Gas leakage monitoring and alerting system for Industries

TEAM ID: PNT2022TMID04334

TEAM LEAD	HARISH M	737819CSR055
TEAM MEMBER 1	GOMANISHWARAN S	737819CSR045
TEAM MEMBER 2	BHARANI S	737819CSR015
TEAM MEMBER 3	BHARANIDHARAN M	737819CSL234

INDEX

1.INTRODUCTION

1.1 Project Overview

IoT is an expanding network of physical devices that are linked with different types of sensors and with the help of connectivity to the internet, they are able to exchange data. Through IoT, internet has now extended its roots to almost every possible thing present around us and is no more limited to our personal computers and mobile phones. Safety, the elementary concern of any project, has not been left untouched by IoT. Gas Leakages in open or closed areas can prove to be dangerous and lethal. The traditional Gas Leakage Detector Systems though have great precision, fail to acknowledge a few factors in the field of alerting the people about the leakage. Therefore, we have used the IoT technology to make a Gas Leakage Detector having Smart Alerting techniques involving calling, sending text message and an e-mail to the concerned authority and an ability to predict hazardous situation so that people could be made aware in advance by performing data analytics on sensor readings.

1.2 Purpose

Fire accidents have been taking place frequently and the threat to human lives and properties is growing in recent years. Some gases are highly inflammable and can burn even at some distance from the source of leakage. Most fire accidents are caused because of a poor-quality rubber tube or the regulator is not turned off when not in use. Therefore, developing the gas leakage alert system is very essential.

2. LITERATURE SURVEY

2.1 Existing problem

In the existing method, gas sensing technology is used. The LPG leakage is detected by the semiconductor sensor. The leakage of gas may happen due to the human error, false chemical reaction, lack of service done in the gas valve. In the existing method, periodic check done by manually and partial sensing methodology is used. When the leakage was happened, it leads to major fire accident Before controlling the fire major accident may happen which leads to heavy loss in industry as well as human life. In addition to that the leak of gas may spread in the atmosphere, it may affect all the living things in an around them. In the existing system MQ5 sensor is used to detect gas leakage. Exhaust fans are used to suck out the gases when the leakage occurs. In the existing method, it raises only alarm whenever Gas leaked or fire is detected at any place in a factory. Due to this alarm, people could start to run haphazardly. Fire Service truck vehicle only control the fire accident.

2.2 References

[1] Rajeev B. Ahuja, Jayant K. Dash, Prabhat Shrivastava, “A comparative analysis of liquefied petroleum gas (LPG) and kerosene related burns”, Burns, Volume 37, Issue 8, December 2011.

[2] Prof. Pankaj C. Warule, Shivam Upadhyay, Snehal S. Shelke, Sumitra K. Khandade, “LPG Detection, Metering and Control System Using Microcontroller”, IJARIIE, Volume 2, Issue 2, 2016.

[3] Ankit Sood, Babalu Sonkar, Atul Ranjan, Mr. Ameer Faisal, “Microcontroller Based LPG Gas Leakage Detector Using GSM Module”, International Journal of Electrical and Electronics Research, Volume 3, Issue2, April- June 2015.

[4] Ashish Shrivastava, Ratnesh Prabhakar, Rajeev Kumar, Rahul Verma, "GSM Based Gas Leakage Detection System", International Journal of Technical Research and Applications", Volume 1, Issue2, May- June 2013.

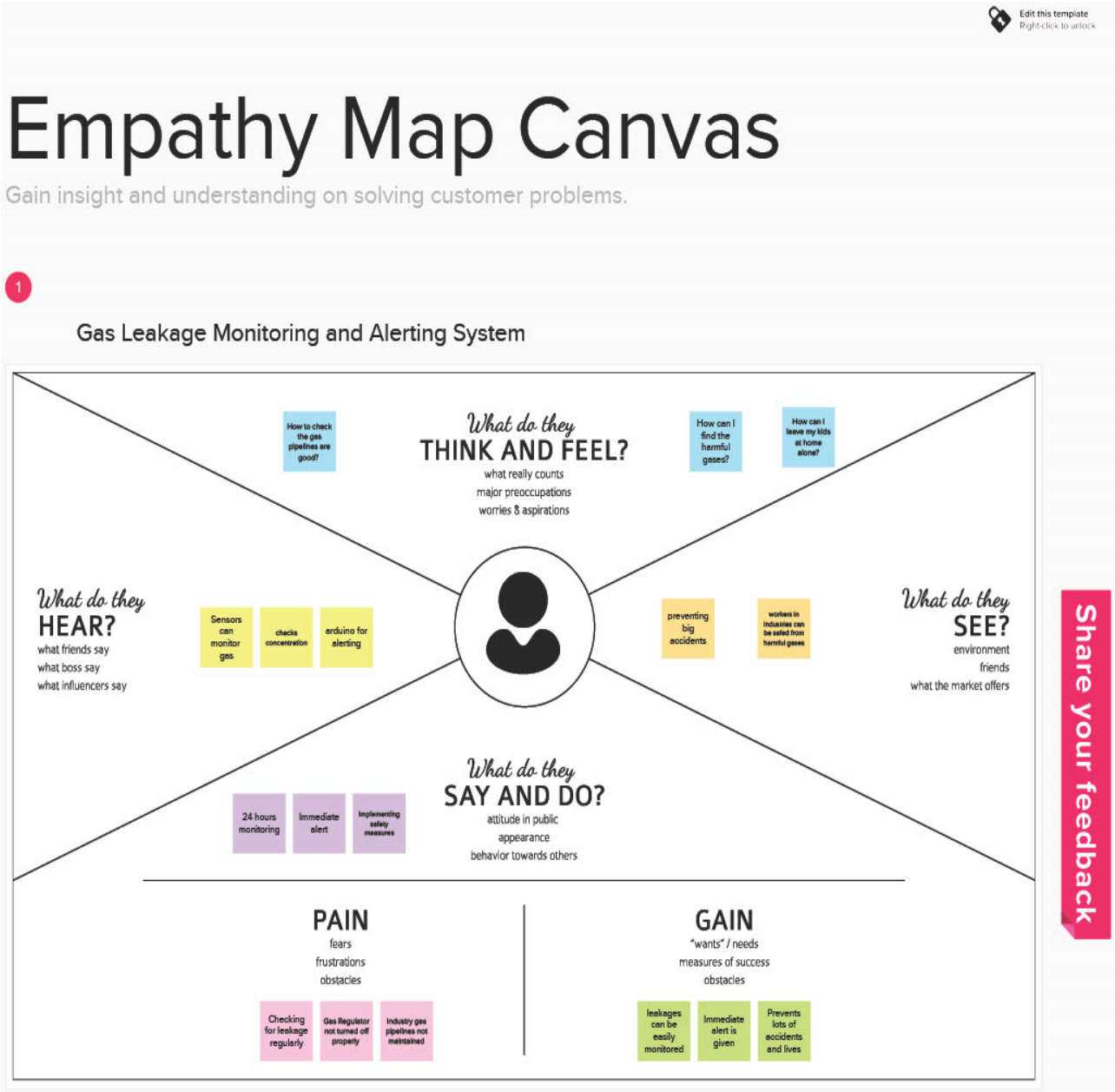
2.3 Problem Statement Definition

The risk of firing, explosion, suffocation all are based on their physical properties such flammability, toxicity etc. The number of deaths due to the explosion of gas cylinders has been increasing in recent years. The reason for such explosion is due to substandard cylinders, old valves, worn out regulators and lack of awareness using gas cylinders add to the risks. Inspections by oil companies found that many LPG consumers are unaware of safety checks of gas cylinders. Another reason is illegal filling of gas cylinder also causes accidents. There is a need for a system to detect and also prevent leakage of LPG.

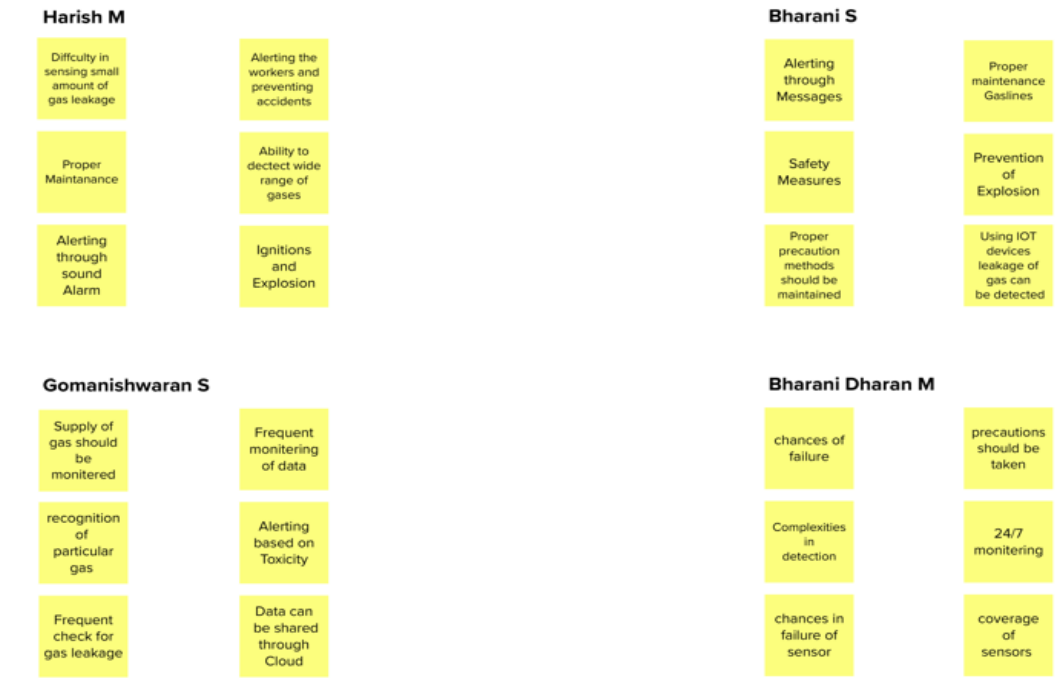
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	worker	Periodic checking of gas leakage	Unable to detect odourless gases and less leakage gases	Because we could not able to identify odourless gases	anxious
PS-2	Industry owner	Protect my industry from accidents	Don't have proper monitoring technology	Because of high initial setup cost	Fear

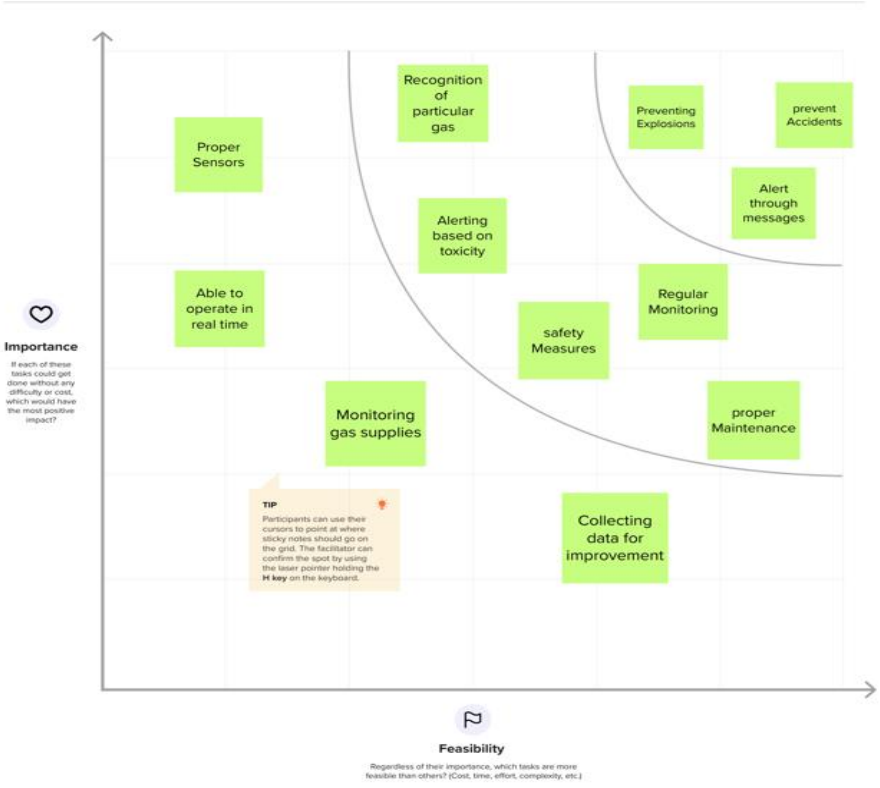
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

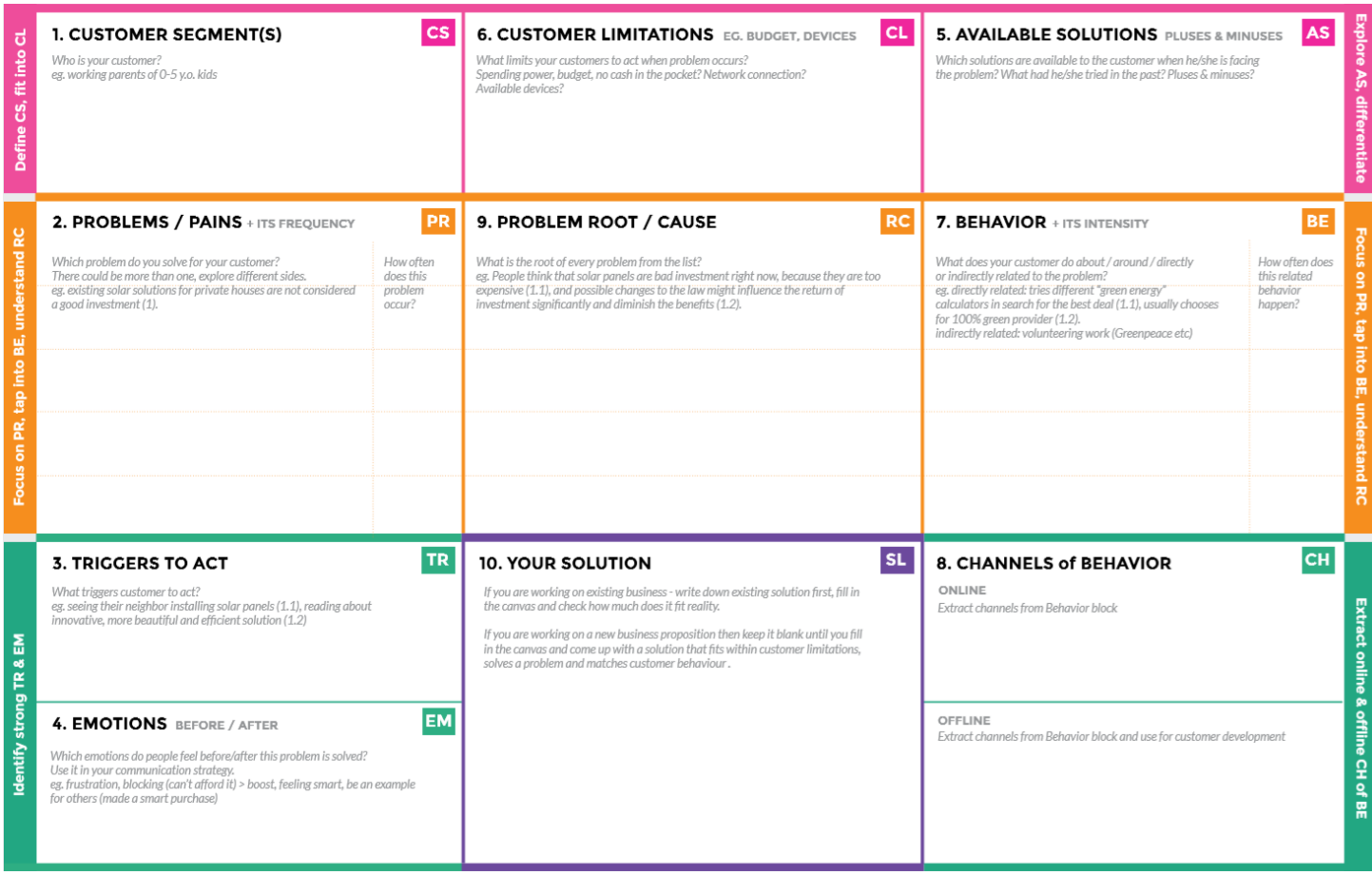




3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	➢ To create the system and an application that are effective in detecting gas and alerting the appropriate parties.
2.	Idea / Solution description	➢ The gas sensors will be fitted at different locations to track gas leaks. ➢ This IOT devices helps in keeping track of dangerous gas emissions. ➢ If a gas leakage is sensed in any region, it will be alerted along with the location. ➢ Users can view the information on the web application.
3.	Novelty / Uniqueness	➢ Works 24/7. ➢ Accurate detecting with level of harmful gases. ➢ Immediate attention/alarm to the workers and people.
4.	Social Impact / Customer Satisfaction	➢ Fault tolerance ➢ Easy operational ➢ Avoids major accidents
5.	Business Model (Revenue Model)	➢ The item is heavily promoted across all mediums. It even protects small-scale industries against calamities because it is affordable. ➢ Everyone can understand how to use the product, therefore it's simple for them to use it correctly for their safest organisation.
6.	Scalability of the Solution	➢ Can be used for small and large industries ➢ Can be used not only in industries but also in home for LPG leakage detection and other places.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

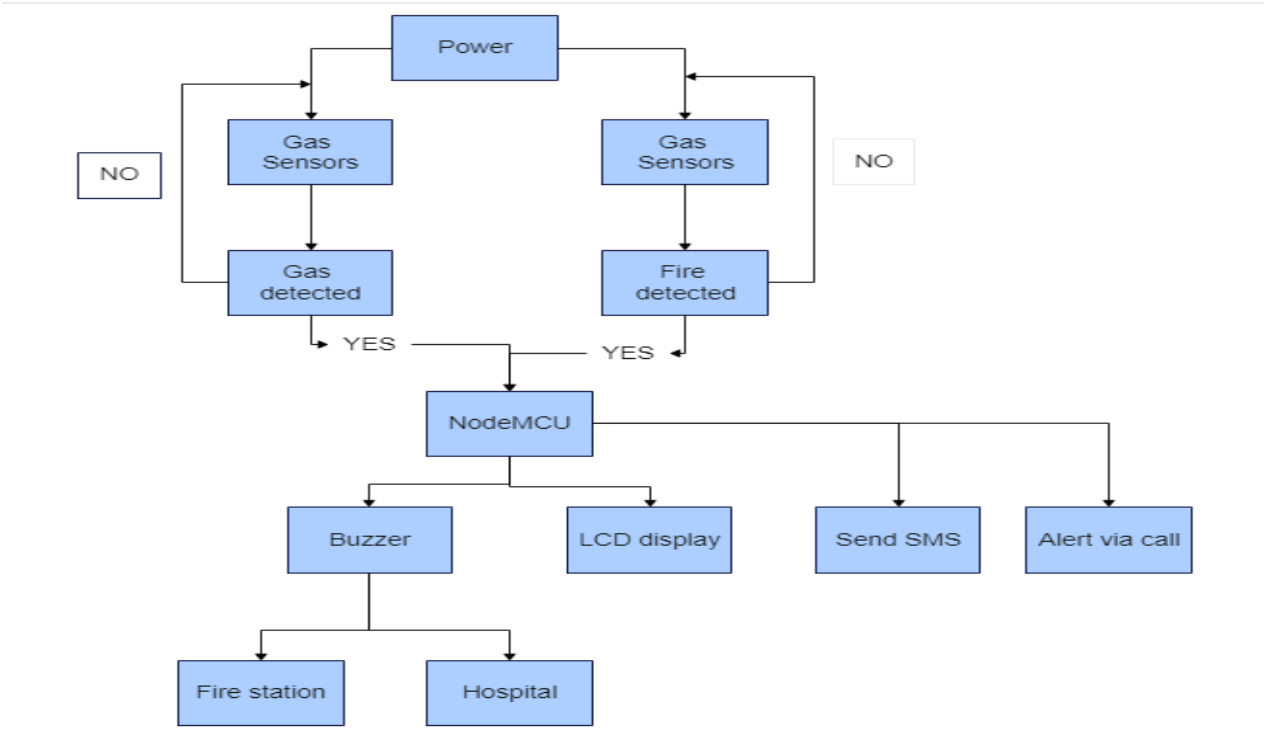
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User alert	Warnings must be sent to the user. Send the message as soon as possible
FR-2	User Understanding	The user could understand the amount and type of gas leaked and detect the location
FR-3	User controls	The user shall be able to turn off the electricity and other gadgets.
FR-4	User feasible	The user shall be able to notify the nearby fire station if gas leakage level is high.
FR-5	User location	The user shall be able to view the location of the gas leaked.

4.2 Non-Functional requirements

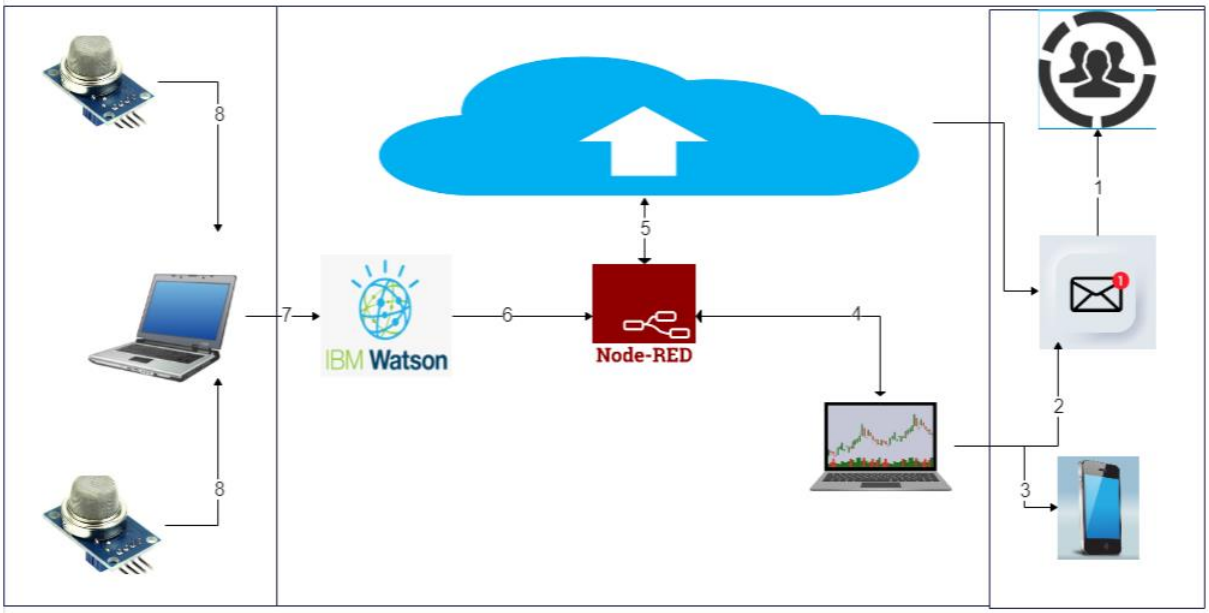
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Easy to operate and can be used effectively even by the uneducated people.
NFR-2	**Security**	The communication between the sensors and the simulator are secured using encryption.
NFR-3	**Reliability**	0% false alarming rate and able to get notifications through SMS, e-mail, or even through call.
NFR-4	**Performance**	Low latency and immediate response to the user and make immediate decision.
NFR-5	**Availability**	The system should work 24/7.
NFR-6	**Scalability**	The system can be used for domestic houses or even for large industry.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & 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	User can install the mobile application	I can access my account / dashboard	High	Sprint-1
		USN-2	User can register their details like email and mobile number	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-3	User can log on to the application using email and password.	I can Login into my application	High	Sprint-1
	Dashboard	USN-4	User can update their details like alternative mobile number etc..	I can view and change my details.	High	Sprint-2
		USN-5	User can view the gas level and the working condition of the sensors.	I can view the data given by the device	High	Sprint-2
Customer (Web user)	Usage	USN-1	User can register through the web	I can receive confirmation	High	Sprint-3

			page with e-mail or phone number.	email & click confirm		
		USN-2	User can log on to the web page using email and password.	I can Login into my application	High	Sprint-3
Customer	Working	USN-1	User can view the details	Act according to the alarm	Medium	Sprint-3
		USN-2	User can view the alert and turn off the power supply.	Act according to the alarm	High	Sprint-4
Customer Care Executive	Action	USN-1	Executer solves the user's problem	I can solve the issues	High	Sprint-4
Administrator	Administration	USN-1	Periodic check the condition of sensors and Stores the user's information	I can maintain the fault tolerance and error rate	High	Sprint-4

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

TITLE	DESCRIPTION	DUE DATE
Literature survey & Information gathering	Literature survey on the selected topic and collect information by referring to the related papers and research projects, journals etc.	3 September 2022
Prepare Empathy Map	Prepare empathy map canvas to understand about the user problems, pains and gains. From the empathised details, prepare the problem statements to be solved.	10 September 2022
Ideation	Conduct a brainstorming session with the teammates and discuss ideas to solve the problem. Prioritize the top 3 ideas based on feasibility.	17 September 2022
Proposed Solution	Prepare the proposed solution	24 September 2022

	which includes the novelty, feasibility, revenue, social impact, scalability etc.	
Problem Solution Fit	Prepare the problem solution fit which includes the causes, problems and solutions of the problem.	1 October 2022
Solution Architecture	Prepare solution architecture that indicates the data flow from the user, model and the website.	1 October 2022
Customer Journey	Prepare the customer journey map to understand the user needs and experience with the application.	8 October 2022
Functional Requirement	Prepare the functional requirement which includes all the features that will be available in the application.	15 October 2022
Technology Architecture	Prepare the technology architecture that defines about the technologies and the IBM cloud features used in the application.	15 October 2022
Data Flow Diagrams	Draw the data flow diagram to indicate the data flow from the user, during the model building and while predicting the result,	15 October 2022
Prepare Milestone & Activity List	Split the entire project into simpler tasks and prepare milestones and activity list of the project.	22 October 2022
Sprint Delivery Plan	Prepare a delivery plan of the project with specific due dates to complete each sprint consisting of a set of functional	22 October 2022

	requirements.	
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop, test and submit the code.	19 November 2022

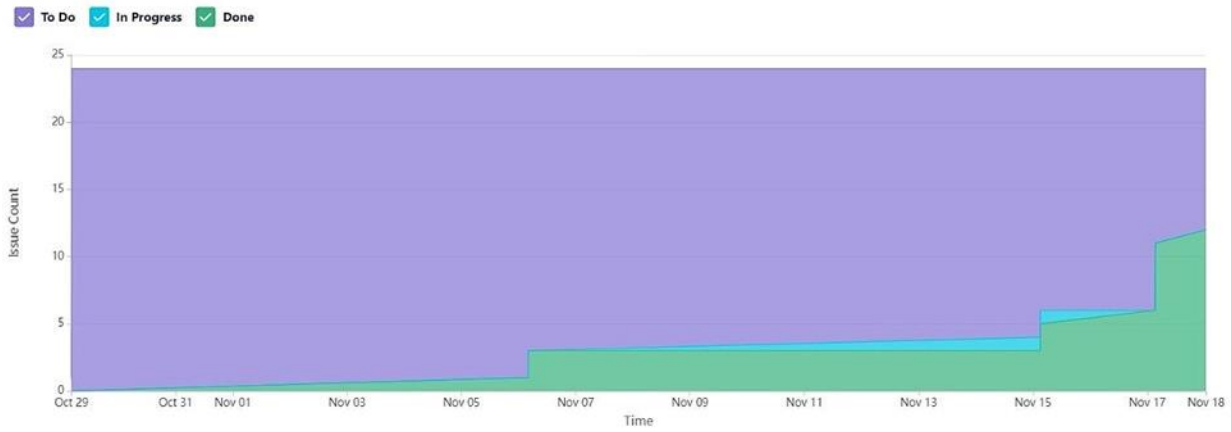
6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Objective	USN-1	The gas sensor should detect the gas	8	High	Bharani, Bharanidharan
Sprint-1	Features	USN-2	The values from the sensor should be displayed in the LCD screen	2	Medium	Bharani, Bharanidharan
Sprint-1	Features	USN-3	Once the detected gas reaches the threshold level, the red color LED should be turned ON.	5	High	Bharani, Bharanidharan
Sprint-1	Features	USN-4	As soon as the detected gas reaches the threshold level, the siren should be turned ON.	5	High	Bharani, Bharanidharan
Sprint-2	Focus	USN-5	The system should send the location where the gas is detected	8	High	Harish, Gomanishwaran
Sprint-2	Focus	USN-6	The system should also send the alerting SMS to the registered phone number	2	High	Harish, Gomanishwaran
Sprint-2	Features	USN-7	The gas leakage pipe should be closed automatically once it attains the threshold value	5	High	Harish, bharani
Sprint-2	Features	USN-8	The system will indicate that the gas leakage pipe is closed in the LCD screen and send SMS to the registered mobile number.	5	Medium	Gomanishwaran,Bharani

Sprint-3	Data Transfer	USN-9	The system should send the data of sensor values along with latitudes and longitudes to the IBM cloud	5	Medium	Harish, Bharanidharan
Sprint-3	Data Transfer	USN-10	The IBM cloud should send the data to Node-Red	5	Medium	Harish, Gomanish
Sprint-3	Data Transfer	USN-11	Data should be collected from the Node-Red and should be sent to the backend of the MIT app.	4	Medium	Harish, Gomanish
Sprint-3	Data Transfer	USN-12	The application should display the details of the gas level and other details to the user through the frontend of the MIT app.	7	High	Harish, Gomanish
Sprint-4	Registration	USN-13	User must first register their email and mobile number in the website	2	High	Gomanishwaran, Bharani
Sprint-4	Registration	USN-14	User should receive confirmation mail and SMS on registration	2	Medium	Gomanishwaran, Bharani
Sprint-4	Login	USN-15	User can login into the web application through email and password.	3	High	Harish, Bharanidharan

Sprint-4	Dashboard	USN-16	User can access the dashboard and make use of available resources.	2	Medium	Harish, Bharanidharan, Bharanidharan
Sprint-4	Focus	USN-17	User should receive an SMS once the leakage is detected.	5	High	Gomanishwaran, Bharani
Sprint-4	Allocation	USN-18	Admin must receive information about the leakage along with location and share exact location and route to the person.	3	High	Harish, Gomanish
Sprint-4	Allocation	USN-19	Admin must allot particular person to look after the leakage in a particular location.	3	High	Harish, Gomanish

6.3 Reports from JIRA



7. CODING & SOLUTIONING

7.1 Feature 1

Fire alert using red light and web alert.

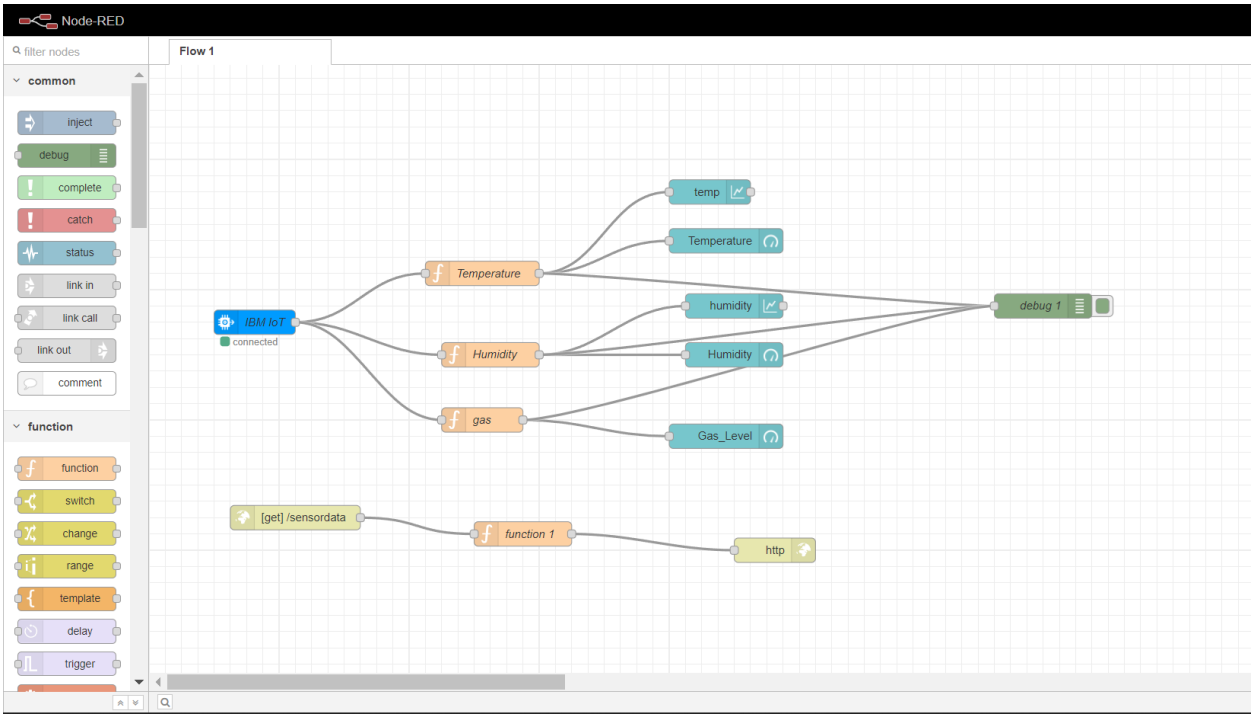
```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHTesp.h"
#include<stdio.h>
#include <stdlib.h>
#define LED 2
const int DHT_PIN = 15;
DHTesp dhtSensor;
int gas;
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
#define ORG "oyi7sh"
#define DEVICE_TYPE "Gas_leakage"
#define DEVICE_ID "154555"
#define TOKEN "WoOgbWlz4q-F4KQKc-"
String data3;
IPAddress myDns(127, 0, 0, 53);
char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wificlient;
PubSubClient client (server, 1883, callback,wificlient);

void setup()
{
  Serial.begin(115200);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  pinMode(LED, OUTPUT);
  delay(10);
  wificonnect();
  mqttconnect();
}
void loop()
{
    TempAndHumidity  data = dhtSensor.getTempAndHumidity();
    gas=random(10000);
    Serial.println("Temp: " + String(data.temperature, 2) + "°C");
    Serial.println("Humidity: " + String(data.humidity, 1) + "%");
```

```
48      Serial.println("gas_val " + String(gas));
49      PublishData(String(data.temperature,2),String(data.humidity,
    1),String(gas),int(data.temperature),int(data.humidity),int(gas));
50      delay(1000);
51      if (!client.loop()) {
52        mqttconnect();
53      }
54  }
55  void PublishData(String temp,String hum,String gas1,int temp1,int hum1,int
    gas2)
56  {
57    mqttconnect();
58    if (gas2>2000)
59    {
60      digitalWrite(LED, HIGH);
61      Serial.println("Fire alert");
62    }
63    else
64    {
65      digitalWrite(LED, LOW);
66      Serial.println("Normal");
67    }
68    String payload = "{\"temperature\":";
69    payload += temp;
70    payload += "," "\"humidity\":\"";
71    payload += hum;
72    payload += "\"";
73    payload += "," "\"gas_level\":\"";
74    payload += gas1;
75    payload += "\"}";
76
77    Serial.print("Sending payload: ");
78    // Serial.println(payload);
79    if (client.publish(publishTopic, (char*) payload.c_str()))
80    {
81      Serial.println("Data sent successfully");
82    }
83    else
84    {
85      Serial.println("Data sent failure");
86    }
87    Serial.println("---");
88  }
89  void mqttconnect()
90  {
```

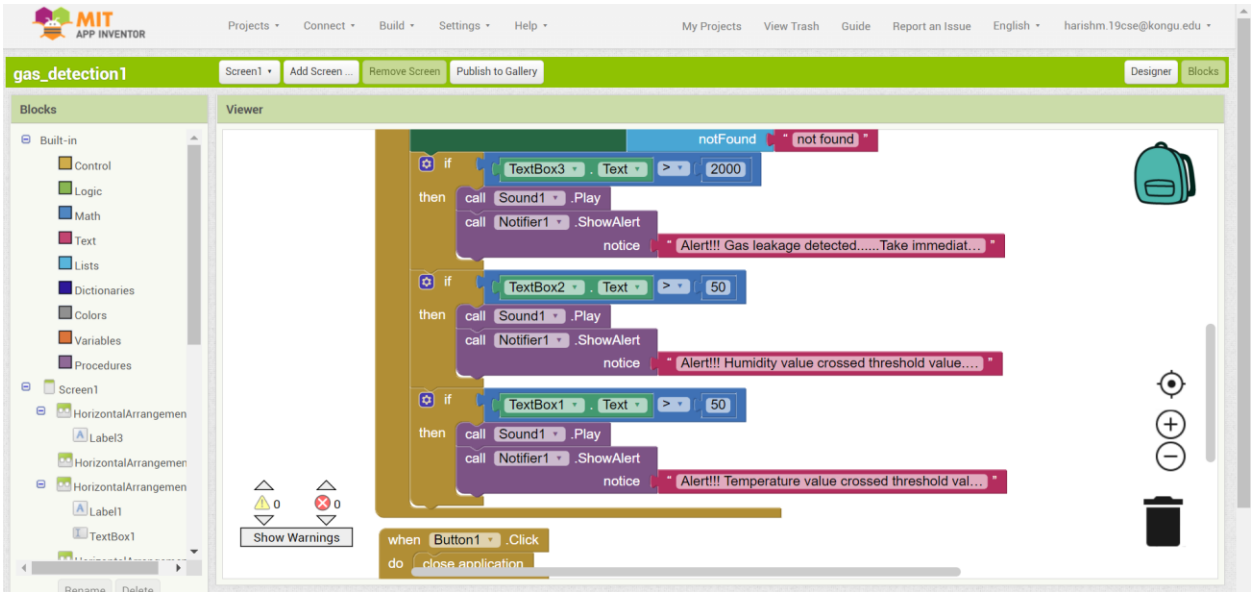
```
  if (!client.connected())
  {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token))
    {
      Serial.print(".");
      delay(500);
    }
    initManagedDevice();
    Serial.println();
  }
}
void wificonnect()
{
  Serial.println();
  Serial.print("Connecting to ");
  WiFi.begin("Wokwi-GUEST", "", 6);
  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);
```

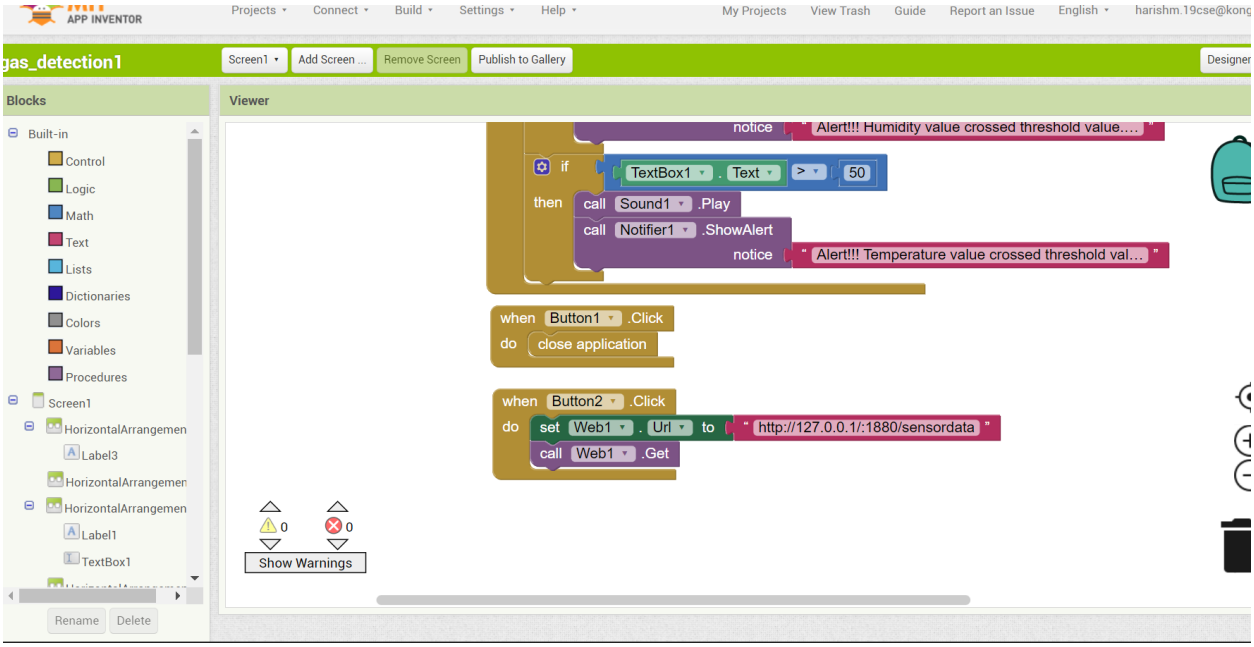
```
135  for (int i = 0; i < payloadLength; i++)
136  {
137    data3 += (char)payload[i];
138  }
139  data3="";
140}
```

7.2 Feature 2
Mobile app notification




MIT APP INVENTOR
Projects
Connect
Build
Settings
Help
My Projects
View Trash
Guide
Report an Issue
English
harishm.19cse@kongu.edu
gas_detection1
Screen1
Add Screen ...
Remove Screen
Publish to Gallery
Designer
Blocks
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
HorizontalArrangemen
Label3
HorizontalArrangemen
HorizontalArrangemen
Label1
TextBox1
Rename
Delete
Show Warnings
when Clock1 .Timer
do set Web1 . Url to " http://127.0.0.1/:1880/sensordata "
call Web1 .Get
when Web1 .GotText
url responseCode responseType responseContent
do set TextBox1 . Text to get responseContent
set responseContent to call Web1 .JsonTextDecode
jsonText get responseContent
notFound " not found "
set TextBox2 . Text to look up in pairs key " humidity "
pairs call Web1 .JsonTextDecode
jsonText get responseContent
notFound " not found "
set TextBox3 . Text to look up in pairs key " gas "
pairs call Web1 .JsonTextDecode
jsonText get responseContent


MIT APP INVENTOR
Projects
Connect
Build
Settings
Help
My Projects
View Trash
Guide
Report an Issue
English
harishm.19cse@kongu.edu
gas_detection1
Screen1
Add Screen ...
Remove Screen
Publish to Gallery
Designer
Blocks
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
HorizontalArrangemen
Label3
HorizontalArrangemen
HorizontalArrangemen
Label1
TextBox1
Show Warnings
notFound " not found "
if TextBox3 . Text > 2000
then call Sound1 .Play
call Notifier1 .ShowAlert
notice " Alert!!! Gas leakage detected.....Take immediat... "
if TextBox2 . Text > 50
then call Sound1 .Play
call Notifier1 .ShowAlert
notice " Alert!!! Humidity value crossed threshold value.... "
if TextBox1 . Text > 50
then call Sound1 .Play
call Notifier1 .ShowAlert
notice " Alert!!! Temperature value crossed threshold val... "
when Button1 .Click
do close application



8.TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Expected Result	Actual Result	Status	links	Executed By
TC_OO1	Functional	IBM cloud	Create the IBM Cloud services which are being used in this project.	IBM Cloud Login ID & Password	1.Go to IBM Cloud signup page 2.Enter e-mail id and other credentials 3.Enter a password	User should sign up IBM cloud and details should be verified	Working as expected	Pass	https://cloud.ibm.com/login	Bharanidharan
TC_OO2	Functional	IBM Cloud	Configure the	IBM Cloud	1.Go to Cloud login	User login to	Working as exp	Pass	https://cloud.ibm.com/login	Bharani

			IBM Cloud services which are being used in completing this project.	Login ID & Password	2.Enter user ID & Password 3.Verify login by the popup display	IBM Cloud and should be navigated to IBM Cloud dashboard page	ected			
TC_OO3	Functional	IBM Watson IoT Platform	IBM Watson IoT platform acts as the mediator to connect the web application to IoT devices,	IBM Watson IoT Platform Login ID & Password	1.Login to IBM Cloud 2.Click Catalog 3.Search IoT and click create 4.Go to resource list and search Internet of Things platform 5.Press Launch	User should be navigated to IBM IoT Watson Platform	Working as expected	Pass	https://oyi7sh.internetofthings.ibmcloud.com/dashboard/	Harish

			so create the IBM Watson IoT platform.		and click Sign in IBM Watson Platform					
TC_OO4	Functional	IBM Watson	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.	IBM Watson IoT Platform Login ID & Password	1.Login to IBM Watson Platform 2. Click Add Device 3.Enter the details and click Finish. Create Device ID & Device type 4.Turn on Device Simulator and click simulation running. Enter the values of gas,	Temperature, Humidity and Gas sensor values should be randomly generate	Working as expected	Pass	Temperature, Humidity and Gas sensor values are generated randomly in simulation	Bharanidharan

					temperature & humidity level 5.Click Send & Save. Verify the displayed result of the levels					
TC_OO5	Functional	IBM Cloud(Node Red)	Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platf	Node Red Installation	1.Install node red and open node red in command prompt 2.Select IBM input in IoT	User should be able to see the Node Red page	Working as expected	Pass	https://cloud.ibm.com/developer/appservice/create-app?starterKit=59c9d5bd-4d31-3611-897a-f94eea80dc9f&defaultLanguage=undefined	Bharani

			orm.							
TC_OO6	Functional	Node Red	Create a Node-RED service.	Node Red Installation	1.Select IBM IoT input in Node. In IBM IoT Watson Platform, go to apps and click on generate API keys. 2.Copy & paste generated API key and token in the IBM IoT input. After entering all details, click the done button. 3.Add debug to the IBM IoT and rename as Msg.payload and	Values of sensors and button for Alarm & Sprinkler ON/OFF should be displayed	Working as expected	Pass	Values of sensors and button for Alarm & Sprinkler ON/OFF is displayed	Harish

					click on done. Click gauge from the dashboard and fill the details & add functions to the gauge. Check the generated values from the debug message. 4.Edit function node, connect them, add another gauge and functions, name them as "Temperature", "Gas" &"Humidity"					

					5.Finally add alarm ON/OFF and Sprinkler ON/OFF buttons to the IBM IoT and debug. Verify the output from NODE RED using Local host link					
TC_OO7	Functional	Python 3.7.0	Develop a python script to publish random sensor data such as temperature, humidity level and	Python 3.7.0 (64 bit) installation	1.Download and install Python 3.7.0 2.Develop python code	User should be able to develop a python code	Working as expected	Pass	https://www.python.org/downloads/release/python-370/	Harish

			Gas level to the IBM IoT platform							
TC_OO8	Functional	Python 3.7.0	After developing python code, commands are received just print the statements which represent the control of the devices.	Python 3.7.0 (64 bit) installation	1.Downl install Python 3.7.0 2.After python code	User should be able to get the results from the developed code	Working as expected	Pass	Get the output from the code	Gomanishwaran
TC_OO9	Functional	IBM Cloudant DB	Publish Data to The IBM Cloud	IBM Cloud Login ID & Password	1.Run the python code 2.Verify the displayed output	User should be able to publish the code	Working as expected	Pass	Publishment of python code	Gomanishwaran
TC_OO10	Web UI	Node Red	Create Web	MIT Inve	1.Go to Node	Sensors	Working as	Pass	Sensors values and command values can be seen in the mobile	Gomanishwaran

		& MIT Inventor	UI in Node - Red	ntor Login ID & password	Red. Select http in & http response. Add functions and select another http in and http response. Connect them to IBM IoT output and function .Print the command statements such as Sprinkler ON/OFF, Alarm ON/OFF and sensor 2.Go to MIT app inventor and create fronten	values and command values should be seen in the mobile application	expected		application	

					d using buttons, horizontal arrangement, text bar, etc. Add blocks and so on to create back end. Verify the output					
TC_OO11	Functional	IBM Cloudant DB	Configure the Node-RED flow to receive data from the IBM IoT platform and also use Cloudant DB nodes to store the receiv	IBM Cloud Login ID & Password	1.Go to IBM cloud, search Cloudant in Catalog, Add new dashboard, go to Node Red 2.Connect to cloudant and verify the results	User should be able to connect the Cloudant and Node Red	Working as expected	Pass	Cloudant is connected by NODE RED	Harish

			ed senso r data in the cloud ant DB							

8.2 User Acceptance Testing

1. *Defect Analysis*

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	6	3	2	2	13
Duplicate	1	0	3	0	4
External	2	2	0	1	5
Fixed	7	3	4	5	19
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	16	13	13	10	52

2. *Test Case Analysis*

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5
Client Application	9	0	0	9
Security	3	0	0	3

Outsource Shipping	1	0	0	1
Exception Reporting	2	0	0	2
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULTS

9.1 Performance Metrics

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Load/Volume Changes	Risk Score	Justification
1	Alarm ON/OFF	Existing	Low	No Changes	Low	>5 to 10%	GREEN	Changes occurs less
2	Sensor values	Existing	Moderate	No Changes	Moderate	>10 to 30%	ORANGE	Some changes occurs

S.No	Project Overview	NFT Test approach	Approvals/SignOff	Assumptions/Dependencies/Risks
1	.ino(ardunio)	ino coding	wokwi.com	Depend on the delivered code
2	Node Red	Sensor & command values	https://nodered.org/	Sensor values
3	MIT Inventor	Alarm/Sprinkler/Sensors notification	https://appinventor.mit.edu/about/termsofservice	Notifications

S. No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Identified Defects (Detected/Closed/Open)	Recommendations	Approvals/SignOff
1	.ino(ardunio)	ardunio coding	Met	Pass	GO	Closed	Efficient code	wokwi.com
2	Node Red	Sensors&command values	Met	Pass	GO	Closed	Sensing the values perfectly	https://nodered.org/
3	MIT Inventor	Alarm/Sprinkler/Sensors notification	Met	Pass	GO	Closed	Notifies the users at correct time	https://appinventor.mit.edu/about/termsofservice

10.ADVANTAGES & DISADVANTAGES

Advantages:

- Detect the concentration of the gases
- The sensor-enabled solution helps prevent the high risk of gas explosions and affecting any casualties within and outside the premises.
- Get real-time alerts about the gaseous presence in the atmosphere
- Prevent fire hazards and explosions
- Supervise gas concentration levels
- Ensure worker's health
- Real-time updates about leakages
- Cost-effective installation
- Data analytics for improved decisions
- Measure oxygen level accuracy
- Get immediate gas leak alerts

Disadvantages:

- Only one gas can be measured with each instrument.
- When heavy dust, steam or fog blocks the laser beam, the system will not be able to take measurements

11.CONCLUSION

Gas leaks cause serious disasters that result in property damage and human injuries. The main causes of gas leaks are poor equipment upkeep and a lack of public awareness. As a result, detecting LPG leaks is critical for avoiding accidents and saving human lives. This paper discussed a system for detecting and alerting LPG leaks. Whenever LPG leakage is detected, this device activates an LED and a buzzer to inform people. This approach is straightforward but dependable. Internet of Things has gained its wide popularity in recent days due to its various streams of applications which has paved way for smooth, safe and easier mode of living style for human beings. One such area of applications includes gas booking and gas leakage detection for both domestic and commercial purposes. Though, several techniques is existing for the same, yet gas leakage detection is one major concern and a challenge.

12.FUTURE SCOPE

In the future, instead of using AC power, the gas leakage detecting system might be created using photovoltaic panels with a battery as a backup power supply to give a continuous supply, as opposed to the current use of AC power. The protection system employs a combination of MQ6 gas sensors, DHT22 temperature sensors, load sensors, smoke and flame sensors, and PIR sensors. A number of sensors must be calculated, taking into account the room's volume, installation position, and other factors. This system assures that if a gas leak happens, it can be tracked more effectively and that occupants may be notified ahead of time, regardless of whether the leak is visible or not, whether the house is vacant or occupied. The best recommendation for a monitoring system is to utilize a WiFi module that allows the user to monitor the gas level in real-time and automate direct management of the safety device system if an unanticipated occurrence occurs. Finally, the safety device employed was the most vital and important aspect. We also suggested that a tripper circuit be built, which would automatically turn off the (MSB) in the event of a fire, and turn off the gas regulator valve via a solenoid valve either from the cylinder from the main switchboard, If an incident occurs, it automatically can switch on the exhaust fan to suck gas to the outside house and sound an alarm and audio buzzer to inform the user or persons around and the user can opening the window, This device monitors the gas and detects any leaks in order to keep people safe.

13.APPENDIX

Source Code

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHTesp.h"
#include<stdio.h>
#include <stdlib.h>
#define LED 2
const int DHT_PIN = 15;
DHTesp dhtSensor;
int gas;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
#define ORG "oyi7sh"
#define DEVICE_TYPE "Gas_leakage"
#define DEVICE_ID "154555"
#define TOKEN "WoOgbWlz4q-F4KQKc-"
String data3;
IPAddress myDns(127, 0, 0, 53);
```

```
char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wificlient;
PubSubClient client (server, 1883, callback,wificlient);

void setup()
{
  Serial.begin(115200);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  pinMode(LED, OUTPUT);
  delay(10);
  wificonnect();
  mqttconnect();
}
void loop()
{
    TempAndHumidity  data = dhtSensor.getTempAndHumidity();
    gas=random(10000);
    Serial.println("Temp: " + String(data.temperature, 2) + "°C");
    Serial.println("Humidity: " + String(data.humidity, 1) + "%");
    Serial.println("gas_val " + String(gas));
    PublishData(String(data.temperature,2),String(data.humidity,
1),String(gas),int(data.temperature),int(data.humidity),int(gas));
    delay(1000);
    if (!client.loop()) {
      mqttconnect();
    }
}
void PublishData(String temp,String hum,String gas1,int temp1,int hum1,int gas2)
{
  mqttconnect();
  if (gas2>2000)
  {
    digitalWrite(LED, HIGH);
    Serial.println("Fire alert");
  }
  else
  {
    digitalWrite(LED, LOW);
    Serial.println("Normal");
  }
```

```
  String payload = "{\"temperature\":";
  payload += temp;
  payload += "," "\"humidity\":\"";
  payload += hum;
  payload += "\"";
  payload += "," "\"gas_level\":\"";
  payload += gas1;
  payload += "\"}";

  Serial.print("Sending payload: ");
  // Serial.println(payload);
  if (client.publish(publishTopic, (char*) payload.c_str()))
  {
    Serial.println("Data sent successfully");
  }
  else
  {
    Serial.println("Data sent failure");
  }
  Serial.println("---");
}
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()
{
  Serial.println();
  Serial.print("Connecting to ");
  WiFi.begin("Wokwi-GUEST", "", 6);
  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++)
  {
    data3 += (char)payload[i];
  }
  data3="";
}
```

14.GitHub & Project Demo Link

Github Link : https://github.com/IBM-EPBL/IBM-Project-23363-1659880680

Demonstration video Link : https://drive.google.com/file/d/1csUbOcnjt-c4oyDLuFJHyAQO-JrOvd6P/view?usp=sharing