IBM-NALAIYTHIRAN

GAS LEAKAGE MONITORING AND ALERTING SYSTEM

PROJECT REPORT

TEAM ID: PNT2022TMID05161

TEAM MEMBERS:

- BHARATH SURYA R
- ASWIN S
- DEEPAK CHANDRU S
- BALAMANIKANDAN A

Project overview:

A wide range of gas is used in Industries for various purposes such as, Heating, Preserving, etc. These gases can also be hazardous in nature. So, monitoring the flow of gas is very important and Leakage is simply unavoidable. As we have already witnessed a couple of calamities it is time to learn from them. This expert system is capable of monitoring the environmental parameters at all times. In case of emergency, using a GSM module it notifies the concerned person to take necessary step to prevent an accident.

Objectives:

- This project helps the industries in monitoring the emission of harmful gases
- In several areas, the gas sensors will be integrated to monitor the gas leakage
- If in any area gas leakage is detected the admins will be notified along with the location
- In the web application, admins can view the sensor parameters

Problem Formulation:

Any gaseous molecule that escapes from a stove, pipeline, cylinder, etc. is considered a gas leak. This may happen on purpose or even accidentally. Given that we are aware that these leaks are harmful to our health, and when it explodes, it poses a serious threat to everyone's safety as well as that of their homes, places of employment, industries, and the environment. we are in need of taking necessary precautions to prevent this calamity.

The Bhopal Disaster and the Vizag Gas Leak are only a couple of the significant disasters that occurred as a result of gas leaks. As far as industrial accidents go, the Bhopal disaster is considered the worst. From this insecticide plant, over 45 tons of methyl isocyanate escaped. Methyl isocyanate is an organic molecule that can be found in insecticides that contain carbamates. The liquid is colorless, lethal, and flammable, and people should stay away from it.

In Vizag the release of styrene from containers that were left unattended for a long time caused a gas leak. This viscous liquid has no color and can spread through fumes. Therefore, a detector needs to be built in a way that it can pick up any type of gas, fume, leak, smoke, etc. The detector may be fitted with particular settings that could help to prevent the problem, however severe and deadly it may be.

PROBLEM STATEMENT

Workers who are engaged with a busy industries packed with gas either harmful or harmless needs a way to monitor their gas pipelines continuously and detect early if there is any leakage of gas in their surroundings so that they can work efficiently on major crises rather than worrying about monitoring or leakage of gas, this will indeed reduce the manpower of that industry and create a peaceful environment.

SOLUTION STATEMENT:

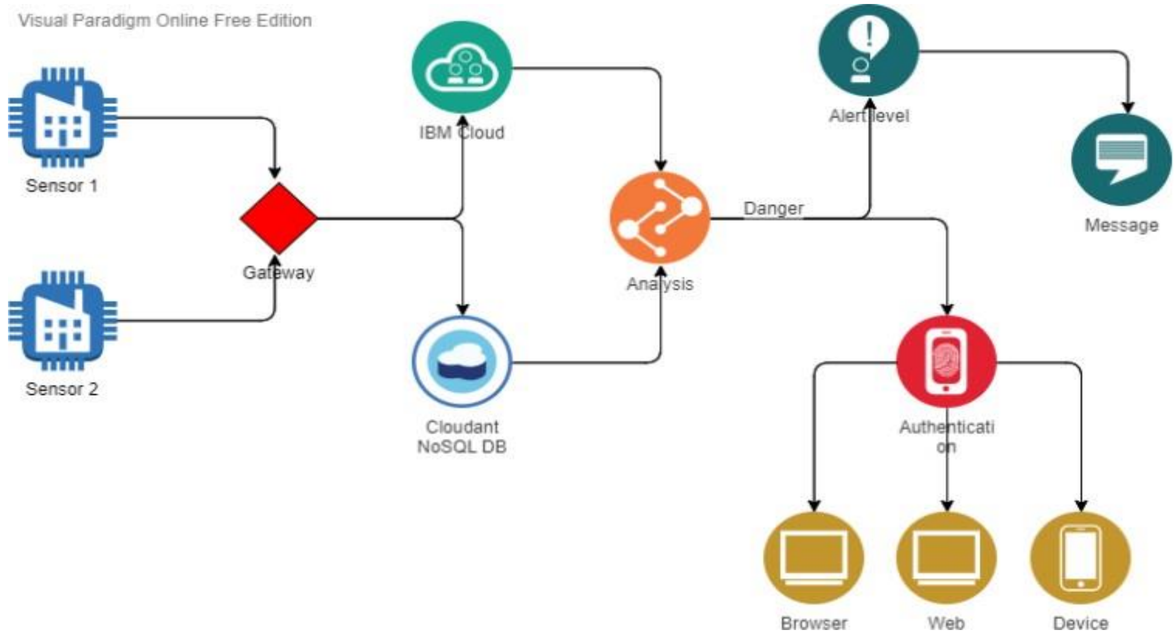
The system can be taken as a small attempt in connecting the existing primary gas detection methods to a mobile platform integrated with IoT platforms. The gases are sensed in an area of 1m radius of the rover and the sensor output datas are continuously transferred to the local server.

The accuracy of MQ sensors are not upto the mark thus stray gases are also detected which creates an amount of error in the outputs of the sensors, especially in case of methane.

Further the availability and storage of toxic gases like hydrogen sulphide also creates problems for testing the assembled hardware. As the system operates outside the pipeline, the complication of system maintenance and material selection of the system in case of corrosive gases is reduced.

Thus the system at this stage can only be used as a primary indicator of leakage inside a plant.

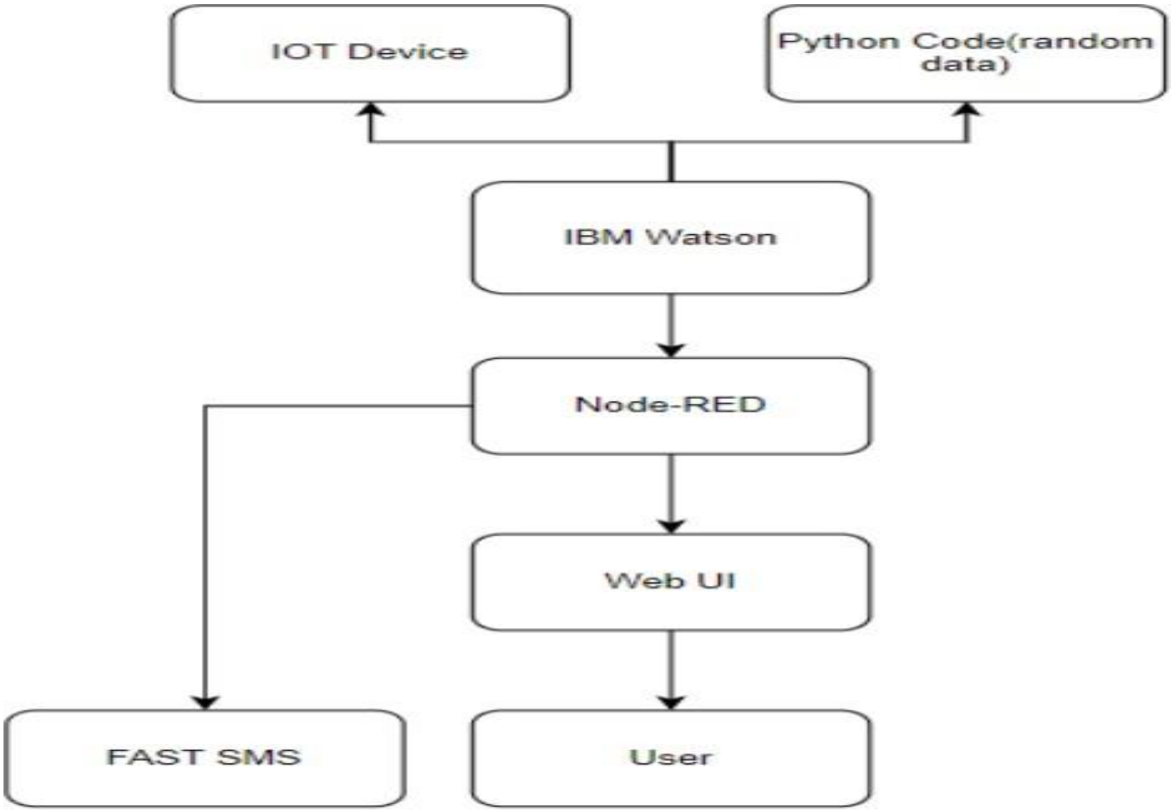
Solution Architecture:



1. Data is gathered from sensor.
2. Data is uploaded to Cloud.
3. Data is displayed in Web.
4. After authentication, Data is displayed in Mobile.
5. In case of emergency Mobile application notifies the user.

Integrity and data confidentiality are the 2 key security features to be taken care of. Tampering the data may lead to serious accidents. User can easily keep an eye on their industry though they stay at home. This system is just like a surveillance system. Instead of theft, here we are surveilling the Gas to prevent accidents.

Proposed Solution:



1. IoT Device:

Our entire work depends on this module. Data gathered by the sensor must be accurate and should be available without any latency. Here we use gas sensor to detect the presence of various gaseous substances like Carbon Mono Oxide, Carbon Di Oxide, Methane, etc. According to the need sensor can be changed to detect other kinds of gas. As each industry deals with different kinds of gases.

2. IBM Watson:

This is where we connect our independent IoT device with internet and to gather the sensed environmental parameters. This here is responsible for collecting the data, so each IoT device is connected with this platform using various meta details of the controller.

3. Node-RED:

Node-RED gathers the data from IBM Watson using nodes. This module acts as a backbone to publish data. Node- RED plays a vital role in creating a dashboard where the user/admin can view the sensed data. At the same time the data is also published in a temporary website, from which our mobile application gathers the data.

4. MIT App:

This is the mobile application where the user must login with their credentials to monitor the environmental parameters. This here is made up of several screen for various purposes such as to log in, to raise alarm, to display the data, etc. An industry might have multiple security personals who might be in need of access to monitor the data, so each individual person has their own unique ID and secret password

5.Alarm

When the percentage of gas in the environment is above 45% an alarm is raised in the user mobile with the help of the application. This alerts the concerned person to make the necessary decisions to avoid a calamity and to save those souls working in the industry.

PROPOSED SOLUTION :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Gas leakage leads to various accidents resulting in loss of human lives and industry properties. Sometimes, the gas leakage cannot be detected by human that has a low sense of smell. Thus, this system will help to detect the presence of gas leakage and alert the users.
2.	Idea / Solution description	It detects the gas leakage by using various sensors. If the gas leakage level is above the threshold level, it sends the alert message through SMS to the user by using GSM module and buzzer the alarm.
3.	Novelty / Uniqueness	We use location tagging and alert service so that the admin and fire department team will be notified the exact location. The system provides constant monitoring and detection of gas leakage along with storage of data in database for predictions and analysis.
4.	Social Impact / Customer Satisfaction	By implementing real-time gas leak detection, industries can monitor their environmental performance, ensure better occupational health, and eliminate potential hazards. Also, early detection of gas leaks can trigger concerned engineers to curtail the spread and keep a safe environment for better health and safety.
5.	Business Model (Revenue Model)	The product can be made compact, cost efficient and easily installable so that all the industries from small scale to large scale can able to buy the product .
6.	Scalability of the Solution	The system is very simple and easy to maintain and cost efficient. It has the capability to works for a period of time without any damage in the system components.

LITERATURE SURVEY:

PROJECT TITLE	AUTHOR	OUTCOME
1. Gas leakage Detection and control sytem	1) Sanjay Kumar 2) Durgesh Kumar	We have also incorporated a feature that will increase the safety even further. We have added the feature to cut off electricity to the house as even a small leakage can turn into a deadlu disaster from a little spark. The sensor which we have used can detect LPG, isobutane, propane , LNG ,and smoto too.
2. Gas Leakage with Auto Ventilation and Smart Management System Using IoT	1) Afsana Mim Anika 2) Ms. Nasrin Akter	It can speak with regulators by utilization of AT orders. A SIM can be embedded into it permitting the highlights, for examp le, making phone calls and sending SMS . In the is framework the module has been used to send SMS.

3. Gas Leakage Detection Based on IOT	1) V Suma 2) Ramya R Shekar 3) Kumar Akshay	The aim of this paper is to present a new system automatically books a cylinder when the gas is about to empty is by sending a notification to the gas agency using wifi using Internet of Things approach.
4. Bengaluru Gas Leakage Detection Based System(ICEA2017)	1) Adil Ahmad 2) Shaik Shaheeda	The author has observed gas leakage and LPG levels where gas leakage occurs automatically. The authors suggests that gas leakage is performed by various gas sensors. Whose author has worked on gas leaks and mentions that we can take care if a found using a sensor and gas booking can be done automatically when a small amount of gas is taken closed.

5. GSM based LPG leakage detection and controlling system(2015)	1) Prof.M.Amsaveni 2) A.Anurupa 3) R.S.AnuPreetha 4) C.Malarvizhi 5) M.Gunasekaran	They proposed their methodology that the system takes an automatic control action after the detection of 0.001% of Gas leakage. This automatic control action provides a mechanical handle for closing the valve. We are increasing the security for human by means of a relay which will shut down the electric power to the house. Also by using GSM, we are sending an alert message to the users and a buzzer is provided for alerting the neighbors about the leakage.
6. Intelligent LPG gas leak detection and automatic gas booking alert system using pic microcontroller	1) Bhavithra S 2) Sushmitha B	LPG gas is primarily used for cooking in our country. This paper focuses on continuous monitoring, booking and

		leakage detection in household LPG cylinder.
7. IoT based level detection of gas for booking management using integrated sensor	1) Mariselvam Venkatakrishnaraj 2) M. Siva Dharshini	Until now gas level is measured using weight machine. But here, the level is monitored using i- sensor (integrated sensor) which is developed using transducer. It is connected to NODE MCU for reading the output then it is monitored using IoT.

Empathy Map

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

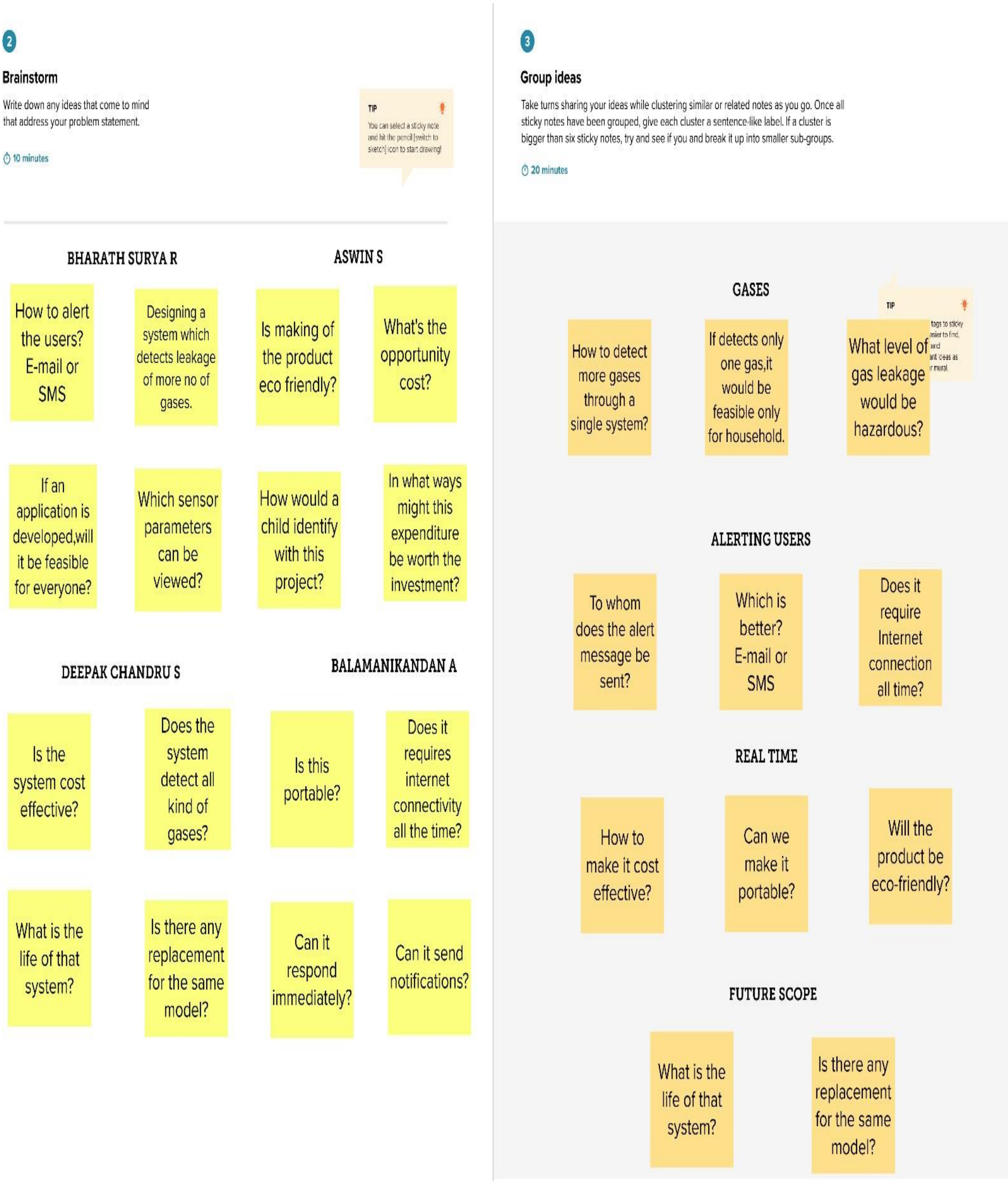
PROBLEM
How might we [your problem statement]?

Key rules of brainstorming

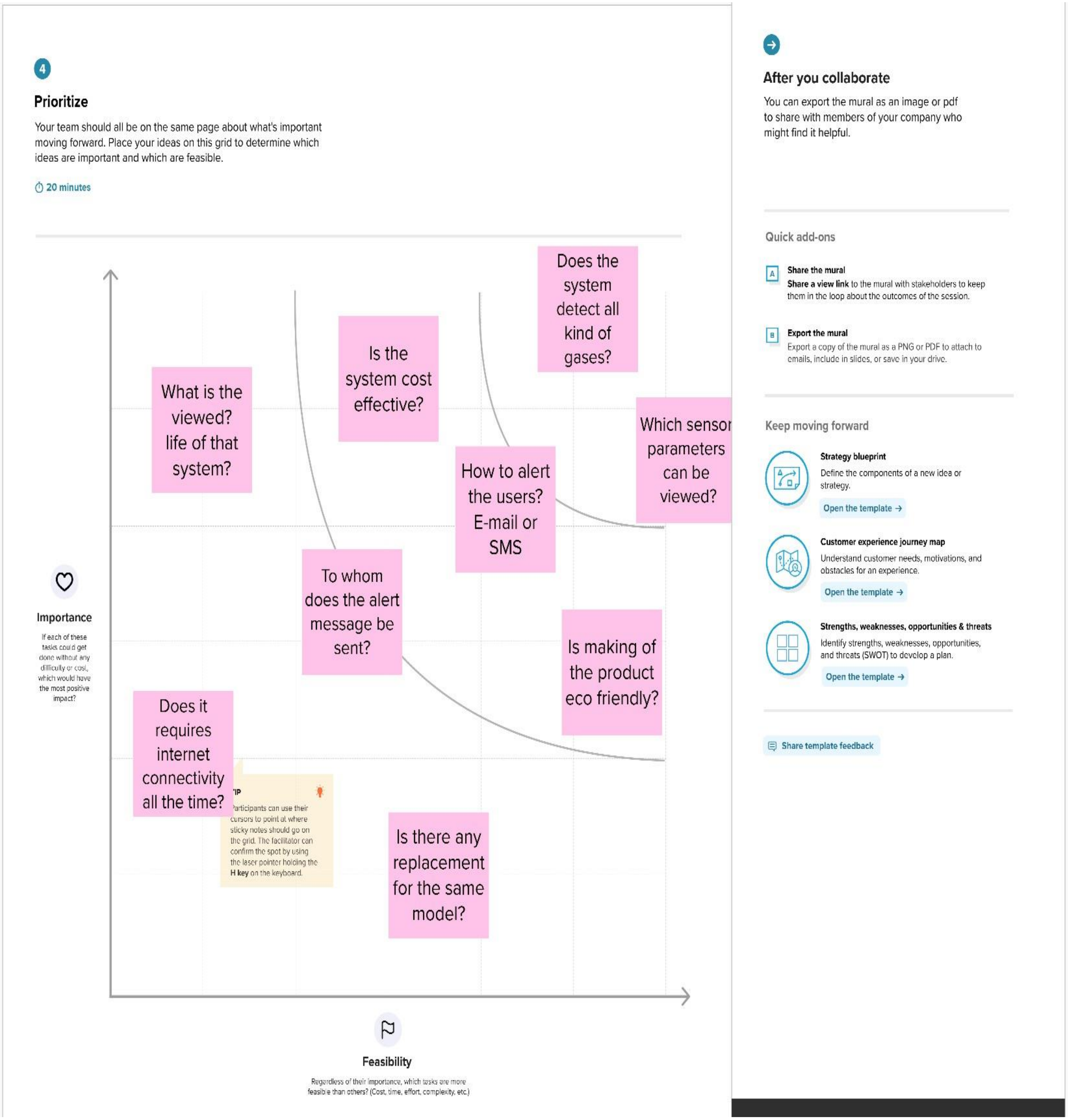
To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

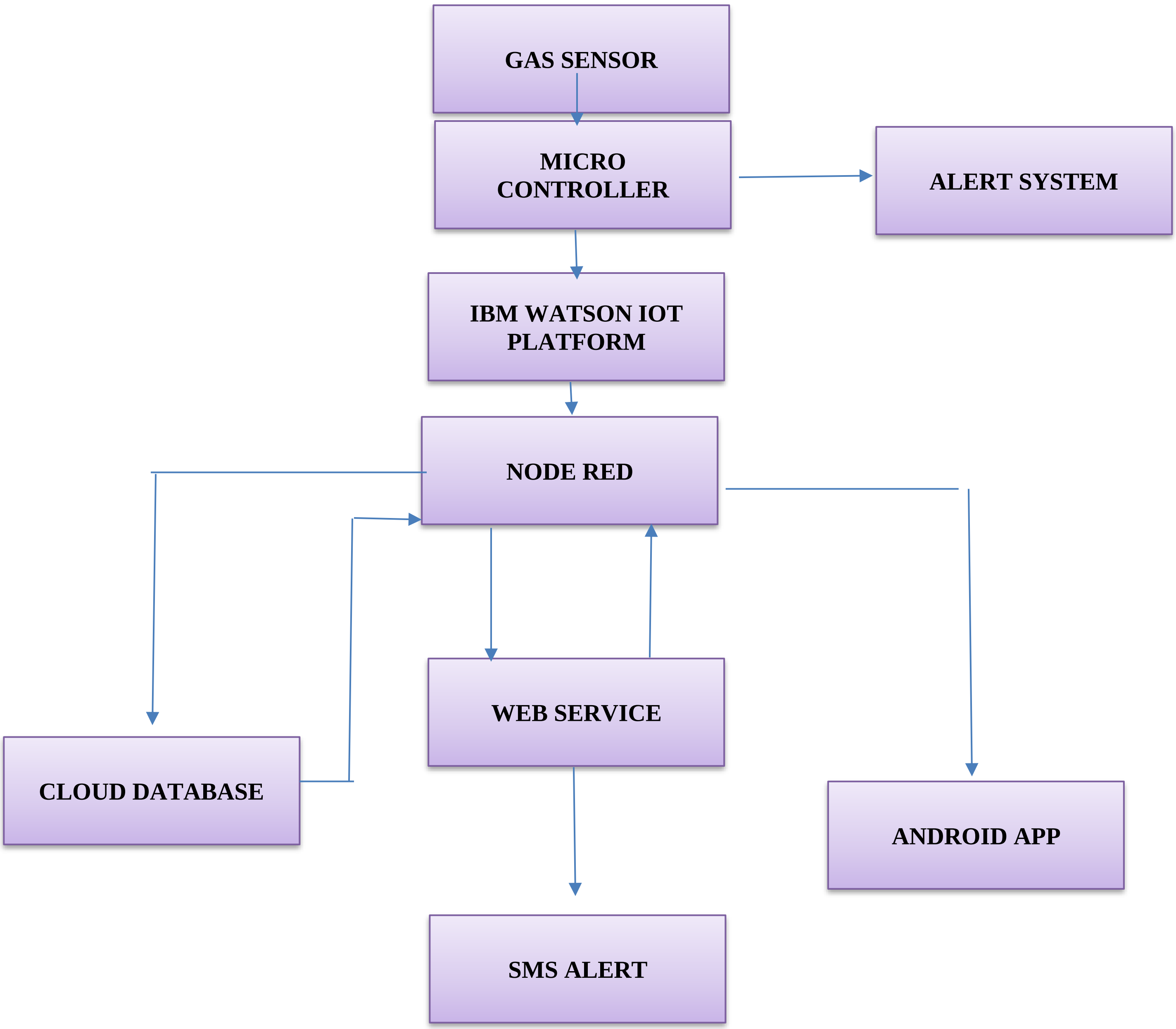
Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



DATA FLOW CHART:

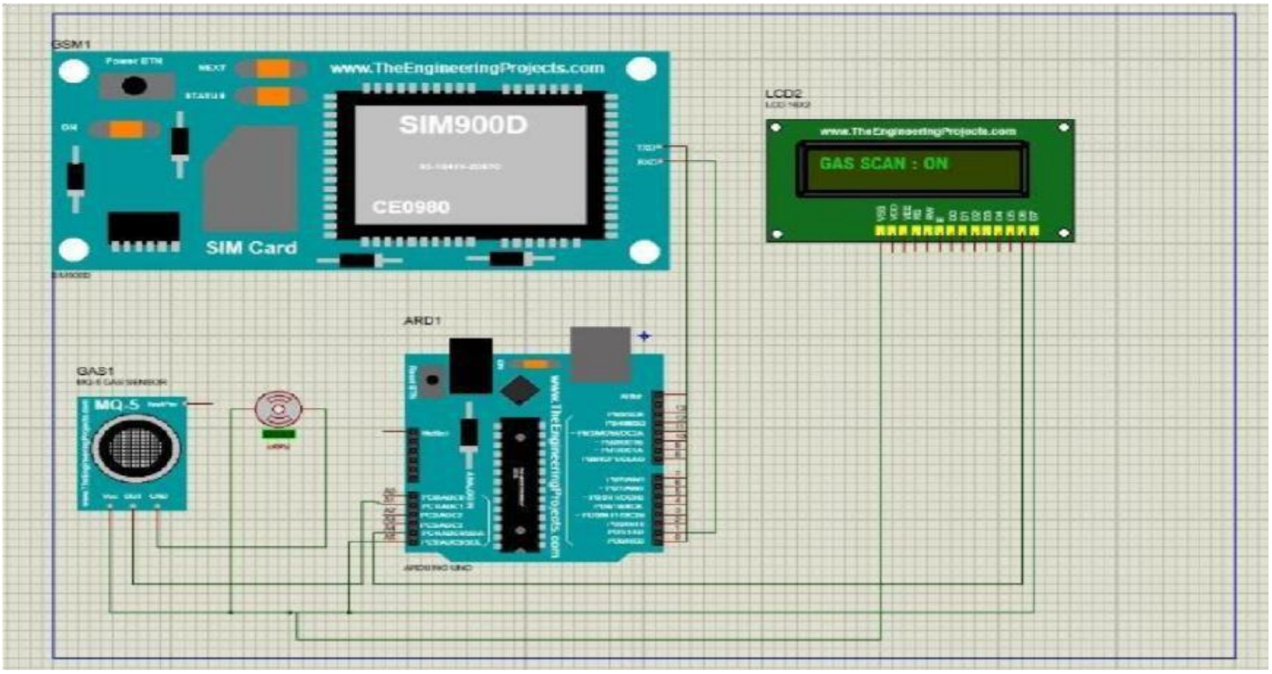


USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	priority	Release
Customer (Industry owner)	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	High	Sprint-1

Customer (Industry owner)	Confirmation	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
Customer (Industry owner)	Authorize	USN-3	As a user, I will enable the supervisor to monitor the gas leakage system status.	I can provide access to supervisor	High	Sprint-1
Customer (supervisor)	Login	USN-4	As a user, I can log into the application by entering email & password.	I can get access to dashboard.	High	Sprint-1
Customer (supervisor)	Monitor	USN-5	As a user, I can monitor the status of the gas leakage system	I can view the status of gas leakage system.	High	Sprint-1
Customer (line workers)	Notification	USN-6	As a user, I can get (alarm system) alert about gas leakage.	I can get alert about gas leak.	Medium	Sprint-2
Customer (supervisors)	Notification	USN-7	As a user, I can get SMS notification & alarming alert about gas leakage.	can get alert about gas leakage.	Medium	Sprint-2
Customer (Industry owner)	Notification	USN-8	As a user, I can get SMS notification about gas leakage.	I can get alert about gas leakage	Medium	Sprint-2
Customer (supervisor)	Sign-Up	USN-9	As a user, I can sign-up using Facebook login.	I can sign-up with the application using Facebook	low	Sprint-3
Customer (Industry owner)	Sign-Up	USN-10	As a user, I can sign-up using Facebook login.	I can sign up with the application using Facebook	low	Sprint-3
Administrator	Service Request	USN-11	As a user, I can request for service in case of any issue with gas leakage monitoring system	I can get service from provider	low	Sprint-3
Administrator	Increased Service	USN-12	As a user, I can request for scaling up the gas leakage monitoring system.	I can get service from the provider.	low	Sprint-4

TECHNICAL ARCHITECTURE:



Technical :

- MQ5 gas sensor
- Arduino uno board
- GSM 800A module
- DC fan
- LCD display (gas scan and alert)

Functional:

- First detects the gas leak.
- Signal goes to Arduino
- DC fan tuns ON
- Alert SMS sent to user's mobile number
- Source valve turned OFF.

HARDWARE REQUIREMENTS:

1) Arduino UNO Microcontroller
2) Green LED 1, Red LED 3
3) 9V Power supply (230V TO 9V STEPDOWN TRANSFORMER)
4) MQ6 Gas sensor
5) GSM 800 Module
6) GSM Sim
7) Connecting wires
8) Project base

SOFTWARE REQUIREMENTS:

- IBM Watson Platform
- Node Red

WORKING:

Step 1: A signal from the microcontroller will go to the display and show gas leakage message there.

Step 2: Simultaneously automatically turns on the DC fan to ventilate the leaked gas, and the source solenoid valve will be turns off

Step 3: Signal from microcontroller activates the GSM module and sends an alert SMS “ALERT GAS LEAKING” to the user’s mobile number.

CODING AND SOLUTION:

```
# Importing Required modules
import time
import sys
import ibmiotf.application # IBM IoT Watson Platform Module
import ibmiotf.device
import tkinter as tk # Python GUI Package
from tkinter import ttk # Python GUI
import time
from threading import Thread


organization = "sgoqkq" # Organization ID
```

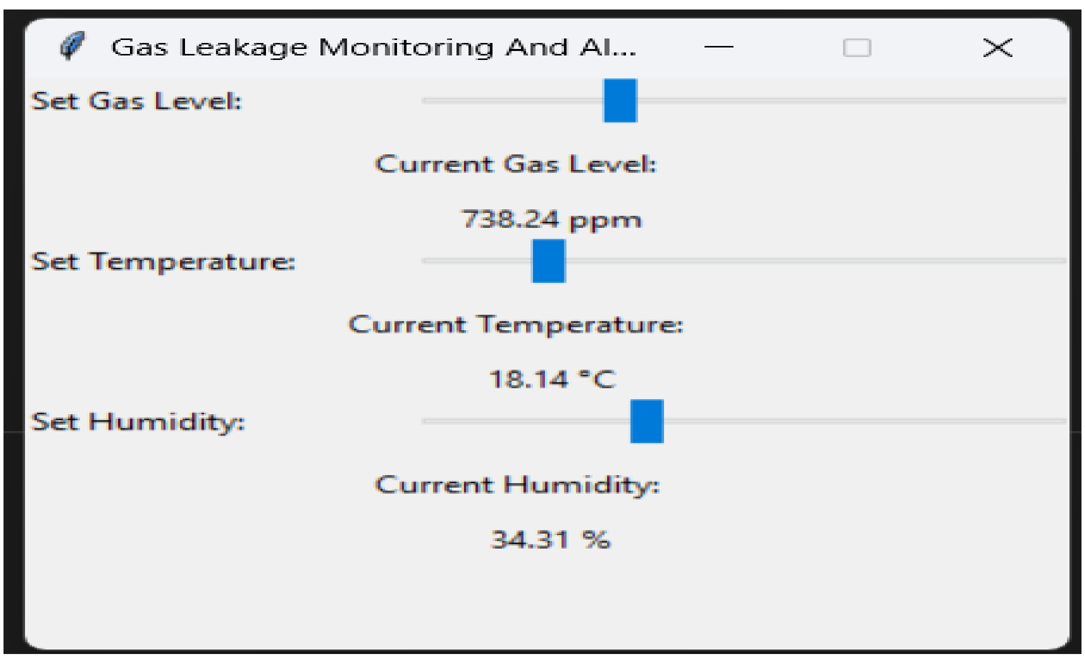
```
deviceType = "Gas_Leakage_Detection_Device"   # Device type
deviceId = "Gas_Leakage_Detection_Device1"    # Device ID
authMethod = "token"    # Authentication Method
authToken = "123456789" #Replace the authtoken


# Tkinter root window
root = tk.Tk()
root.geometry('350x300')  # Set size of root window
root.resizable(False, False) # root window non-resizable
root.title('Gas Leakage Monitoring And Alerting System for Industries
(PNT2022TMID05161)')


# Layout Configurations
root.columnconfigure(0, weight=1)
root.columnconfigure(1, weight=3)


# Temperature and Humidity sliders initialization
current_gas = tk.DoubleVar()
current_temp = tk.DoubleVar()
current_humid = tk.DoubleVar()


# slider - temperature and humidity functions
def get_current_gas(): # function returns current gas level value
    return '{: .2f}'.format(current_gas.get())

def get_current_temp():  # function returns current temperature value
    return '{: .2f}'.format(current_temp.get())


def get_current_humid(): # function returns current humidity value
    return '{: .2f}'.format(current_humid.get())


def slider_changed(event): # Event Handler for changes in sliders
    print('---------')
    print('Gas Level: {: .2f} , Temperature: {: .2f} , Humidity: {:
.2f}'.format(current_gas.get(),current_temp.get(),current_humid.get()))
    print('---------')
    gas_label.configure(text=str(get_current_gas()) +" ppm")  # Displays current gas level
as label content
```

```
    temp_label.configure(text=str(get_current_temp()) +" °C")  # Displays current
temperature as label content
    humid_label.configure(text=str(get_current_humid()) +" %") # Displays current
humidity as label content


# Tkinter Labels

# label for the gas level slider
slider_gas_label = ttk.Label(root,text='Set Gas Level:')
slider_gas_label.grid(column=0,row=0,sticky='w')


#  Gas Level slider
slider_gas = ttk.Scale(root,from_=200,to=2000,orient='horizontal',
command=slider_changed,variable=current_gas)
slider_gas.grid(column=1,row=0,sticky='we')


# current gas level label
current_gas_label = ttk.Label(root,text='Current Gas Level:')
current_gas_label.grid(row=1,columnspan=2,sticky='n',ipadx=10,ipady=10)


# Gas level label (value gets displayed here)
gas_label = ttk.Label(root,text=str(get_current_gas()) +" ppm")
gas_label.grid(row=2,columnspan=2,sticky='n')


# label for the temperature slider
slider_temp_label = ttk.Label(root,text='Set Temperature:')
slider_temp_label.grid(column=0,row=12,sticky='w')


#  temperature slider
slider_temp = ttk.Scale(root,from_=0,to=100,orient='horizontal',
command=slider_changed,variable=current_temp)
slider_temp.grid(column=1,row=12,sticky='we')


# current temperature label
current_temp_label = ttk.Label(root,text='Current Temperature:')
current_temp_label.grid(row=16,columnspan=2,sticky='n',ipadx=10,ipady=10)
```

```
# temperature label (value gets displayed here)
temp_label = ttk.Label(root,text=str(get_current_temp()) +" °C")
temp_label.grid(row=17,columnspan=2,sticky='n')


# label for the humidity slider
slider_humid_label = ttk.Label(root,text='Set Humidity:')
slider_humid_label.grid(column=0,row=20,sticky='w')


#  humidity slider
slider_humid=ttk.Scale(root,from_=0,to=100,orient='horizontal',command=slider_change
d,variable=current_humid)
slider_humid.grid(column=1,row=20,sticky='we')


# current humidity label
current_humid_label=ttk.Label(root,text='Current Humidity:')
current_humid_label.grid(row=34,columnspan=2,sticky='n',ipadx=10,ipady=10)


# humidity label (value gets displayed here)
humid_label=ttk.Label(root,text=str(get_current_humid()) +" %")
humid_label.grid(row=36,columnspan=2,sticky='n')

def publisher_thread():
    thread = Thread(target=publish_data)
    thread.start()

def publish_data():
    # Exception Handling
    try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-
method": authMethod,
                         "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
    # ............................................

    except Exception as e:
        print("Caught exception connecting device: %s" % str(e))
        sys.exit()

    deviceCli.connect()  # Connect to IBM Watson IoT Platform
```

```
    while True:
        temp = int(current_temp.get())
        humid = int(current_humid.get())
        gas_level = int(current_gas.get())

        # Send Temperature & Humidity to IBM Watson IoT Platform
        data = {'gas_level' : gas_level, 'temperature': temp, 'humidity': humid}

        def myOnPublishCallback():
            print("Published Gas Level = %s ppm" % gas_level, "Temperature = %s C" %
temp, "Humidity = %s %%" % humid, "to IBM Watson")

        success = deviceCli.publishEvent("event", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(1)

publisher_thread()

root.mainloop() # startup Tkinter GUI

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

TESTING - TEST CASES: OUTPUT



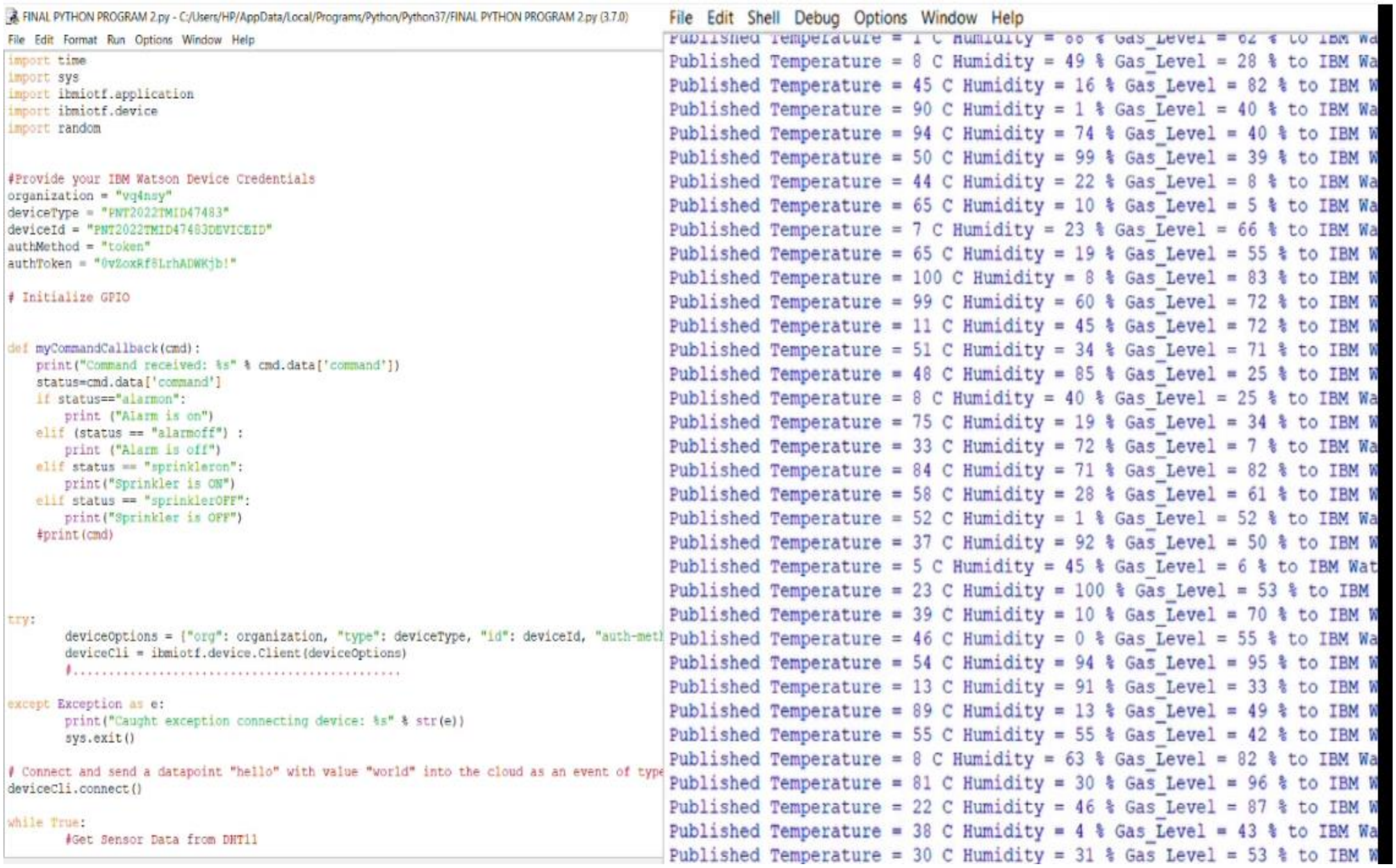
```
TERMINAL

--------
--------
Gas Level:  385.29 , Temperature:  31.86 , Humidity:  63.24
--------
--------
Gas Level:  385.29 , Temperature:  31.86 , Humidity:  62.75
--------
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 385 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
--------
Gas Level:  386.29 , Temperature:  31.86 , Humidity:  62.75
--------
Published Gas Level = 386 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 31 C Humidity = 62 % to IBM Watson
--------
Gas Level:  386.29 , Temperature:  32.86 , Humidity:  62.75
--------
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 62 % to IBM Watson
--------
Gas Level:  386.29 , Temperature:  32.86 , Humidity:  63.75
--------
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 63 % to IBM Watson
--------
Gas Level:  386.29 , Temperature:  32.86 , Humidity:  64.75
--------
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
Published Gas Level = 386 ppm Temperature = 32 C Humidity = 64 % to IBM Watson
```

CODE 2: FEATURE 2

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random


#Provide your IBM Watson Device Credentials
organization = "flp5bl"
deviceType = "iot"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO


def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="alarmon":
        print ("Alarm is on")
    elif (status == "alarmoff") :
        print ("Alarm is off")
    elif status == "sprinkleroff":
        print("Sprinkler is OFF")
    elif status == "sprinkleron":
        print("Sprinkler is ON")
    #print(cmd)




try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
"auth-method": authMethod, "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #..............................................
```

```
except Exception as e:
        print("Caught exception connecting device: %s" % str(e))
        sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud as an
event of type "greeting" 10 times
deviceCli.connect()

while True:
       #Get Sensor Data from DHT11

       temp=random.randint(0,100)
       Humid=random.randint(0,100)
       gas=random.randint(0,100)

       data = { 'temp' : temp, 'Humid': Humid, 'gas' : gas }
       #print data
       def myOnPublishCallback():
           print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %
Humid, "Gas_Level = %s %%" %gas, "to IBM Watson")

       success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
       if not success:
           print("Not connected to IoTF")
       time.sleep(1)

       deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

TEST CASE 2:



ADVANTAGES:

- Low maintenance and low operating costs.
- Reliable technology.
- Get immediate gas leak alerts
- Prevents fire hazards and Explosions

DISADVANTAGES:

- If any problem in sensor the completer system gets failed.
- Continue checking is needed.

Conclusion:

True prevention is not waiting for bad things to happen, it's preventing things from happening in the first place. Installing the proposed system will be one of the best ways to prevent accident. An industry that values employee’s life will always grow faster. This system can detect even a slightest leak in the entire system so not only you can prevent disasters but also you can efficiently track whether the gas is being used or being wasted due to leakage.

FUTURE SCOPE:

In future there will be a heavy demand for technology. So this project will helps to improve the Digitalization of the Industries.

APPENDIX:

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-8703-1658927257

DEMO LINK : https://drive.google.com/file/d/1MyqFAlqj7jj9_QggopCe7OjEp0Hrg0dk/view?usp=share_link