1. **INTRODUCTION**
   1. Project Overview
   2. Purpose
2. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   1. Functional requirement
   2. Non-Functional requirements
5. **PROJECT DESIGN**
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
6. **PROJECT PLANNING & SCHEDULING**
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   1. Feature 1
   2. Feature 2
   3. Database Schema (if Applicable)
8. **TESTING**
   1. Test Cases
   2. User Acceptance Testing
9. **RESULTS**
   1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

GitHub & Project Demo Link

# INTRODUCTION

# Project Overview

The smart fire management system includes a gas, flame, and temperature sensor to detect any environmental changes. The exhaust fans are turned on based on the temperature readings and the presence of any gases. If a flame is detected, the sprinklers will automatically activate. Emergency alerts are sent to the authorities and the Fire Station.

# Purpose

➤ To provide a detect the status of the room using IoT devices

➤ To turn on sprinkler and exhaust fan when there is an accident

➤ To detect the flow of water

➤ To send and store the temperature status in a cloud storage

➤ To provide an easy management system on dashboard

➤ To provide an overview of what is happening to the user

# LITERATURE SURVEY

# Existing Problem

The situation is not ideal because fire management systems in homes and industries are not very reliable, efficient, or cost-effective, and lack advanced processing and features such as an automatic alert system for administrators and authorities. They are using older fire safety systems that cannot even activate the sprinkler system and do not communicate with one another properly to prevent false alarms.

They also monitor the entire system using applications.

* 1. **Reference** <https://pdfs.semanticscholar.org/f3e7/a7c0cf2d448be592421045033506e845e6c2.pdf> <https://www.mdpi.com/2224-2708/7/1/11>

# Problem Statement Definiton

The fire management system in houses and industries is not very reliable, efficient, cost effective, and does not have any advanced processing and does not have any features like automatic alert system for admin and authorities and in many buildings there are using older fire safety system that cannot even activate the sprinkler system and all of them do not communicate with each other properly to prevent false alarms.

# IDEATHON AND PROPOSED SOLUTION

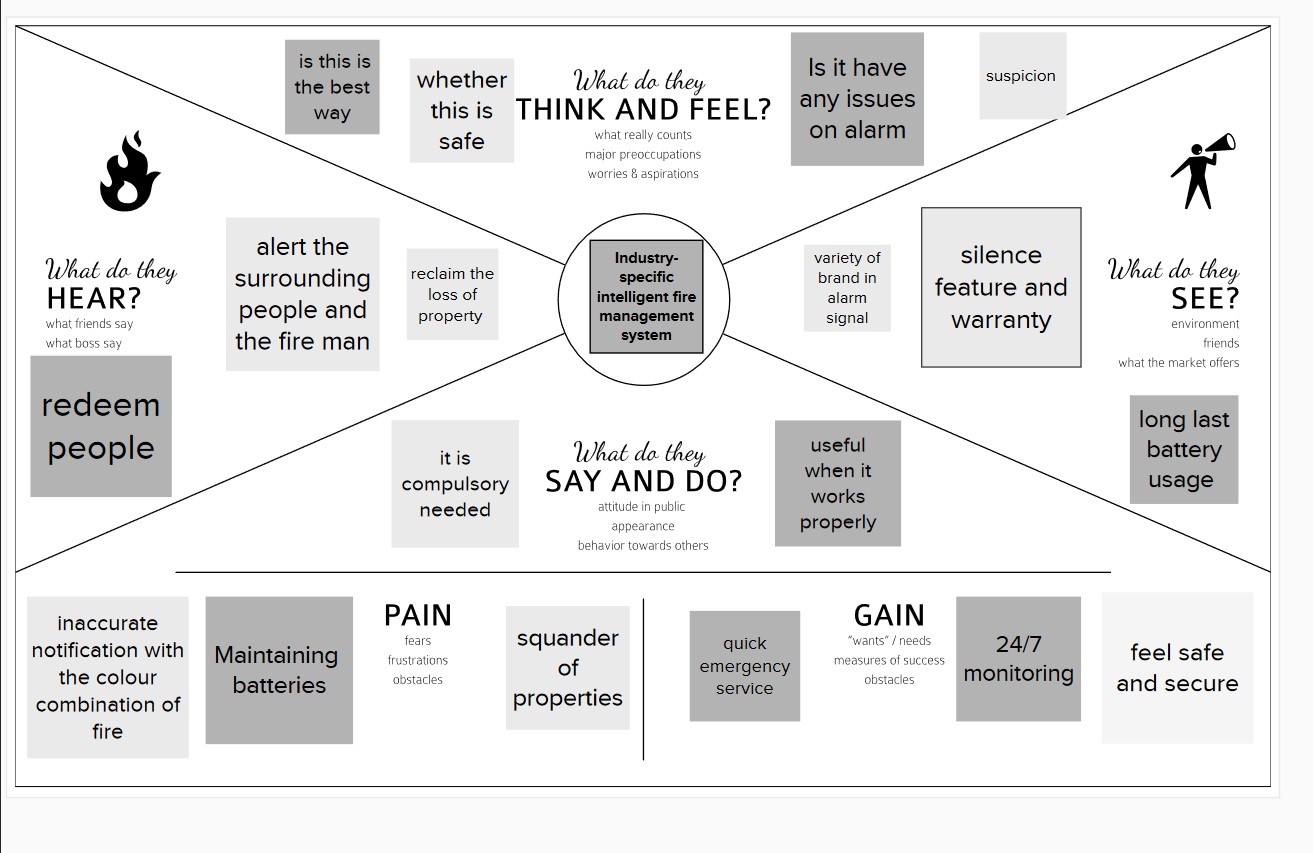
# Empathy Map Canvas

➤ An empathy map is a simple, easy-to-understand visual that captures knowledge about a user's behaviors and attitudes.

➤ It is a useful tool for assisting teams in better understanding their users.

➤ Creating an effective solution necessitates understanding the true problem and the person experiencing it.

➤ The map-making exercise helps participants consider things from the user's perspective, including his or her goals and challenges.



# Ideation and Brainstorming

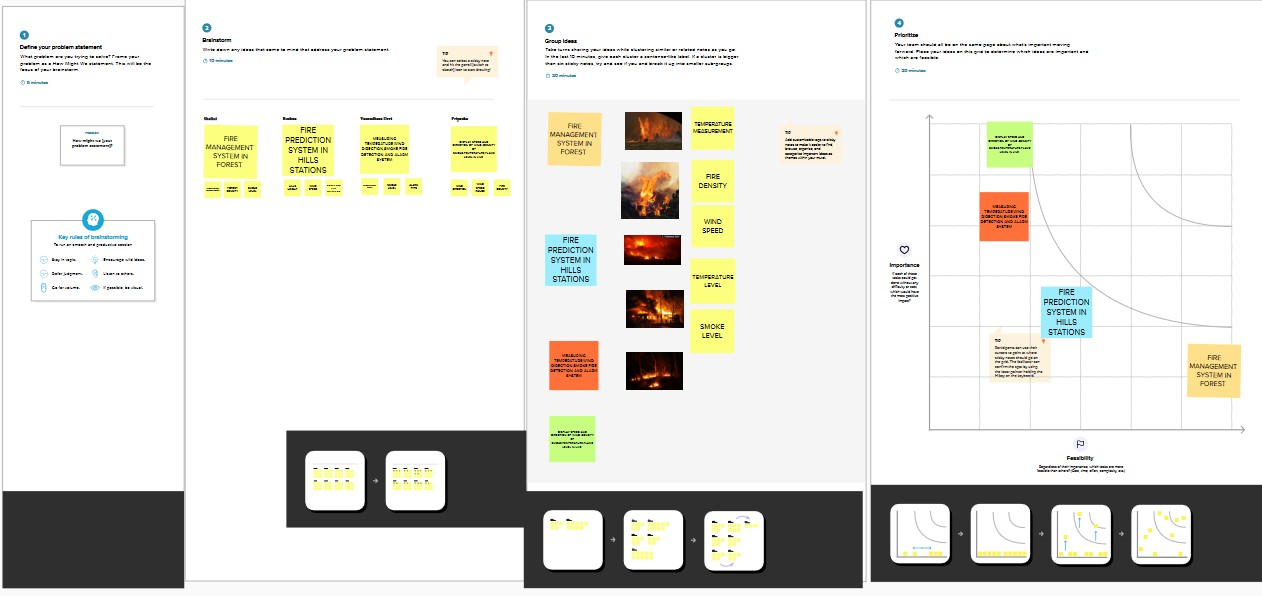
Step 1: Team Gathering, Collaboration and Select the Problem Statement

Team was gathered in mural app for collaboration

The team members are

* + - Shalini
    - Vasundhara devi
    - Roshna
    - Priyanka

Step 2: Brainstorm, Idea Listing and Grouping



Step3: Idea Prioritization

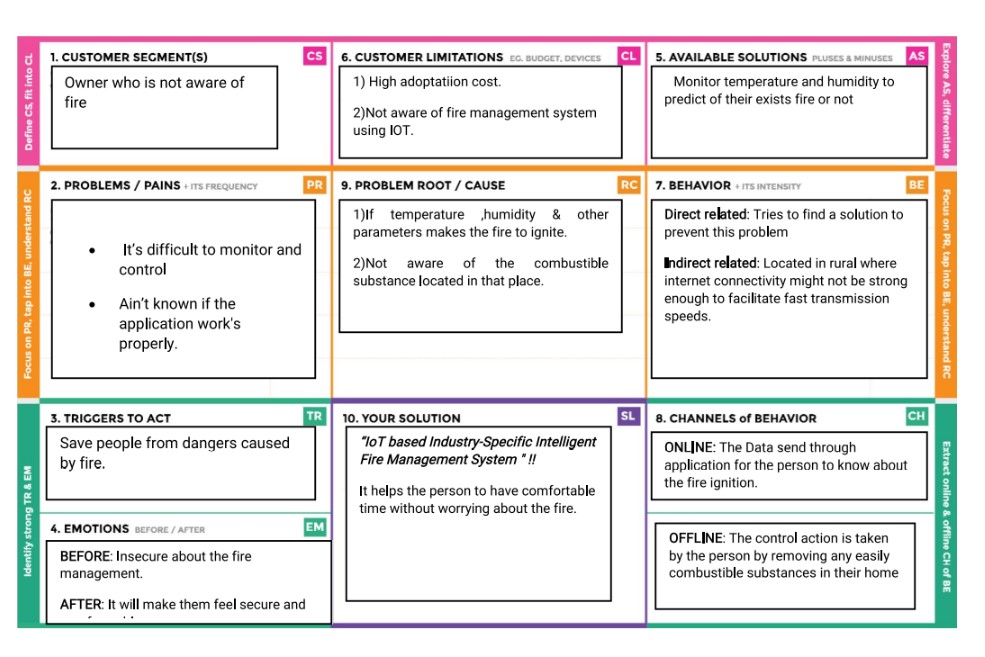
Graphical user interface

Description automatically generated

# Proposed Solution

|  |  |  |
| --- | --- | --- |
| **S No** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | To improve the safety management system in industries. Improving the safety management system against the fire incidents in industries. |
| 2. | Idea / Solution description | To implement the fire safety management in industry based on IOT using Arduino uno board with fire detection and fire extinguisher system. And using some sensors (Humidity sensor, Flame sensor, smoke sensor) with GPS tracking system. |
| 3. | Novelty / Uniqueness | An integrated system of temperature monitoring, gas monitoring, fire detection automatically fire extinguisher with accuracy of information about locations and response through SMS notification and call. |
| 4. | Social Impact / Customer Satisfaction | It early prevents the accident cost by fire in industries. Nearby locations so maximum extend more accurate reliability Compatibility design integrated system |
| 5. | Business Model (Revenue Model) | This product can be utilized by an industry. This can be thought of as a productive and helpful item as industries great many current rescuing people and machine from the fire accident. |
| 6. | Scalability of the Solution | It is trying to execute this technique as we need to introduce an Arduino gadget which was modified with an Arduino that takes received signals from sensors. Easily operatable and can be maintained. Required low time for maintain.  Cost is reasonable value |

* 1. **Proposed Solution Fit**

****

# REQUIREMENT ANALYSIS

* 1. **Functional Requirements**

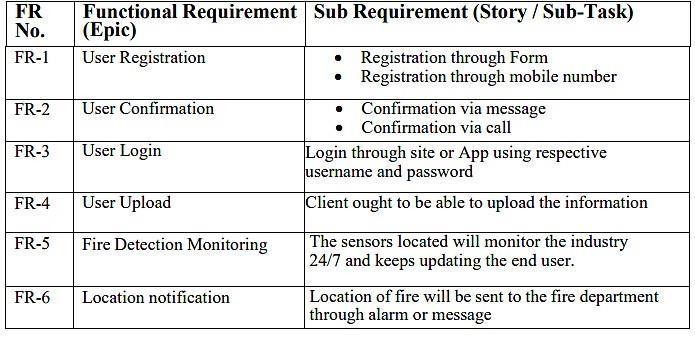
A functional requirement defines a system or component's function, where a function is

➤ Defined as a specification of behavior between inputs and outputs

➤ It defines "what the software system should do"

➤ Defined at the component level

➤ Usually simple to define

➤ Aids in testing the software's functionality

# Non - Functional Requirements

➤ A non-functional requirement defines a software system's quality attribute.

➤ It limits "How should the software system fulfill the functional requirements?"

➤ It is not required Applied to the entire system

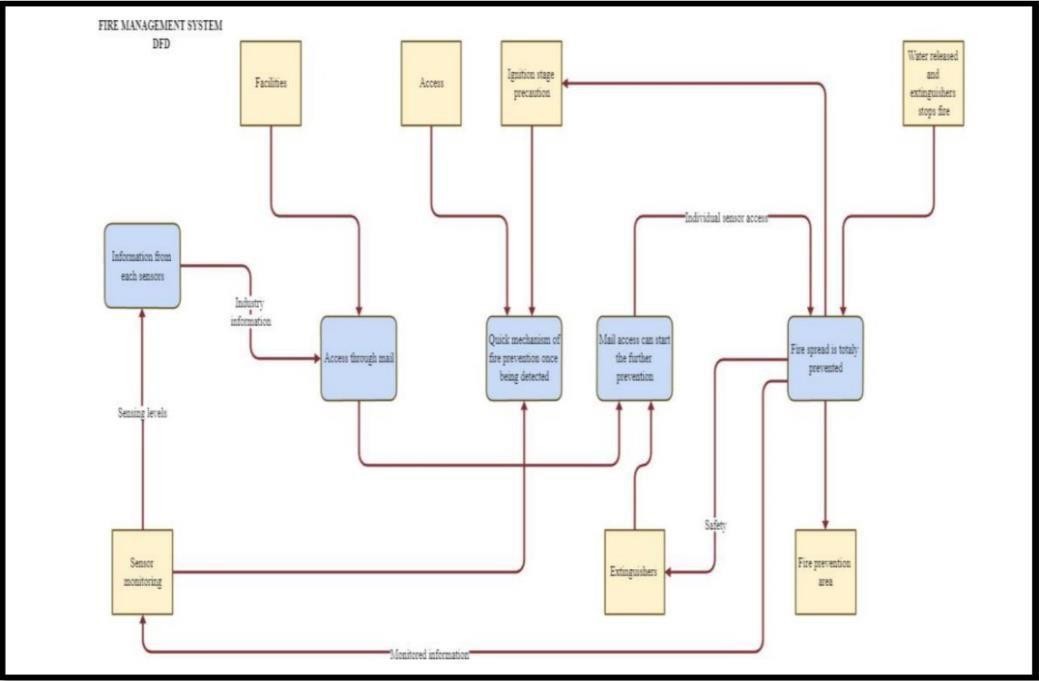
➤ Usually more difficult to define

➤ Aids in the verification of software performance

|  |  |  |
| --- | --- | --- |
| **FR . No** | **Non - Functional Requirement** | **Description** |
| NFR-1 | Usability | * It is the simple and Economic * Easy to use |
| NFR-2 | Security | * The software remains resilient in the face of attacks * The Web application is highly secured |
| NFR-3 | Reliability | * Response timer will be faster * It has high Reliability * The application runs accurately |
| NFR-4 | Performance | If Fire detected it will be immediately notified through the web application, and it also maintain track periodically. |
| NFR-5 | Availability | We will be Monitoring the Industry by day and Night (24/7). In case of Fire detected we willbe intimating the management rapidly. |

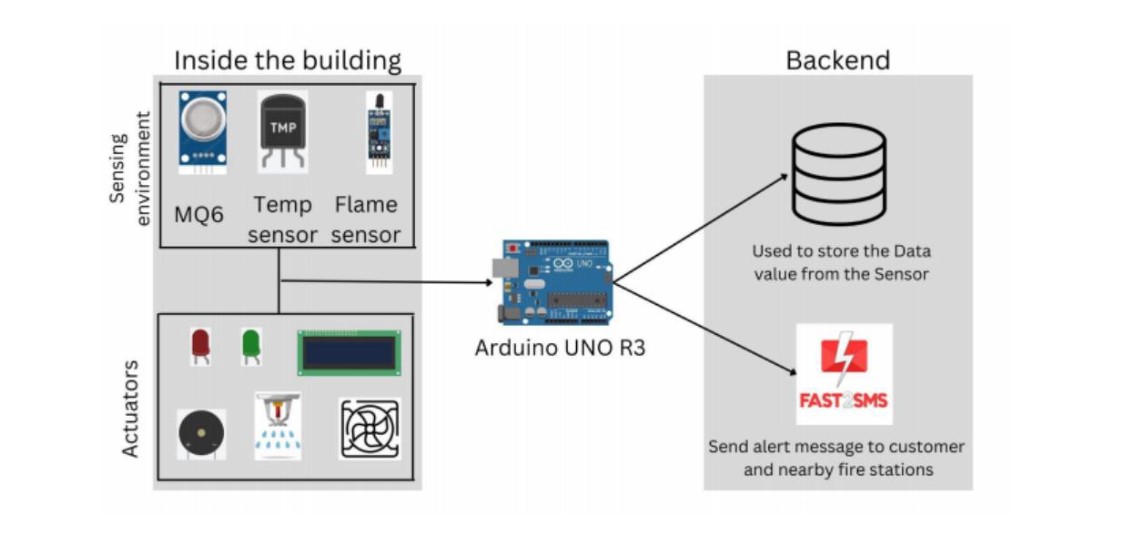
# PROJECT DESIGN

* 1. **Data flow Diagram**



# Solution and Technical

# Architecture Solution Architecture

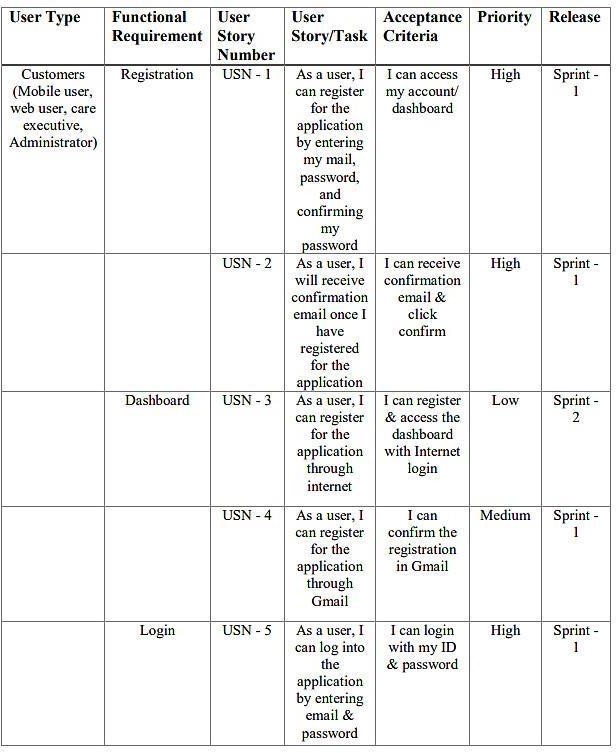
****

**Technical Architecture**

Diagram

Description automatically generated

# User Stories



1. **PROJECT DESIGN AND PLANNING**

# Sprint Planning and Estimation

**Table

Description automatically generated**

* 1. **Sprint Delivery Schedule**

**Table

Description automatically generated**

1. **CODING & SOLUTIONING**

# Feature 1: False Alarm Checking

if(temp < 45 ) { if(flame > 650 ) { accidentstatus = "Need Auditing";

if(canfanoperate) isfanon = true; else isfanon = false; issprinkon = false;

}

else if(flame <= 10)

{

accidentstatus = "nothing happened"; isfanon = false; issprinkon = false;

} }

else if(temp >= 45 && temp <= 55 )

{

if(flame <=650 && flame >100 )

{ if(cansprinkoperate) issprinkon

= true; else issprinkon = false; accidentstatus = "moderate"; if(gas > 160 && canfanoperate )

{ isfanon = true;

} else{ isfanon = false;

} }

else if(flame <= 100 && flame > 10)

{

if(cansprinkoperate) issprinkon = true; else issprinkon = false; isfanon = false;

accidentstatus = "moderate";

} }

else if(temp > 55){ if(flame > 650){

gas = 500 + rand()%500; accidentstatus = "severe"; if(cansprinkoperate) issprinkon = true; else issprinkon = false; if(canfanoperate) isfanon = true; else isfanon = false; }

else if(flame < 650 && flame > 400 )

{ gas = 300 + rand()%500; accidentstatus = "severe"; if(cansprinkoperate) issprinkon = true; else issprinkon = false; if(canfanoperate) isfanon = true; else isfanon = false;

} }

else {

accidentstatus = "Need moderate Auditing"; isfanon = false;

issprinkon = false; } if(issprinkon){

if(flow) { sprinkstatus = "working";

} else

{

sprinkstatus = "not working";

}

}

else if(!issprinkon)

{ sprinkstatus = "ready";

} else { sprinkstatus = "something's wrong";

}

## Explanation

➤ This set of code checks the false alarms and sets the current status

➤ It also handles the permission management of whether a device will work or not

## Feature 2

void PublishData(float temp, int gas ,int flame ,int flow,bool isfanon,bool issprinkon)

{

mqttconnect();

String payload = "{\"temp\":"; payload += temp; payload += "," "\"gas\":"; payload += gas;

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

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

payload += ((flow)?"true":"false");

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

payload += ((isfanon)?"true":"false"); payload += "," "\"issprinkon\":"; payload += ((issprinkon)?"true":"false"); payload

+= "," "\"cansentalert\":"; payload += ((cansentalert)?"true":"false"); payload

+= "," "\"accidentstatus\":"; payload += "\""+accidentstatus+"\""; payload += "," "\"sprinkstatus\":"; payload += "\""+sprinkstatus+"\""; payload += "}";

if (client.publish(publishTopic, (char\*) payload.c\_str())) { Serial.println("Publish ok");// if it sucessfully upload data on the

} else

{

Serial.println("Publish failed");

}

}

## Explanation

➤ It sends the data to IBM Watson Platform

## Feature 3

void callback(char\* subscribetopic, byte\* payload, unsigned int payloadLength)

{

Serial.print("callback invoked for topic: ");

Serial.println(subscribetopic); for (int i = 0; i < payloadLength; i++)

{ data3 += (char)payload[i];

}

Serial.println("data: "+ data3);

const char \*s =(char\*) data3.c\_str(); double pincode = 0;

if(mjson\_get\_number(s, strlen(s), "$.pin", &pincode)){ if(((int)pincode)==137153){ const char \*buf; int len;

if (mjson\_find(s, strlen(s), "$.command", &buf, &len))

{

String command(buf,len); if(command=="\"cantfan\""){ canfanoperate

= !canfanoperate;

}

else if(command=="\"cantsprink\""){ cansprinkoperate = !cansprinkoperate;

}else if(command=="\"sentalert\""){ resetcooldown();

}

}

}

} data3="";

}

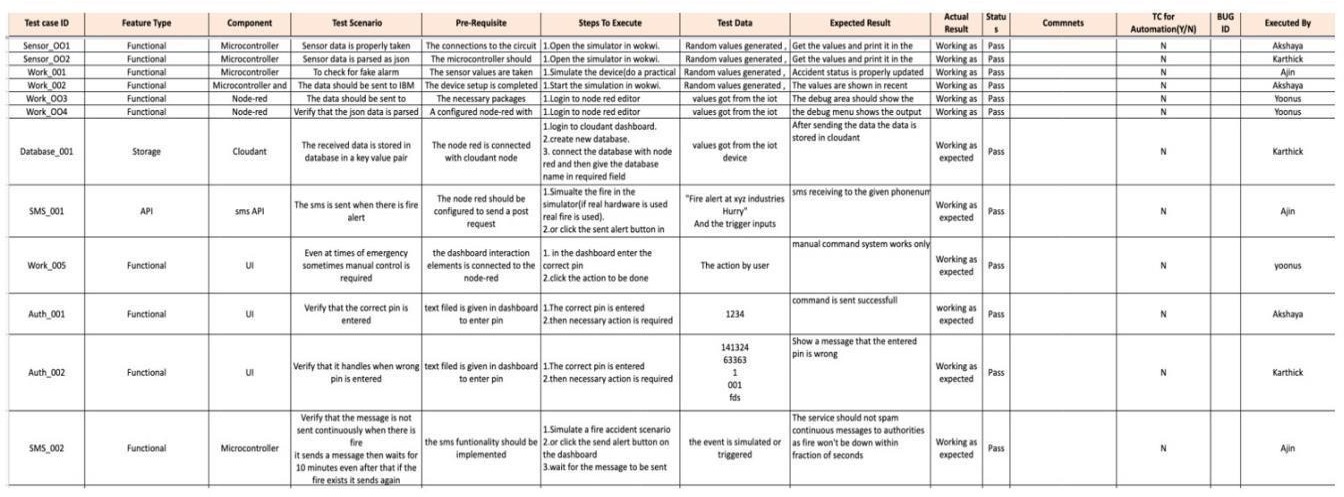
## Explanation

➤ The user's action is received as a command and stored in a buffer

➤ The event in the device is performed in accordance with the command

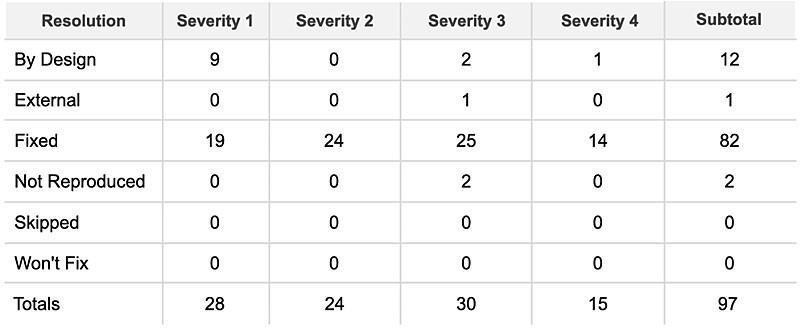
➤ It searches for a secret encrypted pin to perform that event

## TESTING

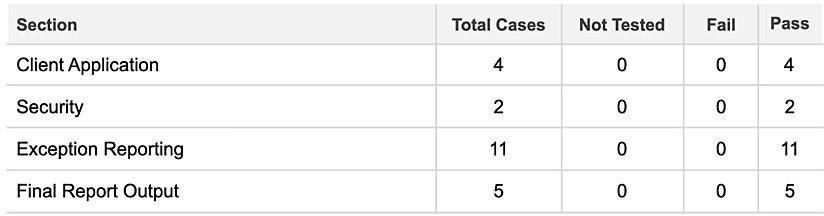


**8.2UAT**

## Defect Analysis



**Test Case Analysis**



## RESULTS

* 1. **Performance Metrics**

## CPU Usage:

The micro version of C++ makes the most efficient use of the CPU. The program runs in O(1) time for each loop, ignoring the network and communication. To improve communication with MQTT, the program sleeps every 1 second. Because the program runs in O(1) time and the compiler optimizes it during compilation, there is less CPU load per cycle. The following instructions are stored on the stack memory and can be popped after execution.

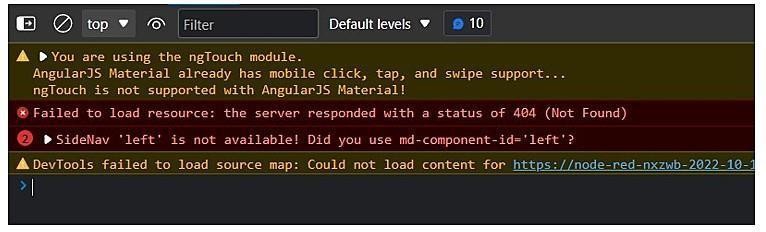
## Memory Usage:

The sensor values and networking data are saved in the ESP32's sram. It's a lot of information because the ESP32 only has 520 KB of memory. To save memory and ensure optimal program execution, the exact addresses for each memory cycle are overwritten with new values.

## Error Rates:

The error rates are very low because the backend and dashboard are handled with node-red.

Exceptions are handled properly so that the system's usability is not affected.

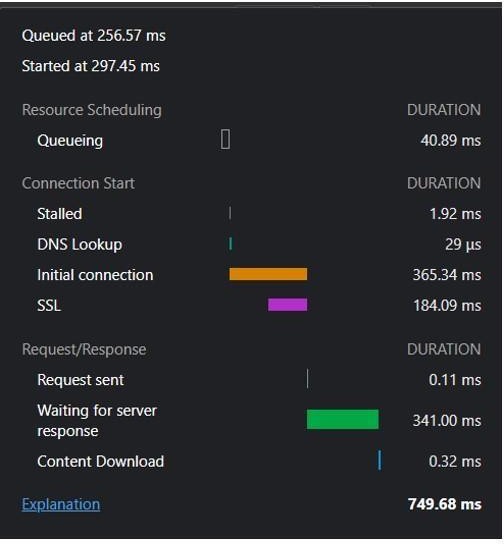


## Latency and Respose Time:

The DOM handling of the received data is optimal and latency is low .After the DOM is loaded the entire site is loaded to the browser.



In addition, the server responds quickly. The average response time is acceptable.

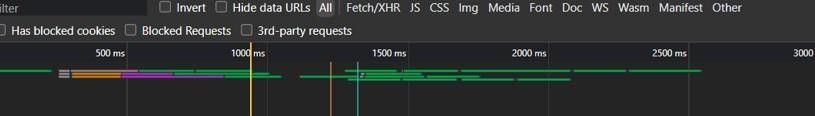


For the data sent from the IoT Device (considering the sleep of one second from the IoT), the response is much faster. We can see the delay caused by the sleep function.

The average time is well over optimal value

Average Time = (5ms + 2600ms)/2

= 1302.5



## Garbage Collection:

The Node framework handles garbage collection on the server side. C++ does not have garbage collection features in IoT devices. However, in this case, it is not necessary because the memory will be used again to store the data. There is no allocation of any dangling pointers or poorly handled address space.

## ADVANTAGES & DISADVANTAGES

**Advantages:**

➤ Active detection of gas leaks and fire outbreaks

➤ SMS alerting of administrators and fire authorities

➤ Turning on/off sprinklers and exhaust fans automatically

➤ To manually turn on/off sprinklers and exhaust fans, as well as send SMS alerts, authentication is required

➤ It detects false fire outbreaks automatically, reducing unnecessary panic

➤ We can confirm that the sprinkler system is functioning properly by using flow sensors

➤ A dashboard can display the status of any device

➤ The dashboard can be viewed by users via a web application

➤ The dashboard can be viewed by users via a web application

## Disadvantages:

➤ Always require an internet connection [only to send the SMS alert]

➤ If the physical device fails, the entire operation fails

➤ Because a large amount of data is stored in the cloud database every second, a large database is required

## CONCLUSION

So we conclude that, our problem premise is solved using IoT devices by developing a smart management system that solves many inherent problems in traditional fire management systems, such as actively monitoring for fire breakouts and gas leakage and sending SMS alerts to administrators and fire authorities.

## FUTURE SCOPE

The existing devices can be modified to work in various specialized environments, as well as scaled to house use to large labs [Because fire accidents can cause significant loss of human lives in homes to large industries], as well as used in public places and vehicles.

## APPENDIX

**ESP32 - Microcontroller:**

The ESP32 is a low-cost, low-power system-on-a-chip microcontroller family with integrated Wi-Fi and dual-mode Bluetooth.

* Memory: 320 KiB SRAM
* CPU: Tensilica Xtensa LX6 Microprocessor @ 160 or 240 MHz
* Power: 3.3 VDC
* Manufacturer: Espressif Systems
* Predecessor: ESP8266

## Sensors:

**DHT22 - Temperature & Humidity Sensor:**

The DHT22 is a simple and inexpensive digital temperature and humidity sensor. It measures the surrounding air with a capacitive humidity sensor and a thermistor and outputs a digital signal on the data pin (no analog input pins needed).

## Flow Sensors:

A flow sensor (also known as a "flow meter") is an electronic device that measures or controls the flow rate of liquids and gases through pipes and tubes.

## MQ5 - Gas Sensor:

Gas sensors (also referred to as gas detectors) are electronic devices that detect and identify various types of gasses. They are frequently used to detect toxic or explosive gases as well as to measure gas concentration.

## Flame Sensor:

A flame-sensor is a type of detector that is intended to detect and respond to the occurrence of a fire or flame. The response to flame detection can be affected by its fitting.

## Source Code:

#include <WiFi.h>//library for wifi

#include <PubSubClient.h>

#include "DHT.h"// Library for dht11

#define DHTPIN 15 // what pin we're connected to

#define DHTTYPE DHT22

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected

void callback(char\* subscribetopic, byte\* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "vg9s67"//IBM ORGANITION ID

#define DEVICE\_TYPE "sprint003"//Device type mentioned in ibm watson IOT Platform

#define DEVICE\_ID "spsprint003"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN "1234567890" //Token

String data3;

float Humidity, Temp;

//-------- Customise the above values --------

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name

char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send

char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING

char authMethod[] = "use-token-auth";// authentication method

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;//client id

//-----------------------------------------

WiFiClient wifiClient; // creating the instance for wificlient

PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing parameter like server id,portand wificredential

void setup()// configureing the ESP32

{

Serial.begin(115200);

dht.begin();

delay(10);

Serial.println();

wificonnect();

mqttconnect();

}

void loop()// Recursive Function

{

Humidity = dht.readHumidity();

Temp = dht.readTemperature();

Serial.print("Temp:");

Serial.println(Temp);

Serial.print("Humidity:");

Serial.println(Humidity);

PublishData(Temp,Humidity);

delay(1000);

if (!client.loop()) {

mqttconnect();

}

}

/\*.....................................retrieving to Cloud...............................\*/

void PublishData(float Temp, float Humidity) {

mqttconnect();//function call for connecting to ibm

/\*

creating the String in in form JSon to update the data to ibm cloud

\*/

String payload = "{\"Temp\":";

payload += Temp;

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

payload += Humidity;

payload += "}";

Serial.print("Sending payload: ");

Serial.println(payload);

if (client.publish(publishTopic, (char\*) payload.c\_str())) {

Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish ok in Serial monitor or else it will print publish failed

} else {

Serial.println("Publish failed");

}

}

void mqttconnect() {

if (!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

initManagedDevice();

Serial.println();

}

}

void wificonnect() //function defination for wificonnect

{

Serial.println();

Serial.print("Connecting to ");

WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the connection

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void initManagedDevice() {

if (client.subscribe(subscribetopic)) {

Serial.println((subscribetopic));

Serial.println("subscribe to cmd OK");

} else {

Serial.println("subscribe to cmd FAILED");

}

}

void callback(char\* subscribetopic, byte\* payload, unsigned int payloadLength)

{

Serial.print("callback invoked for topic: ");

Serial.println(subscribetopic);

for (int i = 0; i < payloadLength; i++) {

//Serial.print((char)payload[i]);

data3 += (char)payload[i];

}

}

}

//handles commands from user side

void callback(char\* subscribetopic, byte\* payload, unsigned int payloadLength)

{

Serial.print("callback invoked for topic: "); Serial.println(subscribetopic); for (int i = 0; i < payloadLength; i++)

{ data3 += (char)payload[i];

}

Serial.println("data: "+ data3); const char \*s =(char\*) data3.c\_str(); double pincode = 0; if(mjson\_get\_number(s, strlen(s), "$.pin", &pincode))

{ if(((int)pincode)==137153)

{

const char \*buf; int len; if (mjson\_find(s, strlen(s), "$.command", &buf, &len)) // And print it

{

String command(buf,len); if(command=="\"cantfan\""){

//this works when there is gas sensor reads high value and if there should be a //manual trigger else it will be automate canfanoperate = !canfanoperate;

}

else if(command=="\"cantsprink\"")

{ cansprinkoperate =

!cansprinkoperate; }

else if(command=="\"sentaert\""){

//this works when there is accident status is severe and if there should be a

//manual trigger else it will be automate resetcooldown();

}

}

}

}

data3="";

}

void resetcooldown()

{

cooldown = 0;

}

//sent alert request to node-red void sendalert()

{

cansentalert= true;

cooldown = 0;

}

**Github Link:** <https://github.com/IBM-EPBL/IBM-Project-53500-1661412276>

**Demo Video:**  <https://drive.google.com/file/d/1kCb9iU2cuCUBZSTWDU9J_vmChAyU-_aS/view?usp=drivesdk>