GAS LEAKAGE MONITORING AND ALERTING SYSTEM FOR INDUSTRIES

Team ID: PNT2022TMID35867

1.INTRODUCTION

1.1.PROJECT OVERVIEW

Gas leakage is one of the major causes of the fire accidents in industries. Hydrogen Sulphide(H2S) ,one of the hazardous gases which is used and produced in oil, gas and chemical industries, is very corrosive, flammable, highly toxic in nature. Even a mild inhale of this gas leads to headache. Exposure on high concentrations can lead to eye irritation, nausea, throat irritation, nausea, unconsciousness and even death. Hence it is necessary to alert the workers in those industries when a toxic gas is released, so that they can evacuate the gas prone area very quickly. Also, the technician should be able to close the main gas valve when a high toxicity of gas is detected.

Our project "Gas Leakage Monitoring and Alerting System for Industries" aims to alert the industrial workers, when a toxic gas is detected. Gas sensor is used for the detection of toxic gases and ESP8266 NodeMCU microcontroller is used for processing. When a toxic gas is detected, gas sensor detects the gas concentration and the microcontroller reads and processes it. The alarm system activates, when a high concentration of gas is detected. The microcontroller is connected to the IBM Watson IoT platform to display the gas concentration value in cloud. An SMS is triggered using the IFTTT web service to the technician when a high toxicity of gas is detected. Also, mobile app is deployed for the workers to view the gas concentration value and toxicity level.

1.2.PURPOSE

The main purpose of the project is to design a prototype that detects presence of toxic gases. It can prevent fire accident & financial losses. The device provides immediate response once gas leakage is detected with alarm system & SMS Alert (In case of High Toxicity Level). The project detects gas leakage by continuous monitoring and trigger alarm system for Moderate gas leakage level (Evacuate the prone zone) and send SMS alert to Technician (in case of High toxicity level) to close respective gas valve immediately. Also, display gas concentration level in Mobile App (with Login Credentials) and Web UI. The Gas sensor data & login Credentials are stored in Cloud database for Future Reference & Login Authentication about gas Concentration level.

2.LITERATURE SURVEY

2.1.EXISTING PROBLEM

TITLE	LITERATURE	DESCRIPTION	LIMITATION
Kodali, R. K., Greeshma, R. N. V., Nimmanapalli, K. P., & Borra, Y. K. Y. (2018, December). IOT based industrial plant safety gas leakage detection system. In *2018 4th international conference on computing communication and automation (ICCCA)* (pp. 1-5). IEEE.	IEEE 2018	The proposed system uses MQ6, MQ4 and MQ135 gas sensors which detect LPG, Methane and Benzene gas leaks respectively and uses ESP-32 as a Wi-Fi module. The sensor's data is stored in UBIDOTS cloud server and IFTTT is used for SMS.	The alarm indication for the detection of gases is not provided. There is no mobile application designed for the industrial worker to monitor the gas concentration.
Zinnuraain, S. M., Hasan, M., Hakque, M. A., & Arefin, M. M. N. (2019, March). Smart gas leakage detection with monitoring and automatic safety system. In *2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)* (pp. 406-409). IEEE.	IEEE 2019	The proposed system uses MQ2 sensor to detect the Liquified Petroleum Gas leakage and uses ATMEGA-2560 microcontroller. BLYNK mobile application is used to display the data collected and LCD is used to display the same.	Non-hazardous and non-toxic gas is used for the detection. Alert SMS service and mobile application hasn't deployed. Alarm system is not used.
Amsaveni, M., Anurupa, A., Preetha, R. A., Malarvizhi, C., & Gunasekaran, M. (2015). Gsm based LPG leakage detection and controlling system. *The International Journal of Engineering and Science (IJES) ISSN (e)*, 2319-1813.	IJES 2015	The proposed system uses MQ6 sensor for LPG gases leakage and PIC microcontroller is used. The buzzer produces an alarm to indicate the gas leakage. Then, the user is alerted by SMS through the GSM module.	Non-hazardous gas is used for detection. Cloud storage is not implemented. Mobile app is not deployed.
Subramanian, M. A., Selvam, N., Rajkumar, S., Mahalakshmi, R., & Ramprabhakar, J. (2020, January). Gas Leakage Detection System using IoT with integrated notifications using Push bullet-A Review. In *2020 Fourth International Conference on Inventive Systems and Control (ICISC)* (pp. 359-362). IEEE.	IEEE 2020	The proposed system uses MQ5 sensor for detection and uses ESP8266 Node-MCU Wi-fi module as microcontroller. Thing Speak -A cloud-based database collection system is used to store user's data Through this tool, push bullet is configured which is used for data transfer from mobile to computer depending on the file size and is used to receive notifications on the web/mobile.	Non-toxic gas is detected. Mobile application is not deployed.

2.2.REFERENCE

1. Kodali, R. K., Greeshma, R. N. V., Nimmanapalli, K. P., & Borra, Y. K. Y. (2018, December). IOT based industrial plant safety gas leakage detection system. In *2018 4th international conference on computing communication and automation (ICCCA)* (pp. 1- 5). IEEE.

2. Zinnuraain, S. M., Hasan, M., Hakque, M. A., & Arefin, M. M. N. (2019, March). Smart gas leakage detection with monitoring and automatic safety system. In *2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)* (pp. 406-409). IEEE.

3. Suma, V., Shekar, R. R., & Akshay, K. A. (2019, June). Gas leakage detection based on IOT. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 1312-1315). IEEE.

4. Amsaveni, M., Anurupa, A., Preetha, R. A., Malarvizhi, C., & Gunasekaran, M. (2015). Gsm based LPG leakage detection and controlling system. *The International Journal of Engineering and Science (IJES) ISSN (e)*, 2319-1813

5. Subramanian, M. A., Selvam, N., Rajkumar, S., Mahalakshmi, R., & Ramprabhakar, J. (2020, January). Gas Leakage Detection System using IoT with integrated notifications using Push bullet-A Review. In *2020 Fourth International Conference on Inventive Systems and Control (ICISC)* (pp. 359-362). IEEE

6. Vijayalakshmi, J., G. Puthilibhai, and SR Leoram Siddarth. "Implementation of Ammonia Gas Leakage Detection & Monitoring System using Internet of Things." *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. IEEE, 2019.

7. Yahaya, S. Z., et al. "IoT Based System for Monitoring and Control of Gas Leaking." *2020 1st International Conference on Information Technology, Advanced Mechanical and Electrical Engineering (ICITAMEE)*. IEEE, 2020

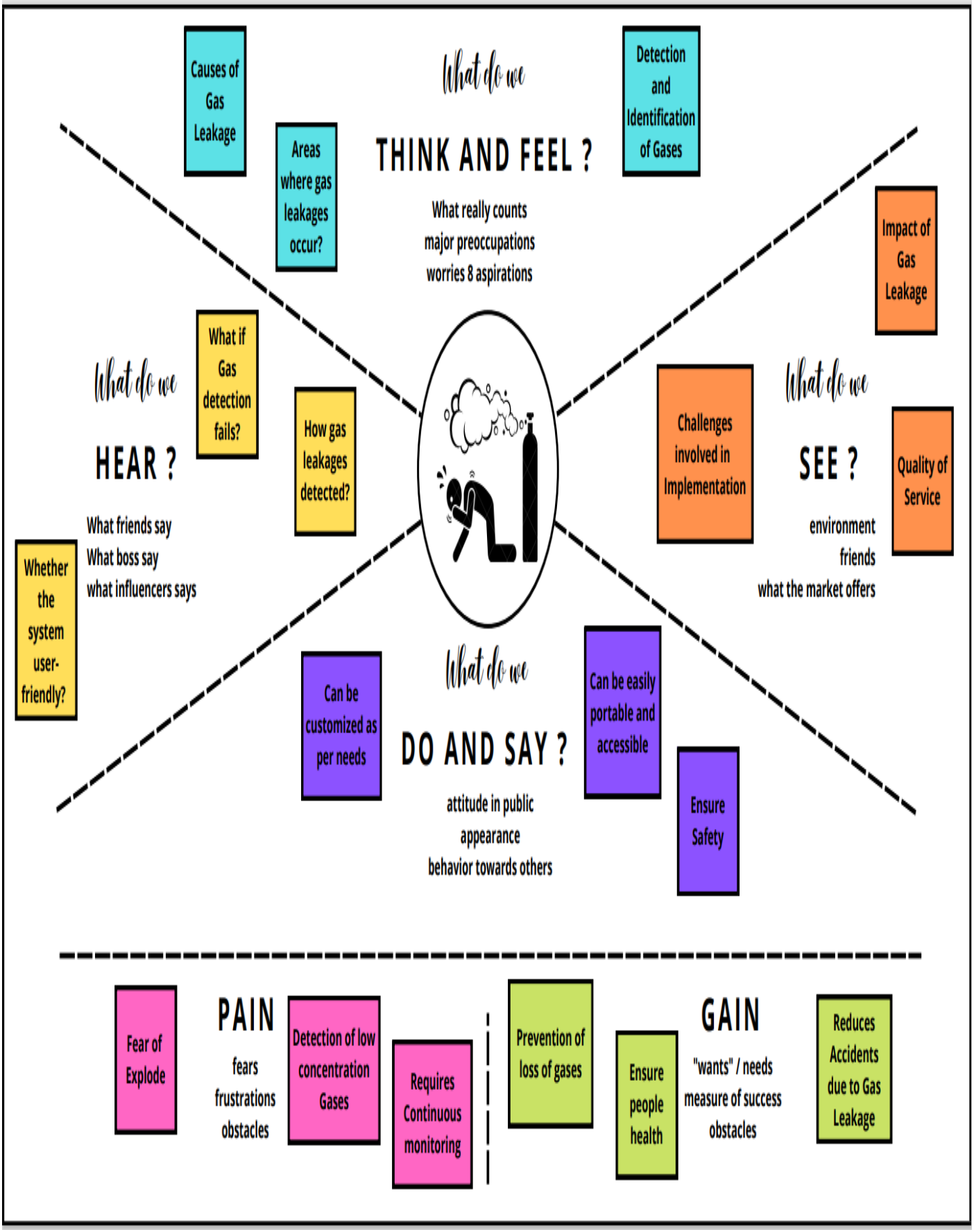
8. Fahim, Md Ibtida, et al. "Design of an IoT Based Gas Wastage Monitoring, Leakage Detecting and Alerting System." *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2021.

2.3.PROBLEM STATEMENT DEFINITION

Problem Statement (PS)	**I am**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	Industrialist	Get User friendly Gas Monitoring & Alerting System	With better usage of Resource & providing Quality of Service	There are huge losses on investment & life	Providing health safety Assurance
PS-2	Industrialist	Get easily Accessible Gas Monitoring & Alerting System	Reducing latency on QoS and Improve machine stability & Reducing cost and Maintenance	It reduces malfunction of system & Saves life	Ensured protected Environment

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)	In Industries, Gas Leakage causes a variety of consequences, including financial loss and human life. This device will aid in the detection of gas leakage and alert people of prone zone.
2.	Idea / Solution description	To design a model that can detect the gas leakage using gas sensor and measure the humidity and temperature which will be uploaded on the IBM cloud for cloud computation and to determine the level of gas leakage and alert the concerned person depending upon the gas leakage via Node red Dashboard. Additionally, sent notification of gas leakage to intended/concerned person via MIT App Inventor.
3.	Novelty / Uniqueness	System provides alert message about the hazardous gas presence in the atmosphere with continuous monitoring and provides real-time updates about gas leakage with location.
4.	Social Impact / Customer Satisfaction	Fire, Suffocation, Death and explosion risks are all determined by physical attributes like flammability and toxicity in industries. This device detect gas leakage and alert message is given prior to the safety limit; it provides enough time to evacuate workers from the particular area. The rate of risk to workers and costly machineries gets lowered.
5.	Business Model (Revenue Model)	Receive instant notifications of the presence of gases in the atmosphere by App-inventor which saves life and losses. Prevent explosions and fire dangers if there's preventive/control measures are available readily once gas leakage detected. Monitor the levels of gas concentration dynamically & maintain assured health status of the workspace. Installation and Maintenance is economic and eco-friendly.
6.	Scalability of the Solution	System can be deployed in any industries by deciding appropriate Gas sensor.

3.4.PROPOSED SOLUTION FIT

<table>
<tr>
<td>CUSTOMER SEGMENT(S)

1. Chemical Industry.
2. Semiconductor Manufacturing Industry.</td>
<td>CUSTOMER CONSTRAINTS

1. Particular gases can only be monitored and detected by the device.
2. In case of network disconnectivity, continuous monitoring of gases and location may not be provided through alert SMS and MIT App inventor.</td>
<td>AVAILABLE SOLUTIONS

1. Alert message can be provided in case of the presence of hazardous gases in the industry with continuous monitoring.
2. By providing real-time updates about gas leakage with location.</td>
</tr>
<tr>
<td>JOBS TO BE DONE/PROBLEMS

1. Health issues like respiratory problems, eye and throat irritation.
2. Affects the growth of plants.
3. Financial loss</td>
<td>PROBLEM ROOT CAUSE

1. Improper installation of tube fittings.
2. Unfit material selection for piping.
3. Loose joints and sealings.
4. Lack of inspection and maintenance of pipes.</td>
<td>BEHAVIOUR

1.Detect the gas leakage using appropriate sensors.
2.Provide continuous monitoring of leakage.
3.Provide real-time data with location.</td>
</tr>
<tr>
<td>TRIGGERS

1.Can identity the particular gas leakage.
2.Provide alert with the exact location
3.Provides continuous monitoring of gas leakage.

EMOTIONS: BEFORE/AFTER

BEFORE:

No monitoring of gas leakage, Causes Health issues & financial losses.

AFTER:

Continuous monitoring of gases and low risks of health issues and financial losses</td>
<td>YOUR SOLUTION

1. Detection of the gas leakage using gas sensor and measure the humidity and temperature which will be uploaded on the IBM cloud for cloud computation.

2. Determine the level of gas leakage and alert the concerned person depending upon the gas leakage via Node red Dashboard.

3. Additionally, send notification of gas leakage to intended/concerned person via MIT App inventor.</td>
<td>CHANNELS OF BEHAVIOUR

ONLINE:

Can monitor the real time gas leakage values through the mobile application and also alert message is provided in prior.

OFFLINE:

Gas leakage values is provided through sensor through continuous monitoring and overcome financial losses and health issues.</td>
</tr>
</table>

4.REQUIREMENT ANALYSIS

4.1.FUNCTIONAL REQUIREMENTS

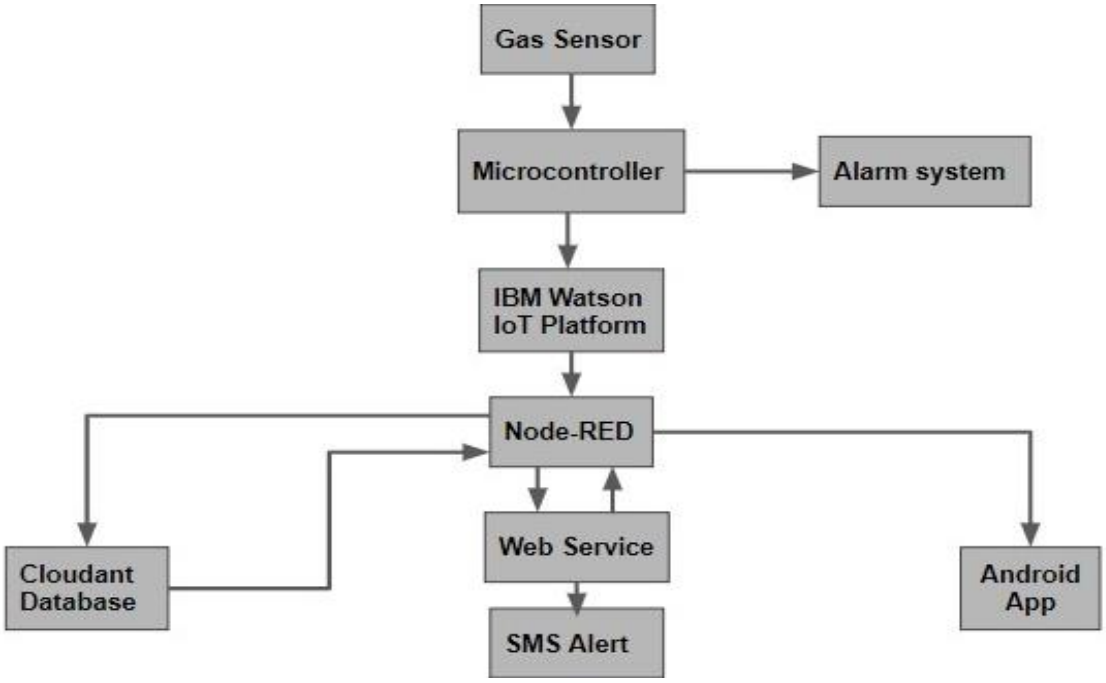
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Design of Gas Detector	This function shall provide reliable detection of flammable and toxic leaks before accident which reaches Toxic concentration that cause Industrial risk of losses.
FR-2	Location of Gas Detection System	This shall be positioned in different levels in an area or module. When gas leakage detected, Alert message with Location of Leakage is received by User.
FR-3	Action of Gas Detection and Alert System	The alarm starts ringing when the gas gets leaked, Doors will be opened automatically (Optional) and the data's will be stored in cloud automatically dynamically and alert message will be Delivered to User's Mobile (especially People in Prone Zone) and to App designed by App-Inventor.
FR-4	Calibration/Testing of Gas Detector	Gas detectors shall be individually identifiable with a self-test function. It shall ensure detection of the presence of intended gas concentration and amount of intended gas leakage.
FR-5	User Action & Performance	When the user gets notified in prior, he could performance prevention control and evacuate entire people in Accident Zone unit.

4.2.NON – FUNCTIONAL REQUIREMENTS

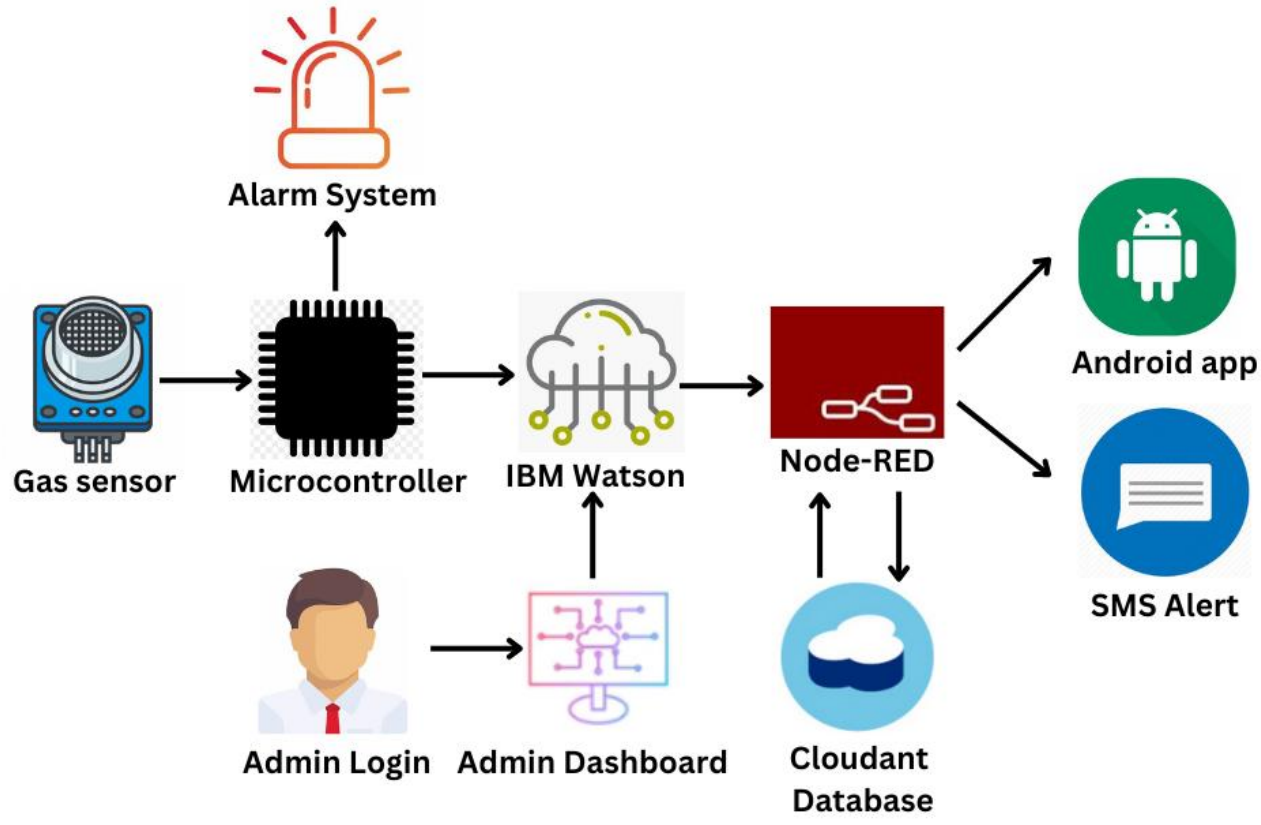
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It updates the data regularly by continuous monitoring thereby protects the workers and ensure protected environment.
NFR-2	Security	As a result of alert by Message to intended user, we can be able to protect both the humans and properties by performing prevention control methods.
NFR-3	Reliability	Can be able to provide accurate values of Gases. It might have a capacity to recognize system malfunction accurately and alerts user/technician to replace Malfunction detection system (Future work).
NFR-4	Performance	This device detects sensitive gases even to minor gas leaks. It performs Speedy operation and it send response faster (Low Latency).
NFR-5	Availability	It can be used for daily; it includes day and nights, because it provides continuous monitoring.
NFR-6	Scalability	Utilization of any Gas Sensors depending upon the Type of Gases utilized in Industries.

5.PROJECT DESIGN

5.1.DATAFLOW DIAGRAM



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 (Industry Worker)	Registration	USN-1	As a user, I can register for the application by entering my email, password and confirming my password.	I can access my account / dashboard	Medium	Sprint-4
Customer (Industry Worker)	Login	USN-2	As a user, I can log into the application by entering username, mobile number & password.	I can get access to dashboard.	High	Sprint-4
Customer (Industry Worker)	Monitor	USN-3	As a user, I can monitor the status of the gas leakage system.	I can view the status of gas leakage system.	High	Sprint-4
Customer (Line Workers)	Notification	USN-4	As a user, I can get (alarm system) alert about gas leakage.	I can get alert about gas leak.	High	Sprint-1
Customer (Technical Worker)	Notification	USN-5	As a user, I can get SMS notification & alarming alert about gas leakage.	I can get SMS alert about gas leakage & requested to close gas valve.	High	Sprint-3
Customer (Industry Worker)	Sign-Up	USN-6	As a user, I can sign-up using unique username, mobile number & password.	I can sign-up and get update about gas leakage.	Low	Sprint-4
Customer (Supervisor)	Sign-Up	USN-7	As a Supervisor, I can manage/monitor login database and sensor database.	Create database for sensor data and user login.	Low	Sprint-2
Supervisor	Web Service	USN-8	As a user, I can create web UI to display sensor data with alert message.	I can get web UI to display data about gas concentration.	Low	Sprint-2

6.PROJECT PLANNING & SCHEDULING

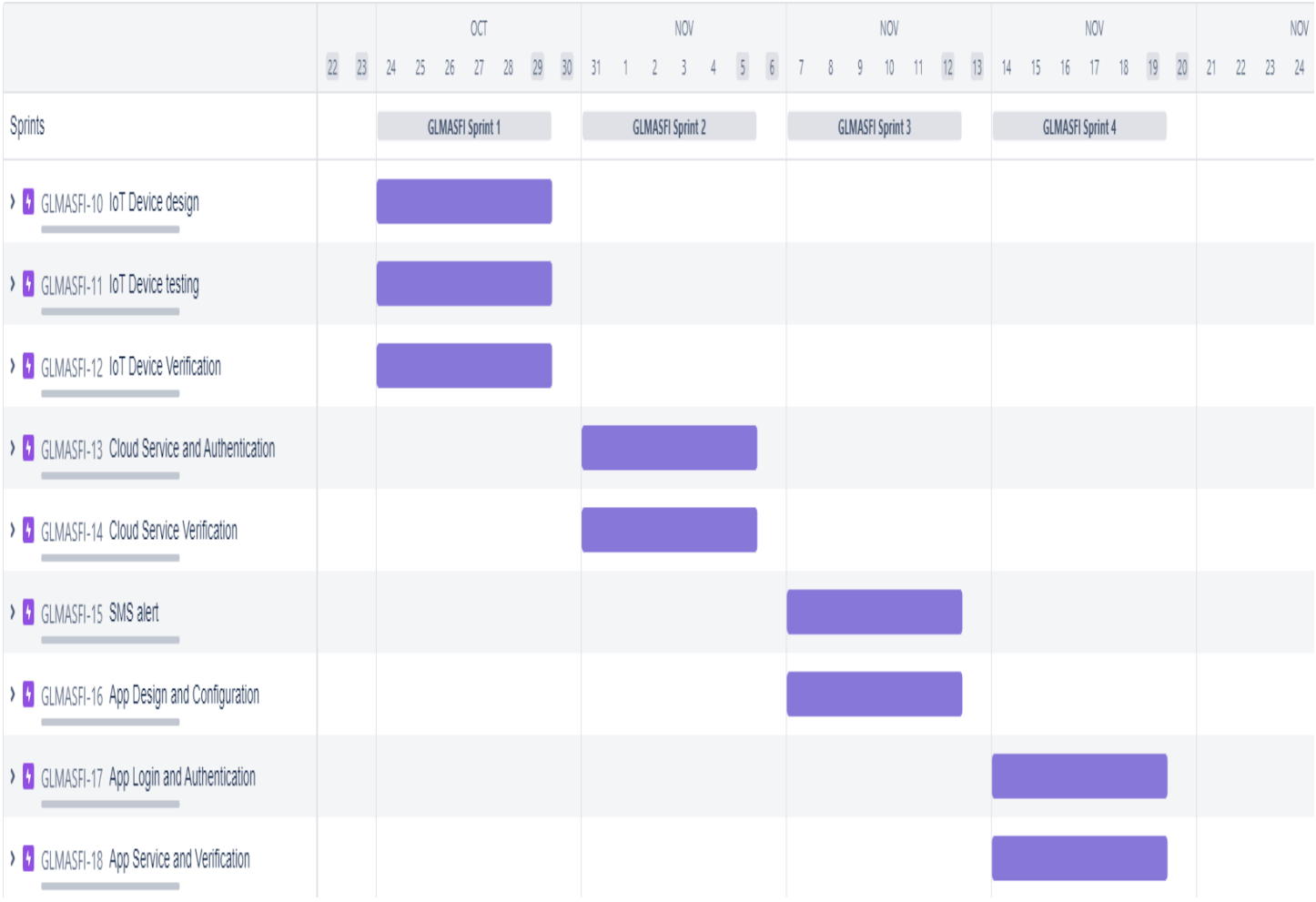
6.1.SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	IoT Device design	USN-1	Design system with gas sensor and microcontroller (Technology stack)	4	High	Prakash E, Joey Infant Rex A, Vasanth Kumar M S, Akhilesh Chandra
Sprint-1	IoT Device testing	USN-2	Testing system with different types of concentration of particular gas (Appropriate gas sensor)	5	High	Joey Infant Rex A, Akhilesh Chandra
Sprint-1	IoT Device Verification	USN-2	Configure Buzzer with Device and Verify Gas Leakage	1	Low	Prakash E, Vasanth Kumar M S
Sprint-2	Cloud Service and Authentication	USN-3	Create Cloud storage, configure cloud with IoT device (IoT platform) and Administered by User with credentials	3	Low	Joey Infant Rex A, Prakash E, Vasanth Kumar M S, Akhilesh Chandra
Sprint-2	Cloud Service Verification	USN-3	Updating cloud database with gas concentration always	4	Medium	Joey Infant Rex A, Prakash E,
Sprint-3	SMS alert	USN-4	Message "Alert" sent to intended user via SMS	5	High	Joey Infant Rex A, Prakash E, Vasanth Kumar M S, Akhilesh Chandra
Sprint-3	App Design and Configuration	USN-4	Design Alert App and Interface Gas Leakage details with Alert App	2	Low	Vasanth Kumar M S, Akhilesh Chandra
Sprint-4	App Login and Authentication	USN-5	Provide Username and Password to Intended User for Security Purpose	4	Medium	Prakash E, Akhilesh Chandra
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	App Service and Verification	USN-6	Provide Gas Leakage Location with Concentration to User via Alert App with Login Credentials	5	High	Joey Infant Rex A, Prakash E, Vasanth Kumar M S, Akhilesh Chandra

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	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3.REPORT FROM JIRA



7.CODING & SOLUTIONING

MQTT (Message Queuing Telemetry Transport):

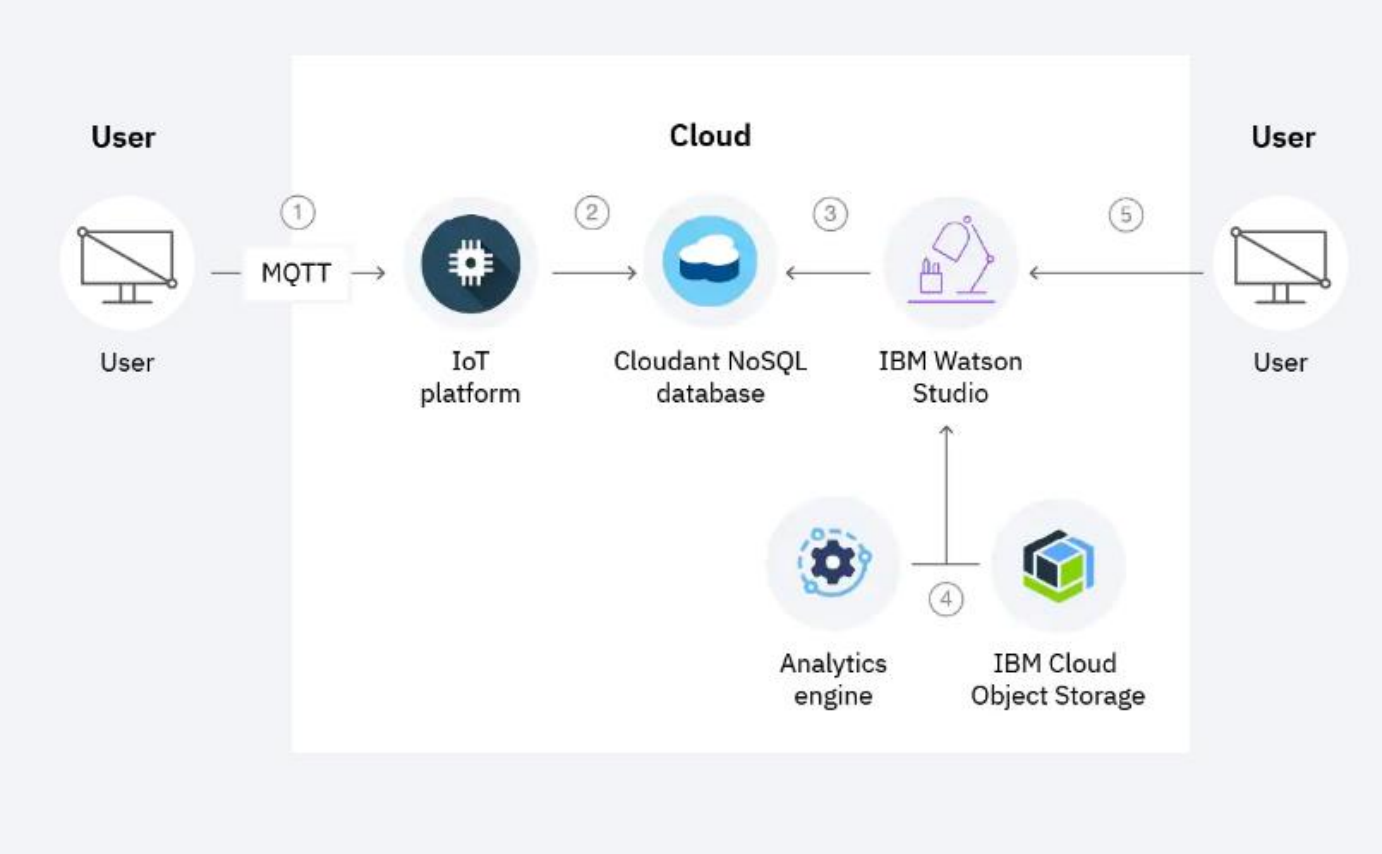
MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. The MQTT broker is the centre of each Publish/Subscribe protocol. It is based on the implementation; a broker can handle up to thousands of simultaneously linked MQTT clients. The broker is answerable for receiving all messages, filtering the messages, deciding who subscribed to every message and sending the message to those subscribed clients. The Broker also manages the sessions of all persistent users, such as subscriptions and missed messages. An asynchronous messaging protocol de-couples the message sender and receiver in both area and time and therefore is extensible in unreliable network environments. MQTT was generated to accumulate information from some tools and then transport that information to the IT framework. It is lightweight, and perfect for isolated monitoring, especially in M2M links needing a small code footprint or definite network bandwidth.

HTTP (Hypertext Transfer Protocol):

HTTP stands for Hypertext Transfer Protocol. It is a protocol used to access the data on the World Wide Web (www). It can be used to transfer the data, also because of its efficiency that allows us to use in a hypertext environment where there are rapid jumps from one document to another document. HTTP is similar to the FTP as it also transfers the files from one host to another host. But HTTP is simpler than FTP as HTTP uses only one connection, i.e., no control connection to transfer the files. HTTP is used to carry the data in the form of MIME-like format. HTTP is similar to SMTP as the data is transferred between client and server. The HTTP differs from the SMTP in the way the messages are sent from the client to the server and from server to the client. SMTP messages are stored and forwarded while HTTP messages are delivered immediately.

Database Schema (Cloudant DB):

IBM Cloudant is a fully managed JSON document database that offers independent serverless scaling of throughput capacity and storage. A fully managed, distributed database optimized for heavy workloads and fast-growing web and mobile apps.



MQTT Protocol – IBM Watson IoT Platform:

User Credentials:

```
#define ORG "wf2kmp"
#define DEVICE_TYPE "GLMASFI_IOT_Device_Cloud_Service"
#define DEVICE_ID "PNT2022TMID35867"
#define TOKEN "PNT2022TMID35867"
const char* ssid = "Airtel-Hotspot-958A";
const char* password = "9889i1bb";
```

Service Authentication:

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[] = "iot-2/evt/status/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
```

Callback Function:

```
void callback(char* topic, byte* payload, unsigned int length) {
 Serial.println("callback invoked");
}
```

Publish data:

```
void PublishData(float senso){
  String payload;
  payload= "{\"Gas_Concentration\":";
  payload += senso;
  payload+="}";
  if (client.publish(topic, (char*) payload.c_str())) {
     Serial.println("Publish ok");
  }
  else {
     Serial.println("Publish failed");
  }
}
```

Host and Client Service:

```
  WiFiClient client2;
  PubSubClient client(server, 1883, callback, client2);
  Serial.print("Connecting to ");
  Serial.print(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("WiFi connected, IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("connecting to ");
  Serial.println(host);
  if (!client2.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }
```

HTTP Service – SMS Alert:

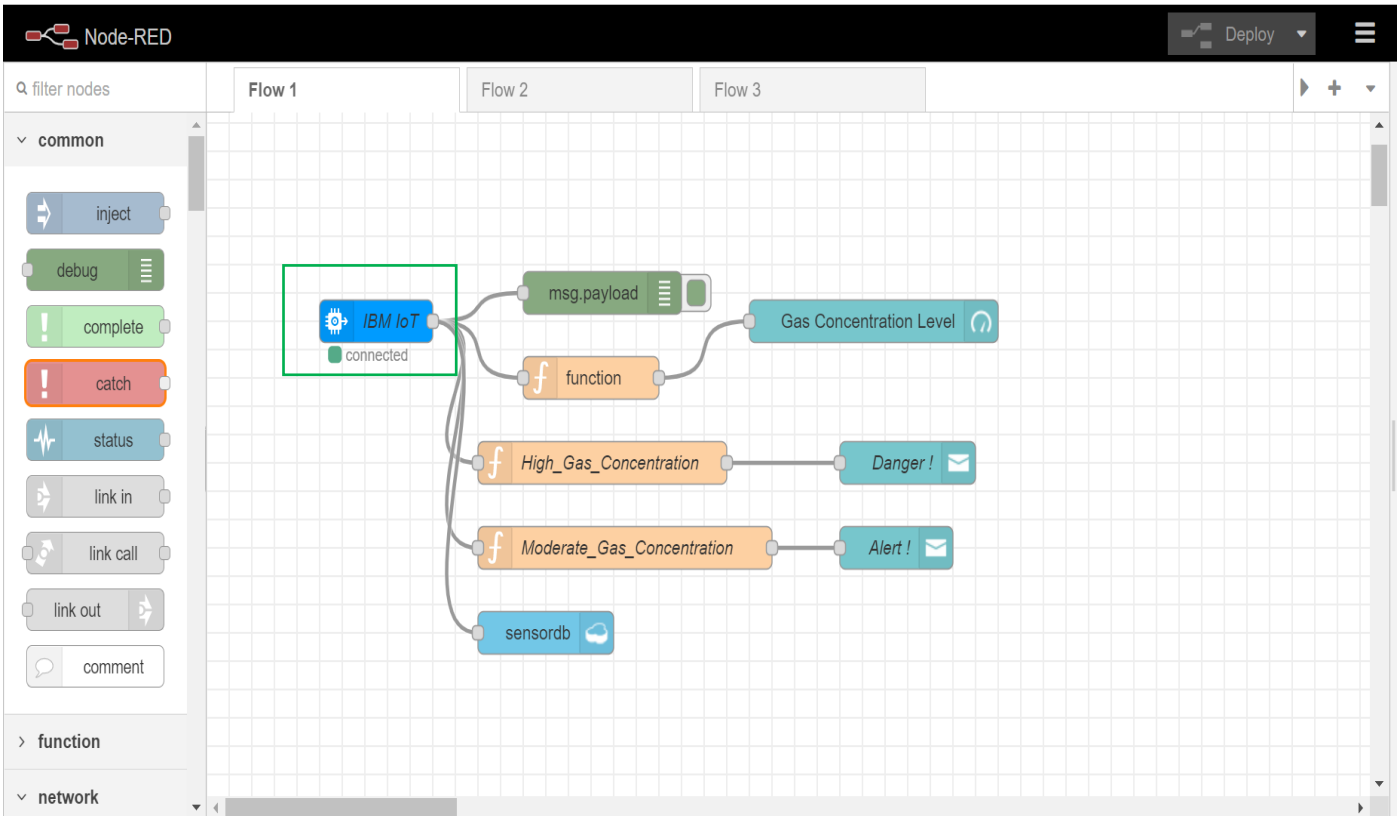
User Credentials:

```
const char* ssid = "Airtel-Hotspot-958A";
const char* password = "9889i1bb";
```

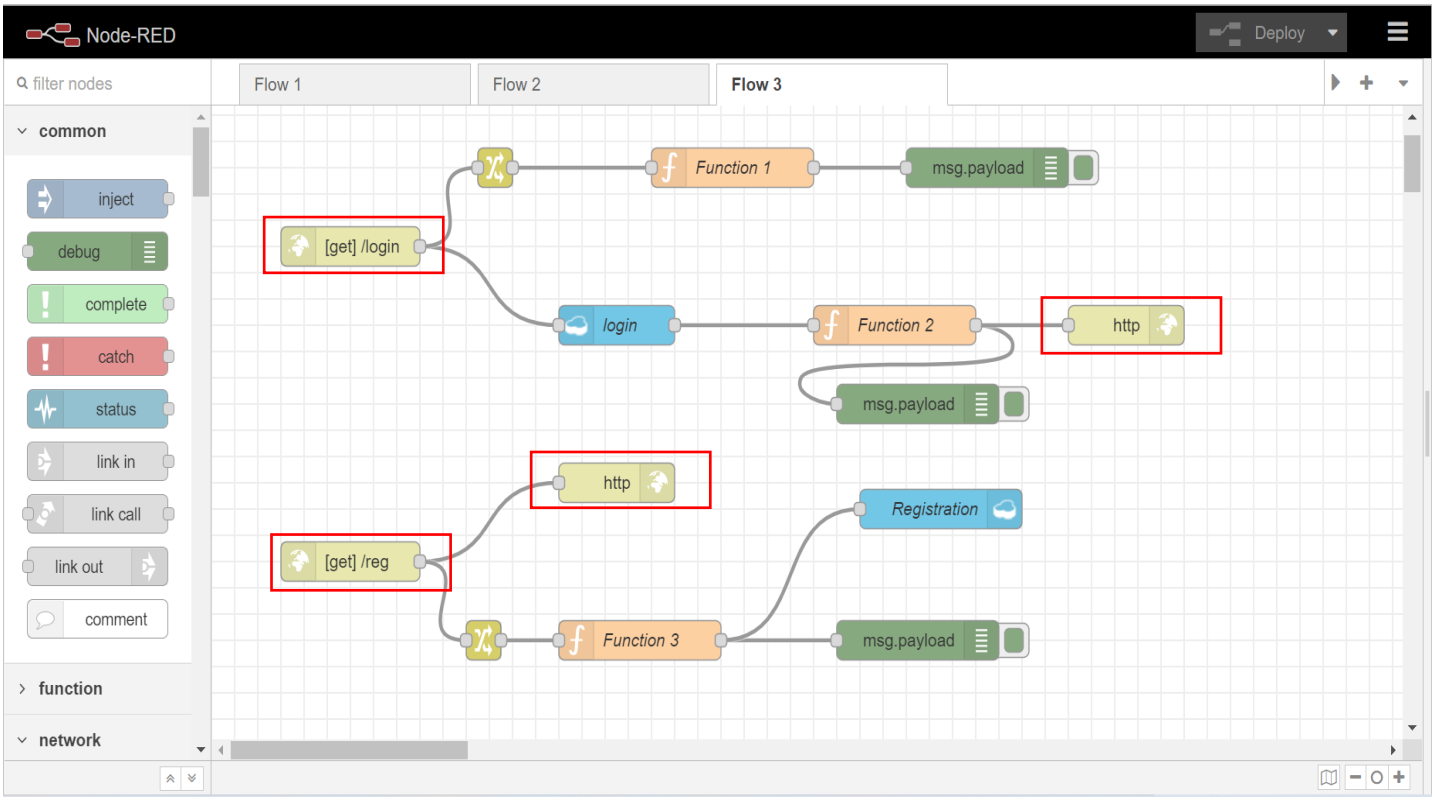
Host & Client Service:

```
const char* host = "maker.ifttt.com";
const int httpsPort = 80;
WiFiClient client1;
  Serial.println();
  Serial.print("Connecting to ");
  Serial.print(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected, IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("connecting to ");
  Serial.println(host);
  if (!client1.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }
  String url = "/trigger/gasle/json/with/key/ktkqqpO7-nkuFo1Dc-jMZX4tNAKchaWS4E6SzY7btPA";
  Serial.print("Requesting URL: ");
  Serial.println(url);
  client1.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
```

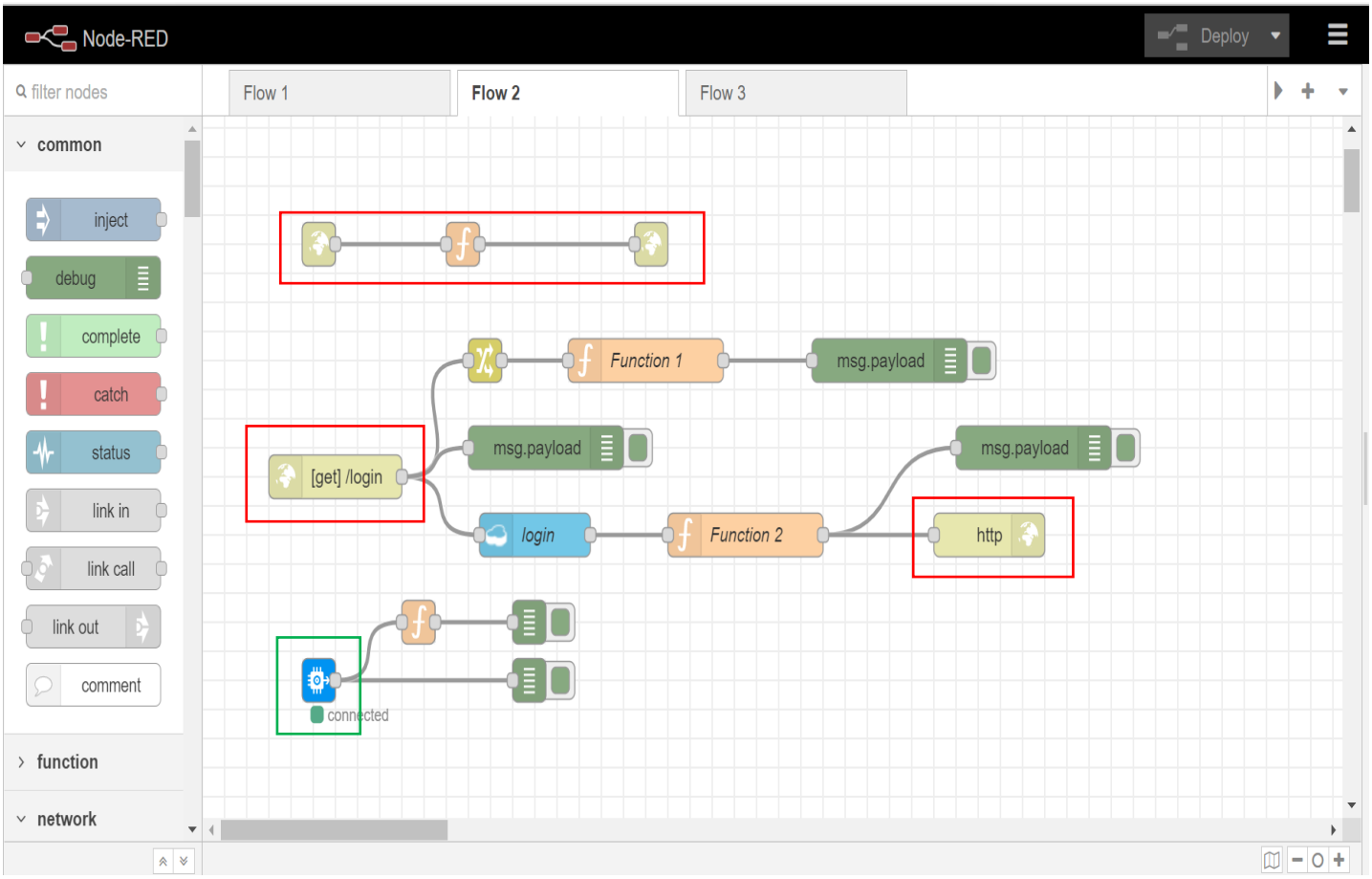
MQTT – Node RED Configuration (Web UI):



MQTT & HTTP – Interface Node RED and App Inventor (App Deployment):
Login & New User Registration:



Interface IBM Watson IoT Platform (MQTT) & App (HTTP) – Gas Sensor data:



METHODOLOGY

- Design of IoT device using Gas Sensor (MQ136), Buzzer and ESP8266 NodeMCU with three concentration level namely Normal (less than 500ppm), Moderate (500-900 ppm) and High (greater than 900 ppm) and buzzer is also configured based on these concentration level namely Normal (no alarm), Moderate (alarm with delay) and High (alarm without delay) using Arduino IDE platform.

- Create IBM Account and open IBM Watson IoT and create new device with unique credentials (including API key). Node RED gets sensor data from ESP8266 (Microcontroller) via IBM Watson IoT platform through MQTT.

- Log into IFTTT account and open applet and create web request to be triggered using webhook to send SMS (using HTTP) to technician (not workers) when high gas leakage occurred, to close gas valve immediately.

- Open Node RED editor and configure Web UI containing gas concentration level gauge with three colours (green – normal, yellow – moderate and red – high) along with notifier displayed on Web UI (for moderate and high-level gas leakage) using sensor made available by IBM Watson IoT platform.

- Develop app interface using Node RED to display sensor data with toxicity level indication using IBM Watson IoT platform (MQTT & HTTP). In addition to that, app have login database with signup pathway to have unique credential and maintain separate database for login authentication.

- Design app using MIT app inventor by adding necessary block and configure IBM Watson IoT platform with app to display sensor data and toxicity level and new user registration & login interface via Node RED (HTTP).

- Create separate database for login credentials and provide authentication to app service using Cloudant DB and separate database for sensor data to be stored for future reference via Node RED from IBM Watson IoT platform (MQTT).

8.TESTING

8.1.TEST CASES

1. **Defect Analysis**

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Issues	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
Defective components	2	0	0	0	2
Improper circuit design	0	0	1	0	1
Include proper libraries and compilation of Arduino IDE	0	3	0	0	3
Poor internet connectivity	0	0	1	1	2
Installation of app	3	0	0	0	3
Sign up and login issues	3	0	0	0	3
SMS Alert delay	0	0	0	1	1
Total	8	3	2	2	15
Bugs Fixed	8	3	2	0	13
Not fixed	0	0	0	2	2

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
IoT Device	3	0	0	3
Cloud Service (IBM Watson IoT platform)	3	0	0	3
Login Credentials Storage (Cloudant DB)	10	1	4	5
Sensor data storage (Cloudant DB)	1	0	0	1
Web UI	5	0	0	5
SMS Alert	1	0	0	1
Mobile App (Login & New User Registration)	4	0	1	3
Mobile App (Sensor data and Toxicity Level)	3	0	0	3

8.2.USER ACCEPTANCE TESTING

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute
ID_1	Functional	IoT Device	1.Verify whether gas is detected by sensor and read by the microcontroller. 2.Verify whether the buzzer rings when the sensor detects toxicity level of gas.	1.Gas sensor 2.ESP8266 NodeMCU microcontroller 3.Buzzer 4.Connecting wires 5.Arduino IDE	1.Interfacing gas sensor and buzzer with the microcontroller. 2.Write the code in Arduino IDE, compile and upload to the microcontroller. 3.When the sensor detects a gas, verify the concentration of the gas detected in the serial monitor of Arduino IDE. 4.Check whether buzzer rings, when toxicity level of gas detected.
ID_2	Functional	IBM Watson IoT Cloud Platform	Configure the IBM Watson cloud platform with the IoT device and verify whether the data received from the device is updated in the cloud.	IBM Watson IoT Platform	1.Send the gas concentration data from the IoT device to the IBM Watson cloud. 2.Verify the received data by logging in to the IBM Watson Platform.
ID_3	User Interface	Web UI Dashboard	Verfiy whether gas concentration value is displayed in the Node RED Dashboard and Toxicity level is indicated in Web UI	Node RED	1. Create Node RED account using User Credentials. 2. Open Node RED Editor and utilise blocks to interface IBM Watson, Cloudant DB and provide function for user-friendly experience. 3. Copy Node RED URL into search bar with 'ui' and opens Web UI created by Node RED and display gas concentration values and Toxicity level is indicated in UI
ID_4	User Interface	Mobile App	Verify whether user can use the app to view the concentration of the gas and toxicity level & provide unique login credentials	MIT App Inventor	1. Create MIT app Inventor account using google credentials, Open new project and Add necessary front end block in designer Interface. 2. Next, for backend, add necessary URL along with Working/ Action blocks to perform front end work with proper configuration. 3. Perform rehersal of App working by using MIT AI2 Companion and verify App working and download Apk file and ready for utilisation.
ID_5	Web service	SMS Alert	Verify whether the Alert SMS is sent to the user when a high concentration of gas is detected	IFTTT	1.Create an applet on IFTTT for receiving a web request to the user with the alert meassage and generate the url to be triggered. 2.Configure the Arduino IDE to trigger the url when a high concentration of gas is detected. 3.Verify whether the alert sms is sent to the user .
ID_6	Storage	Cloud storage database	1.Check whether the gas concentration value from the device is stored and updated in the Cloud storage 2.Check whether the login credentials from the app are also stored for authentication	IBM Cloudant DB	1. Create Cloudant DB Account using User Credentials. 2.Create Separate database for Login Credentials and Sensor data. 3. Interface IBM Watson IoT Platform and Cloundant DB to store data using Node RED. 4. Verify whether Login credentials and sensor data is stored properly

Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
1. Low Level (Normal) 2. Moderate Level - Escape Stage 3. High Level - Danger (Immediate death)	Gas concentration value must be shown in the serial monitor of IDE and the alarm rings when toxicity level of gas detected.	Working as expected	Pass	Nill	Yes	NA	A Joey Infant Rex E Prakash
Poor Internet Connectivity	The data sent from IoT device must be received and displayed in the IBM Watson platform	Working as expected	Pass	Nill	Yes	NA	A Joey Infant Rex E Prakash
1. Three Color - Three Level 2. Display "Alert! Gas Leakage Occurred" - Moderate toxicity Level 3. Display "Danger! High Gas concentration" - High Toxicity level	Data is published instantly on Web UI and Toxicity level is indicated by Notification on Web UI	Working as expected	Pass	Nill	Yes	NA	A Joey Infant Rex E Prakash
1. New User Registration with proper Credentials 2. New User Registration with Improper details (without mobile Number) 3. Login with Correct Credentials (Login Sucessful) 4. Login with Improper Credentials (Login Failed) 5. Display Sensor data and Toxicity level of Gas.	The app must be installed properly and Gas Concentration and Toxicity level is published instantly	Working as expected	Pass	Nill	Yes	NA	A Joey Infant Rex E Prakash
Poor Internet Connectivity	The Alert sms must be sent to the user when a high concentration of gas is detected.	Working as expected	Pass	Nill	Yes	NA	A Joey Infant Rex E Prakash
1. Improper Cloud configuration 2. Poor Internet connectivity	1.The gas concentration value from the device must be stored in the cloudant database. 2.The login credentials from the app also should be stored	Working as expected	Pass	Nill	Yes	NA	A Joey Infant Rex E Prakash

9.RESULTS

9.1.PERFORMANCE METRICS

NFT - Risk Assessment:

Project Name	Gas Leakage Monitoring & Alerting System for Industries
Scope/Feature	New
Functional Changes	Moderate
Hardware Changes	Moderate
Software Changes	High
Impact of Downtime	Lost Revenue: A downtime incident often slows down the customer facing applications & Systems of a company, resulting in significant Revenue Loss
Load/Volume Change	>10 to 30%
Risk	Green

NFT – Detailed Test Plan:

Project Overview	Gas Leakage Monitoring and Alerting System for Industries provides a solution of alerting the industrial workers about the leakage of gases & helps them to close the gas valve (or) evacuate the prone zone based on the toxicity. If any high concentration of gas is detected, alarm is activated and SMS alert is sent to the technician, also using Mobile App, concentration of leakage gas and toxicity level can be viewed.
NFT Test Approach	Performance Testing/Compatibility Testing/Usability Testing/Stress Testing/Scalability Testing/Portability Testing/Efficiency Testing/Reliability Testing
Assumption/ Dependencies/ Risks	1. No Assumptions 2. High Dependency on IBM IoT Platform (Node RED, Cloudant DB, Watson IOT) 3. Less Risk
Justification	The framework is robust and capable of handling the data throughput from the devices

End of Report:

NFR Met	Performance/Scalability/Efficiency/Usability/Compatibility
Test Outcome	Success
Go/No Go Decision	Go
Recommendation	1. New User Registration should be Enhanced 2. Better internet connectivity for occasional Latency 3. Should Pin point Location of Gas Leakage
Identified Defects	None (Closed)

10.ADVANTAGES & DISADVANTAGES

Advantages:

- Once gas leakage occurred, it displayed by Web UI and App and alerted by Alarm System; So, Health problems and death can be avoided.
- Financial losses can be prevented, in case of heavy Gas Leakage, SMS Alert is Triggered to Intended Technician to close Gas Valve Immediately.
- Prevent fire accidents and explosions owing to gas leakage, because, it gets avoided by alarm system & Web and App UI.
- Cost-Efficient and better reliable resource being utilised.
- Provides continuous monitoring for Occurrence of gas leakage and data along login credentials (App) is stored in database for future use.
- Flexibility and Reliability, Since IoT device is extensible to any type of Gas, because only gas sensor need to replace for appropriate gas (e.g., MQ136 Sensor – H_2S gas)

Disadvantages:

- In case of poor internet connectivity, real time gas concentration data may not be sent to cloud service (Low Latency) and device performance gets degraded.
- Unauthorised Person can install gas leakage app easily, sign up and login with app using improper credentials (e.g., Without proper Username & proper Mobile Number).
- Since continuous monitoring of gas concentration is made, the device may not be functional every time and thus requires proper maintenance; if not device malfunction might occur leading to false gas leakage detection and trigger SMS and Alarm System.

11.CONCLUSION

This project "Gas Leakage and Monitoring System for Industries" is an effective solution to alert the industrial workers & industrialist, in case of any toxic gas leakage, thus preventing health issues, fire accidents and financial losses by triggering SMS alert to Technician to close gas valve (in case of high gas leakage) and Alarm System to Industrial workers working in prone zone, in addition to that, display gas concentration level in web UI simultaneously with notifier (moderate and high gas leakage); It also utilises app to display gas concentration value and level with login credentials. Cloudant DB is utilised to store both login credentials database and sensor database and used for future reference and app login authentication.

12.FUTURE SCOPE

App contains gas concentration value and gas toxicity level, in addition to that, it should have “SMART” button. SMART Button gets triggered and close gas valve immediately and automatically, when high gas leakage detected. It is done by Technician/Supervisor, since app have “Admin Login” button in login page, gets directed to Admin Login screen with unique login credentials and directed to another screen containing that “SMART Button”. In addition to that, user dashboard should contain location of gas leakage occurrence, so that future accident can be prevented. Proper formatting of new user registration screen which evaluates user credentials for proper username (no existing name and mobile number), proper number format and strong password (should have at least one special character, capital letter and number and minimum length of password).

13.APPENDIX

SOURCE CODE

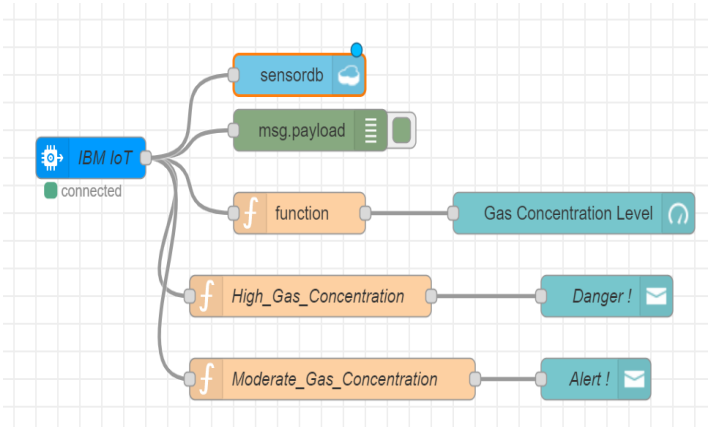
Arduino Code:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define Buzzer D5
#define Green D6
#define Sensor A0
#define ORG "wf2kmp"
#define DEVICE_TYPE "GLMASFI_IOT_Device_Cloud_Service"
#define DEVICE_ID "PNT2022TMID35867"
#define TOKEN "PNT2022TMID35867"

const char* ssid = "Airtel-Hotspot-958A";
const char* password = "9889i1bb";
const char* host = "maker.ifttt.com";
const int httpsPort = 80;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[] = "iot-2/evt/status/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

void PublishData(float);
void callback(char*, byte*, unsigned int);

WiFiClient client1;
WiFiClient client2;
PubSubClient client(server, 1883, callback, client2);

void setup() {
  pinMode(Green, OUTPUT);
  pinMode(Buzzer, OUTPUT);
  pinMode(Sensor, INPUT);
  Serial.begin(115200);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.print(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected, IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("connecting to ");
  Serial.println(host);
  if (!client1.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }
```

```
  if (!client2.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }
}

int msgSent = 0;
void loop() {
  Serial.print("Reconnecting client to ");
  Serial.println(server);
  while (!client.connect(clientId, authMethod, token)) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();

  int sensor = analogRead(Sensor);
  Serial.println(String(sensor));
  PublishData(sensor);
  if (sensor >= 750 && !msgSent) {
    digitalWrite(Green, HIGH);
    digitalWrite(Buzzer, HIGH);
    String url = "/trigger/gasle/json/with/key/ktkqqpO7-nkuFo1Dc-jMZX4tNAKchaWS4E6SzY7btPA";
    Serial.print("Requesting URL: ");
    Serial.println(url);
    client1.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection:
close\r\n\r\n");
    msgSent = 1;
    Serial.println("Gas Concentration Level: High");
  }
  else if (sensor >= 750 && !msgSent) {
    digitalWrite(Green, HIGH);
    digitalWrite(Buzzer, HIGH);
    Serial.println("Gas Concentration Level: High");
  }
  else if (sensor>=620){
    digitalWrite(Buzzer, HIGH);
    digitalWrite(Green, HIGH);
    delay(750);
    digitalWrite(Buzzer,LOW);
    digitalWrite(Green, LOW);
    delay(1000);
    Serial.println("Gas Concentration Level: Moderate");
  }
  else{
    digitalWrite(Green, LOW);
    digitalWrite(Buzzer, LOW);
    Serial.println("Gas Concentration Level: Normal");
  }
  delay(1000);
}

void callback(char* topic, byte* payload, unsigned int length) {
```

```
 Serial.println("callback invoked");
}

void PublishData(float senso){
  String payload;
  payload= "{\"Gas_Concentration\":";
  payload += senso;
  payload+="}";
  Serial.print("Sending payload: ");
  Serial.println(payload);
  if (client.publish(topic, (char*) payload.c_str())) {
    Serial.println("Publish ok");
  }
  else {
    Serial.println("Publish failed");
  }
}
```

Interfacing Node RED & Web UI Code:



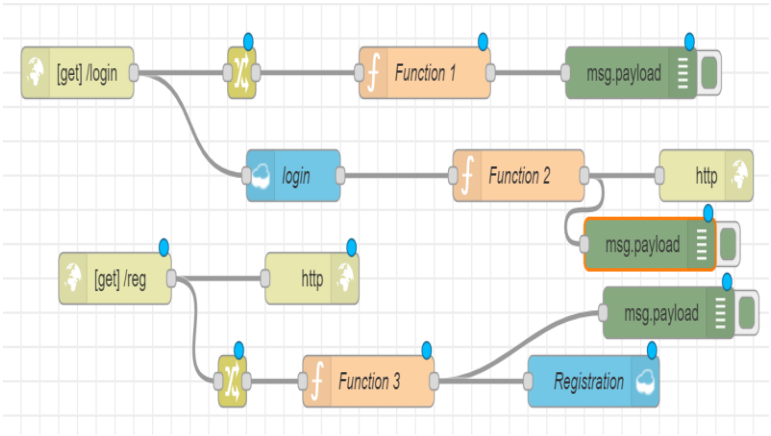
Call function from IBM Watson to Node RED Gauge:

```
var msgt={};
msgt.payload=msg.payload.Gas_Concentration
return msgt;
```

Call function from IBM Watson to Node RED Notifier:

```
//High
msg.payload=msg.payload.Gas_Concentration;
if (msg.payload>=890){
return msg;
}
//Moderate
msg.payload=msg.payload.Gas_Concentration;
if (msg.payload<890 && msg.payload>=720){
return msg;
}
```

Interfacing Node RED & App Code:



New User Registration Access from json format:

```
//New registration
a=msg.payload.reg
s=a.split(",")
name = s[0]
pass = s[1].trim()
mno = s[2].trim()
msg.payload={
   "User Name": name,
   "Password": pass,
   "Mobile Number": mno,
}
return msg;
```

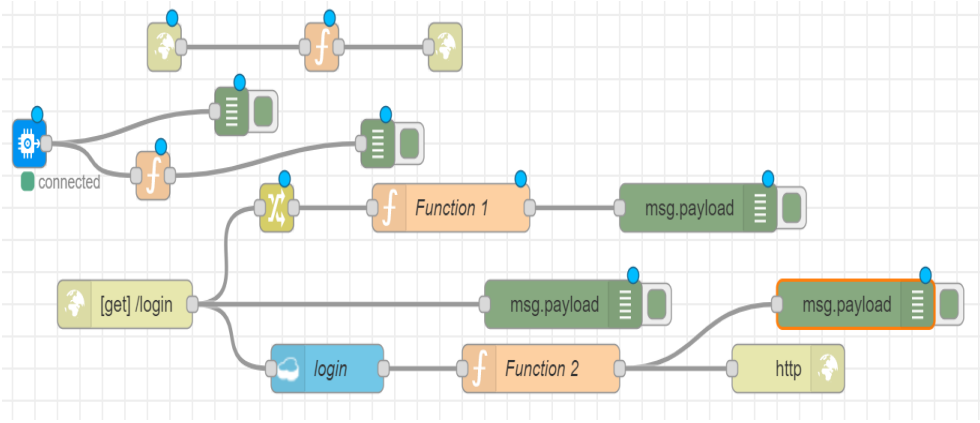
Login Credential Access from json format:

```
//login
a=msg.payload.login
s = a.split(",")
user = s[0]
pass = s[1].trim()
msg.payload={
   "User Name":user,
   "Password":pass   // convert to json format, store in msg.payload
}
global.set('u',user) //the above things are used for checking in DB
global.set('p',pass)
global.set('m',msg.payload)
return msg;
```

Login Authentication from login credential access:

```
//Verification
var a=msg.payload.Gas_Concentration
var e=global.get('u')
var p=global.get('p')
count=0
for (var j in a){
   count=count+1
}
flag=0
for (i=0;i<count;i++){
   if((e==a[i]["User Name"])&(p==a[i]["Password"]))
```

```
    {
      flag = 1       //if flag =1, credential available in DB, send to App.
    }
}
if (flag==1){
    msg.payload = 1
}
if(flag==0){
    msg.payload=0
}
return msg;
```



Display Gas sensor data and toxicity level from IBM Watson:

```
//format of display
a = msg.payload.Gas_Concentration
y = global.get("flag")
if (y == 1){
    if (a > 980){
        toxic="High"
    }
    else if (a < 980 & a>720){
        toxic="Moderate"
    }
    else{
        toxic="Low"
    }
    msg.payload = {
        "level":a,
        "alert":toxic
    }
    global.set("l", a)
    global.set("v", toxic)
    global.set("msg", msg)
    return msg;
}
```

GITHUB & DEMO VIDEO LINK

GITHUB Link:
IBM-EPBL/IBM-Project-453-1658301890: Gas Leakage monitoring & Alerting system for Industries (github.com)

Demo Video Link:
https://drive.google.com/file/d/1GXH-PN1TBr_g6MVgh2u72uN76Vx7rPCZ/view?usp=share_link