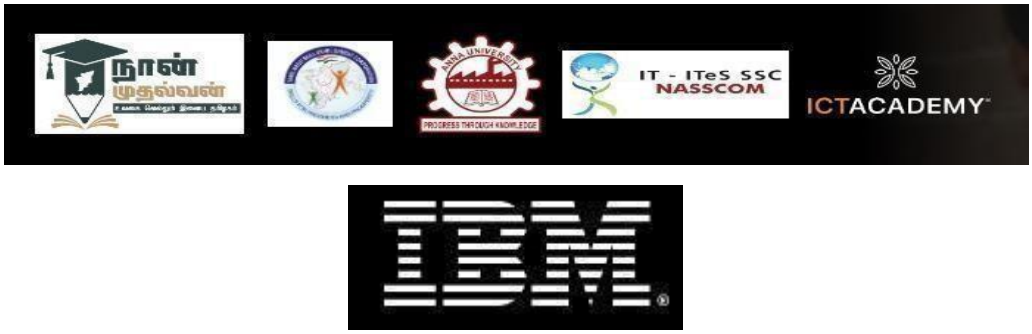



# SIGNS WITH SMART CONNECTIVITY FOR BETTER ROAD SAFETY

## IBM PROJECT REPORT

## Team ID - PNT2022TMID29355

SUBMITTED BY

VIHA ASWINESRIEE SMR - 422519106053
RAAGAVI SELVAKUMAR - 422519106033
PAVITHRA. V - 422519106027
HABINAYA. R - 422519106014

# CONTENTS


1. **INTRODUCTION** 4

1.1 Project Overview

1.2 Purpose

2. **LITERATURE SURVEY** 5

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. **IDEATION & PROPOSED SOLUTION** 6

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. **REQUIREMENT ANALYSIS** 15

4.1 Functional requirement

4.2 Non-Functional requirements

5. **PROJECT DESIGN** 17

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. **PROJECT PLANNING & SCHEDULING** 23

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule
6.3 Reports from JIRA

7. **CODING & SOLUTIONING** (Explain the features added in the project along with code) **27**
7.1 Feature 1
7.2 Feature 2
7.3 Feature 3

8. **TESTING** **29**
8.1 Test Cases
8.2 User Acceptance Testing

9. **RESULTS** **30**
9.1 Performance Metrics

10. **ADVANTAGES & DISADVANTAGES** **30**

11. **CONCLUSION** **31**

12. **FUTURE SCOPE** **31**

13. **APPENDIX** **32**

Source Code
GitHub & Project Demo Link **47**

# 1. INTRODUCTION

## 1.1 Project Overview

- To replace the static signboards, smart connected signboards are used.
- These smart connected sign boards get the speed limitations from a web app using weather API and update automatically.
- Based on the weather changes the speed may increase or decrease.
- Based on the traffic and fatal situations the diversion signs are displayed.
- Guide (Schools), Warning and Service (Hospitals, Restaurants) signs are also displayed accordingly.
- Different modes of operations can be selected with the help of buttons.

## 1.2 Purpose

- Smart Traffic Management is a system to monitor and control traffic signals using sensors to regulate the flow of traffic and to avoid congestion for a smooth flow of traffic.

- Prioritizing traffic like ambulances, police etc. is also one application comes under smart traffic management.

# 2. LITERATURE SURVEY

## 2.1 Existing problem: -

- Analysis of crash data has suggested a link between roadside advertising signs and safety.

- Research suggests that crash risk increases by approximately 25–29% in the presence of digital roadside advertising signs compared to control areas.

- On the other hand, static roadside advertising signs have not been linked with differences in the crash count.

- However, this finding is contrary to previous research that suggests differences in crash counts exist in the presence of static roadside advertising.

- The quantity and quality of available evidence limit our conclusion.
- Fixed object, side swipe and rear end crashes are the most common types of crashes in the presence of roadside advertising signs.
- In addition, drivers showed increased eye fixations and increased drifting between lanes on the road.

## 2.2 References:-

- Cairney and Gunathilake, 2000; Sisiopiku et al., 2015
- Islam, 2015; Sisiopiku et al., 2015
- Yannis et al., 2013, Staffeld (1953) and Ady (1967)

## 2.3 Problem Statement Definition:-

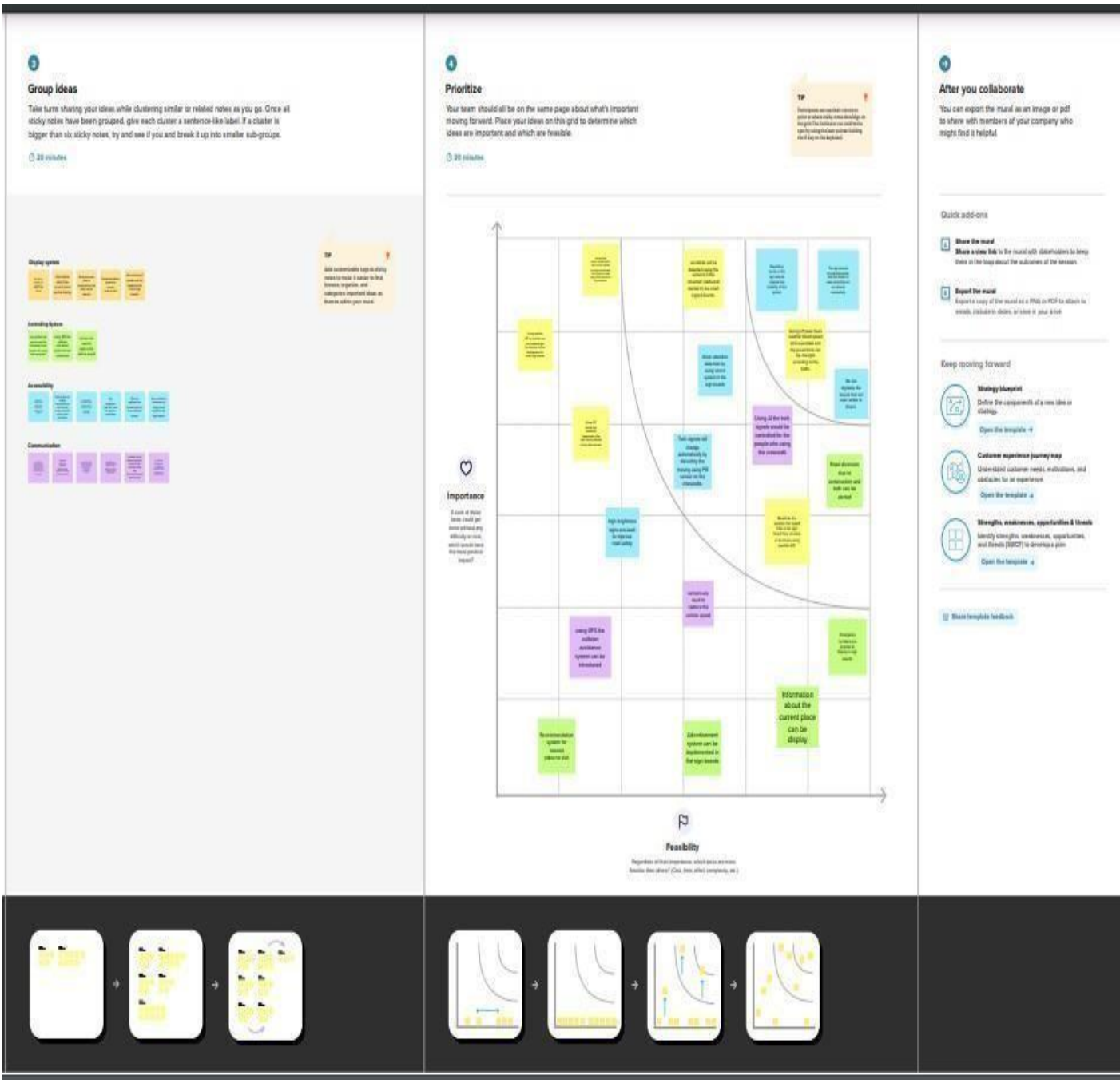
This project will replace the static boards to smart signed boards that will change the speed limits according to the weather climate and show diversion messages if there are accidents in the road and alert messages if there is hospital, schools or any roadworks.

# 3. IDEATION AND PROPOSED SOLUTION

## 3.1 Empathy Map Canvas:-


# 3.2 Ideation & Brainstorming Map :-


Brainstorm
& idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
Before you collaborate
Define your problem statement
Brainstorm
Key rules of brainstorming
Need some inspiration?


Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
Display system
Controlling System
Accessibility
Communication
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible
20 minutes
TIP
Importance
Feasibility
Information about the current place can be display
After you collaborate
You can export the mural as an image or pdf to share with members of your company who might find it helpful.
Quick add-ons
Share the mural
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
Export the mural
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.
Keep moving forward
Strategy blueprint
Define the components of a new idea or strategy.
Open the template
Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template
Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template
Share template feedback

# 3.3 Proposed Solution:-

| S.No. | Parameter | Description |
|---|---|---|
| 1. | **Problem Statement (Problem to be solved)** | To replace the static signboards, smart connected sign boards are used.<br><br>These smart connected sign boards get the speed limitations from a web app using weather API and update automatically.<br><br>Based on the weather changes the speed may increase or decrease<br><br>Based on the traffic and fatal situations the diversion signs are displayed.<br><br>Guide (Schools), Warning and Service (Hospitals, Restaurant) signs are also displayed accordingly.<br><br>Different modes of operations can be selected with the help of buttons. |

| | | |
|---|---|---|
| 2. | **Idea/solution description** | The weather and temperature details are obtained from the OpenWeatherMap API. Using these details, the speed limit will be updated automatically in accordance with the weather conditions. Also, the details regarding any accidents and traffic congestion faced on the particular road are obtained. Based on this, the traffic is diverted followed by a change in map path and the traffic is cleared. So, in the traffic sign board, some buttons will be placed which will be used to make it generic; where each button will be given a functionality such as changing the warning signs, which are predefined and separate signs will be present for both school and |

| S.No. | Parameter | Description |
|---|---|---|
| | | hospital zones. By activating this button, either through the web application or the physical buttons, sign of the board can be changed accordingly, and the speed limit will also be set depending upon the zones. Also, the pedestrians are given an option to change the traffic signs if they want to cross the road. If the pedestrian presses the button that is present on the post at the end of the road, then the traffic will be analyzed immediately. Accordingly, the sign of the traffic signal will be changed. This in turn reduces the frequent changing of the traffic signs even if the pedestrians are not present. |

| | | |
|---|---|---|
| 3. | **Novelty /Uniqueness** | Generic Sign board for all applications that uses both buttons and web service for updation.<br>Pedestrians are given the access to request the sign change of the signal to cross the road |

| | | |
|---|---|---|
| 4. | **Customer Satisfaction** | Diversion reasons will be displayed<br>If there is no traffic, pedestrians can cross the street without waiting.<br>Customer can reach the destination before the expected time |
| 5. | **Business Model** | Since APIs are used to actively monitor the customer's environment, this project employs a business strategy in which revenue will be generated on the basis of the length of time in which the customers actively interact with the product.<br>This product is aimed to be free of cost to the public, but the revenue will be generated by selling this product to the government at a low cost, so there will be less accidents and the public will be aware of the discrepancies or accidents in the particular road. The public will also gain all the information about the road, even if they are checking for an alternate path because of some mishaps that happen on the roads and these functionalities will increase the value of the product in the global market. |

| | | |
|---|---|---|
| 6. | **Scalability of the Solution** | In the future, if any update is required either on the hardware or software side, it can be easily implemented. The hardware components can be directly interfaced with the microcontroller and small modifications can be made in the programming of the existing product. In case of the software, the website application has to be updated with the additional functionality by creating a new section for the updated hardware. So, this will not affect the existing functionality of the product and new functionality can be easily integrated. In addition, a separate circuit will be kept along with the hardware to detect any problem which informs the web application. Also, a notification will be sent to the product service department. |

# 3.4 Problem Solution fit:-


# 4. REQUIREMENT ANALYSIS

# 4.1 Functional Requirements:-

Following are the functional requirements of the proposed solution.

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-I | **User Visibility** | Sign boards should be made with LED's which are bright coloured and are capable of attracting the drivers attention but it should also not be too distracting or blinding cause it may lead to accidents. |
| FR-2 | **User Need** | The smart sign boards should be placed frequently in places it is needed and less in places where it is not needed much to avoid confusion for the user during travel . |
| FR-3 | **User Understanding** | For better understanding of the driver the signs should be big , clear and legible and it can also include illustrations which will make it easily understandable to the driver. |
| FR-4 | **User Convenience** | The display should be big enough that it should even be visible from far distance clearly |

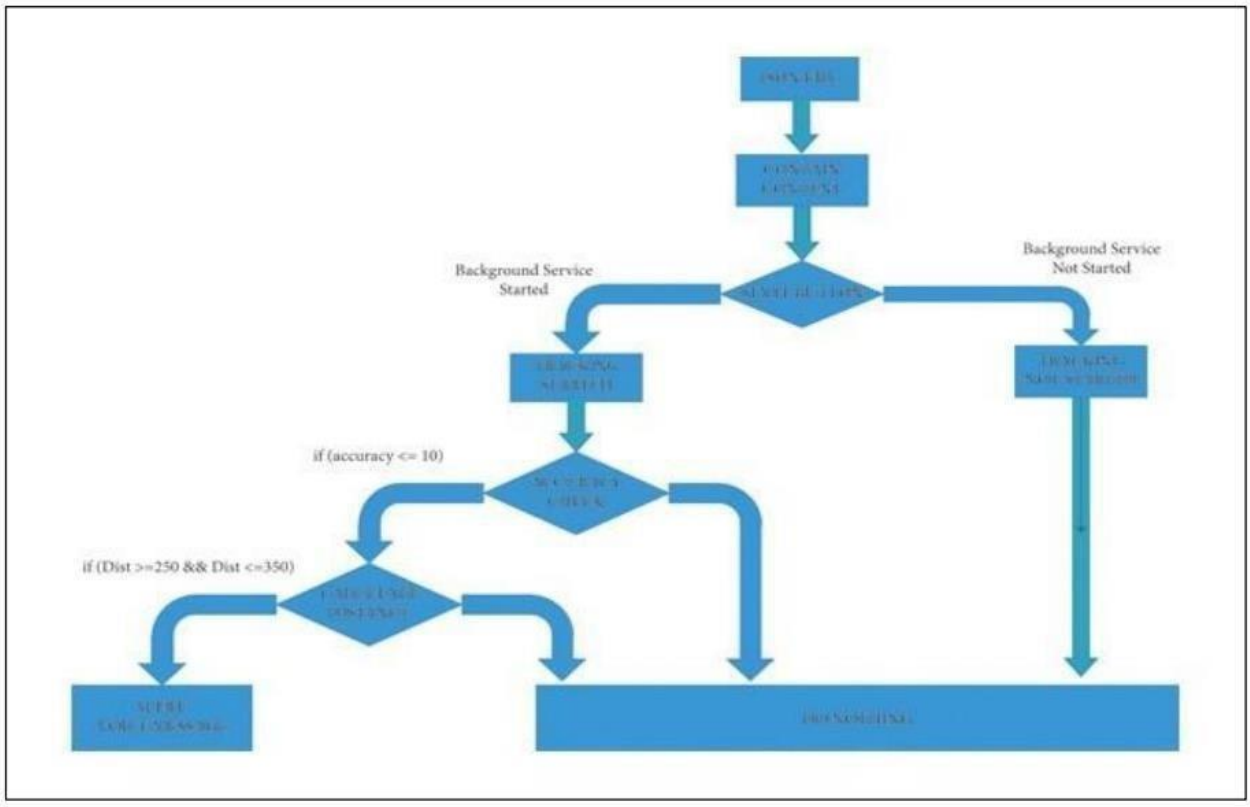
# 4.2 Non-Functional :-

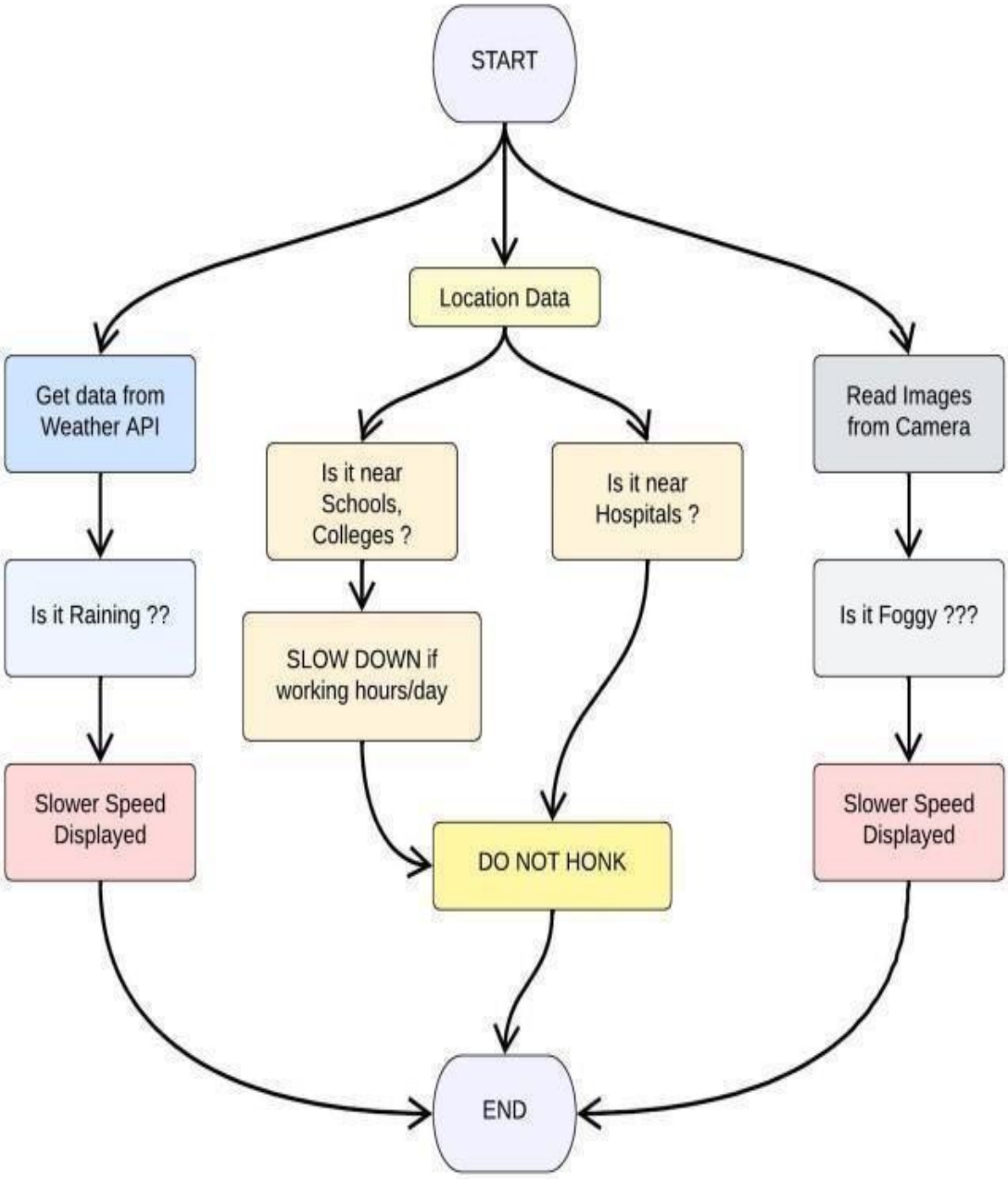
Following are the non-functional requirements of the proposed solution.

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-I | **Usability** | It should be able to upgrade and update when there is a need for it. |
| NFR-2 | **Security** | It should have good security system so that no other person is able to hack and display their own direction. |
| NFR-3 | **Reliability** | It should be able to display to information correctly and error – free. |
| NFR-4 | **Performance** | It should be able to automatically update itself when certain weather or traffic problem occurs. |
| NFR-5 | **Availability** | It should be available 24/7 so that it can be beneficial to the customer i.e., the driver. |
| NFR-6 | **Scalability** | It should be able to easily change and upgrade according to change and need in requirement. |

# 5.PROJECT DESIGN; -

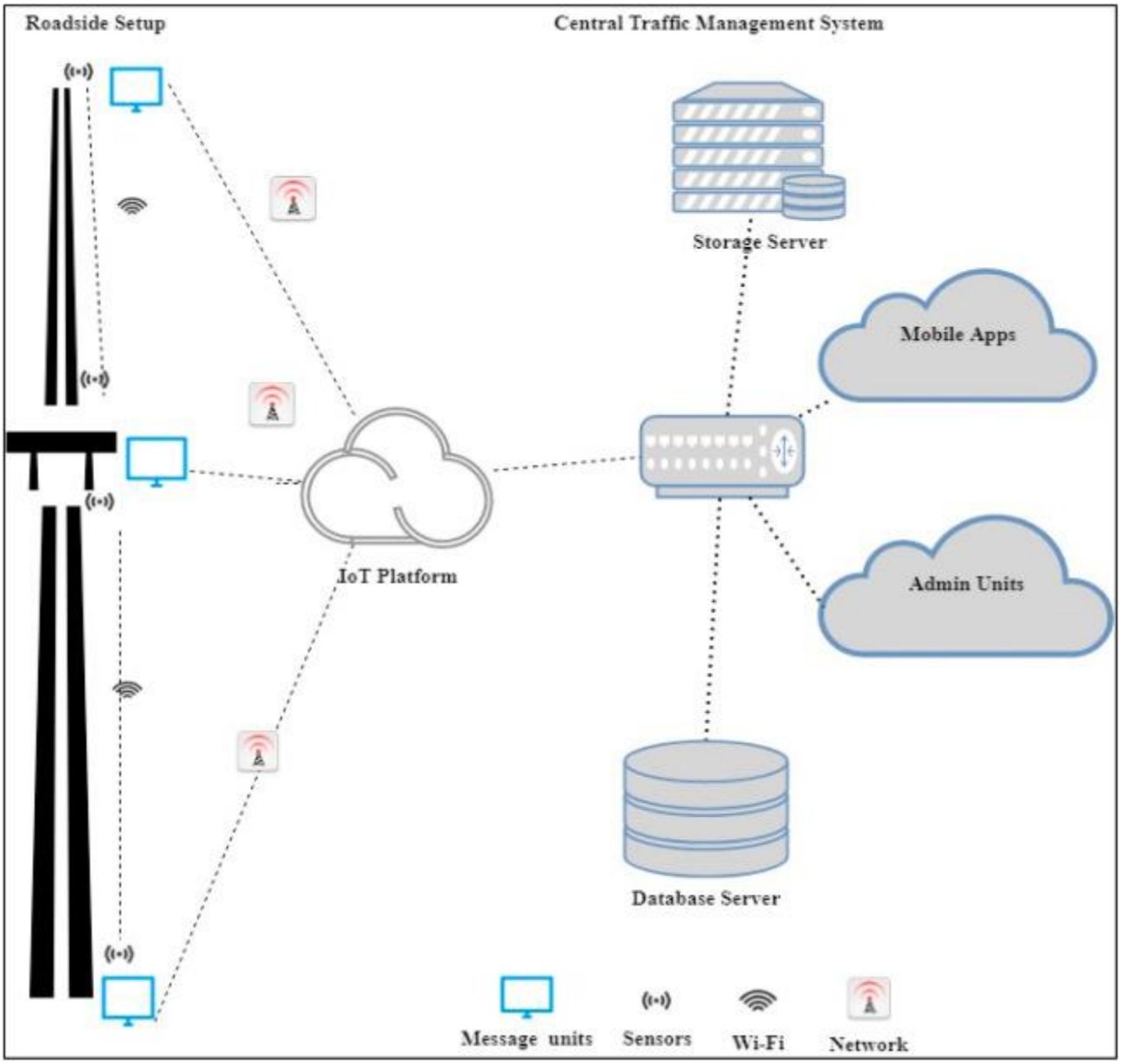
# 5.1 Data Flow Diagram






# 5.2 Solution & Technical Architecture:-

Following is the Technical Architecture with slight change and is **without the implementation of OpenCV API**.




# Following is the Solution Built:-

Table-1: Components & Technologies:

| S.No | Component | Description | Technology |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1. | User Interface | User can interact with the app using MIT App | HTML, CSS, JavaScript / Angular Js / React Js |
| 2. | Application Logic-1 | Detecting traffic and displaying information | Java / Python |
| 3. | Application Logic-2 | Sending notification to nearby users | IBM Watson STT service |
| 4. | Application Logic-3 | Communicating with physical device | IBM Watson Assistant |
| 5. | Database | Storing on local phone storage | IBM Cloud |
| 6. | Cloud Database | Database Service on Cloud | IBM DB2, IBM Cloudant etc. |
| 7. | File Storage | Storing on local phone storage | IBM Block Storage or Other Storage Service or Local Filesystem |
| 8. | External API-1 | Purpose of External API used in the application | Open Weather Map API |
| 9. | External API-2 | Purpose of External API used in the application | IBM Watson Platform, Node - Red |
| 10. | Infrastructure (Server / Cloud) | Application Deployment on Local System / Cloud Local Server Configuration:<br>Cloud Server Configuration: | Local, Cloud Foundry, Kubernetes |

## Table-2: Application Characteristics:

| S. No | Characteristics | Description | Technology |
|---|---|---|---|
| 1. | **Open-Source Frameworks** | *OpenWeatherMap*, *NODERED*, *IBM WATSON*, *MIT App Inventor* | IoT, internet |
| 2. | **Security Implementations** | Private cloud, Limited database access, Security layer implementation, Private key to access physical device | e.g. SHA-256, Encryptions, IAM Controls, Private key etc. |
| 3. | **Scalable Architecture** | Server provisioning, Server availability, Local phone storage | IBM Cloud |
| 4. | **Availability** | 24/7 service, Continuous update, Data maintenance, Private staff | IBM Cloud |
| 5. | **Performance** | Distributed servers, Cloud communication, Notification system, Effective information sharing | IBM Cloud ,python |

# 5.3 User Stories

| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|---|---|---|---|---|---|---|
| Customer (Traveller) | User Visibility | USN-1 | As a user, I can able to read the message on the display and became aware of it | I can able to read the message | High | Sprint-1 |
| | | USN-2 | As a user, I can able to know the nearby warning signs and became aware of it | I can able to know the nearby warning signs | Medium | Sprint-2 |
| | | USN-3 | As a user, I can became aware of crossing by blinking light system and it keeps me awake | I can aware of road crossing places | Medium | Sprint-1 |
| | | USN-4 | As a user, I can able to know the traffic signal countdown and get ready to move | I can know traffic signal countdown | Low | Sprint-4 |
| | User Interaction | USN-5 | As a user, I can get to know about the road conditions through the notification | I can know road conditions in notification | Low | Sprint-4 |
| Customer (Pedestrian) | | USN-6 | As a user, I can able to change the crossing button manually to cross the road | I can change the crossing button manually | High | Sprint-2 |
| Administrator | | USN-7 | As an admin, I can able to change the display messages manually using a security key | I can change the display message manually | Medium | Sprint-2 |
| | Traffic Detection | USN-8 | As an admin, I can able to track the traffic and able to display the message accordingly | I can track the traffic density | High | Sprint-3 |
| | | USN-9 | As an admin, I can able to change the traffic lights based on the traffic detection | I can change traffic lights in case of emergency | High | Sprint-3 |
| Customer (Traveller) | Communication | USN-10 | As a user, I can get the weather alert based on the location | I can get weather alerts | Medium | Sprint-4 |
| Administrator | | USN-11 | As an admin, I can able to alert the diversion to the user based on the traffic | I can alert the diversion to the user | High | Sprint-4 |

# 6. PROJECT PLANNING AND SCHEDULING

## 6.1 Sprint Planning & Estimation: -

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | Registration | USN-1 | As a Driver, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Raagavi Selvakumar |
| Sprint-1 | User Confirmation | USN-2 | As a Driver, I will receive confirmation email once I have registered for the application | 1 | Medium | Viha Aswinesriee SMR |
| Sprint-1 | Login | USN-3 | As a Driver, I can log into the application by entering email | 2 | High | Pavithra .V |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | & password | | | |
| Sprint-2 | Interface Sensor | USN-1 | A sensor interface is a bridge between a device and any attached sensor. The interface takes data collected by the sensor and outputs it to the | 2 | High | Habinaya .R |
| | | | attached device. | | | |
| Sprint-3 | Coding (Accessing datasets) | USN-1 | Coding is a set of instructions used to manipulate information so that a certain input results in a particular output. | 2 | High | Raagavi Selvakumar , Viha Aswinesriee SMR , Pavithra .V and Habinaya .R |
| Sprint-4 | Web Application | USN-1 | As a Driver, I will display the current weather & Automatic diversion for road traffic & Accident | 1 | Medium | Raagavi Selvakumar and Viha Aswinesriee SMR |

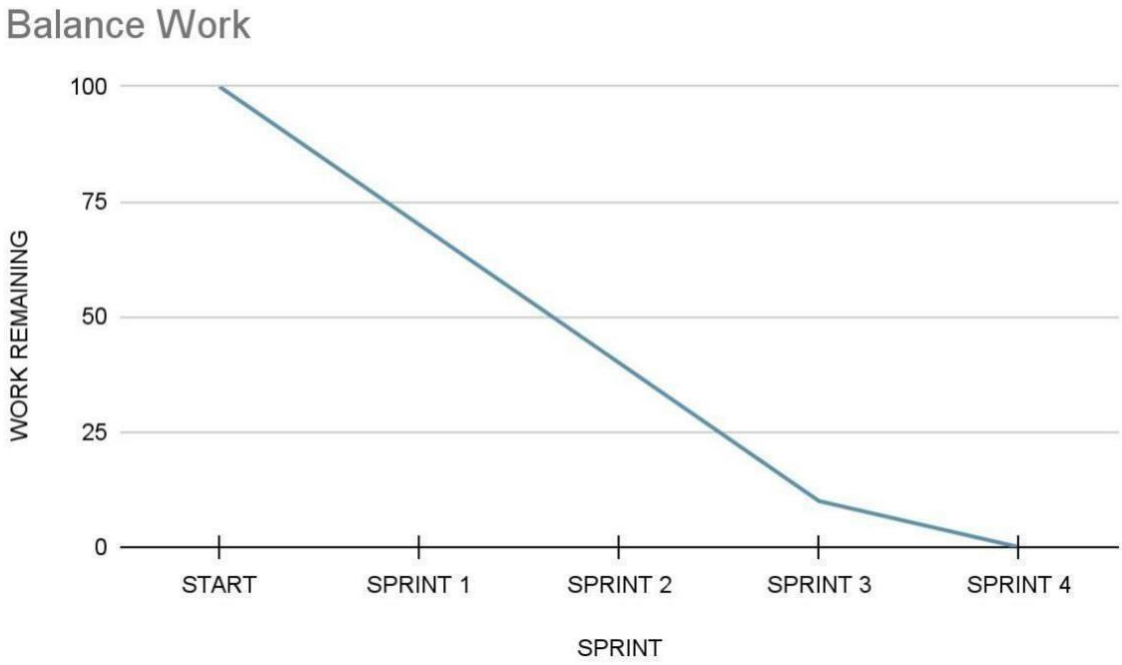
**Project Tracker, Velocity & Burndown Chart: (4 Marks)**

| 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 | 26 Oct 2022 | 29 Oct 2022 | 20 | |
| Sprint-2 | 20 | 5 Days | 30 Oct 2022 | 03 Nov 2022 | 20 | |
| Sprint-3 | 20 | 8 Days | 04 Nov 2022 | 11 Nov 2022 | 20 | |
| Sprint-4 | 20 | 9 Days | 12 Nov 2022 | 20 Nov 2022 | 20 | |

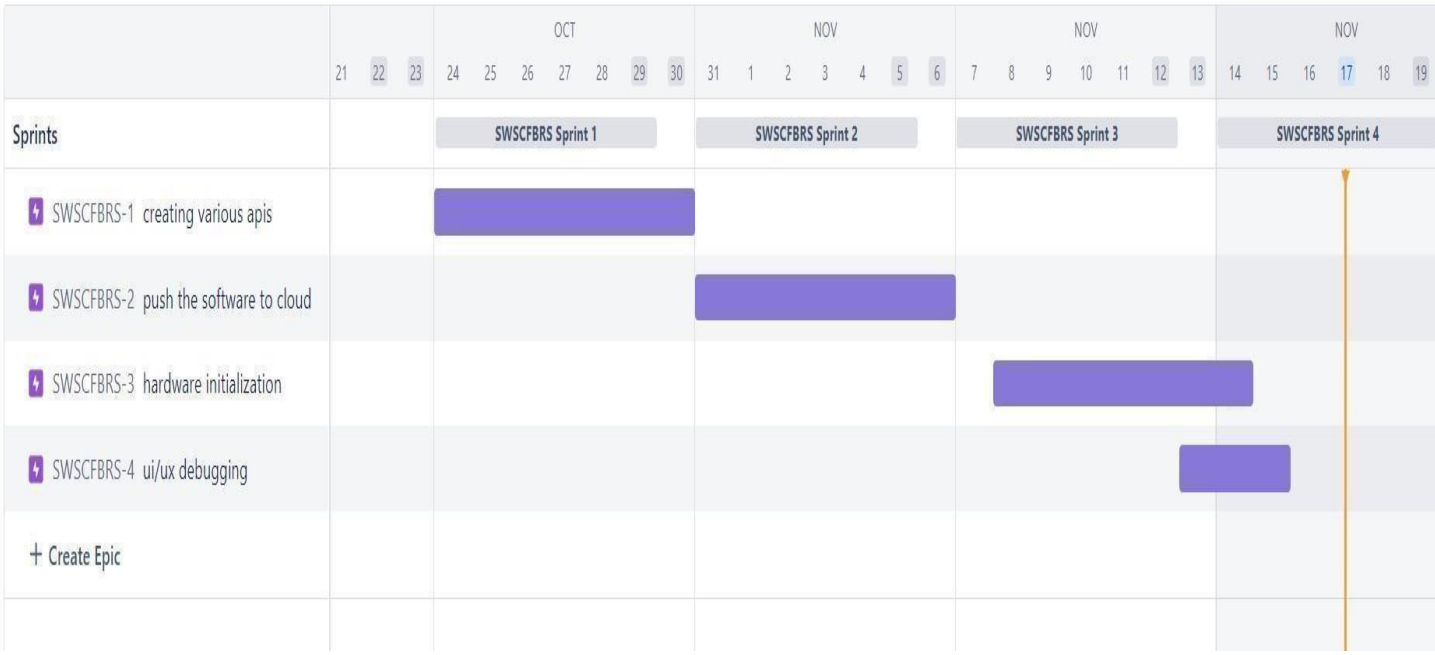
# 6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart:

**Burndown Chart:**

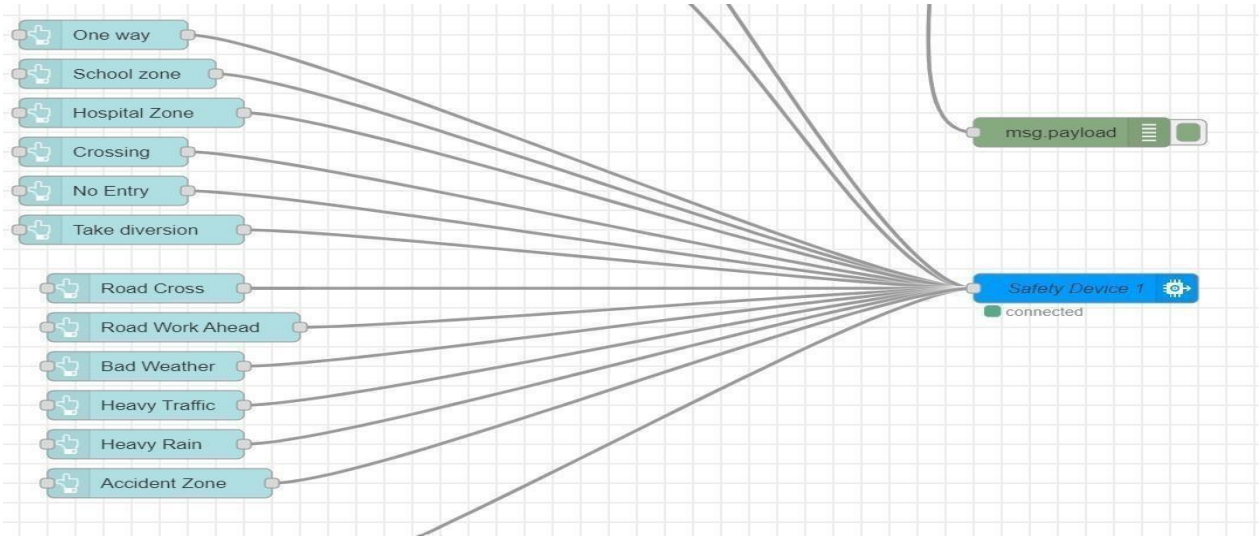



# 6.3 Reports from JIRA Software

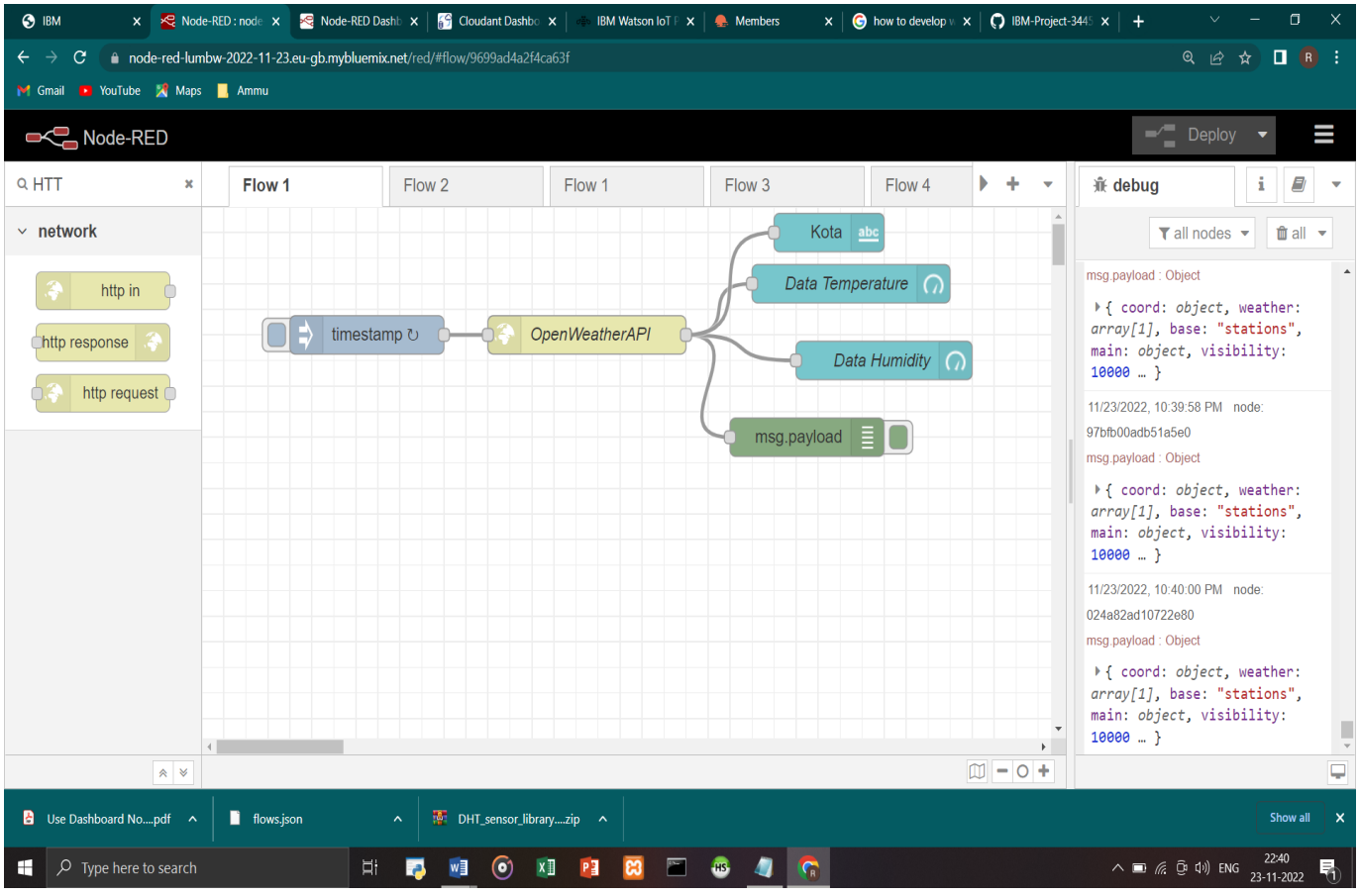



# 7. CODING AND SOLUTIONING

## 7.1 Feature 1 – Admin UI:-

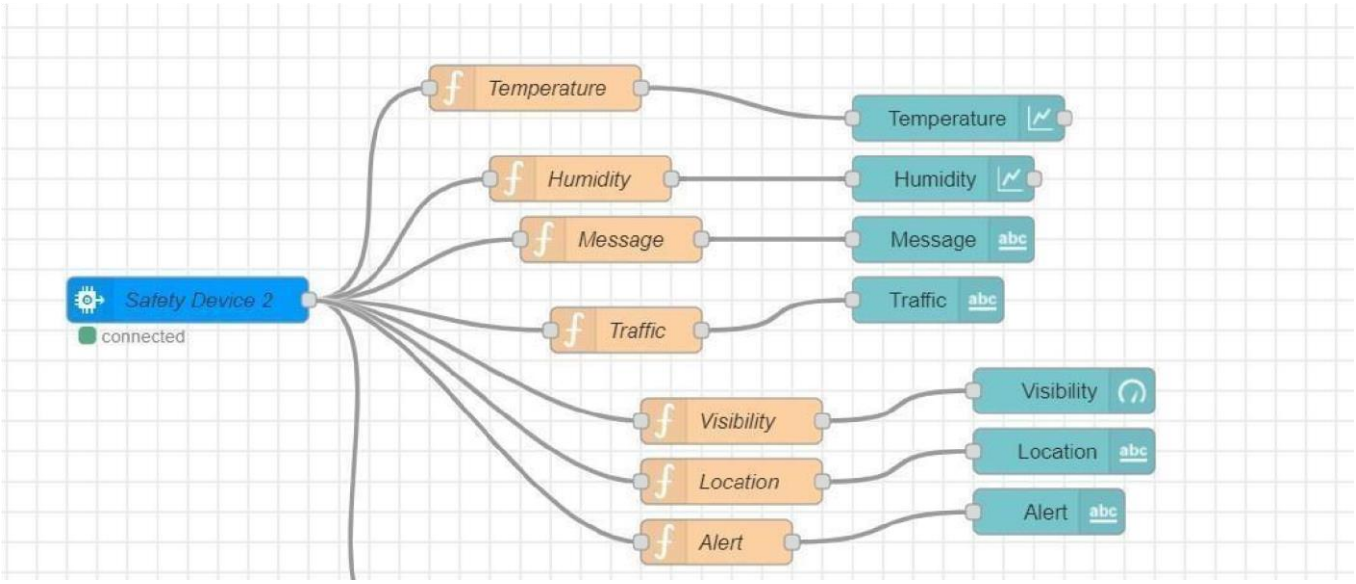



This part of Node RED flow creates aUI for Admin to control the display board to change the sign manually from a remote location. There is also an App for Admin to control the sign boards.



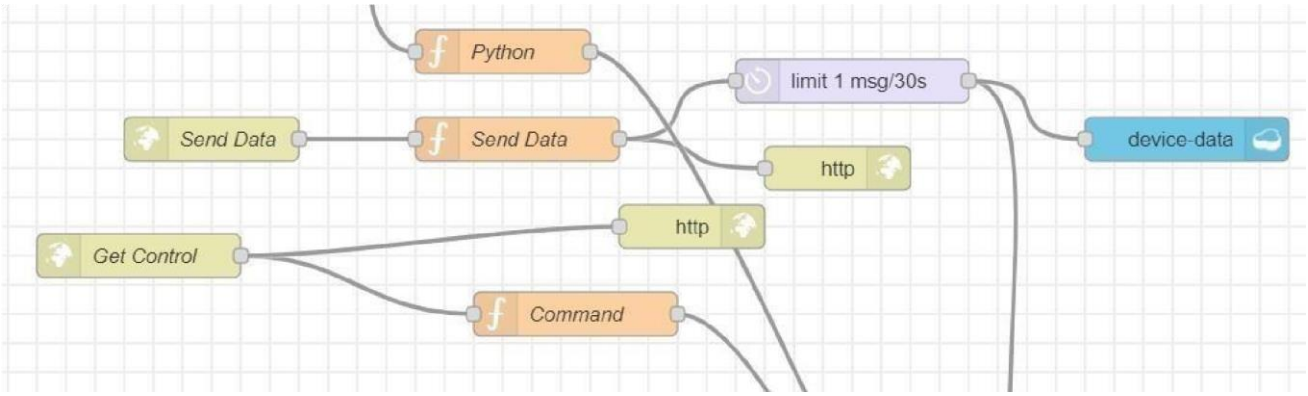
## 7.2 Feature 2 – user UI

This part of Node RED flow Creates a UI for Users Where User can able to see current weather update and road conditions . A phyton code sends the Data to the User UI through IOT Watson




## 7.3 Feature 3 – Data Send Flow

The part of Node RED flow gets the information from HTTP request and store in the cloudant DB named “device data” for every 30 seconds from the MIT App.




# 8. TESTING

## 8.1 Test Cases

**Test Case 1:** Display welcome message on the digital sign board when no data received from IOT Watson.

**Test Case 2:** Display message on the digital sign board based on the information from the IOT Watson.

**Test Case 3:** Display alert message on the digital sign board based on the admin input from the IOT Watson.

**Test Case 4:** Change the traffic lights when manual crossing button is pressed by the user.

## 8.2 User Acceptance Testing

Dynamic changes in the sign boards based on the input from the python to the IOT Watson helps user to avoid traffic and have a safe journey home. The users would welcome this idea to be implemented everywhere.

# 9. RESULTS

### 9,1 Performance Metrics

Based on the IBM pack we chose, the performance of the website varies. Built upon NodeJS, a light and high performance engine, Node RED is capable of handling up to 10,000 requests per second. Moreover, since the system is horizontally scalable, an even higher demand of customers can be served.

# 10. ADVANTAGES & DISADVANTAGES

- **ADVANTAGES**

- Lower battery consumption since processing is done mostly by Node RED servers in the cloud.
- Cheaper and low requirement micro controllers can be used since processing requirements are reduced.
- Longer lasting systems.
- Dynamic Sign updating.

- School/Hospital Zone alerts

- **DISADAVNTAGES**

- The size of the display determines the requirement of the micro controller

- Dependent on OpenWeatherMap API and hence the Weather Data is same for a large area in the scale of cities.

# 11. CONCLUSION :-

Our project is capable of serving as a replacement for static signs for a comparatively lower cost and can be implemented in the very near future. This will help reduce a lot of accidents and maintain a more peaceful traffic atmosphere in the country.

# 12. FUTURE SCOPE:-

Introduction of intelligent road sign groups in real life scenarios could have great impact on increasing the driving safety by providing the end-user (car driver) with the most accurate information regarding the current road and traffic conditions. Even displaying the information of a suggested driving speed and road surface condition

(Temperature, icy, wet or dry surface) could result in smoother traffic flows and, what is more important, in increasing a driver's awareness of the road situation.

# 13. APPENDIX

## SOURCE CODE

### 1. IOT Device (ESP 32)

Sketch.ino:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>//library for display

#define LED 2
#define RED 19 //Red light
#define YELLOW 18 //Yellow light
#define GREEN 5 //Green light
#define CROSS 13
#define CROSSIN 4
#define CROSSOUT 15
#define Crossing 34 //White Button
#define SchoolZone 35 //Yellow Button
#define HospitalZone 32 //Red Button
#define NoEntry 25 //Black Button
#define OneWay 26 //Blue Button
```

```
#define TakeDiversion 27 //Grey Button

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

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

#define ORG "m1r2sh"//IBM ORGANITION ID
#define DEVICE_TYPE "Roadsafety"//Device type mentioned in ibm watson
    IOT Platform
#define DEVICE_ID "safetydevice1"//Device ID mentioned in ibm watson IOT
    Platform
#define TOKEN "!_1ZsGYHI9TsD5kvOu"     //Token
String message; int buttonState = 0,i=30;
#define SCREEN_ADDRESS 0x3C

//-------- 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

//-------------------------------------
Adafruit_SSD1306 oled (128, 64, &Wire, -1);
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);
oled.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
pinMode(LED,OUTPUT);   pinMode(RED,OUTPUT);
pinMode(YELLOW,OUTPUT);   pinMode(GREEN,OUTPUT);
pinMode(CROSSIN,OUTPUT);    pinMode(CROSSOUT,OUTPUT);
pinMode(CROSS, INPUT);   pinMode(Crossing, INPUT);
pinMode(SchoolZone, INPUT);
 pinMode(HospitalZone,  INPUT);
    pinMode(NoEntry, INPUT);   pinMode(OneWay, INPUT);
pinMode(TakeDiversion, INPUT);

  oled.clearDisplay();   oled.setTextSize(1);
oled.setTextColor(WHITE);
oled.setCursor(10, 10);
oled.println("Welcome to Chennai");
oled.setCursor(20, 20);
oled.println("Speed Limit 40");
oled.setCursor(40, 30);   oled.println("Go
Slow!");     oled.display();

  digitalWrite(GREEN, HIGH);
digitalWrite(CROSSOUT, HIGH);   delay(10);   Serial.println();
wificonnect();     mqttconnect();
}
```

```
void loop()// Recursive Function
{
  changeText();
roadCross();
PublishData();   ledBlink();
delay(1000);
if (!client.loop()) {      mqttconnect();
  }
}

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

void PublishData() {     mqttconnect();//function call for
connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
*/
  String payload = "{\"Message\":\"Enter Command to Display\"}";


  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 definition for wificonnect
{
 Serial.println();
 Serial.print("Connecting to ");

 WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to
   establish the connection
 while (WiFi.status() != WL_CONNECTED) {      delay(500);
  Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}
void initManagedDevice() {   if (client.subscribe(subscribetopic))
{
```

```
Serial.println((subscribetopic));
   Serial.println("subscribe to cmd OK");
  } else {
   Serial.println("subscribe to cmd FAILED");
  }
}

void ledBlink(){//function for led blinking system
digitalWrite(LED,LOW);   delay(1000);    digitalWrite(LED,HIGH);
}

void countDown(){//traffic light countdown system
for(i;i>0;i--){    oled.clearDisplay();    oled.setTextSize(3);
oled.setCursor(48, 20);      oled.println(i);
oled.display();      delay(1000);
  }
  oled.setTextSize(1);    i=30;
}

void roadCross(){//manual crossing function
  buttonState = digitalRead(CROSS);   if(buttonState
  == LOW){       trafficOff();
   digitalWrite(CROSSOUT,LOW);
digitalWrite(CROSSIN, HIGH);    countDown();
crossing();      trafficOn();
digitalWrite(CROSSIN, LOW);
digitalWrite(CROSSOUT, HIGH);
  }
}
```

```
void trafficOn(){//traffic light set to go
digitalWrite(RED,HIGH);   delay(1000);
digitalWrite(RED,LOW);
digitalWrite(YELLOW,HIGH);
delay(1500);   digitalWrite(YELLOW,LOW);
digitalWrite(GREEN,HIGH);
}

void trafficOff(){//traffic light set to stop
digitalWrite(GREEN,HIGH);
delay(1000);   digitalWrite(GREEN,LOW);
digitalWrite(YELLOW,HIGH);
delay(1500);   digitalWrite(YELLOW,LOW);
  digitalWrite(RED,HIGH);
}

void crossing(){//crossing display
oled.clearDisplay();          oled.setCursor(20,    25);
oled.println("Crossing Ahead");   oled.setCursor(40,
35);   oled.println("Go Slow!");    oled.display();
}

void schoolZone(){//school zone display
oled.clearDisplay();   oled.setCursor(30,
25);   oled.println("School Zone");
oled.setCursor(28, 35);   oled.println("Do
Not Honk!");     oled.display();
}
```

```
void hospitalZone(){//hospital zone display
oled.clearDisplay();   oled.setCursor(25,
25);   oled.println("Hospital Zone");
oled.setCursor(28, 35);   oled.println("Do
Not Honk!");     oled.display();
}

void noEntry(){//no entry display
  oled.clearDisplay();  oled.setCursor(40,
  25); oled.println("No Entry");
  oled.setCursor(10, 35);   oled.println("No
Vehicles Allowed");     oled.display();
}

void oneWay(){//one way display   oled.clearDisplay();
oled.setCursor(40, 25);   oled.println("One Way");
oled.setCursor(30, 35);   oled.println("Single Lane");
oled.display();
}

void takeDiversion(){//take diversion display
oled.clearDisplay();
oled.setCursor(20, 25);   oled.println("Take
Diversion");          oled.setCursor(10,    35);
oled.println("Bad       Road       Condition");
oled.display();
}

void roadWorkAhead(){//alert road work
```

```
ahead    oled.clearDisplay();   oled.setCursor(40,
15);
oled.println("CAUTION!");   oled.setCursor(17,
25);
  oled.println("Road Work Ahead"); oled.setCursor(15, 35);
  oled.println("Work On Progress");
 oled.display();
}

void badWeather(){//alert bad weather
oled.clearDisplay();   oled.setCursor(40,
15);   oled.println("CAUTION!");
oled.setCursor(30, 25);   oled.println("Bad
Weather");   oled.setCursor(20, 35);
oled.println("Low Visibility");
oled.display();
}

void heavyTraffic(){//alert heavy traffic
oled.clearDisplay();   oled.setCursor(40,
15);   oled.println("CAUTION!");
oled.setCursor(25, 25);
oled.println("Heavy Traffic");
oled.setCursor(22, 35);   oled.println("Take
Diversion");    oled.display();
}

void heavyRain(){//alert heavy rain
oled.clearDisplay();   oled.setCursor(40,
15); oled.println("CAUTION!");
```

```
oled.setCursor(33, 25);
oled.println("Heavy Rain");
oled.setCursor(40, 35);   oled.println("Go
Slow!");     oled.display();
}

void accidentZone(){//alert accident zone
oled.clearDisplay();   oled.setCursor(40, 15);
oled.println("CAUTION!");   oled.setCursor(25,
25);   oled.println("Accident Zone");
oled.setCursor(20, 35);   oled.println("Speed
Limit 30");     oled.display();
}
void changeText(){//change display text
if(digitalRead(Crossing) == LOW){      crossing();
  }else if(digitalRead(SchoolZone) == LOW){       schoolZone();
  }else if(digitalRead(HospitalZone) == LOW){       hospitalZone();
  }else if(digitalRead(NoEntry) == LOW){       noEntry();
  }else if(digitalRead(OneWay) == LOW){       oneWay();
  }else if(digitalRead(TakeDiversion) == LOW){     takeDiversion();
  }
}

void  editText(String  msg){//edit  didplay
text   if(msg == "Crossing"){      crossing();
}else       if(msg       ==       "SchoolZone"){
schoolZone();                }else   if(msg   ==
"HospitalZone"){     hospitalZone();  }else
if(msg == "NoEntry"){     noEntry();    }else
if(msg == "OneWay"){      oneWay();
```

```
  }else if(msg == "TakeDiversion"){      takeDiversion();
}else if(msg
== "RoadCross"){     trafficOff();
digitalWrite(CROSSOUT,LOW);
digitalWrite(CROSSIN,     HIGH);     countDown();
                          crossing();
trafficOn();     digitalWrite(CROSSIN, LOW);
digitalWrite(CROSSOUT,
HIGH);               }else     if(msg     ==
"RoadWorkAhead"){
roadWorkAhead();   }else if(msg ==
"BadWeather"){    badWeather();   }else
    if(msg       ==     "HeavyTraffic"){
heavyTraffic();
  }else if(msg == "HeavyRain"){    heavyRain();
  }else if(msg == "AccidentZone"){    accidentZone();
  }
}

//CallBack function   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++) {      message
+= (char)payload[i];
  }
  editText(message);
  Serial.println("data: "+ message);    message="";
}
```

## 2. PYTHON CODE

```
import time import sys
import ibmiotf.application
import ibmiotf.device
import random as r
import weather
#Provide your IBM Watson Device Credentials organization = "m1r2sh"
deviceType = "Roadsafety" deviceId = "safetydevice2" authMethod = "token"
authToken = "t3USLaRSVT*BbaPGlA"
 #Data List
Message_list = ["Crossing","School Zone","Hospital Zone","No Entry","One
    Way","Take Diversion"]
Traffic_list = ["High","Moderate","Low"]
Notify_list = ["Heavy Traffic","Heavy Rain","Bad Weather","Road Work
    Ahead","Accident Zone"]    myLocation
= "Chennai,IN"
APIKEY = "3833389c301e845d271b287e18bfba2f"
 # Initialize GPIO
 def myCommandCallback(cmd):
   print("Command received: %s" % cmd.data['command'])
print(cmd)
          try:

    device
Options =
{"org": organizat
ion, "type":
deviceTy pe,
"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:
            #Random Data
            Message = r.choice(Message_list)
            Traffic = r.choice(Traffic_list)
            Alert = r.choice(Notify_list)

            #Get Weather from OpenWeatherMap
            weatherData = weather.get(myLocation, APIKEY)

            if Traffic == 'High':
               Message = "Go Slow!"           Alert =  "Heavy
        Traffic"        elif weatherData["weather"] ==
        "['rain']":
               Alert = "Heavy Rain"
            #JSON Data          data = {"d":{ 'temp' :
        round(weatherData["temperature"], 2),                    'humidity' :
        weatherData["humidity"],
                        'visibility' : weatherData["visibility"],
                        'location' : myLocation,
                        'message' : Message,
```

```
                'traffic' : Traffic,
                'alert' : Alert}
            }
      #print data       def
myOnPublishCallback():
         print ("Published Temperature = %s C" %
   round(weatherData["temperature"], 2), "Humidity = %s %%" %
   weatherData["humidity"], "to IBM Watson")

      success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)       if not success:
print("Not connected to IoTF")       time.sleep(5)

      #CallBack
      deviceCli.commandCallback = myCommandCallback

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

Weather.Py:

```
import requests as reqs
def get(myLocation,APIKEY):
apiURL =
f"https://api.openweathermap.org/data/2.5/weather?q={myLocation}&appi
d={APIKEY}"    responseJSON = (reqs.get(apiURL)).json()

  #JSON Object
returnObject = {
      "temperature" : responseJSON['main']['temp'] - 273.15,
      "humidity" : responseJSON['main']['humidity'],
```

```
        "weather" : [responseJSON['weather'][_]['main'].lower() for _ in
    range(len(responseJSON['weather']))],
        "visibility" : responseJSON['visibility']/100
      }
      return(returnObject)
```

## GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-34453-1660235984