**INDUSTRY SPECIFIC INTELLIGENT FIRE**

**MANAGEMENT SYSTEM**

**A PROJECT REPORT**

|  |  |
| --- | --- |
| ***Submitted by*** |  |
| **BOOMESHWARAN V** | **(711119104017)** |
| **CHILAKAM BINDU REDDY** | **(711119104018)** |
| **DANAENDRARAJ R** | **(711119104019)** |
| **GAYAM NAVEENA** | **(711119104023)** |

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

**JANSONS INSTITUTE OF TECHNOLOGY, COIMBATORE**

**ANNA UNIVERSITY: CHENNAI 600 025**

**INDEX**

1. **INTRODUCTION** 
   1. Project Overview 3
   2. Purpose 3
2. **LITERATURE SURVEY**
   1. Existing problem 4
   2. References 6
   3. Problem Statement Definition 6
3. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas 7
   2. Ideation & Brainstorming 8
   3. Proposed Solution 10
   4. Problem Solution fit 11
4. **REQUIREMENT ANALYSIS**
   1. Functional requirement 12
   2. Non-Functional requirements 12
5. **PROJECT DESIGN**
   1. Data Flow Diagrams 14
   2. Solution & Technical Architecture 14
   3. User Stories 16
6. **PROJECT PLANNING & SCHEDULING**
   1. Sprint Planning & Estimation 17
   2. Sprint Delivery Schedule 19
   3. Reports from JIRA 21
7. **CODING & SOLUTIONING**
   1. IBM Watson IOT Platform interfaced with NODE-RED 22
   2. Automated Alerting using Fast2sms 26
8. **TESTING** 
   1. Test Cases 29
   2. User Acceptance Testing 32
9. **RESULTS**
   1. Performance Metrics 34
10. **ADVANTAGES & DISADVANTAGES** **35**
11. **CONCLUSION** **36**
12. **FUTURE SCOPE** **36**
13. **APPENDIX**

Source Code 37

GitHub & Project Demo Link 61

**1.INTRODUCTION**

**1.1 PROJECT OVERVIEW**

IOT devices are hardware devices, such as sensors, gadgets, appliances and other machines that collect and exchange data over the Internet.

➢ Sending random fire and temperature values will be sent to the IBM IOT platform

➢ Sensors values can be viewed in the Web Application

➢ Notifies the admin the random values cross the threshold value

✓ To accomplish this, we have to complete all the activities and tasks listed below:

Create and configure IBM Cloud Services

➢ Create IBM Watson IoT Platform

➢ Create a device & configure the IBM IoT Platform

➢ Create Node-RED service

➢ Create a database in Cloudant DB to store location data

Develop a web Application using Node-RED Service

➢ Develop the web application using Node-RED.

➢ Develop a python script to publish the location details to the IBM IoT platform.

**1.2 PURPOSE**

As the term suggests, fire safety management is all about putting in place the arrangements to help protect both people and property from fire. This encompasses not only helping to prevent a fire from breaking out in the first place, but also helping to shield people from harm in the event that a fire does occur.

* It provides safe environment for the workers who are working in the industries.
* This Application is used to reduce the intensity of the damage.
* The deaths caused by the fire accident can be reduced

**2. LITERATURE SURVEY**

**2.1 EXISTING PROBLEM**

The fire alarm monitoring system of fire protection engineering has formed a complete system, including alarm monitoring, automatic fire control, fire linkage control, and fire data monitoring and analysis modules. This project mainly analyses the fire alarm computer monitoring system in fire engineering. The measuring devices and single-chip microcomputer in the intelligent fire early warning and monitoring system own many advantages in judging the fire situation. The structural form is a structure that organically combines the main network and the secondary network, so it has strong adaptability and can effectively improve the reliability of the building's fire emergency function. At the same time, the fire alarm computer monitoring system in the fire protection engineering can faithfully reflect the small changes in the monitored environmental objects, and can automatically compensate for the factors that prevent fires in the environment, it also can automatically handle the adverse effects caused by electrical interference. The use of software code programming in the system significantly improves the linkage and repairability of the system regarding fire protection.

The benefits of this fire detection system can detect early fire occurrence based on the detection of temperature conditions by accommodating the nature of the fire and able to detect any rise in temperature caused by the existence of the fire. To realize the system, required sensors capable of reading the temperature and smoke. The Arduino Uno microcontroller is the brain control system of the system. At a temperature of> 35 C, the system will activate the DHT 11 and MQ 2 sensors that detect smoke> 50 ppm from fire. The system will activate Buzzer as a warning in the form of the next alarm sound Global Positioning System (GPS) will provide information in the form of coordinates of the location of the point of fire through GSM SIM900 Module Short Message Service (SMS) to the user. The results obtained mq2 = 128 ppm and temperature value = 38 ° C and GPS data with latitude of -3.04798388 and a longitude of 104.78263092. From the data it is seen that the mq2 value reaches> 50ppm and the temperature value reaches> 35 ° C, and the detector outputs buzzer sound and warning notification of coordinate point in the form of SMS containing the message "FIRE available" with the coordination of the location of the fire detected by GPS.

This research refers to an Arduino and Global System for Mobile (GSM) based system for efficient detection of fire hazards. This purpose is industrial and domestic safety, and the primary concern is to avoid the fire hazards that occur to the employees and the properties inside the buildings. As a solution, a smart fire and high-temperature detection system is designed using GSM technology, smoke/temperature sensors, and Arduino technology. A smoke sensor is used to detect the smoke from the fire and a temperature sensor is used to detect temperature increase inside the building. In the event of a fire, an alert message will be sent to the user via short message service (SMS) via the GSM module. Furthermore, when a fire is detected, a signal will be sent to the main power supply circuit breaker via a microcontroller and then the power supply of the particular building will shut down. Results from the test are documented and discussed in this paper. This system helps users to respond immediately to the situation and so improve their safety by protecting their lives and the properties from a disaster.

To help with fire-fighting operations, an alarm application based on Telos B motes was proposed in (Bernardo, Oliveira et al. 2007). The authors used a combination of temperature, light and humidity sensors in difficult access environments. They considered a scattered WSN consisting of several isolated WSNs. The situation, in which sensor nodes are destroyed by fire, was also taken into account. They concluded that mote longevity (avoiding synchronisation costs during idle period) can be applied in the fire situations where a timely response to destructive events is needed. In Bagheri 2007, the author utilised FWI index and his novel k-coverage algorithm to detect forest fires. K-coverage algorithm monitors each point by using k or more sensor nodes to increase fault tolerance. Therefore, some sensors can be put in standby mode to extend network lifetime. Although there are many algorithms to find the minimum number of sensors to be used, they are usually NP complete problems. The proposed k-coverage solution proved to prolong the network life time. Forest fire detection was not the focus of this work and was considered as an application for the novel k-coverage problem

**2.2 REFERENCES**

Bagheri, M. (2007). Efficient K-Coverage Algorithms for Wireless Sensor Networks and Their Applications to Early Detection of Forest Fires. Computing Science, SIMON FRASER UNIVERSITY. MSc.

Baumann, A., M. Boltz, et al. (2008). "A Review and Comparison of Measures for Automatic Video Surveillance Systems." EURASIP Journal on Image and Video Processing.

Bernardo, L., R. Oliveira, et al. (2007). A Fire Monitoring Application for Scattered Wireless Sensor Networks: A peer-to-peer cross-layering approach International Conference on Wireless Information Networks and Systems (WINSYS'07), Barcelona, Spain.

Vescoukis, V., T. Olma, et al. (2007). Experience from a Pilot Implementation of an "In-Situ" Forest Temperature Measurement Network. Personal, Indoor and Mobile Radio Communications (PIMRC2007). Wikipedia EN 54, <http://en.wikipedia.org/wiki/EN_54>.

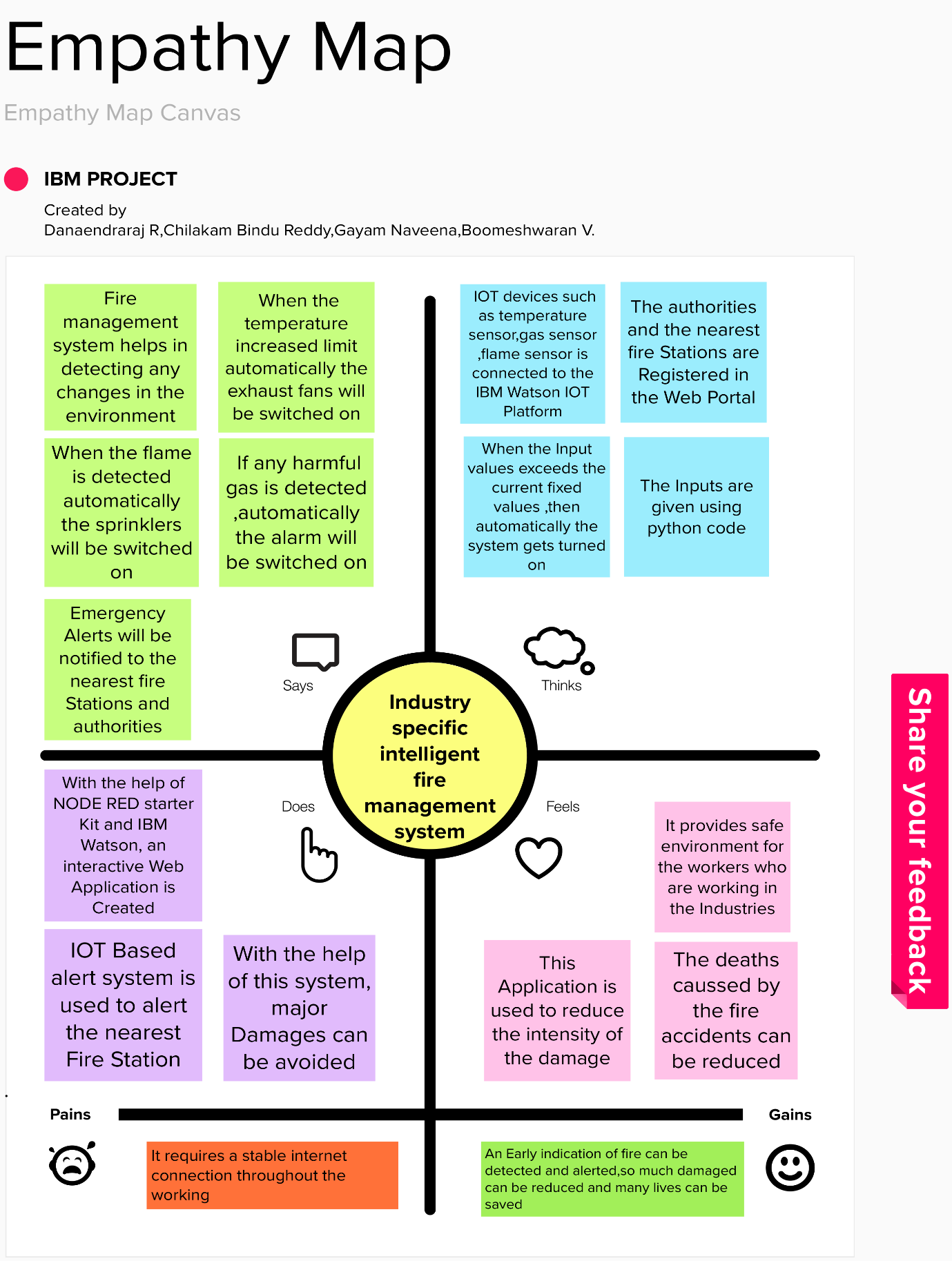
Zhiping, L., Q. Huibin, et al. (2006). The Design of Wireless Sensor Networks for Forest Fire Monitoring System. School of Electronics and Information, Hangzhou Dianzi University, White Paper.

Brain, M. (2000). "How Smoke Detectors Work ", from <http://home.howstuffworks.com/smoke1.htm>.

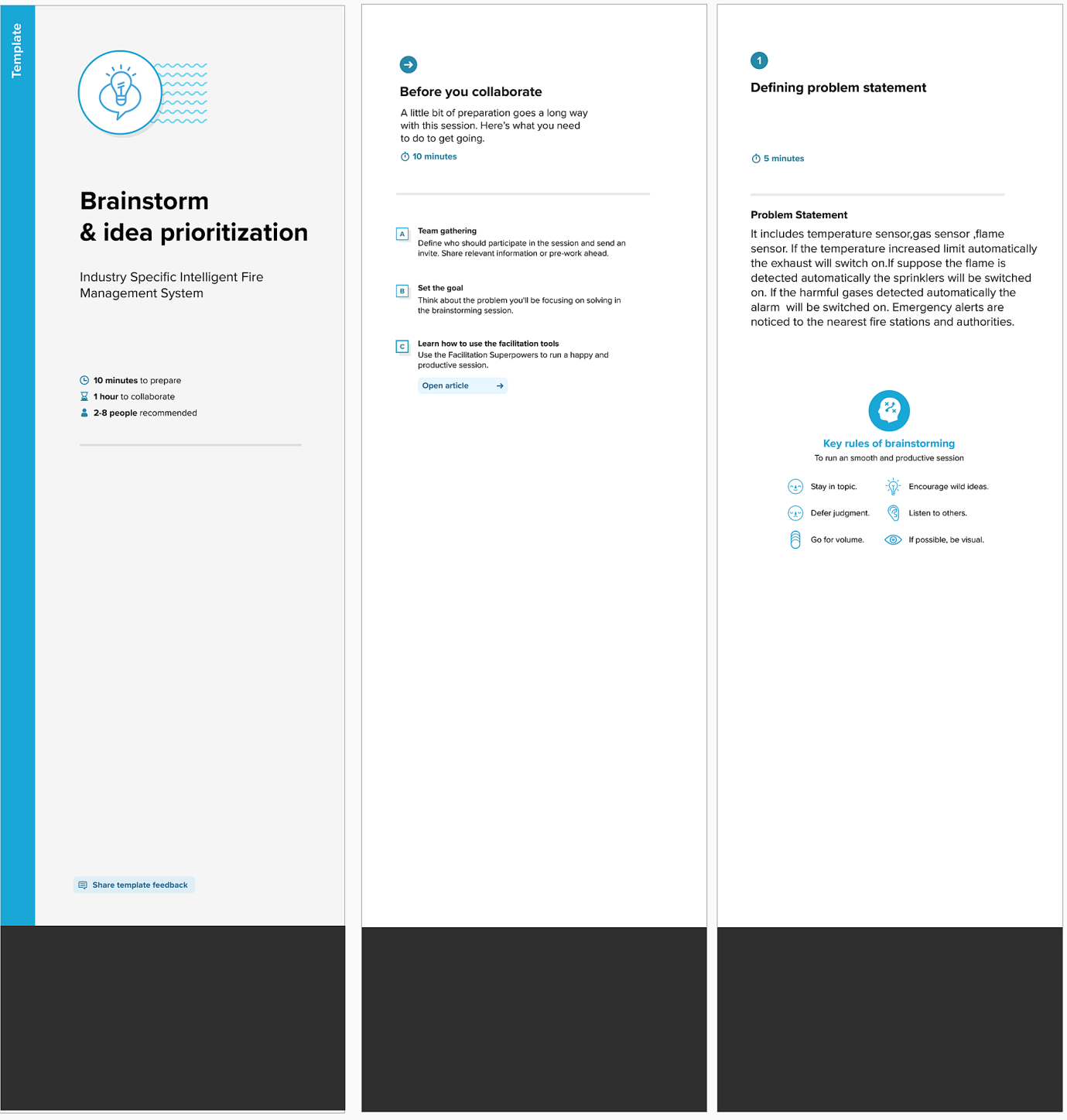
**2.3 PROBLEM STATEMENT DEFNITION**

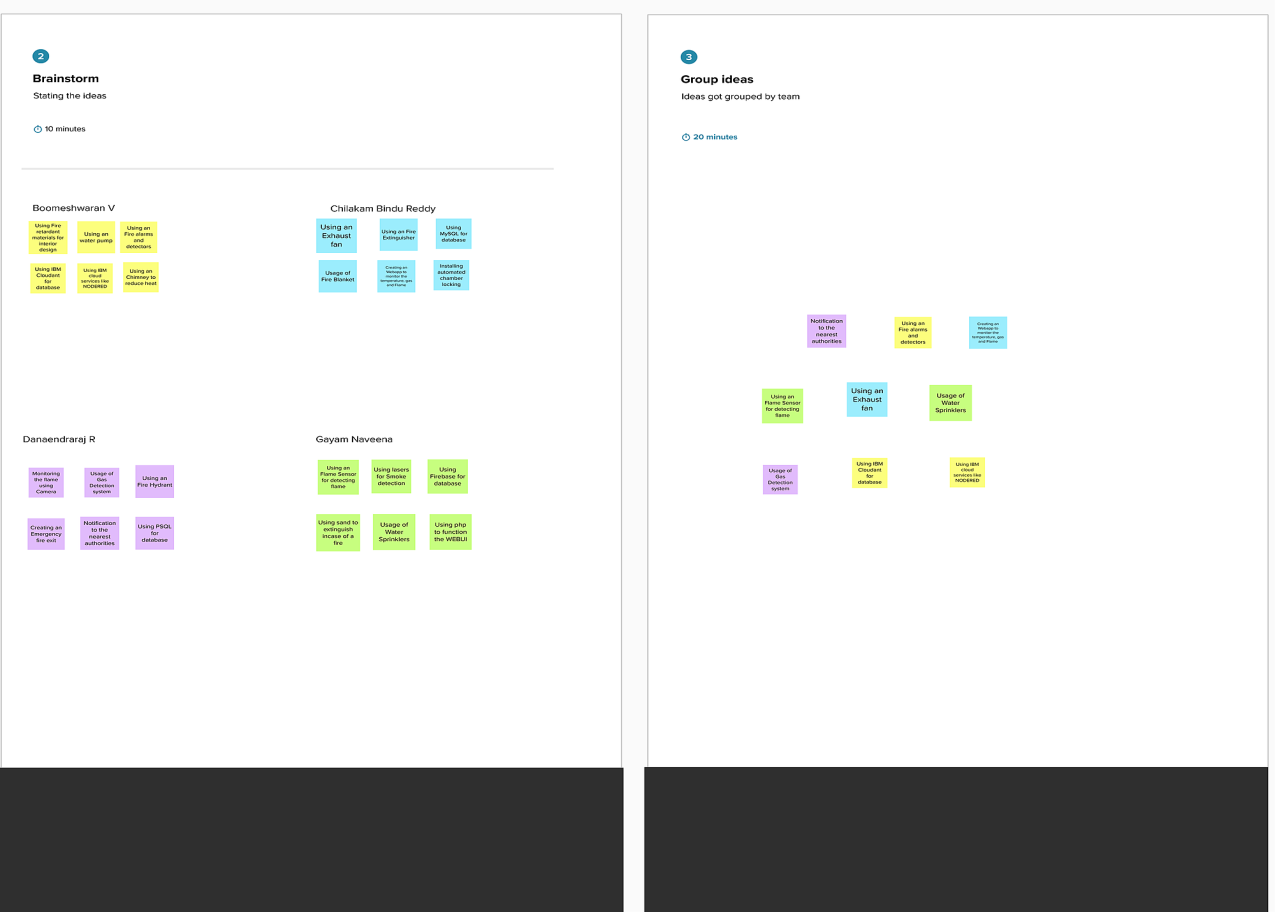
It includes temperature sensor, gas sensor, flame sensor. If the temperature increased limit automatically the exhaust will switch on. If suppose the flame is detected automatically the sprinklers will be switched on. If the harmful gases detected automatically the alarm will be switched on. Emergency alerts are noticed to the nearest fire stations and authorities.

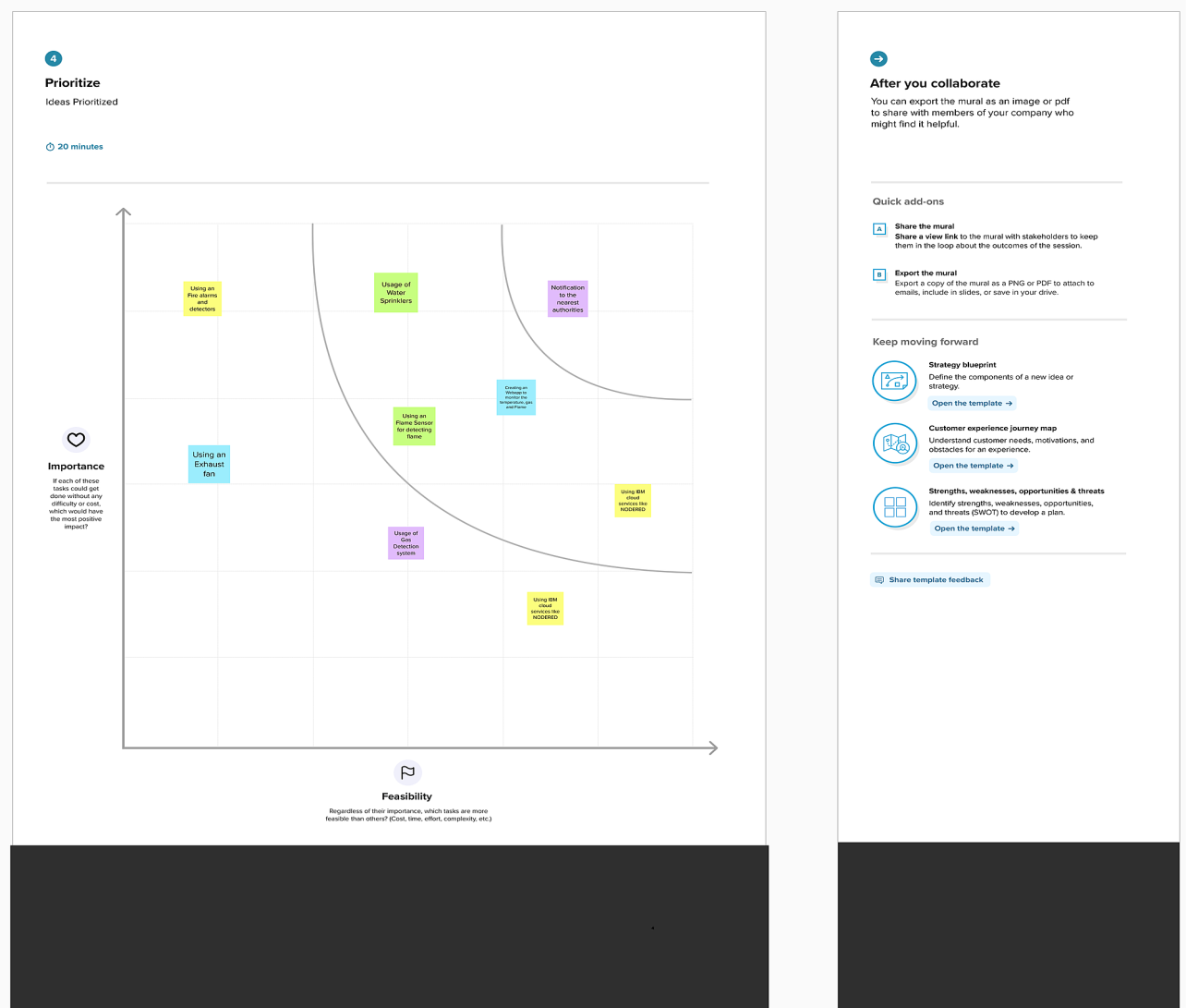
**3.IDEATION AND PROPOSED SOLUTION**

**3.1 EMPATHY MAP CANVAS**

**3.2 IDEATION AND BRAINSTROMING**



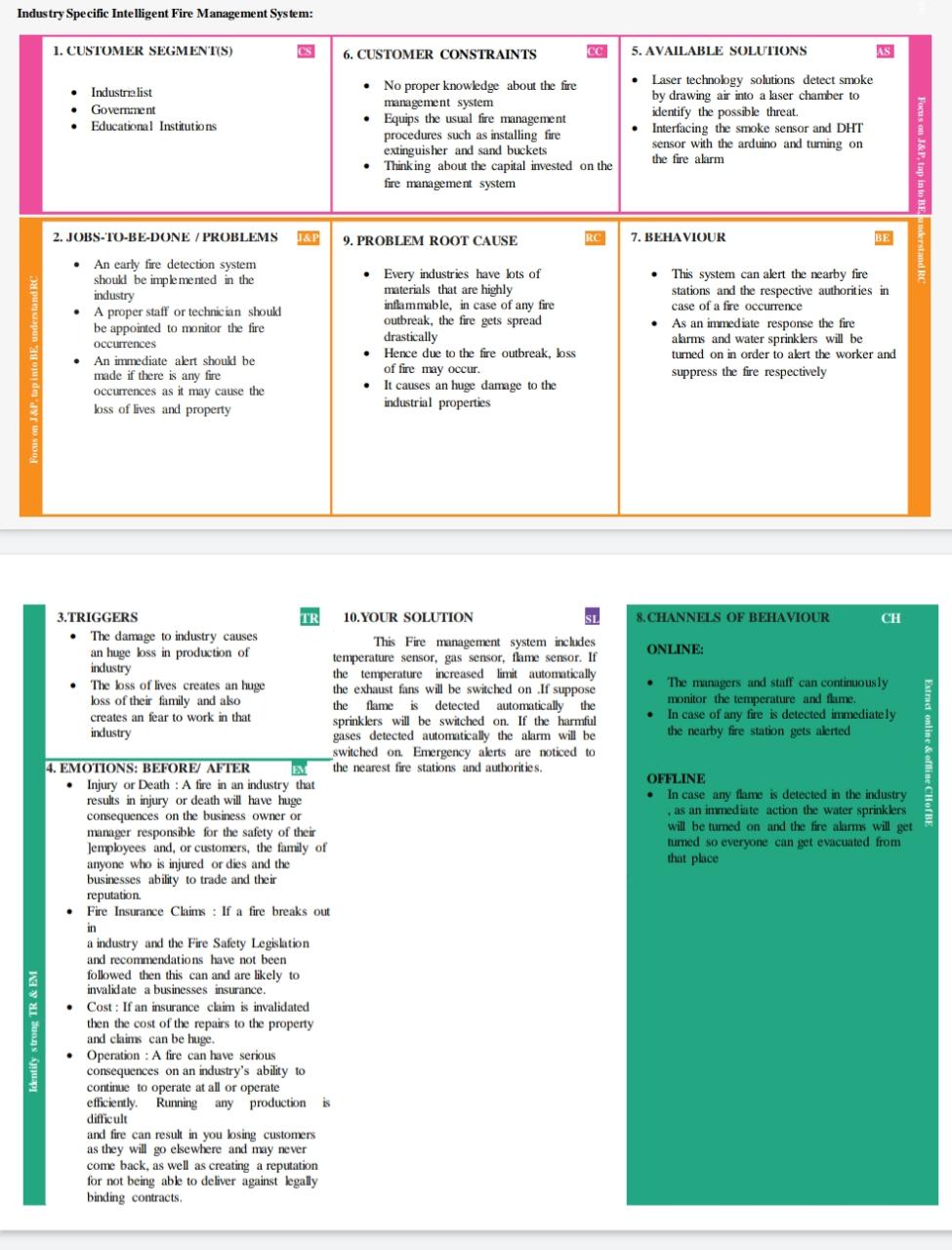




**3.3 PROPOSED SOLUTION**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be  solved) | To improve the Fire management and alert  system in industries. Providing a safe environment for the workers. |
| 2. | Idea / Solution description | To provide fire management and alert system  in industry based on IOT with fire detection. And using some sensors (Gas sensor, Flame  sensor, temperature sensor). |
| 3. | Novelty / Uniqueness | An fire system Integrated with temperature  sensor, gas sensor, flame sensor , exhaust fan  ,water sprinkler and fire alert through SMS to the authority. |
| 4. | Social Impact / Customer Satisfaction | It reduces the damages caused by fire accidents  in industries. And also reduce the damage in  the nearby locations and the environment , reduce the pollution. |
| 5. | Business Model (Revenue Model) | This product can be used in industries .This  system can be used as a safety system in  industries, which reduce the damage and can be used avoid decease from the fire accident. |
| 6. | Scalability of the Solution | This project trying to execute the method with  an help of an IOT gadget which provide a safe environment for workers in industries. Easy to install and maintain .It can be used in multiple  industry area. |

**3.4 PROBLEM FIT SOLUTION**



**4.REQUIREMENT ANALYSIS**

**4.4 FUNCTIONAL REQUIREMENTS**

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form  Registration through phone number |
| FR-2 | User Registration Successful | Registration Successful message in web window  Storing the registered data in IBM Cloundant |
| FR-3 | Creating IOT Devices | Creating Sensors in IOT Watson Platform  Creating a simulation and sending data to NODE-RED Dashboard |
| FR-4 | Monitoring the Sensor values | Creating a Node-RED Dashboard  Publishing the data in the dashboard |
| FR-5 | Sending Emergency Alerts | Fetching the phone number of the registered users and Sending emergency alerts through Fast2SMS |

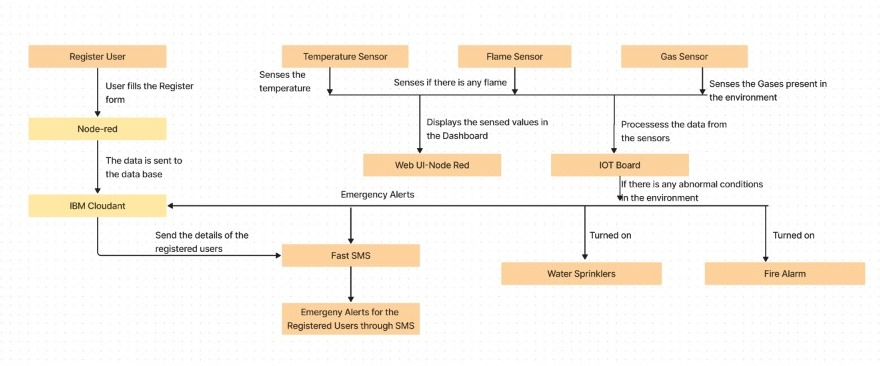
**4.5 NON-FUNCTIONAL REQUIREMENTS**

Following are the non-functional requirements of the proposed solution.

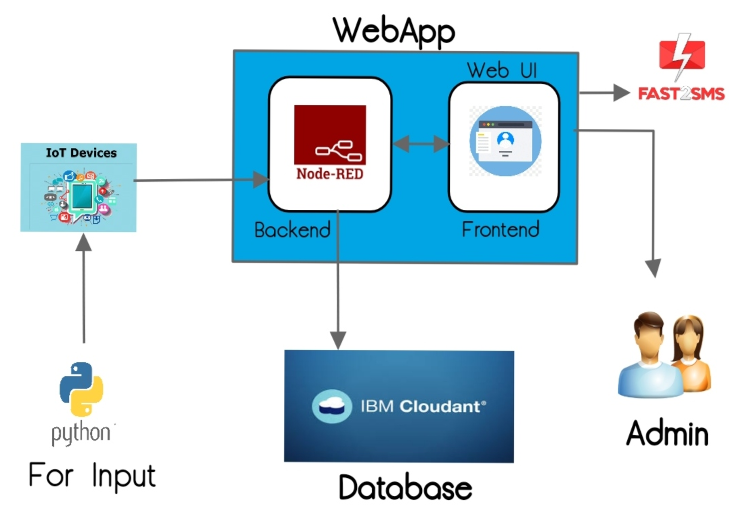
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | The system should be easy for the customers to use. Ease to maintenance.  Low pressure to workload.  The instructions should be straightforward. |
| NFR-2 | **Security** | No should have access to modify the data sent from the sensors.  It should resist the attacks made by the hackers. |
| NFR-3 | **Reliability** | The system should be made fail-safe operation using redundancy.  If one module fails then the parallel module should take over the operation.  Response timer will be faster.  Hardware components are routinely examined. |
| NFR-4 | **Performance** | All the modules should work without any connection interruption.  Quickly Responsive.  Efficiency is higher than fire alarms |
| NFR-5 | **Availability** | The system should be continuously monitoring the environment 24/7.  Ability to use the system for other types of emergency communication.  Based on the user's needs, all function should be offered. |
| NFR-6 | **Scalability** | The system should have the capacity to accommodate a greater amount of usage.  Easy to adoptable for future changes.  Based on Fire alarm signals and should cover the entire Placed. |

**5. PROJECT DESIGN**

**5.1 DATA FLOW DIAGRAMS**



**5.2 SOLUTION AND TECHNICAL ARCHITECTURE**



1. The user should provide information to get registered in the web application and this user database will be stored in IBM cloudant through node-red.

2. Here we use IOT’s such as temperature sensor, gas sensor, flame sensor.

3. Temperature sensor senses the temperature, flame sensor senses if there is any flame and the gas sensor senses the gases present in the environment.

4. The Web UI-Node Red will display the sensed values in the dashboard. Whereas IOT Board process the data from the sensors, if any changes detected in the environment then it will turn on the water sprinklers and fire alarm.

5. It will also send an emergency alert to IBM cloudant so that the registered user will receive a Fast SMS and nearby fire station will be notified

**5.3 USER STORIES**

|  |  |
| --- | --- |
| **User Story Number** | **User Story / Task** |
| **Sprint 1** | |
| USN-1 | As a admin, I can log into the application by entering username & password |
| USN-2 | When the admin, doesn't enter the username it display an error message popup |
| USN-3 | When the admin, doesn't enter the password it displays an error message popup |
| USN-4 | When the admin,enters the invalid credentials it  displays an error popup |
| USN-5 | When the admin enter the correct username and password it redirects to the dashboard |
| **Sprint 2** | |
| USN-1 | Creating device in IBM IOT Watson Platform |
| USN-2 | Devoloping a python script to publish data to ibm cloud |
| **Sprint 3** | |
| USN-1 | Creating a node red dashboard |
| USN-2 | Connecting the node red dashboard and ibm Watson Platform |
| USN-3 | Creating a Registered Form in node red |
| USN-4 | Registering the user to get alerts |
| **Sprint 4** | |
| USN-1 | As a admin, I can monitor the sensor data continuously in the dashboard |
| USN-2 | Retrieving the data from cloudant and send alerts if the values cross the threshold values |
| USN-3 | Python script for sending alerts using fast2sms |

**6. PROJECT PLANNING & SCHEDULING**

**6.1. Sprint Planning & Estimation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-1 | Login | USN-1 | As a admin, I can log into the application by entering username & password | 4 | Medium | Boomeshwaran V |
| Sprint 1 |  | USN-2 | When the admin, doesn't enter the username it display an error message popup | 4 | Medium | Boomeshwaran V |
| Sprint 1 |  | USN-3 | When the admin, doesn't enter the password it displays an error message popup | 4 | Medium | Boomeshwaran V |
| Sprint 1 |  | USN-4 | When the admin,enters the invalid credentials it  displays an error popup | 4 | Medium | Boomeshwaran V |
| Sprint 1 |  | USN-5 | When the admin enter the correct username and password it redirects to the dashboard | 4 | High | Boomeshwaran V |
| Sprint 2 | Creating Device | USN-1 | Creating device in IBM IOT Watson Platform | 10 | High | Chilakam Bindu Reddy |
| Sprint 2 | Python Code | USN-2 | Devoloping a python script to publish data to ibm cloud | 10 | High | Chilakam Bindu Reddy |
| Sprint 3 | Dashboard | USN-1 | Creating a node red dashboard | 4 | Medium | Danaendraraj R |
| Sprint 3 |  | USN-2 | Connecting the node red dashboard and ibm Watson Platform | 5 | High | Danaendraraj R |
| Sprint 3 | IBM Watson IOT | USN-3 | Creating a Registered Form in node red | 6 | High | Danaendraraj R |
| Sprint 3 |  | USN-4 | Registering the user to get alerts | 6 | High | Danaendraraj R |
| Sprint-4 | Monitoring | USN-1 | As a admin, I can monitor the sensor data continuously in the dashboard | 2 | Medium | Gayam Naveena |
| Sprint 4 | Alerts | USN-2 | Retrieving the data from cloudant and send alerts if the values cross the threshold values | 10 | High | Gayam Naveena |
| Sprint 4 | Python | USN-3 | Python script for sending alerts using fast2sms | 8 | High | Gayam Naveena |

**6.2. Sprint Delivery Schedule**

**Project Tracker, Velocity & Burndown Chart:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 4 Days | 31 Oct 2022 | 3 Nov 2022 | 20 | 2 Nov 2022 |
| Sprint-2 | 20 | 5 Days | 04Nov 2022 | 08 Nov 2022 | 20 | 7 Nov 2022 |
| Sprint-3 | 20 | 5 Days | 09 Nov 2022 | 13 Nov 2022 | 20 | 13 Nov 2022 |
| Sprint-4 | 20 | 4 Days | 14 Nov 2022 | 17 Nov 2022 | 20 | 17 Nov 2022 |

**Velocity:**

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)

Sprint 1=AV=Sprint Duration/velocity=20/4=5

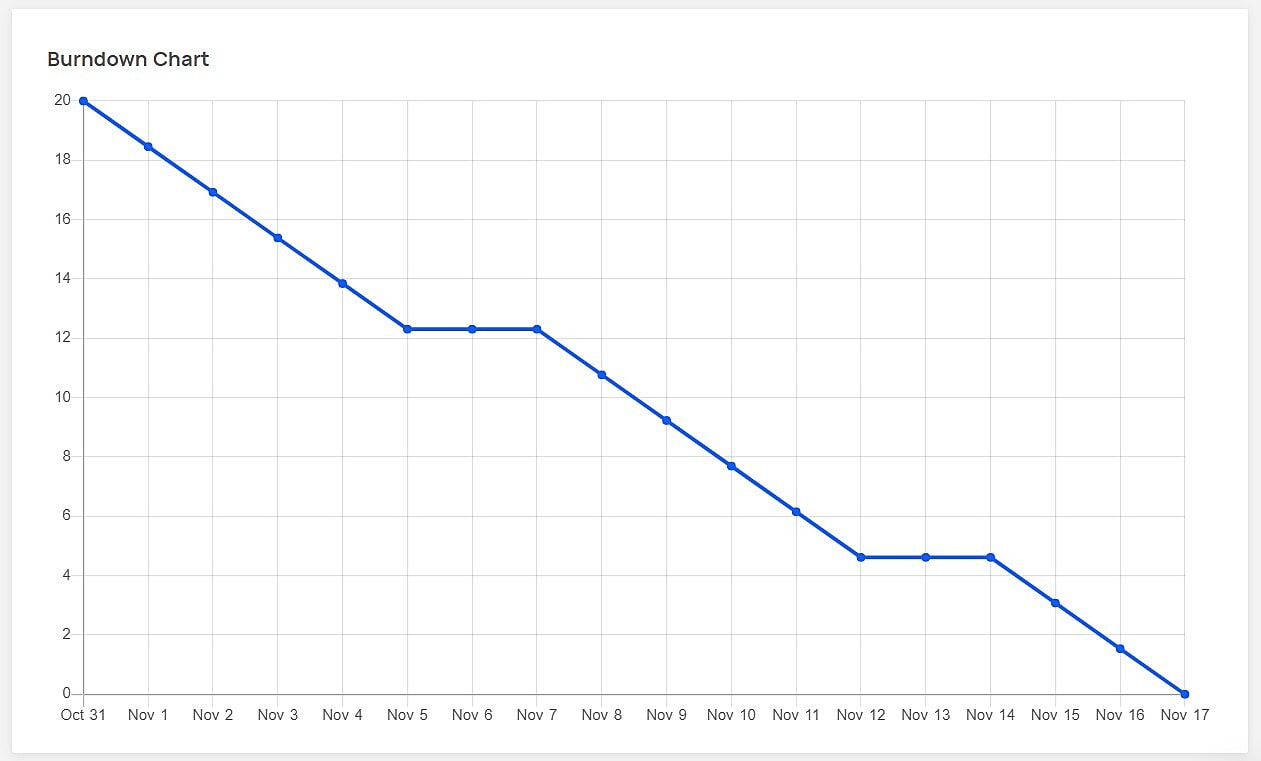
Sprint 2=AV=Sprint Duration/velocity=20/5=4

Sprint 3=AV=Sprint Duration/velocity=20/5=4

Sprint 4=AV=Sprint Duration/velocity=20/4=5

**Burndown Chart:**

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile[software development](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/)methodologies such as [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/). However, burn down charts can be applied to any project containing measurable progress over time.



[**https://www.visual-paradigm.com/scrum/scrum-burndown-chart/**](https://www.visual-paradigm.com/scrum/scrum-burndown-chart/)

[**https://www.atlassian.com/agile/tutorials/burndown-charts**](https://www.atlassian.com/agile/tutorials/burndown-charts)

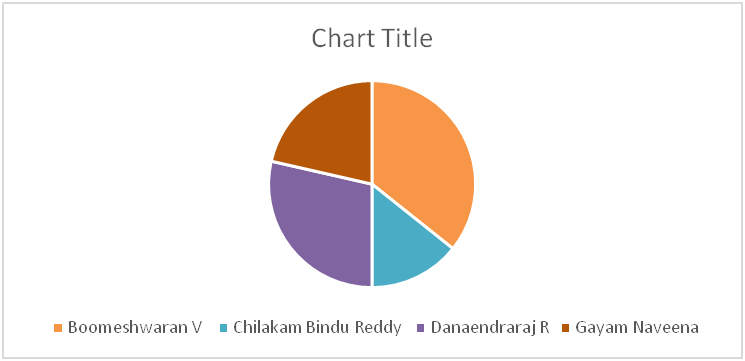
**6.3. Reports from JIRA**

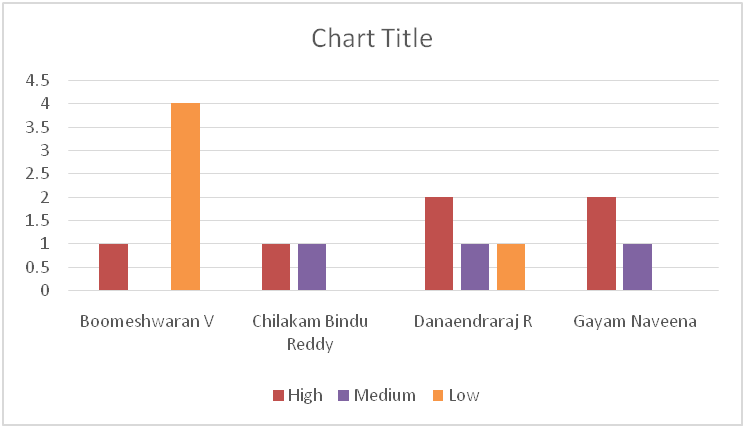
JIRA software is used to create user Stories and issues,plan sprints and distribute task among the software team.Jira Software is part of a family of products designed to help teams of all types manage work. Originally, Jira was designed as a bug and issue tracker. But today, Jira has evolved into a powerful work management tool for all kinds of use cases, from requirements and test case management to agile software development.

Report:

Total Number of User Stories: 14

The Pie chart displays the user story distribution



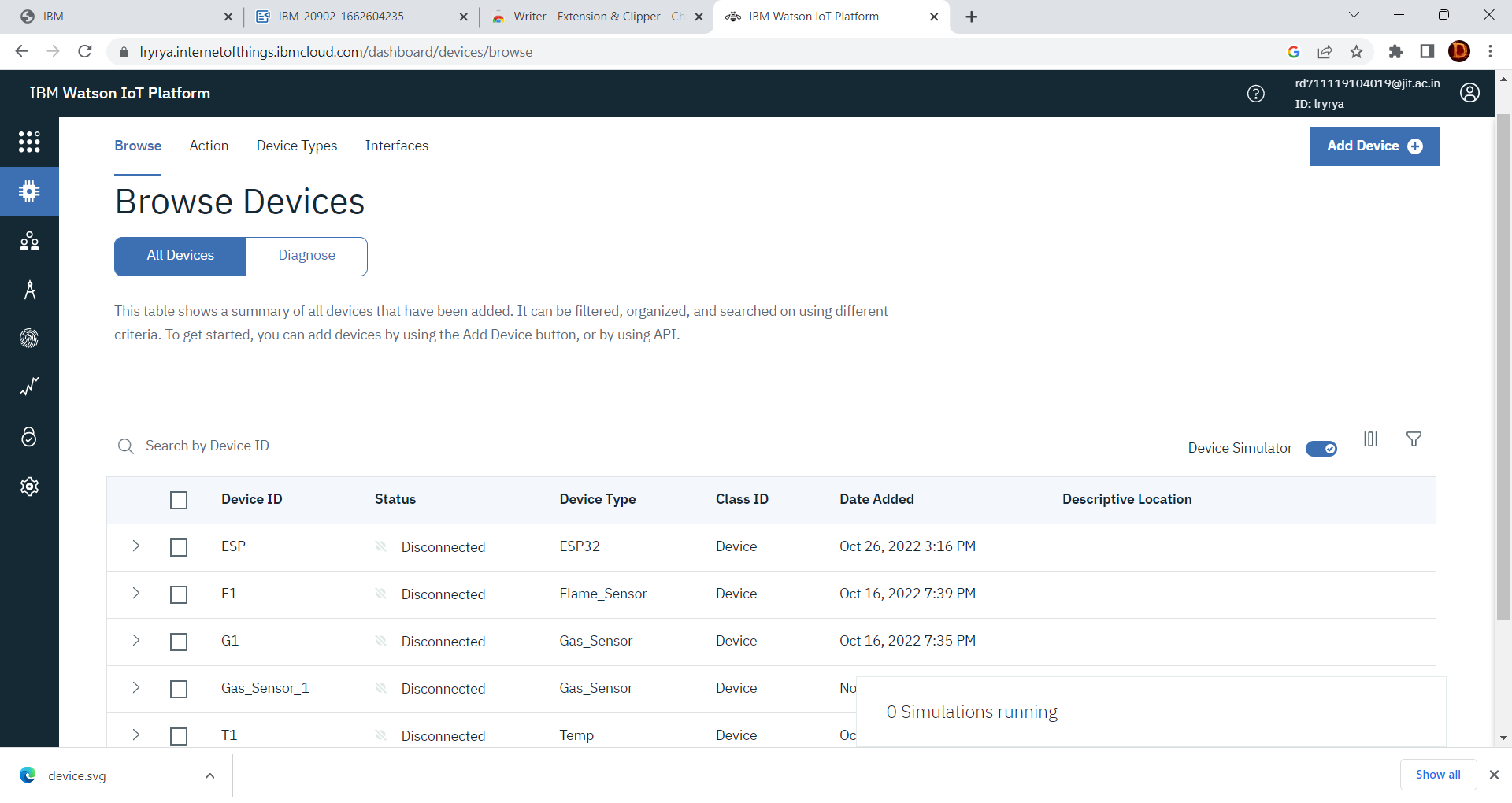
Based on the priority the user story are assigned as

**7.CODING & SOLUTIONING:**

**7.1. IBM Watson IOT Platform interfaced with NODE-RED**

IBM Watson™ IoT Platform is a fully managed, cloud-hosted service that makes it simple to derive value from Internet of Things (IoT) devices.

Simply register and connect your device, be it a sensor, a gateway, or something else, to Watson IoT Platform and start sending data securely up to the cloud using the open, lightweight MQTT messaging protocol. You can set up and manage your devices using your online dashboard or our secure APIs, so that your apps can access and use your live and historical data.



**Basic concepts**

Familiarize yourself with some basic Platform Service concepts.

**Organizations**

When you register with the Platform Service, you are provided with an organization ID, a unique six character identifier for your account. Organizations ensure that your data is only accessible by your devices and applications.

After registration, [devices](https://www.ibm.com/docs/en/watson-iot-platform?topic=product-overview-features#devices)and API keys are bound to a single organization. When an [application](https://www.ibm.com/docs/en/watson-iot-platform?topic=product-overview-features#applications)connects to the service by using an API key, it registers to the organization that is associated with that API key.

**Important**: For security reasons, direct communication between different organizations is not possible. To transmit data between two organizations, you must register one application with each organization and have the two applications communicate with each other to transfer data.

**Devices**

A device can be anything that has a connection to the internet and that can push data into the cloud. However, devices cannot communicate directly with other devices, instead devices accept commands from applications, and send events to applications.

Devices in Platform Service are identified by a unique authentication token. Devices must be registered before they can connect to Platform Service.

Platform Service recognizes two classes of device:

|  |  |
| --- | --- |
| **Device type** | **Description** |
| Managed devices | Devices that contain a device management agent. A device management agent is a set of logic that allows the device to interact with the Platform Service Device Management service by using the Device Management Protocol. Managed devices can perform device management operations, including location updates, firmware downloads and updates, restarts, and factory resets. |
| Unmanaged devices | Devices without a device management agent. Unmanaged devices can connect to the Platform Service and send and receive events and commands, but they cannot send device management requests or perform device management operations. |

**Gateways**

Gateways are specialized devices that have the combined capabilities of an application and a device. This lets them serve as access points for other devices. Devices that cannot connect directly to the internet can access the Platform Service service by first connecting to the gateway device.

Gateways must be registered before they can connect to the service, and like standard devices can be managed or unmanaged.

**Applications**

An application is anything that has a connection to the internet and interacts with data from devices and control the behavior of those devices. Applications identify themselves with the Platform Service by using an API key and a unique application ID. Unlike devices, individual applications do not need to register before they can connect to the Platform Service. However, they must use a valid registered API key.

**Events**

Events are the mechanism by which devices publish data to the Platform Service. Devices control the content of their messages, and assign a name for each event that is sent. The Platform Service uses the credentials that are attached to each event received to determine which device sent the event. This architecture prevents devices from impersonating one another.

Applications can process events in real time, and see the source of the event and the data that is contained if. Applications must be configured to define which devices and events they subscribe to.

**Commands**

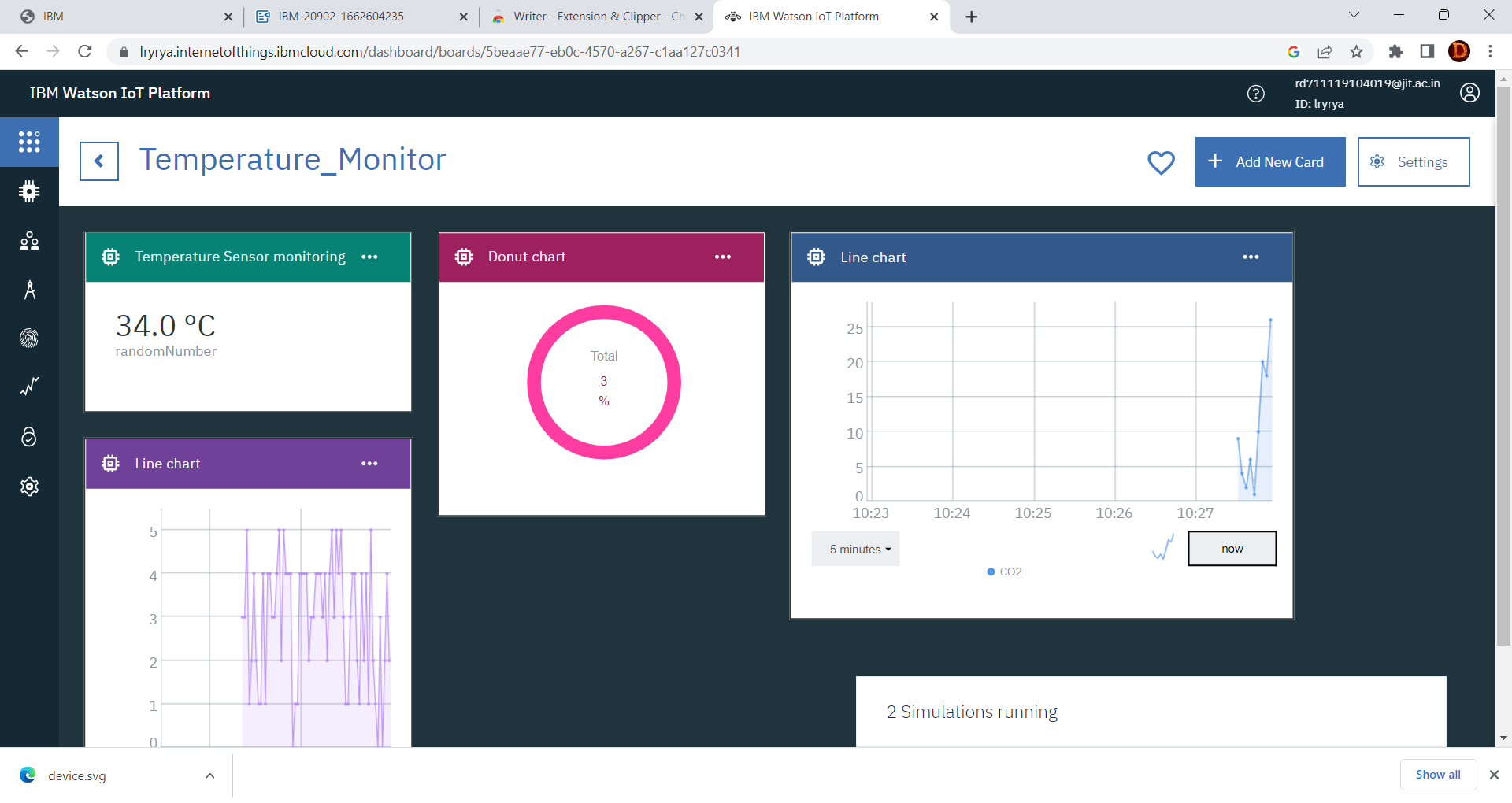
Commands are the mechanism by which applications communicate with devices. Only applications can send commands, and the commands are sent to specific devices. The device must determine which action to take on receipt of a command. Devices can be designed to listen for any command or to subscribe to a specified list of commands.Last even cache. For Lite plans, you can store information for a minimum of one day and a maximum of seven days. For other plans, you can store information for a minimum of one day and a maximum of 45 days.

**Service status**

To find out about the status and any upcoming planned service maintenance updates for Platform Service, go to the following service status page:

[http://status.internetofthings.ibmcloud.com](https://www.ibm.com/links?url=http%3A%2F%2Fstatus.internetofthings.ibmcloud.com)

Using the boards the simulated values can be displayed in different format



**7.2. Automated Alerting using Fast2sms**

**Fast2SMS**is a popular **bulk SMS service provider in India**. It was started in 21st July 2011. Due to its simplicity and ease of use it has become one of the mostly used SMS portals and has 2 million users.



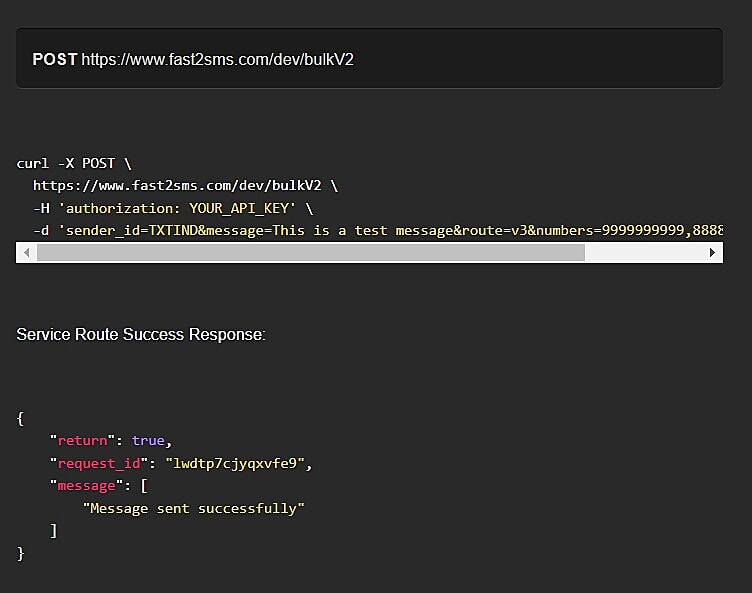
**Features of Fast2SMS**

**Bulk SMS** – Bulk SMS refers to business sending SMS to one or more recipients and can scale up to millions of persons at the same time. It refers to sending large number of messages to a predefined set of customers.

**Quick SMS feature** – Fast2SMS provides a very unique and useful feature which is not available in any other bulk SMS service provider. You can send SMS to DND and Non DND numbers even if you are not registered in the DLT portal.

**API SMS** – API refers to Application Programming Interface. **SMS API integration**is the fastest and simplest way to send automated messages directly from your platform. Fast2SMS provide **API for bulk SMS,** which ensures security and it is a very reliable source of sending data.

**Add contacts with QR** – Fast2SMS offers the facility of adding contacts with QR. QR stands for quick response. The main benefit of QR code is it takes very less space for storing information. It is a form of barcode and has become quite popular these days. In this feature you can add contacts by creating a QR code. Simply you need to add your group name and group URL name and then click on create QR. The QR code gets created



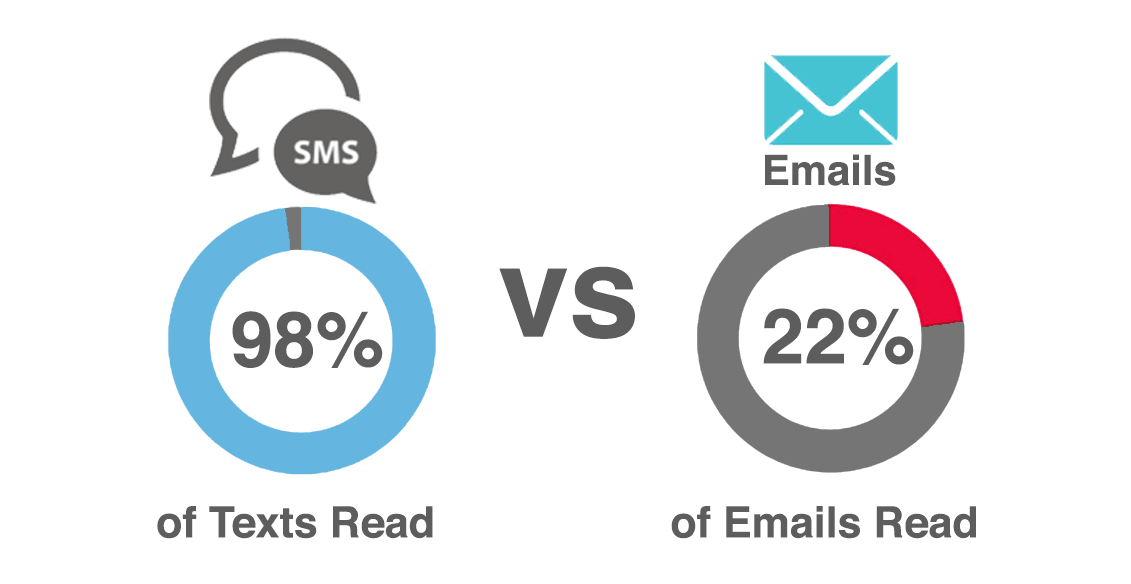
.

There is a lot of debate on the topic whether to go for bulk SMS or email marketing. Let’s look into some facts related to email and SMS:

* The average person receives 178 text messages in a month.
* And they receive approx. 1216 emails in a month.
* 98% of text messages get read in a month
* [22% of emails are read in a month.](https://medium.com/@andrewdjohns/believing-these-4-sms-marketing-myths-will-cost-you-money-1945ef076419)
* 91% of adults have their smartphone within arm’s reach.
* There are 6 billion active mobile phones in the world.
* 6 billion Email accounts are there in the world.
* According to the findings, 19% of people click the URL in the SMS.
* According to the findings, 2% of people click the URL sent in the email.

The picture is clear. SMS is a more powerful way to connect to the target audience and establish rapport with them. The open rate of an SMS is more than that of an email. As everybody of us might have noticed that as soon as a notification sounds comes in our mobile, we immediately grab our phone and check who has sent SMS and the details. This is why SMS has a high readability and is more likely to get opened.

In case of an email we postpone reading it as it is usually very long. The positive points of an SMS are that it is to the point, conveys essential information in a clear cut manner and does not require an internet connection to view it.



So we have decided to send the alert using bulk SMS service through fast2sms

**8.TESTING**

**8.1. Test Cases**

A test case is a document which has a set of conditions or actions that are performed on the software application in order to verify the expected functionality of the feature.

After test scripts, test cases are the second most detailed way of documenting testing work. They describe a specific idea that is to be tested, without detailing the exact steps to be taken or data to be used. For example, in a test case, you document something like ‘Test if coupons can be applied on actual price‘. This doesn’t mention how to apply the coupons or whether there are multiple ways to apply. It also doesn’t mention if the tester uses a link to apply a discount, or enter a code, or have a customer service apply it. They give flexibility to the tester to decide how they want to execute the test.

**Benefits of Writing Test Cases**

The key purpose of a test case is to ensure if different features within an application are working as expected. It helps tester, validate if the software is free of defects and if it is working as per the expectations of the end users. Other benefits of test cases include:

* + - Test cases ensure good test coverage
    - Help improve the quality of software,
    - Decreases the maintenance and software support costs
    - Help verify that the software meets the end user requirements
    - Allows the tester to think thoroughly and approach the tests from as many angles as possible
    - Test cases are reusable for the future – anyone can reference them and execute the test.

So, these are a few reasons why test cases are extremely useful in software testing. Test cases are powerful artifacts that work as a good source of truth for how a system and a particular feature of software works. However, before we deep dive into the lessons for writing top-notch test cases, let us have a basic idea on the terminologies associated with them.

**Test Case Format**

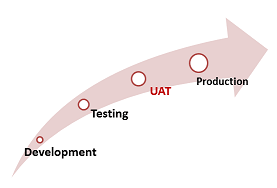
The primary ingredients of a test case are an ID, description, bunch of inputs, few actionable steps, as well as expected and actual results. Let’s learn what each of them is:

* + **Test Case Name:**A test case should have a name or title that is self-explanatory.
  + **Test Case Description:**The description should tell the tester what they’re going to test in brief.
  + **Pre-Conditions:**Any assumptions that apply to the test and any preconditions that must be met prior to the test being executed should be listed here.
  + **Test Case Steps:**The test steps should include the necessary data and information on how to execute the test. The steps should be clear and brief, without leaving out essential facts.
  + **Test Data:**It’s important to select a data set that gives sufficient coverage. Select a data set that specifies not only the positive scenarios but negative ones as well.
  + **Expected Result:**The expected results tell the tester what they should experience as a result of the test steps.
  + **Actual Result:**They specifies how the application actually behaved while test cases were being executed.
  + **Comments:**Any other useful information such as screenshots that tester want’s to specify can be included here.

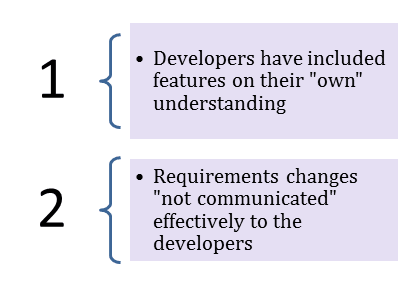
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case no | Test Data | Actions | Pass/Fail | Comments |
| 1 | Temperature: 65C  Gas  NO2:20%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:OFF  Fire Alarm:OFF  Alert:NOT SENT | Pass | Alert Not Sent |
| 2 | Temperature: 55C  Gas  NO2:20%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:OFF  Fire Alarm:OFF  Alert:NOT SENT | Pass | Alert Not Sent |
| 3 | Temperature: 85C  Gas  NO2:20%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:ON  Fire Alarm:ON  Alert: SENT | Pass | Alert Sent Successfully |
| 4 | Temperature: 35C  Gas  NO2:20%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:OFF  Fire Alarm:OFF  Alert:NOT SENT | Pass | Alert Not Sent |
| 5 | Temperature: 45C  Gas  NO2:20%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:ON  Fire Alarm:ON  Alert:SENT | Pass | Alert Sent  Successfully |
| 6 | Temperature: 75C  Gas  NO2:20%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:ON  Fire Alarm: ON  Alert: SENT | Pass | Alert Sent Successfully |
| 7 | Temperature: 15C  Gas  NO2:20%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:OFF  Fire Alarm:OFF  Alert:NOT SENT | Pass | Alert Not Sent |
| 8 | Temperature: 90C  Gas  NO2:20%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:ON  Fire Alarm:ON  Alert: SENT | Pass | Alert Sent  Successfully |
| 9 | Temperature: 25C  Gas  NO2:20%  CO2:10%  CO:2% | Exhaust Fan: OFF  Water Sprinkler:OFF  Fire Alarm:OFF  Alert:NOT SENT | Pass | Alert Not Sent |
| 10 | Temperature: 95C  Gas  NO2:90%  CO2:30%  CO:2% | Exhaust Fan: ON  Water Sprinkler:ON  Fire Alarm:ON  Alert: SENT | Pass | Alert Sent  Successfully |

**8.2. User Acceptance Testing**

**User Acceptance Testing (UAT)**is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.



**Purpose of UAT**



The main **Purpose of UAT**is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved.

UAT is performed by –

 Client

 End users

**Need of User Acceptance Testing**

**Need of User Acceptance Testing**arises once software has undergone Unit, Integration and System testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them, so for testing whether the final product is accepted by client/end-user, user acceptance testing is needed.

* Developers code software based on requirements document which is their “own” understanding of the requirements and **may not actually be what the client needs from the software**.
* Requirements changes during the course of the project may not be communicated effectively to the developers.

**Defect Analysis**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Resolution** | **Severity1** | **Severity2** | **Severity3** | **Severity4** | **Subtotal** |
| ByDesign | 5 | 2 | 2 | 1 | 10 |
| Duplicate | 1 | 0 | 2 | 1 | 4 |
| External | 2 | 3 | 0 | 1 | 6 |
| Fixed | 10 | 2 | 4 | 15 | 31 |
| NotReproduced | 0 | 0 | 1 | 0 | 1 |
| Skipped | 0 | 0 | 1 | 1 | 2 |
| Won'tFix | 0 | 5 | 2 | 1 | 8 |
| Totals | 18 | 12 | 12 | 20 | 52 |

**Test Case Analysis**

This reports hows the number of test cases that have passed, failed, and untested

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Total Cases** | **Not Tested** | **Fail** | **Pass** |
| Login Page | 6 | 0 | 0 | 6 |
| Node Red Dashboard | 35 | 0 | 0 | 35 |
| IBM Watson IOT Platform | 2 | 0 | 0 | 2 |
| Python | 3 | 0 | 0 | 3 |

**9. Result**

**9.1) Performance Metrics**

These metrics are used to track and measure the effectiveness and profitability of various projects.

Each stage of the project is tracked and measured against the goals that the project set out to achieve.

The data compiled from the metrics can be used to plan future projects and gives insight on how to make projects more efficient.

**10) Advantages and Disadvantages**

Advantages:

As far as fire alarm installers go, a wireless system is ideal because they are much easier to install. A wireless system essentially involves mounting the devices to the appropriate locations around a building or room, setting up the actual system and syncing it to WiFi. Compare this to a wired system, which requires fire alarm installers to connect the system to power supplies and ensure cables are connected properly.

Another great advantage of a wireless fire alarm system is it operates off of a battery. This frees up a wall outlet and you can feel safe knowing the system will still work in the event of a power outage. And adding a second or subsequent wireless device is easy if you add on to your home or office.

Disadvantages:

The one thing most fire alarm system inspectors caution against with wireless systems is having to replace the battery. The system is essentially useless if the batteries aren’t charged, since it won’t work properly. There is a bit of a burden to homeowners or business owners to always remember to keep the batteries fresh so the system operates properly when you need it most.

A couple other disadvantages fire alarm system inspectors point out is wireless systems have limited range and don’t have centralized monitoring. Range can be a problem for large offices or homes, since a weak wireless connection may cause the system to not operate reliably. Wireless fire alarm systems also don’t connect directly to the telephone lines, which are linked to the fire departments, so the response to an emergency could be slower as a result.

**11) Conclusion**

Thus this Fire management system play a very important role in Industries, Shops, Malls, Residential complexes, and parking areas. They help in detecting fire or harmful gas at an early stage and can help in saving lives. Commercial Fire detecting systems usually have an alarm signaling, with the help of a buzzer or Siren. We have designed an this system to send an immediate alert through the bulk sms service via fast2sms. These systems do not require any human interaction. In this Industrial fire management system using temperature, flame and gas sensors using the IOT project, we can send LIVE information like Temperature, gas and flame detected by a particular device to the Fire Department so the capital damage can be avoided

**12) Future Scope**

Although their name suggests the opposite of what they do, fireballs actually take the place of a traditional fire extinguisher, covering more space and doing it much faster. If you don’t believe it, you should check out the video in the link. Created by a company called Elide, the fireball can even fight fires when a user cannot be present to use it. As their website states, “When a fire occurs and no one is present, Fire Extinguishing Ball will self-activate when it comes into contact with fire and give a loud noise as a fire alarm. Because of this feature, it can be placed in a fire prone area such as near an electrical circuit breaker or in a kitchen.”

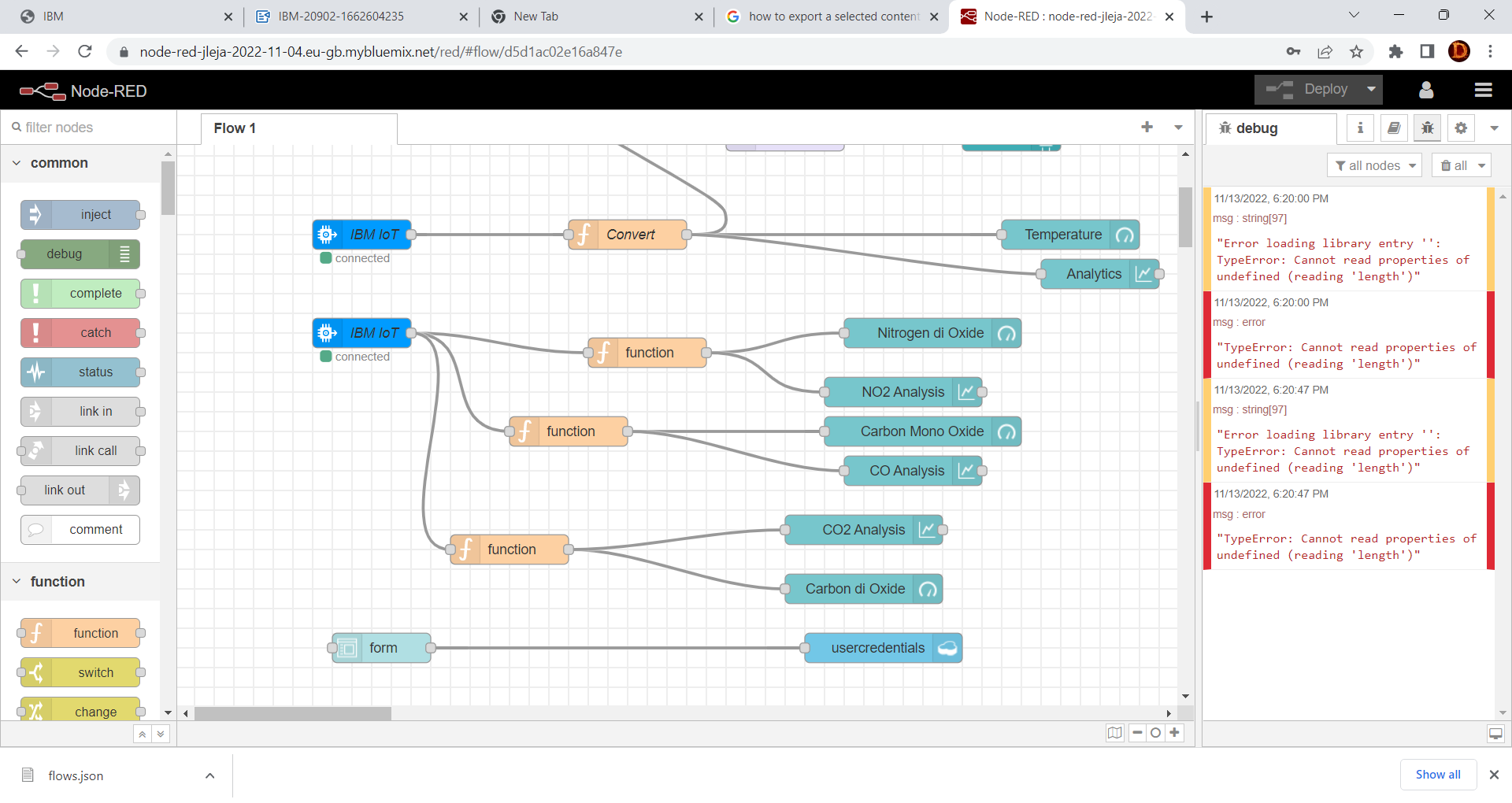
Perhaps most applicable to dealers looking to grow their RMR, wireless devices provide mobile capabilities to homeowners looking to install themselves, or even to take with them when relocating. According to firesystemsltd.co.uk, “Some of the systems on the market are using mesh network for the first time in wireless fire detection technology. The detectors are connected to each other and are using different frequencies on different bandwidths.” For those who look for something truly reliable in any situation, many devices can be connected in wired and non-wired formats. This dual connectivity provides unprecedented coverage and ultimate reliability. Yet, for buildings that are difficult to wire, or consumers who want something simple, wire-free systems will take the market by storm.

As new technologies emerge, dealers should be sure to leverage both timeless and emerging technologies to target more customers. Devices are becoming more and more capable, and regardless their application, they are all evolving to connect to the internet and cooperate with other devices. Consumers look for smart home technology along with commercial products that work in a cohesive environment. Along with investigating which products to sell, we encourage dealers to take a serious look at reliable partners to provide wholesale alarm monitoring services for devices so that consumers never go without protection.

**13)Appendix**

**Source Code**

**Node Red Flows**



**Flows.json**

[

{

"id": "d5d1ac02e16a847e",

"type": "tab",

"label": "Flow 1",

"disabled": false,

"info": "",

"env": []

},

{

"id": "1533d345e1dc36a9",

"type": "ui\_gauge",

"z": "d5d1ac02e16a847e",

"name": "",

"group": "b0b2466424d96a85",

"order": 1,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "Temperature",

"label": "Celsius",

"format": "{{value}}",

"min": 0,

"max": "100",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "40",

"seg2": "60",

"className": "",

"x": 910,

"y": 340,

"wires": []

},

{

"id": "cbd881198ad25821",

"type": "function",

"z": "d5d1ac02e16a847e",

"name": "Convert",

"func": "msg.payload=msg.payload.Temperature\nreturn msg;",

"outputs": 1,

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 460,

"y": 340,

"wires": [

[

"1533d345e1dc36a9",

"1cba1c03ecf32aa5",

"c883c624a3b90fc4"

]

]

},

{

"id": "ea14ed512305484e",

"type": "ibmiot in",

"z": "d5d1ac02e16a847e",

"authentication": "apiKey",

"apiKey": "8d9dadc21a8b3d2f",

"inputType": "evt",

"logicalInterface": "",

"ruleId": "",

"deviceId": "T1",

"applicationId": "",

"deviceType": "Temp",

"eventType": "+",

"commandType": "",

"format": "json",

"name": "IBM IoT",

"service": "registered",

"allDevices": "",

"allApplications": "",

"allDeviceTypes": "",

"allLogicalInterfaces": "",

"allEvents": true,

"allCommands": "",

"allFormats": "",

"qos": 0,

"x": 190,

"y": 340,

"wires": [

[

"cbd881198ad25821"

]

]

},

{

"id": "08daf8c27126b608",

"type": "ibmiot in",

"z": "d5d1ac02e16a847e",

"authentication": "apiKey",

"apiKey": "8d9dadc21a8b3d2f",

"inputType": "evt",

"logicalInterface": "",

"ruleId": "",

"deviceId": "G1",

"applicationId": "",

"deviceType": "Gas\_Sensor",

"eventType": "+",

"commandType": "",

"format": "json",

"name": "IBM IoT",

"service": "registered",

"allDevices": "",

"allApplications": "",

"allDeviceTypes": false,

"allLogicalInterfaces": "",

"allEvents": true,

"allCommands": "",

"allFormats": "",

"qos": 0,

"x": 190,

"y": 440,

"wires": [

[

"cce26a0bf1e99e72",

"d4be4869ed5d4f8c",

"9669c4a7a1ddc6de"

]

]

},

{

"id": "cce26a0bf1e99e72",

"type": "function",

"z": "d5d1ac02e16a847e",

"name": "",

"func": "msg.payload=msg.payload.NO2\nreturn msg;",

"outputs": 1,

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 480,

"y": 460,

"wires": [

[

"152e5e18a22ea640",

"c0b625de4d3f7730"

]

]

},

{

"id": "d4be4869ed5d4f8c",

"type": "function",

"z": "d5d1ac02e16a847e",

"name": "",

"func": "msg.payload=msg.payload.CO\nreturn msg;",

"outputs": 1,

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 400,

"y": 540,

"wires": [

[

"574dedc96208c0aa",

"f3dd2fc7f39bf5f7"

]

]

},

{

"id": "9669c4a7a1ddc6de",

"type": "function",

"z": "d5d1ac02e16a847e",

"name": "",

"func": "msg.payload=msg.payload.CO2\nreturn msg;",

"outputs": 1,

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 340,

"y": 660,

"wires": [

[

"8dae8fa1a7ea7415",

"763a9ae51ae799ec"

]

]

},

{

"id": "152e5e18a22ea640",

"type": "ui\_gauge",

"z": "d5d1ac02e16a847e",

"name": "",

"group": "d2d6e984eb71a537",

"order": 1,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "Nitrogen di Oxide",

"label": "Percentage",

"format": "{{value}}",

"min": 0,

"max": "100",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "20",

"seg2": "25",

"className": "",

"x": 770,

"y": 440,

"wires": []

},

{

"id": "574dedc96208c0aa",

"type": "ui\_gauge",

"z": "d5d1ac02e16a847e",

"name": "",

"group": "d2d6e984eb71a537",

"order": 2,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "Carbon Mono Oxide",

"label": "percentage",

"format": "{{value}}",

"min": 0,

"max": "100",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "3",

"seg2": "5",

"className": "",

"x": 760,

"y": 540,

"wires": []

},

{

"id": "8dae8fa1a7ea7415",

"type": "ui\_gauge",

"z": "d5d1ac02e16a847e",

"name": "",

"group": "d2d6e984eb71a537",

"order": 3,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "Carbon di Oxide",

"label": "Percentage",

"format": "{{value}}",

"min": 0,

"max": "100",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "20",

"seg2": "30",

"className": "",

"x": 700,

"y": 700,

"wires": []

},

{

"id": "8a41db8811f572d9",

"type": "ui\_form",

"z": "d5d1ac02e16a847e",

"name": "",

"label": "",

"group": "c8492bdcd68680a2",

"order": 0,

"width": 0,

"height": 0,

"options": [

{

"label": "Username",

"value": "Username",

"type": "text",

"required": true,

"rows": null

},

{

"label": "Email",

"value": "Email",

"type": "email",

"required": true,

"rows": null

},

{

"label": "Phone Number",

"value": "Phonenumber",

"type": "number",

"required": true,

"rows": null

}

],

"formValue": {

"Username": "",

"Email": "",

"Phonenumber": ""

},

"payload": "",

"submit": "submit",

"cancel": "cancel",

"topic": "topic",

"topicType": "msg",

"splitLayout": "",

"className": "",

"x": 210,

"y": 760,

"wires": [

[

"40123f46207d3c68"

]

]

},

{

"id": "40123f46207d3c68",

"type": "cloudant out",

"z": "d5d1ac02e16a847e",

"name": "",

"cloudant": "3d10849a8821b6a7",

"database": "usercredentials",

"service": "\_ext\_",

"payonly": true,

"operation": "insert",

"x": 720,

"y": 760,

"wires": []

},

{

"id": "7c787911c08c4fcb",

"type": "inject",

"z": "d5d1ac02e16a847e",

"name": "",

"props": [

{

"p": "payload"

},

{

"p": "topic",

"vt": "str"

}

],

"repeat": "",

"crontab": "",

"once": false,

"onceDelay": 0.1,

"topic": "",

"payload": "",

"payloadType": "date",

"x": 160,

"y": 860,

"wires": [

[

"ab2c14742c0e6cb4"

]

]

},

{

"id": "ab2c14742c0e6cb4",

"type": "cloudant in",

"z": "d5d1ac02e16a847e",

"name": "",

"cloudant": "3d10849a8821b6a7",

"database": "usercredentials",

"service": "\_ext\_",

"search": "\_all\_",

"design": "Usercredentials",

"index": "Phonenumber",

"x": 460,

"y": 860,

"wires": [

[

"5cd6db4b75445da3"

]

]

},

{

"id": "5cd6db4b75445da3",

"type": "debug",

"z": "d5d1ac02e16a847e",

"name": "",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"statusVal": "",

"statusType": "auto",

"x": 690,

"y": 920,

"wires": []

},

{

"id": "1d8c19444aee18bf",

"type": "debug",

"z": "d5d1ac02e16a847e",

"name": "",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "false",

"statusVal": "",

"statusType": "auto",

"x": 850,

"y": 20,

"wires": []

},

{

"id": "057ef750bef3fc86",

"type": "inject",

"z": "d5d1ac02e16a847e",

"name": "",

"props": [

{

"p": "payload"

},

{

"p": "topic",

"vt": "str"

}

],

"repeat": "",

"crontab": "",

"once": false,

"onceDelay": 0.1,

"topic": "",

"payload": "",

"payloadType": "date",

"x": 180,

"y": 20,

"wires": [

[

"3c36398fb917a7c9"

]

]

},

{

"id": "3c36398fb917a7c9",

"type": "python-function",

"z": "d5d1ac02e16a847e",

"name": "",

"func": "import requests\nurl = \"https://www.fast2sms.com/dev/bulkV2\"\npayload = \"sender\_id=FSTSMS&message=Alert!&language=english&route=p&numbers=9095057479\"\nheaders = {\n'authorization': \"o0azwVFNHOM5B3hrRxdenyU2cfZujqSpYEX7t8LAgJPb9kliWCugDvo1n0kcY8TGHOt3dIQwsKpLbAJU\",\n'Content-Type': \"application/x-www-form-urlencoded\",\n'Cache-Control': \"no-cache\",\n}\nresponse = requests.request(\"POST\", url, data=payload, headers=headers)\nprint(response.text)\n",

"outputs": 1,

"x": 580,

"y": 20,

"wires": [

[

"1d8c19444aee18bf"

]

]

},

{

"id": "cec01e40a908c48e",

"type": "ui\_led",

"z": "d5d1ac02e16a847e",

"order": 0,

"group": "b0b2466424d96a85",

"width": 0,

"height": 0,

"label": "Water Sprinkler",

"labelPlacement": "left",

"labelAlignment": "left",

"colorForValue": [

{

"color": "#ff0000",

"value": "false",

"valueType": "bool"

},

{

"color": "#008000",

"value": "true",

"valueType": "bool"

}

],

"allowColorForValueInMessage": false,

"shape": "circle",

"showGlow": true,

"name": "",

"x": 850,

"y": 120,

"wires": []

},

{

"id": "1cba1c03ecf32aa5",

"type": "function",

"z": "d5d1ac02e16a847e",

"name": "",

"func": "msg.payload=(msg.payload>65)? true: false;\nreturn msg;",

"outputs": 1,

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 380,

"y": 120,

"wires": [

[

"80a720021c12cce7",

"cebd50b2ed69ccd3",

"56b6fd723b216743"

]

]

},

{

"id": "80a720021c12cce7",

"type": "delay",

"z": "d5d1ac02e16a847e",

"name": "",

"pauseType": "delay",

"timeout": "45",

"timeoutUnits": "seconds",

"rate": "1",

"nbRateUnits": "1",

"rateUnits": "second",

"randomFirst": "1",

"randomLast": "5",

"randomUnits": "seconds",

"drop": false,

"allowrate": false,

"outputs": 1,

"x": 620,

"y": 120,

"wires": [

[

"cec01e40a908c48e"

]

]

},

{

"id": "cebd50b2ed69ccd3",

"type": "delay",

"z": "d5d1ac02e16a847e",

"name": "",

"pauseType": "delay",

"timeout": "30",

"timeoutUnits": "seconds",

"rate": "1",

"nbRateUnits": "1",

"rateUnits": "second",

"randomFirst": "1",

"randomLast": "5",

"randomUnits": "seconds",

"drop": false,

"allowrate": false,

"outputs": 1,

"x": 620,

"y": 180,

"wires": [

[

"1b0bc39e91d2d4dc"

]

]

},

{

"id": "56b6fd723b216743",

"type": "delay",

"z": "d5d1ac02e16a847e",

"name": "",

"pauseType": "delay",

"timeout": "10",

"timeoutUnits": "seconds",

"rate": "1",

"nbRateUnits": "1",

"rateUnits": "second",

"randomFirst": "1",

"randomLast": "5",

"randomUnits": "seconds",

"drop": false,

"allowrate": false,

"outputs": 1,

"x": 620,

"y": 240,

"wires": [

[

"dc9bb5498dd98dfc"

]

]

},

{

"id": "1b0bc39e91d2d4dc",

"type": "ui\_led",

"z": "d5d1ac02e16a847e",

"order": 1,

"group": "b0b2466424d96a85",

"width": 0,

"height": 0,

"label": "Exhaust fan",

"labelPlacement": "left",

"labelAlignment": "left",

"colorForValue": [

{

"color": "#ff0000",

"value": "false",

"valueType": "bool"

},

{

"color": "#008000",

"value": "true",

"valueType": "bool"

}

],

"allowColorForValueInMessage": false,

"shape": "circle",

"showGlow": true,

"name": "",

"x": 850,

"y": 180,

"wires": []

},

{

"id": "dc9bb5498dd98dfc",

"type": "ui\_led",

"z": "d5d1ac02e16a847e",

"order": 2,

"group": "b0b2466424d96a85",

"width": 0,

"height": 0,

"label": "Fire Alarm",

"labelPlacement": "left",

"labelAlignment": "left",

"colorForValue": [

{

"color": "#ff0000",

"value": "false",

"valueType": "bool"

},

{

"color": "#008000",

"value": "true",

"valueType": "bool"

}

],

"allowColorForValueInMessage": false,

"shape": "circle",

"showGlow": true,

"name": "",

"x": 850,

"y": 240,

"wires": []

},

{

"id": "c883c624a3b90fc4",

"type": "ui\_chart",

"z": "d5d1ac02e16a847e",

"name": "",

"group": "b0b2466424d96a85",

"order": 0,

"width": 0,

"height": 0,

"label": "Analytics",

"chartType": "line",

"legend": "false",

"xformat": "auto",

"interpolate": "linear",

"nodata": "Waiting for data",

"dot": false,

"ymin": "0",

"ymax": "100",

"removeOlder": "60",

"removeOlderPoints": "",

"removeOlderUnit": "1",

"cutout": 0,

"useOneColor": false,

"useUTC": false,

"colors": [

"#42b0ff",

"#aec7e8",

"#ff7f0e",

"#1adb1a",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 940,

"y": 380,

"wires": [

[]

]

},

{

"id": "763a9ae51ae799ec",

"type": "ui\_chart",

"z": "d5d1ac02e16a847e",

"name": "",

"group": "5a710181faf582af",

"order": 5,

"width": 0,

"height": 0,

"label": "CO2 Analysis",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "0",

"ymax": "100",

"removeOlder": "60",

"removeOlderPoints": "",

"removeOlderUnit": "1",

"cutout": 0,

"useOneColor": false,

"useUTC": false,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca02c",

"#98df8a",

"#df2a2a",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 700,

"y": 640,

"wires": [

[]

]

},

{

"id": "c0b625de4d3f7730",

"type": "ui\_chart",

"z": "d5d1ac02e16a847e",

"name": "",

"group": "5a710181faf582af",

"order": 3,

"width": 0,

"height": 0,

"label": "NO2 Analysis",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "0",

"ymax": "100",

"removeOlder": "60",

"removeOlderPoints": "",

"removeOlderUnit": "1",

"cutout": 0,

"useOneColor": false,

"useUTC": false,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca02c",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 740,

"y": 500,

"wires": [

[]

]

},

{

"id": "f3dd2fc7f39bf5f7",

"type": "ui\_chart",

"z": "d5d1ac02e16a847e",

"name": "",

"group": "5a710181faf582af",

"order": 4,

"width": 0,

"height": 0,

"label": "CO Analysis",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "0",

"ymax": "100",

"removeOlder": "60",

"removeOlderPoints": "",

"removeOlderUnit": "1",

"cutout": 0,

"useOneColor": false,

"useUTC": false,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca52c",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 750,

"y": 580,

"wires": [

[]

]

},

{

"id": "b0b2466424d96a85",

"type": "ui\_group",

"name": "Temperature",

"tab": "477494a5baf03dbe",

"order": 3,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "8d9dadc21a8b3d2f",

"type": "ibmiot",

"name": "IBM\_IOT",

"keepalive": "60",

"serverName": "lryrya.messaging.internetofthings.ibmcloud.com",

"cleansession": true,

"appId": "",

"shared": false

},

{

"id": "d2d6e984eb71a537",

"type": "ui\_group",

"name": "Gas Sensor",

"tab": "e208207b5344b7ac",

"order": 2,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "c8492bdcd68680a2",

"type": "ui\_group",

"name": "Register Form",

"tab": "94803c79c48f4a14",

"order": 3,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "3d10849a8821b6a7",

"type": "cloudant",

"host": "https://apikey-v2-2uxme1xe4ze15rnf5fh8l40eg4coz66pm8ateuwy93py:0172158934b6c3a3e8b39b0511e7af20@e8e96f53-33f3-4fd1-bb73-6fb98205f7c2-bluemix.cloudantnosqldb.appdomain.cloud",

"name": ""

},

{

"id": "5a710181faf582af",

"type": "ui\_group",

"name": "Gas Analytics",

"tab": "e208207b5344b7ac",

"order": 3,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "477494a5baf03dbe",

"type": "ui\_tab",

"name": "Temperature",

"icon": "dashboard",

"disabled": false,

"hidden": false

},

{

"id": "e208207b5344b7ac",

"type": "ui\_tab",

"name": "Gas Sensor",

"icon": "dashboard",

"disabled": false,

"hidden": false

},

{

"id": "94803c79c48f4a14",

"type": "ui\_tab",

"name": "Register Form",

"icon": "dashboard",

"disabled": false,

"hidden": false

}

]

**Temperature.py**

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

import ibmiotf.api

import requests

import json

#Provide your IBM Watson Device Credentials

organization = "lryrya"

deviceType = "Temp"

deviceId = "T1"

authMethod = "token"

authToken = "123456789"

def Alert():

url = "https://www.fast2sms.com/dev/bulkV2"

my\_data = {

'sender\_id': 'FSTSMS',

'message': 'Alert Fire has been detected in XYZ industry! Kindly Evacuvate from the place',

'language': 'english',

'route': 'p',

'numbers': '9095057479'

}

headers = {

'authorization': 'o0azwVFNHOM5B3hrRxdenyU2cfZujqSpYEX7t8LAgJPb9kliWCugDvo1n0kcY8TGHOt3dIQwsKpLbAJU',

'Content-Type': "application/x-www-form-urlencoded",

'Cache-Control': "no-cache"

}

response = requests.request("POST",

url,

data = my\_data,

headers = headers)

returned\_msg = json.loads(response.text)

print(returned\_msg['message'])

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data['command'])

status=cmd.data['command']

if status=="lighton":

print ("led is on")

else :

print ("led is off")

#print(cmd)

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

#..............................................

except Exception as e:

print("Caught exception connecting device: %s" % str(e))

sys.exit()

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

deviceCli.connect()

while True:

#Get Sensor Data from DHT11

temp=random.randint(0,70)

data = { 'Temperature' : temp}

#print data

def myOnPublishCallback():

print ("Published Temperature = %s C" %temp, "to IBM Watson")

if(temp>65):

Alert();

time.sleep(1)

success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

print("Not connected to IoTF")

time.sleep(1)

deviceCli.disconnect()

**Gas Sensor.py**

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

import ibmiotf.api

import requests

import json

#Provide your IBM Watson Device Credentials

organization = "lryrya"

deviceType = "Gas\_Sensor"

deviceId = "G1"

authMethod = "token"

authToken = "123456789"

def Alert():

url = "https://www.fast2sms.com/dev/bulkV2"

my\_data = {

'sender\_id': 'FSTSMS',

'message': 'Alert Some abnormality is found is XYZ industry! Kindly Evacuvate from the place',

'language': 'english',

'route': 'p',

'numbers': '9095057479'

}

headers = {

'authorization': 'o0azwVFNHOM5B3hrRxdenyU2cfZujqSpYEX7t8LAgJPb9kliWCugDvo1n0kcY8TGHOt3dIQwsKpLbAJU',

'Content-Type': "application/x-www-form-urlencoded",

'Cache-Control': "no-cache"

}

response = requests.request("POST",

url,

data = my\_data,

headers = headers)

returned\_msg = json.loads(response.text)

print(returned\_msg['message'])

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data['command'])

status=cmd.data['command']

if status=="lighton":

print ("led is on")

else :

print ("led is off")

#print(cmd)

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

#..............................................

except Exception as e:

print("Caught exception connecting device: %s" % str(e))

sys.exit()

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

deviceCli.connect()

while True:

#Get Sensor Data from DHT11

NO2=random.randint(0,30)

CO2=random.randint(0,25)

CO=random.randint(0,5)

data = { 'NO2' : NO2 , 'CO2': CO2,'CO' : CO}

#print data

def myOnPublishCallback():

print ("Published Nitrogen di Oxide = %s %" % NO2, "Carbon di oxide = %s %%" % CO2, "Carbon monoxide = %s %%" % CO, "to IBM Watson")

if(NO2>29) or (CO2>24) or (CO>4):

Alert();

time.sleep(1)

success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

print("Not connected to IoTF")

time.sleep(1)

deviceCli.disconnect()

Github Repositry:

https://github.com/IBM-EPBL/IBM-Project-20902-1659766144

Project Demo:

https://youtu.be/QzRIlRW0nvo