1. INTRODUCTION

1.1 Project Overview

To develop an Industry Specific Intelligent Fire Management System.

- The smart fire management system includes a Gas sensor, Flame sensor and temperature sensors to detect any changes in the environment.
- If any flame is detected the sprinklers will be switched on automatically to sprinkle the water.
- Emergency alerts are notified to the authorized owners and Fire station to take immediate postcautions.

1.2 Purpose

The primary purpose of fire alarm system is to provide an early warning of fire so that people can be evacuated & immediate action can be taken to stop or eliminate the fire effect as soon as possible.

2. LITERATURE SURVEY

2.1 Existing Problem

Fire monitoring systems have usually been based on a single sensor such as smoke or flame. These single sensor systems have been unable to distinguish between true and false presence of fire. Consuming energy all day long and being dependent on one sensor that might end with false alert is not efficient and environmentally friendly.

We need a system that is efficient not only in sensing fire accurately, but we also need a solution which is smart. In order to improve upon the results of existing single sensor systems, the smart fire management system includes a Gas sensor, Flame sensor and a temperature sensor. This system also requires a proper network with individual smart devices connected tovarious panels.

2.2 References

[1]. Dhananjeyan S, Mohana Sundaram K, Kalaiyarasi A, Kuppusamy PG. Design and development of blind navigation system using GSM and RFID Technology. Indian Journal of Science and Technology. 2016 Jan; 9(2):1–5.

[2]. Jotheeswaran J, Koteeswaran S. Feature selection using random forest method for sentiment analysis. Indian Journal of Science and Technology. 2016 Jan; 9(3):1–7

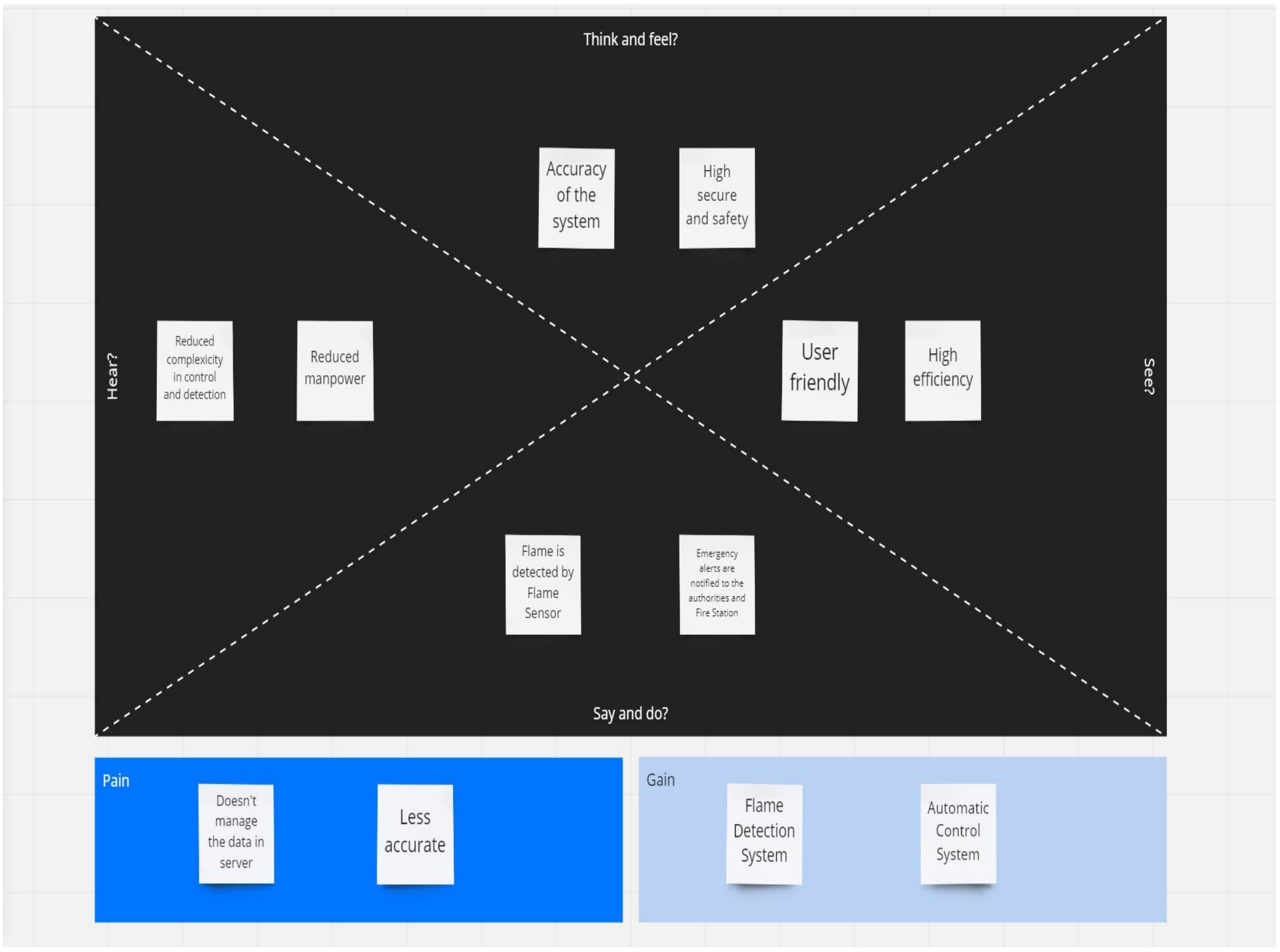
[3]. Goswami A, Bezboruah T, Sarma KC. Design of an embedded system for monitoring and controlling temperature and light. IJEER. 2009: 1(1): 27–36.

2.3 Problem Statement Definition

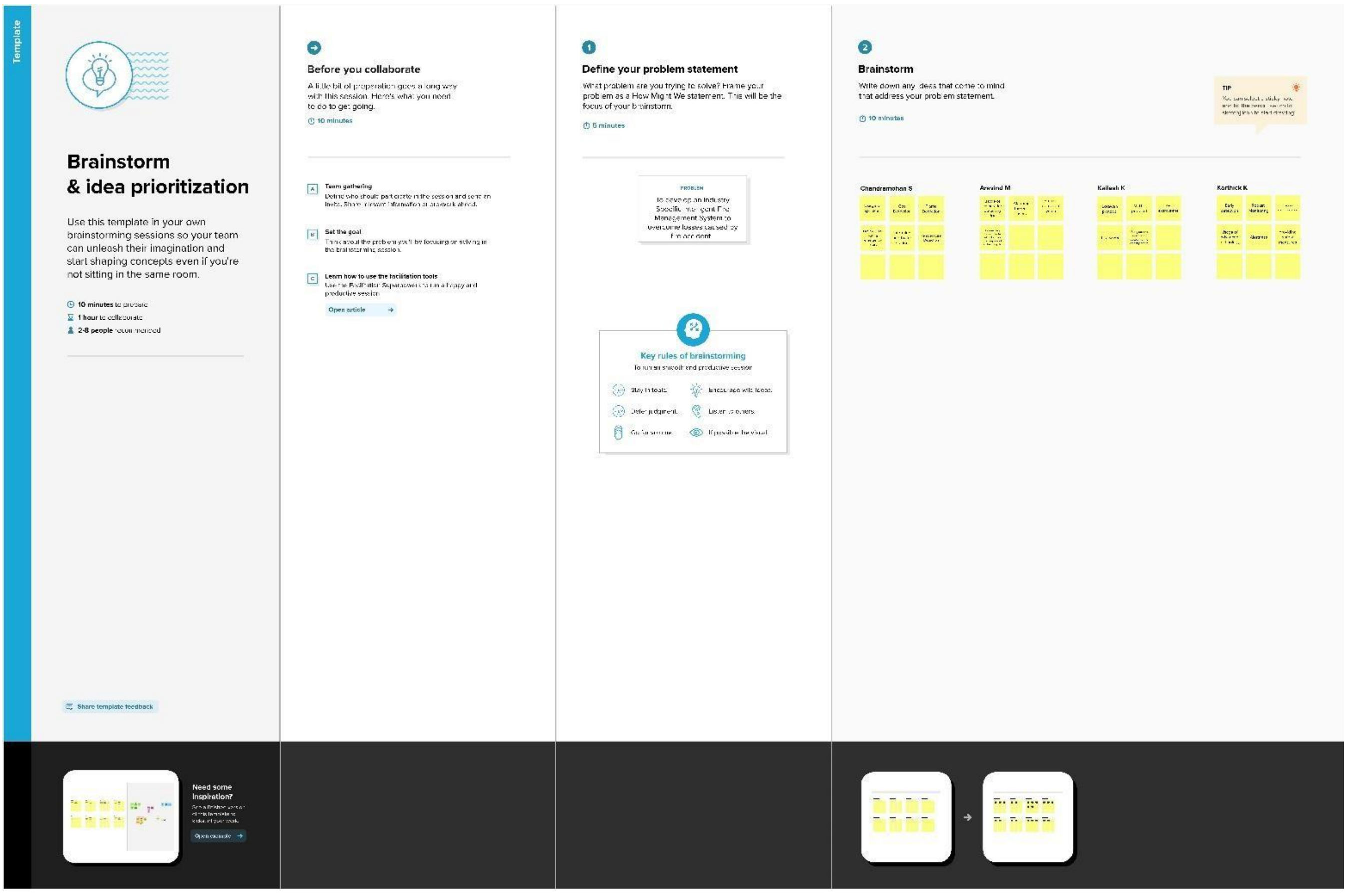
Industry Specific Intelligent fire management system are designed to prevent fire accident due to Gas leakage and flame in industry.

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming


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
To develop an Industry Specific Intelligent Fire Management System to overcome losses caused by fire accident

Key rules of brainstorming
To run an smooth and productive session

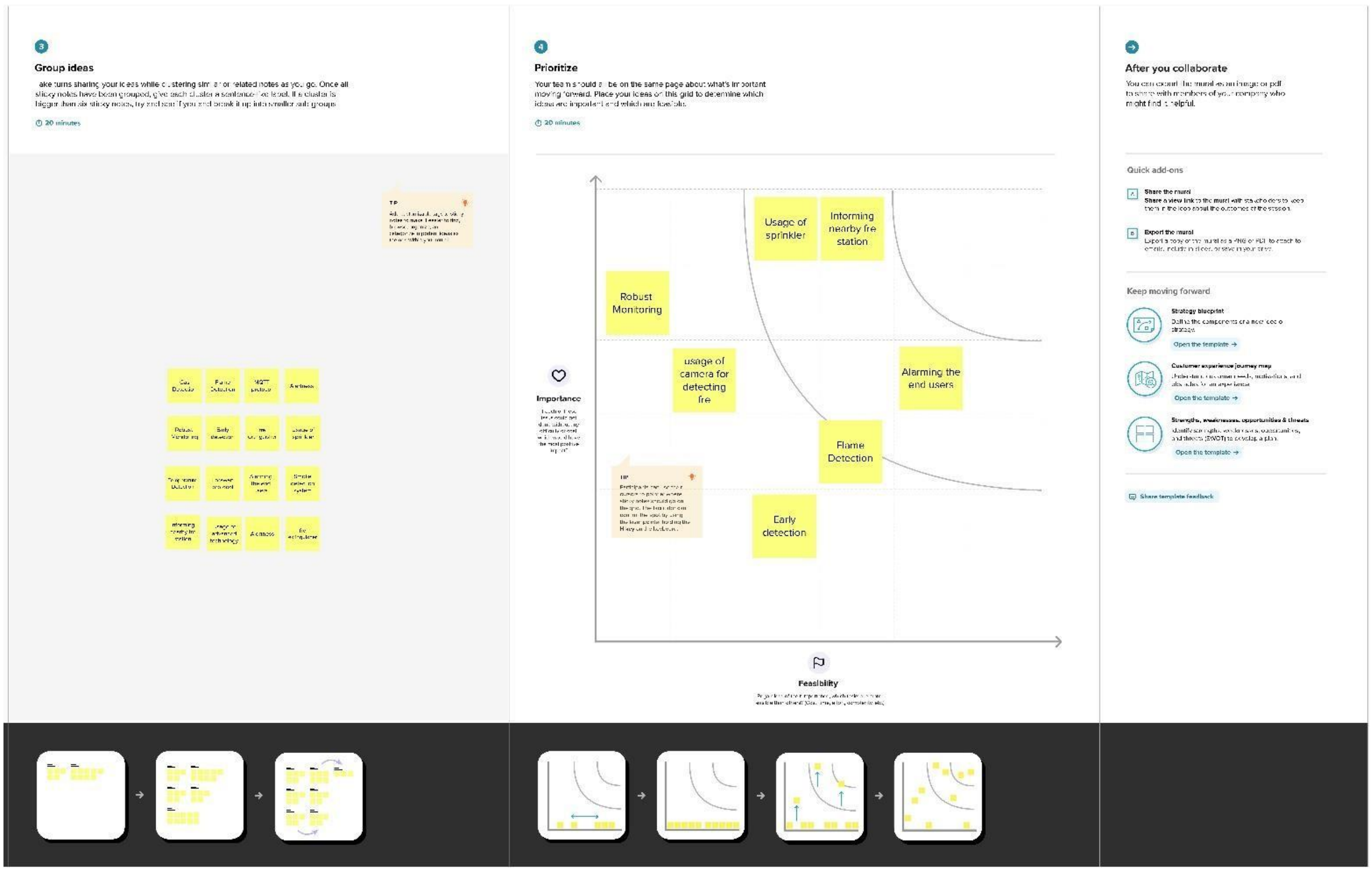
Stay in topic. | Encourage wild ideas.
Defer judgment. | Listen to others.
Go for volume. | If possible, be visual.

2 Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes

TIP
You can select a sticky note and hit the pencil icon to start drawing!

Chandramohan S
Aravind M
Kailesh K
Karthick K

Need some inspiration?
See a finished version of this template to kickstart your work.
Open example

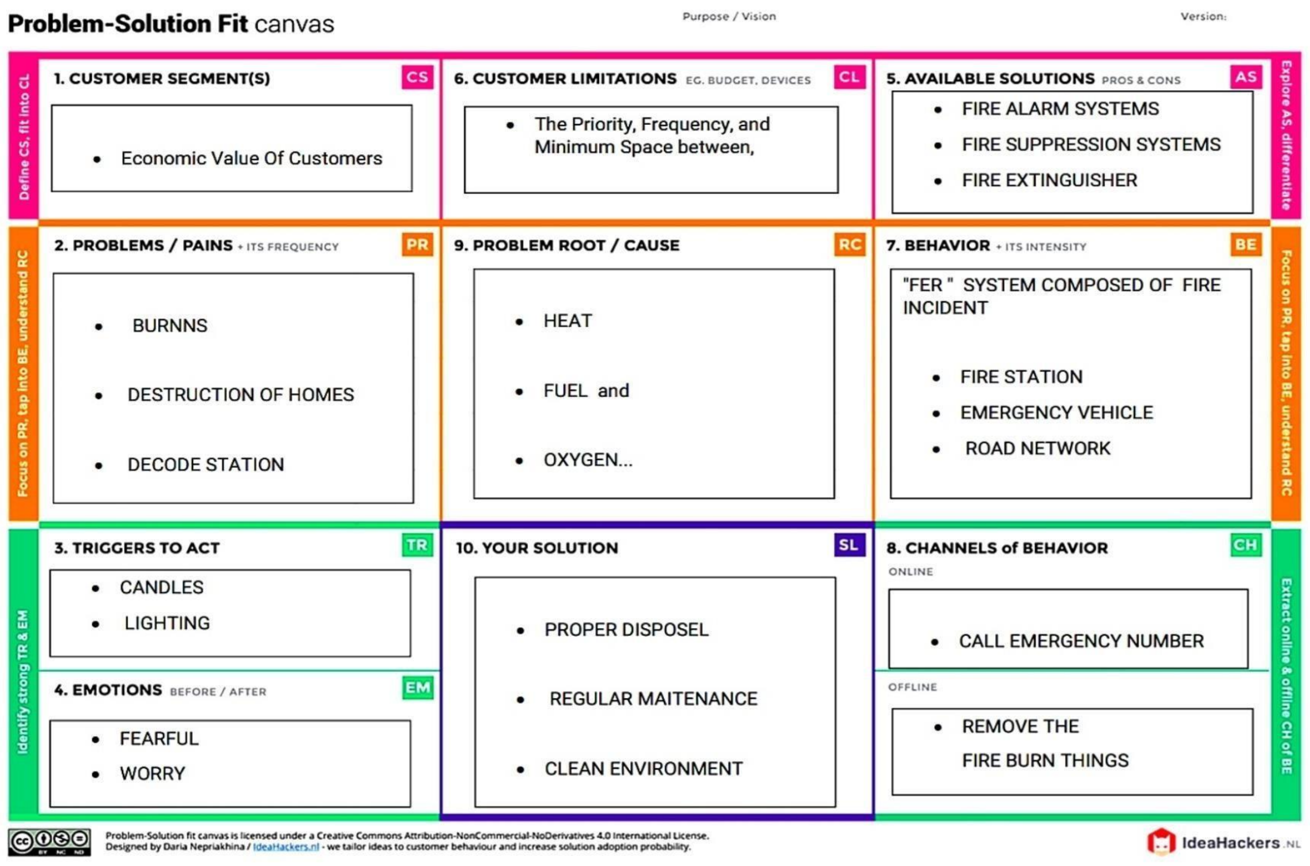

Group ideas
20 minutes
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
Usage of sprinkler
Informing nearby fire station
Robust Monitoring
usage of camera for detecting fire
Alarming the end users
Flame Detection
Early detection
Importance
Feasibility
After you collaborate
Quick add-ons
Share the mural
Export the mural
Keep moving forward
Strategy blueprint
Customer experience journey map
Strengths, weaknesses, opportunities & threats
Open the template
Share template feedback

3.3 Proposed Solution

S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	To improve the safety of the industries by using fire management and alarming system to reduce the loss of lives and properties .
2	Idea / Solution description	To implement the fire safety management in industry based on IOT using Arduino uno board as microcontroller with fire detection and alarming system. Here we are also using some sensors, such as Flame sensor, smoke sensor, gas sensor with GSM for notification system. And also Water sprinkler system used to shut down the fire immediately.

3	Novelty / Uniqueness	An integrated system of temperature monitoring, gas monitoring, alarming system, fire detection with automatic water sprinkler to shut down the fire and also GSM for sending message or notify the authority about the cause.
4	Social Impact / Customer Satisfaction	This system prevents the accident priorly before getting worse. Reduces the risk of lives and properties.
5	Business Model (Revenue Model)	This system can be implemented in any environment for the detection of fire and alerting people and reduce the loss. The most accurate predictions are made by using this system in safety management system.
6	Scalability of the Solution	This system is compact in size. It is easy to maintain as it requires very less time for maintenance. Cost of the system is reasonable.

3.4 Problem Solution Fit



Problem-Solution Fit canvas

Purpose / Vision

Version:

Define CS, fit into CL

1. CUSTOMER SEGMENT(S) CS

- Economic Value Of Customers

6. CUSTOMER LIMITATIONS EG. BUDGET, DEVICES CL

- The Priority, Frequency, and Minimum Space between,

5. AVAILABLE SOLUTIONS PROS & CONS AS

- FIRE ALARM SYSTEMS
- FIRE SUPPRESSION SYSTEMS
- FIRE EXTINGUISHER

Explore AS, differentiate

Focus on PR, tap into BE, understand RC

2. PROBLEMS / PAINS + ITS FREQUENCY PR

- BURNNS
- DESTRUCTION OF HOMES
- DECODE STATION

9. PROBLEM ROOT / CAUSE RC

- HEAT
- FUEL and
- OXYGEN...

7. BEHAVIOR + ITS INTENSITY BE

"FER " SYSTEM COMPOSED OF FIRE INCIDENT

- FIRE STATION
- EMERGENCY VEHICLE
- ROAD NETWORK

Focus on PR, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TO ACT TR

- CANDLES
- LIGHTING

4. EMOTIONS BEFORE / AFTER EM

- FEARFUL
- WORRY

10. YOUR SOLUTION SL

- PROPER DISPOSEL
- REGULAR MAITENANCE
- CLEAN ENVIRONMENT

8. CHANNELS of BEHAVIOR CH

ONLINE

- CALL EMERGENCY NUMBER

OFFLINE

- REMOVE THE FIRE BURN THINGS

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 Functional Requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through website google mail
FR-2	User Confirmation	Confirmation by means of Email or OTP
FR-3	User Login	Login through site or App using respective username and secret word
FR-4	User Access	Get to the app prerequisites
FR-5	User Upload	Client ought to be able to upload the information
FR-6	User Solution	Data report should be generated and delivered to user for per every 24 hours
FR-7	User Data Sync	API interface to increase to invoice system

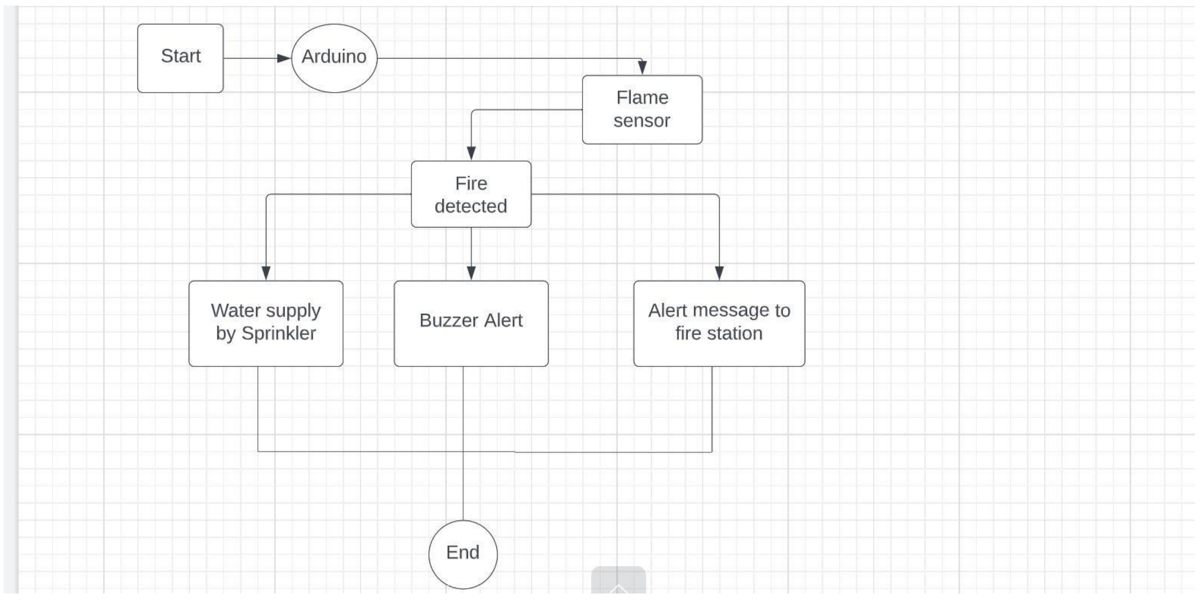
4.2 Non Functional Requirement

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Ease of use necessities can consider language barriers and localization assignments. Ease of use can be assessed from the underneath capacities. Productivity of use. Low seen workload. Simple and straightforward UI.
NFR-2	**Security**	Access permissions for the particular system information may only be changed by the system's data administrator.
NFR-3	**Reliability**	The database update process must roll back all related updates when any update fails.
NFR-4	**Performance**	The front-page stack time must be no more than 2 seconds for clients that get to the site utilizing an VoLTE versatile connection.

NFR-5	**Availability**	Modern module arrangement mustn't affect front page, item pages, and check out pages availability and mustn't take longer than one hour. The rest of the pages that will experience problems must show a notice with a timer showing when the framework is attending to be up once more
NFR-6	**Scalability**	Ready to increment adaptability by including memory, servers, or disk space. On the other hand, we can compress information, utilize optimizing calculations.

5. Project Design

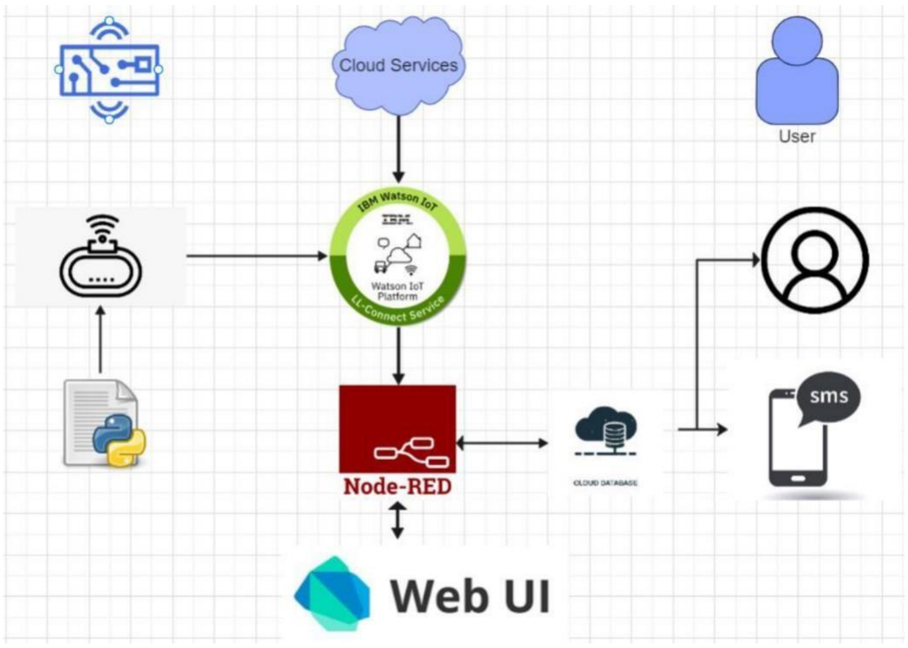
5.1 Data Flow Diagram



5.2 User Stories

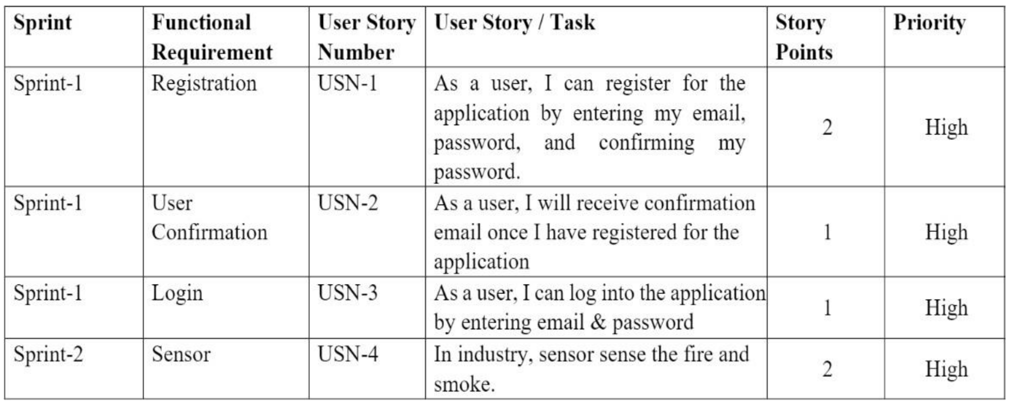
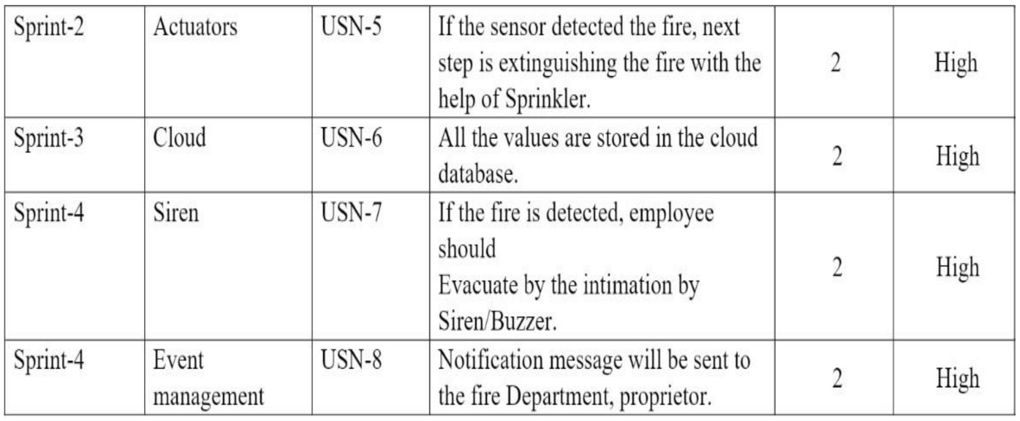
User Type	Functional requirement	User story number	User story/task	Acceptance criteria	Priority	Release
Customer (Mobile user, Web user, Care executive, Administrator)	Registration	USN-1	As a user, I can register for the application by entering my mail, password, and confirming my password	I can access my account/ dashboard	High	Sprint-1
		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
	Dashboard	USN-3	As a user, I can register for the application through internet	I can register & access the dashboard with Internet login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can confirm the registration in Gmail	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login with my id and password	High	Sprint-1

5.3 Solution Architecture



6. PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirement	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High
Sprint-1	User Confirmation	USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	1	High
Sprint-2	Sensor	USN-4	In industry, sensor sense the fire and smoke.	2	High
Sprint-2	Actuators	USN-5	If the sensor detected the fire, next step is extinguishing the fire with the help of Sprinkler.	2	High
Sprint-3	Cloud	USN-6	All the values are stored in the cloud database.	2	High
Sprint-4	Siren	USN-7	If the fire is detected, employee should Evacuate by the intimation by Siren/Buzzer.	2	High
Sprint-4	Event management	USN-8	Notification message will be sent to the fire Department, proprietor.	2	High

7. CODING AND SOLUTIONS

7.1 Feature 1

- IoT device
- IBM Watson Platform
- Node red
- Cloudant DB
- Web UI
- MIT App Inventor
- Python code

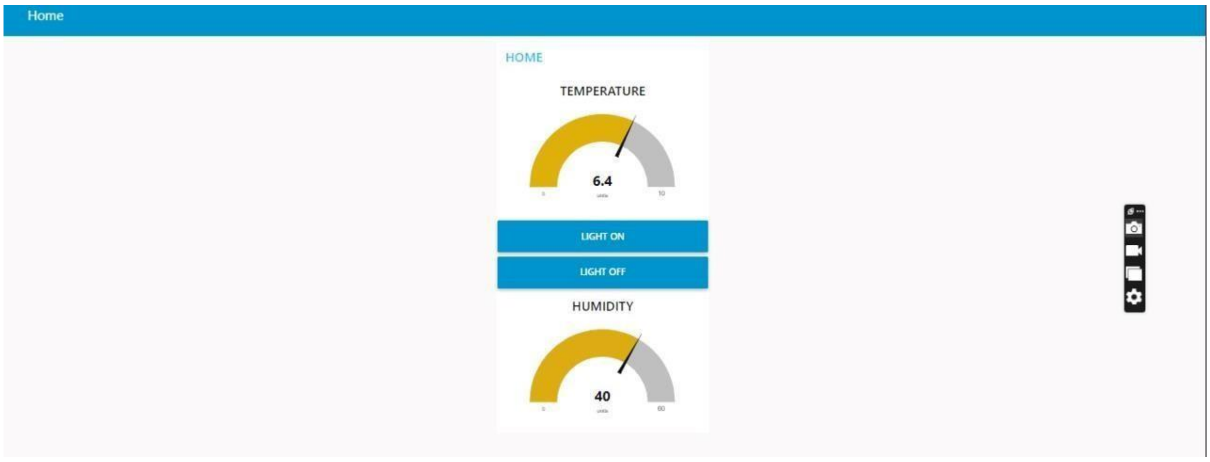
7.2 Feature 2

- Login
- Wokwi

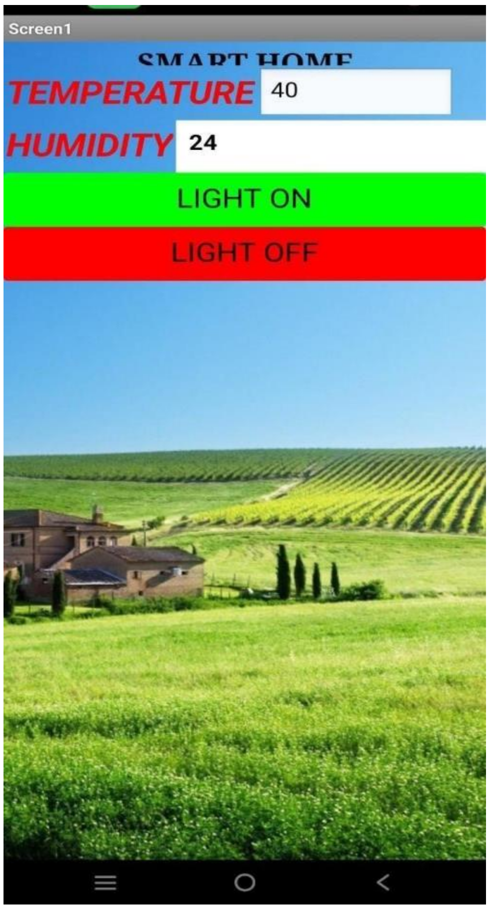
8. TESTING AND RESULTS

8.1 Test Cases

Test case 1



Test case 2



9. ADVANTAGES

- Reduced installation cost.
- They monitor 24/7.
- Improved security in homes, industries and Offices.
- It pin points location of the fire.

10. DISADVANTAGES

- Heat detectors are not considered as life saving devices because they are sensitive only to heat.
- High battery or current consumption will need for these detectors.
- Control panel may need to be replaced if it becomes damaged.

11. CONCLUSION

This gas leakage system can be applied for household safety and many other applications in the industry. Gas leakages and fire outbreaks in industries as wellas houses have lead to wide destruction and losses in the past. So here we proposed a system that detects gas as well as fire outbreaks and alert us accordingly so that proper action may be taken to control it.

12.FUTURE SCOPE

Smoke detectors and alarms are migrating from just the detection of smoke, to combination detectors and multicriteria detector. The future will be with multicriteria detection in which the detector will be more of a sensor, with the detection more for the products of combustion, such as carbon monoxide, carbondioxide, sulfur dioxide ,nitrogen dioxide in addition to heat and particulate matter.

13. APPENDIX

13.1 Source Code

Sprint I

```
 #include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
#define DHTPIN 15
#define DHTTYPE DHT22
#define LED 2
DHT dht (DHTPIN, DHTTYPE);
void callback(char* subscribetopic, byte* payload, unsigned intpayloadLength);
#define ORG "zbgr67"
#define DEVICE_TYPE "fershidevicetype"
#define DEVICE_ID "fershideviceid"
#define TOKEN "fershiageona"
String data3;
float t;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json"; char
subscribetopic[] = "iot-2/cmd/command/fmt/String"; char
authMethod[] = "usetoken-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); dht.begin();
pinMode(LED,OUTPUT); delay(10); Serial.println();
wificonnect(); mqttconnect();
} void
loop(){
t = dht.readTemperature();
Serial.print("temperature:");
Serial.println(t);
PublishData(t);
```

```
delay(1000);
if(!client.loop()){
mqttconnect();
}
}
void PublishData(float temp) {
mqttconnect();
String payload = "{\"temperature\":"; payload += temp; payload += "}";
Serial.print("Sending payload: ");
Serial.println(payload);
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish ok");
}else{
Serial.println("Publish failed");
}
}
void mqttconnect() {
if(!client.connected()){
Serial.print("Reconnecting client to ");
Serial.println(server);
while(!!!client.connect(clientId, authMethod, token)) {
Serial.print("."); delay(500);
}
initManagedDevice();
Serial.println();
}
}
void wificonnect(){
Serial.println();
Serial.print("Connecting to "); WiFi.begin("WokwiGUEST", "", 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++) {
(char)payload[i];
}
Serial.println("data: "+ data3);
if(data3=="lighton") {
Serial.println(data3); digitalWrite(LED,HIGH);
} else {
Serial.println(data3); digitalWrite(LED,LOW);
}
data3="";
}
```

Sprint III

```
#include <WiFi.h>

#include <Wire.h>

#include <SPI.h>

#include "ThingSpeak.h" #include

<WiFiClient.h>

unsigned long myChannelNumber = 2;

const char * myWriteAPIKey = "25V40ZAPI6KIZFGY"; int LED_PIN =

32; const int mq2 = 4; int value = 0;
```

```
int flame_sensor_pin = 10; lame_pin = HIGH;

char ssid[] = "NALAIYA"; char pass[]=

"NALAIYATHIRAN";

WiFiClient client;

#define PIN_LM35 39

#define ADC_VREF_mV 3300.0

#define ADC_RESOLUTION 4096.0

#define RELAY_PIN 17 #define RELAY_PIN1

27

void setup(){

Serial.begin(115200);

pinMode(RELAY_PIN, OUTPUT);

pinMode(RELAY_PIN1, OUTPUT);

Serial.print("Connecting to ");

Serial.println(ssid); WiFi.begin(ssid, pass); int wifi_ctr = 0;

while (WiFi.status() != WL_CONNECTED){ delay(1000);

Serial.print(".");

}

Serial.println("WiFi connected"); ThingSpeak.begin(client);

pinMode(LED_PIN, OUTPUT); pinMode(mq2, INPUT); pinMode (
    flame_sensor_pin , INPUT );

pinMode(BUZZER_PIN, OUTPUT); }

void temperature(){

int adcVal = analogRead(PIN_LM35);

float milliVolt = adcVal * (ADC_VREF_mV /

ADC_RESOLUTION); float tempC = milliVolt / 10;

Serial.print("Temperature: ");

Serial.print(temp
C);
Serial.print("°C");
if(tempC > 60){
Serial.println("Alert");
digitalWrite(BUZZER_PIN, HIGH);
}
else{
digitalWrite(BUZZER_PIN, LOW);
} int x = ThingSpeak.writeField(myChannelNumber,1, tempC,
myWriteAPIKey);
}
void GasSensors(){
int gassensorAnalogmq2 = analogRead(mq2);
Serial.print("mq2 Gas Sensor: ");
Serial.print(gassensorAnalogmq2);
Serial.print("\t");
Serial.print("\t");
Serial.print("\t");
if (gassensorAnalogmq2 > 1500){
Serial.println("mq2Gas");
Serial.println("Alert");
digitalWrite(RELAY_PIN1, HIGH);
}
else{
Serial.println("No mq2Gas");
digitalWrite(RELAY_PIN1, LOW);
delay(100);
```

```
} int a = ThingSpeak.writeField(myChannelNumber,4, gassensorAnalogmq2,
myWriteAPIKey);
}
void flamesensor(){
flame_pin = digitalRead( flame_sensor_pin );
if (flame_pin == LOW ){
Serial.println ( " ALERT: FLAME IS DETECTED" );
digitalWrite (BUZZER_PIN,HIGH ) ;
}
else{
Serial.println ( " NO FLAME DETECTED " );
digitalWrite (BUZZER_PIN , LOW );
}
int value = digitalRead(flame_sensor_pin);
if (value ==LOW) {
Serial.print("FLAME");
digitalWrite(RELAY_PIN, HIGH);
} else {
Serial.print("NO
FLAME");
digitalWrite(RELAY_PIN,
LOW); }
}
void
loop()
{
temperature();
```

```
  GasSensors();
  flamesensor();
}
```

Sprint IV

```
  #include <WiFi.h>
  #include <Wire.h>
  #include <SPI.h>
  #include "ThingSpeak.h"
  #include <WiFiClient.h>
  unsigned long myChannelNumber = 2;
  const char * myWriteAPIKey = "25V40ZAPI6KIZFGY";
  int LED_PIN = 32; const int mq2 = 4; int value = 0;
  int flame_sensor_pin = 10;
  lame_pin = HIGH;
  char ssid[] = "NALAIYA"; char
  pass[]= "NALAIYATHIRAN";
  WiFiClient client;
  #define PIN_LM35 39
  #define ADC_VREF_mV 3300.0
  #define ADC_RESOLUTION 4096.0
  #define RELAY_PIN 17
  #define RELAY_PIN1 27
  void setup(){
  Serial.begin(115200);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(RELAY_PIN1, OUTPUT);
```

```
Serial.print("Connecting to ");
Serial.println(ssid); WiFi.begin(ssid, pass);
int wifi_ctr = 0; while (WiFi.status() !=
WL_CONNECTED){ delay(1000);
Serial.print(".");
}
Serial.println("WiFi connected");
ThingSpeak.begin(client);
pinMode(LED_PIN, OUTPUT);
pinMode(mq2, INPUT); pinMode (
flame_sensor_pin , INPUT );
pinMode(BUZZER_PIN, OUTPUT); }
void temperature(){
int adcVal = analogRead(PIN_LM35);
float milliVolt = adcVal * (ADC_VREF_mV /
ADC_RESOLUTION); float tempC = milliVolt / 10;
Serial.print("Temperature: ");
Serial.print(temp
C);
Serial.print("°C");
if(tempC > 60){
Serial.println("Alert");
digitalWrite(BUZZER_PIN, HIGH);}
else{
digitalWrite(BUZZER_PIN, LOW);
} int x = ThingSpeak.writeField(myChannelNumber,1, tempC,
myWriteAPIKey);
```

```
}
void GasSensors(){
int gassensorAnalogmq2 = analogRead(mq2);
Serial.print("mq2 Gas Sensor: ");
Serial.print(gassensorAnalogmq2);
Serial.print("\t");
Serial.print("\t");
Serial.print("\t");
if (gassensorAnalogmq2 > 1500){
Serial.println("mq2Gas");
Serial.println("Alert");
digitalWrite(RELAY_PIN1, HIGH);}
else{
Serial.println("No mq2Gas");
digitalWrite(RELAY_PIN1, LOW);
delay(100);
} int a = ThingSpeak.writeField(myChannelNumber,4, gassensorAnalogmq2,
myWriteAPIKey);
}
void flamesensor(){
flame_pin = digitalRead( flame_sensor_pin );
if (flame_pin == LOW ){
Serial.println ( " ALERT: FLAME IS DETECTED" );
digitalWrite (BUZZER_PIN,HIGH ) ;}
else{
Serial.println ( " NO FLAME DETECTED " );
digitalWrite (BUZZER_PIN , LOW );
```

```
  }
  int value = digitalRead(flame_sensor_pin);
  if (value ==LOW) {
  Serial.print("FLAME");
  digitalWrite(RELAY_PIN, HIGH);
  } else {
  Serial.print("NO
  FLAME");
  digitalWrite(RELAY_PIN,
  LOW); }
  void  ()
  loop(){
  temperature();
  GasSensors();
  flamesensor();
}
```

13.2 GITHUB

Github link

https://github.com/IBM-EPBL/IBM-Project-17857-1659676763