Industry-specific intelligent fire management system

Using Wokwi IOT Platform To MIT App

Team ID	**PNT2022TMID27962**
Project Name	**Industry Specific intelligent fire management system**

Wokwi Code:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15
#define dhtpin 13     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#include <ESP32Servo.h>

const int servoPin = 18;
Servo servo;

DHT dht1 (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of
dht connected
DHT dht2 (dhtpin, DHTTYPE);
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

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

#define ORG "2piqlm"//IBM ORGANITION ID
#define DEVICE_TYPE "Code"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "123456"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
String data3;

float g , t;
float fl,T;
int f;
int pos;



//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
```

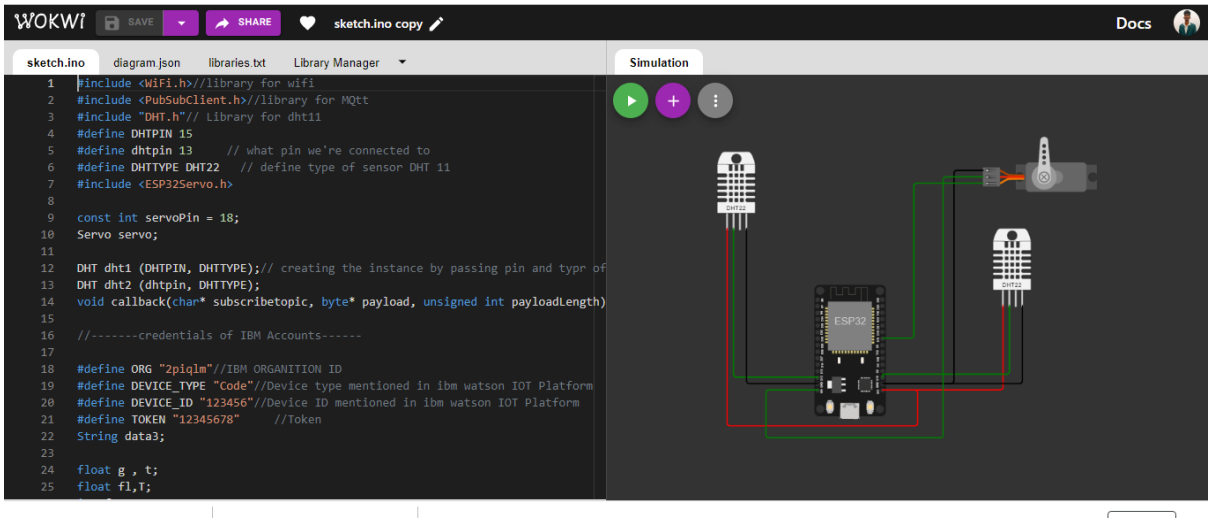
```
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of
event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT
command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
float dist,dur;
String data;
//-----------------------------------------
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);
  dht1.begin();
  dht2.begin();
  delay(10);
  Serial.println();
  servo.attach(servoPin, 500, 2400);
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{

  g = dht1.readHumidity();
  t = dht1.readTemperature();
  fl = dht2.readHumidity();

  Serial.print("temp:");
  Serial.println(t);
  Serial.print("gas:");
  Serial.println(g);
  Serial.print("flame:");
  Serial.println(fl);
  if(g>=50)
  {
    Serial.print("Gas Detected , Fan ON");
    f=1;
    for (pos = 0; pos <= 180; pos += 1) {
    servo.write(pos);
    delay(15);
  }
```

```
  for (pos = 180; pos >= 0; pos -= 1) {
    servo.write(pos);
    delay(15);
  }

  }
  if(fl>=90)
  {
    Serial.print("Flame Detected , Sprinkler ON");
    f=1;
    for (pos = 0; pos <= 180; pos += 1) {
    servo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    servo.write(pos);
    delay(15);
  }

  }


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



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

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


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

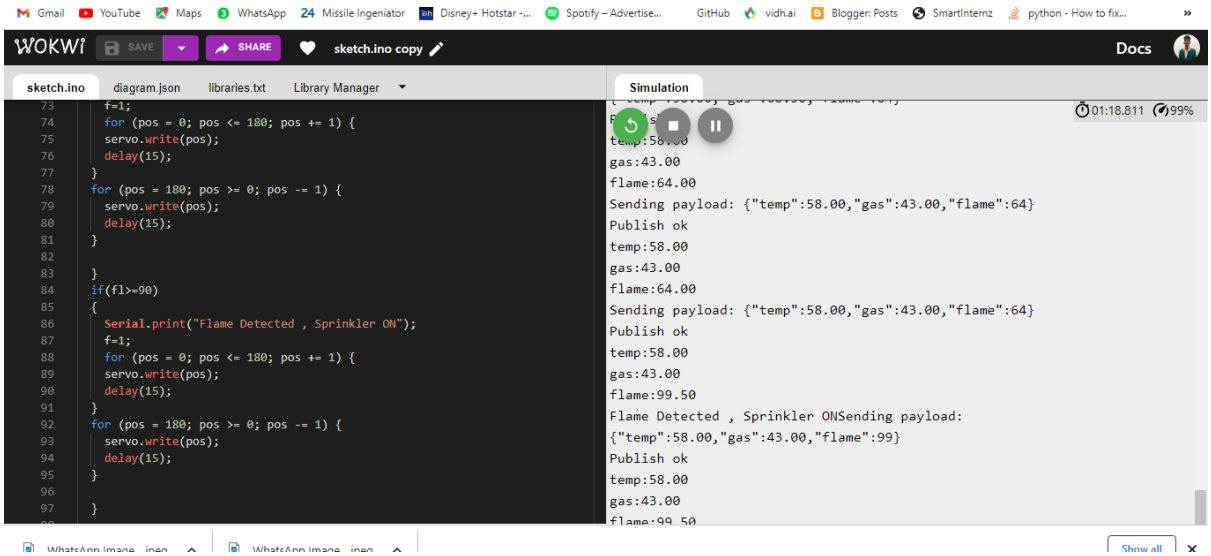
```
  Serial.println(payload);


  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 PublishAlert() {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"alert\":";
  payload += 10000;
  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();
  }
```

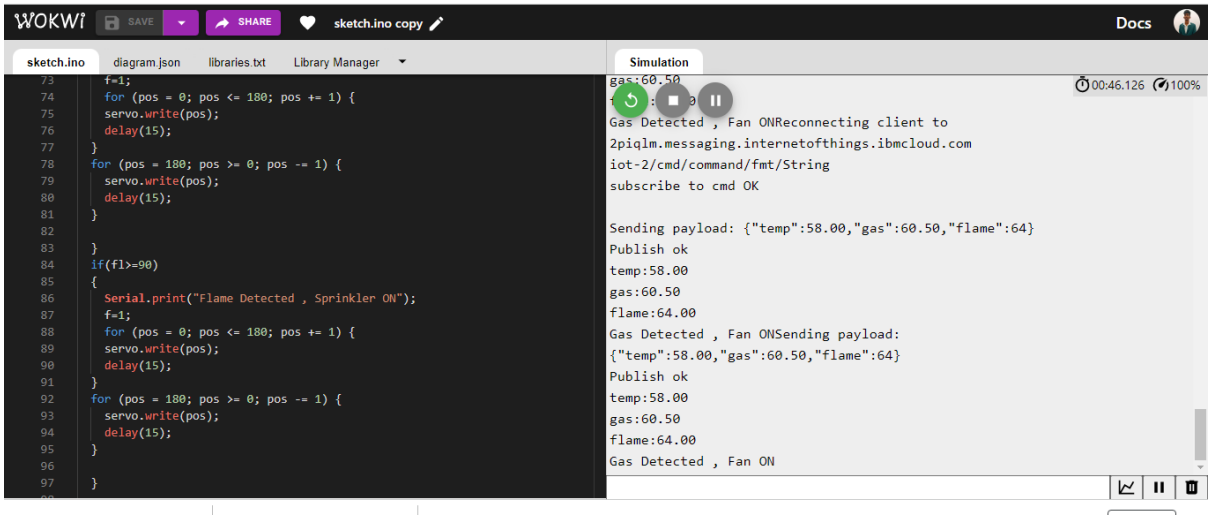
```
}
void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");

  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)
{

}
```

Wokwi Circuit:

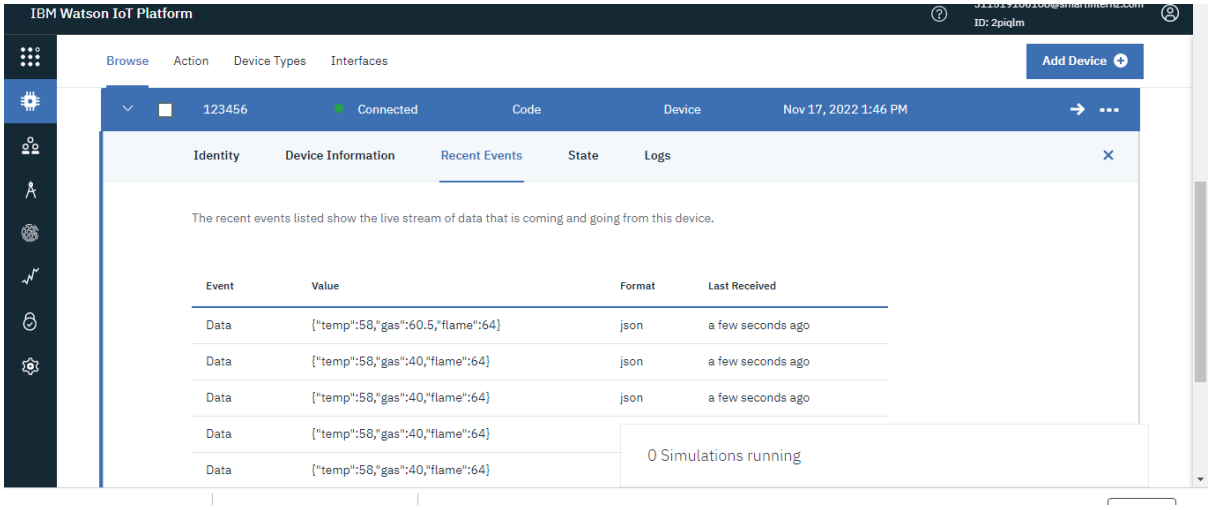


WOKWI OUTPUT:

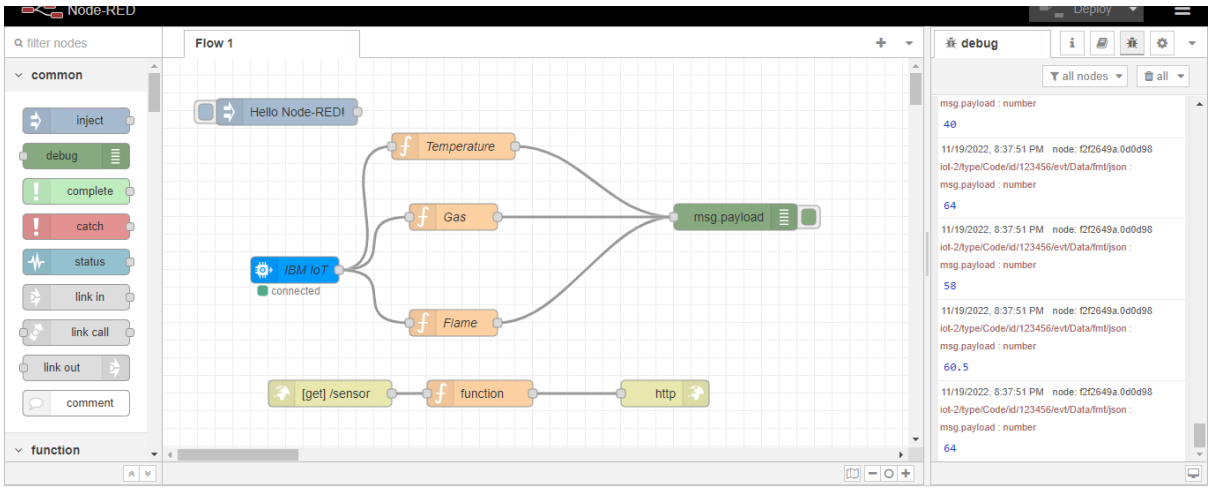




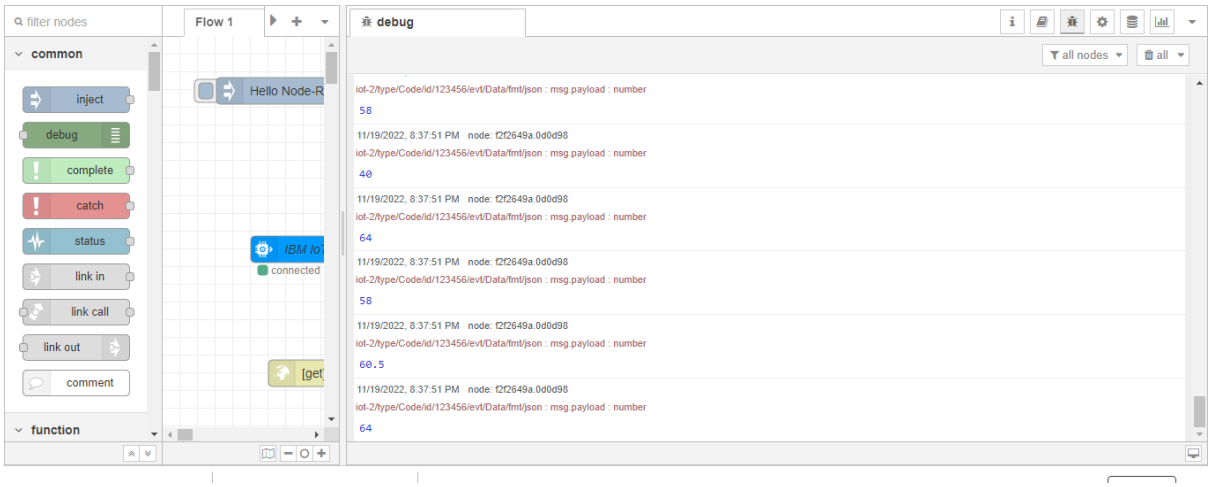
WOKWI TO IBM IOT WATSON:



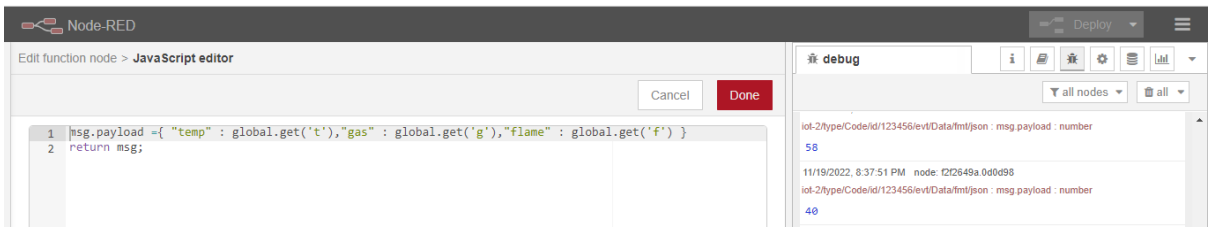
IBM WATSON TO NODE RED:



NODE RED OUTPUT:



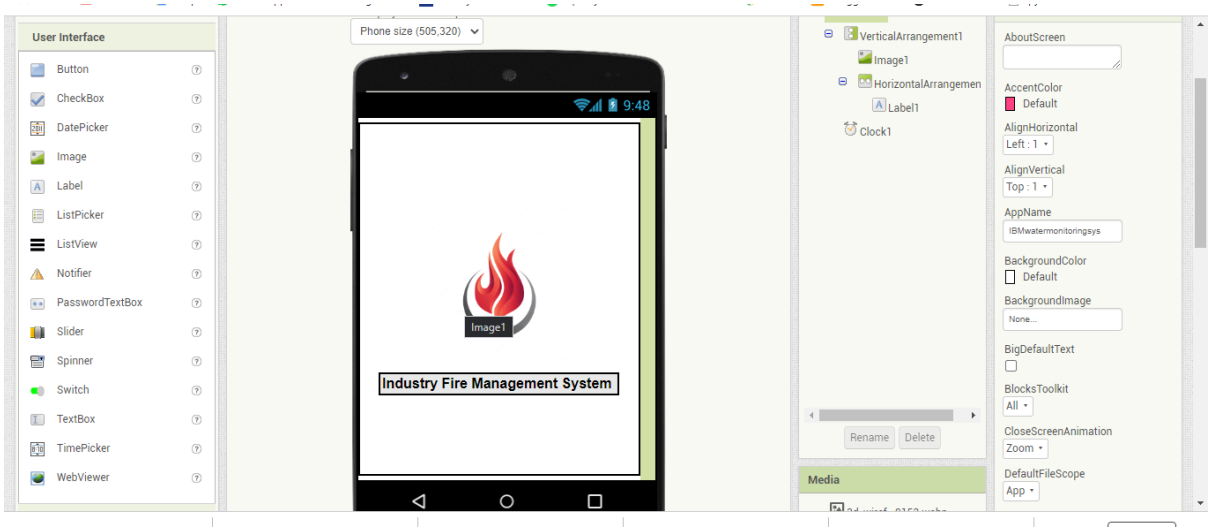
NODE RED FUNCTION CODE:



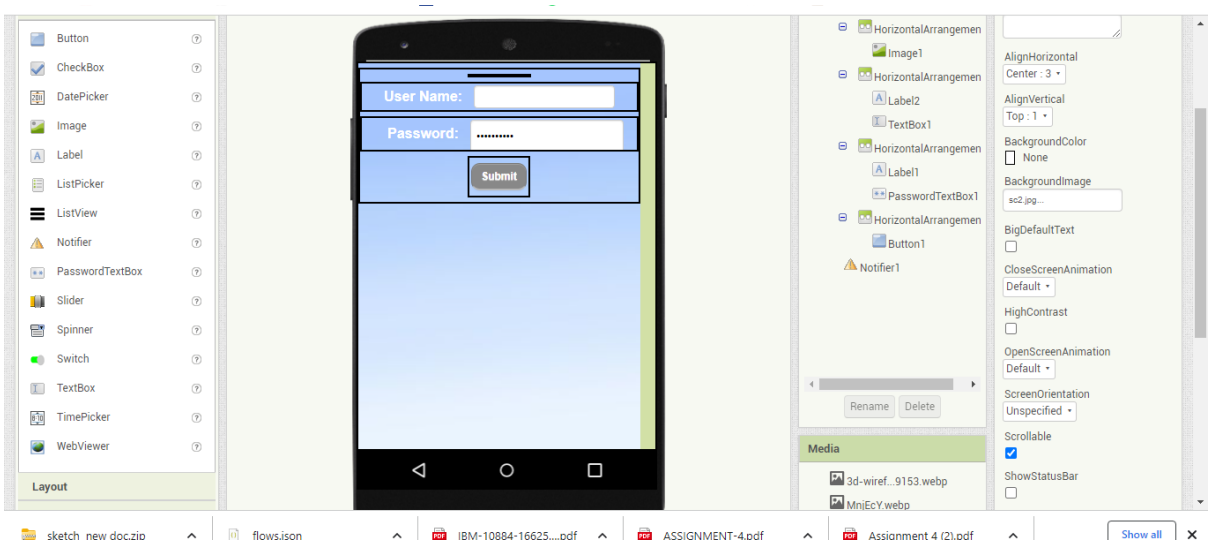
NODE RED TO MIT APP:

APP CREATION: FRONTEND

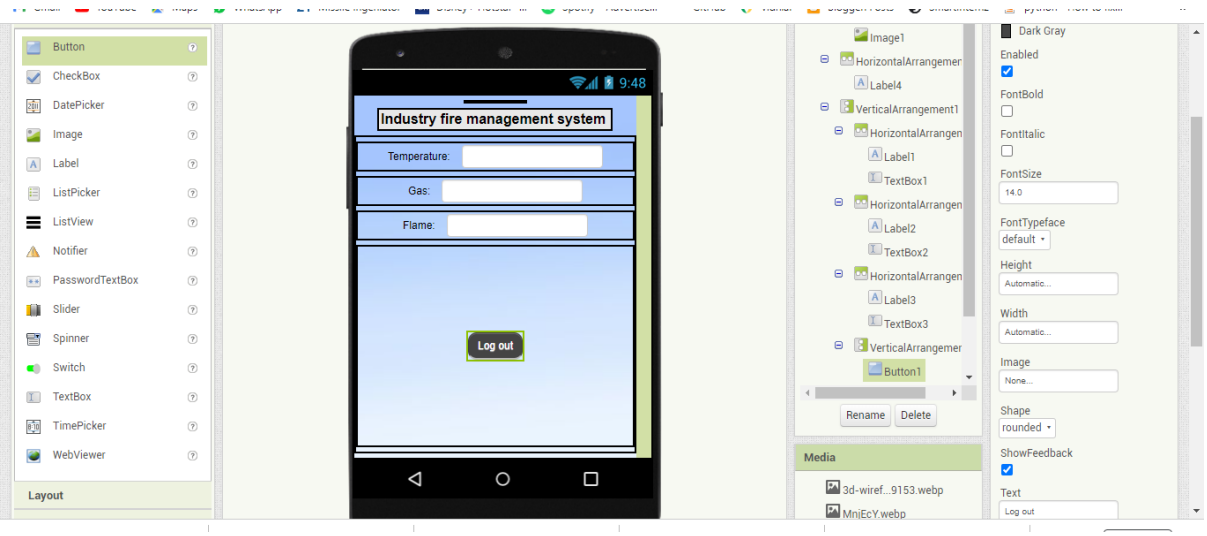
SCREEN 1:



SCREEN 2:

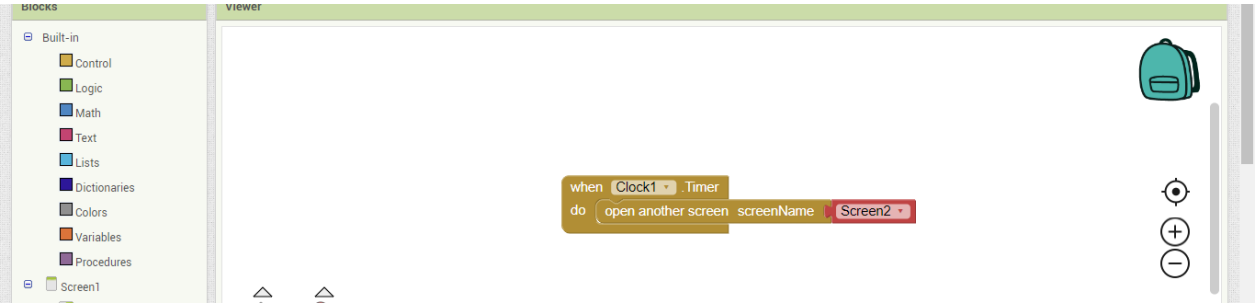


SCREEN 3:

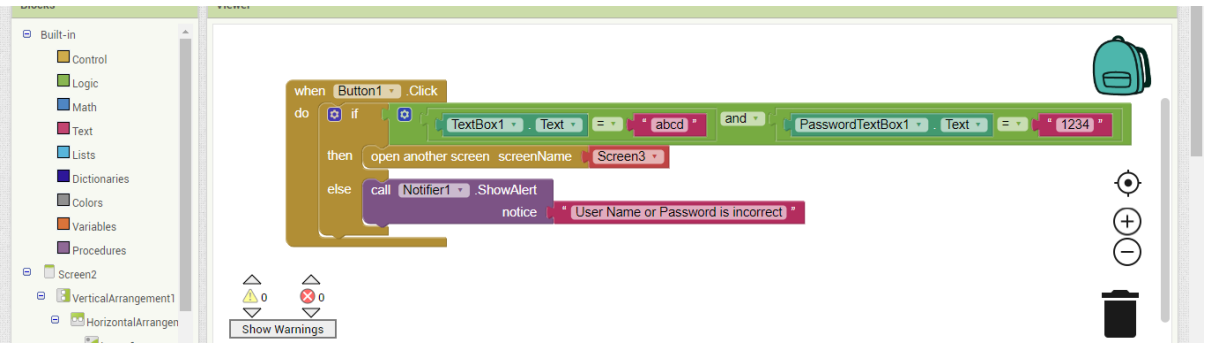


BACKEND BLOCKS:

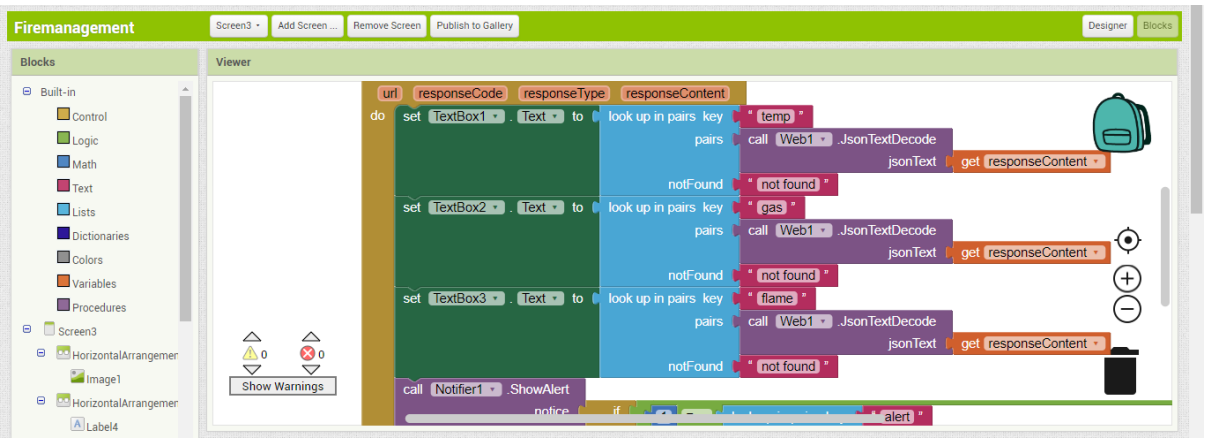
SCREEN 1:



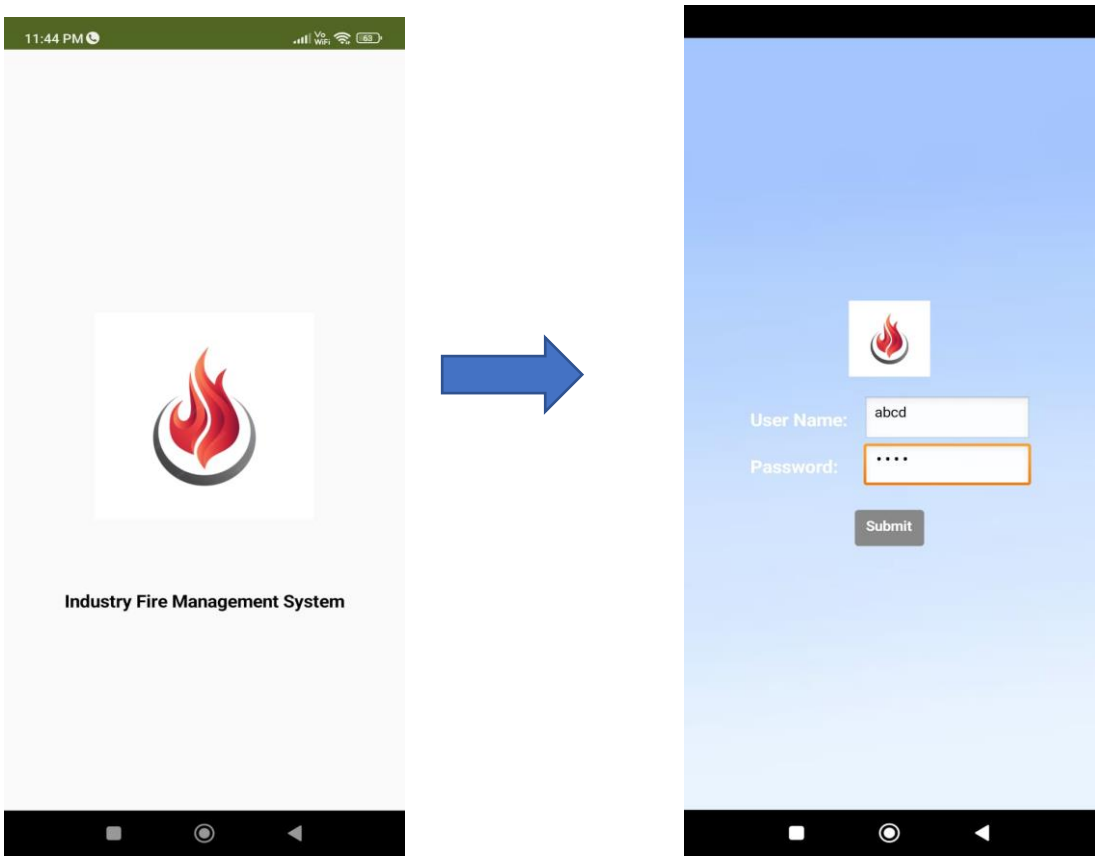
SCREEN 2:

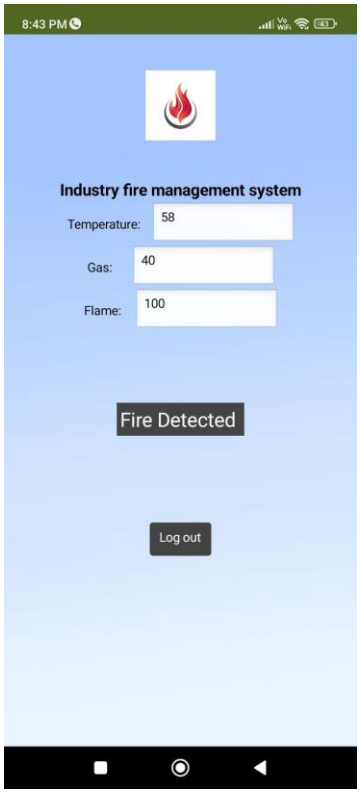


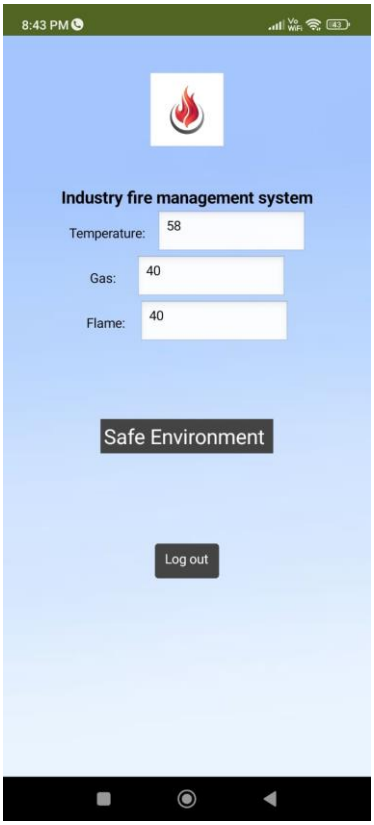
SCREEN 3:



MIT MOBILE APP OUTPUT:







If the values of flame is above 90 the fire is detected. If the values of the flame is below 90 it is safe environment.