



**NAALAIYA THIRAN PROJECT - 2022  
19ECI01-PROFESSIONAL READINESS  
FOR INNOVATION, EMPLOYABILITY  
AND ENTREPRENEURSHIP**



**IOT BASED SMART CROP PROTECTION SYSTEM**

**A PROJECT REPORT**

*Submitted by*

<b>MANIGANDAN S</b>	<b>1904088</b>
<b>UDHAYA PRAKASH T</b>	<b>1904118</b>
<b>VARUN KARTHIK T</b>	<b>1904120</b>
<b>VENKAT RAMANAN M</b>	<b>1904121</b>
<b>HARIHARAN S</b>	<b>2004206</b>

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION  
ENGINEERING**

**COIMBATORE INSTITUTE OF TECHNOLOGY, COIMBATORE – 641 014**

**(Government Aided Autonomous Institution affiliated to Anna University)**

**ANNA UNIVERSITY: CHENNAI 600025**

**NOVEMBER 2022**

# **COIMBATORE INSTITUTE OF TECHNOLOGY**

**(Government aided Autonomous Institution Affiliated to Anna University)**

**COIMBATORE – 641014**

**ANNA UNIVERSITY: CHENNAI 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this report “**SIGNS WITH SMART CONNECTIVITY FOR BETTER ROAD SAFETY**” is the Bonafide work of **MANIGANDAN S(1904088), UDHAYA PRAKASH(1904118), VARUN KARTHIK T(1904120), VENKAT RAMANAN M(1904121) AND HARIHARAN S(2004206)** who carried out **19ECI01 Professional Readiness for Innovation, Employability and Entrepreneurship** project offered by IBM and Anna University ,Chennai.

---

**SIGNATURE**

**Dr. P. MUTHU**

**SUBRAMANIYAN,M.E,Ph.D.,**

**FACULTY MENTOR**

Professor

Department of Electronics and  
Communication Engineering

---

**SIGNATURE**

**Dr.S.UMA**

**MAHESWARI,M.E,Ph.D.,**

**FACULTY EVALUATOR**

Associate Professor

Department of Electronics  
and Communication  
Engineering

---

**SIGNATURE**

**Dr. A. RAJESWARI, M.E. Ph.D.,**

**PRINCIPAL AND HEAD**

Professor

Department of Electronics  
and  
Communication Engineering

## PROJECT CALENDAR

Phase	Phase Description	Week	Dates	Activity Details
1	Preparation Phase (Pre- requisites, Registrations, Environment Set-up, etc.)	2	22 - 27 Aug 2022	Creation GitHub account & collaborate with Project repository in project workspace
2	Ideation Phase (Literature Survey, Empathize, Defining Problem Statement, Ideation)	2	29 Aug – 3rd Sept 2022	Literature survey (Aim, objective, problem statement and need for the project)
		3	5 - 10th Sept 2022	Preparing Empathy Map Canvas to capture the user Pains & Gains
		4	12 - 17 Sept 2022	Listing of the ideas using brainstorming session
3	Project Design Phase -I (Proposed Solution, Problem- Solution Fit, Solution Architecture)	5	19 - 24 Sept 2022	Preparing the proposed solution document
		6	26 Sept - 01 Oct 2022	Preparing problem - solution fit document & Solution Architecture
4	Project Design Phase -II (Requirement Analysis, Customer Journey, Data Flow Diagrams, Technology Architecture)	7	3 - 8 Oct 2022	Preparing the customer journey maps
		8	10 - 15 Oct 2022	Preparing the Functional Requirement Document & Data- Flow Diagrams and Technology Architecture
5	Project Planning Phase (Milestones & Tasks, Sprint Schedules )	9	17 - 22 Oct 2022	Preparing Milestone & Activity List, Sprint Delivery Plan
6	Project Development Phase (Coding & Solutioning, acceptance Testing, Performance Testing)	10	24 - 29 Oct 2022	Preparing Project Development - Delivery of Sprint-1
		11	31 Oct - 5 Nov 2022	Preparing Project Development - Delivery of Sprint-2
		12	7 - 12 Nov 2022	Preparing Project Development - Delivery of Sprint-3
		13	14 - 19 Nov 2022	Preparing Project Development - Delivery of Sprint-4

## TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
	<b>LIST OF FIGURES</b>	<b>iii</b>
	<b>LIST OF TABLES</b>	<b>vi</b>
<b>1</b>	<b>INTRODUCTION</b>  1.1 PROJECT OVERVIEW  1.2 PURPOSE	<b>1</b>
<b>2</b>	<b>LITERATURE SURVEY</b>  2.1 EXISTING PROBLEM  2.2 REFERENCES  2.3 PROBLEM STATEMENT DEFINITION	<b>2</b>
<b>3</b>	<b>IDEATION AND PROPOSED SOLUTION</b>  3.1 EMPATHY MAP CANVAS  3.2 IDEATION AND PROPOSED SOLUTION  3.3 PROPOSED SOLUTION  3.4 PROBLEM SOLUTION FIT	<b>5</b>
<b>4</b>	<b>REQUIREMENT ANALYSIS</b>  4.1 FUNCTIONAL REQUIREMENTS  4.2 NON- FUNCTIONAL REQUIREMENTS  4.3 CUSTOMER JOURNEY	<b>11</b>

<b>5</b>	<b>PROJECT DESIGN</b> 5.1. DATA FLOW DIAGRAMS 5.2 SOLUTION AND TECHNICAL ARCHITECTURE 5.3 USER STORIES	<b>14</b>
<b>6</b>	<b>PROJECT PLANNING AND SCHEDULING</b> 6.1 SPRINT PLANNING AND ESTIMATION 6.2. SPRINT DELIVERY SCHEDULE	<b>21</b>
<b>7</b>	<b>CODING AND SOLUTIONING</b> 7.1 FEATURES	<b>24</b>
<b>8</b>	<b>TESTING</b>	<b>34</b>
<b>9</b>	<b>RESULTS</b> 9.1 APPLICATION 9.2 HARDWARE	<b>35</b>
<b>10</b>	<b>ADVANTAGES AND DISADVANTAGES</b>	<b>40</b>
<b>11</b>	<b>CONCLUSION</b>	<b>41</b>
<b>12</b>	<b>FUTURE SCOPE</b>	<b>42</b>
<b>13</b>	<b>APPENDIX</b> 13.1 SOURCE CODE 13.2 GITHUB & PROJECT DEMO LINK	<b>43</b>

## LIST OF FIGURES

<b>FIGURE NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>3.1</b>	<b>EMPATHY MAP</b>	<b>5</b>
<b>3.2</b>	<b>BRAINSTORM</b>	<b>8</b>
<b>3.3</b>	<b>PROBLEM SOLUTION FIT</b>	<b>10</b>
<b>4.3</b>	<b>CUSTOMER JOURNEY</b>	<b>13</b>
<b>5.1</b>	<b>DATA FLOW DIAGRAM</b>	<b>16</b>
<b>5.2</b>	<b>TECHNICAL ARCHITECTURE</b>	<b>17</b>
<b>7.1</b>	<b>LIBRARIES REQUIRED</b>	<b>24</b>
<b>7.2</b>	<b>OPEN WEATHER API</b>	<b>25</b>
<b>7.3</b>	<b>OPEN WEATHER WEBSITE</b>	<b>26</b>
<b>7.4</b>	<b>GET WEATHER FUNCTIONS</b>	<b>27</b>
<b>7.5</b>	<b>KEY VALUE AND THE STATUS OF THE OPEN WEATHER API</b>	<b>28</b>

<b>7.6</b>	<b>IMPORTING PACKAGES AND CLOUD CREDENTIALS</b>	<b>28</b>
<b>7.7</b>	<b>CONNECTING TO WATSON USING WIOTP PACKAGE</b>	<b>29</b>
<b>7.8</b>	<b>GETTING VALUES FROM OPENWEATHER API</b>	<b>29</b>
<b>7.9</b>	<b>WEATHER RELATED SIGN BOARD CONDITIONS</b>	<b>30</b>
<b>7.10</b>	<b>SCHOOL TIME WARNING CONDITION</b>	<b>30</b>
<b>7.11</b>	<b>PUSHING INTO IBM WATSON CLOUD</b>	<b>31</b>
<b>7.12</b>	<b>ARDUINO SETUP</b>	<b>31</b>
<b>7.13</b>	<b>GETTING VALUE FROM NODE-RED</b>	<b>32</b>
<b>7.14</b>	<b>PARSING THE VALUES</b>	<b>32</b>
<b>7.15</b>	<b>DISPLAYING THE PARSED VALUES</b>	<b>33</b>
<b>9.1</b>	<b>WEATHER WINDOW</b>	<b>35</b>
<b>9.2</b>	<b>HOME PAGE DISPLAYING CLOUDY WEATHER</b>	<b>36</b>
<b>9.3</b>	<b>HOME PAGE DISPLAYING SCHOOL HOURS</b>	<b>37</b>

<b>9.4</b>	<b>LCD DISPLAYING LOCATION</b>	<b>38</b>
<b>9.5</b>	<b>LCD DISPLAYING WEATHER AND TIME</b>	<b>38</b>
<b>9.6</b>	<b>LCD DISPLAYING SPEED LIMIT</b>	<b>39</b>



## LIST OF TABLES

<b>TABLE NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>4.1</b>	<b>FUNCTIONAL REQUIREMENTS</b>	<b>11</b>
<b>4.2</b>	<b>NON FUNCTIONAL REQUIREMENTS</b>	<b>12</b>
<b>5.1</b>	<b>COMPONENTS AND TECHNOLOGIES</b>	<b>18</b>
<b>5.2</b>	<b>APPLICATION CHARACTERISTICS</b>	<b>19</b>
<b>5.3</b>	<b>USER STORIES</b>	<b>20</b>
<b>6.1</b>	<b>MILESTONE AND TASK</b>	<b>22</b>
<b>6.2</b>	<b>SPRINT PLANNING</b>	<b>23</b>
<b>6.3</b>	<b>SPRINT DELIVERY SCHEDULE</b>	<b>23</b>

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1. PROJECT OVERVIEW**

This project proposes a system to monitor and control the traffic signals using sensors to regulate the flow of traffic, to avoid the congestion for smooth flow of traffic and to reduce accidents. Traffic sign boards play an important role to make the traffic in shape and to control and manage the traffic on roads. Many times, the driver misses the sign boards while driving due to various reasons like insufficient light, fog, rain, traffic and also accidents may be caused because of improper signs or wrong signs. To overcome this static sign boards are replaced with smart dynamic sign boards which can change the sign with the command from the server. If there is rainfall then the roads will be slippery and the speed limit should be decreased.

The weather conditions are obtained and the sign and speed limit are in accordance with the conditions. The data of the road diversions, accident prone areas, school zone, hospital zone, industry zone and the information sign boards are obtained and stored in cloud. This data is retrieved and dynamically displayed on the sign boards accordingly. So in this project we will replace the static signboards with smart connected sign boards. Also based on the traffic and fatal situations the diversion signs are displayed. Guide, Warning and Service signs are also displayed accordingly. Different modes of operations can be selected and performed in the smart traffic sign board.

#### **1.2 PURPOSE**

Replace the static sign boards in the roads with the dynamic sign boards which provide the advantage of changing the content of the board whenever there is need of an update in the sign board.

Using the sign boards will be efficient in preventing accidents which are occurring in the roads due to the speeding . It can also provide the changes in the environment to the drivers and can provide the alert to the drivers. The real time weather details will be imported from the open weather map API service which is then connected to the cloud and through Watson Iot Platform and exchanging the data and to display values.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 EXISTING PROBLEM**

The Existing sign boards are dynamic any update on the boards will take more amount of time, thus we need the immediate solution for the road due accident happening in the road. A proper solution for this needed to be done so that the traffic jam and the accident would be avoided in the road. There is a problem of inefficient rerouting due to the construction and repair in the roads.

#### **2.2 REFERENCES**

[1] C. Dhule, R. Agrawal (2021) In this article, a real LED display board is created with the help of the raspberry pi. Continuous monitoring of the environment allows for the identification of temperature, humidity and moisture around the surrounding which helps in deciding the weather around the point or area. the Raspberry pi microcontroller-based board is used for the data acquisition strategy and the use of analog and digital sensors. Temperature, humidity, light intensity and gas concentrations can be monitored in real-time. The data fetching is done from the Google API and the data is uploaded to the web application and through it the data is hosted on the server. From the raspberry pi the data are imported and decision is made. The results are presented in the LED. From this paper , The method to upload and import the data from the web application is been learned and the LED interface with the raspberry pi is also been learned. The size and shape of an electronic warning sign board, which must display a varied message in real-time circumstances day and night, may vary depending on the natural setting and the size of the audience.

[2] Bhatt D. P., & Tiwari M. (2019) In this article, The RFID tag will come into play where both sign board and the vehicle will be equipped with the tag. The process that goes on the Device is that the RFID in the sign board will detect the speed of each and every vehicle which is travelling in the road using the RFID tag the speed of the car will be detected and the speaker, display on the sign board will indicate the vehicle to slow down the car and notes down the speed of the data if the speeding does not goes don't then it will transfer it to the cloud stating that the rules are violated. This project uses the arduino uno and long-distance UHF RFID element. Traffic sign boards are crucial for keeping traffic organized and for managing and controlling it on the highways. The driver frequently misses

the sign boards while driving for a variety of reasons, such as inadequate lighting, fog, rain, traffic, etc. In this paper, a framework for Smart Traffic Sign Boards (STSB) is proposed

[3] Bin C. & Jian L. (2009) This article primarily discusses the job of integrating the control system and the data collection process based on obtaining current weather information from the signals required to produce the given data. The second half of this article discusses hardware control. An integrated smart digital display board is the result of all interactions. It is simple to access, supports numerous screens, and has strong content integration. They are ideal for creating smart digital boards because of their distinctive property. The project obtains weather data from a variety of stations that are accessible globally, including an international weather data source like weather, open weather maps, weather group API, Foreca, -Sky Sky, and earth weather online. The project's chief manager can independently publicise the information together with the weather data and is designed to construct an effective system to display the most accurate real-time weather data. In addition to measuring temperature, humidity, and air-speed data, as well as controlling air and pollution, the Application-Programming Interface (API) is utilised to retrieve weather data.

## **2.3 PROBLEM STATEMENT DEFINITION**

In present Systems the road signs and the speed limits are Static. But the road signs can be changed in some cases. We can consider some cases when there are some road diversions due to heavy traffic or due to accidents then we can change the road signs accordingly if they are digitalized. If there is construction or an repair working is going on the road then the vehicle must be intimated with the sign to reroute it position so that to avoid the traffic jam on the road and it will efficient for the worker to complete the works in the given ample amount of time.

This project proposes a system which has digital sign boards on which the signs can be changed dynamically. If there is rainfall then the roads will be slippery and the speed limit would be decreased. There is a web app through which you can enter the data of the road diversions, accident prone areas and the information sign boards can be entered through web

app. This data is retrieved and displayed on the sign boards accordingly. So in this project we will replace the static signboards with smart connected sign boards.

Based on the weather changes the speed may increase or decrease. So these smart connected sign boards get the speed limitations from a web app using weather API and update automatically. Also Based on the traffic and fatal situations the diversion signs are displayed. Guide (Schools), Warning and Service (Hospitals, Restaurant) signs at the position of the schools will be placed on the few distance before the institution

## CHAPTER 3

### IDEATION & PROPOSED SOLUTION

#### 3.1 EMPATHY MAP CANVAS

An empathy map is a simple, easy picture that summarizes information about a user's actions and views. It is a helpful tool to assist teams in comprehending their users.

It's essential to understand both the actual issue and the individual who is experiencing it in order to develop a workable solution. Participants learn to think about situations from the user's perspective, including goals and problems, through the exercise of making the map. The empathy map of Signs with Smart Connectivity for Better Road Safety is shown in 3.1

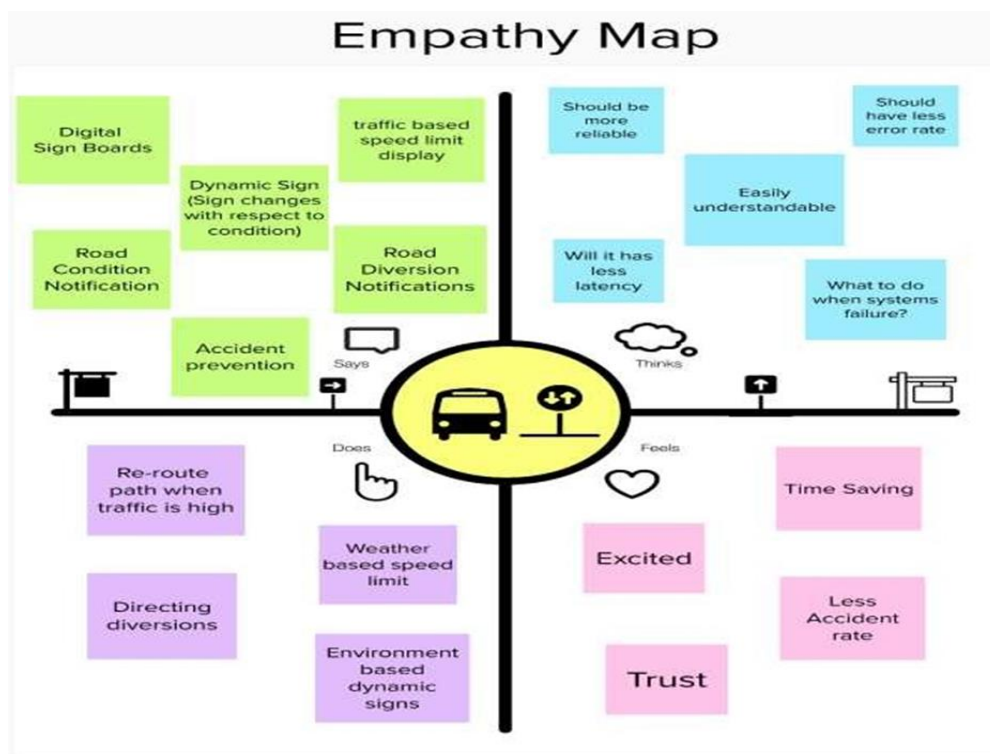


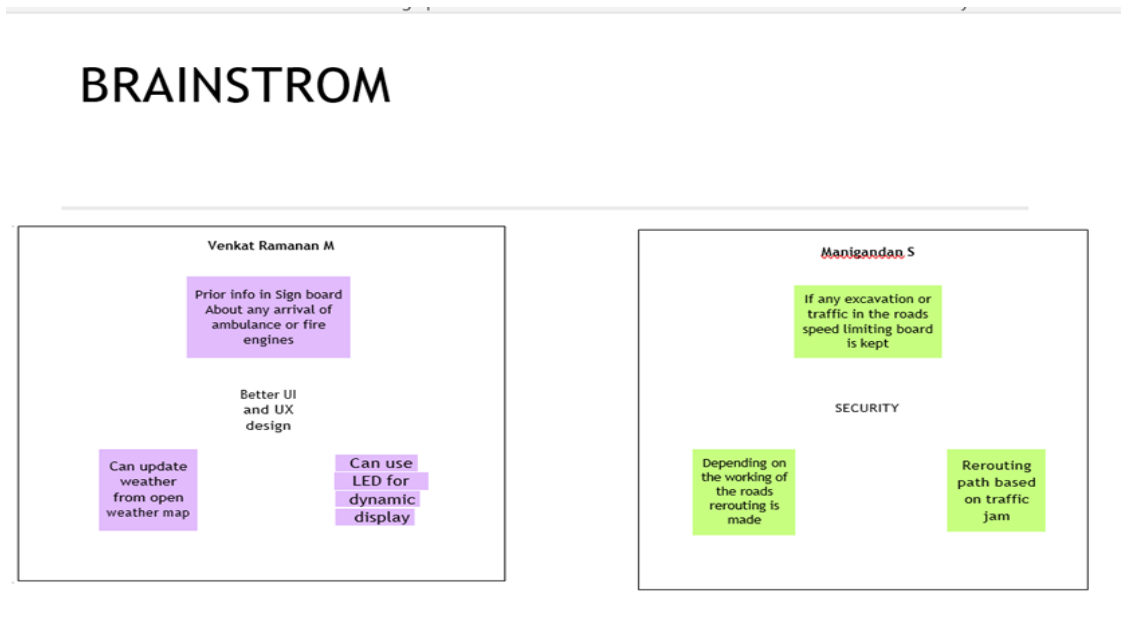
Figure 3.1 Empathy Map

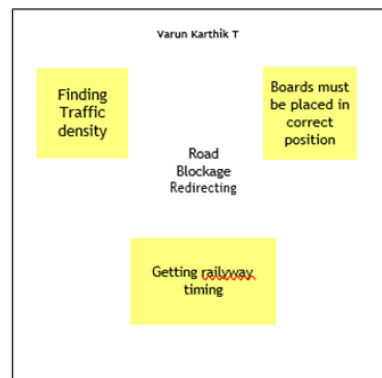
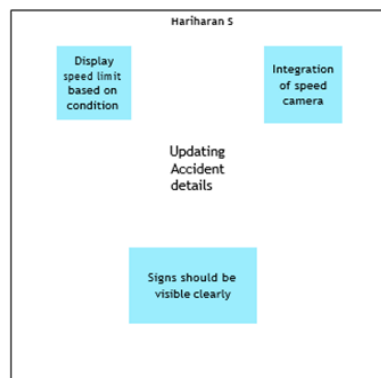
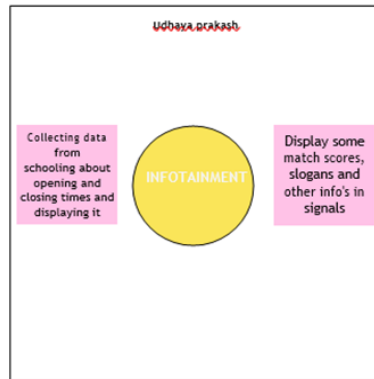
#### 3.2 IDEATION AND BRAINSTROMING

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A group of people are frequently gathered for a brainstorming session to generate either fresh, general ideas or solutions to specific problems or circumstances. The main distinction between ideation and brainstorming is that

brainstorming is nearly often done in groups, ideation is typically seen as being more of a solitary endeavor. Both brainstorming and ideation are techniques developed to generate fresh, insightful notions, ideas, and perceptions. They are also ways to imagine fresh frameworks and tackle systemic issues.

The brainstorm of Signs with Smart Connectivity for Better Road Safety is shown in Figure 3.2



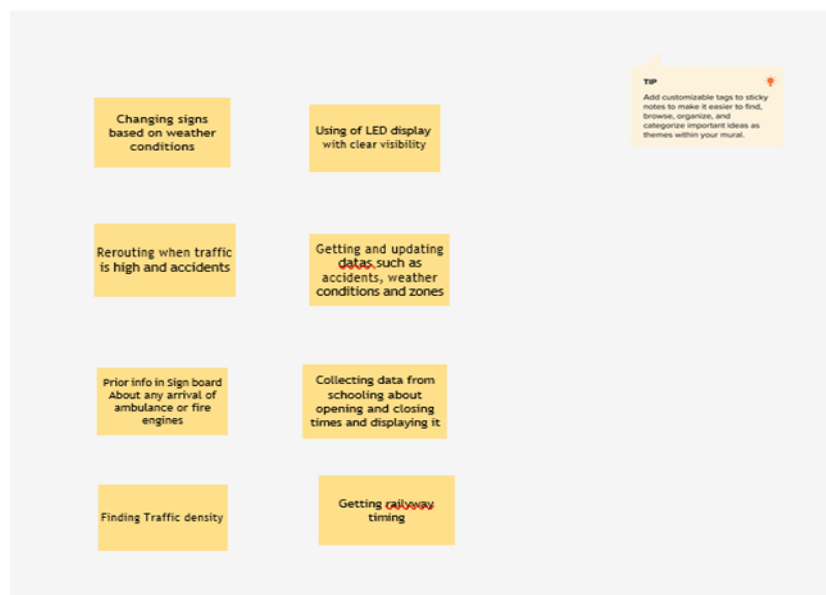


3

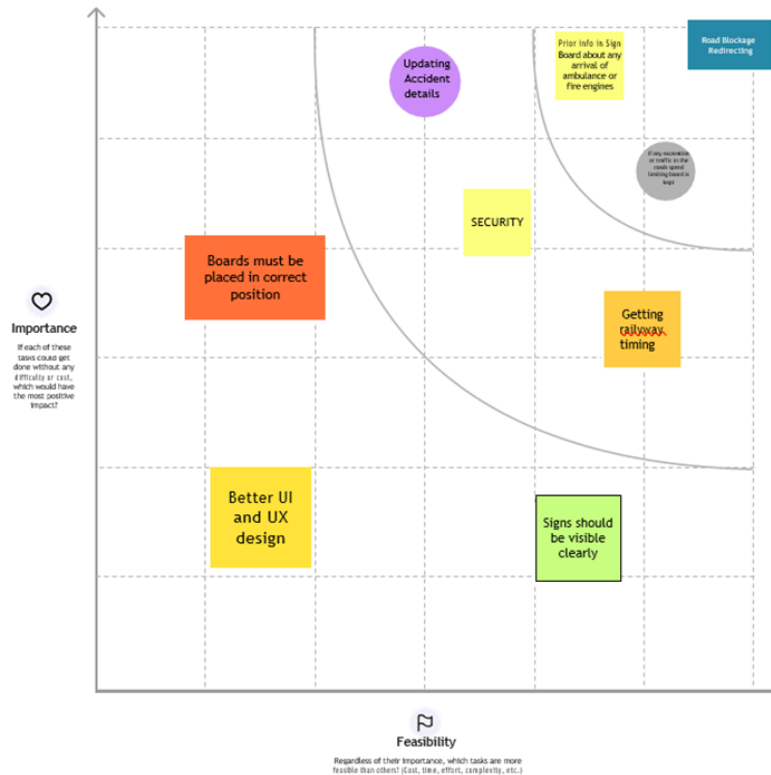
### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes







**Figure 3.2 Brainstorm**

### 3.3 PROPOSED SOLUTION

#### 3.3.1 PROBLEM TO BE SOLVED

Sign boards in the roads are using the traditional approach requires a lot of time to the information in the board and changing it after sometimes would be not useful, and it is very challenging for the vehicle to check speed limit for each and every place if the sign are not clear.

#### 3.3.2 IDEA / SOLUTION DESCRIPTION

The conventional sign board does not create more effect on the roads which is easily avoid. Instead creating an smart connectivity sign board which indicates the current situation of the road which include traffic around the area, accident zone, any institute around the area, hospital. This can achieve by sensing the condition of the area in an regular interval, creating an time based signal for the institute near area.

### **3.3.3 NOVELTY / UNIQUENESS**

The sign board are made digital instead of static. Considering different factors different dynamic sign can be displayed in the sign board.

### **3.3.4 SOCIAL IMPACT / CUSTOMER SATISFACTION**

Based on weather, traffic diversion signs, guide signs and warning signs are displayed to the public to avoid and decrease accident rate.

### **3.3.5 BUSINESS MODEL (REVENUE MODEL)**

The large scale implementation of smart sign board will give better road safety and decrease the accident rate.

## **3.4 PROBLEM SOLUTION FIT**

The phrase "problem-solution fit" simply indicates that an issue has been identified with a client and that the client's problem has been addressed by the solution that has been discovered. A crucial step in achieving product-market fit is finding the right problem-solution match. Below is a list of the structure of the problem-solution fit.

**Novelty:** The sign board are made digital instead of static. Considering different factors different dynamic sign can be displayed in the sign board.

**Feasibility of idea:** The conventional sign board does not create more effect on the roads which is easily avoid. Instead creating smart connectivity sign board which indicates the current situation of the road which include traffic around the area, accident zone, any institute around the area, hospital. This can achieve by sensing the condition of the area in an regular interval, creating an time based signal for the institute near area.

**Business model:** The large scale implementation of smart sign board will give better road safety and decrease the accident rate.

**Social impact:** Based on weather, traffic diversion signs, guide signs and warning signs are displayed to the public to avoid and decrease accident rate.

**Scalability:** The existing sign boards are replaced into digital sign boards and placed in the same location with better position and visibility. Traffic diversion are implemented with better route. Thus, this model reduces the travel time accident can be prevented.

The problem solution fit for Signs with Smart Connectivity for Better Road Safety is shown in Figure 3.3

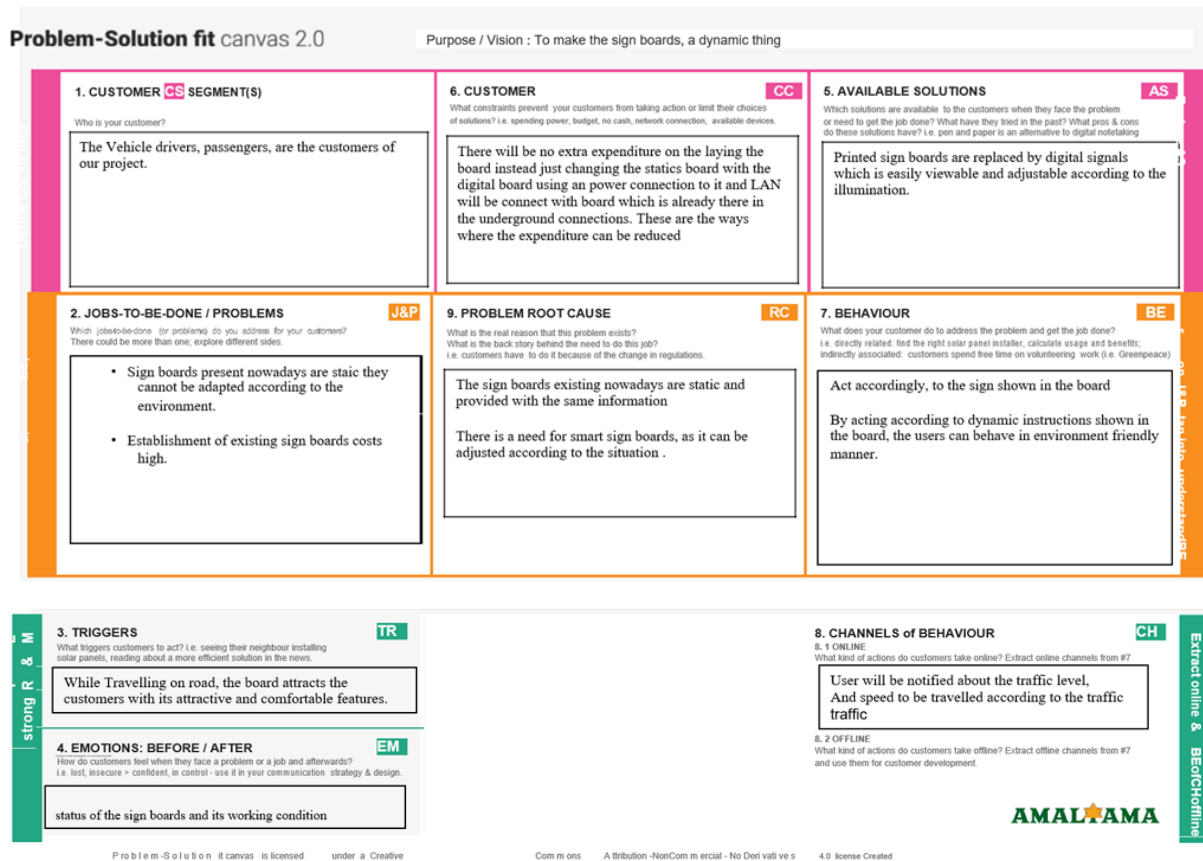


Figure 3.3 Problem Solution fit

## CHAPTER 4

### REQUIREMENT ANALYSIS

#### 4.1 FUNCTIONAL REQUIREMENTS

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

The following table 4.1 shows the functional requirements Signs with Smart Connectivity for Better Road Safety

**Table 4.1 Functional Requirements**

FR No.	Functional Requirement(Epic)	Sub Requirement(Story/Sub-Task)
FR-1	User Visibility	Sign Boards should be made with LED's which are bright colored and are placed in a position where it is attracted by the drivers but it should also not be too bright and distracting. The Board should not in a place which hides the part of road thus blinding <u>cause</u> it may lead to accidents.
FR-2	User Understanding	For better understanding of the driver, the signs should be repeated and it should in an order where the driver understands it properly. The sign should be in a symbol model thus the driver will understand without spending more time on it.
FR-3	User Convenience	The display should be big enough that it should even be visible from far distance clearly.

## 4.2 NON-FUNCTIONAL REQUIREMENTS

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like Portability, Security, Maintainability, Reliability, Scalability, Performance, Reusability, Flexibility.

The following table 4.2 shows the Non-Functional Requirements for Signs with Smart Connectivity for Better Road Safety

**Table 4.2 Non -Functional Requirement**

FR No.	Non-Functional Requirements	Description
NFR-1	<b>Usability</b>	It should be able to update in a time interval using the sensor and based on the data provided and it should be easily upgradable because of the technical advancement <u>so</u> it will be feasible for the interpreter to implement the changes easily.
NFR-2	<b>Security</b>	The sign Board should be highly <u>secured</u> , <u>no</u> one should have a chance to access which may cause the sign board to <u>give wrong signs</u> .
NFR-3	<b>Reliability</b>	It should able to produce proper sign irrespective of the cause and sign board should not produce error.
NFR-4	<b>Performance</b>	It should be able to automatically update itself when certain weather or traffic problem occurs.
NFR-5	<b>Availability</b>	It should be available 24/7 so that it can be beneficial to the customer <u>i.e</u> the driver.
NFR-6	<b>Scalability</b>	It should be able to easily change and upgrade according to change and need in requirement.

### 4.3 CUSTOMER JOURNEY

A customer journey map is a visual storyline of every engagement a customer has with a service, brand, or product. The creation of a journey map puts the organization directly in the mind of the consumer, so they can see and understand their customer's processes, needs, and perceptions.

The customer journey for Signs with Smart Connectivity for Better Road Safety is shown in Figure 4.3

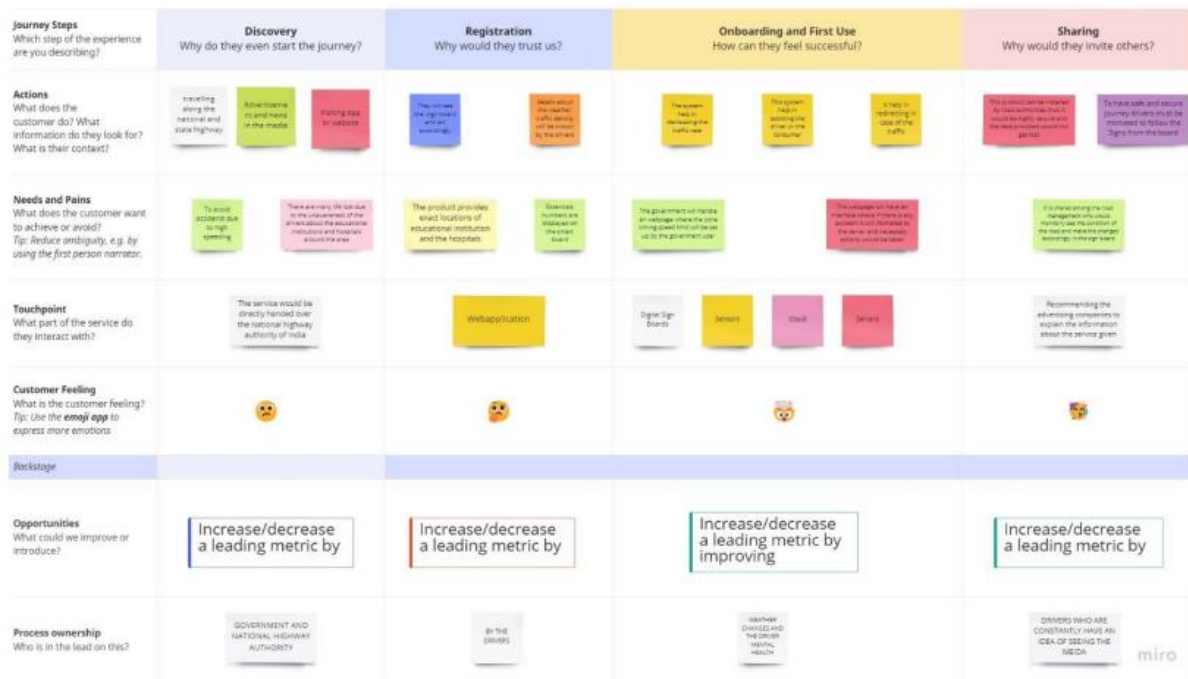


Figure 4.3 Customer Journey

## **CHAPTER 5**

### **PROJECT DESIGN**

#### **5.1 DATA FLOW DIAGRAMS**

##### **5.1.1 DATA FLOW**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems. There are four main elements of a DFD — external entity, process, data store, and data flow.

##### **External entity**

An external entity, which are also known as terminators, sources, sinks, or actors, are an outside system or process that sends or receives data to and from the diagrammed system. They’re either the sources or destinations of information, so they’re usually placed on the diagram’s edges. External entity symbols are similar across models except for Unified, which uses a stick-figure drawing instead of a rectangle, circle, or square.

##### **Process**

Process is a procedure that manipulates the data and its flow by taking incoming data, changing it, and producing an output with it. A process can do this by performing computations and using logic to sort the data, or change its flow of direction. Processes usually start from the top left of the DFD and finish on the bottom right of the diagram.

**Data store**

Data stores hold information for later use, like a file of documents that's waiting to be processed. Data inputs flow through a process and then through a data store while data outputs flow out of a data store and then through a process.

**Data flow**

Data flow is the path the system's information takes from external entities through processes and data stores. With arrows and succinct labels, the DFD can show the direction of the data flow.





**Figure 5.1 Data flow diagram**

The data flow diagram for Signs with Smart Connectivity for Better Road Safety is shown in Figure 5.1

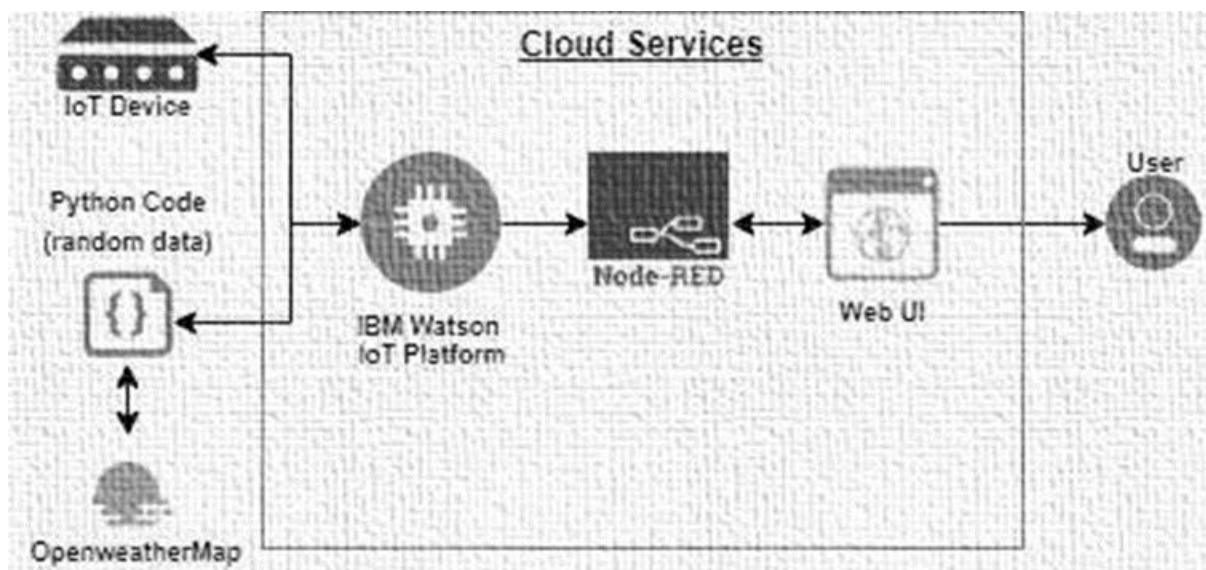
## 5.2 SOLUTION AND TECHNICAL ARCHITECTURE

### 5.2.1 DATA FLOW

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

The figure 5.2 shows the data flow for Signs with Smart Connectivity for Better Road Safety.



**Figure 5.2 Technical Architecture**

### 5.2.2 COMPONENTS AND TECHNOLOGIES

The components and technologies used for Signs with Smart Connectivity for Better Road Safety is shown in following table 5.1

**Table 5.1 Components & Technologies**

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App.	HTML, CSS, JavaScript / Angular Js
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson IoT Platform
4.	Application Logic-3	Logic for a process in the application	NODE-RED service
5.	Application Logic-4	Logic for a process in the application	Open Weather Map
6.	Database	Data Type, Configurations etc.	MySQL
7.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant
8.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
9.	External API-1	Purpose of External API used in the application	IBM Weather <a href="#">API</a> ,MQTT
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server <a href="#">Configuration</a> .	Local, Cloud Foundry, Kubernetes

### 5.2.3 APPLICATION CHARACTERISTICS:

The applications used for Signs with Smart Connectivity for Better Road Safety is shown in following table 5.2

**Table 5.2 Application Characteristics**

<u>S.No</u>	Characteristics	Description	Technology
1.	Open-Source Frameworks	There <u>are</u> no open-source framework in this application.	<u>Python</u> <u>NODE-RED</u>
2.	Security Implementations	List all the security / access controls implemented, use of firewalls	Encryption
3.	Scalable Architecture	User are provided with traffic symbol <u>online</u> . Give awareness to road rules.	IBM cloud
4.	Availability	Controller <u>recommendation</u> , <u>Symbol</u> ,Road rules , accident provided zones are available in applications.	IBM <u>Waston</u> Assistant
5.	Performance	Artificial Intelligence (AI) such as Machine Learning (ML) algorithms are very helpful to improve the performance of the overall road safety management.	AI such as Machine learning

### 5.3 USER STORIES

In software development and product management, a user story is an informal, natural language description of one or more features of a software system. A user story is a tool used in Agile software development to capture a description of a software feature from an end-user perspective. A user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement.

The user story for Smart solutions for the railway system is shown in following table 5.3

**Table 5.3 User stories**

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	Medium	Sprint-1
		USN-2	As a user, I can get the information of speed limits of the road, traffic level and route information.	I can receive speed limit constraints	High	Sprint-1
		USN-3	As a user, I can get the updates of route in case of traffic or any fatal situations like accidents or natural disasters for the way I travelling, based on that I can get better navigation	I can get traffic updates while I travelling in the route	High	Sprint-2
	Login	USN-4	As a user, I can log in or logging out using credentials.	I can access the application using my account.	Low	Sprint-1
	Dashboard	USN-5	As a user, the interface should be easy to access and without any problem	I can access the application easily with no problem	High	Sprint-1
Customer (Web user)	Data generation	USN-6	As a user, I can get utilize the weather information using web app when the weather conditions suddenly change.	I can get the updates of the weather conditions via the web app	High	Sprint-2
Customer Care Executive	Service Providing	USN-7	As a customer care executive, the app should work 24x7, if the server is down, it should be rectified soon Signs must be clearly visible.	I can access the app anytime and anywhere	High	Sprint-2

## CHAPTER 6

### PROJECT PLANNING AND SCHEDULING

#### 6.1. SPRINT PLANNING AND ESTIMATION

Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

The sprint is a set period of time where all the work is done. However, before leap into action it is necessary to set up the sprint. It needs to decide on how long the time box is going to be, the sprint goal, and where it is going to start. The sprint planning session kicks off the sprint by setting the agenda and focus. If done correctly, it also creates an environment where the team is motivated, challenged, and can be successful.

The Table 6.1 shows the Milestones & Tasks for Signs with Smart Connectivity for Better Road Safety.

**Table 6.1. Milestones & Task**

<b>TITLE</b>	<b>DESCRIPTION</b>	<b>DATE</b>
<b>Literature Survey &amp; Information Gathering</b>	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	28 SEPTEMBER 2022
<b>Prepare Empathy Map</b>	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	25 SEPTEMBER 2022
<b>Ideation</b>	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	26 SEPTEMBER 2022
	document, which includes the	

<b>Proposed Solution</b>	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact	29 SEPTEMBER 2022
<b>Problem Solution Fit</b>	Prepare problem - solution fit	30 SEPTEMBER 2022
<b>Solution Architecture</b>	document. document.	28 SEPTEMBER 2022
<b>Customer Journey</b>	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	20 OCTOBER 2022
<b>Functional Requirement</b>	Prepare the functional requirement document.	8 OCTOBER 2022
<b>Data Flow Diagrams</b>	Draw the data flow diagrams and submit for review.	9 OCTOBER 2022
<b>Technology Architecture</b>	Prepare the technology architecture diagram.	10 OCTOBER 2022
<b>Prepare Milestone &amp; Activity List</b>	Prepare the milestones & activity list of the project.	21 OCTOBER 2022
<b>Project Development - Delivery of Sprint-1, 2, 3 &amp; 4</b>	Develop & submit the developed code by testing it.	IN PROGRESS..

**Table 6.2 sprint planning**

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User Story / Task</b>	<b>Story Points</b>	<b>Priority</b>	<b>Team Members</b>
Sprint-1	Initializing resources	Create and initialize accounts in various public APIs like OpenWeatherMap API, Node-red, IBM cloud.	1	Low	Venkat Ramanan, Manigandan
Sprint-1	Local/Software run	Develop the Python code in which the test cases are provided like Weather, Traffic density, School timings	1	Medium	Venkat Ramanan, Varun Karthick
Sprint-2	Dump the server/software to cloud	Dump the code from Sprint 1 to cloud so it can be accessed from anywhere	2	Medium	Udhayaprakash, Hariharan, Manigandan
Sprint-3	Initializing hardware	Integrate the hardware which is able to access the cloud functions and change the output based on the processed output.	2	High	Varun Karthick, Hariharan, Venkat Ramanan
Sprint-4	UI/UX optimization and debugging	Optimize the hardware and software based on the error and precision of the observed output and make the system efficient	2	Low	Hariharan, Udhayaprakash, Varun Karthick

The Table 6.2 shows the sprint planning and estimation for Signs with Smart Connectivity for Better Road Safety.

## 6.2 SPRINT DELIVERY SCHEDULE

**Table 6.3 sprint delivery schedule**

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



## CHAPTER 7

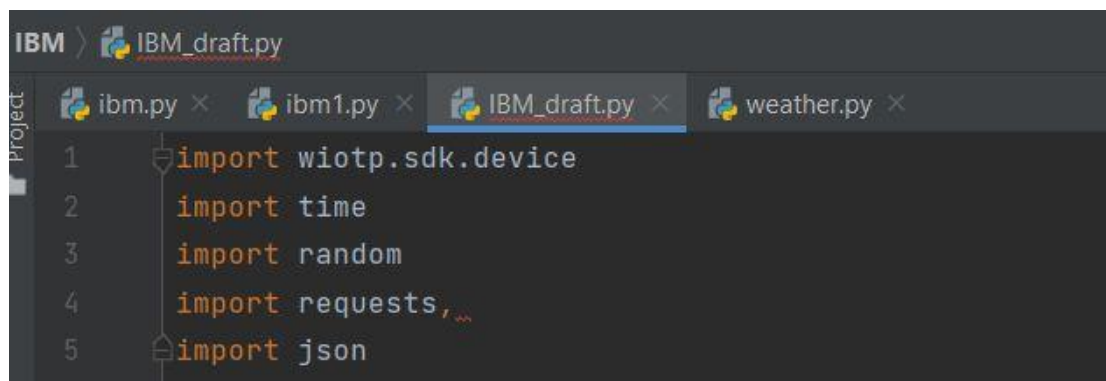
### CODING AND SOLUTIONING

#### 7.1 FEATURES

- Warning and Alert will be posted on the sign if there has been any accident around the areas and Alerts during the school timing to slow down the vehicle will be posted in the sign board
- Online database using Google Sheets to store the results of the sensors
- Using weather API to notify the current temperature, humidity, whether it is raining or very sunny will be intimated to the cloud and the cloud server will provide immediate response which is displayed in the sign board.

**Basic explanation:**

**Libraries required:**

A screenshot of a code editor window. The title bar shows 'IBM' and the file name 'IBM\_draft.py'. The editor has four tabs: 'ibm.py', 'ibm1.py', 'IBM\_draft.py' (which is active), and 'weather.py'. The code in the active tab is as follows:

```
1 import wiotp.sdk.device
2 import time
3 import random
4 import requests, ...
5 import json
```

**Fig 7.1 Libraries required**

**Wiotp.sdk.device:**

Wiotp.sdk.device is a python SDK for IBM Watson IoT platform. IBM Watson IoT service provides a platform that lets you manage, communicate with, and consume data from connected devices and gateway.

**Random:**

Sometimes we want the computer to pick a random number in a given range, pick a random element from a list, pick a random card from a deck, flip a coin, etc. The random module provides access to functions that support these types of operations.

In this case random is used to generate the random values between specified float values to recreate sensor data.

**Requests:**

The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).

**JSON:**

JSON is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays. It is a common data format with diverse uses in electronic data interchange, including that of web applications with servers.

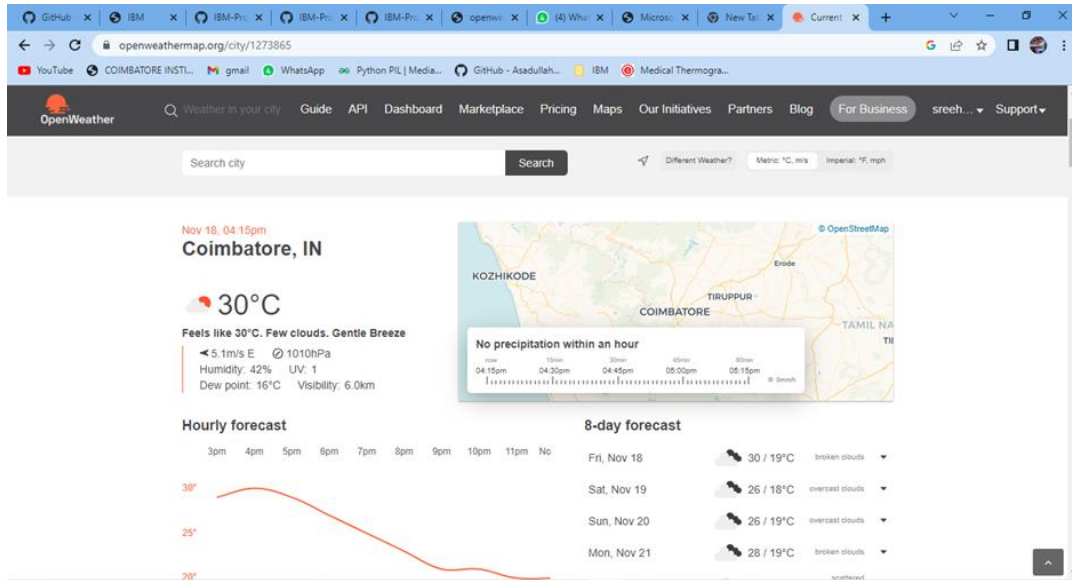
**Time:**

Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. In this program the time library is used to get the current time so that the data can be stored using time stamps.

**Open weather API:**

**Fig 7.2 Open weather API**

Open Weather Map is an online service, owned by OpenWeather Ltd, that provides global weather data via API, including current weather data, forecasts, nowcasts and historical weather data for any geographical location. The company provides a minute-by-minute hyperlocal precipitation forecast for any location.



**Fig 7.3 Open Weather Website**

Open weather api is used because this is one of the most reliable weather tracking API among all others and agriculture is the top most sector the open weather api is used. This API also provides a free tier with about 60 API calls per minute that is one api call for every second which is more than enough for our use case.

### **Getting weather data using API:**

Get\_weather function api key and base url of the open weather map api is used to fetch the weather data and parameters such as temperature, humidity, pressure and a small weather description.

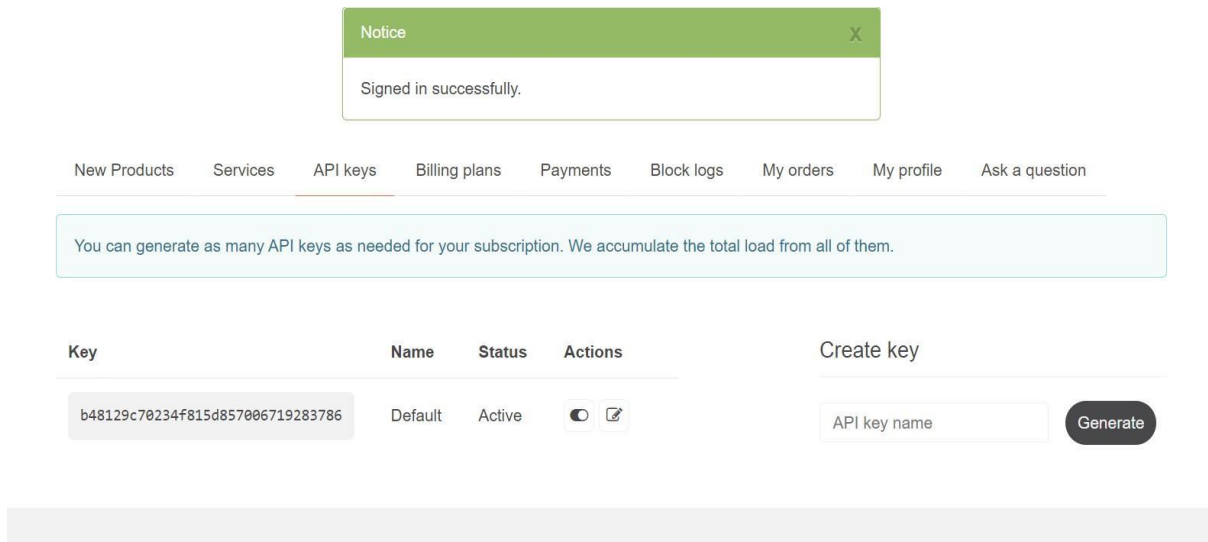
This function returns the weather data to the main function where the parameters are checked with their thresholds before calling the send sms function.

```
iotopen.py - C:\Users\Welcome\AppData\Local\Programs\Python\Python311\iotopen.py (3.8.2)
File Edit Format Run Options Window Help

import wiotp.sdk.device
import time
import random
import requests,json
import datetime
myConfig = {
    "identity": {
        "orgId": "tmwrsrv",
        "typeId": "Sprint",
        "deviceId": "sprint12"
    },
    "auth": {
        "token": "XoMwjrjwBijreIuFK"
    }
}
)
CITY = "Coimbatore"
API_KEY = "9111b726e6aa664188c5a2924f15f78e"
URL= "https://api.openweathermap.org/data/2.5/weather?q=Coimbatore,%20IN&appid=9111b726e6aa664188c5a2924f15f78e"
response = requests.get(URL)
if response.status_code == 200:
    data = response.json()
    main = data['main']
    temp = round(main['temp'] - 273,2)
    humy = main['humidity']
    pres = main['pressure']
    rept = data['weather']
    report = rept[0]['description']
    time = datetime.datetime.now()
    morning = time.replace(hour=11, minute=59, second=0, microsecond=0)
    if time <= morning:
        me = '8.30 AM - 9.30 AM'
    else:
        me = '3.45 PM - 5.00 PM'
def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    templ=temp
    hum=humy
    pre = pres
    wea = report
    myData={'location':CITY,'temperature':temp, 'humidity':hum, 'pressure':pre, 'weather_report':wea, 'Schooltiming':me}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
```

**Fig 7.4 Get Weather function**



**Fig 7.5 Key values and the status of the open weather API**

## Importing Packages and Cloud Credential



**Fig 7.6 Importing Packages and Cloud Credentials**

## Connecting to Watson using WIOTP Package

```
17 def myCommandCallback(cmd):
18     print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
19     m=cmd.data['command']
20     random.random()
21     client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
22     client.connect()
```

**Fig 7.7 Connecting to Watson using WIOTP Package**

## Getting Values from OpenWeather API

```
24 while True:
25     BASE_URL = "https://api.openweathermap.org/data/2.5/weather?"
26     CITY = "Coimbatore"
27     API_KEY = "46faa4ab6fede1d9ae549b90d91253f2"
28     URL = BASE_URL + "q=" + CITY + "&appid=" + API_KEY
29     response = requests.get(URL)
30
31     if response.status_code == 200:
32         data = response.json()
33         main = data['main']
34         temp = round(main['temp'] - 273, 2)
35         humy = main['humidity']
36         pres = main['pressure']
37         rept = data['weather']
38         report = rept[0]['description']
```

**Fig 7.8 Getting Values from OpenWeather API**

## Weather Related Sign Board conditions

```
38     temp1=temp
39     if(temp1 < 5):
40         alert = "Snow fall may occur"
41     hum=humy
42     if(hum>85):
43         alert = "Probablity of raining is high. Take rain coat with you"
44     pre = pres
45     wea = report
46     if(wea == "clear sky"):
47         wea_alt1 = "HAPPY JOURNEY!!!"
48         spd_alt1 = "GO @ < 80Kmph!!!"
49     elif(wea == "few clouds" or "haze"):
50         wea_alt1 = "CLOUDY DAY :)"
51         spd_alt1 = "GO @ < 60Kmph!!!"
52     elif (wea == "overcast clouds"):
53         wea_alt1 = "RAIN MAY COME!!"
54         spd_alt1 = "GO @ < 30Kmph!!!"
55     elif(wea == "shower rain"):
56         wea_alt1 = "SLIPPERY ROAD :("
57         spd_alt1 = "GO @ < 20Kmph!!!"
58     elif(wea == "rain" or "moderate rain"):
59         wea_alt1 = "SLIPPERY ROAD :("
60         spd_alt1 = "GO @ < 20Kmph!!!"
61     elif(wea == "thunderstorm"):
62         wea_alt1 = "HEAVY RAIN!!!"
63         spd_alt1 = "GO @ < 10Kmph!!!"
64     elif (wea == "snow"):
65         wea_alt1 = "SNOW ON ROAD!!!"
66         spd_alt1 = "GO @ < 10Kmph!!!"
67     elif(wea == "mist"):
68         wea_alt1 = "TURN ON FOG LAMP"
69         spd_alt1 = "GO @ < 20Kmph!!!"
70     else:
71         wea_alt1 = "HAPPY JOURNEY :)"
72         spd_alt1 = "GO @ < 80Kmph!!!"
```

Fig 7.9 Weather Related Sign Board conditions

## School Time Warning Condition

```
73     xyz = time.ctime()
74     ctime = int(xyz[11:13])
75     if (ctime > 8 and ctime < 10) or (ctime > 15 and ctime < 18):
76         me = 'SCHOOL TIMING GO SLOW SPEED LIMIT: 15Kmph'
77         spd_alt1 = "GO @ < 15Kmph!!!"
78     else:
79         me = xyz[11:19]
```

Fig 7.10 School Time Warning Condition

## Publishing into IBM Watson Cloud

```
80 myData={'location':CITY,'temperature':temp, 'humidity':hum, 'pressure':pre,
81        'weather_report':wea,'wea_alt':wea_alt1,'spd_alt':spd_alt1,'schl_tmng':me}
82 client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
83 print("Published data Successfully: %s", myData)
84 client.commandCallback = myCommandCallback
85 time.sleep(1)
86 client.disconnect()
```

**Fig 7.11 Publishing into IBM Watson Cloud**

## Arduino Setup

```
#include <string>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include "ArduinoJson.h"
#include <HTTPClient.h>
LiquidCrystal_I2C lcd(0x27,20,4);
const char* ssid      = "Vr";
const char* password = "venkat123";
String payload;
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(50);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    lcd.begin();
    lcd.backlight();
}
```

**Fig 7.12 Arduino Setup**



## Getting values from NODE-RED

```
void link(void) {
    HTTPClient http;
    String url="https://node-red-fjaeq-2022-11-12.eu-gb.mybluemix.net/data";
    //Serial.print(url);
    Serial.print("Making a request");
    http.begin(url.c_str()); //Specify the URL and certificate
    http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
    int httpCode = http.GET();
    if (httpCode > 0) { //Check for the ing code
        payload = http.getString();
        Serial.println(httpCode);
        Serial.println(payload);
    }
    else {
        Serial.println("Error on HTTP request");
    }
    http.end();
}
```

**Fig 7.13 Getting values from NODE-RED**

## Parsing the values

```
void loop() {
    // put your main code here, to run repeatedly:
    link();

    //JSON
    Serial.println("Parsing start: ");

    //char JSONMessage[] = payload; //Original message

    StaticJsonBuffer<1000> JSONBuffer; //Memory pool
    JsonObject& parsed = JSONBuffer.parseObject(payload); //Parse message

    if (!parsed.success()) { //Check for errors in parsing

        Serial.println("Parsing failed");
        delay(5000);
        ;
    }

    const char * sensorType = parsed["location"]; //Get sensor type value
    const char * sensorType1 = parsed["weather_report"];
    const char * sensorType2 = parsed["wea_alt"];
    const char * sensorType3 = parsed["schl_tmng"];
    const char * sensorType4 = parsed["spd_alt"];
    int value = parsed["humidity"]; //Get value of sensor measurement
    double value1 = parsed["temperature"];
    int value2 = parsed["pressure"];
```

**Fig 7.14 Parsing the values**

## Displaying the parsed values in LCD

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Welcome IBM IOT");
lcd.setCursor(0, 1);
lcd.print("Loc: ");
lcd.print(sensorType);
delay(1500);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp: ");
lcd.print(value1);
lcd.print((char)223);
lcd.print("C");
lcd.setCursor(0, 1);
lcd.print("Hum: ");
lcd.print(value);
lcd.print("%");
delay(1500);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Wea: ");
lcd.print(sensorType1);
lcd.setCursor(0, 1);
lcd.print("Time: ");
lcd.print(sensorType3);
delay(1500);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(sensorType2);
lcd.setCursor(0, 1);
lcd.print(sensorType4);
}
```

---

**Fig 7.15** Displaying the parsed values in LCD

## CHAPTER 8

### TESTING

#### 8.1 TESTCASES

The test cases for the mobile application is listed below in the table 8.1. All the features of the mobile application is tested.

**Table 8.1 Test cases for the mobile application**

5	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data
6	Dashboard_TC_001	Function	Home page	School Timing	Knowing about the school time	1. Open the Application	NIL
7	Dashboard_TC_002	Function	Home page	Speed	Knowing about the weather	1. Open the Application	NIL
8	Dashboard_TC_003	Function	Home page	Weather	NIL	1. Open the Application	NIL
9	Dashboard_TC_004	Function	Weather Report	Weather	NIL	1. Open the Application 2. Click Weather Report	NIL
10	Dashboard_TC_005	Function	Weather Report	Back to Front Page	NIL	1. Open the Application 2. Click Weather Report 3. Press Back	NIL

5	Test case ID	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
6	Dashboard_TC_001	Alert during school time	Working as expected	Pass	.	Y	.	Hariharan S
7	Dashboard_TC_002	Speed Limit	Working as expected	Pass	.	Y	.	Manigandan S
8	Dashboard_TC_003	Weather Condition	Working as expected	Pass	.	Y	.	Varun Karthik T
9	Dashboard_TC_004	Weather Report of that Location	Working as expected	Pass	.	Y	.	Venkat Ramanan M
10	Dashboard_TC_005	Back to Home Page	Working as expected	Pass	.	Y	.	Udhayaprakash T

## CHAPTER 9

### RESULTS

#### 9.1 APPLICATION

##### Weather Window

The weather window displaying location, weather, temperature, pressure and humidity is shown in figure 9.1

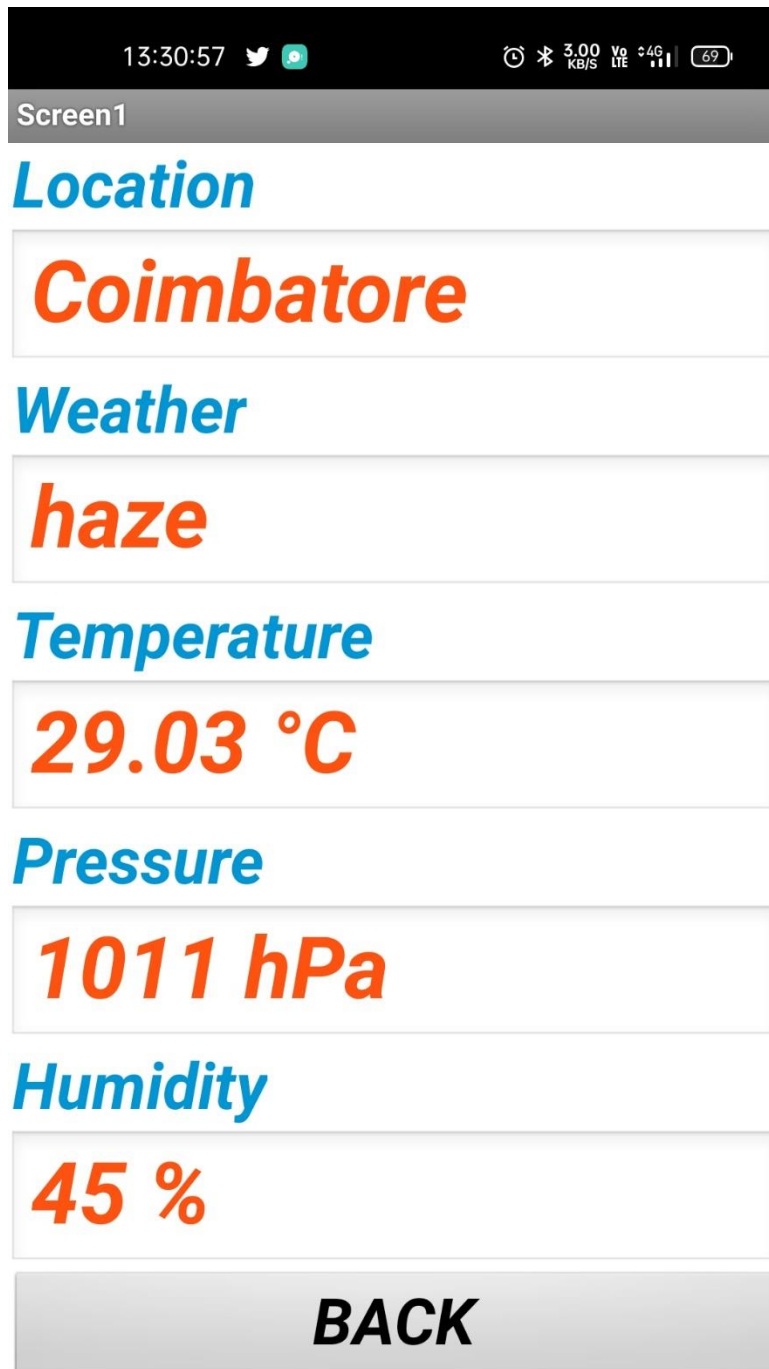


Fig 9.1 Weather window

## Home page

The home page showing the details of time, weather and speed limit for cloudy weather is show in figure 9.2

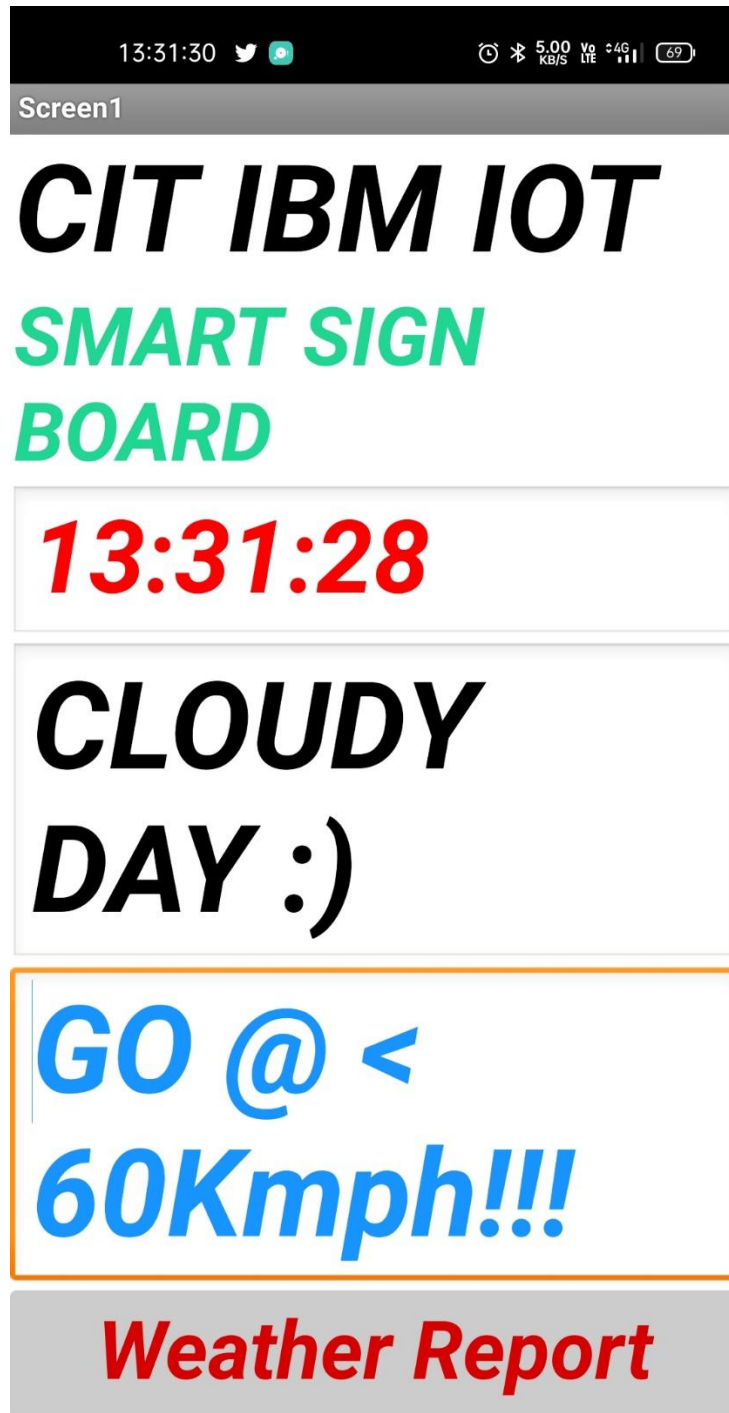


Fig 9.2 Home page displaying Cloudy Weather

The figure is the home page showing the details of time, weather and speed limit for school timing is shown in figure 9.3.

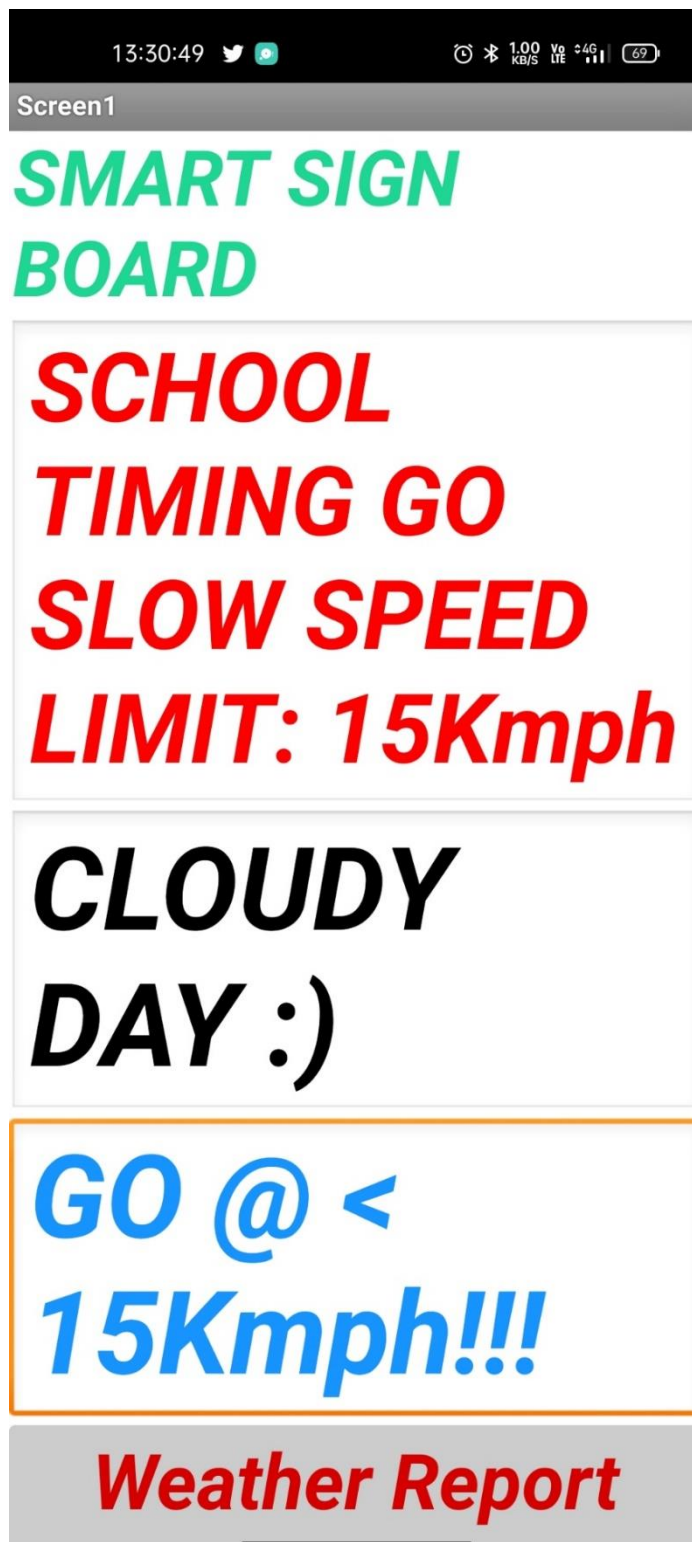


Fig 9.3 Home page displaying school hour

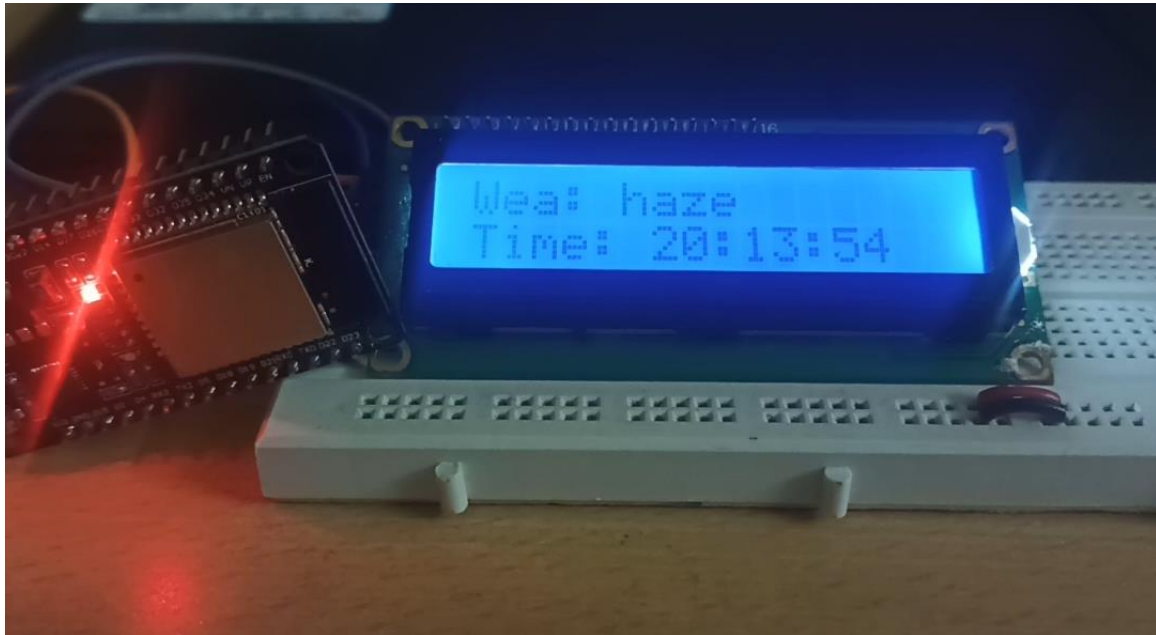
## 9.2 HARDWARE

The output from esp32 after fetching location is shown in figure 9.4



**Fig 9.4 LCD Displaying location**

The output from esp32 after fetching weather and time is shown in figure 9.5



**Fig 9.5 LCD Displaying weather and time**

The output from esp32 after fetching speed limit is shown in figure 9.6



**Fig 9.6 LCD displaying speed limit**



## **CHAPTER 10**

### **ADVANTAGES AND DISADVANTAGES**

- Easier to understand - Drivers and other road users receive signals from traditional sign boards regarding hazard or other information. Smart sign boards, on the other hand, are simpler to understand because they not only alert drivers to threats but also enable them to take immediate action. It gives both drivers and traffic controllers real-time messages. The conventional suppliers of road safety sign boards are frequently let down because of this.
- Provides information in advance - Advanced information on the upcoming road conditions, orders, stop warnings, and oncoming traffic warnings are provided in real-time via smart traffic sign boards. Although they operate significantly differently from modern sign boards, traditional sign boards are nevertheless useful for warning, guiding, and instructing drivers. The driver must exert extra effort to pay attention to the messages.
- Relies on technology - Smart sign boards can be used to develop an intelligent transportation system since they utilize a number of cutting-edge technologies, including Bluetooth Low Energy. It uses every opportunity to tell drivers about information. Traditional sign boards, on the other hand, must be perfectly designed and placed so as to catch the attention of drivers.
- If the vehicle violates the traffic rules then the system is not capable of identification of the vehicle and the energy management for the system is more and the implementation of conventional energy resources takes more investment.

## **CHAPTER 11**

### **CONCLUSION**

Propose framework converts the normal traffic sign boards to Smart sign boards that can dynamically changes the speed limit depending upon the weather conditions and school timings and rush hour. It is as an input for the smart traffic management for smart cities. Normal sign boards are many at times missed by human attention and accidents may happen due to sudden occurrence of such unexpected feature of the roads. This work is useful for intelligent traffic management and driving assistance for the new driver for a particular area or city. As a result the accidents will be lesser due sudden occurrence of unusual features of the road.

## **CHAPTER 12**

### **FUTURE SCOPE**

Future scope of this work, the system may be enabled with GPS and Machine Learning mechanism to record the Sign Boards of the daily route in the system, the work also can be done to limit the frequency interferences due to multi objects communication for smart-cities. The RFID tags may be replaced with Glyphs to be implemented with Augmented Reality. Work can also be extended towards reducing the latency to read and process the message as per the speed of the vehicle.

## CHAPTER 13

### APPENDIX 1

#### SOURCE CODE

##### Python Code

```
import wiotp.sdk.device
import time
import random
import requests

myConfig = {
    "identity": {
        "orgId": "10ilxy",
        "typeId": "temphum",
        "deviceId": "ticece"
    },
    "auth": {
        "token": "&1of4gQxPXz-JD(0?E"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
    random.random()
    client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
    client.connect()

while True:
    BASE_URL = "https://api.openweathermap.org/data/2.5/weather?"
    CITY = "Coimbatore"
    API_KEY = "46faa4ab6fede1d9ae549b90d91253f2"
    URL = BASE_URL + "q=" + CITY + "&appid=" + API_KEY
    response = requests.get(URL)
    if response.status_code == 200:
        data = response.json()
        main = data['main']
        temp = round(main['temp'] - 273, 2)
        humy = main['humidity']
        pres = main['pressure']
        rept = data['weather']
        report = rept[0]['description']
```

```

temp1=temp
if(temp1 < 5):
    alert = "Snow fall may occur"
hum=humy
if(hum>85):
    alert = "Probablity of raining is high. Take rain coat with you"
pre = pres
wea = report
if(wea == "clear sky"):
    wea_alt1 = "HAPPY JOURNEY!!!"
    spd_alt1 = "GO @ < 80Kmph!!!"
elif(wea == "few clouds" or "haze"):
    wea_alt1 = "CLOUDY DAY :)"
    spd_alt1 = "GO @ < 60Kmph!!!"
elif (wea == "overcast clouds"):
    wea_alt1 = "RAIN MAY COME!!"
    spd_alt1 = "GO @ < 30Kmph!!!"
elif(wea == "shower rain"):
    wea_alt1 = "SLIPPERY ROAD :("
    spd_alt1 = "GO @ < 20Kmph!!!"
elif(wea == "rain" or "moderate rain"):
    wea_alt1 = "SLIPPERY ROAD :("
    spd_alt1 = "GO @ < 20Kmph!!!"
elif(wea == "thunderstorm"):
    wea_alt1 = "HEAVY RAIN!!!"
    spd_alt1 = "GO @ < 10Kmph!!!"
elif (wea == "snow"):
    wea_alt1 = "SNOW ON ROAD!!!"
    spd_alt1 = "GO @ < 10Kmph!!!"
elif(wea == "mist"):
    wea_alt1 = "TURN ON FOG LAMP"
    spd_alt1 = "GO @ < 20Kmph!!!"
else:
    wea_alt1 = "HAPPY JOURNEY :)"
    spd_alt1 = "GO @ < 80Kmph!!!"
xyz = time.ctime()
ctime = int(xyz[11:13])
if (ctime > 8 and ctime < 10) or (ctime > 15 and ctime < 18):
    me = 'SCHOOL TIMING GO SLOW SPEED LIMIT: 15Kmph'
    spd_alt1 = "GO @ < 15Kmph!!!"
else:
    me = xyz[11:19]
myData={'location':CITY,'temperature':temp, 'humidity':hum, 'pressure':pre,
        'weather_report':wea,'wea_alt':wea_alt1,'spd_alt':spd_alt1,'schl_tmng':me}

```

```

    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0,
onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(1)
client.disconnect()

```

## Arduino C Code

```

#include <string>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include "ArduinoJson.h"
#include <HTTPClient.h>
LiquidCrystal_I2C lcd(0x27,20,4);
const char* ssid    = "Vr";
const char* password = "venkat123";
String payload;
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(50);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    lcd.begin();
    lcd.backlight();
}
void link(void) {
    HTTPClient http;
    String url="https://node-red-fjaeq-2022-11-12.eu-gb.mybluemix.net/data";
//Serial.print(url);

```

```

Serial.print("Making a request");
http.begin(url.c_str()); //Specify the URL and certificate
http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
int httpCode = http.GET();
    if (httpCode > 0) { //Check for the ing code
        payload = http.getString();
        Serial.println(httpCode);
        Serial.println(payload);
    }
    else {
        Serial.println("Error on HTTP request");
    }
http.end();
}
void loop() {
    // put your main code here, to run repeatedly:
    link();

    //JSON
    Serial.println("Parsing start: ");

    //char JSONMessage[] = payload; //Original message

    StaticJsonBuffer<1000> JSONBuffer; //Memory pool
    JsonObject& parsed = JSONBuffer.parseObject(payload); //Parse message

    if (!parsed.success()) { //Check for errors in parsing

        Serial.println("Parsing failed");
        delay(5000);
        ;

    }

    const char * sensorType = parsed["location"]; //Get sensor type value
    const char * sensorType1 = parsed["weather_report"];
    const char * sensorType2 = parsed["wea_alt"];
    const char * sensorType3 = parsed["schl_tmng"];
    const char * sensorType4 = parsed["spd_alt"];
    int value = parsed["humidity"]; //Get value of sensor measurement
    double value1 = parsed["temperature"];
    int value2 = parsed["pressure"];
    Serial.print("Location: ");

```

```

Serial.println(sensorType);
Serial.print("Temperature: ");
Serial.println(value1);
Serial.print("Pressure: ");
Serial.println(value2);
Serial.print("Humidity: ");
Serial.println(value);
Serial.print("Weather Report: ");
Serial.println(sensorType1);
Serial.print("Message: ");
Serial.println(sensorType2);
Serial.print("Timing: ");
Serial.println(sensorType3);
Serial.print("Speed Limit: ");
Serial.println(sensorType4);
Serial.println();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Welcome IBM IOT");
lcd.setCursor(0, 1);
lcd.print("Loc: ");
lcd.print(sensorType);
delay(1500);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp: ");
lcd.print(value1);
lcd.print((char)223);
lcd.print("C");
lcd.setCursor(0, 1);
lcd.print("Hum: ");
lcd.print(value);
lcd.print("%");
delay(1500);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Wea: ");
lcd.print(sensorType1);
lcd.setCursor(0, 1);
lcd.print("Time: ");
lcd.print(sensorType3);
delay(1500);
lcd.clear();

```



```
lcd.setCursor(0, 0);  
lcd.print(sensorType2);  
lcd.setCursor(0, 1);  
lcd.print(sensorType4);  
}
```

GITHUB LINK: <https://github.com/IBM-EPBL/IBM-Project-6237-1658825044>

APP URL:

<https://drive.google.com/file/d/1xcG2vBkVIXDUMd4ZoiPCGoZmUdCm2a8P/view?usp=sharing>

VIDEO LINK:

[https://drive.google.com/drive/folders/1pivt1EQmV1wIPFgnAHiYvuxA9dhujjww?usp=share\\_link](https://drive.google.com/drive/folders/1pivt1EQmV1wIPFgnAHiYvuxA9dhujjww?usp=share_link)