INDUSTRY - SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

PROJECT REPORT

TEAM ID	PNT2022TMID19220
FACULTY MENTOR	SARMILA K B

TEAM MEMBERS:

HARI HARAN S **(722819104039)**

HARI PRASATH T **(722819104041)**

JAYAVARDHAN P **(722819104050)**

MANIESH M **(722819104074)**

Title	Page No
1. INTRODUCTION	
1.1. Project Overview	3
1.2. Purpose	3
2. LITERATURE SURVEY	
2.1. Existing Problem	3
2.2. References.	4
2.3. Problem Statement Definition	5
3. IDEATION & PROPOSED SOLUTION	
3.1. Empathy Map Canvas.	5
3.2. Ideation & Brainstorming	6
3.3. Proposed Solution	7
3.4. Problem Solution Fit	8
4. REQUIREMENT ANALYSIS	
4.1. Functional Requirement.	8
4.2. Non - Functional Requirement.	9
5. PROJECT DESIGN	
5.1. Data Flow Diagram	10
5.2. Solution & Technical Requirements.	11
5.3. User Stories	12
6. PROJECT PLANNING & SCHEDULING	
6.1. Sprint Planning & Estimation	13
6.2. Sprint Delivery Schedule.	13
6.3. Report from JIRA	14
7. CODING & SOLUTIONING	
7.1. Web Application	16
7.2. Node-red web Dashboard	35
7.3. Wokwi Simulation	36
7.4. Python Simulation	44
7.5. Mobile Application	46
8. TESTING	
8.1. Test Cases.	47
8.2. User Acceptance Testing	49
9. RESULTS.	
9.1. Performance Metrics.	50
10. ADVANTAGES & DISADVANTAGES	52
11. CONCLUSION.	52
12. FUTURE SCOPE.	53
13. APPENDIX	53
Github & Project Demo Link	54

1. INTRODUCTION

1.1. Project Overview

A gas, flame, and temperature sensor is part of the smart fire management system to monitor any modifications in the environment. Based on the temperature data and the presence of any gases, the exhaust fans are turned on. The sprinklers will automatically turn on if a flame is found. The Fire Station and the authorities receive emergency alerts.

1.2. Purpose

- To provide an IoT device that can detect the status of the room.
- When there is an accident, to activate the sprinkler system and exhaust fan.
- To identify water movement.
- The temperature status will be sent and stored in a cloud storage.
- To offer a simple dashboard management system.

2. LITERATURE SURVEY

2.1. Existing Problem

The scenario is not optimal since fire management systems in homes and corporations lack advanced processing and features, such as an automatic alarm system for administration and authorities, and are not very dependable, efficient, or cost-effective. They are deploying outdated fire protection technologies that can't even turn on the sprinkler system and can't effectively communicate with one another to avoid false alerts. Applications are used to analyse the overall structure as well.

2.2. Reference

1. Rehman A, Qureshi MA, Ali T, Irfan M, Abdullah S, Yasin S, Draz U, Glowacz A, Nowakowski G, Alghamdi A, Alsulami AA. Smart Fire Detection and Deterrent System for Human Savior by Using Internet of Things (IoT). Energies.

2. Johnsaida N, Rahul LV, Shalini T. IOT Based Smart Fire Emergency Response System. International Journal for Advance Research and Development.

3. Mani V, Abhilasha G, Lavanya S. Iot based smart energy management system. International Journal of Applied Engineering Research.

4. Wehmeier G, Mitropetrosb K. Fire protection in the chemical industry. CHEMICAL ENGINEERING.

5. Zhang YC, Yu J. A study on the fire IOT development strategy. Procedia Engineering.

6. Schultz CA, Mclntyre KB, Cyphers L, Kooistra C, Ellison A, Moseley C. Policy design to support forest restoration: The value of focused investment and collaboration.

7. Montorio R, Pérez-Cabello F, Alves DB, García-Martín A. Unitemporal approach to fire severity mapping using multispectral synthetic databases and Random Forests. Remote Sensing of Environment.

8. Wehbe R, Shahrour I. A BIM-Based Smart System for Fire Evacuation. Future Internet.

9.https://www.apollo-fire.co.uk/news/challenges-and-solutionsto-managing-a-fire-detection-system-in-a-hospital/

10. Thompson MP, MacGregor DG, Dunn CJ, Calkin DE, Phipps J. Rethinking the wildland fire management system. Journal of Forest.

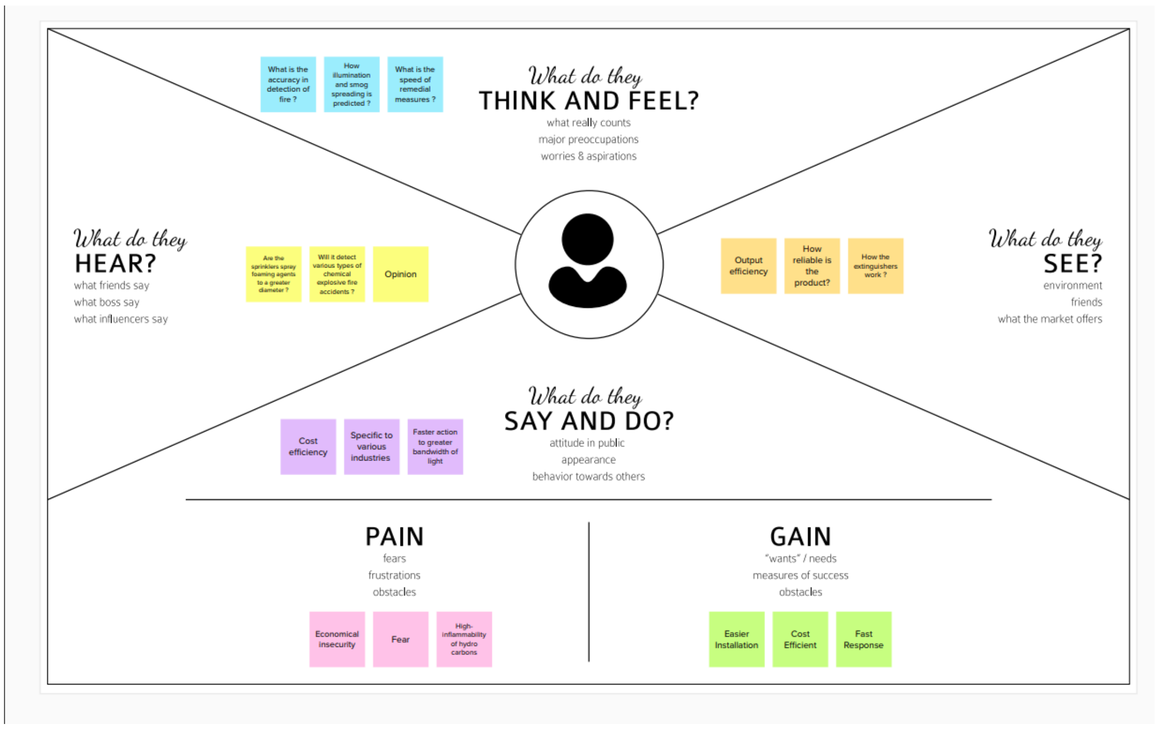
2.3. Problem Statement Definition

Many buildings still use outdated fire safety systems that cannot even activate the sprinkler system, and they all improperly communicate with one another to prevent false alarms. The fire management systems in homes and businesses are not very dependable, efficient, or cost-effective, and they lack any features like automatic alert systems for admin and authorities

3. IDEATHON AND PROPOSED SOLUTION

3.1. Empathy Map Canvas

- An empathy map is a straightforward, simple-to-understand picture that records information regarding a user's actions and attitudes.
- It is a helpful tool for helping teams understand their users better.
- Understanding the true problem and the person experiencing it is necessary for developing an effective solution.
- The process of creating a map encourages participants to think about issues from the viewpoint of the user, including goals and difficulties.


What is the accuracy in detection of fire ?
How illumination and smog spreading is predicted ?
What is the speed of remedial measures ?
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
What do they HEAR?
what friends say
what boss say
what influencers say
Are the sprinklers spray foaming agents to a greater diameter ?
Will it detect various types of chemical explosive fire accidents ?
Opinion
Output efficiency
How reliable is the product?
How the extinguishers work ?
What do they SEE?
environment
friends
what the market offers
Cost efficiency
Specific to various industries
Faster action to greater bandwidth of light
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
PAIN
fears
frustrations
obstacles
Economical insecurity
Fear
High-inflammability of hydro carbons
GAIN
"wants" / needs
measures of success
obstacles
Easier Installation
Cost Efficient
Fast Response

3.2. Ideation and Brainstroming

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

Team was gathered in mural app for collaboration

The team members are

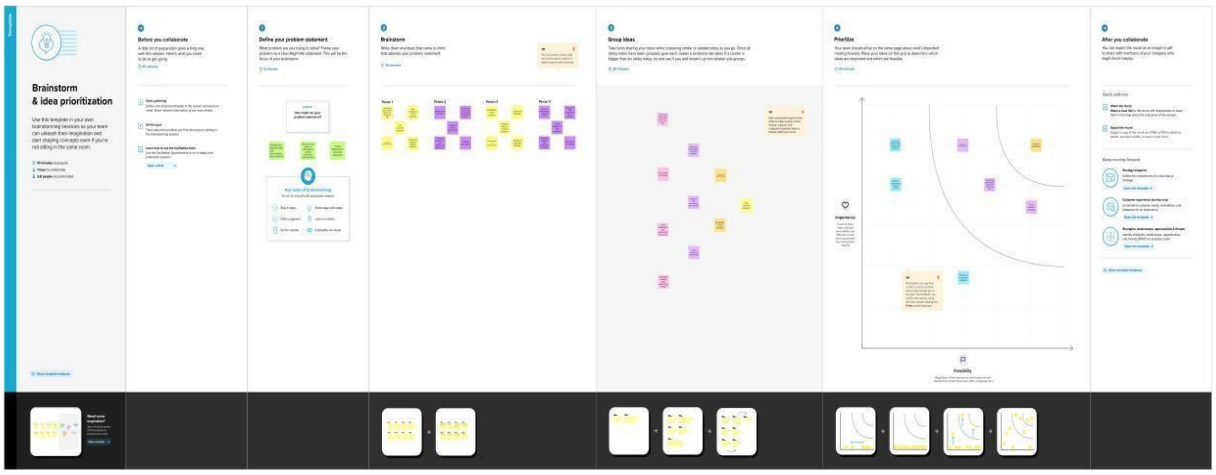
Hari haran S

Hariprasath T

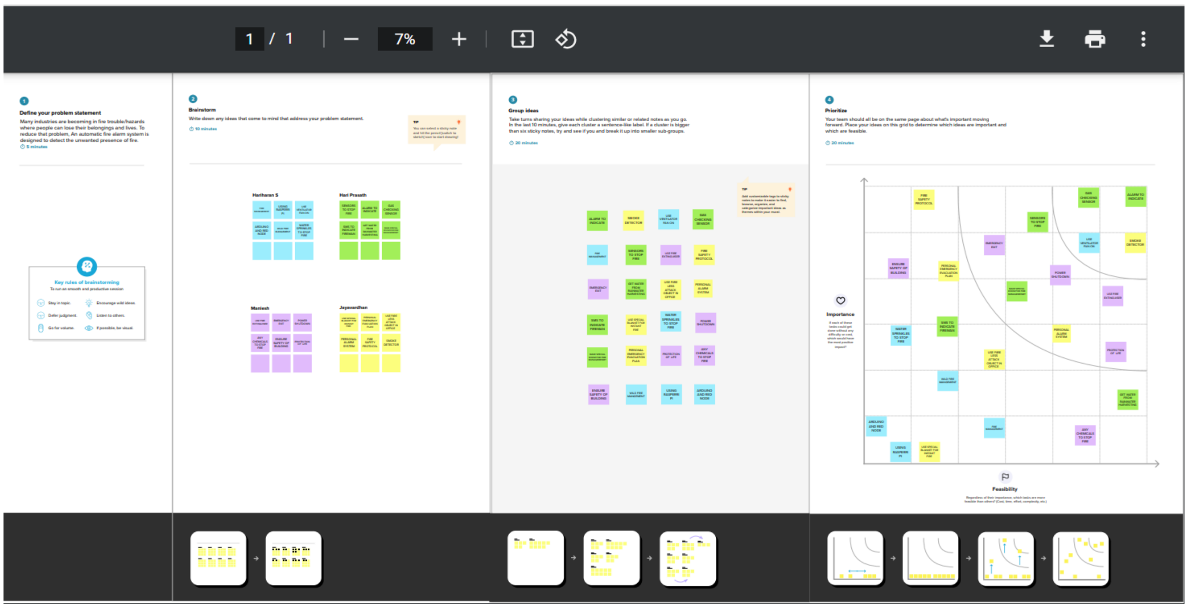
Maniesh M

Jayavardhan P

Step 2: Brainstorm, Idea Listing and Grouping



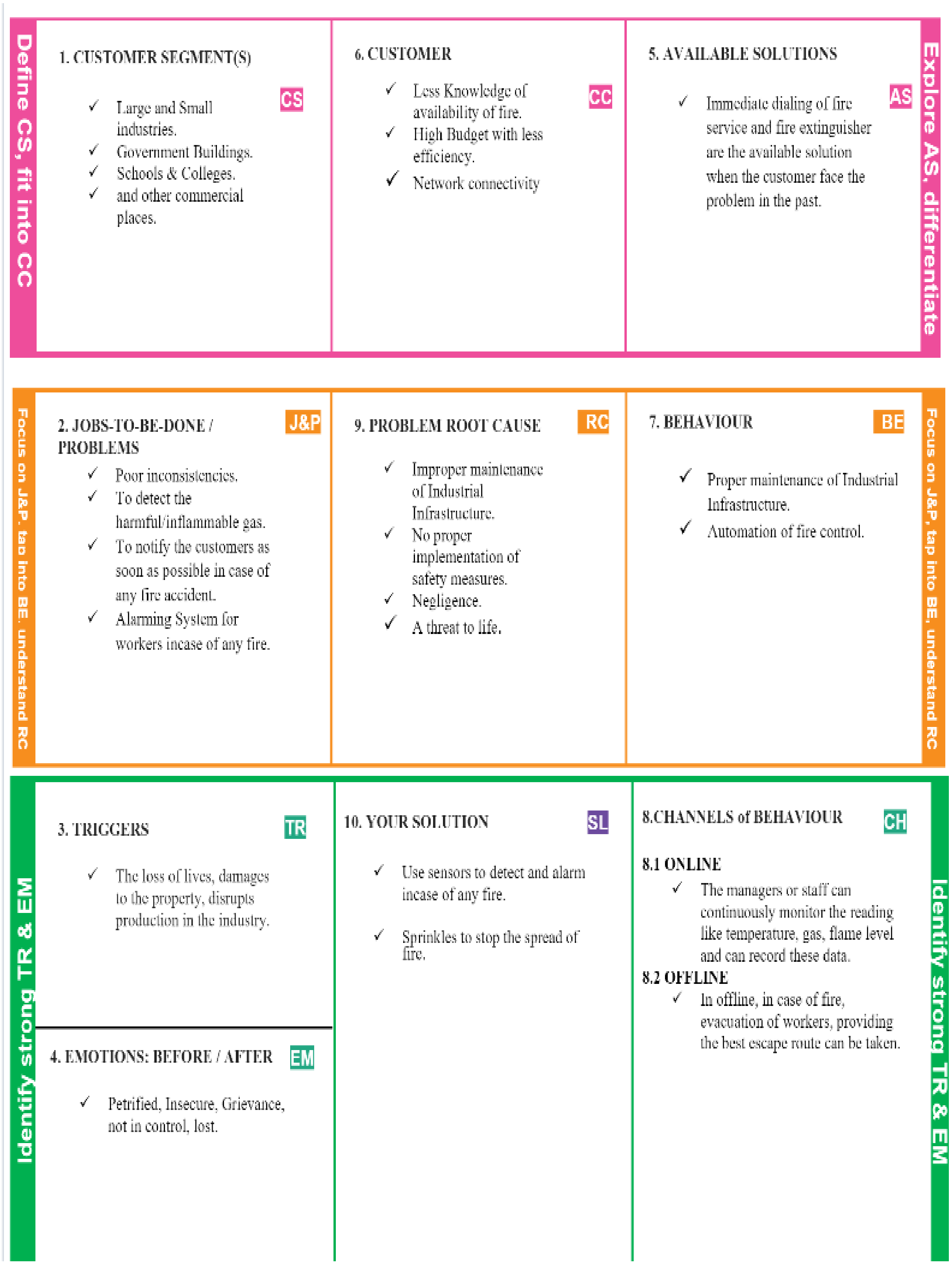
Step3: Idea Prioritization



3.3. Proposed Solution

S No	**Paramete r**	**Descriptio n**
1.	Problem Statement (Problem to be solved)	Many industries are becoming in fire trouble/hazards where people can lose their belongings and lives. To reduce that problem, an automatic fire alarm system is designed to detect the unwanted presence of fire by monitoring environmental changes associated with combustion
2.	Idea / Solution description	To create a device which periodically monitors the temperature of surroundings and updates the status straight to the server. It continuously monitors the temperature and triggers the alarm when the temperature exceed.
3.	Novelty / Uniqueness	Automatic fire alarm systems can be used to notify people to evacuate in the event of a fire or other emergency, to summon emergency services, and to prepare the structure and associated systems to control the spread of fire and smoke.
4.	Social Impact / Customer Satisfaction	The IOT detects and senses the fire using many sensors that we use and it helps the customers to access with the immediate notification and the timely access.
5.	Business Model (Revenue Model)	The product can be made compact, cost efficient and easily installable so that all the small scale to large scale industries can afford to buy the product which creates profit and increases the sale.
6.	Scalability of the Solution	This is complete system makes it easily expandable and business efficient for the fire detection, with the significant cost.

3.4. Proposed Solution Fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- ✓ Large and Small industries.
- ✓ Government Buildings.
- ✓ Schools & Colleges.
- ✓ and other commercial places.

6. CUSTOMER CC

- ✓ Less Knowledge of availability of fire.
- ✓ High Budget with less efficiency.
- ✓ Network connectivity

5. AVAILABLE SOLUTIONS AS

- ✓ Immediate dialing of fire service and fire extinguisher are the available solution when the customer face the problem in the past.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

- ✓ Poor inconsistencies.
- ✓ To detect the harmful/inflammable gas.
- ✓ To notify the customers as soon as possible in case of any fire accident.
- ✓ Alarming System for workers incase of any fire.

9. PROBLEM ROOT CAUSE RC

- ✓ Improper maintenance of Industrial Infrastructure.
- ✓ No proper implementation of safety measures.
- ✓ Negligence.
- ✓ A threat to life.

7. BEHAVIOUR BE

- ✓ Proper maintenance of Industrial Infrastructure.
- ✓ Automation of fire control.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

- ✓ The loss of lives, damages to the property, disrupts production in the industry.

4. EMOTIONS: BEFORE / AFTER EM

- ✓ Petrified, Insecure, Grievance, not in control, lost.

10. YOUR SOLUTION SL

- ✓ Use sensors to detect and alarm incase of any fire.
- ✓ Sprinkles to stop the spread of fire.

8.CHANNELS of BEHAVIOUR CH

8.1 ONLINE

- ✓ The managers or staff can continuously monitor the reading like temperature, gas, flame level and can record these data.

8.2 OFFLINE

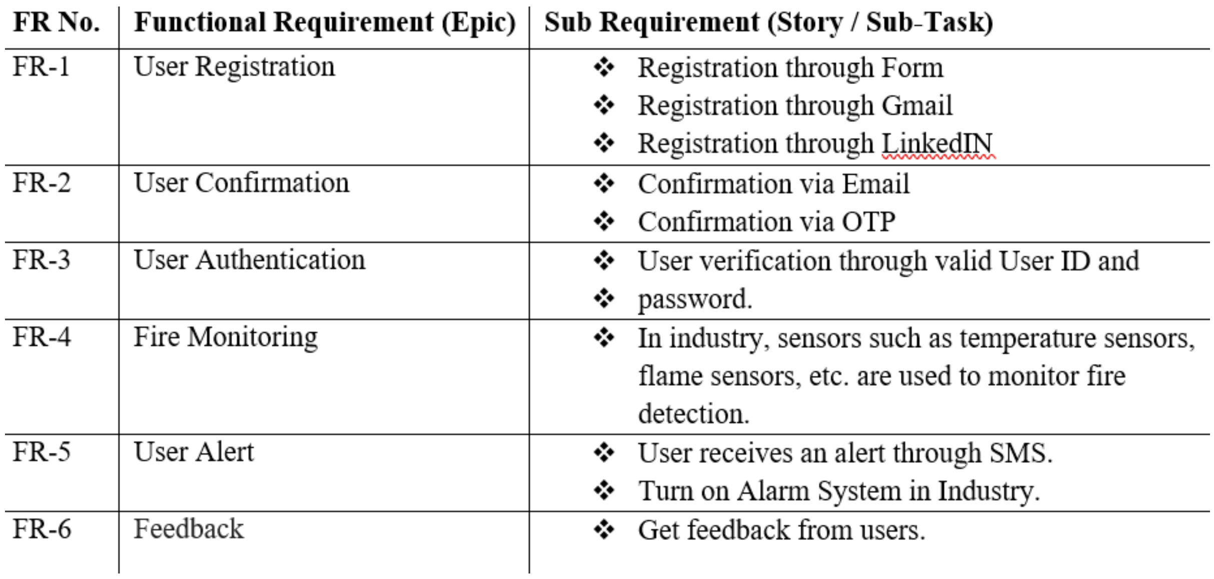
- ✓ In offline, in case of fire, evacuation of workers, providing the best escape route can be taken.

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1. 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



FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	❖ Registration through Form ❖ Registration through Gmail ❖ Registration through LinkedIN
FR-2	User Confirmation	❖ Confirmation via Email ❖ Confirmation via OTP
FR-3	User Authentication	❖ User verification through valid User ID and ❖ password.
FR-4	Fire Monitoring	❖ In industry, sensors such as temperature sensors, flame sensors, etc. are used to monitor fire detection.
FR-5	User Alert	❖ User receives an alert through SMS. ❖ Turn on Alarm System in Industry.
FR-6	Feedback	❖ Get feedback from users.

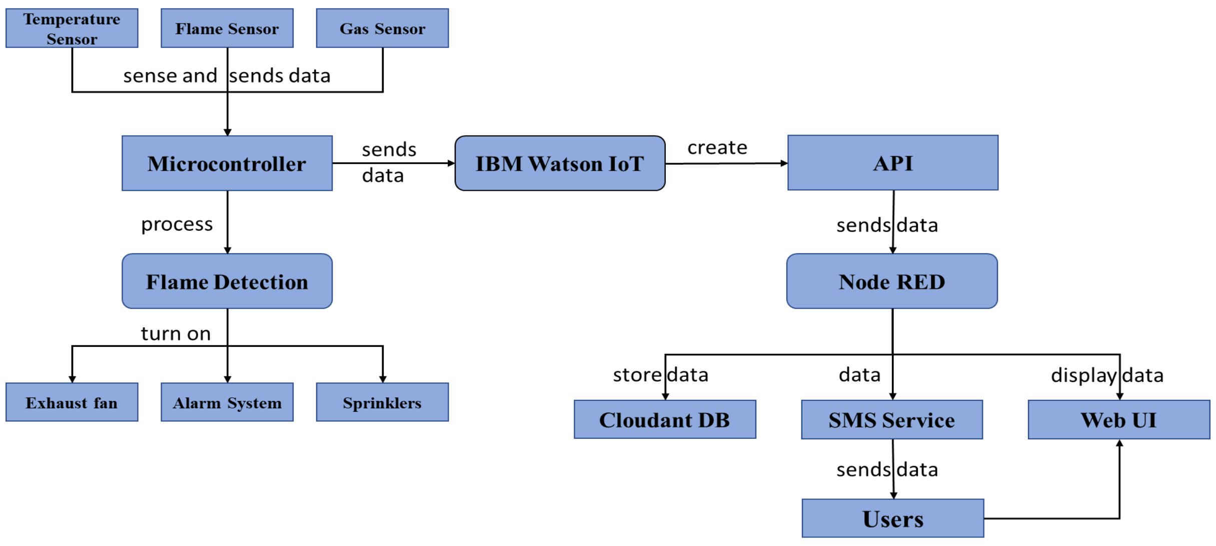
4.2. Non - Functional Requirements

- 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**	**Descriptio n**
NFR-1	Usability	● Simple, affordable, and convenient. ● Language barriers and localization requirements are among the requirements for usability. ● Efficiency of use can be used to evaluate usability.
NFR-2	Security	● Data management and transmission using secure protocols. ● Data security and encryption.
NFR-3	Reliability	It might be able to precisely identify the smoke and avoid issuing an erroneous alert or warning.
NFR-4	Performance	For customers who browse the website via a VoLTE mobile connection and Sensors preserve recordings and send them to the cloud, the front-page load time cannot exceed two seconds.
NFR-5	Availability	● Users have 24/7 access to the system. ● real-time monitoring system.
NFR-6	Scalability	The system is expandable with many sensors or with many bosses.

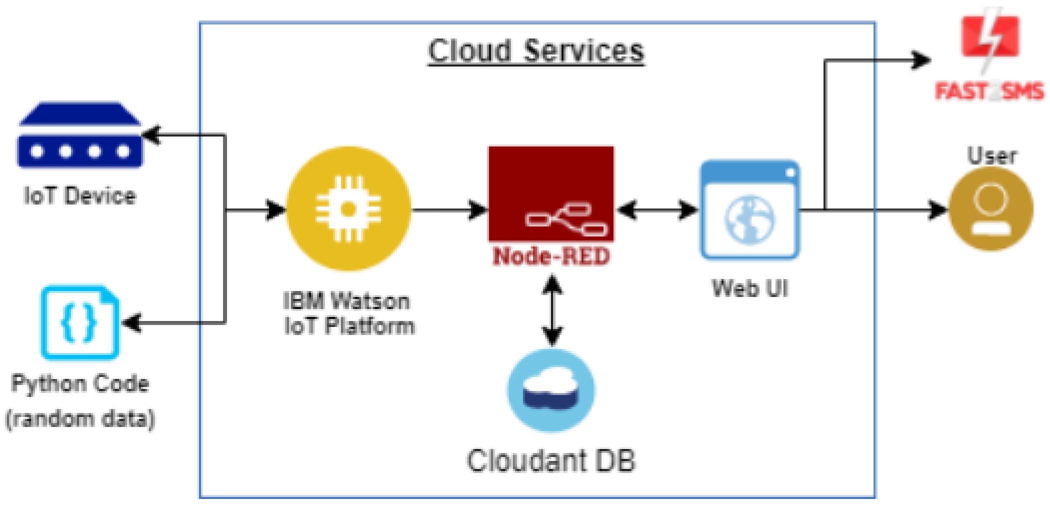
5. PROJECT DESIGN

5.1. Data flow Diagram

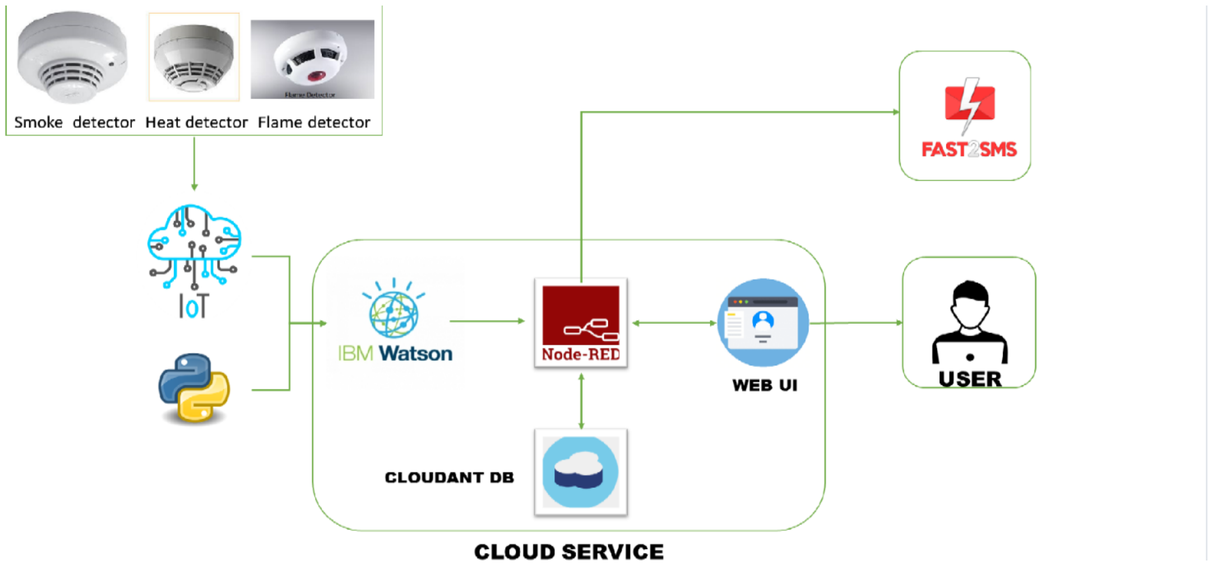


5.2. Solution and Technical

Architecture Solution Architecture



Technical Architecture



5.3. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirmingmy password.	I can access my account / dashboard	High	Sprint-1
Customer	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	Authorize	USN-3	As a user, I will enable the supervisor to monitor the Temperature status.	I can provide access to supervisor.	High	Sprint-1
Customer	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	Monitor	USN-5	As a user, I can monitor the status of the Temperature status	I can view the status of fire monitoring system	High	Sprint-1
Customer	Notification	USN-6	As a user, I can get (alarm system) alert aboutfire accident.	I can get alert about fire detection.	Medium	Sprint-2
Customer	Notification	USN-7	As a user, I can get SMS notification & alarming alert about fire accident	I can get alert about fire detection.	Medium	Sprint-2
Customer	Notification	USN-8	As a user, I can get SMS notification aboutfire accident.	I can get alert about fire detection.	Medium	Sprint-2
Customer	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	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 smart fire monitoring system	I can get service from provider	Low	Sprint-3
Administrator	Increasedservice	USN-12	As a user, I can request for scaling up the smart fire monitoring system.	I can get service from the provider.	Low	Sprint-4

6. PROJECT DESIGN AND PLANNING

6.1. Sprint Planning and Estimation

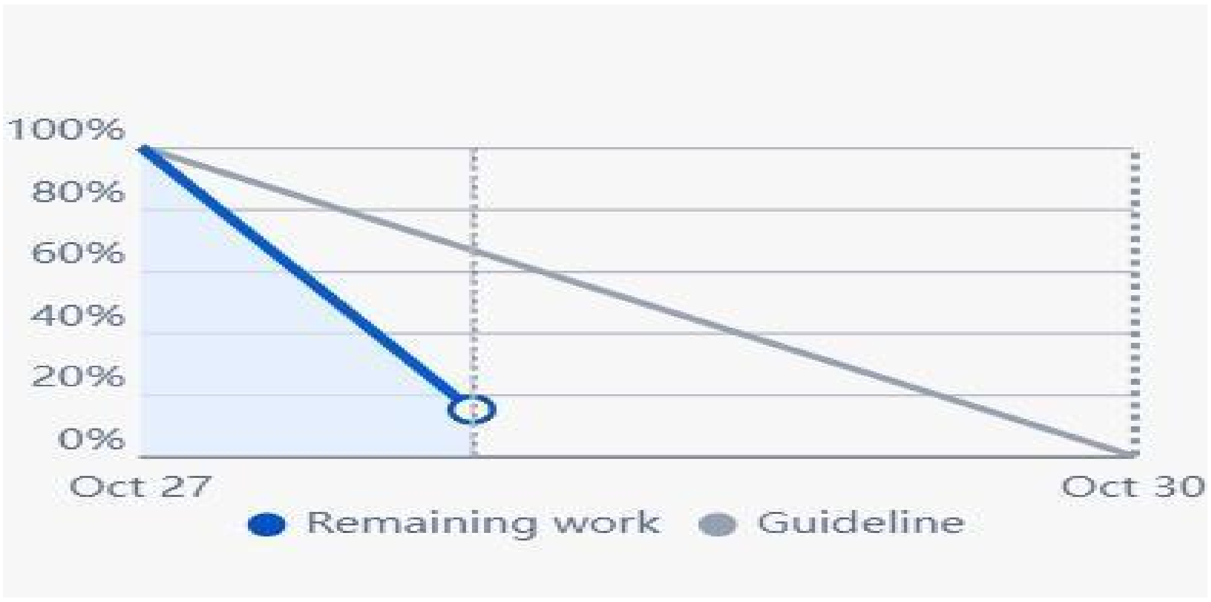
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	Low	Hari haran S
Sprint-1	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	3	Medium	Hari prasath T
Sprint-1	Registration	USN-3	As a user, I can log into the application by entering email & password.	5	Medium	Maniesh M
Sprint-1	Monitoring	USN-4	As a user, I will enable the supervisor/admin to monitor the fire management system .	10	High	Jayavardhan P
Sprint-2	Notification	USN-5	As a user, I can check the proper delivery of alerts and SMS	10	High	Hari haran S
Sprint-2	Notification	USN-6	As a user, I can get alarming alert about fire accident.	10	High	Hari prasath T
Sprint-3	Login	USN-7	As a user, I can sign up using Facebook login	2	Low	Jayavardhan P
Sprint-3	Login	USN-8	As a user, I can sign up using Google Login.	2	Low	Maniesh M
Sprint-3	Service Request	USN-9	As a user, I can request for service in case of any issue with the fire management system.	8	Medium	Hari haran S
Sprint-3	Service Request	USN-10	As a user, I can request for scaling up the fire management system.	8	Medium	Hari prasath T
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Monitoring	USN-11	As a user, I can have mobile application for mobile application for alerting system	10	High	Hari haran S
Sprint-4	Monitoring	USN-12	As a user, I can monitor the storage of data in IBM cloudant.	10	High	Hari prasath T

6.2. Sprint Delivery Schedule

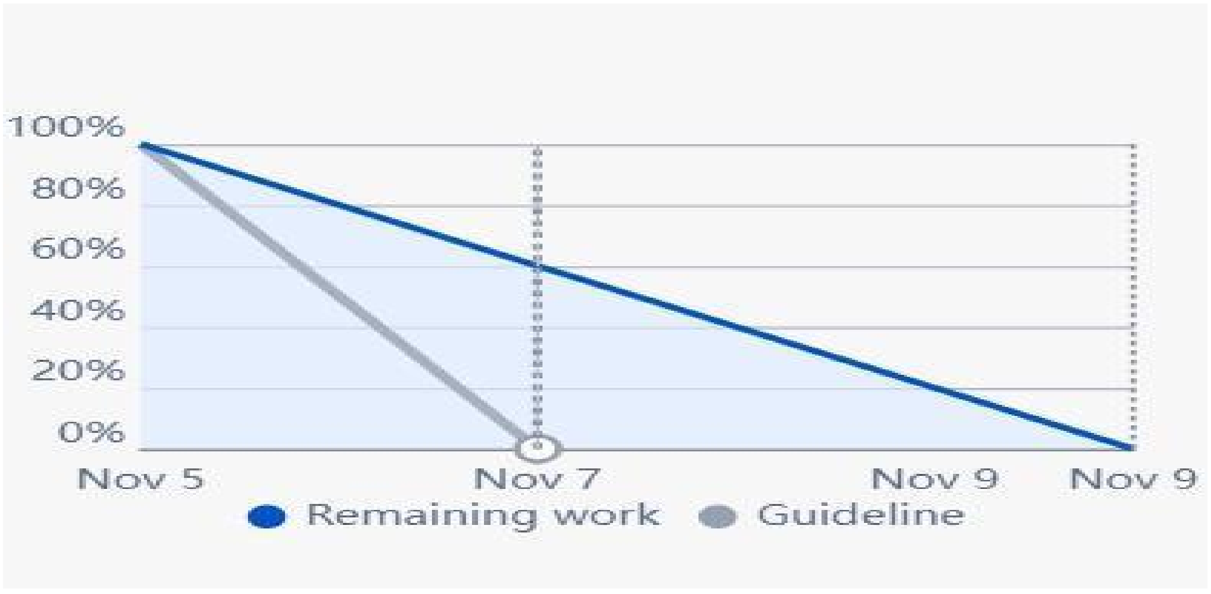
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3. Reports from JIRA

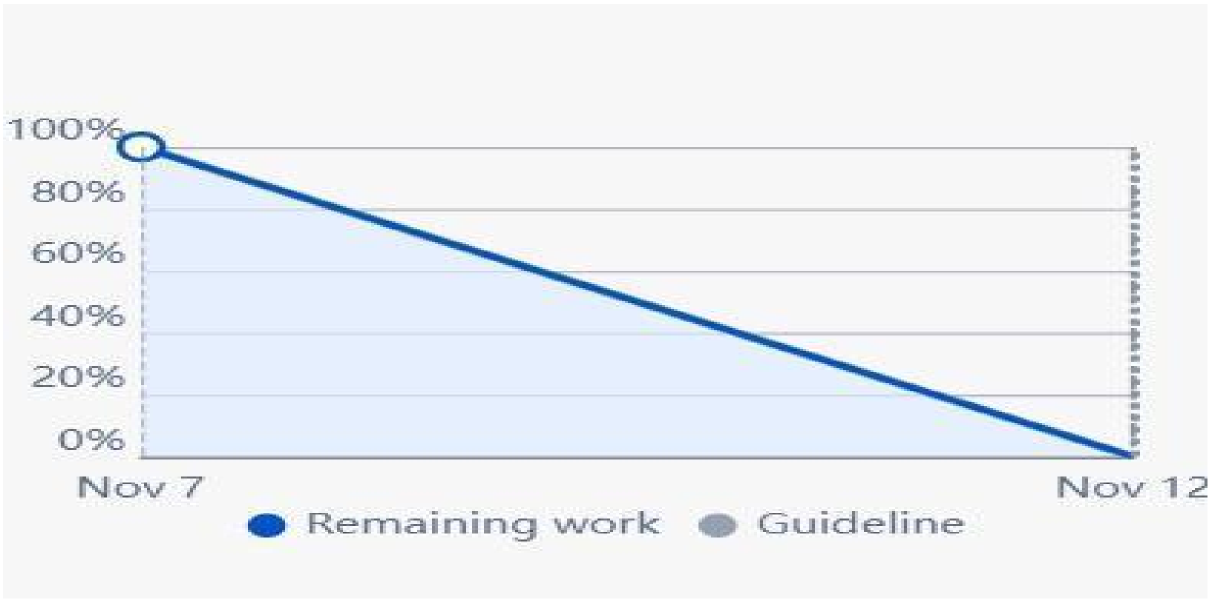
Sprint 1



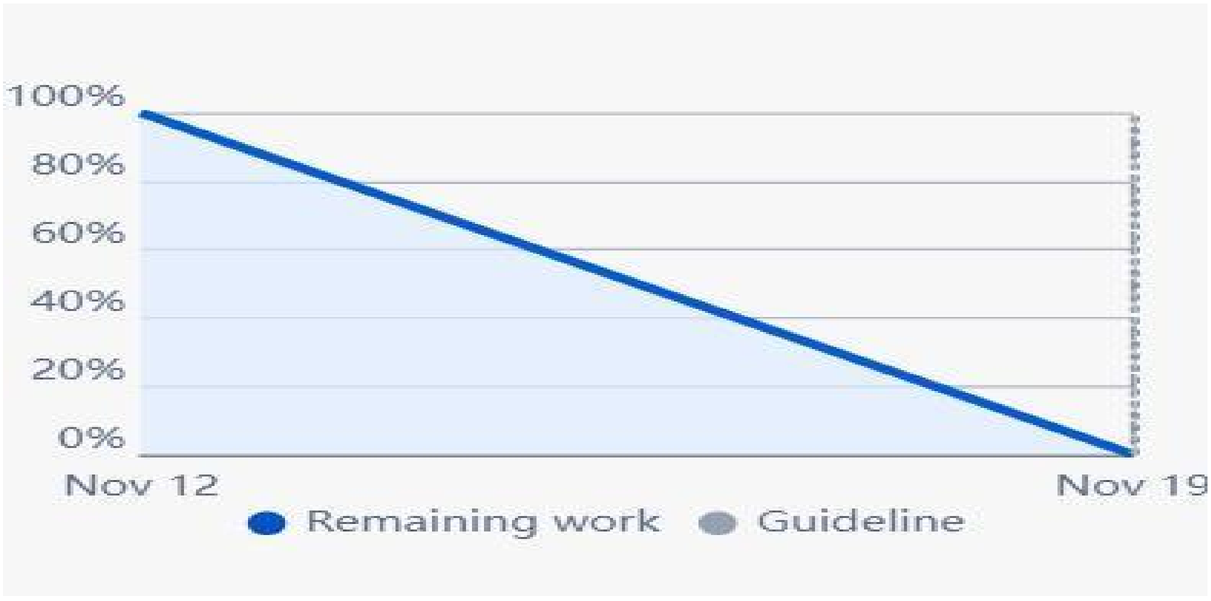
Sprint 2



Sprint 3



Sprint 4



7. CODING & SOLUTIONING

7.1. Web Application

The web application is used to provide an interface to the user create an account in the Fire System management application. The web application allows the user to create an account in the application by providing User Email, User Password and Mobile Number. And allows the user to log in to their account using their email and password. The application also provides the feature to create an account using Gmail account through Sign in With Google option and to create an account using Facebook account through Sign in With Facebook option. The mobile number entered by the user is used to send the alert message through the Twilio Messaging Service whenever the fire is detected. The logged in user can view the flame,
Temperature and Gas Level through in embedded Node-RED Dashboard.

Code Split Up:

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Nunito:ital,wght@0,400;0,600;0,700;
0,800;1,300&display=swap" rel="stylesheet">
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

App.css

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
  font-family: "Open Sans", sans-serif;
}

.image {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 20em;
}
.image img {
  width: 100%;
  height: 100%;
  padding-top: 10px;
}

.button1 {
  border: none;
  outline: none;
  width: 100%;
  padding: 15px 0;
  color: #fff;
  font-size: 16px;
  letter-spacing: 1px;
  background: #28a99e;
  cursor: pointer;
  border-radius: 8px;
}

button:hover {
    background-color: #2386c8;
    box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}

.login {
  width: 100%;
  min-height: 105vh;
  background: #e9e9e9;
  display: flex;
}

.login .loginContainer {
  margin: auto;
  width: 100%;
  max-width: 470px;
  min-height: 650px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  background-color: #fff;
  box-shadow: 0px 0px 12px 2px rgba(15, 15, 15, 0.2);
  border-radius: 12px;
  padding: 5px 30px;
```

```
}

.login .loginContainer label {
  color: rgb(0, 0, 0);
  margin: 14px 0;
  display: block;
  font-size: 20px;
  line-height: 1;
}

.login .loginContainer input {
  width: 100%;
  border: 1px solid #ccc;
  outline: none;
  border-radius: 8px;
  font-size: 16px;
  padding: 11px 20px;
  background: rgba(255, 252, 252);
  color: rgb(0, 0, 0);
}

.login .loginContainer input:hover {
  background-color: #ffffff;
  box-shadow: 0px 0px 14px 0.3px #0e81ce96;
}

.login .loginContainer .btnContainer {
  width: 100%;
  padding: 24px 0;
}
```

```
.login .loginContainer .btnContainer p {
  margin: 18px 0 0 0;
  text-align: right;
  color: rgb(0, 0, 0);
}

.login .loginContainer .btnContainer p span {
  color: rgb(15, 176, 235);
  font-weight: 500;
  margin-left: 0px;
  cursor: pointer;
  transition: all 400ms ease-in-out;
}

.login .loginContainer .btnContainer p span:hover {
  color: red;
}

.login .loginContainer .errorMsg {
  color: red;
  font-size: 16px;
}

.background {
  margin: auto;
  text-align: center;
  max-width: 360px;
  min-height: 230px;
  padding: 30px;
  box-shadow: 0 2px 2px 0 rgb(0 0 0 / 14%), 0 3px 1px -2px rgb(0 0 0 / 20%), 0 1px
5px 0 rgb(0 0 0 / 12%);
```

```
}

.button2{
  margin-top: 50px;
  border: none;
  outline: none;
  width: 50%;
  padding: 15px 0;
  color: #fff;
  font-size: 16px;
  letter-spacing: 1px;
  background: #28a99e;
  cursor: pointer;
  border-radius: 8px;
}

.button3 {
  position:fixed;
  position:relative;
  left:40%;
  right:20%;
  bottom:5%;
  top:60%;
  margin-top:600px;
  border: none;
  outline: none;
  width: 20%;
  padding: 15px 0;
  color: #fff;
  font-size: 16px;
  letter-spacing: 1px;
  background: #28a99e;
  cursor: pointer;
  border-radius: 8px;
}

.centerline {
  margin: auto;
  border-top: 1px solid black;
  width: 15%;
  padding: 8px;
}

.buttonContainer {
  width: 100%;
  padding: 10px 0;
}

.icon {
  width: 20px;
  /* margin-left: 10px; */
  margin-right: 5px;

}

.loginbtn{
  border: 2px solid #28a99e;
  outline: none;
  width: 100%;
  padding: 15px 0;
  color: rgb(0, 0, 0);
  font-size: 16px;
  letter-spacing: 1px;
  background: #ffffff;
  cursor: pointer;
  border-radius: 8px;
  display: flex;
  align-items: center;
  justify-content: center;
}

.loginbtn:hover{
  background-color: #28a99e;
  box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
```

App.js

```
import React,{ useState, useEffect } from 'react';
import {firebase} from './firebase';
import Login from './Login';
import Dashboard from './Dashboard';
import './App.css';
import EmailVerify from './EmailVerify';

const App = () => {
  const [user, setUser] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [emailError, setEmailError] = useState('');
  const [passwordError, setPasswordError] = useState('');
  const [hasAccount, setHasAccount] = useState(false);
const clearInputs = () =>{
  setEmail('');
  setPassword('');
}

const clearErrors = () => {
  setEmailError('');
  setPasswordError('');
}

const handleLogin = () => {
  clearErrors();
  firebase
    .auth()
    .signInWithEmailAndPassword(email, password)
    .catch(err => {
      switch(err.code){
        case "auth/invalid-email":
        case "auth/user-disabled":
        case "auth/user-not-found":
          setEmailError(err.message);
          break;
        case "auth/wrong-password":
          setPasswordError(err.message);
          break;
      }
    });
};

const handleSignup = () => {
  clearErrors();
  firebase
    .auth()
    .createUserWithEmailAndPassword(email, password).then(() => {
      firebase.auth().currentUser.sendEmailVerification()
    .then(() => {
      console.log('Email sent');
    });
  }
  )
    .catch(err => {
      switch(err.code){
        case "auth/email-already-in-use":
        case "auth/invalid-email":
          setEmailError(err.message);
          break;
        case "auth/weak-password":
          setPasswordError(err.message);
          break;
      }
    });
};


const handleLogout = () => {
  firebase.auth().signOut();
};

const authListener = () => {
  firebase.auth().onAuthStateChanged(user => {
    if(user){
      clearInputs();
      setUser(user);

    }else{
      setUser("");
    }
  })
}

useEffect(() => {
  authListener();
}, []);

return (
  <div className='App'>
    { user ? user.emailVerified ?  (
      <Dashboard handleLogout = { handleLogout }/>
     ):<EmailVerify handleLogout = {handleLogout}/>:(
      <Login
        email = { email }
        setEmail = { setEmail }
        password = { password }
        setPassword = { setPassword }
        handleLogin = { handleLogin }
        handleSignup = { handleSignup }
        hasAccount = { hasAccount }
        setHasAccount = { setHasAccount }
        emailError = { emailError }
        passwordError = { passwordError }
      />
     )}
  </div>
  );
};

export default App;
```

Dashboard.js

```
import React from "react";

const Dashboard = (props) => {
   const {
      handleLogout
   } = props;
   return (
      <div>
         <iframe
            title="Node-RED Dashboard"
            style={{
               width: "100%",
               height: "100%",
               overflowY: "hidden",
               top: 0,
               left: 0,
               position: "absolute",
            }}
            src="https://node-red-ejfms-2022-11-09.au-syd.mybluemix.net/ui/"
         ></iframe>
         <div>
            <button className="button3" onClick={handleLogout}>Logout</button>
         </div>
      </div>
   );
};

export default Dashboard;
```

EmailVeriy.js

```
import React from "react";
import './App.css';



const EmailVerify = ({handleLogout}) => {
   return (
       <div className="background">
           <h2>Please verify your Email.</h2>
           <button className="button2" onClick={handleLogout}>Back to
Login</button>
       </div>
   );
};
export default EmailVerify;
```

firebase.js

```
// Import the functions you need from the SDKs you need
import firebase from 'firebase/compat/app';
import 'firebase/compat/auth';
import 'firebase/compat/firestore';
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries
// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyD2EuKxrWivpRsXxti5GHNmyojbWpznnQE",
  authDomain: "fire-management-sys-19220.firebaseapp.com",
  projectId: "fire-management-sys-19220",
  storageBucket: "fire-management-sys-19220.appspot.com",
  messagingSenderId: "42010274392",
  appId: "1:42010274392:web:f982bb0a10744e0756782d",
  measurementId: "G-3KZC1Y5M62"
};


// Initialize Firebase
const firebaseApp = firebase.initializeApp(firebaseConfig);
const auth = firebase.auth();
const googleProvider = new firebase.auth.GoogleAuthProvider();
const fbProvider = new firebase.auth.FacebookAuthProvider();

export {firebase, firebaseApp, auth, googleProvider, fbProvider};
```

index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
```

```
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import EmailVerify from './EmailVerify';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App/>
  </React.StrictMode>
);
```

Login.js

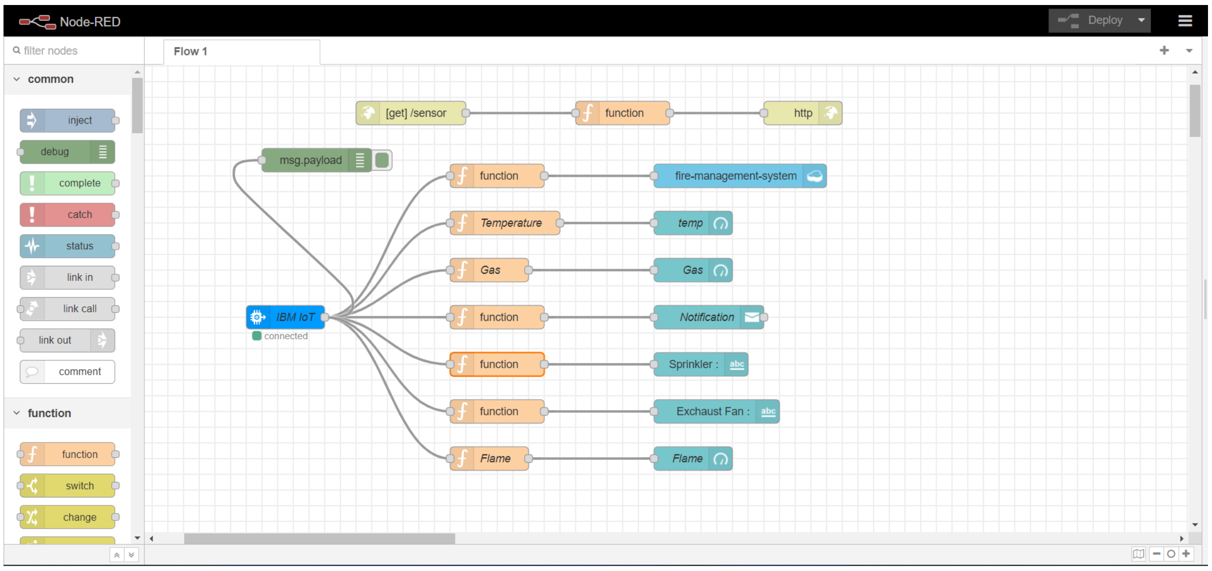
```
import React from "react";
import loginImg from '../src/image/login.jpg'
import { auth, fbProvider, googleProvider } from "./firebase";

const Login = (props) => {

    const {
        email,
        setEmail,
        password,
        setPassword,
        handleLogin,
        handleSignup,
        hasAccount,
        setHasAccount,
        emailError,
        passwordError
    } = props;

    const googleLogin = async ()=> {
        try {
            await auth.signInWithPopup(googleProvider);
            setEmail(await auth.currentUser);
        } catch (error) {
            console.log(error);
        }
    };

    const facebookLogin = async ()=> {
        try {
            await auth.signInWithPopup(fbProvider);
            setEmail(await auth.currentUser);
        } catch (error) {
            console.log(error);
        }
    };

    console.log(email);

    return (
        <section className="login">
          <div className="loginContainer">
            <div className="image">
              <img src={loginImg} alt=""/>
            </div>
            <div className="form">
              <label>Email</label>
              <input
                type="text"
                autoFocus
                required
                value={email}
                placeholder = "Email"
                onChange={(e) => setEmail(e.target.value)}
              />
              <p className="errorMsg">{emailError}</p>
              <label>Password</label>
              <input
                type="password"
                required
                value={password}
                placeholder = "Password"
                onChange={(e) => setPassword(e.target.value)}
              />
              <p className="errorMsg">{passwordError}</p>
            </div>
            <div className="btnContainer">
              {hasAccount ? (
                <>
                <button className="button1" onClick={handleSignup}>Sign
    up</button>
```

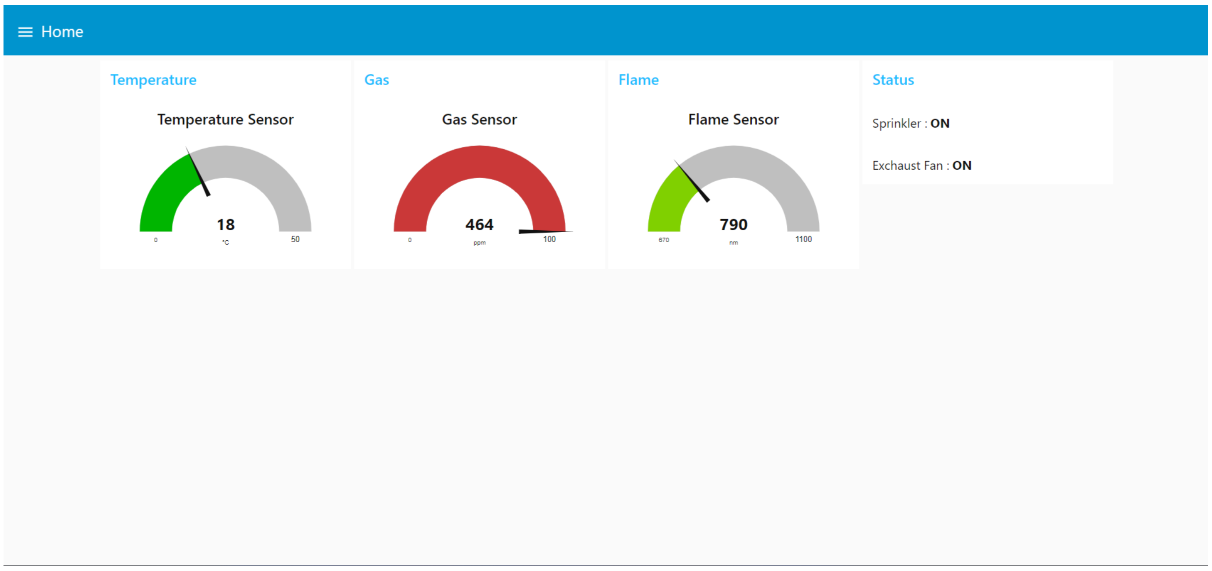
```
                <p>Already have an account? <span onClick={() =>
setHasAccount(!hasAccount)}>Sign in</span></p>
                </>
            ):(
                <>
                <button className="button1" onClick={handleLogin}>Log
in</button>
                <p>Don't have an account? <span onClick={() =>
setHasAccount(!hasAccount)}>Sign up</span></p>
                </>
            )}
        </div>
        <div className="centerline"></div>
        <div className="buttonContainer" onClick={googleLogin}>

            <button className="loginbtn">
                <img className="icon"
src="https://upload.wikimedia.org/wikipedia/commons/thumb/5/53/Google_%22G%2
2_Logo.svg/1200px-Google_%22G%22_Logo.svg.png"
                alt=""/>
                Login with Google
            </button>
        </div>
        <div className="buttonContainer" onClick={facebookLogin}>
            <button className="loginbtn">
                <img className="icon"
src="https://toppng.com/uploads/preview/facebook-logo-png-transparent-facebook-f-l
ogo-sv-11563088711q5rgq6hd0v.png" alt=""/>
                Login with Facebook
            </button>
        </div>
```

```
          </div>

        </section>
    );
};
export default Login;
```

7.2. Node-red web Dashboard

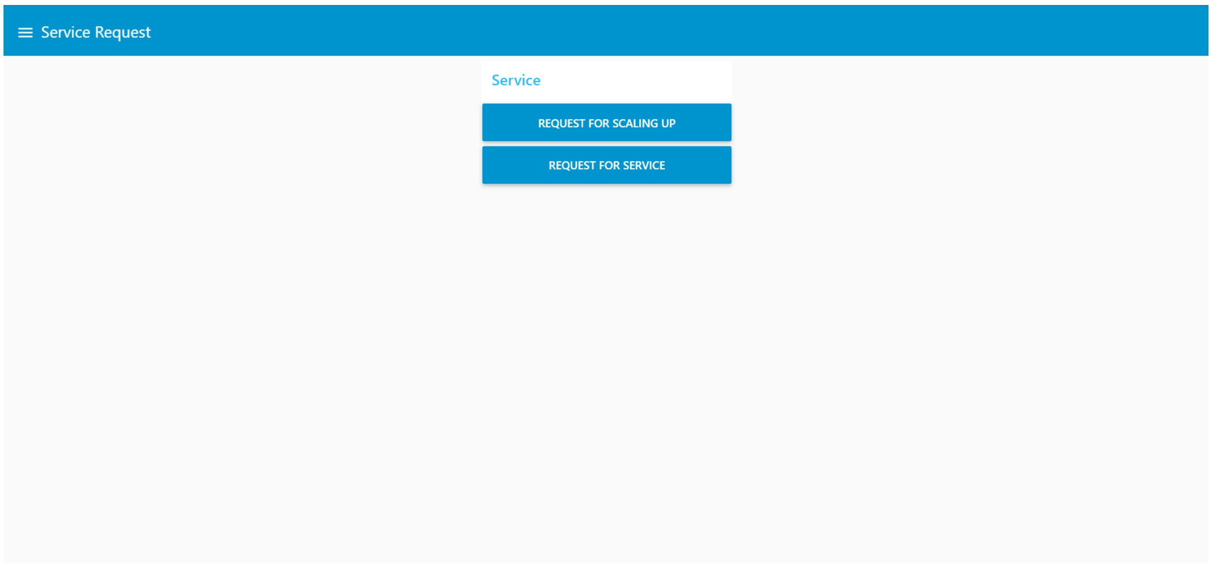
The Node-RED Dashboard is configured in the Node-RED





Sprinkler : **ON**

Exhaust Fan : **ON**



7.3. Wokwi Simulation:

//PNT2022TMID19220

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2
#define BUZZER 14
#define EXFAN1 13
#define EXFAN2 4
#define SPRKLR 12

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C LCD = LiquidCrystal_I2C(0x27, 20, 4);

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 "qm6k13"//IBM ORGANITION ID
#define DEVICE_TYPE "iot_device"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "123456789"     //Token
String data3;
float h, t, s,f,fan1=0,fan2=0,spr=0;



//-------- 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();
 pinMode(LED,OUTPUT);
 pinMode(BUZZER,OUTPUT);
 pinMode(EXFAN1,OUTPUT);
 pinMode(EXFAN2,OUTPUT);
 digitalWrite(LED,LOW);
 digitalWrite(BUZZER,LOW);
 digitalWrite(EXFAN1,LOW);
 digitalWrite(EXFAN2,LOW);
 pinMode(SPRKLR,OUTPUT);
 digitalWrite(SPRKLR,LOW);
 delay(10);
 Serial.println();
 wificonnect();
 mqttconnect();
 LCD.init();
 LCD.backlight();
 LCD.setCursor(0, 0);
 LCD.print("Connecting to ");
 LCD.setCursor(0, 1);
 LCD.print("WiFi ");
 delay(1000);
 LCD.clear();
}

void loop()// Recursive Function
{
  LCD.setCursor(0,2);
  LCD.print("Smoke: ");
  LCD.setCursor(0, 0);
  LCD.print("Temp: ");
  LCD.setCursor(14, 0);
  LCD.print("C");
  LCD.setCursor(0, 1);
  LCD.print("Humid: ");
  LCD.setCursor(14, 1);
  LCD.print("%");
h = dht.readHumidity();
t = dht.readTemperature();
s = random(0,2000);
f = random(0,1100);
if (s>400)
{
  Serial.print("Smoke: ");
  Serial.println("Detected");
  digitalWrite(BUZZER,HIGH);
   LCD.setCursor(7, 2);
  LCD.print("YES");
  LCD.setCursor(0, 3);
  LCD.print("WARNING!FIREACCIDENT");
  digitalWrite(EXFAN1,HIGH);
  digitalWrite(EXFAN2,HIGH);
   fan1=1;
    fan2=1;
}
else{
    Serial.print("Smoke: ");
    Serial.println("Not Detected");
    digitalWrite(BUZZER,LOW);
    LCD.setCursor(7, 2);
   LCD.print(" NO");
   LCD.setCursor(0, 3);
  LCD.print("                    ");
   digitalWrite(EXFAN1,LOW);
   digitalWrite(EXFAN2,LOW);
    fan1=0;
     fan2=0;
  }
  if(f>300)
  {
     Serial.print("Flame: ");
     Serial.println("Detected");
     spr=1;
     digitalWrite(SPRKLR,HIGH);
  }
  else
  {
     Serial.print("Flame: ");
     Serial.println("NOT Detected");
     spr=0;
     digitalWrite(SPRKLR,LOW);
  }
  Serial.print("temp:");
  Serial.println(t);
  LCD.setCursor(7, 0);
  LCD.print(t);
  Serial.print("Humid:");
  Serial.println(h);
  LCD.setCursor(7, 1);
  LCD.print(h);

  PublishData(t, h, s,fan1,spr);
  delay(1000);
  if (!client.loop()) {
   mqttconnect();
  }
}
```

```
/*....................................retrieving to Cloud..............................*/

void PublishData(float temp, float humid, float smoke,float fan1,float spr) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"Temperature\":";
  payload += temp;
  payload += "," "\"Flame\":";
  payload += humid;
  payload += "," "\"Gas\":";
  payload += smoke;
   payload += "," "\"Exhaust_Fans_Status\":";
  payload += fan1;
     payload += "," "\"Sprinkler_Status\":";
  payload += spr;
  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");
  LCD.setCursor(0, 0);
  LCD.print("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];
  }
  Serial.println("data: "+ data3);
  if(data3=="lighton")
  {
Serial.println(data3);
digitalWrite(LED,HIGH);
  }
  else
  {
Serial.println(data3);
digitalWrite(LED,LOW);
  }
data3="";
```

```
}
```

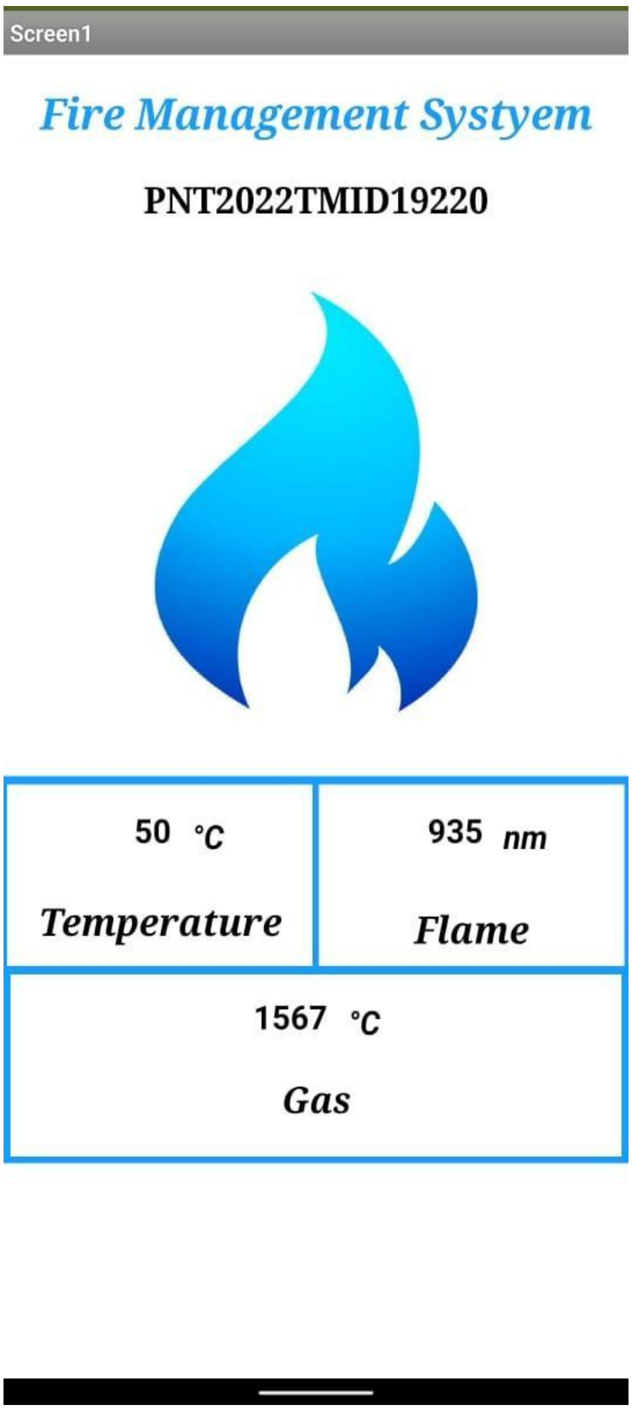
7.4. Python Simulation:

Code:

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

#Provide your IBM Watson Device Credentials
organization = "qm6k13" # repalce it with organization ID
deviceType = "iot_device" #replace it with device type
deviceId = "1234" #repalce with device id
authMethod = "token"
authToken = "123456789"#repalce with token

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()
```

```
while True:
        # T=randint(0, 50);
        # G=randint(0, 2000);
        # F=randint(0, 1100);
        T=33;
        G=90;
        F=90;
        #Send Temperature, Gas & Flame Readings to IBM Watson
        sprinkler = 0 if(F>=300 or (F>=300 and G>=500)) else 1
        exchaustFan = 0 if(G>=500 or (F>=300 and G>=500)) else 1
        data = { 'Temperature' : T, 'Gas': G , 'Flame': F ,'Sprinkler_Status':
sprinkler,'Exchaust_Fans_Status' : exchaustFan}
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % T, "Flame = %s nm" % F,
"Gas = %s ppm" % G, "Sprinklers_Status = %s " % sprinkler, "Exchaust
_Fan_Status = %s " % exchaustFan, "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)


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

7.5. Mobile Application



Screen1

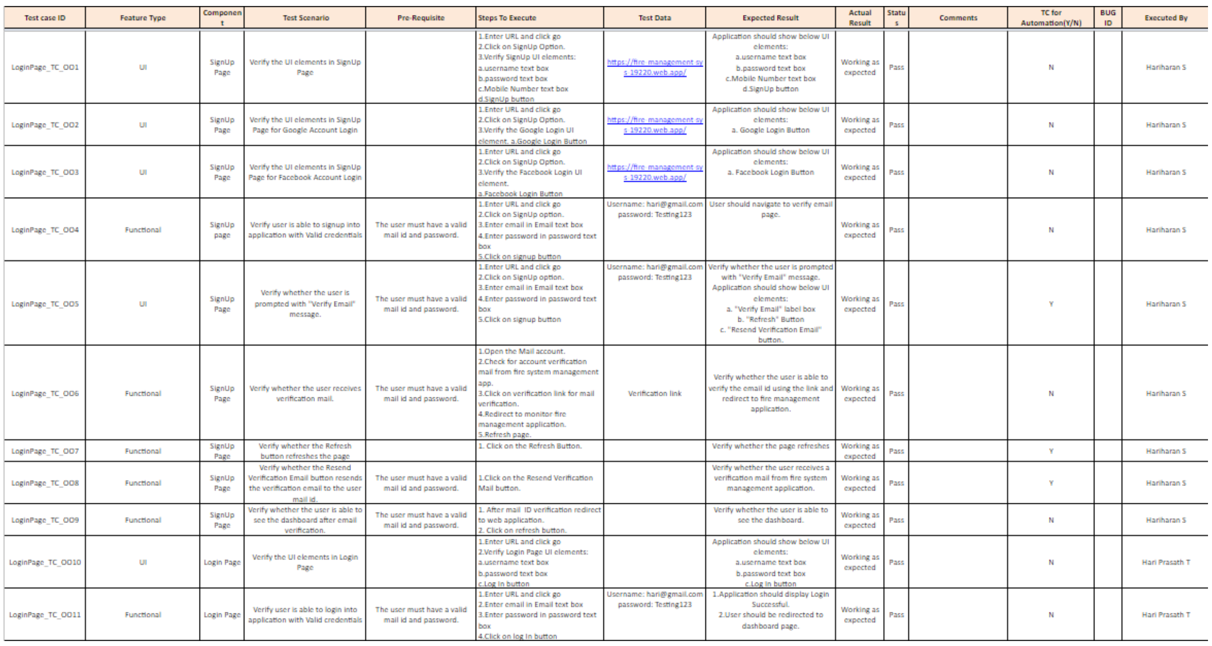
Fire Management Systyem

PNT2022TMID19220

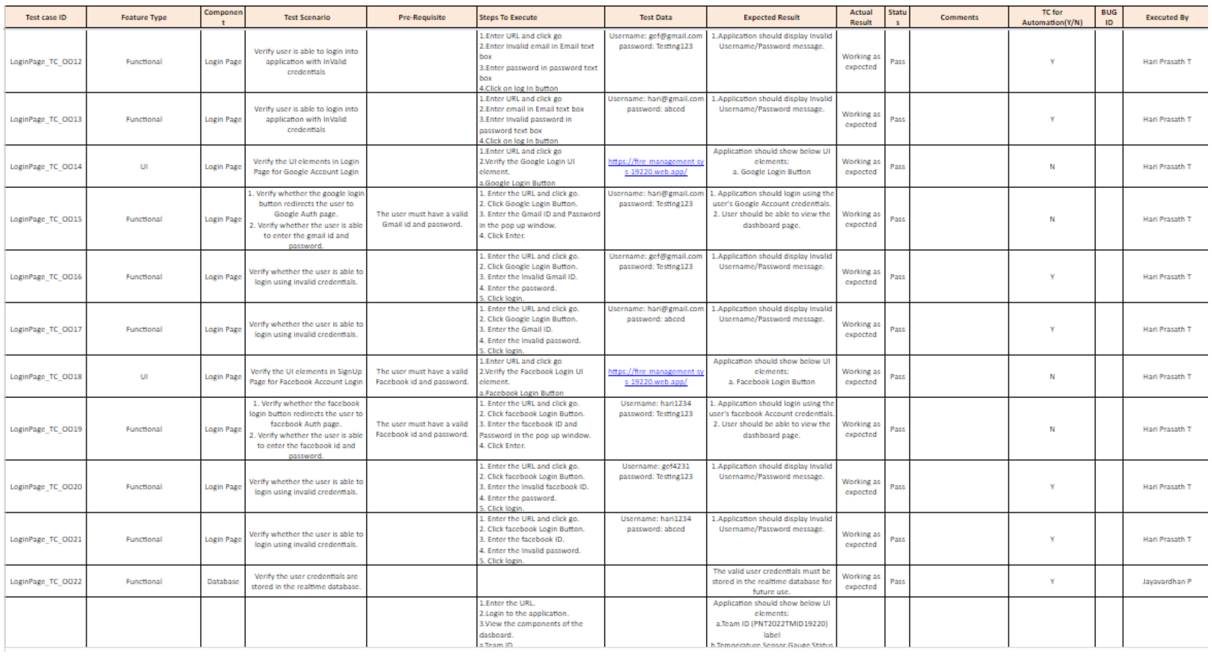
50 °C	935 nm
Temperature	Flame
1567 °C	
Gas	

8. TESTING

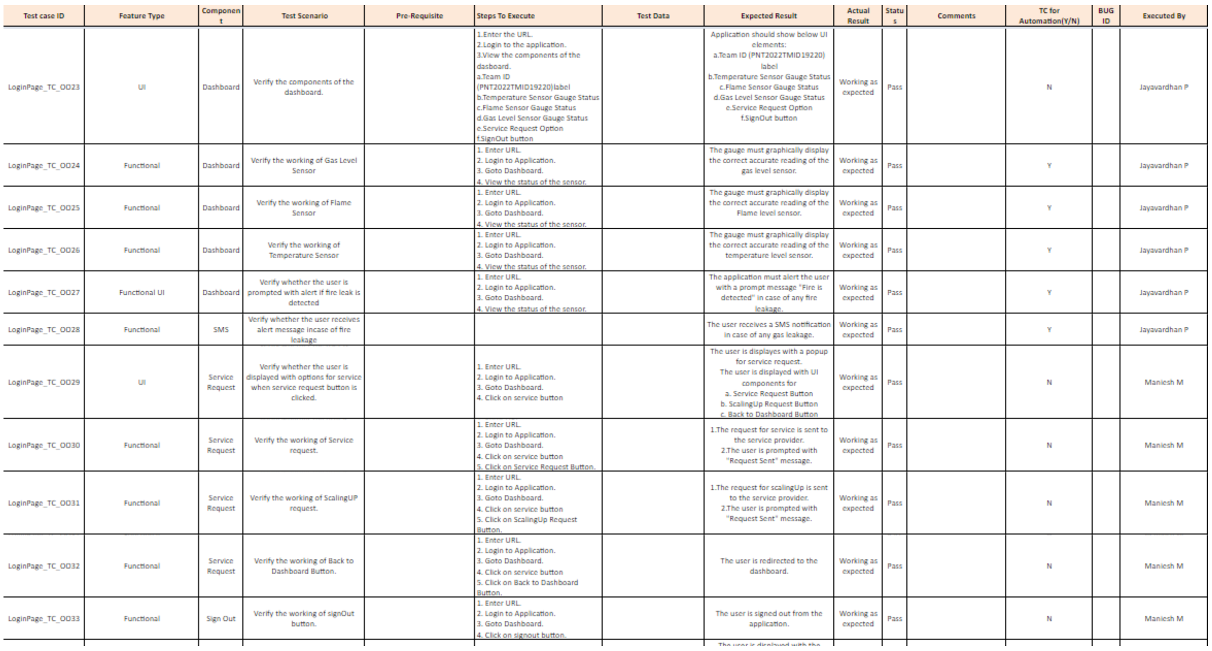
8.1. Test Cases



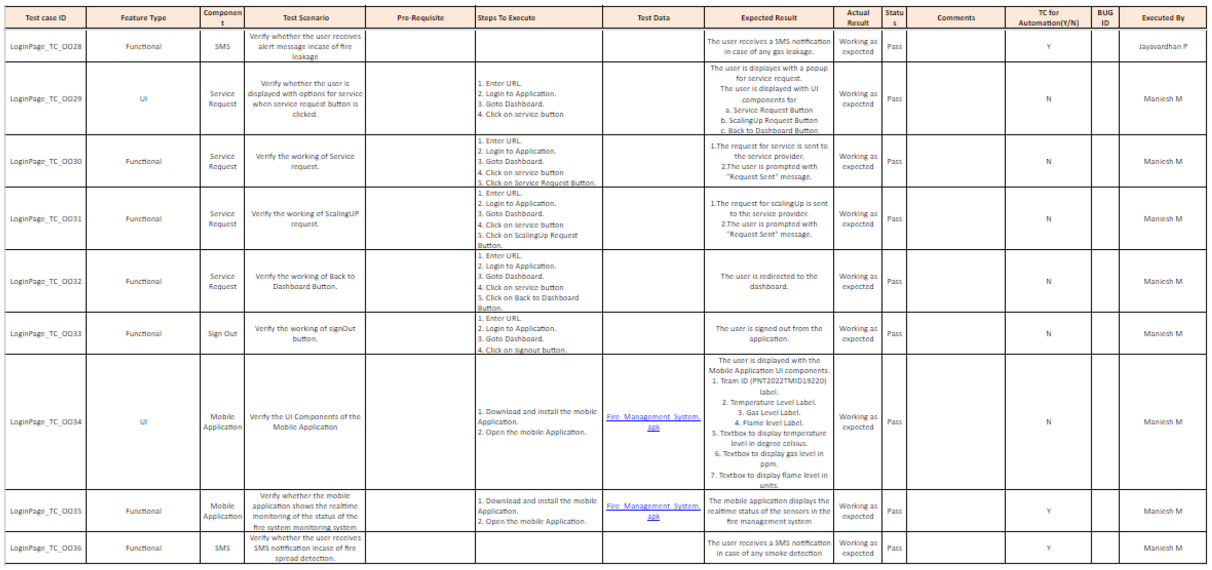
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	UI	SignUp Page	Verify the UI elements in SignUp Page		1.Enter URL and click go 2.Click on SignUp Option. 3.Verify SignUp UI elements: a.username text box b.password text box c.Mobile Number text box d.SignUp button	https://fire-management-sys-19220.web.app/	Application should show below UI elements: a.username text box b.password text box c.Mobile Number text box d.SignUp button	Working as expected	Pass		N		Hariharan S
LoginPage_TC_OO2	UI	SignUp Page	Verify the UI elements in SignUp Page for Google Account Login		1.Enter URL and click go 2.Click on SignUp Option. 3.Verify the Google Login UI element. a.Google Login Button	https://fire-management-sys-19220.web.app/	Application should show below UI elements: a. Google Login Button	Working as expected	Pass		N		Hariharan S
LoginPage_TC_OO3	UI	SignUp Page	Verify the UI elements in SignUp Page for Facebook Account Login		1.Enter URL and click go 2.Click on SignUp Option. 3.Verify the Facebook Login UI element. a.Facebook Login Button	https://fire-management-sys-19220.web.app/	Application should show below UI elements: a. Facebook Login Button	Working as expected	Pass		N		Hariharan S
LoginPage_TC_OO4	Functional	SignUp page	Verify user is able to signup into application with Valid credentials	The user must have a valid mail id and password.	1.Enter URL and click go 2.Click on SignUp option. 3.Enter email in Email text box 4.Enter password in password text box 5.Click on signup button	Username: hari@gmail.com password: Testing123	User should navigate to verify email page.	Working as expected	Pass		N		Hariharan S
LoginPage_TC_OO5	UI	SignUp Page	Verify whether the user is prompted with "Verify Email" message.	The user must have a valid mail id and password.	1.Enter URL and click go 2.Click on SignUp option. 3.Enter email in Email text box 4.Enter password in password text box 5.Click on signup button	Username: hari@gmail.com password: Testing123	Verify whether the user is prompted with "Verify Email" message. Application should show below UI elements: a. "Verify Email" label box b. "Refresh" Button c. "Resend Verification Email" button.	Working as expected	Pass		Y		Hariharan S
LoginPage_TC_OO6	Functional	SignUp Page	Verify whether the user receives verification mail.	The user must have a valid mail id and password.	1.Open the Mail account. 2.Check for account verification mail from fire system management app. 3.Click on verification link for mail verification. 4.Redirect to monitor fire management application. 5.Refresh page.	Verification link	Verify whether the user is able to verify the email id using the link and redirect to fire management application.	Working as expected	Pass		N		Hariharan S
LoginPage_TC_OO7	Functional	SignUp Page	Verify whether the Refresh button refreshes the page		1. Click on the Refresh Button.		Verify whether the page refreshes	Working as expected	Pass		Y		Hariharan S
LoginPage_TC_OO8	Functional	SignUp Page	Verify whether the Resend Verification Email button resends the verification email to the user mail id.	The user must have a valid mail id and password.	1.Click on the Resend Verification Mail button.		Verify whether the user receives a verification mail from fire system management application.	Working as expected	Pass		Y		Hariharan S
LoginPage_TC_OO9	Functional	SignUp Page	Verify whether the user is able to see the dashboard after email verification.	The user must have a valid mail id and password.	1. After mail ID verification redirect to web application. 2. Click on refresh button.		Verify whether the user is able to see the dashboard.	Working as expected	Pass		N		Hariharan S
LoginPage_TC_OO10	UI	Login Page	Verify the UI elements in Login Page		1.Enter URL and click go 2.Verify Login Page UI elements: a.username text box b.password text box c.Log In button		Application should show below UI elements: a.username text box b.password text box c.Log In button	Working as expected	Pass		N		Hari Prasath T
LoginPage_TC_OO11	Functional	Login Page	Verify user is able to login into application with Valid credentials	The user must have a valid mail id and password.	1.Enter URL and click go 2.Enter email in Email text box 3.Enter password in password text box 4.Click on log In button	Username: hari@gmail.com password: Testing123	1.Application should display Login Successful. 2.User should be redirected to dashboard page.	Working as expected	Pass		N		Hari Prasath T



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO12	Functional	Login Page	Verify user is able to login into application with InValid credentials		1.Enter URL and click go 2.Enter invalid email in Email text box 3.Enter password in password text box 4.Click on log In button	Username: gef@gmail.com password: Testing123	1.Application should display Invalid Username/Password message.	Working as expected	Pass		Y		Hari Prasath T
LoginPage_TC_OO13	Functional	Login Page	Verify user is able to login into application with InValid credentials		1.Enter URL and click go 2.Enter email in Email text box 3.Enter invalid password in password text box 4.Click on log In button	Username: hari@gmail.com password: abcd	1.Application should display Invalid Username/Password message.	Working as expected	Pass		Y		Hari Prasath T
LoginPage_TC_OO14	UI	Login Page	Verify the UI elements in Login Page for Google Account Login		1.Enter URL and click go 2.Verify the Google Login UI element. a.Google Login Button	https://fire-management-sys-19220.web.app/	Application should show below UI elements: a. Google Login Button	Working as expected	Pass		N		Hari Prasath T
LoginPage_TC_OO15	Functional	Login Page	1. Verify whether the google login button redirects the user to Google Auth page. 2. Verify whether the user is able to enter the gmail id and password.	The user must have a valid Gmail id and password.	1. Enter the URL and click go. 2. Click Google Login Button. 3. Enter the Gmail ID and Password in the pop up window. 4. Click Enter.	Username: hari@gmail.com password: Testing123	1. Application should login using the user's Google Account credentials. 2. User should be able to view the dashboard page.	Working as expected	Pass		N		Hari Prasath T
LoginPage_TC_OO16	Functional	Login Page	Verify whether the user is able to login using invalid credentials.		1. Enter the URL and click go. 2. Click Google Login Button. 3. Enter the invalid Gmail ID. 4. Enter the password. 5. Click login.	Username: gef@gmail.com password: Testing123	1.Application should display Invalid Username/Password message.	Working as expected	Pass		Y		Hari Prasath T
LoginPage_TC_OO17	Functional	Login Page	Verify whether the user is able to login using invalid credentials.		1. Enter the URL and click go. 2. Click Google Login Button. 3. Enter the Gmail ID. 4. Enter the invalid password. 5. Click login.	Username: hari@gmail.com password: abcd	1.Application should display Invalid Username/Password message.	Working as expected	Pass		Y		Hari Prasath T
LoginPage_TC_OO18	UI	Login Page	Verify the UI elements in SignUp Page for Facebook Account Login	The user must have a valid Facebook id and password.	1.Enter URL and click go 2.Verify the Facebook Login UI element. a.Facebook Login Button	https://fire-management-sys-19220.web.app/	Application should show below UI elements: a. Facebook Login Button	Working as expected	Pass		N		Hari Prasath T
LoginPage_TC_OO19	Functional	Login Page	1. Verify whether the facebook login button redirects the user to facebook Auth page. 2. Verify whether the user is able to enter the facebook id and password.	The user must have a valid Facebook id and password.	1. Enter the URL and click go. 2. Click facebook Login Button. 3. Enter the facebook ID and Password in the pop up window. 4. Click Enter.	Username: hari1234 password: Testing123	1. Application should login using the user's facebook Account credentials. 2. User should be able to view the dashboard page.	Working as expected	Pass		N		Hari Prasath T
LoginPage_TC_OO20	Functional	Login Page	Verify whether the user is able to login using invalid credentials.		1. Enter the URL and click go. 2. Click facebook Login Button. 3. Enter the invalid facebook ID. 4. Enter the password. 5. Click login.	Username: gef4231 password: Testing123	1.Application should display Invalid Username/Password message.	Working as expected	Pass		Y		Hari Prasath T
LoginPage_TC_OO21	Functional	Login Page	Verify whether the user is able to login using invalid credentials.		1. Enter the URL and click go. 2. Click facebook Login Button. 3. Enter the facebook ID. 4. Enter the invalid password. 5. Click login.	Username: hari1234 password: abcd	1.Application should display Invalid Username/Password message.	Working as expected	Pass		Y		Hari Prasath T
LoginPage_TC_OO22	Functional	Database	Verify the user credentials are stored in the realtime database.				The valid user credentials must be stored in the realtime database for future use.	Working as expected	Pass		Y		Jayavardhan P
					1.Enter the URL. 2.Login to the application. 3.View the components of the dasboard. a.Team ID		Application should show below UI elements: a.Team ID (PNT2022TMID19220) label b.Temperature Sensor Gauge Status						



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO23	UI	Dashboard	Verify the components of the dashboard.		1.Enter the URL. 2.Login to the application. 3.View the components of the dasboard. a.Team ID (PNT2022TMID19220)label b.Temperature Sensor Gauge Status c.Flame Sensor Gauge Status d.Gas Level Sensor Gauge Status e.Service Request Option f.SignOut button		Application should show below UI elements: a.Team ID (PNT2022TMID19220) label b.Temperature Sensor Gauge Status c.Flame Sensor Gauge Status d.Gas Level Sensor Gauge Status e.Service Request Option f.SignOut button	Working as expected	Pass		N		Jayavardhan P
LoginPage_TC_OO24	Functional	Dashboard	Verify the working of Gas Level Sensor		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. View the status of the sensor.		The gauge must graphically display the correct accurate reading of the gas level sensor.	Working as expected	Pass		Y		Jayavardhan P
LoginPage_TC_OO25	Functional	Dashboard	Verify the working of Flame Sensor		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. View the status of the sensor.		The gauge must graphically display the correct accurate reading of the Flame level sensor.	Working as expected	Pass		Y		Jayavardhan P
LoginPage_TC_OO26	Functional	Dashboard	Verify the working of Temperature Sensor		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. View the status of the sensor.		The gauge must graphically display the correct accurate reading of the temperature level sensor.	Working as expected	Pass		Y		Jayavardhan P
LoginPage_TC_OO27	Functional UI	Dashboard	Verify whether the user is prompted with alert if fire leak is detected		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. View the status of the sensor.		The application must alert the user with a prompt message "Fire is detected" in case of any fire leakage.	Working as expected	Pass		Y		Jayavardhan P
LoginPage_TC_OO28	Functional	SMS	Verify whether the user receives alert message incase of fire leakage				The user receives a SMS notification in case of any gas leakage.	Working as expected	Pass		Y		Jayavardhan P
LoginPage_TC_OO29	UI	Service Request	Verify whether the user is displayed with options for service when service request button is clicked.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on service button		The user is displayes with a popup for service request. The user is displayed with UI components for a. Service Request Button b. ScalingUp Request Button c. Back to Dashboard Button	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO30	Functional	Service Request	Verify the working of Service request.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on service button 5. Click on Service Request Button.		1.The request for service is sent to the service provider. 2.The user is prompted with "Request Sent" message.	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO31	Functional	Service Request	Verify the working of ScalingUP request.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on service button 5. Click on ScalingUp Request Button.		1.The request for scalingUp is sent to the service provider. 2.The user is prompted with "Request Sent" message.	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO32	Functional	Service Request	Verify the working of Back to Dashboard Button.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on service button 5. Click on Back to Dashboard Button.		The user is redirected to the dashboard.	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO33	Functional	Sign Out	Verify the working of signOut button.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on signout button.		The user is signed out from the application.	Working as expected	Pass		N		Maniesh M
							The user is displayed with the						



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO28	Functional	SMS	Verify whether the user receives alert message incase of fire leakage				The user receives a SMS notification in case of any gas leakage.	Working as expected	Pass		Y		Jayavardhan P
LoginPage_TC_OO29	UI	Service Request	Verify whether the user is displayed with options for service when service request button is clicked.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on service button		The user is displayes with a popup for service request. The user is displayed with UI components for a. Service Request Button b. ScalingUp Request Button c. Back to Dashboard Button	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO30	Functional	Service Request	Verify the working of Service request.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on service button 5. Click on Service Request Button.		1.The request for service is sent to the service provider. 2.The user is prompted with "Request Sent" message.	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO31	Functional	Service Request	Verify the working of ScalingUP request.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on service button 5. Click on ScalingUp Request Button.		1.The request for scalingUp is sent to the service provider. 2.The user is prompted with "Request Sent" message.	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO32	Functional	Service Request	Verify the working of Back to Dashboard Button.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on service button 5. Click on Back to Dashboard Button.		The user is redirected to the dashboard.	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO33	Functional	Sign Out	Verify the working of signOut button.		1. Enter URL. 2. Login to Application. 3. Goto Dashboard. 4. Click on signout button.		The user is signed out from the application.	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO34	UI	Mobile Application	Verify the UI Components of the Mobile Application		1. Download and install the mobile Application. 2. Open the mobile Application.	Fire_Management_System.apk	The user is displayed with the Mobile Application UI components. 1. Team ID (PNT2022TMID19220) label. 2. Temperature Level Label. 3. Gas Level Label. 4. Flame level Label. 5. Textbox to display temperature level in degree celsius. 6. Textbox to display gas level in ppm. 7. Textbox to display flame level in units.	Working as expected	Pass		N		Maniesh M
LoginPage_TC_OO35	Functional	Mobile Application	Verify whether the mobile application shows the realtime monitoring of the status of the fire system monitoring system		1. Download and install the mobile Application. 2. Open the mobile Application.	Fire_Management_System.apk	The mobile application displays the realtime status of the sensors in the fire management system	Working as expected	Pass		Y		Maniesh M
LoginPage_TC_OO36	Functional	SMS	Verify whether the user receives SMS notification incase of fire spread detection.				The user receives a SMS notification in case of any smoke detection	Working as expected	Pass		Y		Maniesh M

8.2. UAT

Defect Analysis

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	8	4	1	0	13
Duplicate	1	0	3	0	4
External	4	3	0	1	8
Fixed	13	7	4	1	25
Not Reproduced	0	0	1	0	1
Skipped	1	0	0	0	1
Won't Fix	0	0	0	0	0
Totals	27	14	9	2	52

Test Case Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	3	0	0	3
Client Application	36	0	0	36
Security	4	0	0	4
Outsource Shipping	1	0	0	1
Exception Reporting	10	0	0	10
Final Report Output	4	0	0	4
Version Control	1	0	0	1

9. RESULTS

9.1. Performance Metrics

CPU Usage:

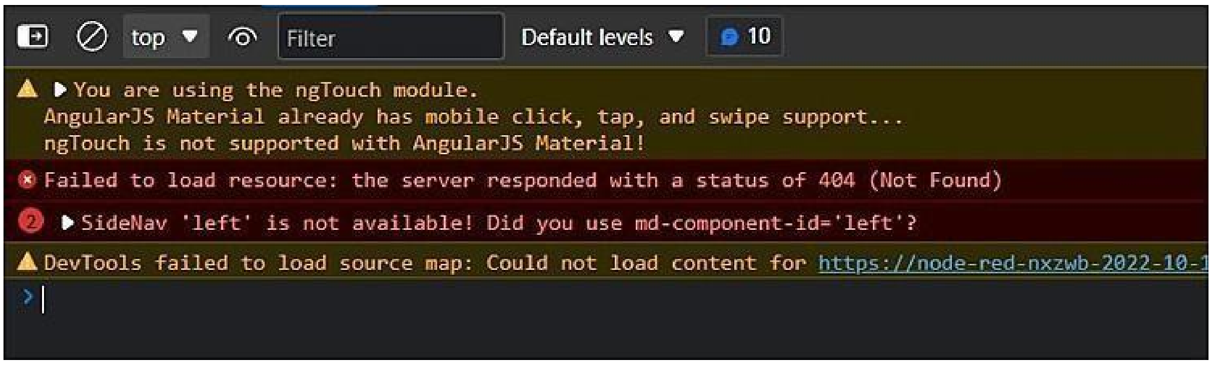
The micro version of python makes the most efficient use of the CPU. The program runs in O(1) time for each loop, ignoring the network and communication. To improve communication with MQTT, the program sleeps every 1 second. Because the program runs in O(1) time and the compiler optimizes it during compilation, 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:

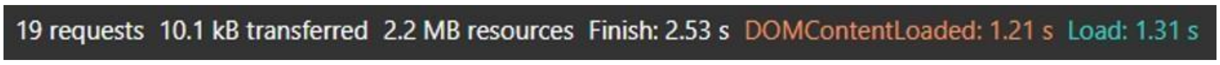
The error rates are very low because the backend and dashboard are handled with node-red.Exceptions are handled properly so that the system's usability is not affected.



```
▶ You are using the ngTouch module.
AngularJS Material already has mobile click, tap, and swipe support...
ngTouch is not supported with AngularJS Material!
Failed to load resource: the server responded with a status of 404 (Not Found)
2 ▶ SideNav 'left' is not available! Did you use md-component-id='left'?
DevTools failed to load source map: Could not load content for https://node-red-nxzwb-2022-10-1
```

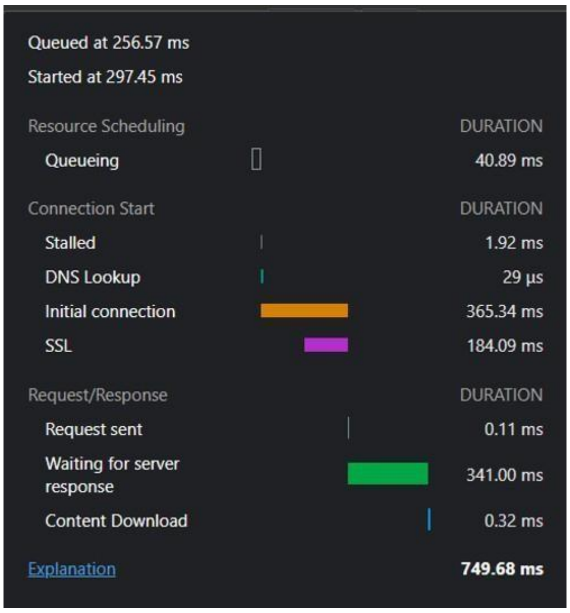
Latency and Respose Time:

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.



19 requests 10.1 kB transferred 2.2 MB resources Finish: 2.53 s DOMContentLoaded: 1.21 s Load: 1.31 s

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



Queued at 256.57 ms
Started at 297.45 ms

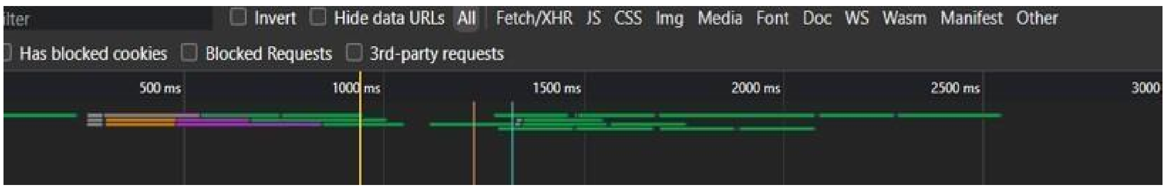
	DURATION
Resource Scheduling	DURATION
Queueing	40.89 ms
Connection Start	DURATION
Stalled	1.92 ms
DNS Lookup	29 µs
Initial connection	365.34 ms
SSL	184.09 ms
Request/Response	DURATION
Request sent	0.11 ms
Waiting for server response	341.00 ms
Content Download	0.32 ms
Explanation	749.68 ms

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.

10. ADVANTAGES & DISADVANTAGES

Advantages:

- Proactive detection of fires and
- SMS notification of administrators and fire departments
- Automatically turning on and off exhaust fans and sprinklers
- Authentication is needed to manually turn on/off exhaust fans, sprinklers, and send SMS notifications.
- It immediately recognises bogus fire outbreaks, which lessens needless anxiety.
- Using flow sensors, we can verify that the sprinkler system is operational.
- A dashboard can show any device's status.
- Users can access the dashboard through a web application
- Users can access the dashboard through a web application.

Disadvantages:

- Internet access is always necessary just to deliver the SMS alert
- The entire operation fails if the hardware device malfunctions.
- A huge database is needed since the cloud database stores a lot of data every second.

11. CONCLUSION

By developing a smart management system that actively monitors for fire breakouts and gas leaks, sends Sms notifications to administrators and fire authorities, and addresses many other inherent issues with traditional fire management systems, we can therefore conclude that our problem premise is solved utilising IoT devices

12. FUTURE SCOPE

While fire mishaps can result in major loss of human life in both households and huge industries, the existing devices can be scaled up for use in houses and large labs, as well as in public spaces and automobiles.

13. 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.

Github Link: **https://github.com/IBM-EPBL/IBM-Project-6429-1658828933**

Demo Video Link:

https://github.com/IBM-EPBL/IBM-Project-6429-1658828933/tree/main/Final%20Deliverables/Demo%20Video