**Project Development Phase**

**Sprint – 4**

|  |  |
| --- | --- |
| Team ID | PNT2022TMID01003 |
| Project Name | Hazardous Area Monitoring for Industrial Plant powered by IoT |
| Maximum Marks |  |

**Task:**

A mobile application for monitoring the Environment parameters around the region of an industry has been developed using MIT App Inventor.

**Screens Information:**

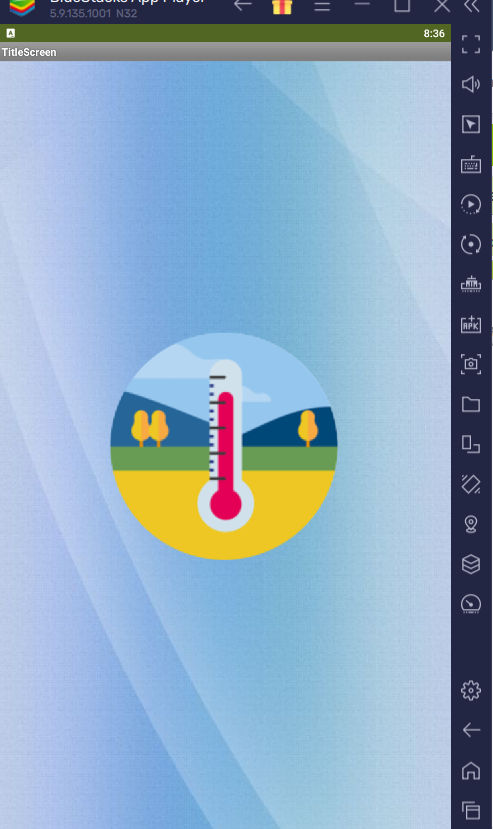
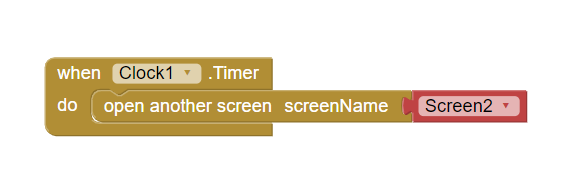
1. **Screen – 1:** It is the entry screen of the mobile application and will be displayed only for 2000 milli-seconds.

2. **Screen – 2:** It is the login page of the application. Each user has their own user id and password, which is known only to them. After validating the credential, the user can access the data captured by the placed device.

3. **Screen – 3:** Environmental parameters in the area of the industry like temperature is obtained via sensors and is sent to the mobile device.

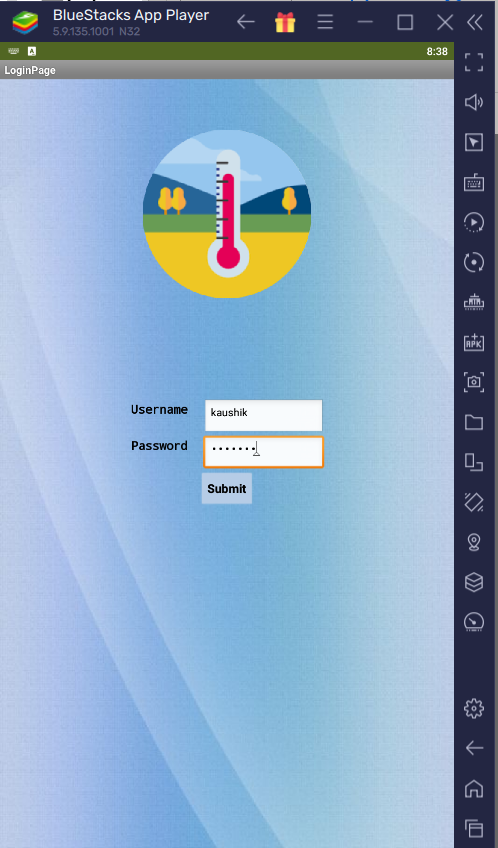
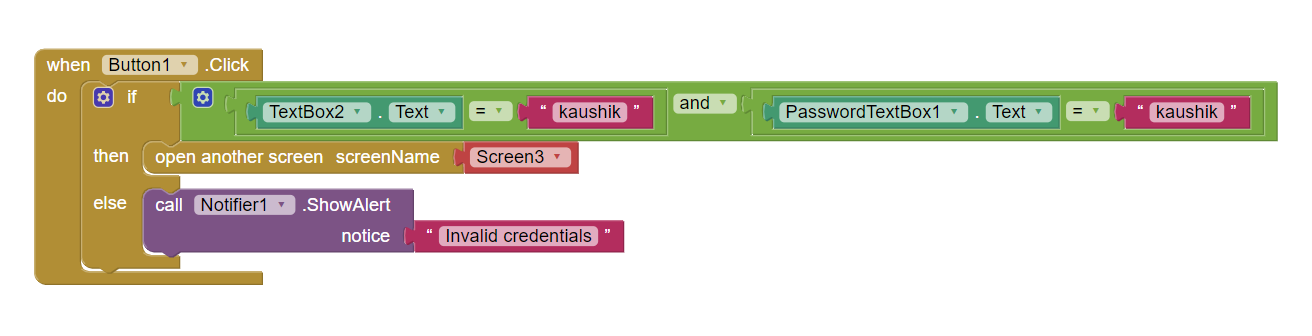
**Screen 1:**

**Designer & Blocks**

**Screen 2:**

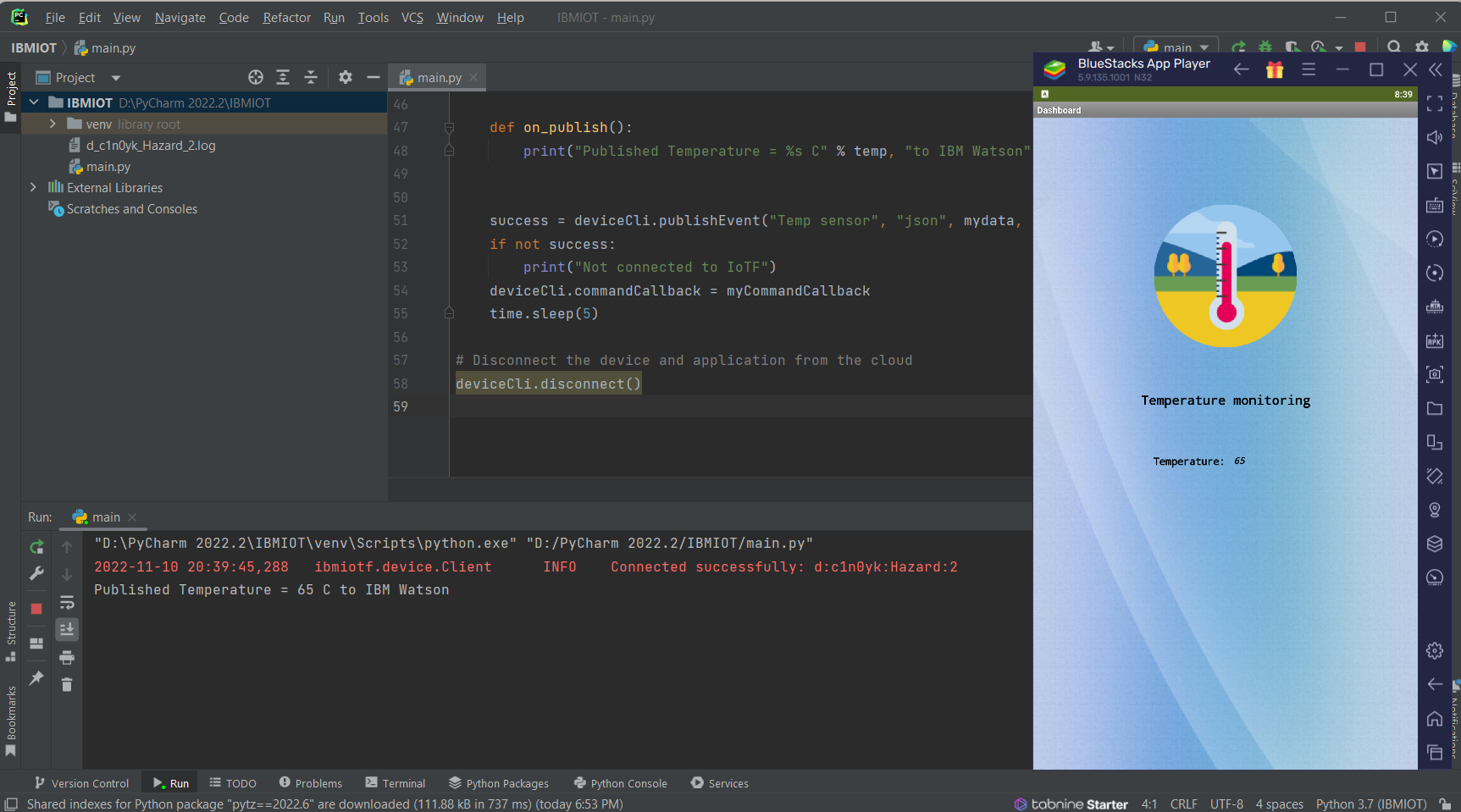
**Designer & Blocks**

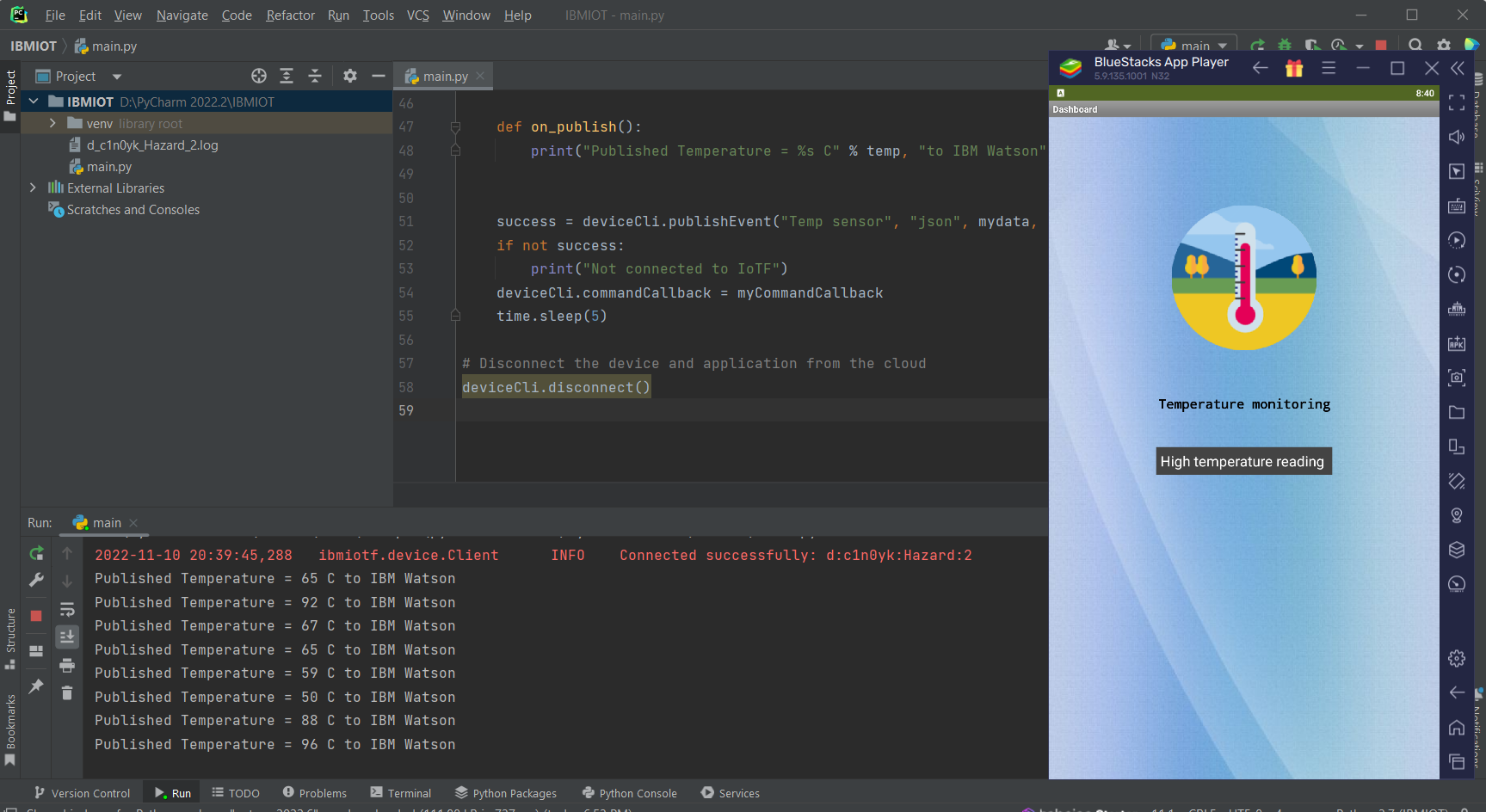
**Screen 3:**

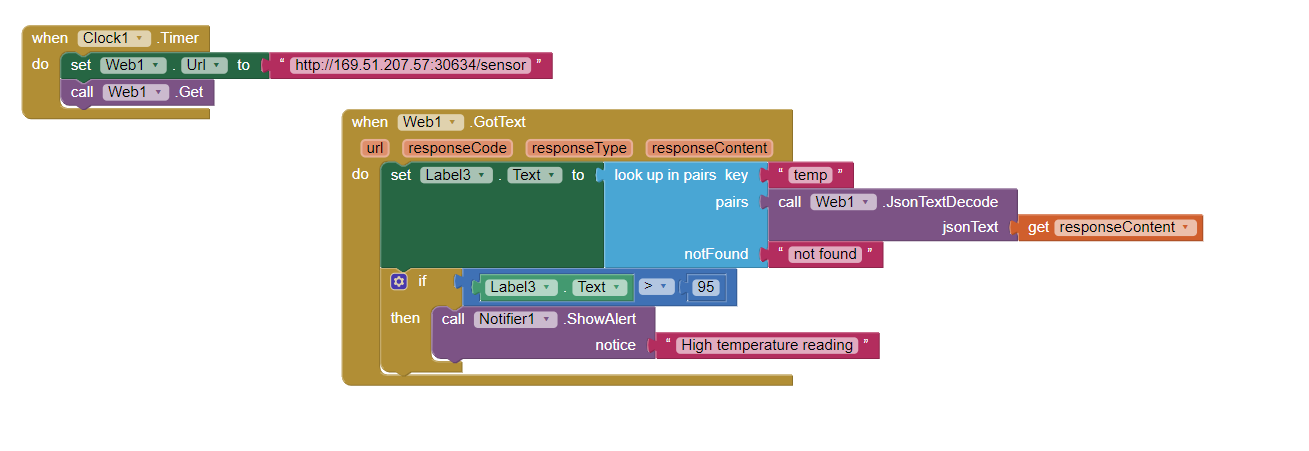
**Designer & Blocks**

*Case 1 (When the temperature is within limit):*



*Case 2 (When temperature exceeds normal (95 C) value):*





**Source code:**

import time  
import sys  
import ibmiotf.application  
import ibmiotf.device  
import random  
  
# Provide your IBM Watson Device Credentials  
organization = "c1n0yk"  
deviceType = "Hazard"  
deviceId = "2"  
authMethod = "token"  
authToken = "123456789"  
  
  
# Initialize GPIO  
def myCommandCallback(cmd):  
 print(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 ibmiotf.ConnectionException as e:  
 print("Caught exception connecting device: %s" % str(e))  
 sys.exit()  
deviceCli.connect()  
  
while True:  
 # Get Sensor Data from DHT11  
  
 temp = random.randint(50, 100)  
  
 mydata = {'temp': temp}  
  
  
 def on\_publish():  
 print("Published Temperature = %s C" % temp, "to IBM Watson")  
  
  
 success = deviceCli.publishEvent("Temp sensor", "json", mydata, qos=0, on\_publish=on\_publish)  
 if not success:  
 print("Not connected to IoTF")  
 deviceCli.commandCallback = myCommandCallback  
 time.sleep(5)  
  
# Disconnect the device and application from the cloud  
deviceCli.disconnect()

# WOKWI CODE:

#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 "iagqzu"//IBM ORGANITION ID

#define DEVICE\_TYPE "Deepak"//Device type mentioned in ibm watson IOT Platform #define DEVICE\_ID "123"//Device ID mentioned in ibm watson IOT Platform #define TOKEN "12345678" //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 = "{\"temp\":"; payload += temp;

payload += "," "\"Humid\":"; 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="";

}

# WOKWI OUTPUT:

