



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



**SMART SOLUTION FOR RAILWAYS**

**A PROJECT REPORT**

*Submitted by*

<b>TEAM ID</b>	<b>PNT2022TMID52824</b>
<b>ABISHEAK.S</b>	<b>CITC1904063</b>
<b>ARUNAGIRISH.B</b>	<b>CITC1904069</b>
<b>DHASWANTH..N.G</b>	<b>CITC 1904074</b>
<b>GOKUL.R</b>	<b>CITC 1904077</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 “**SMART SOLUTION FOR RAILWAYS**” is the Bonafide work of **S.ABISHEAK(1904063), B.ARUNAGIRISH(1904069), N.G.DHASWANTH(1904074), R.GOKUL(1904077)** who carried out **19ECI01 Professional Readiness for Innovation, Employability and Entrepreneurship** project offered by IBM and Anna University ,Chennai.

---

**SIGNATURE**

**Dr.B.BHUVANESHWARI,**  
**M.E.Ph.d.,**

**FACULTY MENTOR**

Assistant Professor

Department of Electronics

and Communication

Engineering

---

**SIGNATURE**

**Dr.S.UMA,**

**FACULTY EVALUATOR**

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.

## TABLE OF CONTENTS

Phase	Phase Description	Week	Dates	Activity Details	Page
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	
7	Results and Discussion			Simulation Outputs and Summary	
8	Conclusion				
9	Appendix I			Source code	
10.	Appendix II			Assignments	

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1. PROJECT OVERVIEW**

Indian Railways is the largest railway network in Asia and world's second largest network operated underneath a single management. It is difficult to manually check for track cracks due to its size. This invention addresses the issue by using an ultrasonic sensor attached to a moving component driven by a stepper motor to detect track cracks. The device can go back and forth over the track thanks to an ultrasonic sensor, and if there is a problem, it sends information to a cloud server so that the railway department can be alerted to cracks in time and potentially save many lives. This is an IoT application that uses a system that is cost effective. This efficient approach of monitoring and evaluating rail tracks in real-time could help to prevent accidents. This system continuously checks the rail stress, analyses and sends the concerned authorities rail break alarms such potential buckling circumstances, rail bending, and wheel impact load detection.

### **1.2 PURPOSE**

The Internet is essentially a network of interconnected computers. However, as the world changes, so does its use, which is no longer limited to email and web browsing. Internet of Things (IoT) was first discussed in relation to the creation of smart homes, smart rural areas, and e-health care. Today's internet also deals with embedded sensors. The term "Internet of Things" describes the connection or communication between two or more devices without the involvement of humans or computers. Connected devices can detect their surroundings according to sensors or actuators onboard. IOT consists of four main parts: sensing the device, gaining access to the device, processing the device's data, and offering applications and services. It also offers data security and privacy in addition to this. Every facet of our everyday life has been impacted by automation. To decrease human effort and save time, further advancements are being made in practically every industry. The same is being considered when automation is being introduced into track testing. Since it gives businesses the functionality they need to operate, railroad track is a crucial component of any company's asset base. It is necessary to resolve issues brought on by railroad concerns. The Indian railroad's most recent technique involves following the train track, which is labor-intensive and takes a lot of time.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 EXISTING SYSTEM**

In the Existing train tracks are manually researched. LED (Light Emitting Diode) and LDR (Light Dependent Resistor) sensors cannot be implemented on the block of the tracks. The input image processing is a clamorous system with high cost and does not give the exact result. The Automated Visual Test Method is a complicated method as the video color inspection is implemented to examine the cracks in rail track which does not give accurate result in bad weather. This traditional system delays transfer of information.

Dr.Velayutham.R et al., (2017) proposed a journal about Controlling railway gates using smart phones by tracking trains with GPS. The proposed model uses GPS (Global Positioning System), a Microcontroller and an SDK (Software Development Kit). GPS is used to avoid railway gate level crossing accidents. It is better than the conventional process of monitoring using sensors, GPS tracking is efficient and needs less maintenance.

Francesca Righetti et al., (2020) proposed a journal about Failure management strategies for IoT-based railways systems. The proposed model uses wireless and Power Line Communication technologies and IoT. It is an all-in-one railway system and it performs well even when problems arise. The failure management strategies on rail-road switches to increase reliability.

Gauransh Singh (2020) proposed a journal about Security System for Railway Crossings using Machine Learning. The proposed system make use of OpenCV, Machine learning and Arduino. Display concept is used to know the exact timing for closing of gates. It reduces the human efforts to a great extent and also increased the system's accuracy. It demonstrates the system's accuracy and effective security level.

Taslim Ahmed (2020) proposed a journal about the Binary World of Zero Death Toll by Implementing a Sustainable Powered Automatic Railway Gate Control System. The proposed system uses ATmega16, Proteus and Code Vision AVR. The Power consumption of the system is very less compared to others since it uses solar power. It shows Increased efficiency of railway gate control.

## **2.2 REFERENCES**

- [1] Dr.Velayutham, R,Sangeethavani.T, Sundaralakshmi.K, "Controlling railway gates using smart phones by tracking trains with GPS"( 2017) International Conference on Circuit ,Power and Computing Technologies (ICCPCT)
- [2] Francesca Righetti, Carlo Vallati , Giuseppe Anastasi, "Failure management strategies for IoT-based railways systems"(2020) IEEE International Conference on Smart Computing (SMARTCOMP)
- [3].Gauransh Singh et.al, "Security System for Railway Crossings using Machine Learning", 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)
- [4] Taslim Ahmed et.al, "Into the Binary World of Zero Death Toll by Implementing a Sustainable Powered Automatic Railway Gate Control System", 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2-4 July, Bangalore, India.

## **2.3 PROBLEM STATEMENT DEFINITION**

Among the various modes of transport, railways are one of the biggest modes of transport in the world. Though there are competitive threats from airlines, luxury buses, public transports, and personalized transports the problem statement is to answer the question “What are the problems faced by the passengers while travelling by train at station and on board.

## CHAPTER 3

### IDEATION AND 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 Smart solutions for the railway system is shown in Fig 3.1.

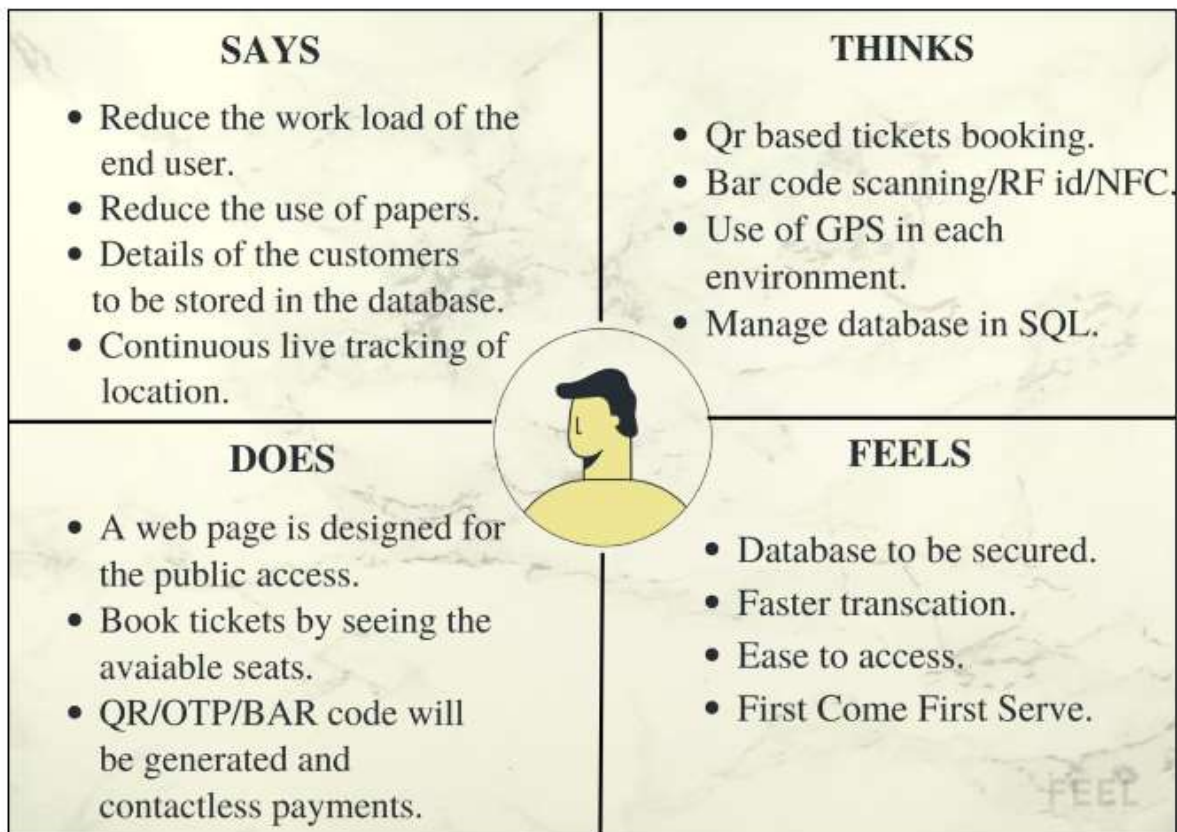


Figure 3.1 Empathy Map

### 3.2 IDEATION & BRAINSTORMING

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 Smart solutions for railway system is shown in Figure 3.2



**Figure 3.2 Brainstorm**

### 3.3 PROPOSED SOLUTION

#### 3.3.1 PROBLEM TO BE SOLVED

Booking tickets using the traditional approach requires a lot of time, and it is very challenging for the ticket collector to check each ticket individually. Additionally, the position of the train is unknown to the passengers.

#### 3.3.2 IDEA / SOLUTION DESCRIPTION

The general public has access to a website where they may view the available seats and purchase tickets. The individual who reserved the train will receive a QR code, which must be presented to the ticket collector while boarding the train. The ticket collectors can identify the personal information by scanning the QR code.

The train has a GPS module to be tracked. The Web app regularly updates the journey's live status. When the ticket collector scans the QR Code, all of the client booking information



will be stored in the database with a special ID and be retrievable.

### **3.3.3 NOVELTY / UNIQUENESS**

Pay using any method, including Net Banking, Credit Cards, and Debit Cards, and the information is safely kept in a database.

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

Users can select the seats in the train that are most comfortable for them, and since the tickets come with QR codes, checking in is simple for ticket collectors as well. Passengers save a ton of time because payments are made online. The GPS is precise, and the website is quite secure.

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

The Internet of Things, cloud computing, big data, navigation, and Node Red service are all utilized in our concept.

## **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.

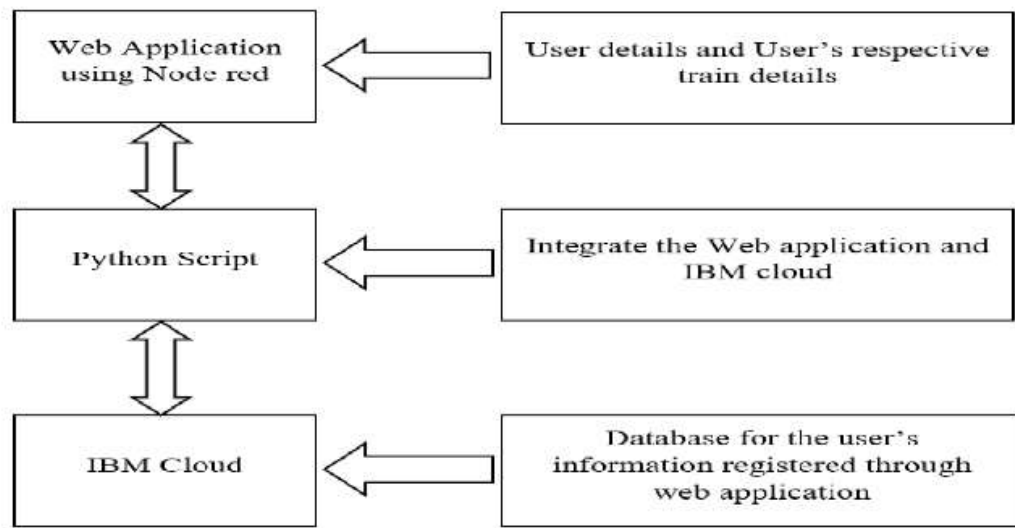
**Customer state fit:** To make sure one understands the target group, their limitations and their currently available solutions, against which one is going to compete.

**Problem-Behavior fit:** To help one to identify the most urgent and frequent problems, understand the real reasons behind them and see which behavior supports it.

**Communication-Channel fit:** To help one to sharpen the communication with strong triggers, emotional messaging and reaching customers via the right channels.

**Solution guess:** Translate all the validated data one has gathered into a solution that fits the customer state and his/her limitations, solves a real problem and taps into the common behavior of the target group.

The problem solution fit for Smart solutions for railway system is shown in Figure 3.3



**Figure 3.3 Problem Solution fit**

## CHAPTER 4

### 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 for Smart solutions for the railway system.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Web page
FR-2	User Confirmation	Confirmation via Message Confirmation via OTP
FR-3	IBM cloud	All details of users are available in cloud
FR-4	Web application	Contains details like boarding station, destination, seat options, name, age, mobile number etc.
FR-5	QR code generator	QR code contains the registration details of the user that will be sent via message

**Table 4.1 Functional Requirements**

#### 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 Smart solutions for the railway system.

<b>FR No.</b>	<b>Non-Functional Requirement</b>	<b>Description</b>
NFR-1	<b>Usability</b>	User can use the Web application to book tickets and can check the status of the trains.
NFR-2	<b>Security</b>	The user details are stored in the cloud with high encryption and these details are non-shareable.
NFR-3	<b>Reliability</b>	The confirmation of the tickets will be sent immediately to the user after the confirmation of payment. The payment details of the users are also well secured and confirm payment through OTP.
NFR-4	<b>Performance</b>	Everything will be done quickly. Since it is online booking, Users can book their tickets comfortably in their places without going to railway station
NFR-5	<b>Availability</b>	The Web application will be available for all the users to book tickets and all the boarding details of the users are also available in the web application. Users can also check the availability of the trains through the Web application.
NFR-6	<b>Scalability</b>	This idea can also be upgraded by adding some additional features in the future.

**Table 4.2 Non -Functional Requirement**

## **CHAPTER 5**

### **PROJECT DESIGN**

#### **5.1 DATA FLOW DIAGRAM**

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.

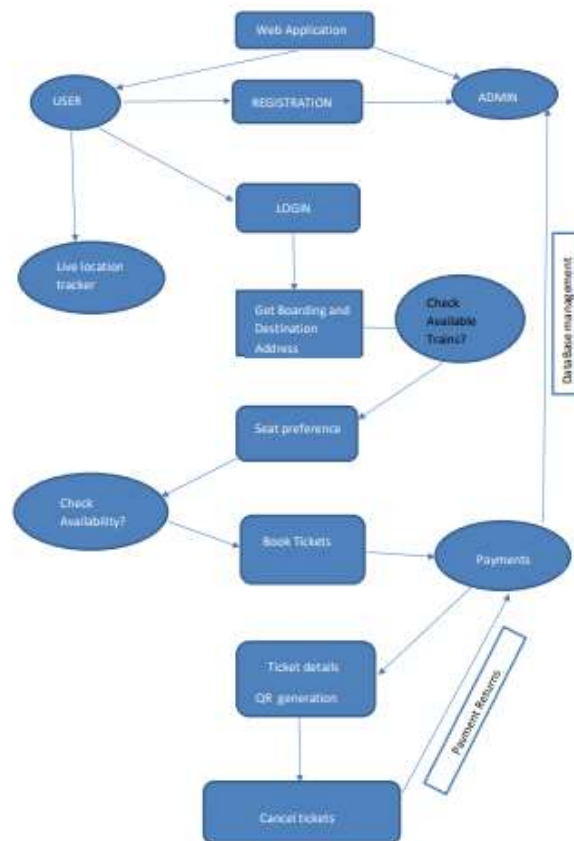
##### **Data store**

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.

The data flow diagram for Smart solutions for the railway system is shown in Figure 5.1



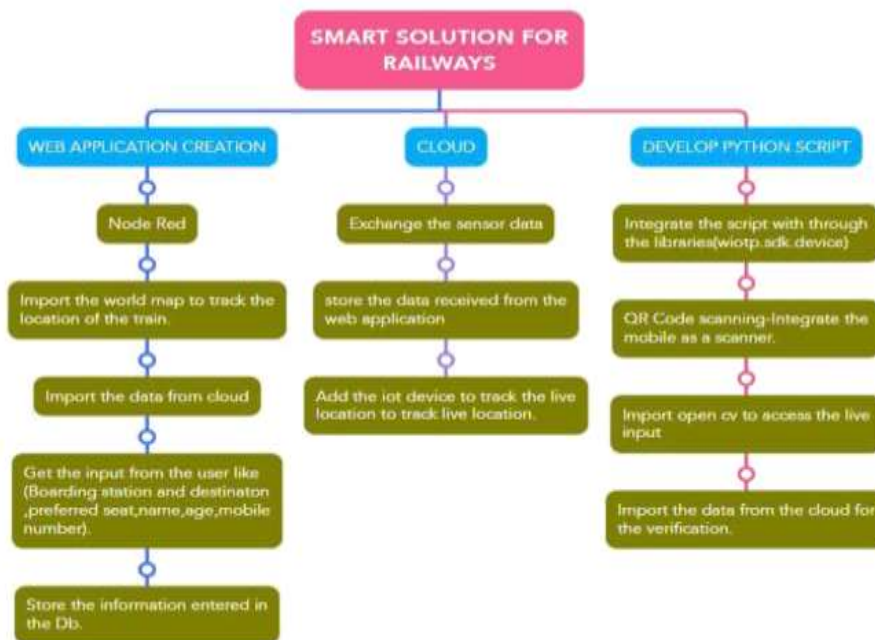
**Figure 5.1 Data flow diagram**

## 5.2 SOLUTION ARCHITECTURE

The general public has access to a website where they may view the available seats and purchase tickets. The individual who reserved the train will receive a QR code, which must be presented to the ticket collector while boarding the train. The ticket collectors can identify the personal information by scanning the QR code. The train has a GPS module to be tracked. The Web app regularly updates the journey's live status. When the ticket collector scans the QR Code, all of the client booking information will be stored in the database with a special ID and be retrievable.

The web application and IoT devices are connected using the IBM Watson IoT platform. Python script is used to integrate Cloudant DB with Python Code and publish data to the IBM IoT Platform. There are two Python programmes: one to read a QR code and retrieve data from Cloudant DB, and the other to publish position (latitude and longitude) data to the IBM IoT Platform. For the purpose of booking a seat on the train, gather user input (basic information) and generate a QR code. The data is created into a QR Code and stored in Cloudant DB as json.

The solution architecture of Smart solutions for the railway system is shown in Figure 5.2.



**Figure 5.2 Solution Architecture**

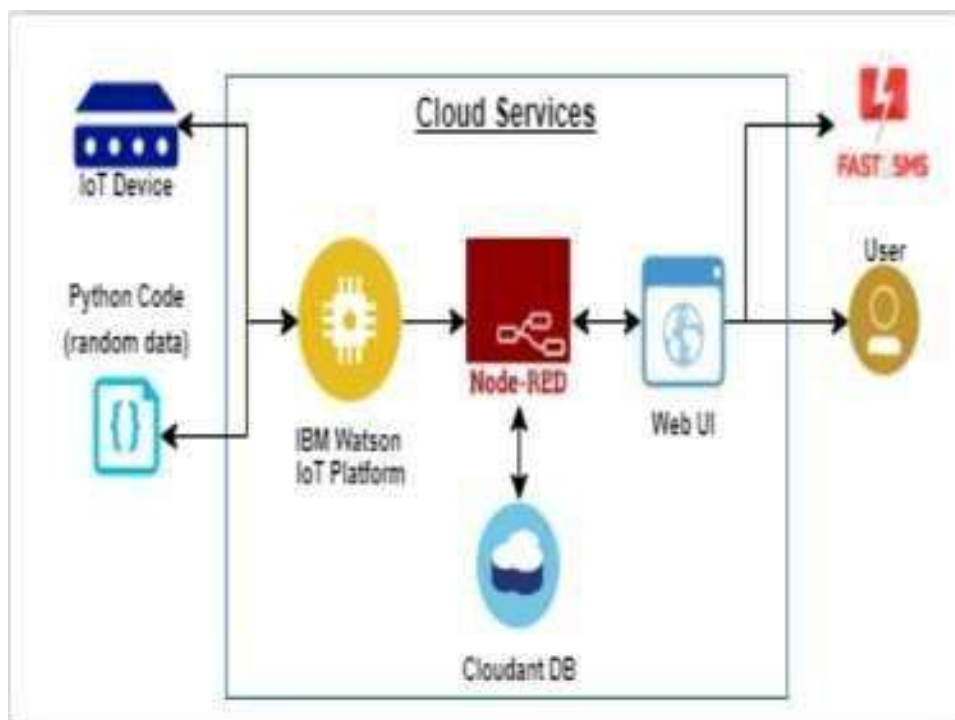
## 5.3 TECHNOLOGY ARCHITECTURE

### 5.3.1 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 5.3 shows the data flow for Smart solutions for railway system.



**Figure 5.3 Data flow diagram**



#### Guidelines:

- Using the Web application, a user books a ticket based on the availability of the seats by giving the general required information.
- Once a user clicks on the submit button, a QR code is generated with a Unique ID and the data is stored in the Cloudant DB with that Unique ID.
- Users can save the QR code for further process.
- In python code, a Ticket collector can scan the QR code and extract the information from the QR Code i.e., Unique ID. With that Unique ID, data is fetched from the Cloudant DB, if it is not found, then it displays Not a Valid Ticket.
- Also, the live location of the train will be published to IBM IoT platform using python code
- The train location can be tracked from a Web Application.

#### 5.3.2 COMPONENTS AND TECHNOLOGIES

The components and technologies used for Smart solutions for the railway system is shown in following table 5.1

S.No	Component	Description	Technology
1.	IoT Device	How user interacts with application	Cloud technology
2.	Python code	python code for publishing the location (latitude and longitude) data to the IBM IoT Platform and the other python code to read the QR Code and fetch the data from Cloudant DB.	Python
3.	IBM Watson IoT Platform	IBM Watson IoT platform acts as the mediator to connect the web application to IoT device, so create the IBM Watson IoT platform.	Analytics and information retrieval
4.	Node Red	Connect to IBM IoT platform and get the location, store the data in Cloudant DB.	Java script , cloud technology
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.

7.	Web UI	we get the expected output by providing the desired user input where the QR Code is generated and the same data is stored in the form of json in Cloudant DB.	python
8.	Fast SMS	To confirm ticket booking	python
9.	user	Take user input (Basic Information) for booking a seat on the train	Python

**Table 5.1 Components & Technologies**

### 5.3.3 APPLICATION CHARACTERISTICS:

The applications used for Smart solutions for railway system is shown in following table 5.2.

S.No	Character istics	Description	Technology
1.	Usability	Users can use the Web application to book tickets and can check the status of the trains.	QR code
2.	Security	The user details are stored in the cloud with high encryption and these details are non-shareable.	Encryptions, QR code
3.	Reliability	The confirmation of the tickets will be sent immediately to the user after the confirmation of payment. The payment details of the users are also well secured and confirm payment through OTP.	IBM Watson IoT Platform, Node Red
4.	Availabilit y	The Web application will be available for all the users to book tickets and all the boarding details of the users are also available in the web application. Users can also check the availability of the trains through the Web application.	IBM Watson IoT Platform, Node Red
5.	Performan ce	Everything will be done quickly. Since it is online booking, Users can book their tickets	IBM Watson IoT Platform, Node Red

		comfortably in their places without going to railway station	
6.	Scalability	This idea can also be upgraded by adding some additional features in the future.	IBM Watson IoT Platform, Node Red

**Table 5.2 Application Characteristics**

## 5.4 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.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer  (Mobile user)	Reserving ticket	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	High	Sprint -1
Customer	Reserving ticket	USN-2	As a user, I will receive confirmation email	I can receive confirmation	High	Sprint -1

(Mobile user)			once I have registered for the application	email & click confirm		
Customer	Reserving ticket	USN-3	As a user, I can register for the application and	I can register & access the	Low	Sprint -2
(Mobile user)			enter the details for reserving the ticket.	dashboard with Facebook		
				Login		
Customer	Dashboard	Users	The details will be stored safely	I can access it using	Medium	Sprint-3
(Mobile user)				database		
Customer (Web user)	Reserving ticket	User	Enter the details and click submit button to	I can use the QR code	High	Sprint-1
			book ticket	which is been generated		
Customer Care	Connecting the	Customer	Connects with the service by logging in	Can get connected with	Medium	Sprint-3
Executive	service provider			the server		
Administrator		Admin	The data is given by the user	Can add or update the	High	Sprint-1

**Table 5.3 User stories**

## CHAPTER 6

### PROJECT PLANNING & 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 Smart solutions for railway system.

<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
<b>Proposed Solution</b>	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact,	29 SEPTEMBER 2022

	scalability of solution, etc.	
<b>Problem Solution Fit</b>	Prepare problem - solution fit document.	30 SEPTEMBER 2022
<b>Solution Architecture</b>	Prepare solution architecture 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 6.1 Milestones & Task**

The Table 6.2 shows the sprint planning and estimation for Smart solutions for railway system

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Story Points</b>	<b>Priority</b>	<b>Team Members</b>
Sprint-1	Login	USN-1	Passenger can login to the website with their email and password and they can book their ticket	10	High	Dhaswanth.N.G
Sprint-1	Login	USN-2	Passenger can login to web application for booking tickets.	10	High	Abisheek.S
Sprint-2	Dashboard	USN-3	As a passenger, I will receive confirmation email once I have registered for the Application.	20	Low	Arunagiri.B
Sprint-3	Dashboard	USN-4	Receive confirmation mail.	20	Medium	Gokul.R
Sprint-4	Dashboard	USN-5	The user can book ticket with the help of travel agency by providing the necessary details	20	High	Dhaswanth.N.G

**Table 6.2 sprint planning**

## **CHAPTER 7**

### **CODING & SOLUTIONING**

#### **7.1 FEATURE**

- IOT device
- IBM Watson platform
- Node red
- Cloudant DB
- Web UI
- Geofence
- MIT App
- Python code

#### **7.2. FEATURE 2**

- Registration
- Login
- Verification
- Ticket Booking
- Payment
- Ticket Cancellation
- Adding Queries

#### **7.3. CODE**

##### **7.3.1. CODE FOR QR CODE GENERATION**

```
import cv2

import numpy as np

import time

import pyzbar.pyzbar as pyzbar

from ibmcloudant.cloudant_v1 import CloudantV1

from ibmcloudant import CouchDbSessionAuthenticator
```



```

from ibm_cloud_sdk_core.authenticators import BasicAuthenticator

authenticator = BasicAuthenticator('apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz','b0ab119f45d3e6255eabb9
78e7e2f0')

service = CloudantV1(authenticator=authenticator)

service.set_service_url('https://apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb9
78e7e2f0')

cap = cv2.VideoCapture(0)

font = cv2.FONT_HERSHEY_PLAIN

while True:

    __, frame = cap.read()

    decodedObjects = pyzbar.decode(frame)

    for obj in decodedObjects:

        a = obj.data.decode('UTF-8')

        cv2.putText(frame, "Ticket", (50,50), font, 2,

            (255, 0, 0), 3)

    try:

        response = service.get_document(

            db = 'booking',

```

```
        doc_id = a

    ).get_result()

    print(response)

    time.sleep(5)

except Exception as e:

    print("Not a Valid Ticket")

    time.sleep(5)


cv2.imshow("Frame", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

    break

cap.release()

cv2.destroyAllWindows()

client.disconnect().
```

### 7.3.2.CODE FOR TRAIN TICKET BOOKING

```
import wiotp.sdk.device

import time

#import random

myConfig = {

    "identity": {

        "orgId": "4k7d8l",

        "typeId": "IBM",

        "deviceId": "63697477"

    },

    "auth" : {

        "token": "2(zMmQG_bjWs3d+-7b"

    }

}


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()


def pub(data):
```

```

        client.publishEvent(eventId = "status", msgFormat = "json", data =
myData, qos = 0)

        print("Published data Successfully: %s", myData)

while True:

    myData = {'name':'Train1','lat':17.6387448,'lon':78.4754336}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat':17.6341908,'lon':78.4744722}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat':17.6340889,'lon':78.4745052}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat': 17.6248626,'lon':78.4720259}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat':17.6188577,'lon':78.4698726}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat':17.6132382,'lon':78.4707318}

    pub(myData)

    time.sleep(3)

    client.commandCallback = myCommandCallback

client.disconnect()

```

## CHAPTER 8

### TESTING

#### 8.1 CLOUDANT DATABASE

A new database document is created and document ID is obtained and this ID can be used to link the Node RED service with the database. The metadata of the DB like token key and ID is also generated. Figure 8.1(a) and 8.2(b) shows the cloudant database dashboard with unique ID and key.

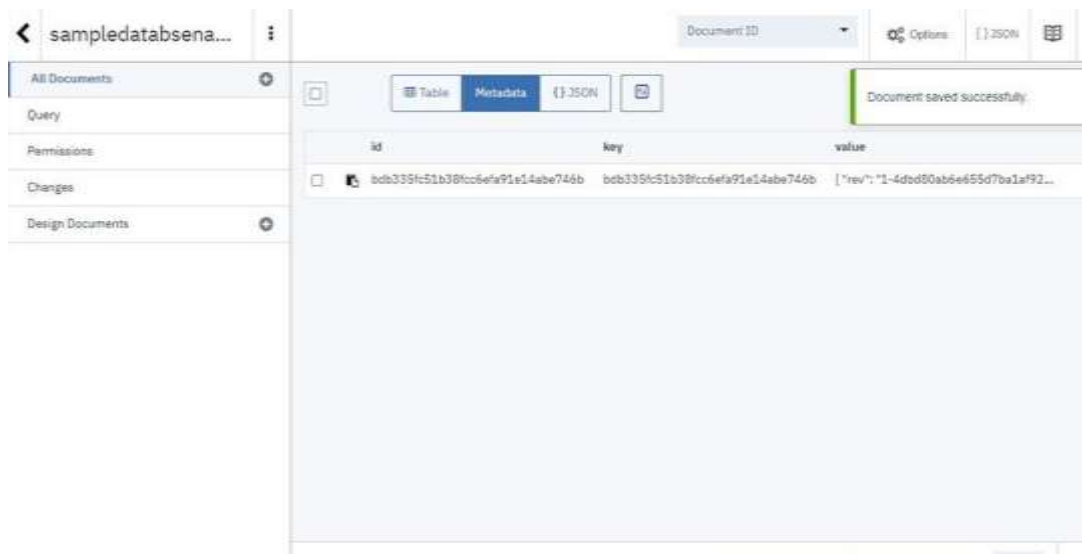


Figure 8.1(a) Cloudant database

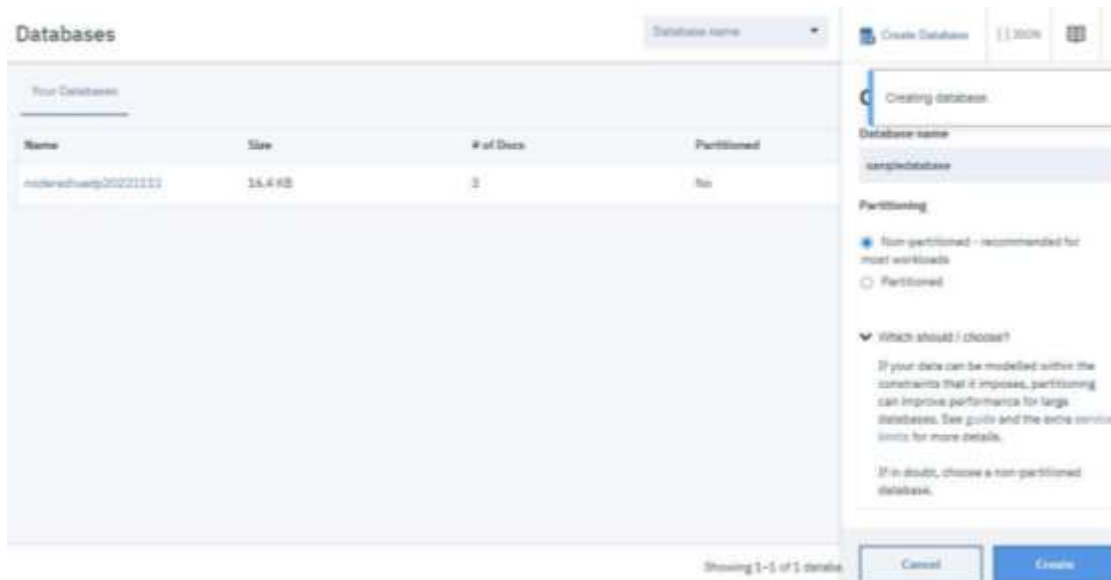
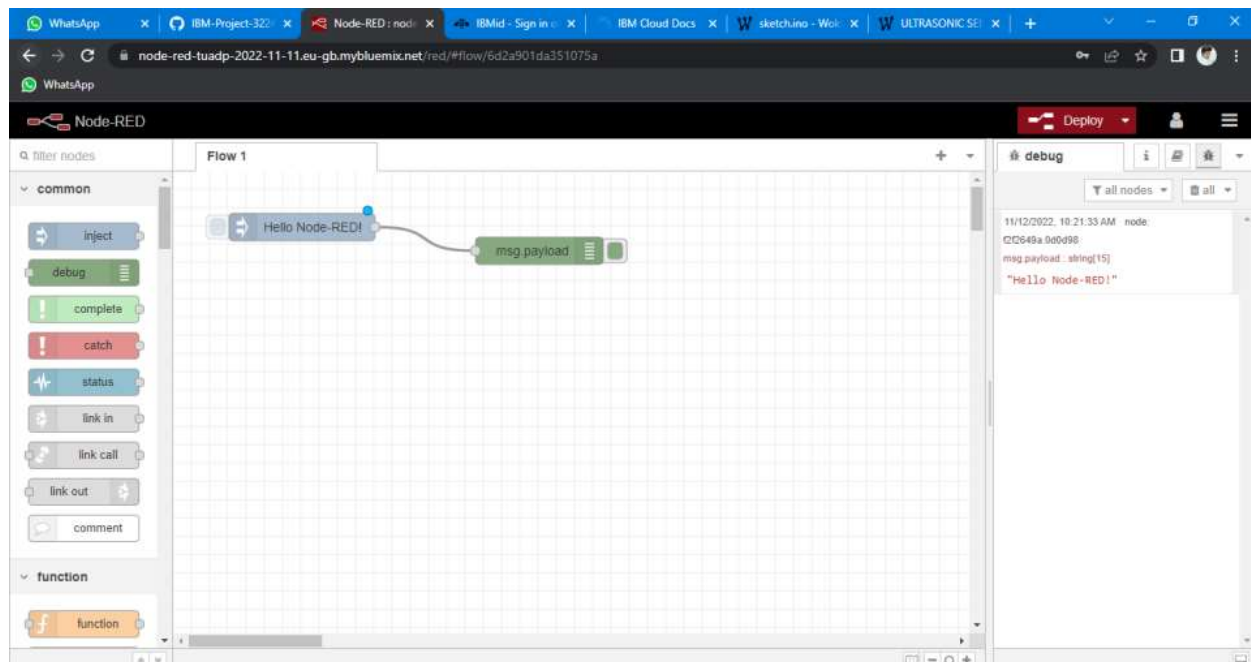


Figure 8.1(b) Sample Database

## 8.2 NODE RED SERVICE

Node-RED is a flow-based development tool for visual programming developed originally by IBM. Figure 8.2(a) depicts the Node RED service dashboard. Using the blocks of nodes, an application can be easily developed without coding.



**Figure 8.2 Node RED service**

## 8.3 SENDING LOCATION TO IBM WATSON IOT PLATFORM

A python script was developed to send the location coordinates to IBM watson IoT platform. Location data is sent in the form of JSON format to the cloud and using the specific key values, the latitude and longitude data will be identified. Figure 8.3(a) represents the developed python script. Figure 8.3(b) and 8.3(c) represents the data received in the IBM cloud platform.

```
#!/usr/bin/env python
import json
import wiotp.sdk.device
import time

myConfig={
    "identity":{
        "orgid": "9w42t3",
        "typeid": "nodemculocation",
        "deviceid": "1234567890"
    },
    "auth":{
        "token": "NWAsdaFn4h5dPg8sQd"
    }
}

client=wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    name="Smartbridge"
    #in area location
    #latitude 17.4225176
    #longitude 78.5458842
    #out area location
    latitude=17.4219272
    longitude=78.5488783
    myData={"name": name, "lat":latitude, "lon": longitude}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Data published to IBM IoT platform: ",myData)
    time.sleep(5)
client.disconnect()
```

Figure 8.3(a) Python code to send location coordinates

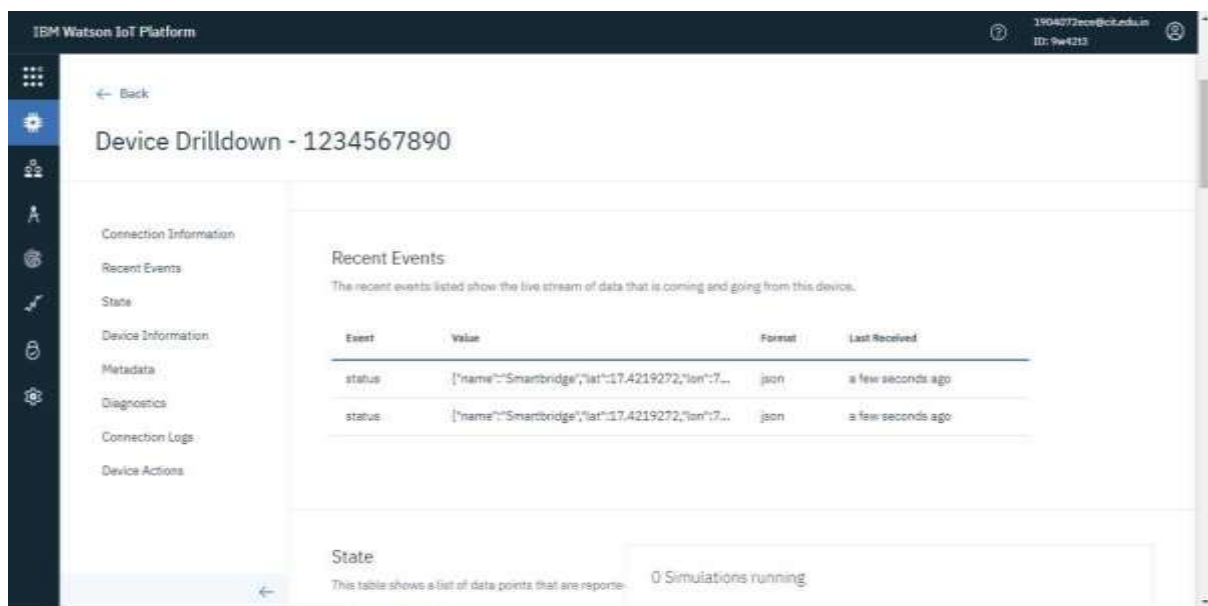
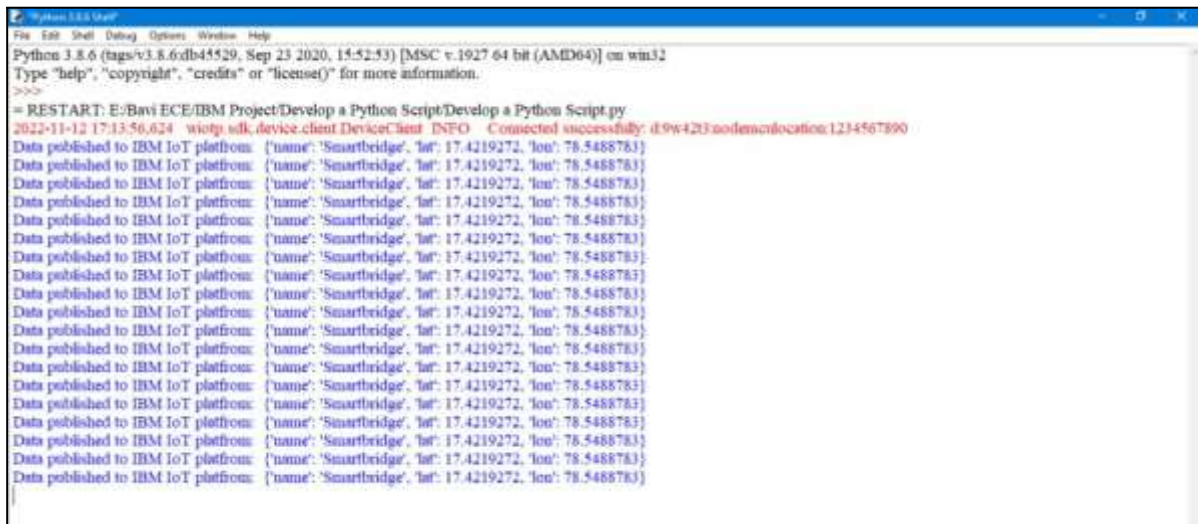


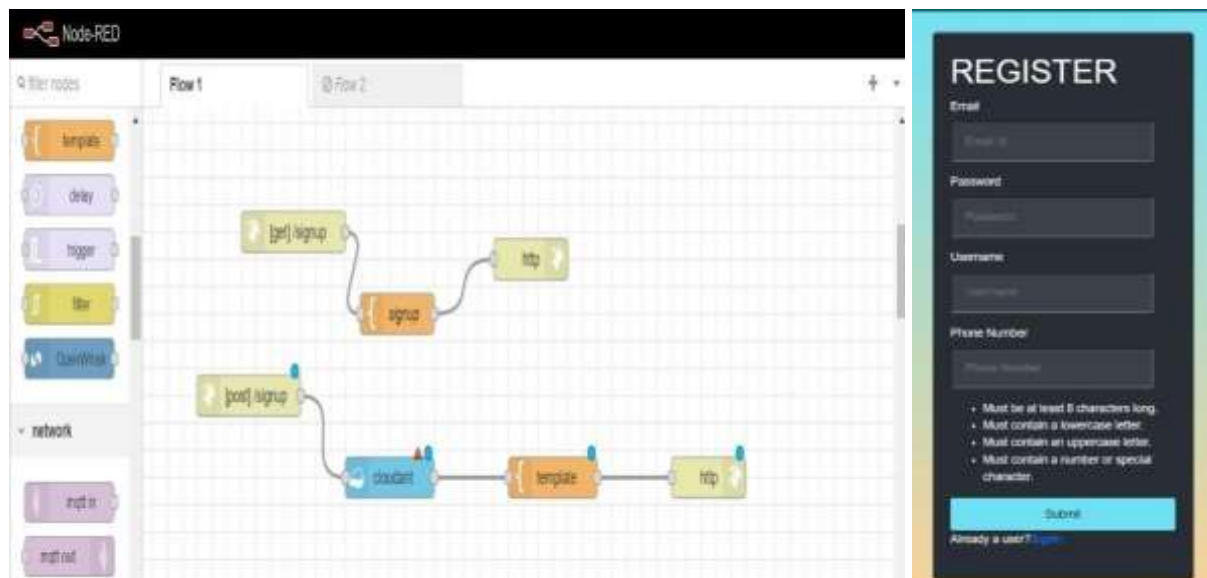
Figure 8.3(b) IBM Watson IoT platform



### Figure 8.3(c) Data publication to IBM cloud

## 8.4. NODE RED SIGN-UP PAGE

In this module, a signup page using Node RED service is developed. Using this page, the user can register themselves to the application by providing their valid email ID, password, username and phone number. Figure 8.4(a) shows the Node RED page and the output obtained in the web application.



**Figure 8.4 Signup page using Node RED service**

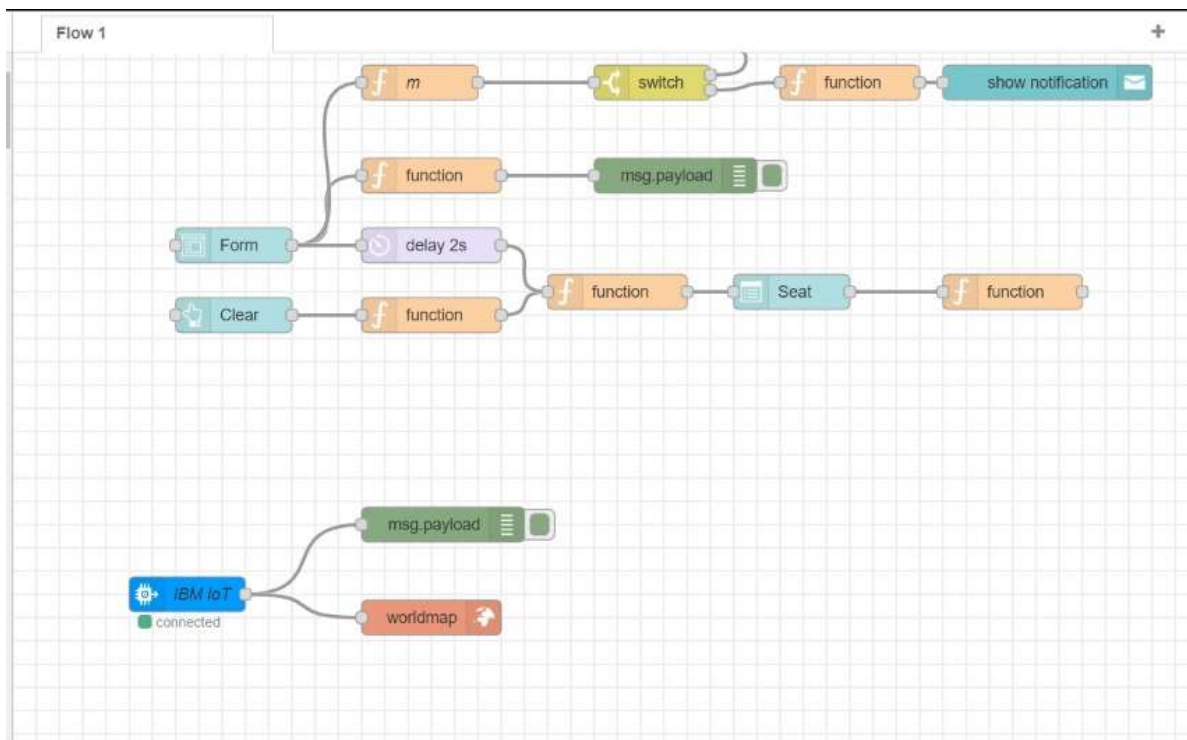


## 8.5. NODE RED FOR TICKET BOOKING

In this module, the user detail page and ticket booking page has been developed using the node red service. Features like getting the input from the user like Boarding station, destination station, seat preference, user details like name, age, mobile number has been created as shown in figure 8.5(a) and 8.5(b). The world map tool is here used for monitoring the live location of the train, module for qr generation is also been created by using the nodes available in palette.



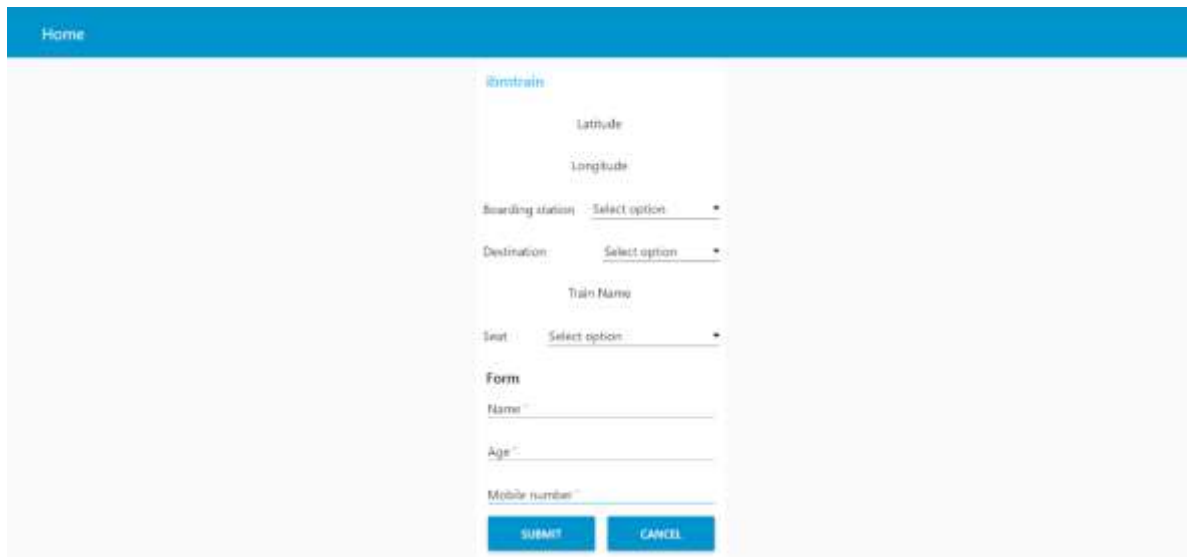
**Figure 8.5(a) Web page for Ticket Booking using node red.**



**Figure 8.5(b) Development of web page For ticket booking**

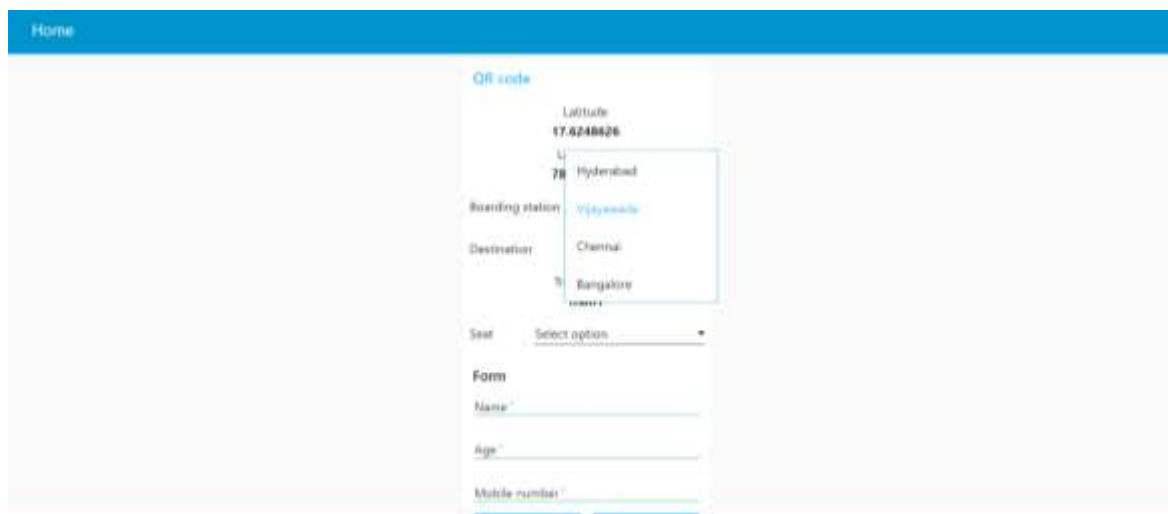
## **8.6.WEB PAGE**

In this module, the nodes which are created for services using node red service has been deployed, and by using the command in with the web link of the node red service has been run in a URL and webpage has been published. Figure 8.6(a) interprets the home page of the Webpage created, where the user can enter their details and their boarding and destination address.



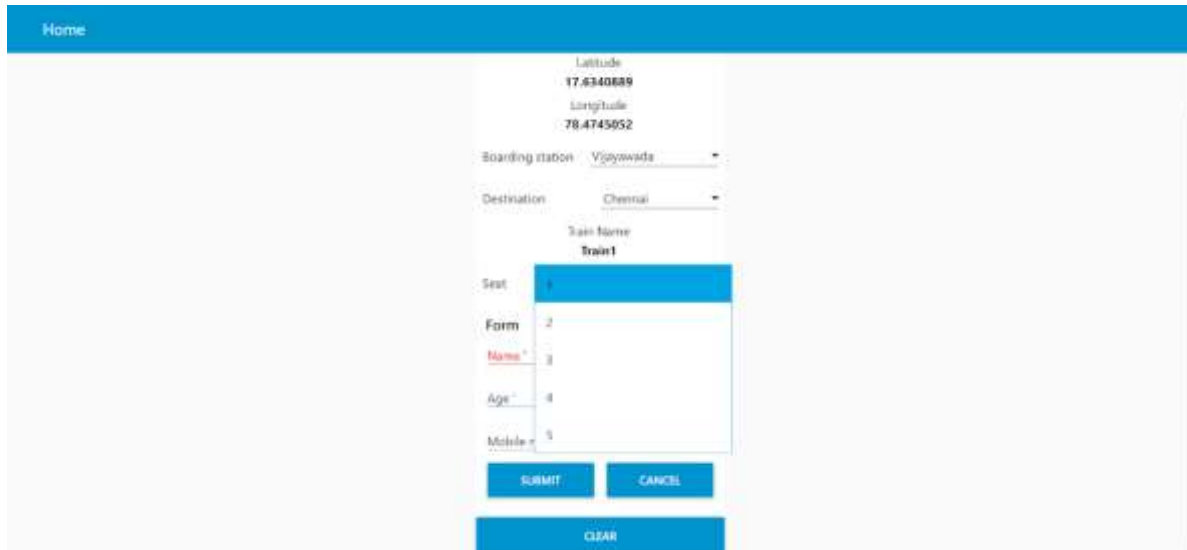
**Figure 8.6(a) Home page**

Once after the creation of the web page the user can enter the boarding and destination address of which they need to travel. The user can select the boarding and destination address which would be stored in database accordingly which is shown in figure 8.6(b).



**Figure 8.6(b) Boarding and Destination Input**

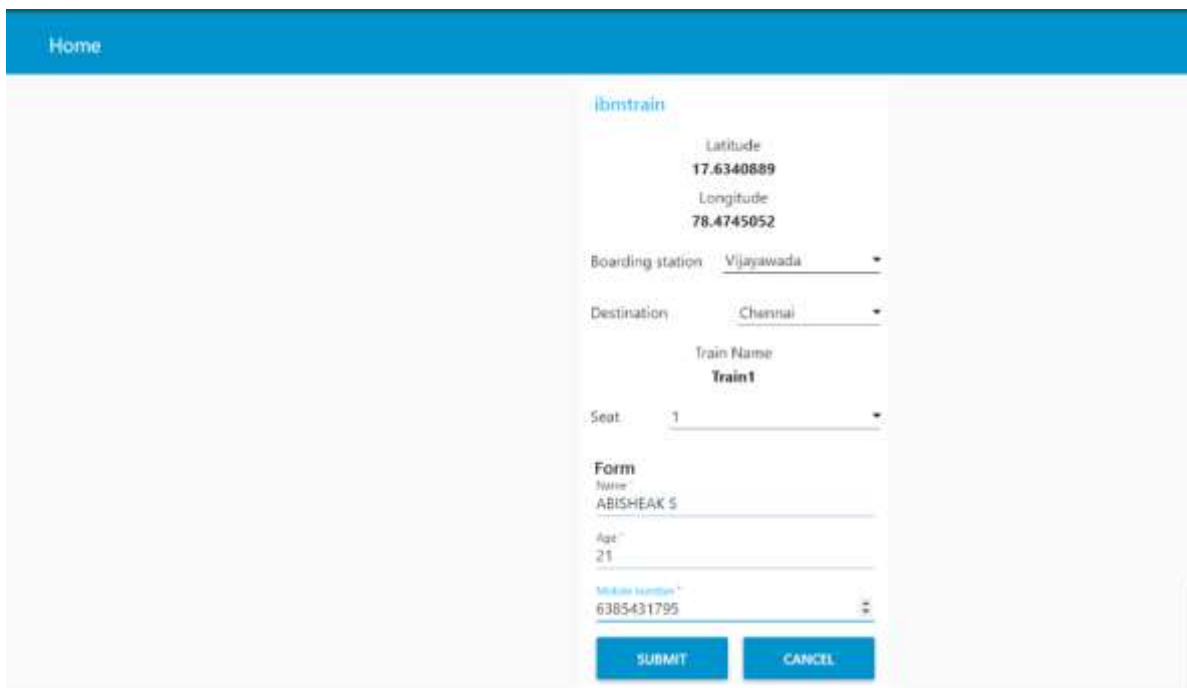
After selecting the boarding and destination address the user can select the seat preference as per their requirement as shown in figure 8.6(c).



The screenshot shows a web application interface with a blue header bar labeled "Home". The main content area is white and contains a form. At the top of the form, the coordinates "Latitude 17.6340889" and "Longitude 78.4745052" are displayed. Below these are two dropdown menus: "Boarding station" with "Vijayawada" selected and "Destination" with "Chennai" selected. Underneath is a label "Train Name" followed by "Train1". A "Seat" dropdown menu is open, showing a list of options: "1", "2", "3", "4", and "5". Below the seat selection are three buttons: "SUBMIT", "CANCEL", and "CLEAR".

**Figure 8.6(c) Seat preference**

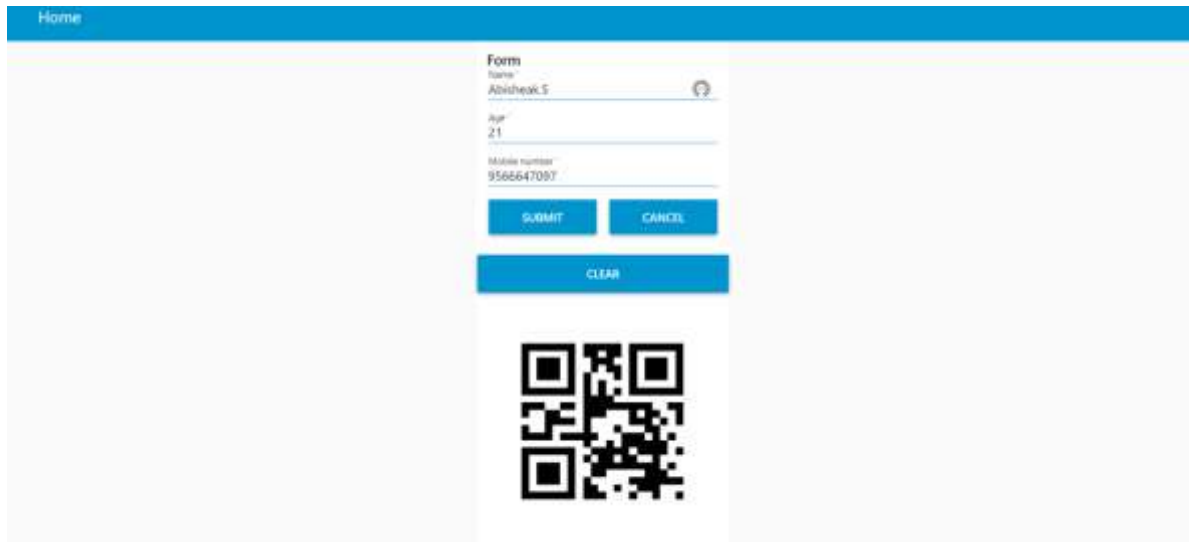
Once after selecting the seat preference the user can enter their personal details like name, age, mobile number and click on submit for the confirmation of the ticket as shown in figure 8.6(d)



The screenshot shows the same web application interface as Figure 8.6(c), but with the "Seat" dropdown menu closed. The "Form" section now contains input fields for "Name" (filled with "ABISHAK S"), "Age" (filled with "21"), and "Mobile number" (filled with "6385431795"). The "SUBMIT" and "CANCEL" buttons are still present at the bottom of the form.

**Figure 8.6(d) User details input**

After submitting the details, a QR code will be generated as shown in figure 8.6(e) through this user can carry on boarding the train and scan at the travel premises for checking.



The screenshot displays a mobile application interface with a blue header bar labeled "Home". Below the header, a form titled "Form" is centered. The form contains the following fields and controls:

- Name:** A text input field containing the value "Abhishek S".
- Age:** A text input field containing the value "21".
- Mobile number:** A text input field containing the value "9566647097".
- Buttons:** Below the form fields are three buttons: "SUBMIT" (blue), "CANCEL" (blue), and "CLEAR" (blue).
- QR Code:** Below the buttons is a square QR code.

**Figure 8.6(e) QR generation**

## CHAPTER 9

### 9.RESULTS

#### 9.1. PERFORMANCE METRICS

Fig 9.1 shows the performance metrics of Smart solutions for railway system

					Date	19-11-2022							
					Team ID	PNT2022TMID							
					Project Name	smart solutions							
					Maximum Mark	4 marks							
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation (Y/N)	BUG ID	Executed By
1	Functional	Registration	Registration through the form by		1.click on register.2.fill in the registration form.3.click register		Registration form to be filled will be displayed	Working as expected	Pass				Abisheak.S
2	UI	Generating OTP	Generate the otp for further process		1.Generating an OTP number		User can register using the phone numbers.gmail, facebook,etc	Working as expected	Pass				B.Arunagirish
3	Functional	OTP verification	Verify user otp using the email		1.Enter the gmail id and enter the password.2. click submit	Username: chalam@gmail.com password: Testing123	Otp verified is to be displayed	Working as expected	Pass				Dhaswanth.N.G
4	Functional	Login page	Verify user is able to log into application		1.Enter into the login page.2. Click on my account drop down button.3. Enter an invalid user name/emailid.4. Enter the valid password in the password textbox.5.Click on the login button.	Username: chalam@gmail.com password: Testing1236786 86786876876	Application should show 'incorrect email or password' validation message.	Working as expected	Pass				Gokul.R
5	Functional	Display train details	The user can view the available train details		1. As a user, I can enter the start and destination address to get the list of the trains available connecting my desired route	Username: chalam@gmail.com password: Testing1236786 86786876876	A user can view about the available trains to enter start and destination details.	Working as expected	fail				Abisheak.S
6	Functional	Booking	user can provide the basic details like name,age,etc		1.Enter the method of reservation.2. Enter name, age,mobileetc.3.Enter the how many tickets wants to be booked.4.also enter the number members details like		Ticket booked to be displayed	Working as expected	Pass				Arunagirish.b
7	UI	Booking seats	User can choose the class.seat/berth if the preferred seat/berth isn't available					Working as expected	Pass				Dhaswanth.N.G
8	Functional	payment	User can choose the payment method like card/upi,etc				1.payment method	Working as expected	Pass				Gokul.R
9	Functional	Redirectional	user can be redirected to the selected page				1.After payments	Working as expected	Pass				Abisheak.S
10	functional	Ticket Generation	a user can download the ticket also Download the ticket				1.Save the payment	Working as expected	Pass				Arunagirish.b
11	UI	Ticket status	The user can whether the tickets booked are confirmed.				1.Known the status	Working as expected	Pass				Dhaswanth.N.G
12	functional	Reminder	User tends to know through notification		1.message app		User gets notification	Working as expected	Pass				Gokul.R
13	functional	GPS tracking	User can track the live location of train		1.Tracking train for getting the info		Tracking process	Working as expected	Pass				Abisheak.S
14	functional	Ticket cancellation	User can cancel the tickets if any		1.Tickets need to be cancelled		Tickets cancelled	Working as expected	Pass				B.Arunagirish
15	functional	Feedback	Answer the queries		Get the feedback		Analysed feedback	Working as expected	Pass				Dhaswanth.N.G

Fig 9.1 performance matrices

## **CHAPTER 10**

### **ADVANTAGES AND DISADVANTAGES**

#### **ADVANTAGES**

- Openness – compatibility between different system modules, potentially from different vendors;
- Orchestration – ability to manage large numbers of devices, with full visibility over them;
- Dynamic scaling – ability to scale the system according to the application needs, through resource virtualization and cloud operation;
- Automation – ability to automate parts of the system monitoring application, leading to better performance and lower operation costs.

#### **DISADVANTAGES**

- Approaches to flexible, effective, efficient, and low-cost data collection for both railway vehicles and infrastructure monitoring, using regular trains;
- Data processing, reduction, and analysis in local controllers, and subsequent sending of that data to the cloud, for further processing;
- Online data processing systems, for real-time monitoring, using emerging communication technologies;
- Integrated, interoperable, and scalable solutions for railway systems preventive maintenance.

## **CHAPTER 11**

### **CONCLUSION**

Accidents occurring in the Railway transportation system cost a large number of lives. So this system helps us to prevent accidents and give information about faults or cracks in advance to railway authorities. So that they can fix them and accident cases become less. This project is cost effective. By using more techniques, they can be modified and developed according to their applications. By this system many lives can be saved by avoiding accidents. The idea can be implemented in large scale in the long run to facilitate better safety standards for rail tracks and provide effective testing infrastructure for achieving better results in the future.

## **CHAPTER 12**

### **FUTURE SCOPE**

In future CCTV systems with IP based cameras can be used for monitoring the visual videos captured from the track. It will also increase security for both passengers and railways. GPS can also be used to detect exact location of track fault area; IP cameras can also be used to show fault with the help of video. Locations on Google maps with the help of sensors can be used to detect in which area track is broken



## CHAPTER 13

### 13.1 SOURCE CODE

#### 13.1.1 CODE FOR QR CODE GENERATION

```
import cv2

import numpy as np

import time

import pyzbar.pyzbar as pyzbar

from ibmcloudant.cloudant_v1 import CloudantV1

from ibmcloudant import CouchDbSessionAuthenticator

from ibm_cloud_sdk_core.authenticators import BasicAuthenticator


authenticator = BasicAuthenticator('apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz','b0ab119f45d3e6255eabb9
78e7e2f0')

service = CloudantV1(authenticator=authenticator)

service.set_service_url('https://apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb9
78e7e2f0')


cap = cv2.VideoCapture(0)

font = cv2.FONT_HERSHEY_PLAIN


while True:

    _, frame = cap.read()

    decodedObjects = pyzbar.decode(frame)
```

```

for obj in decodedObjects:

    a = obj.data.decode('UTF-8')

    cv2.putText(frame, "Ticket", (50,50), font, 2,
                (255, 0, 0), 3)

    try:

        response = service.get_document(

            db = 'booking',

            doc_id = a

        ).get_result()

        print(response)

        time.sleep(5)

    except Exception as e:

        print("Not a Valid Ticket")

        time.sleep(5)

cv2.imshow("Frame", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

    break

cap.release()

cv2.destroyAllWindows()

client.disconnect().

```

### 13.1.2 CODE FOR STORING DATA IN DATABASE:

```
var m = global.get('m')

var d = new Date()

var utc = d.getTime() + (d.getTimezoneoffset() * 60000);

var offset = 5.5;

newDate = new Date(utc + (3600000*offset));

var n = newDate.toISOString()

var date = n.slice(0,10)

var time = n.slice(11,19)

var d1 = date + ',' + time

msg.payload = {

  "_id" : d1,

  "Name" : m.name,

  "Age" : m.age,

  "Mobile" : global.get('b'),

  "Destination" : global.get('d'),

  "Seat" : global.get('s')

}

return msg;
```

### **13.1.3 CODE FOR SEAT PREFERENCE:**

```
global.set('s1',0)

global.set('s2',0)

global.set('s3',0)

global.set('s4',0)

var a1 = [1,2,3,4,5]

global.set('a',a1)

msg.payload = global.get('a')

return msg;
```

### **13.1.4 CODE FOR MONITORING SEAT:**

```
var a = global.get('a')

var s = []

for (let i = 0; i<a.length;i++){

    s.push(a[i])

}

if (s.length == 0){

    msg.options = [{"No seats available":0}]

}

else{

    msg.options = s

}

msg.payload = s

return msg;
```

### 13.1.5 CODE FOR TRAIN TICKET BOOKING

```
import wiotp.sdk.device

import time

#import random

myConfig = {

    "identity": {

        "orgId": "4k7d8l",

        "typeId": "IBM",

        "deviceId": "63697477"

    },

    "auth" : {

        "token": "2(zMmQG_bjWs3d+-7b"

    }

}

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()
```

```

def pub(data):

    client.publishEvent(eventId = "status", msgFormat = "json", data =
myData, qos = 0)

    print("Published data Successfully: %s", myData)

while True:

    myData = {'name':'Train1','lat':17.6387448,'lon':78.4754336}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat':17.6341908,'lon':78.4744722}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat':17.6340889,'lon':78.4745052}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat': 17.6248626,'lon':78.4720259}

    pub(myData)

    time.sleep(3)

    myData = {'name':'Train1','lat':17.6188577,'lon':78.4698726}

    pub(myData)

    myData = {'name':'Train1','lat':17.6132382,'lon':78.4707318}

    pub(myData)

    time.sleep(3)

    client.commandCallback = myCommandCallback

client.disconnect()

```

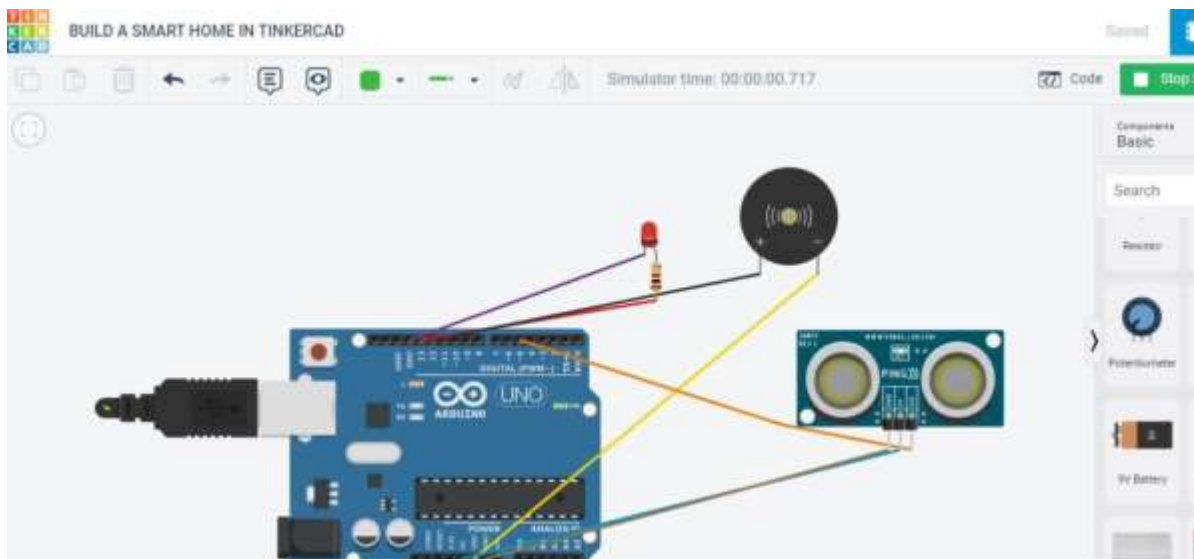
## 13.2 ASSIGNMENTS

### 13.2.1 ASSIGNMENT 1

```
// C++ code
//
void setup()
{
  pinMode(13,OUTPUT);
  pinMode(12,OUTPUT);
  Serial.begin(7000);
}

void loop()
{
  digitalWrite(13, HIGH);
  Serial.println("13LED ON");
  digitalWrite(12,OUTPUT);
  Serial.println("12LED ON");
}
```

**SIMULATION:**



### 13.2.2 ASSIGNMENT 2

#### CODE:

```
int t=2;
int e=3;
void setup()
{
  Serial.begin(9600);
  pinMode(t,OUTPUT);
  pinMode(e,INPUT);
  pinMode(12,OUTPUT);
}

void loop()
{
  //ultrasonic sensor
  digitalWrite(t,LOW);
  digitalWrite(t,HIGH);
  delayMicroseconds(10);
  digitalWrite(t,LOW);
  float dur=pulseIn(e,HIGH);
  float dis=(dur*0.0343)/2;
  Serial.print("Distance is: ");
  Serial.println(dis);

  //LED ON
  if(dis>=100)
```

```
    //LED ON
    if(dis>=100)
    {
      digitalWrite(8,HIGH);
      digitalWrite(7,HIGH);
    }

    //Buzzer For ultrasonic Sensor
    if(dis>=100)
    {
      for(int i=0; i<=30000; i=i+10)
      {
        tone(12,i);
        delay(1000);
        noTone(12);
        delay(1000);
      }
    }

    //Temperate Sensor
    double a= analogRead(A0);
```



```

    //Temperate Sensor
    double a= analogRead(A0);
    double t=((a/1024)*5)-0.5)*100;
    Serial.print("Temp Value: ");
    Serial.println(t);
    delay(1000);

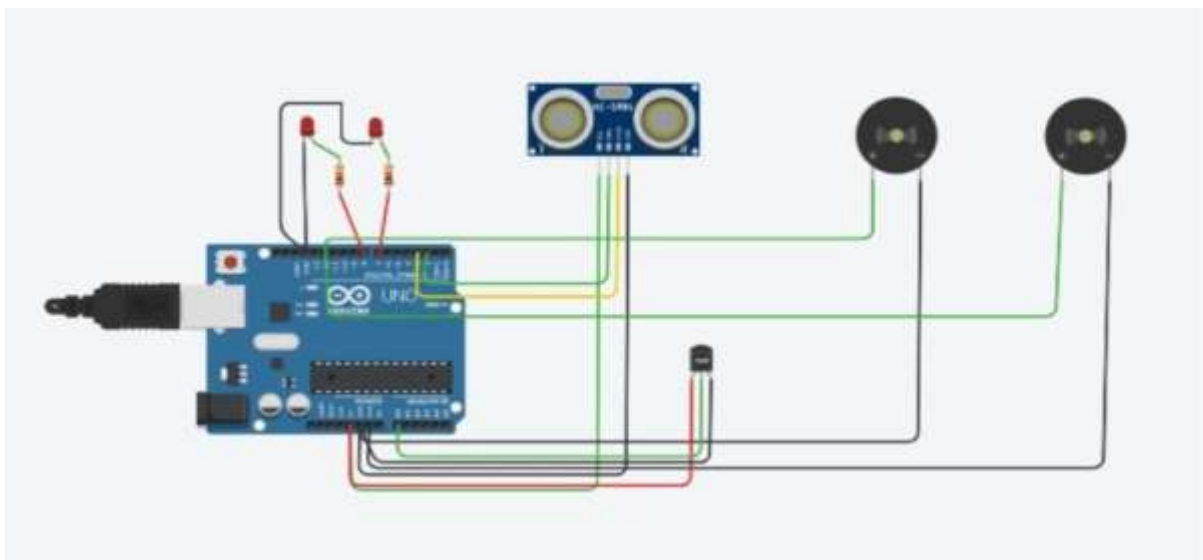
    //LED ON
    if(t>=100)
    {
        digitalWrite(8,HIGH);
        digitalWrite(7,HIGH);
    }

    //Buzzer for Temperature Sensor
    if(t>=100)
    {
        for(int i=0; i<=30000; i=i+10)
        {
            tone(12,i);
            delay(1000);
            noTone(12);
            delay(1000);
        }
    }

    //LED OFF
    if(t<100)
    {
        digitalWrite(8,LOW);
        digitalWrite(7,LOW);
    }
}

```

## OUTPUT



### 13.2.3 ASSIGNMENT 3

```
from gpiozero import  
  
Button button =  
  
Button(21)  
  
while True:  
    print(button.is_pressed)  
while True:  
    if  
        button.is_presse
```

Add an LED

```
from gpiozero import Button,  
LED led = LED(25)  
while True:  
    button.wait_for_press()  
    led.on()  
    button.wait_for_release()  
    led.off()  
while True:  
    led.on()  
    button.wait_for_press()  
    led.off()  
    button.wait_for_release()
```

Traffic lights

```

from gpiozero import Button,
TrafficLights lights = TrafficLights(25, 8,
7)
while True:
    button.wait_for_press()
    lights.on()
    button.wait_for_release()
    lights.off()

```

### Traffic lights sequence

```

from time import
sleep while True:
    lights.green.on()
    sleep(1)

```

```

    lights.red.on()
    sleep(1) lights.off()
while True:
    button.wait_for_press()
    lights.green.on() sleep(1)
    lights.amber.on() sleep(1)
    lights.red.on() sleep(1)

    lights.off()

```

### 13.2.4 ASSIGNMENT 4

#### Title:

Write code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events. Upload document with wokwi share link and images of IBM cloud.

Wowki Link: <https://wokwi.com/projects/348308707259449940>

#### Source Code:

```
#include <WiFi.h>//library for wifi
#include <WiFiClient.h>
#include <PubSubClient.h>//library for MQTT
#include <ArduinoJson.h>
// creating the instance by passing pin and typr of dht connected float distance;
#define sound_speed 0.034 int
trigpin=18;      int
echopin=19; int led=5; int
LED=9;          long
duration;
String message;// creating the instance by passing pin and typr of dht connected

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

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

#define ORG "4k7d8l"//IBM ORGANITION ID
#define DEVICE_TYPE "IBM"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "63697477"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "2(zMmQG_bjWs3d+-7b"
//Tok

en String data3; float
h, t; //
Customise the above
values
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event
perform and format in which data to be send char subscribetopic[] = "iot-
2/cmd/command/fmt/String";// cmd REPRESENT command type AND
COMMAND IS TEST OF FORMAT STRING char
authMethod[] = "use-token-auth";// authentication method char token[] =
TOKEN;
```

```

char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id

// .
WiFiClient wifiClient; // creating the instance for wificlient PubSubClient client(server,
1883, callback ,wifiClient); //calling the predefined client id by passing parameter like
server id,portand wificredential
void setup()// configureing the ESP32
{
    Serial.begin(115200);
    pinMode(trigpin,OUTPUT);
    pinMode(echopin,INPUT);
    pinMode(led,OUTPUT); delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{

    digitalWrite(trigpin,LOW);
    digitalWrite(trigpin,HIGH);
    delay(1000);
    digitalWrite(trigpin,LOW); duration=pulseIn(echopin,HIGH);
    distance=duration*sound_speed/2;
    Serial.println("distance"+String(distance)+"cm"); if(distance<100)
    {
        message="Alert";
        digitalWrite(led,HIGH);
    } else {
        message="No problem";
        digitalWrite(led,LOW);
    }
    delay(1000);
}

```

```

PublishData(distance,message);
    // if (!client.loop()) {
    //     mqttconnect();
    // }
}

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

void PublishData(float d, String a) { mqttconnect();//function call for
    connecting to ibm
    /*
        creating the String in in form JSON to update the data to ibm cloud
    */
    DynamicJsonDocument doc(1024);
    String payload;      doc["Distance:
    "]=d;      doc["Message: "]=a;
    serializeJson(doc, payload);

    Serial.print("Sending payload: ");
    Serial.println(payload);

    if (client.publish(publishTopic, (char*) payload.c_str())) { Serial.println("Publish ok");//
    if it sucessfully upload data on the cloud then it will print publish ok in Serial monitor or
    else it will print publish failed
    } else {
        Serial.println("Publish failed");
    }
}

void mqttconnect() {      if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);

    while (!client.connect(clientId, authMethod, token)) {
        Serial.print(".");
        delay(500);
    }
    initManagedDevice();
    Serial.println();
}
}

```

```

}
void wificonnect() //function defination for wificonnect
{
    Serial.println(); Serial.print("Connecting
    to ");

    WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the
    connection
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");

    }
    Serial.println(""); Serial.println("WiFi
    connected"); Serial.println("IP address:
    "); Serial.println(WiFi.localIP());
}

void initManagedDevice() {
    if (client.subscribe(subscribetopic)) {
        Serial.println((subscribetopic));
        Serial.println("subscribe to cmd OK");
    } else {
        Serial.println("subscribe to cmd FAILED");
    }
}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
    Serial.print("callback invoked for topic: ");
    Serial.println(subscribetopic);
    for (int i = 0; i < payloadLength; i++) {
        //Serial.print((char)payload[i]);
        data3 += (char)payload[i];
    }
    Serial.println("data: " + data3);
    if(data3=="lighton") {
        Serial.println(data3);
        digitalWrite(LED,HIGH);
    }
    else
    {
        Serial.println(data3);
        digitalWrite(LED,LOW;

```

```

    } data3="";
}

```

## Output:

The screenshot shows the WOKWI IDE with a sketch titled "ULTRASONIC SENSOR copy". The sketch code is as follows:

```

1 #include <SPI.h> //library for SPI
2 #include <PinChangeInt.h>
3 #include <PubSubClient.h> //library for MQTT
4 // creating the instance by passing pin and type of dht connected
5 float distance;
6 #define sound_speed 0.034
7 int trigpin=4;
8 int echopin=5;
9 int led=6;
10 int ledOn;
11 long duration;
12 String message; // creating the instance by passing pin and type of dht connected
13 void callback(char* topic, byte* payload, unsigned int payloadLength){
14 //----- credentials of IoT accounts -----
15 //----- customize the above values -----
16 #define ONE "8K78K1" //IoT ID/username ID
17 #define DEVICE_TYPE "ESP32" //device type mentioned in the Watson IoT Platform
18 #define DEVICE_ID "83687477" //device ID mentioned in the Watson IoT Platform
19 #define TOKEN "2129408_93663de-0b" //token
20 String data;
21 float h, t;
22 //----- customize the above values -----
23 char server[] = ONE ".messaging.internetofthings.ibmcloud.com"; // Server Name
24 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event perform
25 char subscribeTopic[] = "iot-2/cmd/command/fmt/string"; // cmd. MESSAGEID command type and
26 char authMethod[] = "use-token-auth"; // authentication method
27 char token[] = TOKEN;
28 char clientId[] = "d:" ONE "/" DEVICE_TYPE "/" DEVICE_ID "/" clientId;

```

The simulation window shows the following output:

```

distance44.98cm
Sending payload: {"distance":44.98},{"message":"Alert"}
Publish ok
distance44.95cm
Sending payload: {"distance":44.95},{"message":"Alert"}
Publish ok
distance44.95cm

```

The screenshot shows the WOKWI IDE with a sketch titled "ULTRASONIC SENSOR copy". The sketch code is as follows:

```

1 #include <SPI.h> //library for SPI
2 #include <PinChangeInt.h>
3 #include <PubSubClient.h> //library for MQTT
4 // creating the instance by passing pin and type of dht connected
5 float distance;
6 #define sound_speed 0.034
7 int trigpin=4;
8 int echopin=5;
9 int led=6;
10 int ledOn;
11 long duration;
12 String message; // creating the instance by passing pin and type of dht connected
13 void callback(char* topic, byte* payload, unsigned int payloadLength){
14 //----- credentials of IoT accounts -----
15 //----- customize the above values -----
16 #define ONE "8K78K1" //IoT ID/username ID
17 #define DEVICE_TYPE "ESP32" //device type mentioned in the Watson IoT Platform
18 #define DEVICE_ID "83687477" //device ID mentioned in the Watson IoT Platform
19 #define TOKEN "2129408_93663de-0b" //token
20 String data;
21 float h, t;
22 //----- customize the above values -----
23 char server[] = ONE ".messaging.internetofthings.ibmcloud.com"; // Server Name
24 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event perform
25 char subscribeTopic[] = "iot-2/cmd/command/fmt/string"; // cmd. MESSAGEID command type and
26 char authMethod[] = "use-token-auth"; // authentication method
27 char token[] = TOKEN;
28 char clientId[] = "d:" ONE "/" DEVICE_TYPE "/" DEVICE_ID "/" clientId;

```

The simulation window shows the following output:

```

distance368.93cm
Sending payload: {"distance":368.93},{"message":"No problem"}
Publish ok
distance368.93cm
Sending payload: {"distance":368.93},{"message":"No problem"}
Publish ok
distance368.93cm

```



IBM Watson IoT Platform

Device ID: a3697477 | Status: Connected | Device Type: IBM | Class ID: Device | Last Active: Nov 14, 2022 4:46 PM

Identity | Device Information | Recent Events | State | Logs

The recent events listed show the raw stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"Distance":168.9129905,"Message":"No pres..."}	json	a few seconds ago
Data	{"Distance":168.9129905,"Message":"No pres..."}	json	a few seconds ago

Items per page: 50 | 1 of 1 item



GITHUB LINK: <https://github.com/IBM-EPBL/IBM-Project-3973-1658677143>

APP URL: [https://node-red-jhl1n-2022-11-06.eu-gb.mybluemix.net/?\\_ga=2.135432625.1083714255.1667710803-1525077761.1659454031](https://node-red-jhl1n-2022-11-06.eu-gb.mybluemix.net/?_ga=2.135432625.1083714255.1667710803-1525077761.1659454031)

DEMO LINK:

<https://drive.google.com/file/d/1c3MMklf59ocxBi9Q36dMtJx6ytP35CCc/view?usp=drivesdk>

REFERENCE:

[1].Dr. Velayutham.R,Sangeethavani.T, Sundaralakshmi.K, "Controlling railway gates using smart phones by tracking trains with GPS"( 2017) International Conference on Circuit ,Power and Computing Technologies (ICCPCT)

[2].Francesca Righetti, Carlo Vallati , Giuseppe Anastasi, "Failure management strategies for IoT-based railways systems"(2020) IEEE International Conference on Smart Computing (SMARTCOMP)

[3].Gauransh Singh et.al, "Security System for Railway Crossings using Machine Learning", 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)

[4].Taslim Ahmed et.al, "Into the Binary World of Zero Death Toll by Implementing a Sustainable Powered Automatic Railway Gate Control System", 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2-4 July, Bangalore, India