action	USN-5	As a user, I can View Temperature Readings.	I can view the data by app.	Medium	Sprint-2
		USN-6	As a user, I can view any flame is detected in the place.	I can view it by app.	High	Sprint-3
Administrator	Storage	USN-7	As an Administrator I can store the data in cloud database.	All data are stored in Cloud.	High	Sprint-4

6.PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

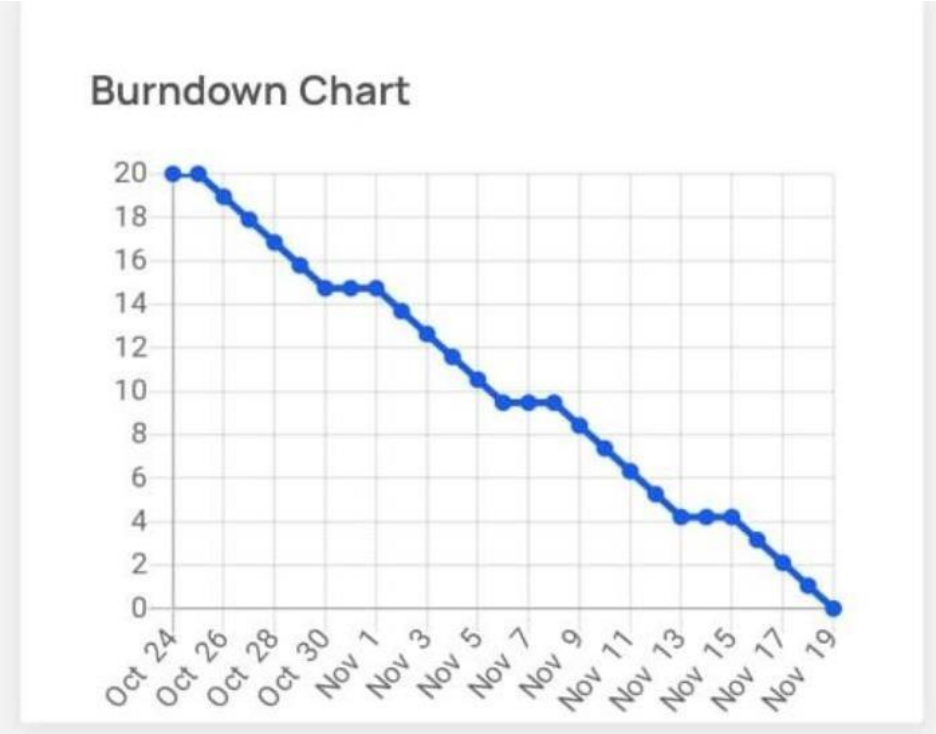
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can download the application.	2	High	Michael Jones J, Dhivakar B, Ahamed Jamaldeen S. Maheshwaran P
		USN-2	As a user, I can register to application with my e-mail.	1	High	Michael Jones J, Dhivakar B, Ahamed Jamaldeen S. Maheshwaran P
		USN-3	As a user, I will receive confirmation email or OTP to SMS once I have registered for the application.	2	High	Michael Jones J, Dhivakar B, Ahamed Jamaldeen S. Maheshwaran P

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Login	USN-4	As a user, I can log into the application by entering email & password.	2	High	Michael Jones J, Dhivakar B, Ahamed Jamaldeen S. Maheshwaran P
	Action	USN-5	As a user, I can View Temperature Readings.	1	Medium	Michael Jones J, Dhivakar B, Ahamed Jamaldeen S. Maheshwaran P
Sprint-3	Action	USN-6	As a user, I can view any flame is detected in the place.	2	High	Michael Jones J, Dhivakar B, Ahamed Jamaldeen S. Maheshwaran P
Sprint-4	Storage	USN-7	As an Administrator I can store the data in cloud database.	2	High	Michael Jones J, Dhivakar B, Ahamed Jamaldeen S. Maheshwaran P

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	20	6 Days	24 Oct 2022	30Oct 2022	20	30 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	06 Nov 2022	20	31 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	07 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	14 Nov 2022

6.3 REPORTS


Burndown Chart
20
18
16
14
12
10
8
6
4
2
0
Oct 24
Oct 26
Oct 28
Oct 30
Nov 1
Nov 3
Nov 5
Nov 7
Nov 9
Nov 11
Nov 13
Nov 15
Nov 17
Nov 19

7.CODING & SOLUTIONING

7.1 FEATURE 1

```
import wiotp.sdk.device
import time
import random

myConfig = {
    "identity": {
        "orgId": "0lchi7",
        "typeId": "FLAME_SMOKE",
        "deviceId": "20020214"
    },
    "auth": {
        "token": "uczr+)jdVFJVbBygLP"
    }
}


def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m = cmd.data['command']


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

while True:
    flame =random.randint(0, 200)
    smoke =random.randint(0, 100)
    myData = {'flame': flame, 'smoke': smoke}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(10)
client.disconnect()
```

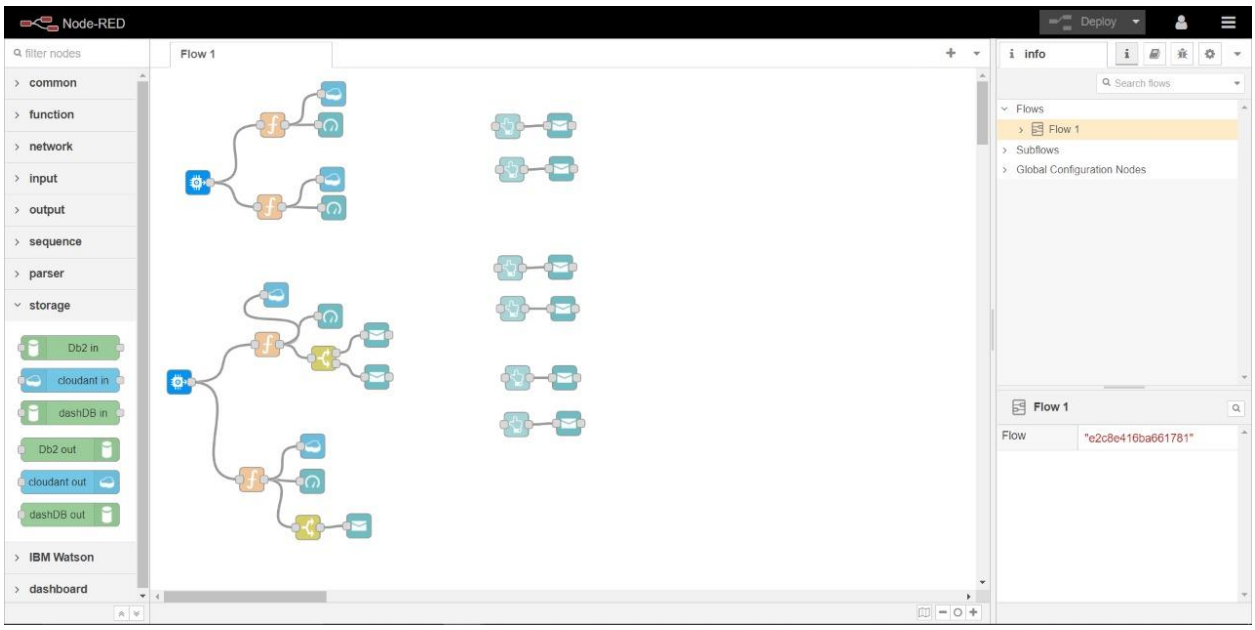
```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected
```

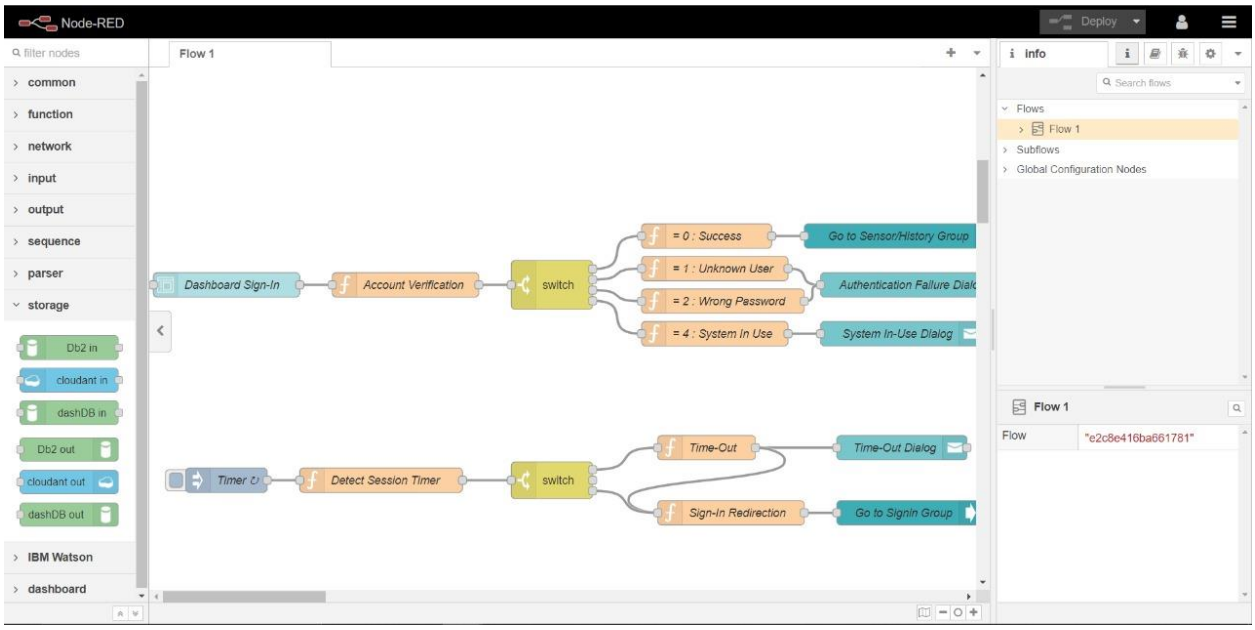
```
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "0lchi7"//IBM ORGANITION ID
#define DEVICE_TYPE "DHT22_Sensor"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "14022002"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "z)aPZNK2tyveCR68li"     //Token
String data3;
float h, t;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and
format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT command type
AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by
passing parameter like server id,portand wificredential


void setup()// configureing the ESP32
{
  Serial.begin(115200);
  dht.begin();
  pinMode(LED,OUTPUT);
  delay(10);
  Serial.println();
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{

  h = dht.readHumidity();
  t = dht.readTemperature();
  Serial.print("temp:");
  Serial.println(t);
  Serial.print("Humid:");
```

```
  Serial.println(h);

  PublishData(t, h);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}



/.....................................retrieving to Cloud................................./

void PublishData(float temp, float humid) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"Temperature\":";
  payload += temp;
  payload += "," "\"Humidity\":";
  payload += humid;
  payload += "}";

  Serial.print("Sending payload: ");
  Serial.println(payload);

  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish
ok in Serial monitor or else it will print publish failed
  } else {
    Serial.println("Publish failed");
  }
}


void mqttconnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }
     initManagedDevice();
     Serial.println();
  }
}
void wificonnect() //function defination for wificonnect
```

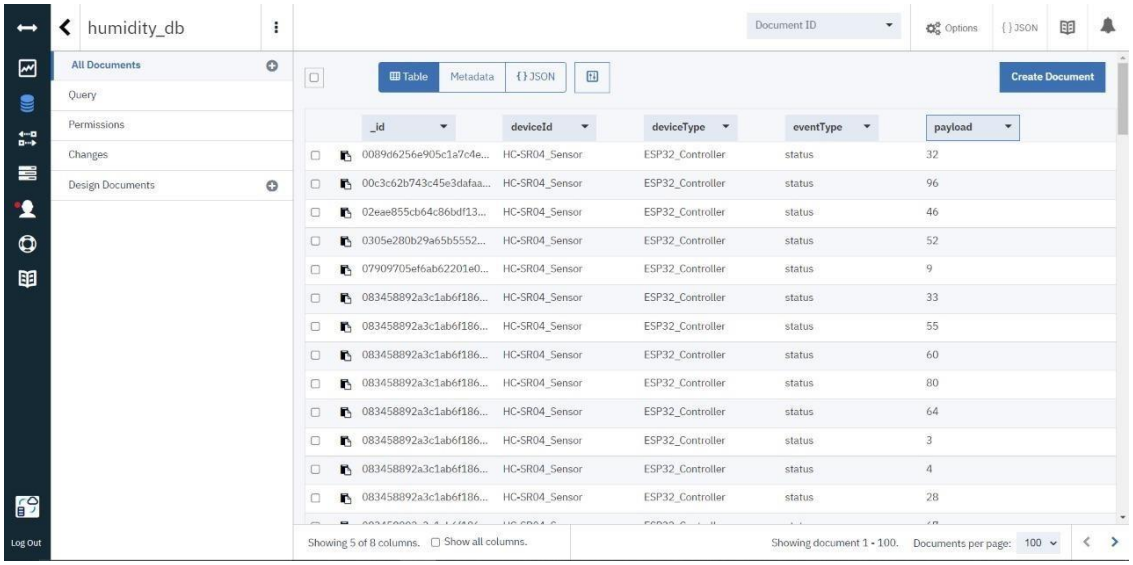
```
{
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic);
  for (int i = 0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]);
    data3 += (char)payload[i];
  }
  Serial.println("data: "+ data3);
  if(data3=="lighton")
  {
Serial.println(data3);
digitalWrite(LED,HIGH);
  }
  else
  {
Serial.println(data3);
digitalWrite(LED,LOW);
  }
data3="";
}
```

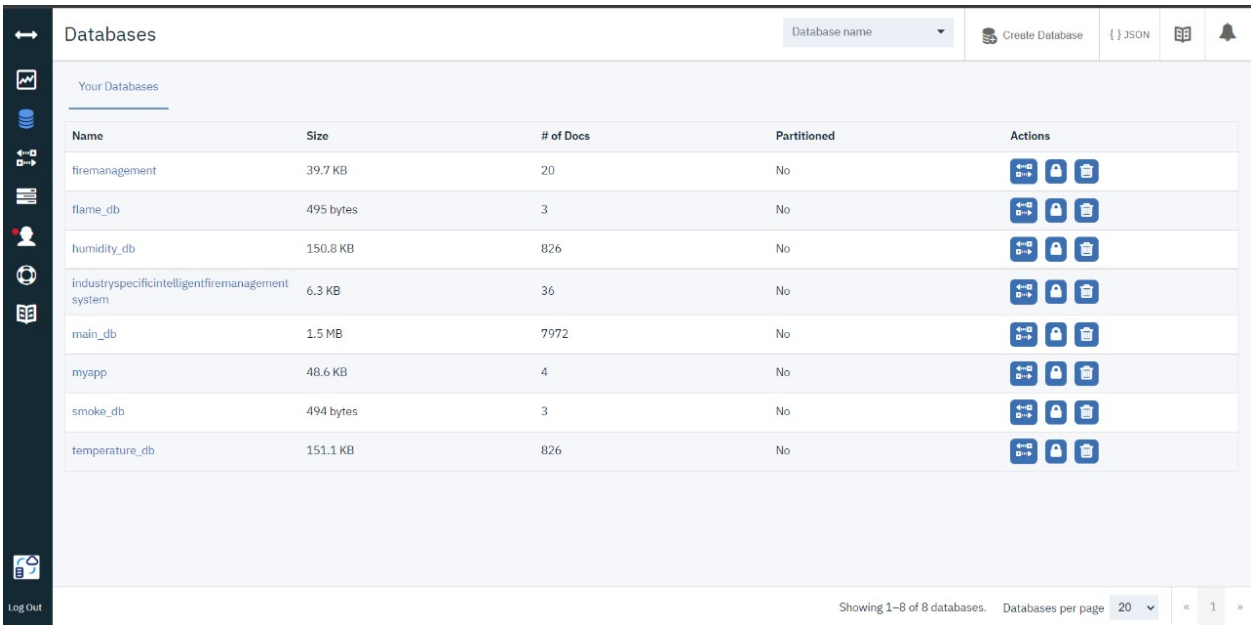
7.2 FEATURE 2

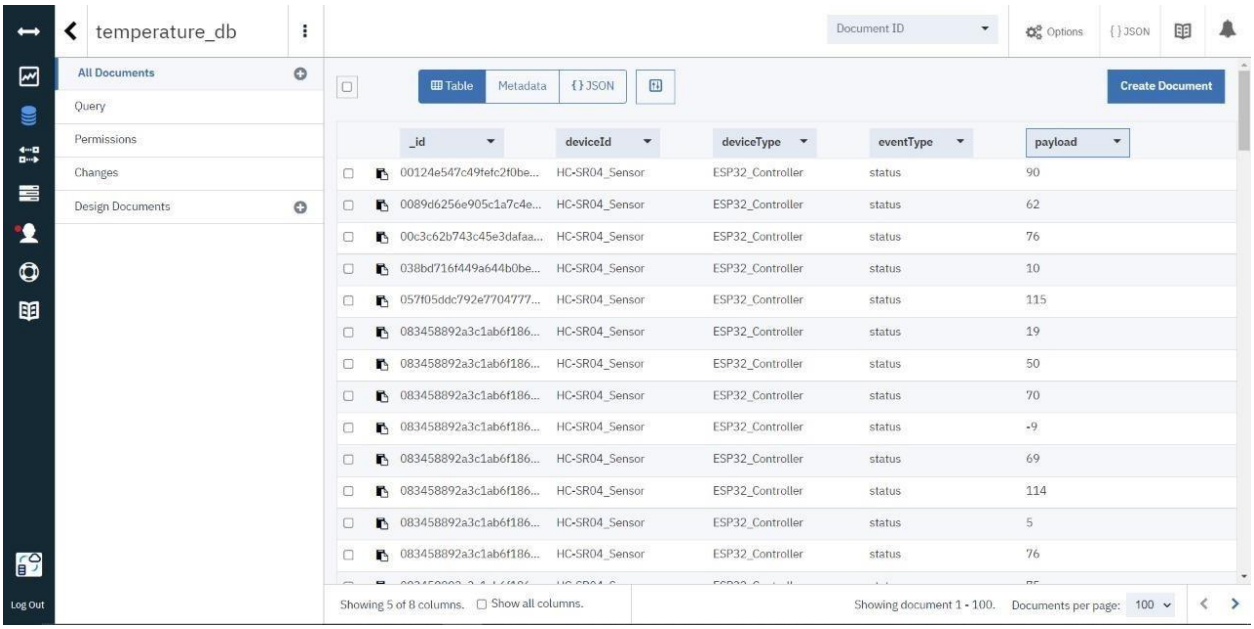

Node-RED
Deploy
filter nodes
Flow 1
common
function
network
input
output
sequence
parser
storage
Db2 in
cloudant in
dashDB in
Db2 out
cloudant out
dashDB out
IBM Watson
dashboard
info
Search flows
Flows
Flow 1
Subflows
Global Configuration Nodes
Flow
"e2c8e416ba661781"

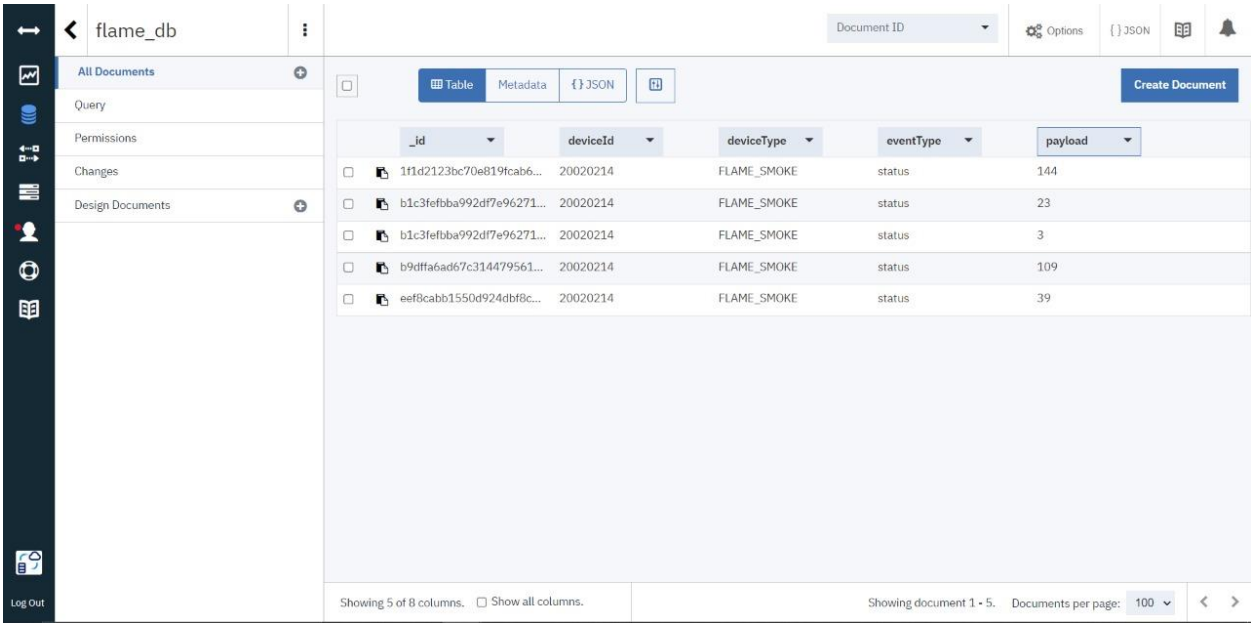

Node-RED
Deploy
filter nodes
Flow 1
common
function
network
input
output
sequence
parser
storage
Db2 in
cloudant in
dashDB in
Db2 out
cloudant out
dashDB out
IBM Watson
dashboard
info
Search flows
Flows
Flow 1
Subflows
Global Configuration Nodes
Dashboard Sign-In
Account Verification
switch
= 0 : Success
Go to Sensor/History Group
= 1 : Unknown User
= 2 : Wrong Password
= 4 : System In Use
System In-Use Dialog
Timer
Detect Session Timer
switch
Time-Out
Time-Out Dialog
Sign-In Redirection
Go to SignIn Group
Flow
"e2c8e416ba661781"

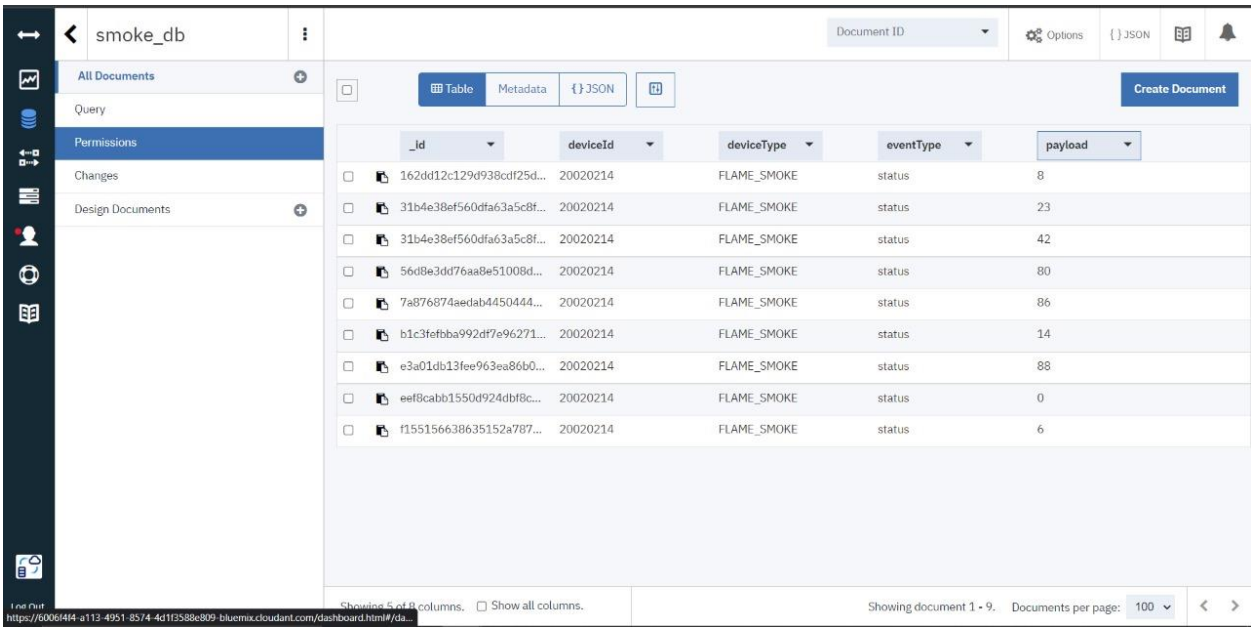
7.3 DATABASE SCHEMA






temperature_db
Document ID
Options
{ } JSON
All Documents
Query
Permissions
Changes
Design Documents
Table
Metadata
{ } JSON
Create Document
_id
deviceId
deviceType
eventType
payload
00124e547c49fefc2f0be... HC-SR04_Sensor ESP32_Controller status 90
0089d6256e905c1a7c4e... HC-SR04_Sensor ESP32_Controller status 62
00c3c62b743c45e3dafaa... HC-SR04_Sensor ESP32_Controller status 76
038bd716f449a644b0be... HC-SR04_Sensor ESP32_Controller status 10
057f05ddc792e7704777... HC-SR04_Sensor ESP32_Controller status 115
083458892a3c1ab6f186... HC-SR04_Sensor ESP32_Controller status 19
083458892a3c1ab6f186... HC-SR04_Sensor ESP32_Controller status 50
083458892a3c1ab6f186... HC-SR04_Sensor ESP32_Controller status 70
083458892a3c1ab6f186... HC-SR04_Sensor ESP32_Controller status -9
083458892a3c1ab6f186... HC-SR04_Sensor ESP32_Controller status 69
083458892a3c1ab6f186... HC-SR04_Sensor ESP32_Controller status 114
083458892a3c1ab6f186... HC-SR04_Sensor ESP32_Controller status 5
083458892a3c1ab6f186... HC-SR04_Sensor ESP32_Controller status 76
Log Out
Showing 5 of 8 columns.
Show all columns.
Showing document 1 - 100.
Documents per page: 100


flame_db
Document ID
Options
{ } JSON
All Documents
Query
Permissions
Changes
Design Documents
Table
Metadata
{ } JSON
Create Document
_id
deviceId
deviceType
eventType
payload
1f1d2123bc70e819fcab6... 20020214 FLAME_SMOKE status 144
b1c3fefbba992df7e96271... 20020214 FLAME_SMOKE status 23
b1c3fefbba992df7e96271... 20020214 FLAME_SMOKE status 3
b9dffa6ad67c314479561... 20020214 FLAME_SMOKE status 109
eef8cabb1550d924dbf8c... 20020214 FLAME_SMOKE status 39
Log Out
Showing 5 of 8 columns.
Show all columns.
Showing document 1 - 5.
Documents per page: 100



8. TESTING

8.1 TEST CASE

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
NodeRED_TC_001	Functional	Login Page	Verify user is able to see the Login		1.Enter URL and click go 2.Enter username 'mike' and password 'mike'	https://my-app-14.eu-gb.mybluemix.net/ui	Login page should display and redirect to home page after authentication	Working as expected	Pass
NodeRED_TC_002	Functional	Home page	Check if data recieves from sensor		1.Run the Python script and wokwi simulation	https://my-app-14.eu-gb.mybluemix.net/ui	Data received is displayed in UI	Working as expected	Pass
NodeRED_TC_003	Functional	Home page	Check if alert box is triggered if flameis detected		1.Run the Python script and wokwi simulation	Sample temperature, humidty, flame ,smoke are given.	An alert dialog box is obtained	Working as expected	Pass
NodeRED_TC_004	Functional	Home page	Check whether the data received are uploaded to the IBM Cloud.	Should have logged into web UI	1. Create Cloudant account and create necessary database. 2. Using the cloudant node in Node RED upload data to Cloudant	Sample data received from python script and wokwi simulation.	The data need to be stored in cloudant .	Working as expected	Pass
NodeRED_TC_005	UI	Home page	Verify user is able to access buttons	Should have logged into web UI	1.Click on buttons	Button is Clicked	An alert dialog box is obtained	Working as expected	Pass
NodeRED_TC_006	UI	Home page	If there is need to turn on fan. A pop up will be given in top left corner	Should have run puthon and wokwi simulation	1.Run the Python script and wokwi simulation	Sample temperature, humidty, flame ,smoke are given.	pop up showed	Working as expected	Pass
Python_TC_005	Functional	Sensor Data	When Sensor data is received it should be sent to the Node-Red		1.Collect data from sensor 2.Convert data to json format 3.Connect with Ibm Watson device	Data received in IBM watson device	Event data received	Working as expected	Pass
Wokwi_TC_006	Functional	DHT22 Sensor Data	When Sensor data is received it should be sent to the Node-Red		1.Connect DHT22 sensor to ESP32 and start simulation 2.Connect with IBM Watson	Data received in IBM watson device	Event data received	Working as expected	Pass

8.2 USER ACCEPTANCE TESTING

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issuesof the Industry-Specific intelligent fire management System project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report showsthe 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	1	1	0	0	2
Duplicate	1	0	0	0	1
External	0	1	2	0	3
Fixed	3	2	1	1	7
Not Reproduced	0	0	1	0	1
Skipped	0	1	1	1	3
Won't Fix	0	1	2	0	3
Totals	5	6	7	2	20

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Node Red engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Python Script	3	0	0	3
IBM Cloud	9	0	0	9
Final Report Output	4	0	0	4

9. RESULTS
9.1 PERFORMANCE METRICS

The accuracy of the app is 96.5 %.

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Fire alarms save lives.
- Fire alarms reduce property loss.
- Fire alarms shorten your recovery time.
- 24/7 Monitoring.

DISADVANTAGES

- Cost,not competitively priced for larger systems.
- Detection of smoke or a fire is done by zone, which could be multiple areasrather than specifying a specific location.

This could delay emergency responders from locating the fire.

- Inability to provide detail in formation, such as device

locations, etc…

- No details on event history.

11. CONCLUSION

The objective of this project was to design and develop a simple, reliable, efficient, and automatic fire management system that has a precise and quick notification mechanism. Appropriate sensors were used to detect increase or decrease in temperature, smoke, flame and humidity, as the onset of fire. A step-by-step approach was followed in the design of the system. The design was carried outbased on the study and analysis of existing similar systems and user perceptions. A prototype of the system was implemented and tested in home and office environments. Several tests were conducted, and the results were analysed to ensure that the system produced the intended results. The system has been implemented and tested, showing satisfactory performance.

12. FUTURE SCOPE

- Add high pressure water mist
- Could introduce drones for fire management.
- Wireless devices to provide mobile capabilities to home owners.
- Could add Sound-Triggered feature.
- To make the alarm and sprinklers turn on automatically.

13. APPENDIX

13.1 SOURCE import wiotp.sdk.device

```
import time
import random

myConfig = {
    "identity": {
        "orgId": "0lchi7",
        "typeId": "ESP32_Controller",
        "deviceId": "HC-SR04_Sensor"
    },
    "auth": {
        "token": "pKM00oySMURsO5npF)"
    }
}


def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s"
cmd.data['command'])
   m = cmd.data['command']
```

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

while True:
    temperature =random.randint(-20, 500)
    humidity =random.randint(0, 100)
   myData = {'temperature': temperature, 'humidity': humidity}
 client.publishEvent(eventId="status", msgFormat="json", data=myData,
qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(60)
client.disconnect()
```

13.2 GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-15213-1659594983

13.3 PROJECT DEMO LINK:

https://drive.google.com/drive/folders/1btVSEe6vEhHUslYyw0COOQF0VKPKdvhK?usp=sharing