



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**R.M.K.ENGINEERING COLLEGE**

(An Autonomous Institution)

**R.S.M. Nagar, Kavaraipettai-601 206**

**PROJECT BASED EXPERIENTIAL LEARNING PROGRAM (NALAIYA  
THIRAN)**

**SMART SOLUTIONS FOR RAILWAYS**

**A PROJECT REPORT**

**Submitted by**

**KOTTI SAI SRI HARSHA (111719106078)**

**Jithendra Naga Sri Venkata Syam Chaluvadi (111719106062)**

**Koushik Pulivarthi (111719106079)**

**Karna Govardan Manikanta Reddy (111719106065)**

**TEAM ID : PNT2022TMID16062**

# **CONTENTS**

1. **INTRODUCTION**
  1. Project Overview
  2. Purpose
2. **LITERATURE SURVEY**
  1. Existing problem
  2. References
  3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
  1. Empathy Map Canvas
  2. Ideation & Brainstorming
  3. Proposed Solution
  4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
  1. Functional requirement
  2. Non-Functional requirements
5. **PROJECT DESIGN**
  1. Data Flow Diagrams
  2. Solution & Technical Architecture
  3. User Stories
6. **PROJECT PLANNING & SCHEDULING**
  1. Sprint Planning & Estimation
  2. Sprint Delivery Schedule
  3. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
  1. Feature 1
  2. Feature 2
8. **TESTING**
  1. Test Cases
  2. User Acceptance Testing
9. **RESULTS**
  1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**
  - Source Code
  - GitHub & Project Demo Link

# **INTRODUCTION**

## **1.1) PROJECT\_OVERVIEW**

The goal of our project is to develop smart solutions for railways, it has been designed to reduce the work load of the user and also the use of paper. Here in this project, we have all the features shown below.

### **Features:**

- A Web page is designed for the public where they can book tickets by seeing the available seats.
- After booking the train, the person will get a QR code which has to be shown to the Ticket Collector while boarding the train.
- The ticket collectors can scan the QR code to identify the personal details.
- A GPS module is present in the train to track it. The live status of the journey is updated in the Web app continuously
- All the booking details of the customers will be stored in the database with a unique ID and they can be retrieved back when the Ticket Collector scans the QR Code.

## **1.2) PURPOSE**

The main purpose behind our project is to speed up the process of ticketing and prevent wastage of time at the ticket counters present at railway stations. This also resolves the issue with the running status of trains where people get confused on where the train is located at. With the help of our service, people can book tickets or easily locate and track trains from anywhere across the world.

We have also worked on easing the task of ticket collectors to check and verify tickets of passengers. A unique QR code is given to every customer upon a successful transaction which is scanned later by the ticket collector for verification. Once the ticket collector scans the QR code, he gets full details of the person in journey.

# **LITERATURE SURVEY**

## **2.1) EXISTING PROBLEM**

Even after significant improvement in technology, the major problem that still persists in our railway system is the waiting at ticket counters for long durations and confusion regarding the running of trains whether the passengers were early or missed it.

These problems gave burden to the passengers waiting at the ticketing kiosks by wasting their time which they can use it for their prioritized tasks, and the confusions regarding the train running status gave rise to unnecessary worries to the commuters.

## **2.2) REFERENCES**

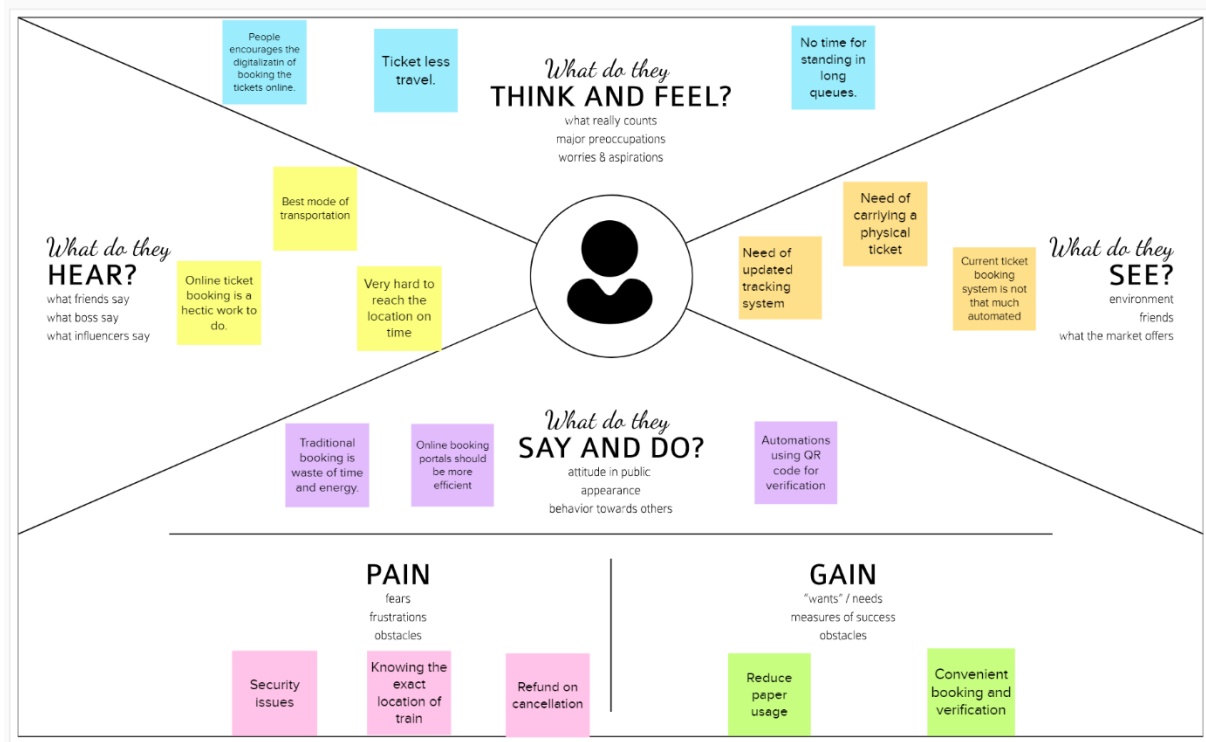
1. Roman Khoebal, Teeravisit Laohapensaeng, Rounsan Chaisricharoen, "Passenger Monitoring Model for easily Accessible Public City Trams/Trains" (2015).
2. Parag Chatterjee, Asoke Nath, "Application of smart computing in Indian Railway Systems" (2014).
3. Sana Khoja, Maithili Kadam, "Android Suburban Railway Ticketing with GPS as Ticket Checker" (2012).
4. Sujith Kumar, K.M.Yatheendra Parvan, V.Sumathy, Thejeswari C.K, "Novel Approach for Smart Indian Railways" (2017).
5. Sarvath Saba, Sharon Philip, Shri harsha, Mukund Naik, Sudeep Sherry, "A Review on IOT based automated seat allocation and verification using QR code"(2022)

## **2.3) Problem Statement:**


1. People wait for long duration at the ticketing queues to purchase a train ticket
2. Ticket collector goes through different documents to verify the ticket of an individual, this could be a time-consuming process.
3. People are unaware of the train running status, if they were early or missed it.

# IDEATION AND PROPOSED SOLUTION

## 3.1) EMPATHY MAP CANVAS



### 3.2) IDEATION AND BRAINSTORMING:



## Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare

1 hour to collaborate

2-6 people recommended

Share template feedback

4

### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

Open article

5

### Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

Exercise

Have the leads of people to facilitate and teach their teams easily

Key rules of brainstorming

To run an smooth and productive session

May to begin.

Encourage wild ideas.

Defer judgment.

Listen to others.

Go for volume.

If possible, be visual.

Need some inspiration?

See a limited version of this template to inspire your work.

Open example



3

## Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TP

You can select a sticky note and tell the group (project or class) how to check (over to next drawing)

### Hardware

Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons

### Software

Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons

### Knowledge

Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons

### Materials

Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons

3

## Group ideas

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

20 minutes

TP

Add sentence-like labels to sticky notes to make it easier to find, organize, compare, and integrate. Repeat ideas as often as needed within your mind.

### Affordable

Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons

### Convenience

Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons

### Security

Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons

### Data Processing

Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons
Microcontroller	LED display	4 buttons

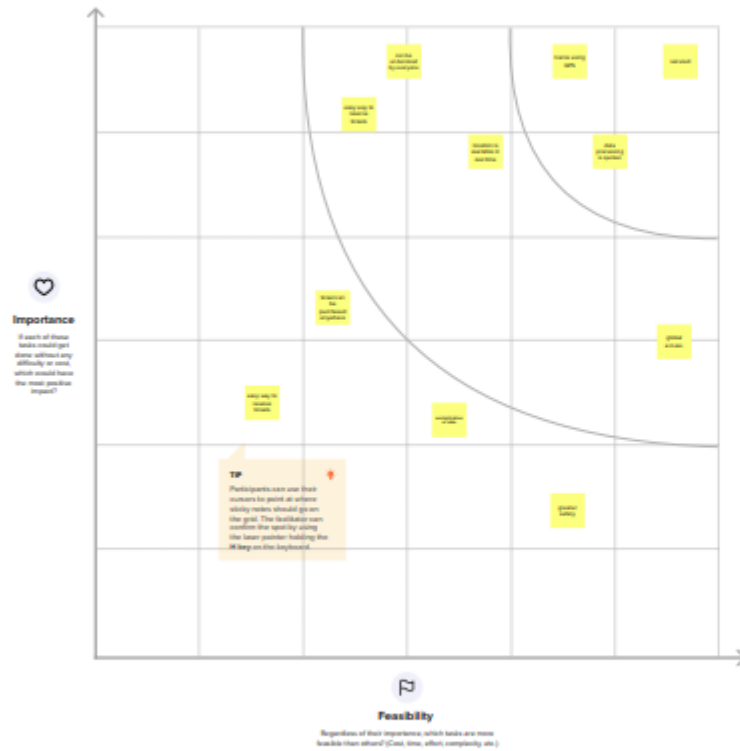


4

### Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

30 minutes



5

### After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

#### Quick add-ons

- Show the mural**  
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- Export the mural**  
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save to your drive.

#### Keep moving forward

- Strategy blueprint**  
Define the components of a new idea or strategy.  
[Open the template](#)
- Customer experience journey map**  
Understand customer needs, motivations, and obstacles for an experience.  
[Open the template](#)
- Strengths, weaknesses, opportunities & threats**  
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.  
[Open the template](#)

[Show template feedback](#)



### **3.3) PROPOSED SOLUTION**

- A Web page is designed for the public where they can book tickets by seeing the available seats. After booking the train, the person will get a QR code which has to be shown to the Ticket Collector while boarding the train.
- The ticket collectors can scan the QR code to identify the personal details.
- A GPS module is present in the train to track it. The live status of the journey is updated in the Web app continuously making it easier for people to locate it.

### 3.4) PROBLEM SOLUTION FIT

Define CS, fit into CL	<b>1. CUSTOMER SEGMENT(S)</b> <b>CS</b>  Commuters
Focus on PR, tap into BE, understand RC	<b>2. PROBLEMS / PAINS</b> + ITS FREQUENCY <b>PR</b>  Customers face several hardships such as visiting the railway station, standing in long queues and time wastage in booking train tickets  This happens every time a user tries to book ticket.
Identify strong TR & EM	<b>3. TRIGGERS TO ACT</b> <b>TR</b> As it provides a hassle free way to book train tickets, the customers will be interested in using the application for their benefit.
	<b>4. EMOTIONS</b> BEFORE / AFTER <b>EM</b> The commuters will have a relief as they can book and track the train with ease as it was an absurd task to book tickets and locate train in the past.

<b>6. CUSTOMER LIMITATIONS</b> <small>EG. BUDGET, DEVICES</small> <b>CL</b> Customer should have access to a smartphone when needed and the internet connectivity should be reliable to carry out processing without errors	<b>5. AVAILABLE SOLUTIONS</b> <small>PROS &amp; CONS</small> <b>AS</b> Tickets are booked via ticket counter at railway stations, but they demand long free time of people. People had to visit the station every time they need tickets.	Explore AS, differentiate
<b>9. PROBLEM ROOT / CAUSE</b> <b>RC</b> In the past, lack of technology gave rise to this problem.  Lack of awareness is still causing this problem.	<b>7. BEHAVIOR + ITS INTENSITY</b> <b>BE</b> Taking feedback on different aspects varying from 1 to 10	Focus on PR, tap into BE, understand RC
<b>10. YOUR SOLUTION</b> <b>SL</b> Web page is designed for the public where they can book tickets. GPS module is present in the train to track it	<b>8. CHANNELS of BEHAVIOR</b> <b>CH</b> <b>ONLINE</b> By providing feedback or filling an online form  <b>OFFLINE</b> Visiting the office present at major railway stations or contacting customer care through call	Extract online & offline CH of BE

# **REQUIREMENT ANALYSIS**

**There are two types: i) Functional Requirements**

**ii) Non – Functional Requirements**

## **4.1) FUNCTIONAL REQUIREMENTS**

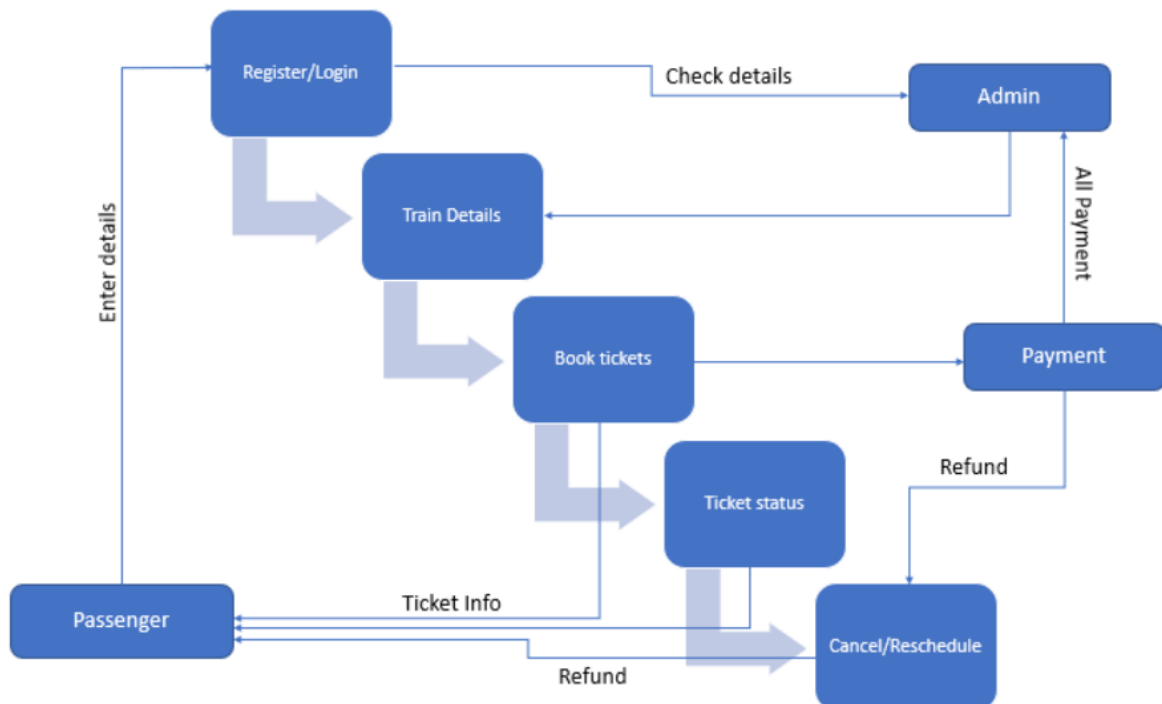
- Registration
  - Through Email
  - Through Phone Number
  - Through Google, Facebook
- User Confirmation
  - Email Verification
  - OTP
  - Confirmation via Call
- Journey Details
  - To and from station information
  - Class of Travel
  - Date of Travel
- Train Information
  - Available seats
  - Class availability
  - Quota (General, Senior citizen, Tatkaal)
- Passenger Information
  - Name
  - Gender
  - Date Of Birth
  - Citizen Proof
  - Contact Details
- Payment
  - Through Cards
  - Through Net Banking
  - Through UPI/Wallet
  - Through Voucher
- Information Storage
  - Databases to store passenger details
  - Databases to store train details
  - QR code generator
- Train Tracking
  - GPS module API
  - Real time location upload

## **4.2) NON - FUNCTIONAL REQUIREMENTS**

- Usability
  - Server should be responsive
  - It should accommodate a greater number of users
  - Should be maintained regularly
- Reliability
  - User should have a secure transaction
  - User data should be protected from third party access
- Performance
  - Server should be working 24x7
  - Server should not experience any crashed due to overload
  - It should be responsive
- Availability
  - Ticketing should be available through website or app
  - Offline ticketing must be present at railway stations
- Scalability
  - Users interacting with our service should feel secured and reliable
- Security
  - User data must be protected
  - Encryption of user data is needed to prevent third party from accessing it
  - Payments should take place through a secure portal

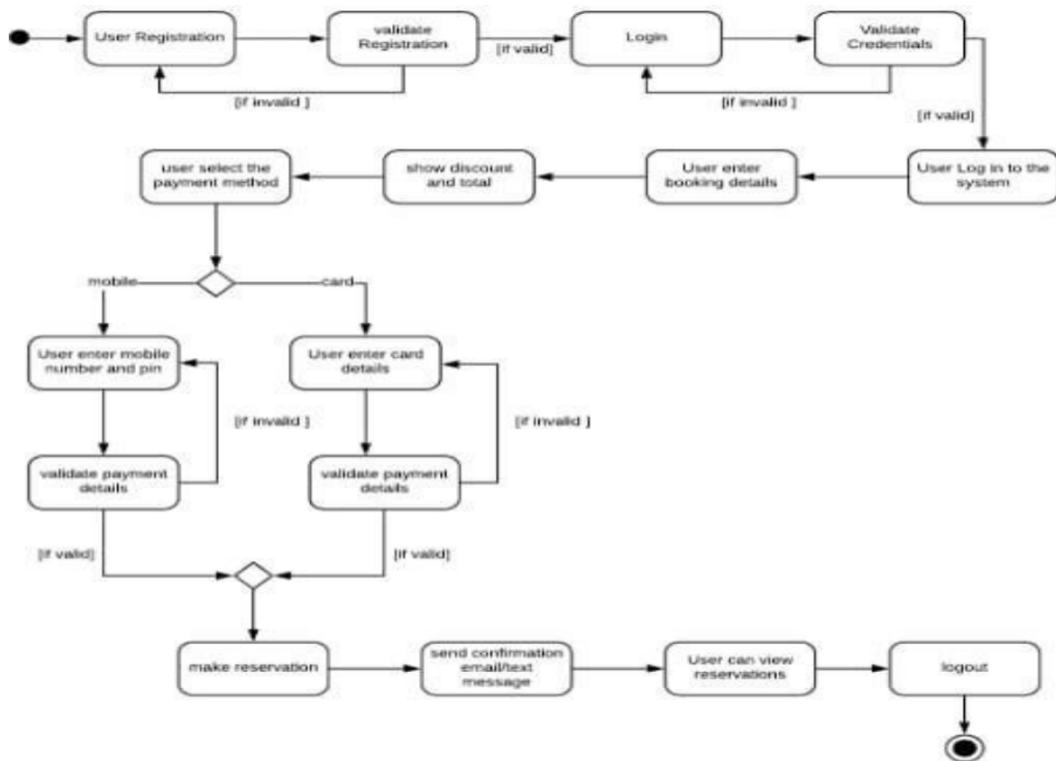
# PROJECT DESIGN

## 5.1) DATA FLOW DIAGRAMS





## 5.2) SOLUTION AND TECHNICAL ARCHITECTURE



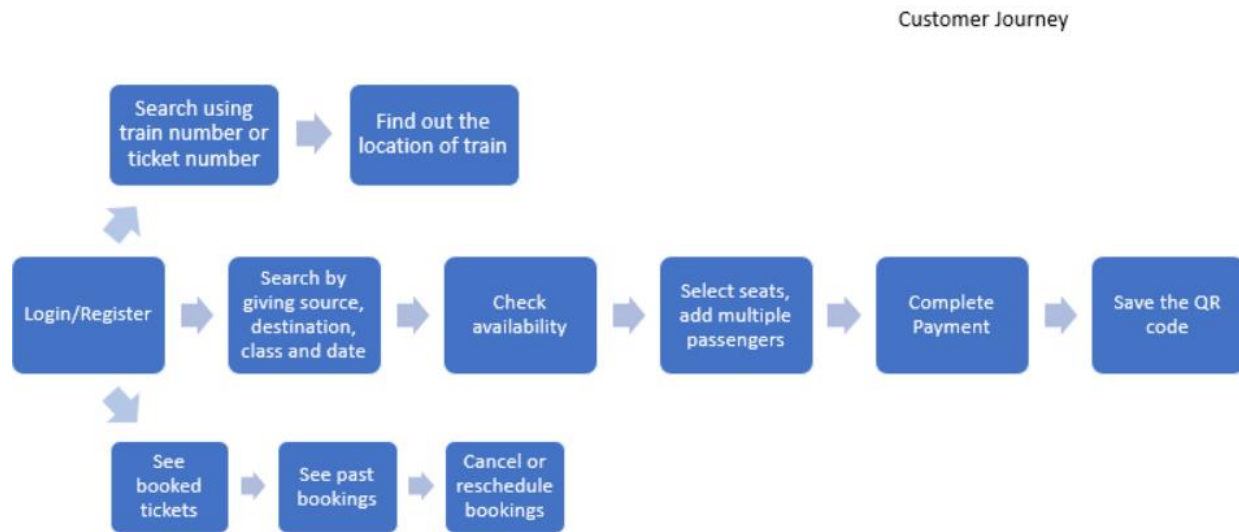
## COMPONENTS AND TECHNOLOGIES

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

## APPLICATION CHARACTERISTICS

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAMControls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Technology used
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

### 5.3) CUSTOMER JOURNEY / USER STORIES



## **PROJECT PLANNING AND SCHEDULING**

### **6.1) SPRINT DELIVERY PLANNING AND ESTIMATION**

<b>Sprint</b>	<b>Total Story Points</b>	<b>Duration</b>	<b>Sprint Start Date</b>
Sprint-1	20	6 Days	24 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022

<b>Sprint End Date (Planned)</b>	<b>Story Points Completed (as on Planned End Date)</b>	<b>Sprint Release Date (Actual)</b>
29 Oct 2022	20	29 Oct 2022
05 Nov 2022	20	05 Nov 2022
12 Nov 2022	20	12 Nov 2022
19 Nov 2022	20	19 Nov 2022

### 6.3) JIRA REPORTS

Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

smart solutions for rail...

Software project

PLANNING

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / smart solutions for railways

Backlog

H

Epic

Insights

SSFR Sprint 1

24 Oct – 29 Oct

(5 issues)

000

Complete sprint

SSFR Sprint 2

31 Oct – 5 Nov

(7 issues)

000

Complete sprint

SSFR Sprint 3

7 Nov – 12 Nov

(6 issues)

000

Complete sprint

SSFR Sprint 4

14 Nov – 19 Nov

(6 issues)

000

Complete sprint

Backlog

(0 issues)

000

Create sprint

Your backlog is empty.

+ Create issue

Quickstart

smart solutions for rail...

Software project

PLANNING

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / smart solutions for railways

# SSFR Sprint 1

Node red service and generation of QR

Q

H

Epic

GROUP BY: None

Insights

TO DO 1 ISSUE

IBM node fix

IN PROGRESS 2 ISSUES

programming nodes

deployment of nodes

IN REVIEW 2 ISSUES

node red flow

generation of qr

DONE

You're in a team-managed project

Learn more

Quickstart

	OCT						NOV						NOV						NOV							
	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Sprints	SSFR Sprint 1						SSFR Sprint 2						SSFR Sprint 3						SSFR Sprint 4							
SSFR-14 python programming																										
SSFR-15 mit app inventor																										
SSFR-16 node red application																										

# CODING AND SOLUTIONING

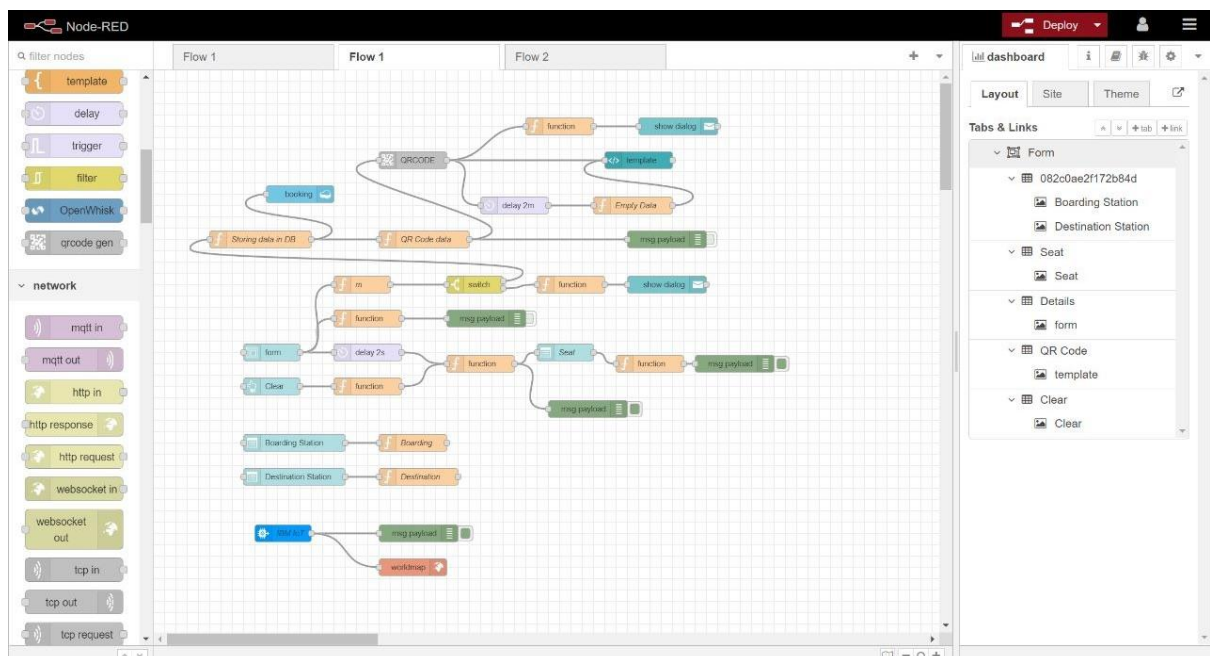
## 7.1) FEATURE 1

### NODE RED SERVICE

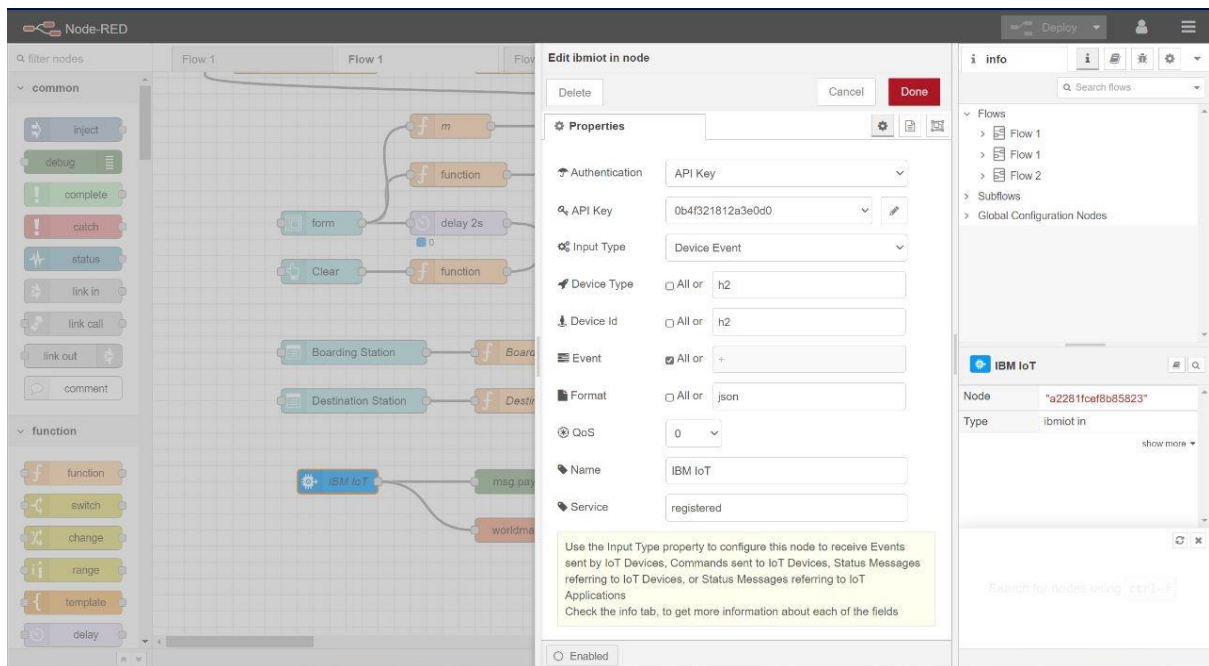
Main application

- Connect to cloud
- View map
- Searching for trains
- Storing database
- Booking
- QR code generation

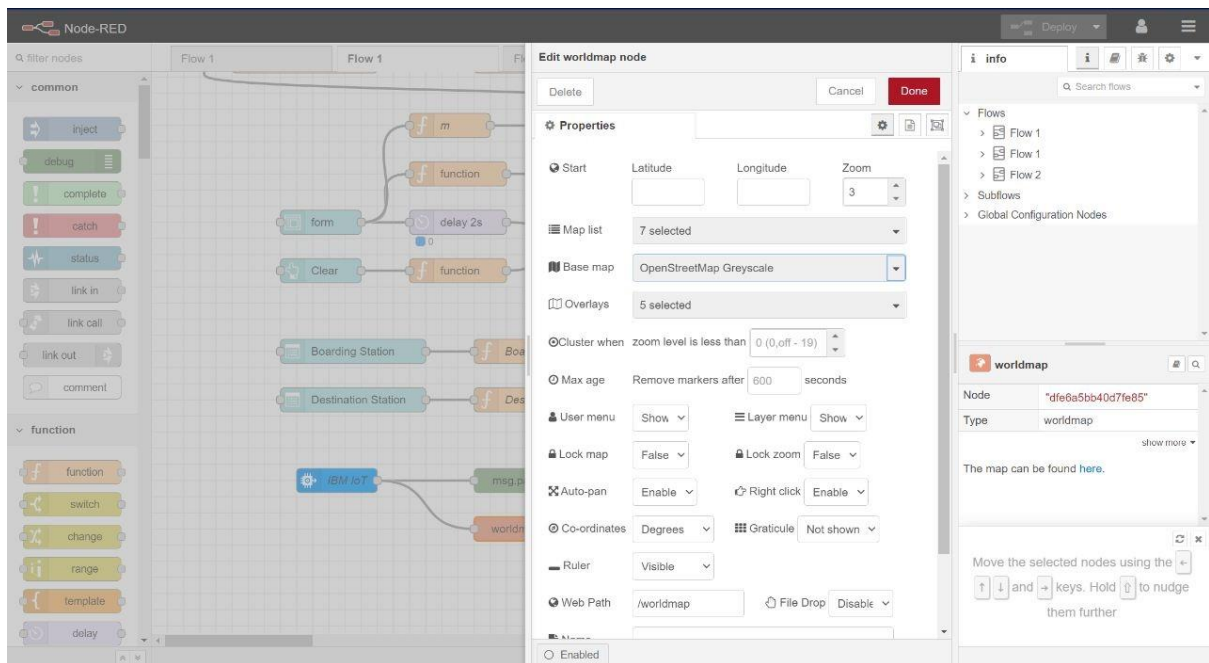
### FLOWCHART



# IBM IOT



# FOR WORLD MAP





## FOR BOARDING STATION

The screenshot shows the Node-RED web interface. On the left, the 'common' node palette includes 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link call', 'link out', and 'comment'. The 'function' palette includes 'function', 'switch', 'change', 'range', 'template', and 'delay'. The main workspace contains a flow with a 'form' node, a 'Clear' button, and two dropdown nodes: 'Boarding Station' and 'Destination Station'. The 'Boarding Station' node is selected, and its configuration is shown in the 'Edit dropdown node' panel. The configuration includes: Group: '[Form] Group', Size: '6 x 1', Label: 'Boarding Station', Tooltip: 'optional tooltip', Placeholder: 'Select option', and Options: a list of four items (Ongole, Hyderabad, Tirupati, Chennai) each with a dropdown arrow and a text input field. The 'Allow multiple selections from list' checkbox is unchecked, and the 'if msg arrives on input, pass through to output' checkbox is checked. The 'Enabled' checkbox is also checked. The right sidebar shows the 'info' panel with a search bar and a list of flows. The 'Boarding Station' node is highlighted, showing its ID 'b01cb2fed24b8dc3' and type 'ui\_dropdown'.

## FOR DESTINATION

The screenshot shows the Node-RED web interface. On the left, the 'common' node palette includes 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link call', 'link out', and 'comment'. The 'function' palette includes 'function', 'switch', 'change', 'range', 'template', and 'delay'. The main workspace contains a flow with a 'form' node, a 'Clear' button, and two dropdown nodes: 'Boarding Station' and 'Destination Station'. The 'Destination Station' node is selected, and its configuration is shown in the 'Edit dropdown node' panel. The configuration includes: Group: '[Form] Group', Size: '6 x 1', Label: 'Destination Station', Tooltip: 'optional tooltip', Placeholder: 'Select option', and Options: a list of four items (Hyderabad, Ongole, Tirupati, Chennai) each with a dropdown arrow and a text input field. The 'Allow multiple selections from list' checkbox is unchecked, and the 'if msg arrives on input, pass through to output' checkbox is checked. The 'Enabled' checkbox is also checked. The right sidebar shows the 'info' panel with a search bar and a list of flows. The 'Destination Station' node is highlighted, showing its ID '1cf5a29844f03b1' and type 'ui\_dropdown'. A tooltip is visible over the 'ctrl+click' text, stating 'ctrl+click in the workspace to open the quick-add dialog'.

## FUNCTION MODE

The screenshot displays the Node-RED web interface. On the left, a flow diagram is visible with nodes including 'booking', 'Storing data in DB', 'QR CODE', 'QR Code data', 'form', 'delay 2s', 'Clear', 'Boarding Station', 'Boarding', 'Destination Station', and 'Destination'. The central panel is titled 'Edit function node' and shows the 'On Message' tab. The code editor contains the following JavaScript code:

```
1 global.set('m',msg.payload)
2 var a = global.get('s')
3 if(a==1 || a==2 || a==3 || a==4 || a==5){
4   msg.payload = 0
5 }
6 else{
7   msg.payload = 1
8 }
9 return msg;
```

The right sidebar shows the 'info' panel with a search bar and a list of flows. Below this, the selected node 'm' is shown with its ID 'f73c6ccad80fa4f2' and type 'function'. At the bottom of the sidebar, there is a button to export the selected nodes or the current tab with the keyboard shortcut 'ctrl-e'.

## DELAY MODE

The screenshot displays the Node-RED web interface. On the left, the same flow diagram is visible, but with a 'delay 2m' node added to the flow. The central panel is titled 'Edit delay node' and shows the 'Action' tab. The configuration is set to 'Delay each message' with a 'Fixed delay' of '5' seconds. The 'Name' field is empty. The right sidebar shows the 'info' panel with a search bar and a list of flows. Below this, the selected node 'delay 2s' is shown with its ID '64cf3399fcca9ab6' and type 'delay'. At the bottom of the sidebar, there is a message: 'Dragging a node onto a wire will splice it into the link'.

## FOR STORING DATABASE

The screenshot shows the Node-RED interface with a flow for storing data in a database. The flow includes a 'booking' node, a 'QR CODE' node, a 'Storing data in DB' function node, and a 'QR Code data' node. The 'Storing data in DB' node is selected, and its configuration is shown in the 'Edit function node' panel.

```
1 var m=global.get('m')
2 var d=new Date();
3 var utc=d.getTime()+d.getTimezoneOffset()*60000;
4 var offset=5.5;
5 newDate=new Date(utc+(3600000*offset));
6 var n=newDate.toISOString()
7 var date=n.slice(8,19)
8 var time=n.slice(11,19)
9 var d=date+', '+time
10
11 msg.payload={
12   "_id":d,
13   "Name":m.Name,
14   "Age":m.Age,
15   "Mobile":m.Num,
16   "Gender":m.Gen,
17   "boarding":global.get('b'),
18   "destination":global.get('d'),
19   "Seat":global.get('s')
20 }
21 return msg;
```

## FOR BOOKING (OUTPUT)

The screenshot shows the Node-RED interface with a flow for booking output. The flow includes a 'booking' node, a 'QR CODE' node, a 'Storing data in DB' function node, and a 'QR Code data' node. The 'booking' node is selected, and its configuration is shown in the 'Edit cloudant out node' panel.

Service: External cloudant or couchdb service  
Server: Add new cloudant...  
Database: booking  
Operation: insert  
☒ Only store msg.payload object?  
Name: Name

# QR CODE GENERATOR

The screenshot displays the Node-RED web interface. On the left, a flow diagram is visible with nodes including 'booking', 'Storing data in DB', 'QR Code data', 'delay 2m', 'form', 'Clear', 'Boarding Station', 'Destination Station', and 'QRcode'. The central panel shows the 'Edit qrcode gen node' configuration. The 'Name' field is set to 'QRcode'. The 'Type' is set to 'Html-Link or Text'. The 'Text or URL' field contains 'https://example.com'. The right sidebar shows the 'info' panel with a search bar and a list of flows. Below the list, the 'QRcode' node is highlighted, showing its ID '771e00ab9c084301' and type 'qrcode-generator'.

# FUNCTION NODE TEMPLATE

The screenshot displays the Node-RED web interface. On the left, a flow diagram is visible with nodes including 'booking', 'Storing data in DB', 'QR Code data', 'delay 2m', 'form', 'Clear', 'Boarding Station', 'Destination Station', and 'QRcode'. The central panel shows the 'Edit template node' configuration. The 'Template type' is set to 'Widget in group'. The 'Group' is set to '[Form] QR Code'. The 'Size' is set to '5 x 5'. The 'Class' field is empty. The 'Name' field is set to 'Name'. The 'Template' field contains the HTML code: `<img src={{msg.payload}} />`. The right sidebar shows the 'info' panel with a search bar and a list of flows. Below the list, the 'template' node is highlighted, showing its ID 'c704f4a6e5cd8418' and type 'ui\_template'.

## 7.2) FEATURE 2

### QR CODE SCANNER

- SCAN QR CODE AND VIEW DETAILS OF THE PASSENGER

```
{ '_id': '2022-11-16,00:23:01', '_rev': '1-308e83e170520d7d1c05e7c13be16eb8', 'Name': 'vijay', 'Age': 20, 'Mobile': 9873216540, 'boarding': 'Sale  
m', 'destination': 'Coimbatore', 'Seat': 3}  
{ '_id': '2022-11-16,00:23:01', '_rev': '1-308e83e170520d7d1c05e7c13be16eb8', 'Name': 'vijay', 'Age': 20, 'Mobile': 9873216540, 'boarding': 'Sale  
m', 'destination': 'Coimbatore', 'Seat': 3}  
{ '_id': '2022-11-16,00:23:01', '_rev': '1-308e83e170520d7d1c05e7c13be16eb8', 'Name': 'vijay', 'Age': 20, 'Mobile': 9873216540, 'boarding': 'Sale  
m', 'destination': 'Coimbatore', 'Seat': 3}
```



### VIEW LOCATION OF TRAINS

- GET THE COORDINATES OF THE TRAIN
- VIEW TRAIN LOCATION IN MOBILE APPLICATION

# TESTING

## 8.1) TEST CASES

### Test case 1

Form

Boarding Station

Ongole

Destination

Hyderabad

Seat

1

Clear

Details

Name \*

koushik

Age \*

21


Mobile.No \*

9999192928

SUBMIT

CANCEL

QR Code



## TICKET RESERVATION

QR Code

Age \*

Mobile.No \*

CANCEL

Ticket Booked

OK

# DATABASE

The screenshot shows the IBM Cloudant Dashboard interface. The browser address bar displays the URL: `https://b849b986-ec7d-4ab7-a888-31983aa929d4-bluemix.cloudant.com/dashboard.html#/database/ibm_railways/_all_docs`. The left sidebar contains navigation options: All Documents, Query, Permissions, Changes, and Design Documents. The main content area shows a table of documents with columns 'id', 'key', and 'value'. The table lists five documents, each with a unique ID and key, and a JSON value. A 'Create Document' button is visible in the top right corner. At the bottom, it indicates 'Showing document 1 - 5' and 'Documents per page: 20'.

id	key	value
2022-11-18,00:04:21	2022-11-18,00:04:21	{ "_rev": "1-e5e53903061108316efb67d1884..." }
2022-11-18,00:39:38	2022-11-18,00:39:38	{ "_rev": "1-71e451866573d5a1dfc0aa62b412..." }
2022-11-18,00:48:07	2022-11-18,00:48:07	{ "_rev": "1-6a00bbd31873e68a7d6468f7870c..." }
2022-11-18,09:08:34	2022-11-18,09:08:34	{ "_rev": "1-0cced0c304b2cba862e4373796e4..." }
2022-11-18,09:24:26	2022-11-18,09:24:26	{ "_rev": "1-a58deaae253fed4f1aa6cab018d38..." }

The screenshot shows the IBM Cloudant Dashboard interface for a specific document. The browser address bar displays the URL: `https://b849b986-ec7d-4ab7-a888-31983aa929d4-bluemix.cloudant.com/dashboard.html#/database/ibm_railways/2022-11-18%2C00%3A48%3A07`. The left sidebar contains navigation options: All Documents, Query, Permissions, Changes, and Design Documents. The main content area shows the details of a document with ID '2022-11-18,00:48:07'. The document is displayed in a JSON format. A 'Save Changes' button is visible in the top left corner. At the top right, there are buttons for 'Upload Attachment', 'Clone Document', and 'Delete'.

```
{
  "_id": "2022-11-18,00:48:07",
  "_rev": "1-6a00bbd31873e68a7d6468f7870c2a53",
  "Name": "Koushik",
  "Age": 21,
  "Mobile": 9999192918,
  "boarding": "Ongole",
  "destination": "Hyderabad",
  "Seat": 3
}
```

## Test case 2

Form

Boarding Station: Vijayawada

Destination: Nellore

Seat: Select option

QR Code

Details

Name \*

Age \*

Mobile Number \*

Ticket is Generated/Reserved

OK

Clear

CLEAR

ibm\_railways > 2022-11-18,10:52:25

Save Changes Cancel

Upload Attachment Clone Document Delete

```
1- 1
2  {
3    "_id": "2022-11-18,10:52:25",
4    "_rev": "1-20543612dd34d8fe0886ac8c9fc60da9",
5    "Name": "Harsha",
6    "Age": 21,
7    "Mobile": 9999999917,
8    "boarding": "Vijayawada",
9    "destination": "Nellore",
10   "Seat": 2
11 }
```

Log Out




# RESULTS

Form

Boarding Station: Vijayawada

Destination: Nellore

QR Code



Clear

CLEAR

Seat: Select option

Details

Name \*

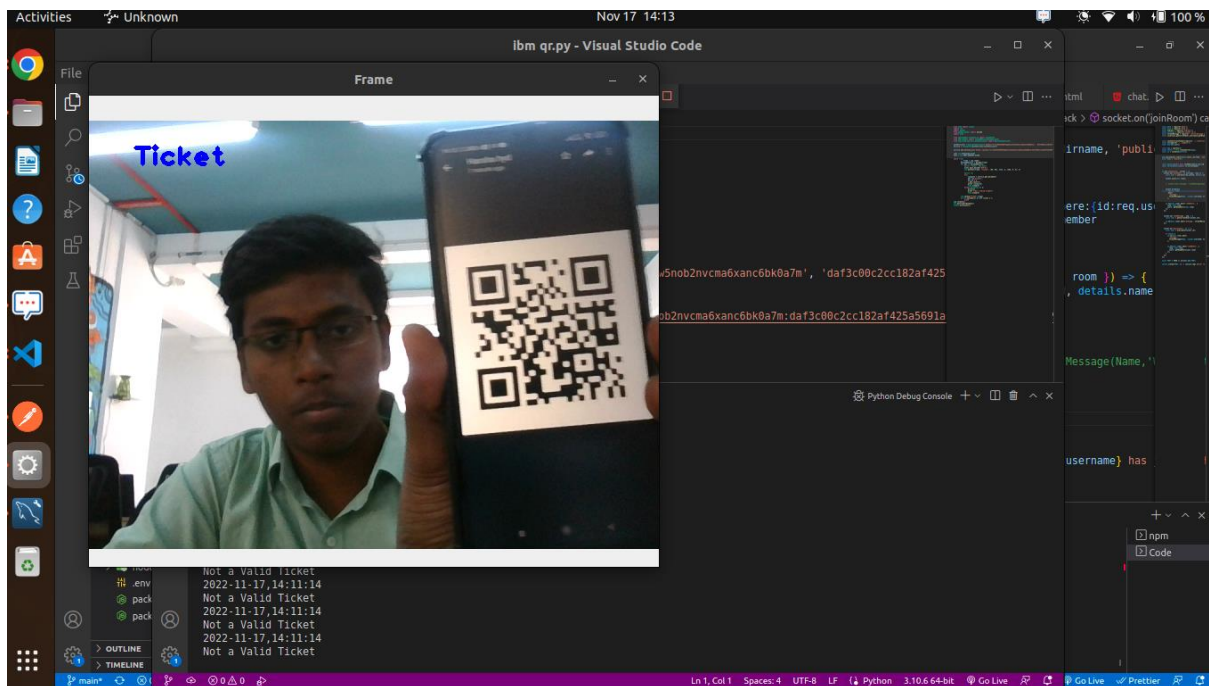
Age \*

Mobile No \*

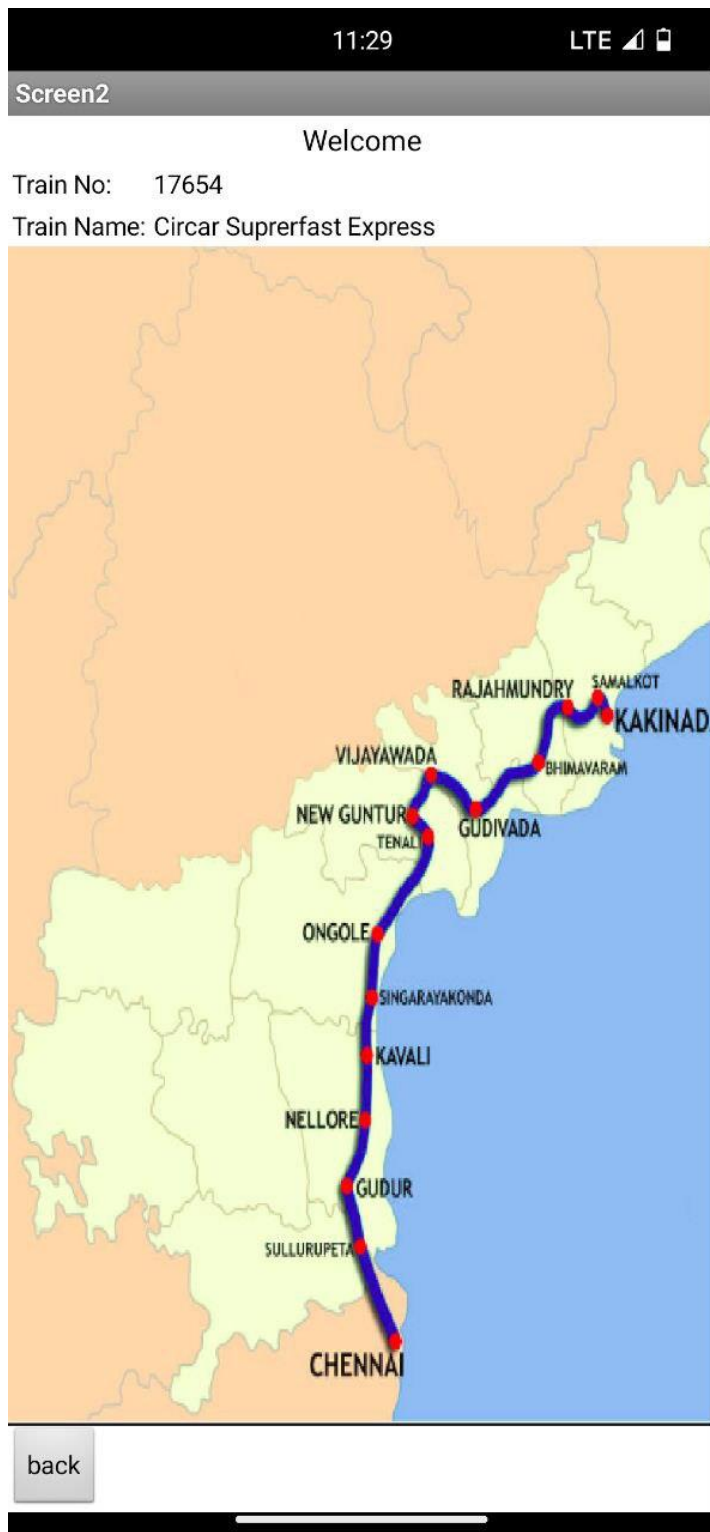
CEL

Ticket is Generated/Reserved

OK



## Location in mobile application



## **11) CONCLUSION**

Using the application, user can book train tickets based on availability of seats in particular train. Once tickets are available, they can book them by inputting their general information. Upon completion of payment, the data gets stored in Cloudant DB with unique ID for every transaction and a QR code is generated for every ticket. The ticket collector can scan the QR code to get information of the passenger, if the QR is correct, the details of the user are displayed, if the QR is invalid, it displays "Not a Valid ticket". Apart from ticketing, our application also allows the users to find out the live location and running status of the train.

## **12) FUTURE SCOPE**

Cloud computing and IOT are integrated now to ease the ticketing system and tracking in railways. In near future, Internet of Things and Artificial Intelligence can be combined to make railways safer and faster. Artificial Intelligence can be used to determine delay and arrival time so that the passenger can act accordingly. By the use of Internet of Things, things such as maintenance of tracks, repairs and services can be carried out with ease.

## 13) APPENDIX

### 13.1) SOURCE CODE

#### QR SCAN CODE:

```
from http import client

import cv2

import pyzbar

from pyzbar.pyzbar import decode

import time


from ibmcloudant.cloudant_v1 import CloudantV1

from ibmcloudant import CouchDbSessionAuthenticator

from ibm_cloud_sdk_core.authenticators import BasicAuthenticator


authenticator = BasicAuthenticator('apikey-v2-x5kge2ue09set5w3y7zqwppcsu8yssidj438ufbetnx',
'75782580a79baa3efdec4c2191edd956')

service = CloudantV1(authenticator=authenticator)


service.set_service_url('https://apikey-v2-
x5kge2ue09set5w3y7zqwppcsu8yssidj438ufbetnx:75782580a79baa3efdec4c2191edd956@b849b9
86-ec7d-4ab7-a888-31983aa929d4-bluemix.cloudantnosqldb.appdomain.cloud')

cap= cv2.VideoCapture(0)

font = cv2.FONT_HERSHEY_PLAIN


while True:

    _, frame = cap.read()
```

```
decodedObjects = decode(frame)

for obj in decodedObjects:

    #print ("Data", obj.data)

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

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


    #print (a)

    try:

        response = service.get_document(

            db='IBM_railways',

            doc_id = a

        ).get_result()

        print (response)

        time.sleep(5)

    except Exception as e:

        print(a)

        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()
```

## LOCATION CODE

```
import wiotp.sdk.device
```

```
import time
```

```
import random
```

```
myConfig = {
```

```
    "identity": {
```

```
        "orgId": "ekscpp",
```

```
        "typeId": "h1",
```

```
        "deviceId": "h1"
```

```
    },
```

```
    "auth": {
```

```
        "token": "12345678"
```

```
    }
```

```
}
```

```
def myCommandCallback(cmd):
```

```
    print("The Message received from IBM IoT Platform is : %s" % cmd.data['command'])
```

```
    m=cmd.data['command']
```

```
def pub(data):
```

```
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
```

```
    print("Data is published Successfully:%s",myData)
```

```
client = wiotp.sdk.device.DeviceClient(config=myConfig)

client.connect()

while True:

    myData={'name':'Train1','lat':10.184363,'lon': 77.922702}

    pub(myData)

    time.sleep(3)

    myData={'name':'Train1','lat':10.213225,'lon': 77.898765}

    pub(myData)

    time.sleep(3)

    myData={'name':'Train1','lat':10.285035,'lon': 77.921569}

    pub(myData)

    time.sleep(3)

    myData={'name':'Train1','lat':10.343369,'lon': 77.958056}

    pub(myData)

    time.sleep(3)

    myData={'name':'Train1','lat':10.356829,'lon': 77.980861}

    pub(myData)

    time.sleep(3)

    client.commandCallback = myCommandCallback

client.disconnect()
```

### **13.2) Github and Project Demo Link**

**<https://github.com/IBM-EPBL/IBM-Project-10120-1659096994>**

#### **Demo video Link**

**[https://github.com/IBM-EPBL/IBM-Project-10120-1659096994/blob/main/Final%20Deliverables/demo\\_video1.mp4](https://github.com/IBM-EPBL/IBM-Project-10120-1659096994/blob/main/Final%20Deliverables/demo_video1.mp4)**