PROJECT REPORT

PROJECT NAME : SMART SOLUTIONS FOR RAILWAYS

TEAM ID : PNT2022TMID03560

TEAM LEADER : GOLLA VINAY KUMAR

TEAM MEMBERS : VADLAMUDI SAIGURU KRISHNA

KALAMAKUNTLA BHAVANI

THEETLA RAJ VARUN KUMAR

TABLE OF CONTENTS

1.INTRODUCTION

1.1 PROJECT OVERVIEW

This software is used to book a train ticket using a customizable online user interface. By purchasing the ticket using the online interface, the user receives a unique ID and generates a QR code that comprises all of their personal information, including boarding and destination. The QR code is mostly used for checking, which facilitates the task of verifying the authenticity of the ticket for the ticket checker. After making a reservation, the user will receive a special ID and QR code. Via the perspective of the ticket checkers, they can receive a unique login from the online UI. The ticket checker scans the QR code using a QR code reader.The Ticket Checker can access the passenger's booking information from the cloud IOT by scanning the QR code.

1.2 PURPOSE

The Internet is required for computer networks to connect. However, as the globe evolves, its use is becoming more widespread than merely online surfing and email. The modern internet, which also interacts with embedded sensors, has led to the development of smart homes, smart rural communities, and e-health. Care's etc. established the concept of IOT. The term "Internet of Things" refers to the connection or communication between two or more devices without any human-to-human or humanto-computer contact. Connected devices employ sensors or actuators to perceive the environment around them. The four essential components of IOT include sensing the device will get access to the device, processing the device's data, and providing applications and services. Additionally, it provides data security and privacy.Automation has an effect on every aspect of our everyday lives. More developments are being developed almost in every business to decrease human effort and save time. When attempting to automate track testing, the same is taken into account. Railroad track is an essential part of every company's asset base since it makes it possible for them to run their operations as normal. Issues with railways must be resolved in order to avoid more problems. The most current method used by the Indian railroad necessitates spending a lot of time and effort following the railway lines..

2.LITERATURE SURVEY

PAPER NAME	AUTHOR	YEAR	METHOLOGY	MERITS	DEMERITS

Passenger Monitoring Model for easily Accessible Public City Trams/Trains.	Roman Khoeblal, Teeravisit Laohapensaeng, Roungsan Chaisricharoen	2015	Passenger monitoring, passenger control RFID distance reading, ticket control, RFID ticket inspection.	It is possible to travel cross country with a single public transportation card, using transport systems of several transport operators.	Applicable only for passenger monitoring.
Application of smart computing in Indian Railway Systems.	Parag Chatterjee, Asoke Nath	2014	By Interlinking unique identification system with train ticket reservation system by using video surveillance, rail sensors, biometric input devices and multimedia displays.	Reduces manual effort in passenger data entry. Provides security verification.	Significant investment is needed. Risk of database.
Android Suburban Railway Ticketing with GPS as Ticket Checker.	Sana Khoja, Maithili Kadam	2012	Android, SQ lite, Cloud Database, ASR, QR Code.	E-Ticket facility, enabling reuse and replacement of components.	QR Codes before the user enters or leaves the station, where the user can have access which is risk in ticket booking.

Novel Approach for Smart Indian Railways.	Sujith Kumar, K.M.Yatheendra Parvan, V.Sumathy, Thejeswari C.K	2017	Digitalization, Smart Railways, Aadhar Card, Smartphone, Identity Verification.	Employ a mobile application through which passengers can access various ticketing options in user friendly and efficient manner.	Biometric database is risk of hacking.
A Review on IOT based automated seat allocation and verification using QR code.	Sarvath Saba, Sharon Philip, Shriharsha, Mukund Naik, Sudeep Sherry	2022	The system lets the passenger to have a comfortable journey by checking the temperature first for normal and then the count for avoid crowd using the QR Code.	This model proposes a radical change in train operation and passenger experience. One of the many steps towards a more digitized society as a part of the "Digital India" movement proposed in 2015 by the Prime Minister.	The system is not fool-proof and requires a dramatic change in the existing system in terms of the people allowed on platforms, etc. but baby steps matter.

2.2 REFERENCES

1. Roman Khoeblal, Teeravisit Laohapensaeng, Roungsan Chaisricharoen, “Passenger Monitoring Model for easily Accessible Public City Trams/Trains” (2015).

2. Parag Chatterjee, Asoke Nath, “Application of smart computing in IndianRailway Systems” (2014).

3. Sana Khoja, Maithili Kadam, “Android Suburban Railway Ticketing with GPS as Ticket Checker”

4. Sujith Kumar, K.M.Yatheendra Parvan, V.Sumathy, Thejeswari C.K, “Novel Approach for Smart Indian Railways” (2017). 5. Sarvath Saba, Sharon Philip, Shriharsha, Mukund Naik, Sudeep Sherry, “A Review (2012).

2.3 PROBLEM STATEMENT

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS - 1	User	Book a ticket through application	Unable to book ticket properly	Lack of Guidance in those application	Cofused
PS – 2	Passenger	Book a train Seat Berth	Not Sure information about the berth	Evert seating showing as same	Irritated
PS – 3	Passenger	Give a feedback or complaint about my journey	I couldn't able to do that	There is no option like that in application	Hate
PS – 4	Government	Avoid Ticketless traveling in Railways	Some people are not following the rule	There is no checking while entering the platform	Worst

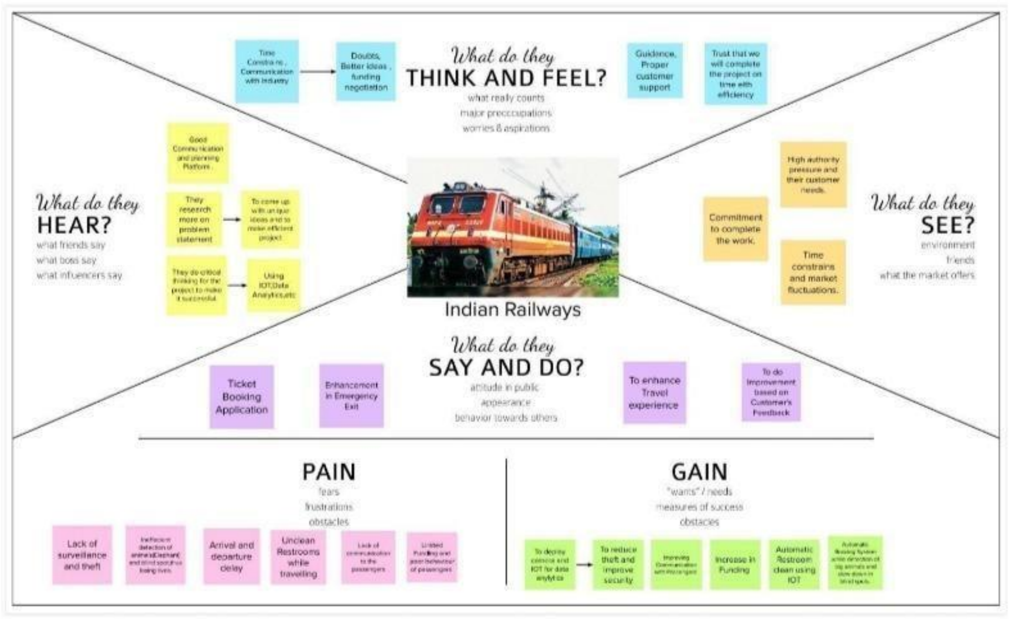
3.IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

Empathy Map Canvas:

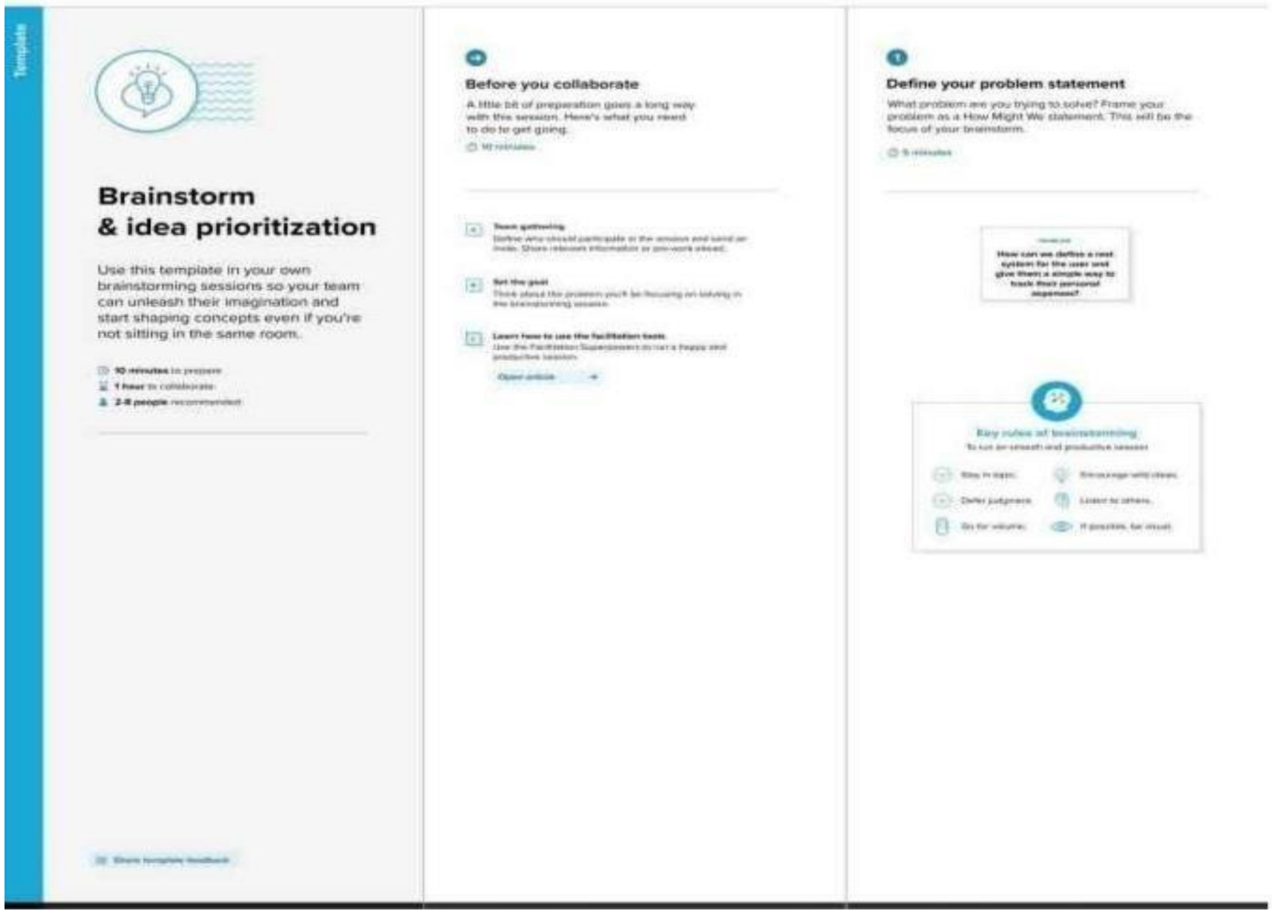
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

Reference:
https://app.mural.co/invitation/mural/smartinternzibmiotsmartsolut6184/1662790391718?sender=ub72a907284043284ab647148&key=b27221cd-3c1b-44a4-bd15-1947bfd7faff



Time Constrains, Communication with Industry
Doubts, Better ideas, funding negotiation
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Guidance, Proper customer support
Trust that we will complete the project on time with efficiency
Good Communication and planning Platform.
What do they HEAR?
what friends say
what boss say
what influencers say
They research more on problem statement
To come up with unique ideas and to make efficient project
They do critical thinking for the project to make it successful
Using IOT,Data Analytics,etc
Indian Railways
High authority pressure and their customer needs.
Commitment to complete the work.
Time constrains and market fluctuations.
What do they SEE?
environment
friends
what the market offers
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Ticket Booking Application
Enhancement in Emergency Exit
To enhance Travel experience
To do Improvement based on Customer's Feedback
PAIN
fears
frustrations
obstacles
Lack of surveillance and theft
Arrival and departure delay
Unclean Restrooms while travelling
Lack of communication to the passengers
Limited Funding and poor behaviour of passengers
GAIN
"wants" / needs
measures of success
obstacles
To deploy camera and IOT for data analytics
To reduce theft and improve security
Improving Communication with Passengers
Increase in Funding
Automatic Restroom clean using IOT

3.2 IDEATION AND BRAINSTORMING



Template

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
- **2-8 people** recommended

Share template feedback

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 →

1 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

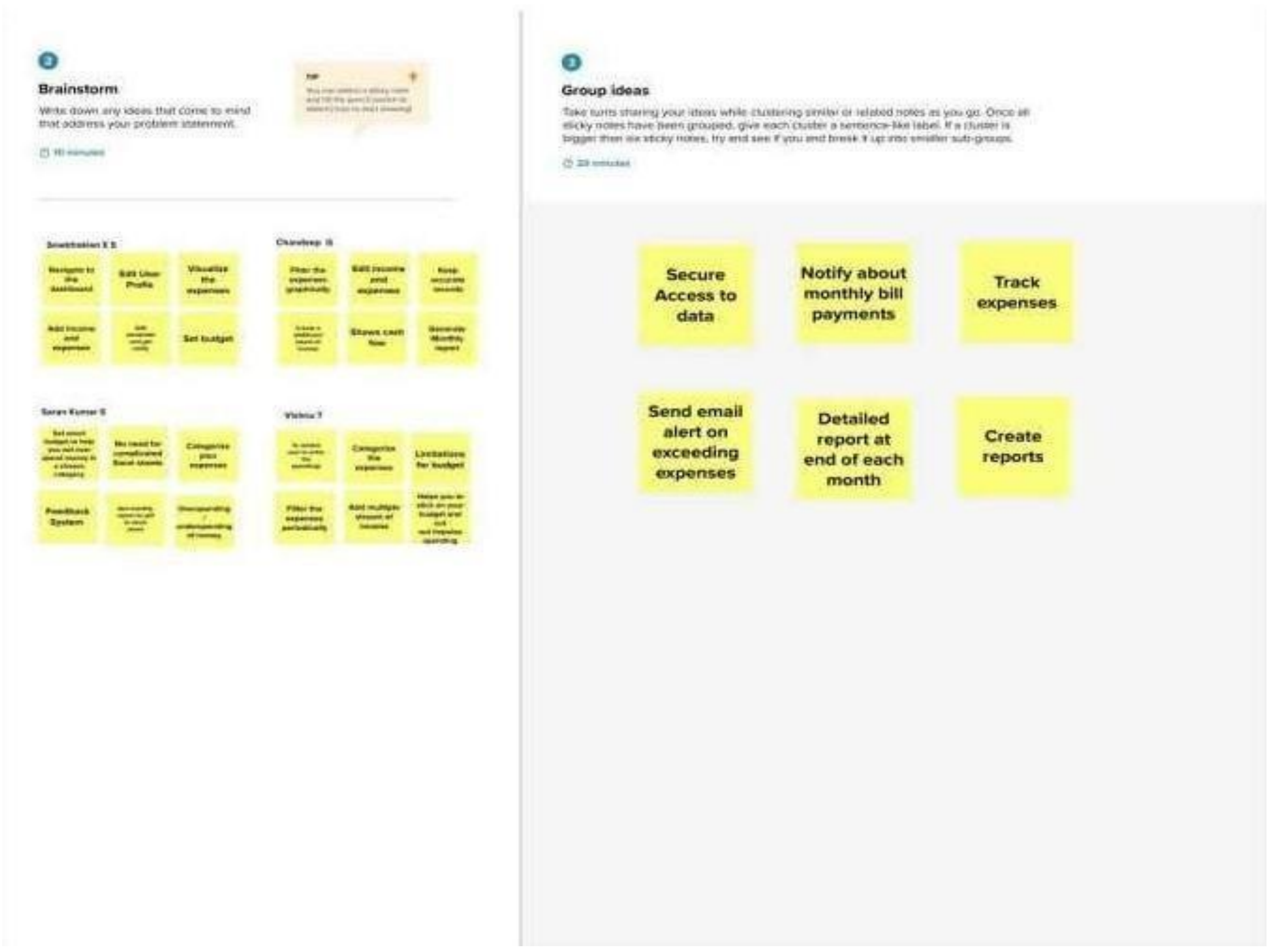
[illegible]

How can we define a rest system for the user and give them a simple way to track their personal expenses?

Key rules of brainstorming

To run an smooth and productive session

Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.



Brainstorm

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

10 minutes

TIP

[illegible]

Sreekrishnan E S

- Navigate to the dashboard
- Edit User Profile
- Visualize the expenses
- Add income and expenses
- [illegible]
- Set budget

Chandeep S

- Filter the expenses graphically
- Edit income and expenses
- Keep accurate records
- [illegible]
- Shows cash flow
- Generate Monthly report

Saran Kumar S

- Set smart budget to help you not over spend money in a chosen category
- No need for complicated Excel sheets
- Categorize your expenses
- Feedback System
- [illegible]
- Overspending / underspending of money

Vishnu T

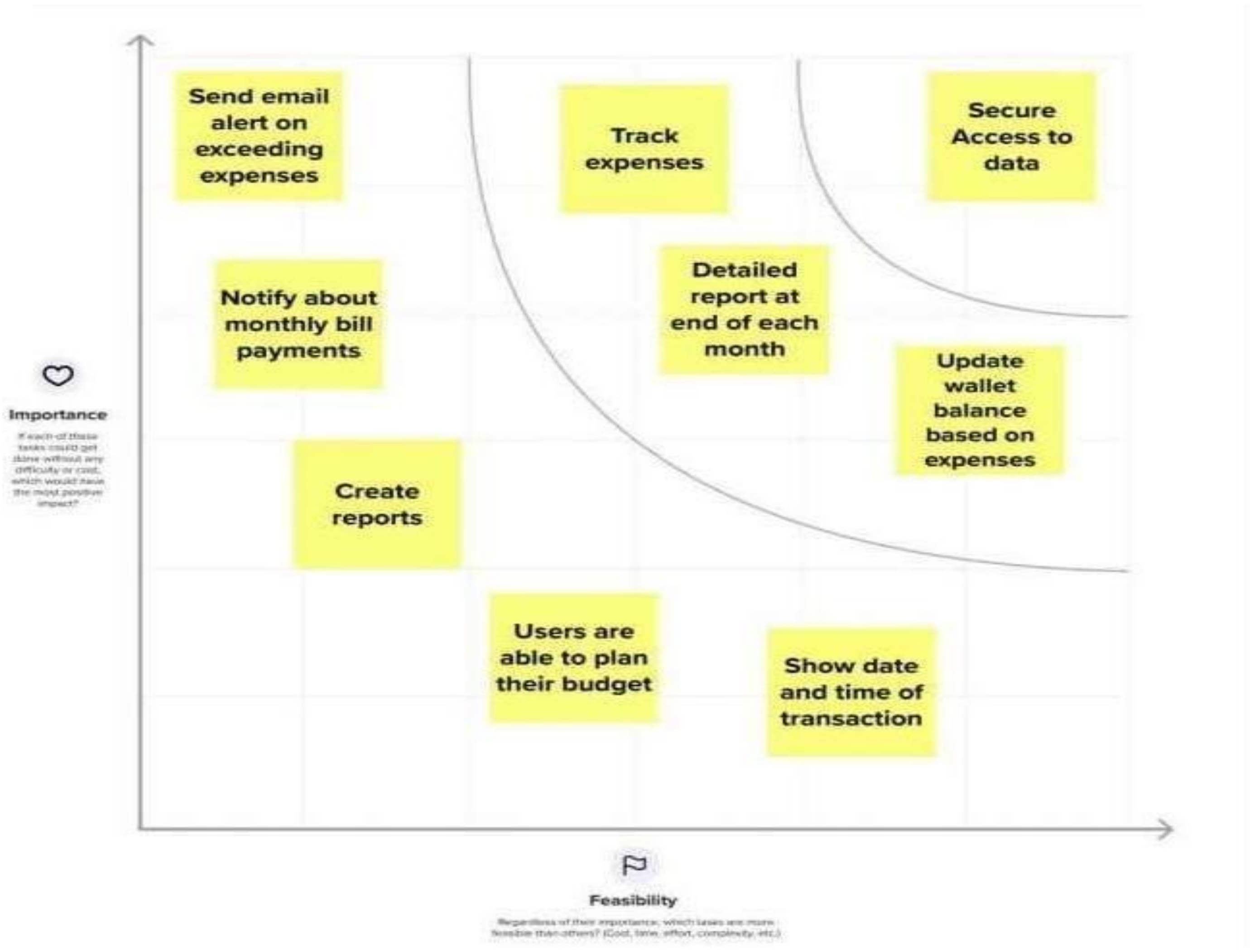
- [illegible]
- Categorize the expenses
- Limitations for budget
- Filter the expenses periodically
- Add multiple stream of income
- Helps you to stick on your budget and cut out impulsive spending

Group ideas

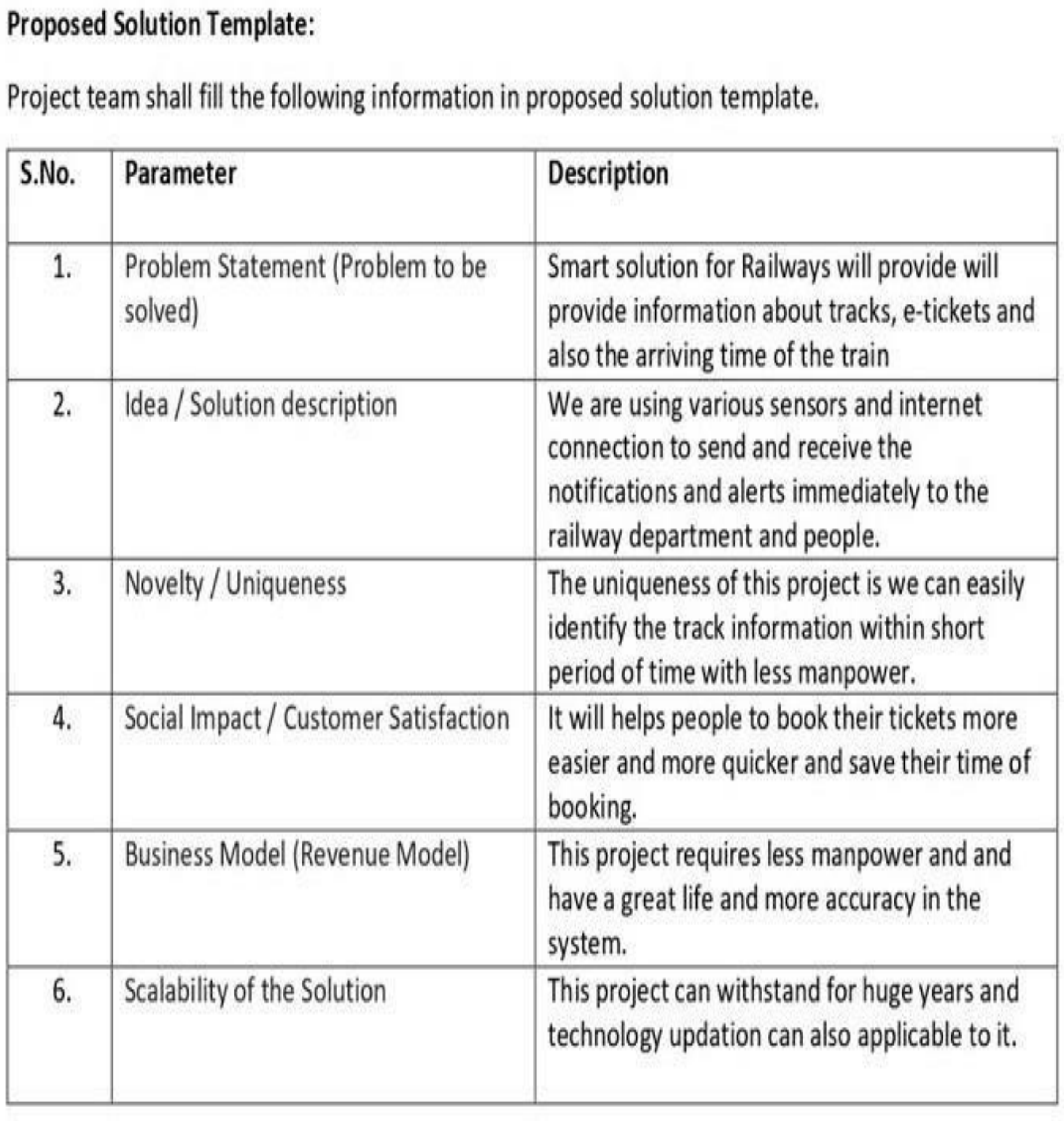
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

- Secure Access to data
- Notify about monthly bill payments
- Track expenses
- Send email alert on exceeding expenses
- Detailed report at end of each month
- Create reports


Send email alert on exceeding expenses
Track expenses
Secure Access to data
Notify about monthly bill payments
Detailed report at end of each month
Update wallet balance based on expenses
Create reports
Users are able to plan their budget
Show date and time of transaction
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 PROPOSED SOLUTION

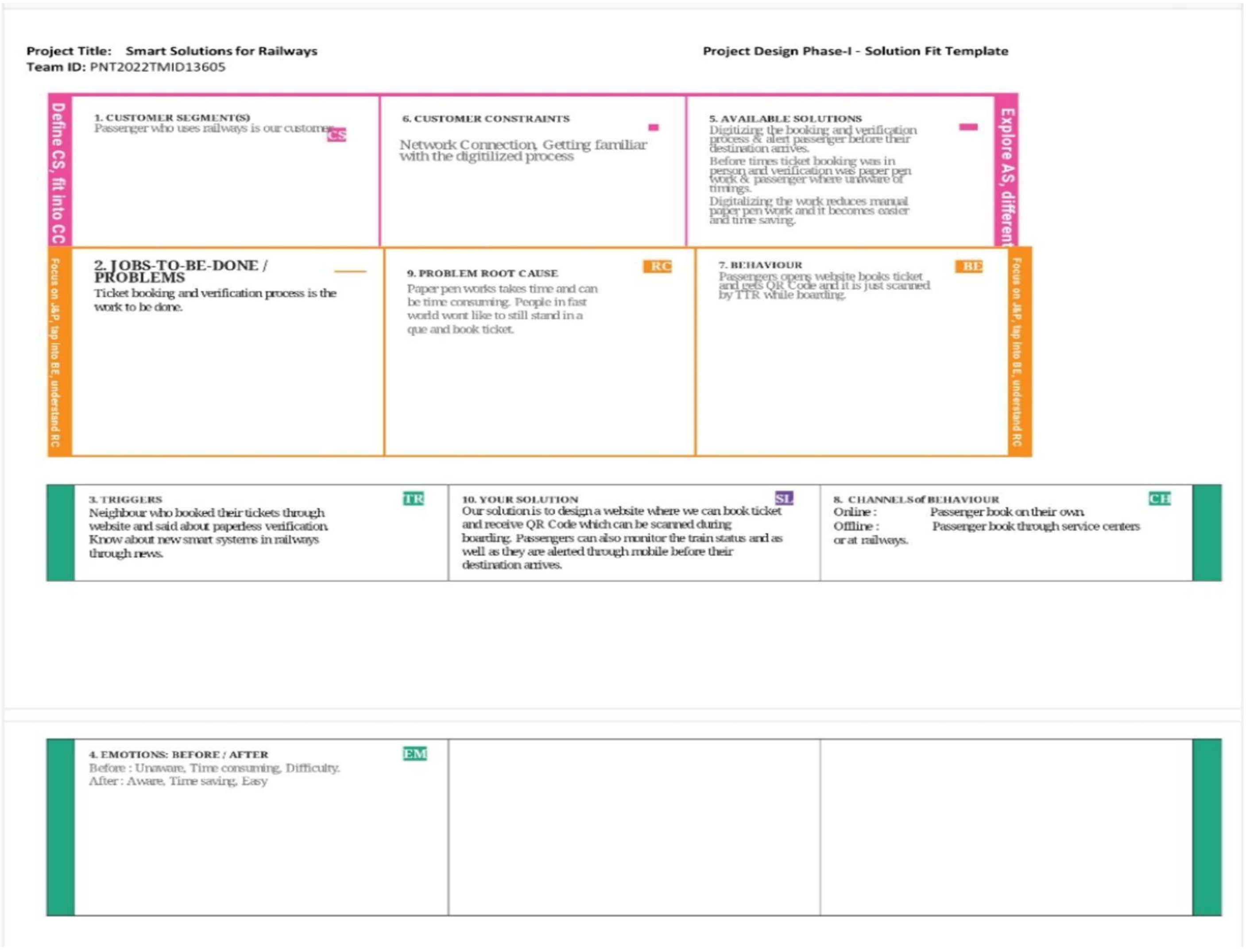


Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Smart solution for Railways will provide will provide information about tracks, e-tickets and also the arriving time of the train
2.	Idea / Solution description	We are using various sensors and internet connection to send and receive the notifications and alerts immediately to the railway department and people.
3.	Novelty / Uniqueness	The uniqueness of this project is we can easily identify the track information within short period of time with less manpower.
4.	Social Impact / Customer Satisfaction	It will helps people to book their tickets more easier and more quicker and save their time of booking.
5.	Business Model (Revenue Model)	This project requires less manpower and and have a great life and more accuracy in the system.
6.	Scalability of the Solution	This project can withstand for huge years and technology updation can also applicable to it.

3.4 PROBLEM SOLUTION FIT



Project Title: Smart Solutions for Railways
Team ID: PNT2022TMID13605

Project Design Phase-I - Solution Fit Template

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS
Passenger who uses railways is our customer

6. CUSTOMER CONSTRAINTS
Network Connection, Getting familiar with the digitilized process

5. AVAILABLE SOLUTIONS
Digitizing the booking and verification process & alert passenger before their destination arrives.
Before times ticket booking was in person and verification was paper pen work & passenger where unaware of timings.
Digitalizing the work reduces manual paper pen work and it becomes easier and time saving.

Explore AS, different

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS
Ticket booking and verification process is the work to be done.

9. PROBLEM ROOT CAUSE RC
Paper pen works takes time and can be time consuming. People in fast world wont like to still stand in a que and book ticket.

7. BEHAVIOUR BE
Passengers opens website books ticket and gets QR Code and it is just scanned by TTR while boarding.

Focus on J&P, tap into BE, understand RC

3. TRIGGERS TR
Neighbour who booked their tickets through website and said about paperless verification. Know about new smart systems in railways through news.

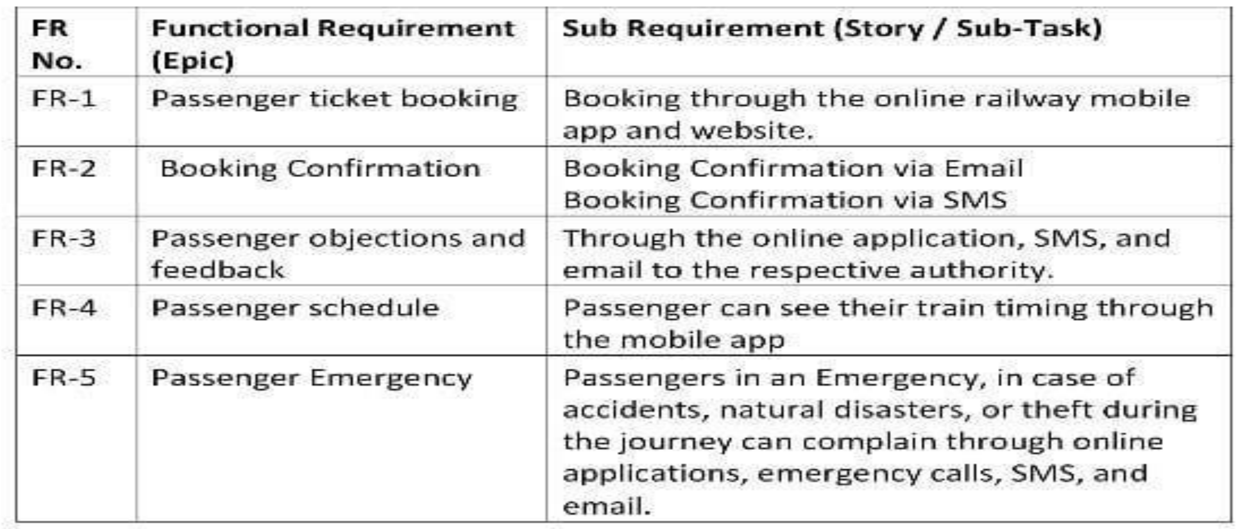
10. YOUR SOLUTION SL
Our solution is to design a website where we can book ticket and receive QR Code which can be scanned during boarding. Passengers can also monitor the train status and as well as they are alerted through mobile before their destination arrives.

8. CHANNELS of BEHAVIOUR CH
Online : Passenger book on their own.
Offline : Passenger book through service centers or at railways.

4. EMOTIONS: BEFORE / AFTER EM
Before : Unaware, Time consuming, Difficulty.
After : Aware, Time saving, Easy

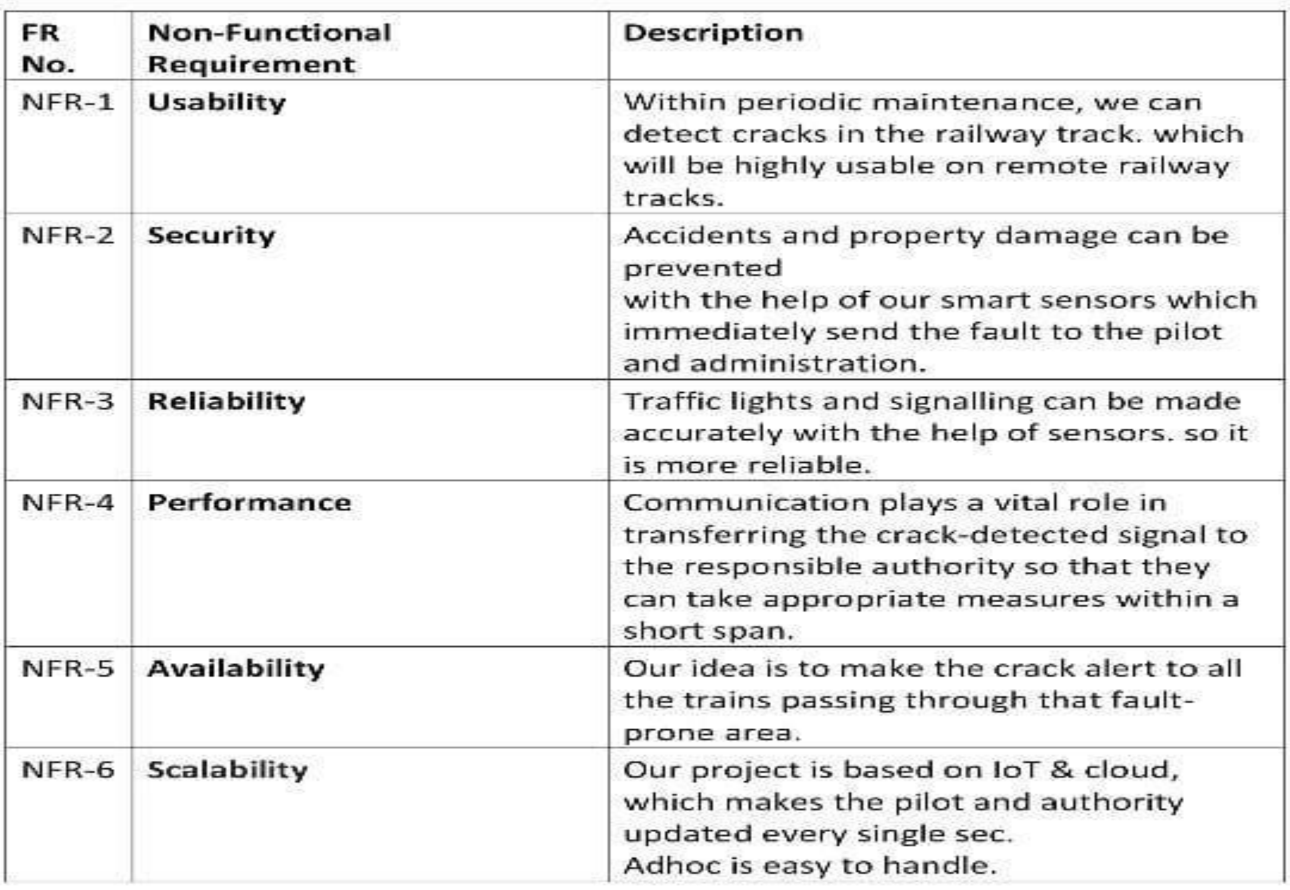
4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS



FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Passenger ticket booking	Booking through the online railway mobile app and website.
FR-2	Booking Confirmation	Booking Confirmation via Email Booking Confirmation via SMS
FR-3	Passenger objections and feedback	Through the online application, SMS, and email to the respective authority.
FR-4	Passenger schedule	Passenger can see their train timing through the mobile app
FR-5	Passenger Emergency	Passengers in an Emergency, in case of accidents, natural disasters, or theft during the journey can complain through online applications, emergency calls, SMS, and email.

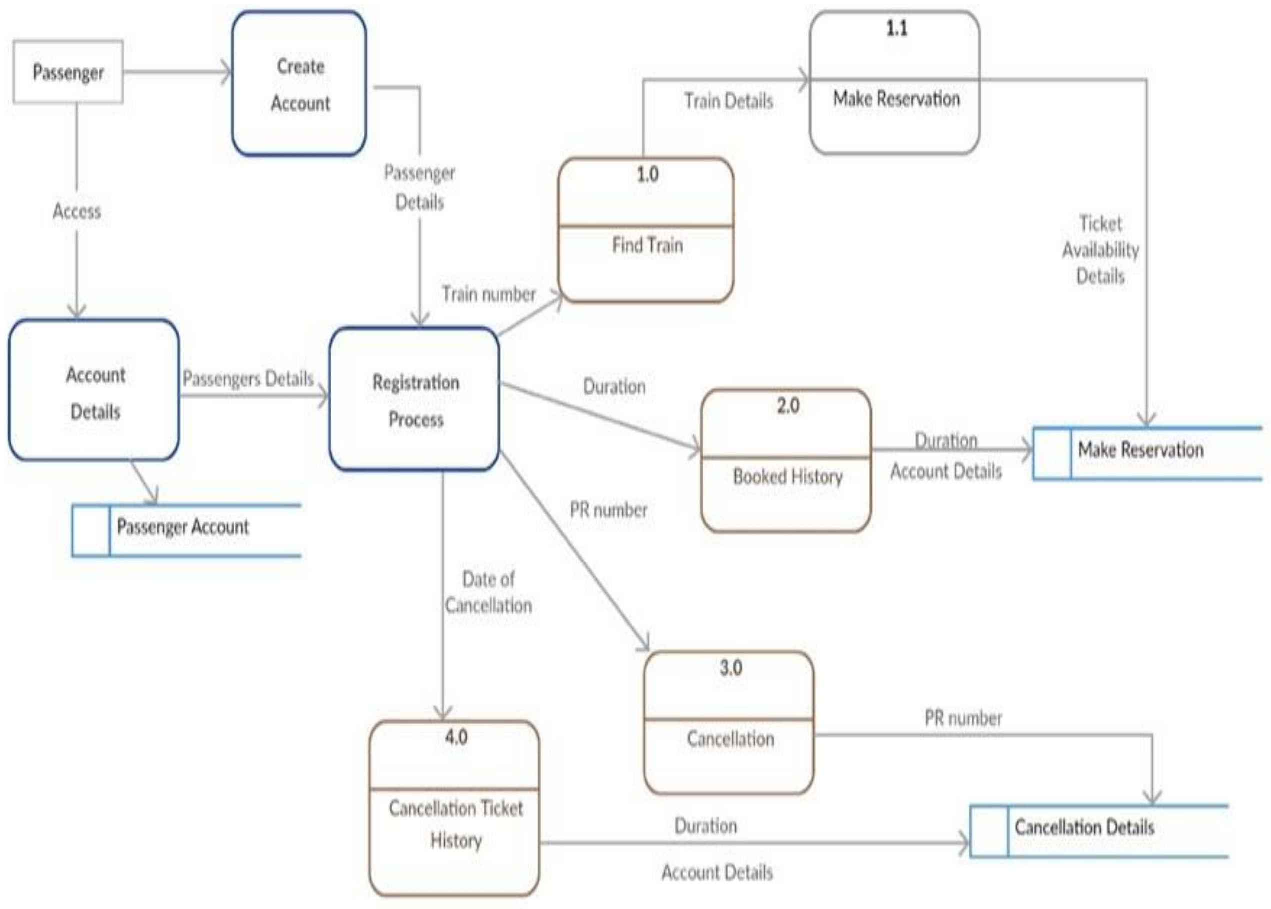
4.2 NON – FUNCTIONAL REQUIREMENTS



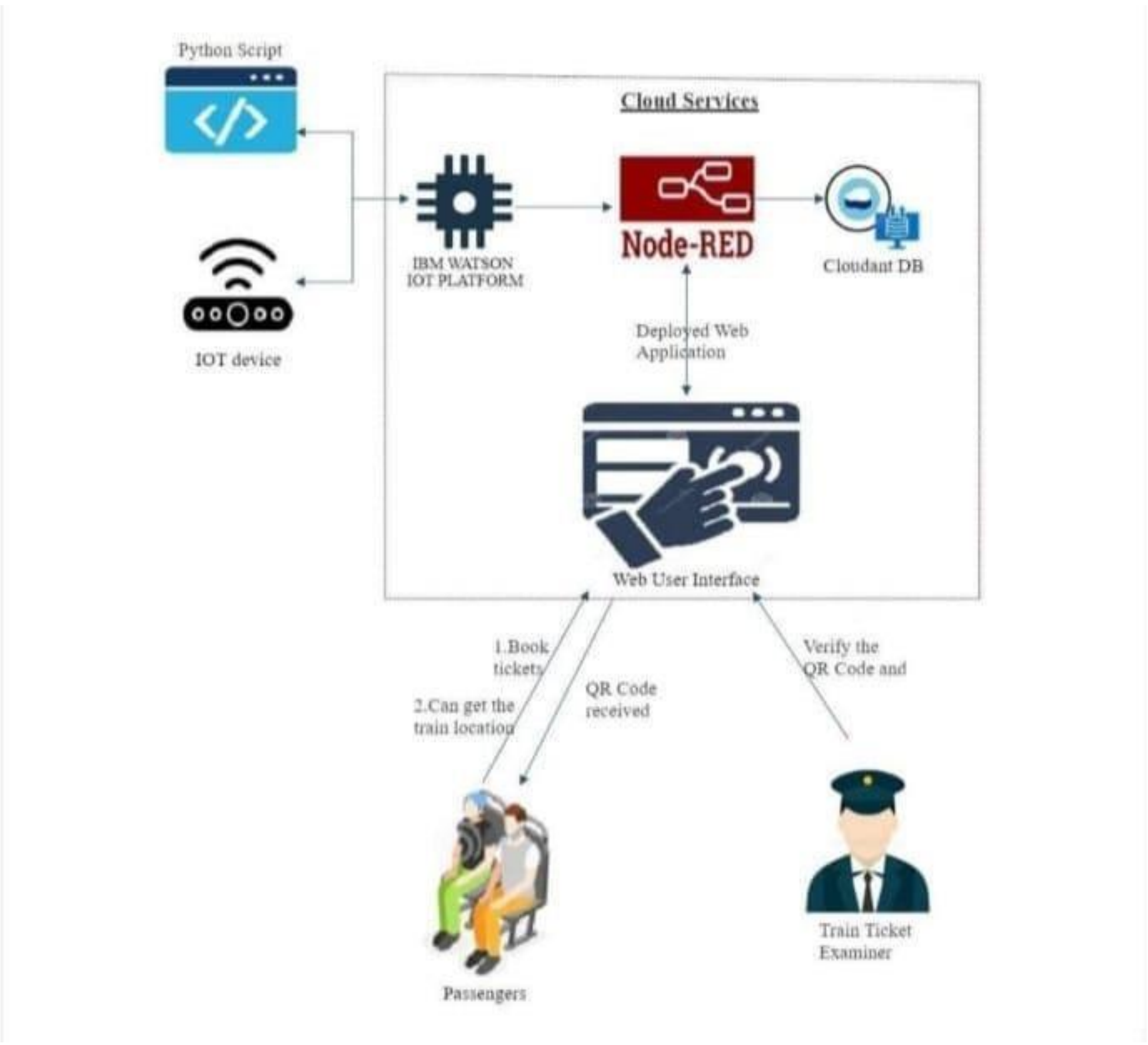
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Within periodic maintenance, we can detect cracks in the railway track. which will be highly usable on remote railway tracks.
NFR-2	**Security**	Accidents and property damage can be prevented with the help of our smart sensors which immediately send the fault to the pilot and administration.
NFR-3	**Reliability**	Traffic lights and signalling can be made accurately with the help of sensors. so it is more reliable.
NFR-4	**Performance**	Communication plays a vital role in transferring the crack-detected signal to the responsible authority so that they can take appropriate measures within a short span.
NFR-5	**Availability**	Our idea is to make the crack alert to all the trains passing through that fault-prone area.
NFR-6	**Scalability**	Our project is based on IoT & cloud, which makes the pilot and authority updated every single sec. Adhoc is easy to handle.

5.PROJECT DESIGN

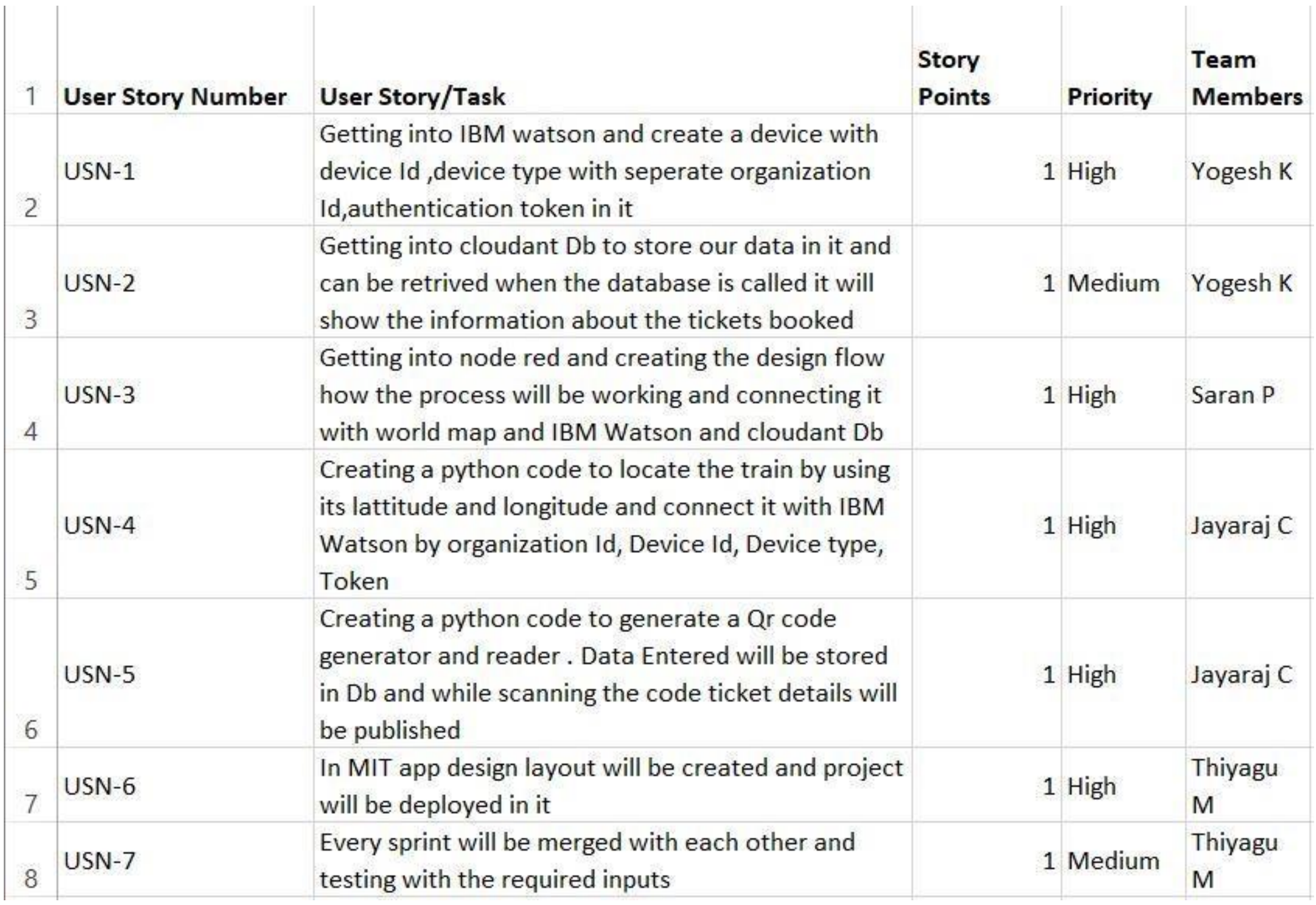
5.1 DATA FLOW DIAGRAM



5.2 SOLUTION AND TECHNICAL ARCHITECTURE

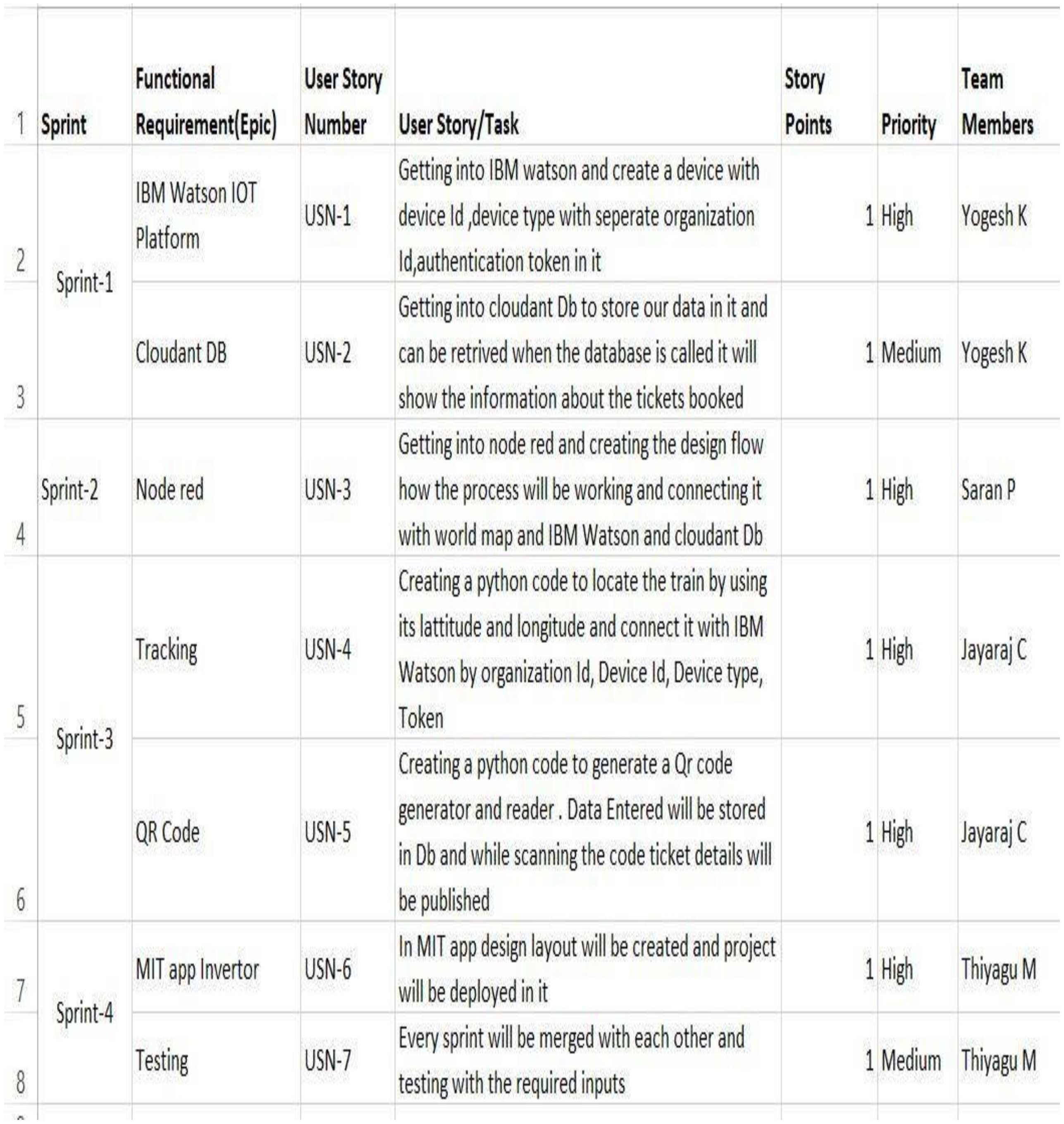

Python Script
Cloud Services
IBM WATSON IOT PLATFORM
Node-RED
Cloudant DB
IOT device
Deployed Web Application
Web User Interface
1.Book tickets
2.Can get the train location
QR Code received
Verify the QR Code and
Passengers
Train Ticket Examiner

5.3 USER STORIES



1	User Story Number	User Story/Task	Story Points	Priority	Team Members
2	USN-1	Getting into IBM watson and create a device with device Id ,device type with seperate organization Id,authentication token in it	1	High	Yogesh K
3	USN-2	Getting into cloudant Db to store our data in it and can be retrived when the database is called it will show the information about the tickets booked	1	Medium	Yogesh K
4	USN-3	Getting into node red and creating the design flow how the process will be working and connecting it with world map and IBM Watson and cloudant Db	1	High	Saran P
5	USN-4	Creating a python code to locate the train by using its lattitude and longitude and connect it with IBM Watson by organization Id, Device Id, Device type, Token	1	High	Jayaraj C
6	USN-5	Creating a python code to generate a Qr code generator and reader . Data Entered will be stored in Db and while scanning the code ticket details will be published	1	High	Jayaraj C
7	USN-6	In MIT app design layout will be created and project will be deployed in it	1	High	Thiyagu M
8	USN-7	Every sprint will be merged with each other and testing with the required inputs	1	Medium	Thiyagu M

6.PROJECT PLANNING AND SCHEDULING

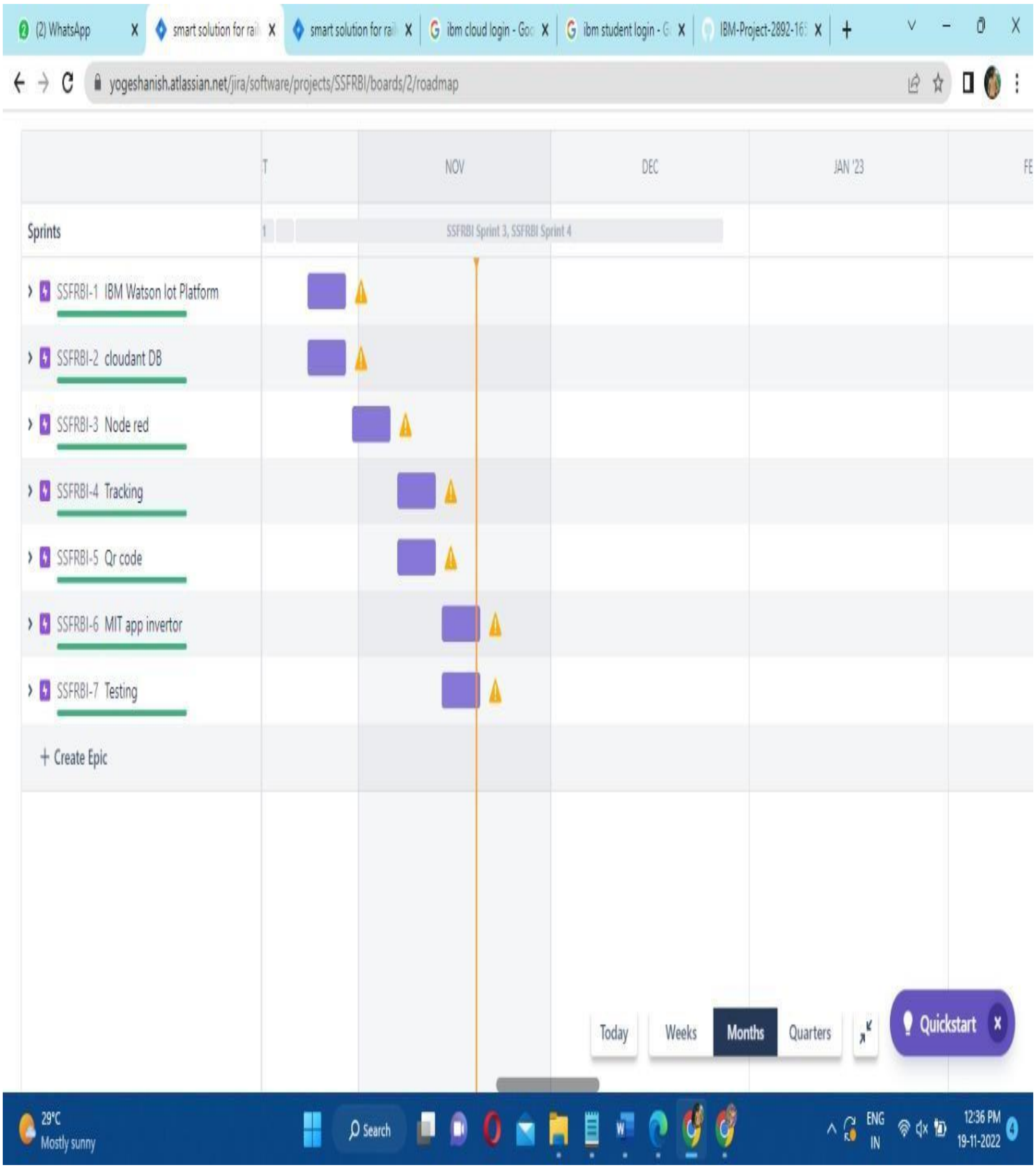


1	Sprint	Functional Requirement(Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Members
2	Sprint-1	IBM Watson IOT Platform	USN-1	Getting into IBM watson and create a device with device Id ,device type with seperate organization Id,authentication token in it	1	High	Yogesh K
3		Cloudant DB	USN-2	Getting into cloudant Db to store our data in it and can be retrived when the database is called it will show the information about the tickets booked	1	Medium	Yogesh K
4	Sprint-2	Node red	USN-3	Getting into node red and creating the design flow how the process will be working and connecting it with world map and IBM Watson and cloudant Db	1	High	Saran P
5	Sprint-3	Tracking	USN-4	Creating a python code to locate the train by using its lattitude and longitude and connect it with IBM Watson by organization Id, Device Id, Device type, Token	1	High	Jayaraj C
6		QR Code	USN-5	Creating a python code to generate a Qr code generator and reader . Data Entered will be stored in Db and while scanning the code ticket details will be published	1	High	Jayaraj C
7	Sprint-4	MIT app Invertor	USN-6	In MIT app design layout will be created and project will be deployed in it	1	High	Thiyagu M
8		Testing	USN-7	Every sprint will be merged with each other and testing with the required inputs	1	Medium	Thiyagu M

6.2 SPRINT DELIVERY SCHEDULE

Sprint	**Total Story Point s**	**Duratio n**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date(Actual)**
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	5 Nov 2022
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov2022

JIRA:-


yogeshanish.atlassian.net/jira/software/projects/SSFRBI/boards/2/roadmap
NOV
DEC
JAN '23
Sprints
SSFRBI Sprint 3, SSFRBI Sprint 4
+ Create Epic
Today
Weeks
Months
Quarters
Quickstart
29°C Mostly sunny
12:36 PM 19-11-2022

SSFRBI-1 IBM Watson Iot Platform
SSFRBI-2 cloudant DB
SSFRBI-3 Node red
SSFRBI-4 Tracking
SSFRBI-5 Qr code
SSFRBI-6 MIT app invertor
SSFRBI-7 Testing

7.CODING AND SOLUTIONING

7.1 FEATURE 1

- IoT Device
- IBM Watson platform
- Node Red
- Cloudant DB
- Web UI
- Geofence ○ MIT App
- Python Code

7.2 FEATURE 2

- Registration
- Seats
- Name
- Age
- Mobile Number
- Boarding Station
- Destination Station

IBM code:-

```
import wiotp.sdk.device import time import random
    myConfig = { "identity": { "orgId": "625xj1",
    "typeId": "GPS", "deviceId":"12345"
    },
"auth": {

"token": "wOU&i?aL*2Le008hJ&"

}

}

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, onPublish=None)
print ("Published data Successfully: %s", myData)


while True:

myData={'name': 'Train1', 'lat': 17.6387448,
'lon': 78.4754336} pub (myData) time.sleep

(3)

#myData={'name': 'Train2', '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 ()
```

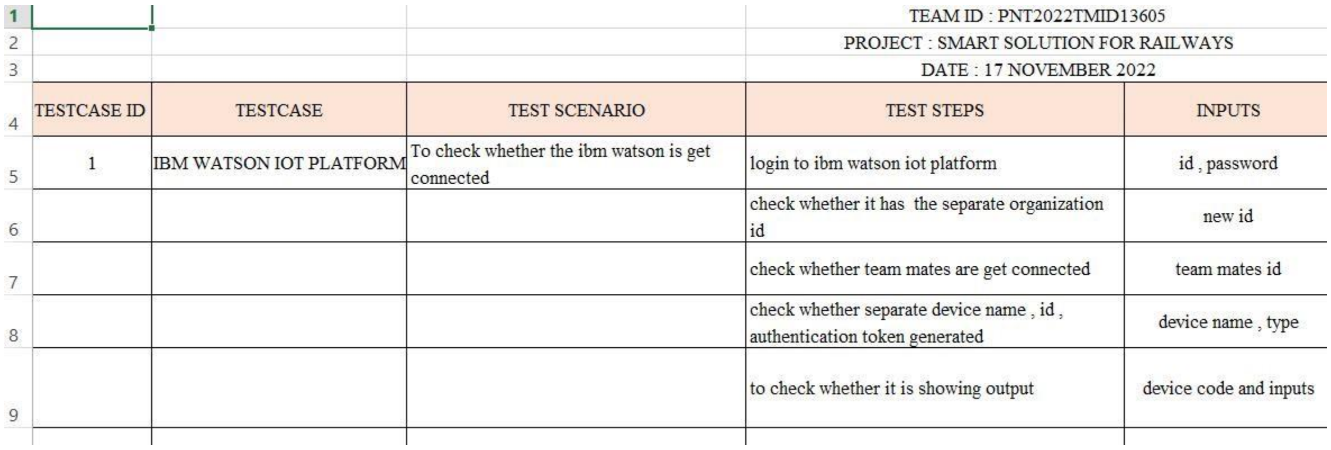
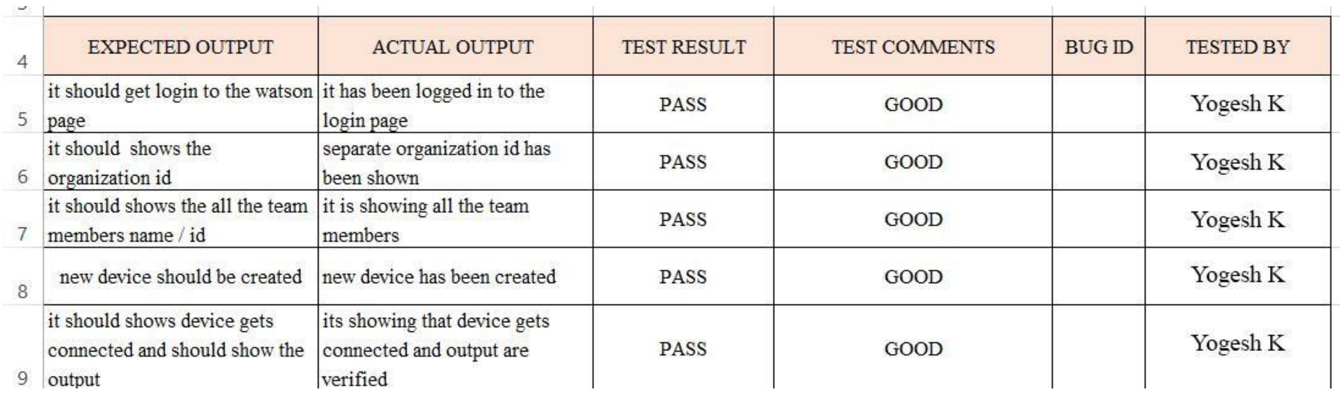
QR CODE:-

```
import cv2 import numpy as np import time import pyzbar from

ibmcloudant import cloudant_v1 from ibmcloudant import

CouchDbSessionAuthenticator from

ibm_cloud_sdk_core.authenticators import BasicAuthenticator


authenticator = BasicAuthenticator('apikey-v2-
acv8gh5fnu0u4mh2f8c5x975ae5rnphr3jxkr5d9ril','c1dd4db6e976d915751882f688e410ec') service

= cloudant_v1(authenticator=authenticator)


service.set_service_url('https://apikey-v2-
acv8gh5fnu0u4mh2f8c5x975ae5rnphr3jxkr5d9ril:c1dd4db6e976d915751882f688e410ec@adad2af9 -59c4-41bb-
b4b4-806f0d6962b2-bluemix.cloudantnosqldb.appdomain.cloud')


cap= cv2.VideoCapture(0) font =

cv2.FONT_HERSHEY_PLAIN


while True: _, frame = cap.read() decodedObjects =

pyzbar.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='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()
```

8.TESTING

8.1 TEST CASES

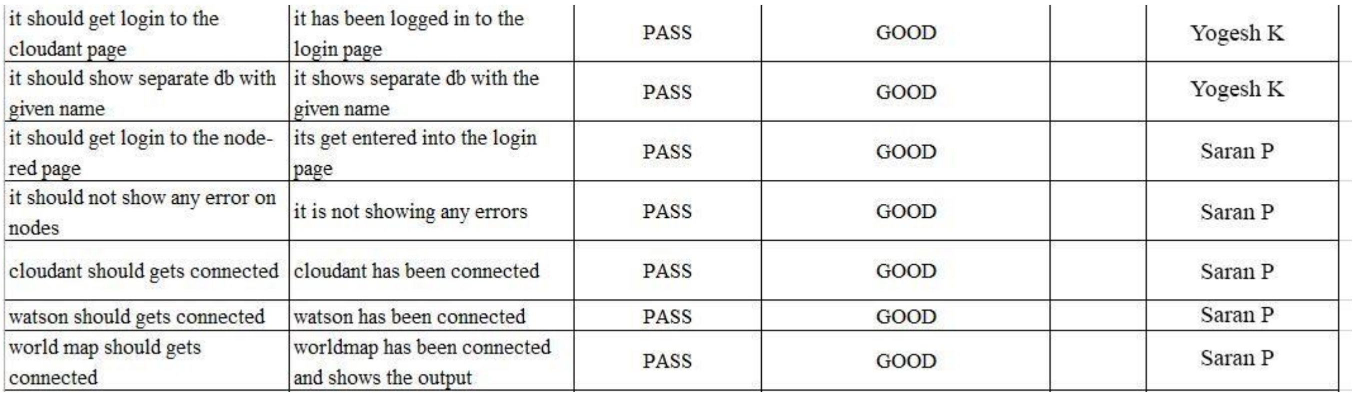



TEAM ID : PNT2022TMID13605

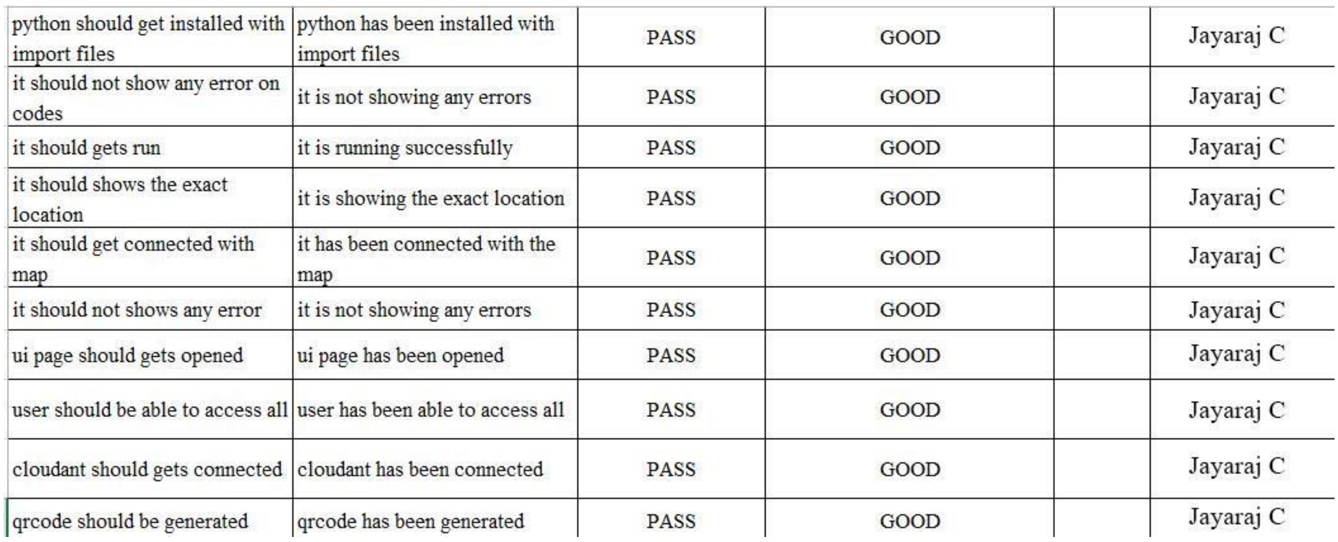
PROJECT : SMART SOLUTION FOR RAILWAYS

DATE : 17 NOVEMBER 2022

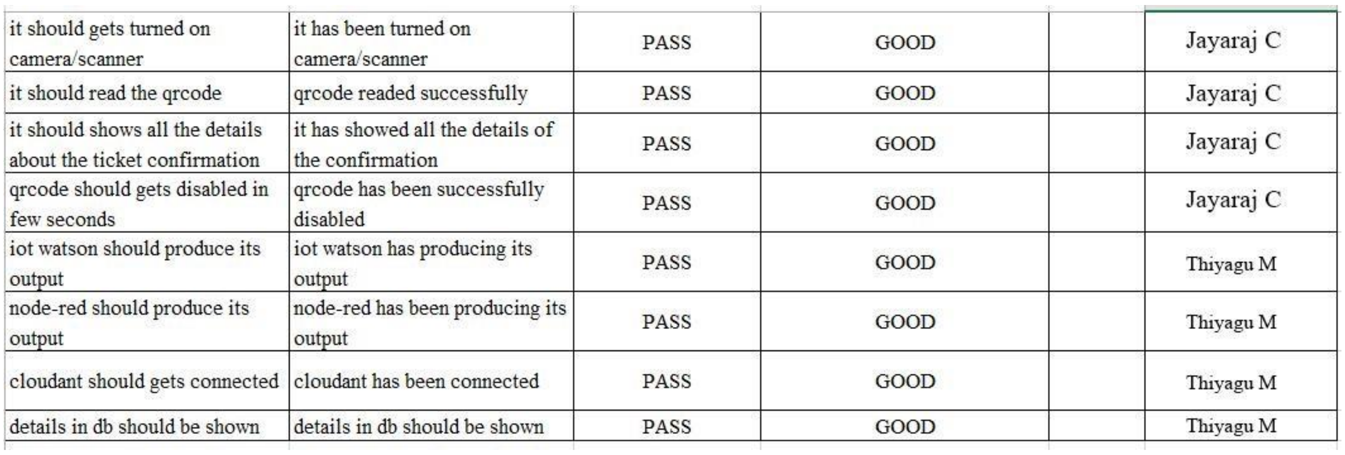
TESTCASE ID	TESTCASE	TEST SCENARIO	TEST STEPS	INPUTS	EXPECTED OUTPUT	ACTUAL OUTPUT	TEST RESULT	TEST COMMENTS	BUG ID	TESTED BY
1	IBM WATSON IOT PLATFORM	To check whether the ibm watson is get connected	login to ibm watson iot platform	id , password	it should get login to the watson page	it has been logged in to the login page	PASS	GOOD		Yogesh K
			check whether it has the separate organization id	new id	it should shows the organization id	separate organization id has been shown	PASS	GOOD		Yogesh K
			check whether team mates are get connected	team mates id	it should shows the all the team members name / id	it is showing all the team members	PASS	GOOD		Yogesh K
			check whether separate device name , id , authentication token generated	device name , type	new device should be created	new device has been created	PASS	GOOD		Yogesh K
			to check whether it is showing output	device code and inputs	it should shows device gets connected and should show the output	its showing that device gets connected and output are verified	PASS	GOOD		Yogesh K



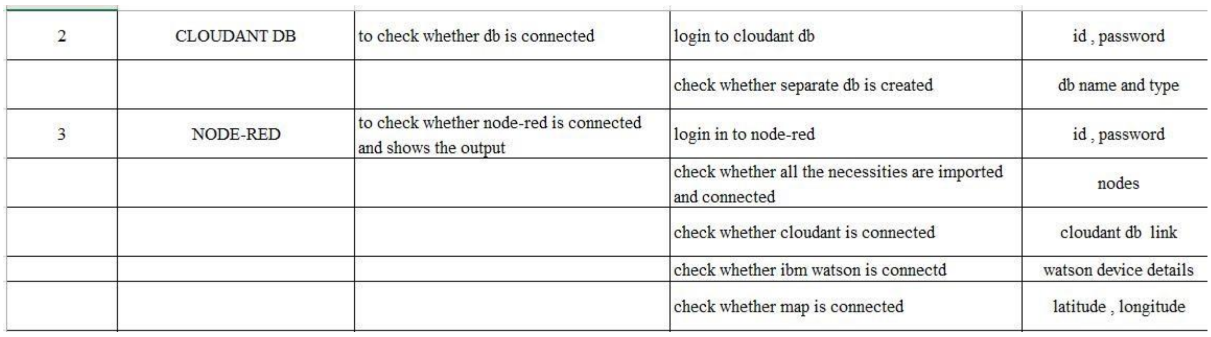
it should get login to the cloudant page	it has been logged in to the login page	PASS	GOOD		Yogesh K
it should show separate db with given name	it shows separate db with the given name	PASS	GOOD		Yogesh K
it should get login to the node-red page	its get entered into the login page	PASS	GOOD		Saran P
it should not show any error on nodes	it is not showing any errors	PASS	GOOD		Saran P
cloudant should gets connected	cloudant has been connected	PASS	GOOD		Saran P
watson should gets connected	watson has been connected	PASS	GOOD		Saran P
world map should gets connected	worldmap has been connected and shows the output	PASS	GOOD		Saran P



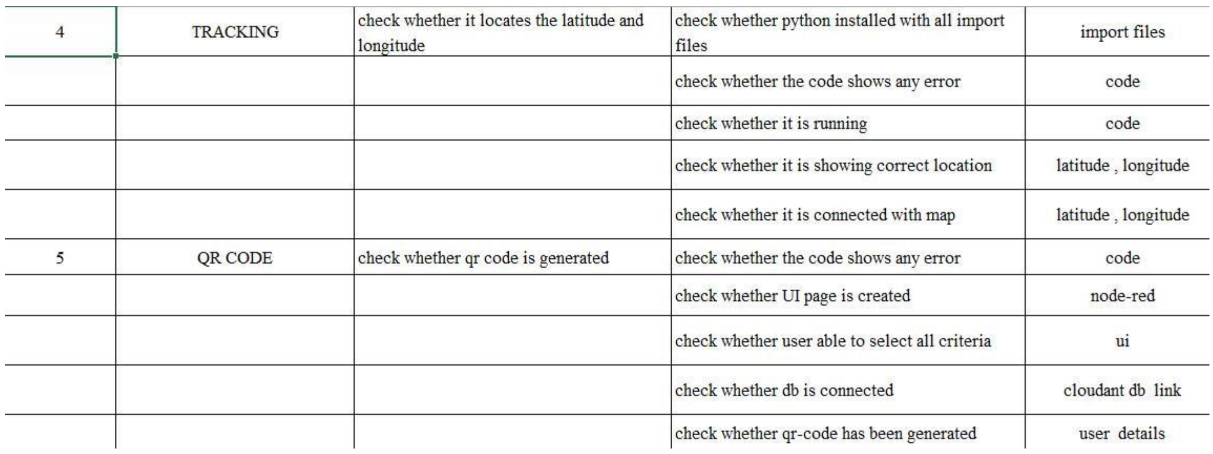
python should get installed with import files	python has been installed with import files	PASS	GOOD		Jayaraj C
it should not show any error on codes	it is not showing any errors	PASS	GOOD		Jayaraj C
it should gets run	it is running successfully	PASS	GOOD		Jayaraj C
it should shows the exact location	it is showing the exact location	PASS	GOOD		Jayaraj C
it should get connected with map	it has been connected with the map	PASS	GOOD		Jayaraj C
it should not shows any error	it is not showing any errors	PASS	GOOD		Jayaraj C
ui page should gets opened	ui page has been opened	PASS	GOOD		Jayaraj C
user should be able to access all	user has been able to access all	PASS	GOOD		Jayaraj C
cloudant should gets connected	cloudant has been connected	PASS	GOOD		Jayaraj C
qrcode should be generated	qrcode has been generated	PASS	GOOD		Jayaraj C



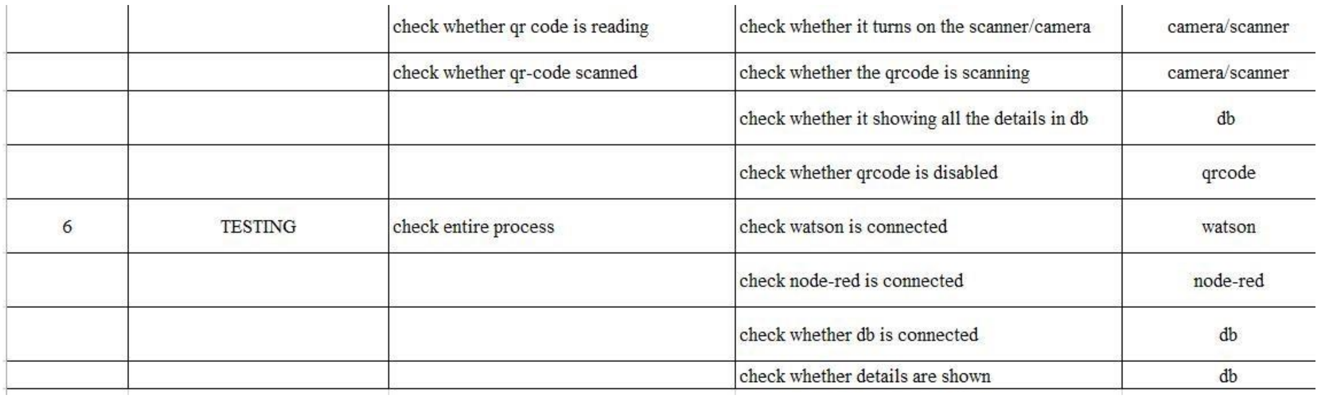
it should gets turned on camera/scanner	it has been turned on camera/scanner	PASS	GOOD		Jayaraj C
it should read the qrcode	qrcode readed successfully	PASS	GOOD		Jayaraj C
it should shows all the details about the ticket confirmation	it has showed all the details of the confirmation	PASS	GOOD		Jayaraj C
qrcode should gets disabled in few seconds	qrcode has been successfully disabled	PASS	GOOD		Jayaraj C
iot watson should produce its output	iot watson has producing its output	PASS	GOOD		Thiyagu M
node-red should produce its output	node-red has been producing its output	PASS	GOOD		Thiyagu M
cloudant should gets connected	cloudant has been connected	PASS	GOOD		Thiyagu M
details in db should be shown	details in db should be shown	PASS	GOOD		Thiyagu M



2	CLOUDANT DB	to check whether db is connected	login to cloudant db	id , password
			check whether separate db is created	db name and type
3	NODE-RED	to check whether node-red is connected and shows the output	login in to node-red	id , password
			check whether all the necessities are imported and connected	nodes
			check whether cloudant is connected	cloudant db link
			check whether ibm watson is connectd	watson device details
			check whether map is connected	latitude , longitude



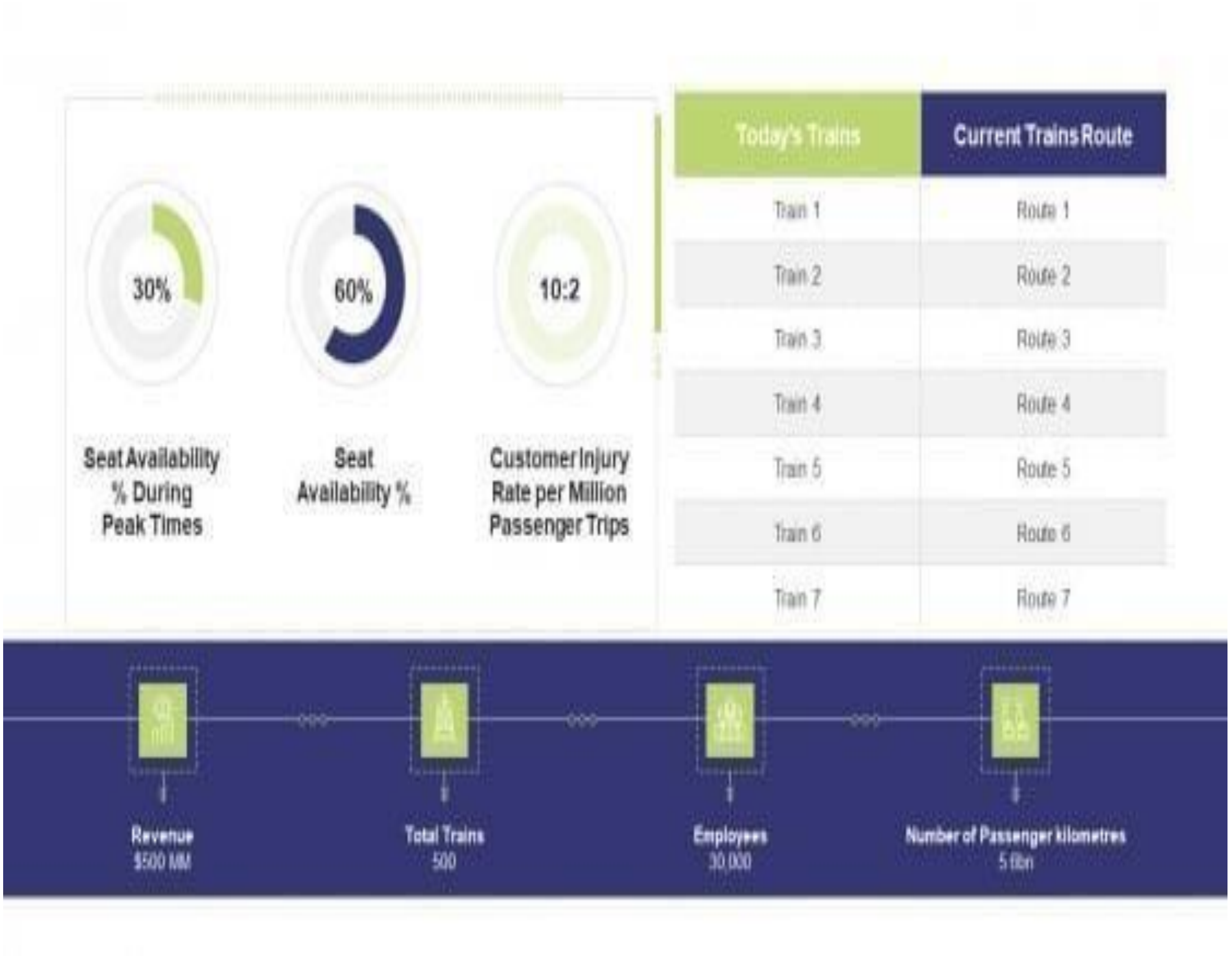
4	TRACKING	check whether it locates the latitude and longitude	check whether python installed with all import files	import files
			check whether the code shows any error	code
			check whether it is running	code
			check whether it is showing correct location	latitude , longitude
			check whether it is connected with map	latitude , longitude
5	QR CODE	check whether qr code is generated	check whether the code shows any error	code
			check whether UI page is created	node-red
			check whether user able to select all criteria	ui
			check whether db is connected	cloudant db link
			check whether qr-code has been generated	user details



		check whether qr code is reading	check whether it turns on the scanner/camera	camera/scanner
		check whether qr-code scanned	check whether the qrcode is scanning	camera/scanner
			check whether it showing all the details in db	db
			check whether qrcode is disabled	qrcode
6	TESTING	check entire process	check watson is connected	watson
			check node-red is connected	node-red
			check whether db is connected	db
			check whether details are shown	db

9.RESULTS

9.1 PERFORMANCE METRICES



Today's Trains	Current Trains Route
Train 1	Route 1
Train 2	Route 2
Train 3	Route 3
Train 4	Route 4
Train 5	Route 5
Train 6	Route 6
Train 7	Route 7

10.ADVANTAGES AND DISADVANTAGES

10.1 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.

10.2 DISADVANTAGES

- Approaches to flexible, effective, efficient, and lowcost 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.

11.CONCLUSION

Accidents involving the rail transportation system result in a large loss of life. Therefore, by alerting the railroad authorities in advance of any problems or cracks, so that they can be corrected and the number of accidents lowers, this technology helps in the prevention of accidents. This project is affordable. By utilising extra strategies, they can be strengthened and expanded in accordance with their applications. This device can save a great deal of lives by averting accidents. It is feasible to apply the idea on a broad scale over the long term in order to support increased standards for rail track safety and provide a productive testing environment for better future results.

12.FUTURE SCOPES

In the future, visual videos taken from the track can be monitored using CCTV systems with IP-based cameras. Additionally, it will make trains and people safer. In addition to using GPS to pinpoint the precise location of a track fault area, IP cameras can also be utilised to visually demonstrate a fault. With the use of sensors, locations on Google maps can be utilised to identify where a track is damaged.

13.APPENDIX

13.1 SOURCE PROGRAM

```
import math, random import os import smtplib import sqlite3 import requests
from bs4 import BeautifulSoup from django.contrib.auth.base_user import
AbstractBaseUser from django.db import models import logging import
pandas as pd import pyttsx3 from plyer import notification import time
```

```
import numpy as np import matplotlib.pyplot as plt from PIL import Image,
ImageDraw from pickle import load,dump import smtplib, ssl from
email.mime.text import MIMEText from email.mime.multipart import
MIMEMultipart import email from email import encoders from
email.mime.base import MIMEBase import attr from flask import Blueprint,
flash, redirect, request, url_for from flask.views import MethodView from
flask_babelplus import gettext as _ from flask_login import current_user,
login_required from pluggy import HookimplMarker from tkinter import*
base = Tk() base.geometry("500x500") base.title("registration form") labl_0
= Label(base, text="Registration form",width=20,font=("bold", 20))
labl_0.place(x=90,y=53)

lb1= Label(base, text="Enter Name", width=10, font=("arial",12))
lb1.place(x=20, y=120) en1= Entry(base) en1.place(x=200, y=120)
lb3= Label(base, text="Enter Email", width=10, font=("arial",12))
lb3.place(x=19, y=160) en3= Entry(base) en3.place(x=200, y=160)
lb4= Label(base, text="Contact Number", width=13,font=("arial",12))
lb4.place(x=19, y=200) en4= Entry(base) en4.place(x=200, y=200)
lb5= Label(base, text="Select Gender", width=15, font=("arial",12))
lb5.place(x=5, y=240) var = IntVar()
Radiobutton(base, text="Male", padx=5,variable=var, value=1).place(x=180,
y=240)
Radiobutton(base, text="Female", padx =10,variable=var,
value=2).place(x=240,y=240)
Radiobutton(base, text="others", padx=15, variable=var,
value=3).place(x=310,y=240) list_of_cntry = ("United States", "India",
"Nepal", "Germany") cv = StringVar() drplist= OptionMenu(base, cv,
*list_of_cntry) drplist.config(width=15) cv.set("United States") lb2=
Label(base, text="Select Country", width=13,font=("arial",12))
lb2.place(x=14,y=280) drplist.place(x=200, y=275) lb6= Label(base,
text="Enter Password", width=13,font=("arial",12)) lb6.place(x=19, y=320)
```

```
en6= Entry(base, show='*') en6.place(x=200, y=320) lb7= Label(base,
text="Re-Enter Password", width=15,font=("arial",12)) lb7.place(x=21, y=360)
en7 =Entry(base, show='*') en7.place(x=200, y=360)
Button(base, text="Register", width=10).place(x=200,y=400) base.mainloop()
def generateOTP() :
# Declare a digits variable # which stores all digits digits = "0123456789" OTP
= ""
# length of password can be changed # by changing value in range for
i in range(4) :
OTP += digits[math.floor(random.random() * 10)] return OTP
# Driver code If name== "main" : print("OTP of 4 digits:",
generateOTP()) digits="0123456789" OTP="" for i in range(6):
OTP+=digits[math.floor(random.random()*10)] otp = OTP + " is
your OTP" msg= otp s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()
s.login("Your Gmail Account", "You app password") emailid = input("Enter
your email: ") s.sendmail('&&&&&&&&&&&',emailid,msg) a =
input("Enter Your OTP >>: ")  if a == OTP: print("Verified") else:
print("Please Check your OTP again") root = Tk() root.title("Python: Simple
Login Application") width = 400 height = 280 screen_width =
root.winfo_screenwidth() screen_height = root.winfo_screenheight() x =
(screen_width/2) - (width/2) y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y)) root.resizable(0, 0)
USERNAME = StringVar() PASSWORD = StringVar()
Top = Frame(root, bd=2, relief=RIDGE) Top.pack(side=TOP, fill=X) Form
= Frame(root, height=200) Form.pack(side=TOP, pady=20)
lbl_title = Label(Top, text = "Python: Simple Login Application", font=('arial',
15))
lbl_title.pack(fill=X) lbl_username = Label(Form, text = "Username:",
font=('arial', 14), bd=15) lbl_username.grid(row=0, sticky="e")
lbl_password = Label(Form, text = "Password:", font=('arial', 14), bd=15)
```

```
lbl_password.grid(row=1, sticky="e") lbl_text = Label(Form)
lbl_text.grid(row=2, columnspan=2) username = Entry(Form,
textvariable=USERNAME, font=(14)) username.grid(row=0, column=1)
password = Entry(Form, textvariable=PASSWORD, show="*", font=(14))
password.grid(row=1, column=1) def Database():
global conn, cursor conn = sqlite3.connect("pythontut.db")
cursor = conn.cursor()
cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id
INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, username TEXT, password TEXT)")
cursor.execute("SELECT * FROM `member` WHERE `username` =
'admin' AND `password` = 'admin'") if
cursor.fetchone() is None:
cursor.execute("INSERT INTO `member` (username, password)
VALUES('admin', 'admin')") conn.commit()
def Login(event=None): Database() if USERNAME.get() == "" or
PASSWORD.get() == "": lbl_text.config(text="Please complete the
required field!", fg="red") else: cursor.execute("SELECT * FROM
`member` WHERE `username`
= ? AND `password` = ?", (USERNAME.get(), PASSWORD.get())) if
cursor.fetchone() is not None:
HomeWindow() USERNAME.set("") PASSWORD.set("")
lbl_text.config(text="")


else:
lbl_text.config(text="Invalid username or password", fg="red")
USERNAME.set("") PASSWORD.set("")
cursor.close() conn.close() btn_login = Button(Form, text="Login",
width=45, command=Login) btn_login.grid(pady=25, row=3,
columnspan=2) btn_login.bind('<Return>', Login)
```

```
def HomeWindow(): global Home root.withdraw() Home = Toplevel()
Home.title("Python: Simple Login Application") width = 600 height =
500 screen_width = root.winfo_screenwidth() screen_height =
root.winfo_screenheight() x = (screen_width/2) - (width/2) y =
(screen_height/2) - (height/2) root.resizable(0, 0)
Home.geometry("%dx%d+%d+%d" % (width, height, x, y)) lbl_home =
Label(Home, text="Successfully Login!", font=('times new roman', 20)).pack()
btn_back = Button(Home, text='Back', command=Back).pack(pady=20, fill=X)
def Back():
Home.destroy() root.deiconify() def
getdata(url):
r = requests.get(url) return r.text
# input by geek from_Station_code = "GAYA" from_Station_name = "GAYA"
To_station_code = "PNBE" To_station_name = "PATNA" # url url =
"https://www.railyatri.in/booking/trains-between-
stations?from_code="+from_Station_code+"&from_name="+from_Stat
ion_name+"+JN+&journey_date=+Wed&src=tbs&to_code=" + \
To_station_code+"&to_name="+To_station_name + \ "+JN+&user_id=-
1603228437&user_token=355740&utm_source=dwebsearch_tbs_search_
trains" # pass the url
# into getdata function htmldata = getdata(url) soup
= BeautifulSoup(htmldata, 'html.parser')
# find the Html tag # with find() # and convert into string  data_str = ""
for item in soup.find_all("div", class_="col-xs-12 TrainSearchSection"):
data_str = data_str + item.get_text() result = data_str.split("\n")
print("Train between "+from_Station_name+" and "+To_station_name)
print("")
# Display the result for item in result:
if item != "": print(item) print("\n\nTicket Booking
System\n") restart = ('Y') while restart !=
('N','NO','n','no'):
```

```
print("1.Check PNR status") print("2.Ticket Reservation")
option = int(input("\nEnter your option : ")) if option ==
1:
print("Your PNR status is t3") exit(0)"))  elif option
== 2: people = int(input("\nEnter no. of Ticket you
want :
name_l = [] age_l = [] sex_l = []
for p in range(people):
name = str(input("\nName : ")) name_l.append(name)
age = int(input("\nAge : ")) age_l.append(age) sex =
str(input("\nMale or Female : ")) sex_l.append(sex)
"))  restart = str(input("\nDid you forgot someone?
y/n: if restart in ('y','YES','yes','Yes'): restart = ('Y')
else : x = 0 print("\nTotal Ticket : ",people) for p in
range(1,people+1):
print("Ticket : ",p) print("Name :
", name_l[x]) print("Age : ",
age_l[x]) print("Sex : ",sex_l[x])
x += 1 class
User(AbstractBaseUser): """
User model. """
USERNAME_FIELD = "email"
REQUIRED_FIELDS = ["first_name", "last_name"] email =
models.EmailField( verbose_name="E-mail", unique=True
)
first_name = models.CharField( verbose_name="First name", max_length=30
)
last_name = models.CharField( verbose_name="Last name", max_length=40
)
```

```
city = models.CharField( verbose_name="City", max_length=40
)
stripe_id = models.CharField( verbose_name="Stripe ID", unique=True,
max_length=50, blank=True, null=True
)
objects = UserManager() @property def
get_full_name(self):
return f"{self.first_name} {self.last_name}"
class Meta:
verbose_name = "User" verbose_name_plural = "Users" class
Profile(models.Model): """ User's profile. """ phone_number =
models.CharField( verbose_name="Phone number", max_length=15)
date_of_birth = models.DateField( verbose_name="Date of birth"
)
postal_code = models.CharField( verbose_name="Postal code",
max_length=10, blank=True
)
address = models.CharField( verbose_name="Address", max_length=255,
blank=True
)
class Meta: abstract = True class UserProfile(Profile): """ User's
profile model. """ user = models.OneToOneField( to=User,
on_delete=models.CASCADE, related_name="profile",
)
group = models.CharField( verbose_name="Group type",
choices=GroupTypeChoices.choices(), max_length=20,
default=GroupTypeChoices.EMPLOYEE.name,
)
Def str(self): return self.user.email class Meta: # user 1 -
employer user1, _ = User.objects.get_or_create(
email="foo@bar.com", first_name="Employer",
last_name="Testowy", city="Białystok", )
user1.set_unusable_password() group_name = "employer" _profile1, _ =
UserProfile.objects.get_or_create( user=user1,
date_of_birth=datetime.now() - timedelta(days=6600),
group=GroupTypeChoices(group_name).name, address="Myśliwska 14",
postal_code="15-569", phone_number="+48100200300",
)
# user2 - employee user2, _ = User.objects.get_or_create()
email="bar@foo.com", first_name="Employee",
last_name="Testowy", city="Białystok",
)
user2.set_unusable_password() group_name = "employee" _profile2,
_ = UserProfile.objects.get_or_create() user=user2,
date_of_birth=datetime.now() - timedelta(days=7600),
group=GroupTypeChoices(group_name).name, address="Myśliwska 14",
postal_code="15-569", phone_number="+48200300400",
)
response_customer = stripe.Customer.create() email=user.email,
description=f"EMPLOYER - {user.get_full_name}",
name=user.get_full_name, phone=user.profile.phone_number,
)
user1.stripe_id = response_customer.stripe_id user1.save() mcc_code,
url = "1520", "https://www.softserveinc.com/" response_ca =
stripe.Account.create() type="custom", country="PL",
email=user2.email, default_currency="pln",
business_type="individual", settings={"payouts": {"schedule":
{"interval": "manual", }}}, requested_capabilities=["card_payments",
"transfers", ], business_profile={"mcc": mcc_code, "url": url},
individual={
```

```
"first_name": user2.first_name, "last_name": user2.last_name, "email":
user2.email,
"dob": {
"day": user2.profile.date_of_birth.day, "month":
user2.profile.date_of_birth.month, "year": user2.profile.date_of_birth.year, },
"phone": user2.profile.phone_number, "address": {
"city": user2.city,
"postal_code": user2.profile.postal_code, "country": "PL",
"line1": user2.profile.address,
},
},
)
user2.stripe_id = response_ca.stripe_id user2.save() tos_acceptance =
{"date": int(time.time()), "ip": user_ip},
stripe.Account.modify(user2.stripe_id, tos_acceptance=tos_acceptance)
passport_front = stripe.File.create( purpose="identity_document",
file=_file, # ContentFile object stripe_account=user2.stripe_id,
)
individual = { "verification": {
"document": {"front": passport_front.get("id"),}, "additional_document":
{"front": passport_front.get("id"),},
} } stripe.Account.modify(user2.stripe_id, individual=individual)
new_card_source = stripe.Customer.create_source(user1.stripe_id,
source=token)

stripe.SetupIntent.create( payment_method_types=["card"],
customer=user1.stripe_id, description="some description",
payment_method=new_card_source.id,
)
payment_method = stripe.Customer.retrieve(user1.stripe_id).default_source
payment_intent = stripe.PaymentIntent.create( amount=amount,
currency="pln", payment_method_types=["card"], capture_method="manual",
customer=user1.stripe_id, # customer payment_method=payment_method,
application_fee_amount=application_fee_amount,
transfer_data={"destination": user2.stripe_id}, # connect account
description=description, metadata=metadata,
)
payment_intent_confirm = stripe.PaymentIntent.confirm(
payment_intent.stripe_id, payment_method=payment_method
)
stripe.PaymentIntent.capture( payment_intent.id,
amount_to_capture=amount
)
stripe.Balance.retrieve(stripe_account=user2.stripe_id) stripe.Charge.create(
amount=amount, currency="pln", source=user2.stripe_id,
description=description
)
stripe.PaymentIntent.cancel(payment_intent.id) unique_together =
("user", "group") @attr.s(frozen=True, cmp=False, hash=False,
repr=True) class UserSettings(MethodView):
form = attr.ib(factory=settings_form_factory) settings_update_handler =
attr.ib(factory=settings_update_handler) decorators = [login_required]
def get(self):
return self.render() def
post(self):
if self.form.validate_on_submit(): try:
self.settings_update_handler.apply_changeset( current_user,
self.form.as_change()
)
except StopValidation as e:

self.form.populate_errors(e.reasons) return self.render() except
PersistenceError:
```

```
logger.exception("Error while updating user settings") flash(_("Error while
updating user settings"), "danger") return self.redirect() flash(_("Settings
updated."), "success") return self.redirect() return self.render() def
render(self):
return render_template("user/general_settings.html", form=self.form) def
redirect(self):
return redirect(url_for("user.settings"))
@attr.s(frozen=True, hash=False, cmp=False, repr=True) class
ChangePassword(MethodView):
form = attr.ib(factory=change_password_form_factory)
password_update_handler =
attr.ib(factory=password_update_handler) decorators =
[login_required] def get(self):
return self.render() def
post(self):
if self.form.validate_on_submit(): try:
self.password_update_handler.apply_changeset( current_user,
self.form.as_change()
)
except StopValidation as e: self.form.populate_errors(e.reasons) return
self.render() except PersistenceError:
logger.exception("Error while changing password") flash(_("Error while
changing password"), "danger") return self.redirect() flash(_("Password
updated."), "success") return self.redirect() return self.render()


def render(self):
return render_template("user/change_password.html", form=self.form) def
redirect(self):
return redirect(url_for("user.change_password"))
@attr.s(frozen=True, cmp=False, hash=False, repr=True) class
ChangeEmail(MethodView):
```

```
form = attr.ib(factory=change_email_form_factory) update_email_handler =
attr.ib(factory=email_update_handler) decorators = [login_required] def
get(self):
return self.render() def
post(self):
if self.form.validate_on_submit(): try:
self.update_email_handler.apply_changeset( current_user,
self.form.as_change()
)
except StopValidation as e: self.form.populate_errors(e.reasons) return
self.render() except PersistenceError:
logger.exception("Error while updating email") flash(_("Error while updating
email"), "danger") return self.redirect() flash(_("Email address updated."),
"success") return self.redirect() return self.render()
def render(self):
return render_template("user/change_email.html", form=self.form) def
redirect(self):
return redirect(url_for("user.change_email")) def berth_type(s): if
s>0 and s<73:
if s % 8 == 1 or s % 8 == 4: print (s), "is lower berth" elif s % 8 == 2 or s % 8
== 5: print (s), "is middle berth" elif s % 8 == 3 or s % 8 == 6: print (s), "is
upper berth" elif s % 8 == 7: print (s), "is side lower berth" else:
print (s), "is side upper berth" else:
print (s), "invalid seat number"
# Driver code s = 10
berth_type(s) s             # fxn call for berth type
= 7
berth_type(s) s             # fxn call for berth type
= 0
```

```
berth_type(s) counter=0  # fxn call for berth type class Ticket:


def                    init    (self,passenger_name,source,destination): self.
                       passenger_name=passenger_name
self.                  source=source
self.                  destination=destination self.Counter=Ticket.counter
Ticket.counter+=1


def validate_source_destination(self): if (self.
 source=="Delhi" and (self.    destination=="Pune" or
self.   destination=="Mumbai" or self.
 destination=="Chennai" or self.
 destination=="Kolkata")): return True else:
return False def
generate_ticket(self ): if True:


 ticket_id=self.__source[0]+self.__destination[0]+"0"+str(self.Counter) print(
"Ticket id will be:",      ticket_id)
else:
return False def get_ticket_id(self): return self.ticket_id def
get_passenger_name(self): return self.__passenger_name def
get_source(self):
if self.source=="Delhi": return self.source else:
print("you have written invalid soure option") return None def
get_destination(self):
if self.destination=="Pune": return self.     destination
elif self.destination=="Mumbai": return self. destination


elif self.destination=="Chennai": return self. destination elif
self.destination=="Kolkata": return self. destination else:
return None
```

```
# user define function # Scrape the data
def getdata(url): r = requests.get(url)
return r.text
# input by geek train_name = "03391-rajgir-new-delhi-clone-
special-rgd-to-ndls" # url url = "https://www.railyatri.in/live-
train-status/"+train_name
# pass the url
# into getdata function htmldata = getdata(url) soup
= BeautifulSoup(htmldata, 'html.parser')

# traverse the live status from # this Html code data = [] for
item in soup.find_all('script', type="application/ld+json"):
data.append(item.get_text())
# convert into dataframe df = pd.read_json(data[2])
# display this column of # dataframe print(df["mainEntity"][0]['name'])
print(df["mainEntity"][0]['acceptedAnswer']['text'])
Speak method def Speak(self, audio):
# Calling the initial constructor # of pyttsx3
engine = pyttsx3.init('sapi5') # Calling the
getter method voices =
engine.getProperty('voices')
# Calling the setter method engine.setProperty('voice', voices[1].id)
engine.say(audio) engine.runAndWait() def Take_break():
Speak("Do you want to start sir?") question = input()
if "yes" in question:  Speak("Starting Sir") Sir.")
mins",

as you have" affect your eyes",  if
"no" in question:
```

Speak("We will automatically start after 5 Mins time.sleep(5*60)

Speak("Starting Sir")

# A notification we will held that

# Let's Start sir and with a message of # will tell you to take a break after 45 # mins for 10 seconds while(True): notification.notify(title="Let's Start sir", message="will tell you to take a break after 45 timeout=10)

# For 45 min the will be no notification but # after 45 min a notification will pop up. time.sleep(0.5*60) Speak("Please Take a break Sir") notification.notify(title="Break Notification", message="Please do use your device after sometime

"been continuously using it for 45 mins and it will timeout=10)

# Driver's Code

If name== '__main ':

Take_break() data_path = 'data.csv' data = pd.read_csv(data_path,

names=['LATITUDE', 'LONGITUDE'], sep=',') gps_data =

tuple(zip(data['LATITUDE'].values, data['LONGITUDE'].values)) image =

Image.open('map.png', 'r') # Load map image. img_points = [] for d in

gps_data:

x1, y1 = scale_to_img(d, (image.size[0], image.size[1])) # Convert GPS coordinates to image coordinates.

img_points.append((x1, y1)) draw = ImageDraw.Draw(image) draw.line(img_points, fill=(255, 0, 0), width=2) # Draw converted records to the map image.

image.save('resultMap.png') x_ticks = map(lambda x: round(x, 4), np.linspace(lon1, lon2, num=7)) y_ticks = map(lambda x: round(x, 4), np.linspace(lat1, lat2, num=8))

y_ticks = sorted(y_ticks, reverse=True) # y ticks must be reversed due to conversion to image coordinates.

fig, axis1 = plt.subplots(figsize=(10, 10))

axis1.imshow(plt.imread('resultMap.png')) # Load the image to matplotlib plot.

```
axis1.set_xlabel('Longitude')  axis1.set_ylabel('Latitude')
axis1.set_xticklabels(x_ticks)
axis1.set_yticklabels(y_ticks) axis1.grid() plt.show()
class tickets:
def init(self): self.no_ofac1stclass=0 self.totaf=0 self.no_ofac2ndclass=0
self.no_ofac3rdclass=0 self.no_ofsleeper=0 self.no_oftickets=0 self.name="
self.age=" self.resno=0 self.status=" def ret(self): return(self.resno) def
retname(self): return(self.name) def display(self): f=0
fin1=open("tickets.dat","rb") if not fin1: print "ERROR" else: print
n=int(raw_input("ENTER PNR NUMBER : ")) print "\n\n"
print ("FETCHING DATA . . .".center(80)) time.sleep(1)
print print('PLEASE WAIT...!!'.center(80)) time.sleep(1)
os.system('cls') try: while True: tick=load(fin1)
if(n==tick.ret()):
f=1 print "="*80 print("PNR
STATUS".center(80)) print"="*80 print
print "PASSENGER'S NAME :",tick.name print print
"PASSENGER'S AGE :",tick.age print print "PNR
NO :",tick.resno print print "STATUS :",tick.status
print print "NO OF SEATS BOOKED :
",tick.no_oftickets print except: pass fin1.close()
if(f==0):
print


print "WRONG PNR NUMBER..!!"
print
def pending(self): self.status="WAITING LIST" print "PNR NUMBER
:",self.resno print time.sleep(1.2) print "STATUS =
",self.status print print "NO OF SEATS BOOKED :
",self.no_oftickets print def confirmation (self):
```

```
self.status="CONFIRMED" print "PNR NUMBER :
",self.resno print time.sleep(1.5) print "STATUS =
",self.status print def cancellation(self): z=0
f=0 fin=open("tickets.dat","rb") fout=open("temp.dat","ab") print
r= int(raw_input("ENTER PNR NUMBER : ")) try: while(True):
tick=load(fin) z=tick.ret() if(z!=r):
dump(tick,fout) elif(z==r):
f=1 except:
pass fin.close() fout.close() os.remove("tickets.dat")
os.rename("temp.dat","tickets.dat") if (f==0):
print
print "NO SUCH RESERVATION NUMBER FOUND"
print time.sleep(2) os.system('cls')
else: print
print "TICKET CANCELLED" print"RS.600 REFUNDED        "
def reservation(self):
trainno=int(raw_input("ENTER THE TRAIN NO:")) z=0
f=0 fin2=open("tr1details.dat") fin2.seek(0) if not fin2:
print "ERROR" else: try: while True:
n=int(raw_input("ENTER PNR NUMBER : ")) print "\n\n"
print ("FETCHING DATA . . .".center(80)) time.sleep(1)
print print('PLEASE WAIT...!!'.center(80)) time.sleep(1)
os.system('cls') try: while True: tick=load(fin1)
if(n==tick.ret()):
f=1 print "="*80 print("PNR
STATUS".center(80)) print"="*80 print
 print "PASSENGER'S NAME :",tick.name print print
"PASSENGER'S AGE :",tick.age print print "PNR NO
:",tick.resno print print "STATUS :",tick.status print print
```

```
"NO OF SEATS BOOKED : ",tick.no_oftickets print
except: pass fin1.close() if(f==0):
print
print "WRONG PNR NUMBER..!!"
print
def pending(self): self.status="WAITING LIST" print "PNR NUMBER
:",self.resno print time.sleep(1.2) print "STATUS = ",self.status
print print "NO OF SEATS BOOKED : ",self.no_oftickets print
def confirmation (self): self.status="CONFIRMED" print "PNR
NUMBER : ",self.resno print time.sleep(1.5) print "STATUS =
",self.status print def cancellation(self): z=0 f=0
fin=open("tickets.dat","rb") fout=open("temp.dat","ab") print r=
int(raw_input("ENTER PNR NUMBER : ")) try: while(True):
tick=load(fin) z=tick.ret() if(z!=r): dump(tick,fout) elif(z==r):
f=1 except:
pass fin.close() fout.close() os.remove("tickets.dat")
os.rename("temp.dat","tickets.dat") if (f==0):
print
print "NO SUCH RESERVATION NUMBER FOUND"
print time.sleep(2) os.system('cls')
else: print
print "TICKET CANCELLED" print"RS.600 REFUNDED        "
def reservation(self):
trainno=int(raw_input("ENTER THE TRAIN NO:")) z=0
f=0 fin2=open("tr1details.dat") fin2.seek(0)
if not fin2: print "ERROR" else: try:
while True: tr=load(fin2) z=tr.gettrainno() n=tr.gettrainname() if (trainno==z):
print print "TRAIN NAME IS : ",n f=1 print
print "-"*80 no_ofac1st=tr.getno_ofac1stclass()
```

```
no_ofac2nd=tr.getno_ofac2ndclass() no_ofac3rd=tr.getno_ofac3rdclass()
no_ofsleeper=tr.getno_ofsleeper()                              if(f==1):
fout1=open("tickets.dat","ab")
NAME ")
print
self.name=raw_input("ENTER THE PASSENGER'S
print self.age=int(raw_input("PASSENGER'S AGE : "))
print print"\t\t SELECT A CLASS YOU WOULD
LIKE TO TRAVEL IN :- " print "1.AC FIRST CLASS"
print
print "2.AC SECOND CLASS"
print
print "3.AC THIRD CLASS"
print
print "4.SLEEPER CLASS"
print c=int(raw_input("\t\t\tENTER YOUR CHOICE = "))
os.system('cls') amt1=0 if(c==1):

self.no_oftickets=int(raw_input("ENTER NO_OF FIRST CLASS AC SEATS
TO BE BOOKED : "))
i=1 while(i<=self.no_oftickets):
self.totaf=self.totaf+1 amt1=1000*self.no_oftickets i=i+1
print print "PROCESSING. .", time.sleep(0.5)  print ".",
time.sleep(0.3) print'.' time.sleep(2) os.system('cls')
print "TOTAL AMOUNT TO BE PAID = ",amt1
self.resno=int(random.randint(1000,2546)) x=no_ofac1st-self.totaf print
if(x>0):
self.confirmation() dump(self,fout1) break else:
self.pending() dump(tick,fout1) break
```

```
elif(c==2): self.no_oftickets=int(raw_input("ENTER NO_OF
SECOND CLASS AC SEATS TO BE BOOKED : ")) i=1
def menu(): tr=train() tick=tickets() print
print "WELCOME TO PRAHIT AGENCY".center(80)
while True:  print print "="*80 print " \t\t\t\t
RAILWAY" print print "="*80 print print "\t\t\t1.
**UPDATE TRAIN DETAILS." print print "\t\t\t2.
TRAIN DETAILS. " print print "\t\t\t3.
RESERVATION OF TICKETS." print print "\t\t\t4.
CANCELLATION OF TICKETS. " print
print "\t\t\t5. DISPLAY PNR STATUS." print print "\t\t\t6. QUIT." print"** -
office use     " ch=int(raw_input("\t\t\tENTER YOUR CHOICE : "))
os.system('cls') print
"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\tLOADI NG. .",
time.sleep(1) print ("."), time.sleep(0.5) print (".") time.sleep(2) os.system('cls')
if ch==1:
j="*****" r=raw_input("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\tENTER THE
PASSWORD: ")
os.system('cls') if (j==r):
x='y' while (x.lower()=='y'): fout=open("tr1details.dat","ab") tr.getinput()
dump(tr,fout) fout.close() print"\n\n\n\n\n\n\n\n\n\n\n\t\t\tUPDATING TRAIN
LIST PLEASE WAIT . .", time.sleep(1) print ("."), time.sleep(0.5) print ("."),
time.sleep(2) os.system('cls') print "\n\n\n\n\n\n\n\n\n\n\n"
x=raw_input("\t\tDO YOU WANT TO ADD ANY MORE TRAINS DETAILS
? ") os.system('cls') continue elif(j<>r):
print"\n\n\n\n\n" print "WRONG
PASSWORD".center(80) elif ch==2:
fin=open("tr1details.dat",'rb') if not fin:
print "ERROR" tick.display() elif
ch==6: quit()
```

```
raw_input("PRESS ENTER TO GO TO BACK MENU".center(80))
os.system('cls') menu() sender_email = "my@gmail.com"
receiver_email = "your@gmail.com" password = input("Type your
password and press enter:") message = MIMEMultipart("alternative")
message["Subject"] = "multipart test" message["From"] = sender_email
message["To"] = receiver_email # Create the plain-text and HTML
version of your message text = """\ Hi,
How are you?
Real Python has many great tutorials: www.realpython.com""" html
= """\
<html>
<body>
<p>Hi,<br>
How are you?<br>
<a href="http://www.realpython.com">Real Python</a> has many great
tutorials.
</p>
</body>
</html> """
# Turn these into plain/html MIMEText objects part1 = MIMEText(text,
"plain") part2 =
MIMEText(html, "html")
# Add HTML/plain-text parts to MIMEMultipart message # The email client
will try to render the last part first message.attach(part1) message.attach(part2)
# Create secure connection with server and send email context =
ssl.create_default_context() with smtplib.SMTP_SSL("smtp.gmail.com",
465, context=context) as server:
server.login(sender_email, password) server.sendmail( sender_email,
receiver_email, message.as_string()
)
```

```
subject = "An email with attachment from Python"
body = "This is an email with attachment sent from Python" sender_email =
"my@gmail.com" receiver_email = "your@gmail.com"
password = input("Type your password and press enter:")
# Create a multipart message and set headers message = MIMEMultipart()
message["From"] = sender_email message["To"] = receiver_email
message["Subject"] = subject message["Bcc"] = receiver_email #
Recommended for mass emails # Add body to email
message.attach(MIMEText(body, "plain")) filename = "document.pdf" # In
same directory as script # Open PDF file in binary mode with
open(filename, "rb") as attachment:
# Add file as application/octet-stream
# Email client can usually download this automatically as attachment part =
MIMEBase("application", "octet-stream") part.set_payload(attachment.read())
# Encode file in ASCII characters to send by email
encoders.encode_base64(part)
# Add header as key/value pair to attachment part part.add_header(
"Content-Disposition", f"attachment; filename= {filename}",
)
# Add attachment to message and convert message to string
message.attach(part) text = message.as_string()
# Log in to server using secure context and send email context =
ssl.create_default_context() with smtplib.SMTP_SSL("smtp.gmail.com",
465, context=context) as server:
server.login(sender_email, password) server.sendmail(sender_email,
receiver_email, text) api_key = "Your_API_key" # base_url
variable to store url base_url = "https://api.railwayapi.com/v2/pnr-
status/pnr/"
# Enter valid pnr_number pnr_number = "6515483790"
# Stores complete url address complete_url = base_url +
pnr_number + "/apikey/" + api_key + "/"
```

```
# get method of requests module # return response object response_ob
= requests.get(complete_url)
# json method of response object convert
# json format data into python format data result = response_ob.json()
# now result contains list # of nested dictionaries if
result["response_code"] == 200:
# train name is extracting
# from the result variable data train_name = result["train"]["name"] #
train number is extracting from # the result variable data
train_number = result["train"]["number"]
# from station name is extracting # from the result variable data from_station
= result["from_station"]["name"]
# to_station name is extracting from # the result variable data to_station
= result["to_station"]["name"]
# boarding point station name is
# extracting from the result variable data boarding_point =
result["boarding_point"]["name"] # reservation upto
station name is
# extracting from the result variable data reservation_upto =
result["reservation_upto"]["name"]
# store the value or data of "pnr" # key in pnr_num variable pnr_num =
result["pnr"]
# store the value or data of "doj" key # in variable date_of_journey variable
date_of_journey = result["doj"]
# store the value or data of
# "total_passengers" key in variable total_passengers = reult["total_passengers"] # store the
value or data of "passengers" # key in variable passengers_list passengers_list =
result["passengers"]
# store the value or data of
# "chart_prepared" key in variable chart_prepared = result["chart_prepared"]
# print following values print(" train
name : " + str(train_name) + "\n train
number : " + str(train_number)
+ "\n from station : " + str(from_station)
+ "\n to station : " + str(to_station)
+ "\n boarding point : " + str(boarding_point)
+ "\n reservation upto : " + str(reservation_upto)
+ "\n pnr number : " + str(pnr_num)
+ "\n date of journey : " + str(date_of_journey)
+ "\n total no. of passengers: " + str(total_passengers)
+ "\n chart prepared : " + str(chart_prepared))
# looping through passenger list for
passenger in passengers_list:
# store the value or data # of "no" key in variable passenger_num
= passenger["no"]
# store the value or data of
# "current_status" key in variable current_status = passenger["current_status"]
# store the value or data of
# "booking_status" key in variable booking_status =
passenger["booking_status"] # print following
values print(" passenger number : " +
str(passenger_num) + "\n current status : " +
str(current_status) + "\n booking_status : " +
str(booking_status)) else:

print("Record Not Found") NAME
")
tr=load(fin2) z=tr.gettrainno() n=tr.gettrainname() if (trainno==z):
print print "TRAIN NAME IS : ",n f=1 print
print "-"*80 no_ofac1st=tr.getno_ofac1stclass()
```

```
no_ofac2nd=tr.getno_ofac2ndclass() no_ofac3rd=tr.getno_ofac3rdclass()
no_ofsleeper=tr.getno_ofsleeper()                                        if(f==1):
fout1=open("tickets.dat","ab") print
self.name=raw_input("ENTER THE PASSENGER'S
print self.age=int(raw_input("PASSENGER'S AGE : "))
print print"\t\t SELECT A CLASS YOU WOULD
LIKE TO TRAVEL IN :- " print "1.AC FIRST CLASS"
print
print "2.AC SECOND CLASS"
print
print "3.AC THIRD CLASS"
print
print "4.SLEEPER CLASS"
print c=int(raw_input("\t\t\tENTER YOUR CHOICE = "))
os.system('cls') amt1=0 if(c==1):
self.no_oftickets=int(raw_input("ENTER NO_OF FIRST CLASS AC SEATS
TO BE BOOKED : "))
i=1 while(i<=self.no_oftickets):
self.totaf=self.totaf+1 amt1=1000*self.no_oftickets i=i+1
print print
"PROCESSING. .",
time.sleep(0.5) print ".", time.sleep(0.3) print'.' time.sleep(2) os.system('cls')
print "TOTAL AMOUNT TO BE PAID = ",amt1
self.resno=int(random.randint(1000,2546)) x=no_ofac1st-self.totaf print
if(x>0):
self.confirmation() dump(self,fout1) break else:
self.pending() dump(tick,fout1)
break
```

```
elif(c==2): self.no_oftickets=int(raw_input("ENTER NO_OF
SECOND CLASS AC SEATS TO BE BOOKED : ")) i=1
def menu(): tr=train() tick=tickets() print
print "WELCOME TO PRAHIT AGENCY".center(80)
while True: print print "="*80 print " \t\t\t\t
RAILWAY" print print "="*80 print print "\t\t\t1.
**UPDATE TRAIN DETAILS." print print "\t\t\t2.
TRAIN DETAILS. " print print "\t\t\t3.
RESERVATION OF TICKETS." print print "\t\t\t4.
CANCELLATION OF TICKETS. " print print "\t\t\t5.
DISPLAY PNR STATUS."  print print "\t\t\t6. QUIT."
print"** - office use          "
ch=int(raw_input("\t\t\tENTER YOUR CHOICE : "))
os.system('cls') print
"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t
\t\t\t\t\t\tLOADI NG. .", time.sleep(1) print ("."),
time.sleep(0.5) print (".") time.sleep(2) os.system('cls')
if ch==1:
j="*****" r=raw_input("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\tENTER THE
PASSWORD: ")
os.system('cls') if (j==r):
x='y' while (x.lower()=='y'): fout=open("tr1details.dat","ab") tr.getinput()
dump(tr,fout) fout.close() print"\n\n\n\n\n\n\n\n\n\n\n\t\t\tUPDATING TRAIN
LIST PLEASE WAIT . .", time.sleep(1) print ("."), time.sleep(0.5) print ("."),
time.sleep(2) os.system('cls') print "\n\n\n\n\n\n\n\n\n\n\n"
x=raw_input("\t\tDO YOU WANT TO ADD ANY MORE TRAINS DETAILS
? ") os.system('cls') continue elif(j<>r):
print"\n\n\n\n\n" print "WRONG
```

```
PASSWORD".center(80) elif ch==2:
fin=open("tr1details.dat",'rb') if not fin:
print "ERROR" tick.display() elif
ch==6: quit()
raw_input("PRESS ENTER TO GO TO BACK MENU".center(80))
os.system('cls') menu() sender_email = "my@gmail.com"
receiver_email = "your@gmail.com" password = input("Type your
password and press enter:")

message = MIMEMultipart("alternative") message["Subject"] = "multipart test"
message["From"] = sender_email message["To"] = receiver_email # Create the
plain-text and HTML version of your message text = """\ Hi,
How are you?
Real Python has many great tutorials: www.realpython.com""" html
= """\
<html>
<body>
<p>Hi,<br>
How are you?<br>
<a href="http://www.realpython.com">Real Python</a> has many great
tutorials.
</p>
</body>
</html> """
# Turn these into plain/html MIMEText objects part1 = MIMEText(text,
"plain") part2 =
MIMEText(html, "html")
# Add HTML/plain-text parts to MIMEMultipart message # The email client
will try to render the last part first message.attach(part1) message.attach(part2)
```

# Create secure connection with server and send email context = ssl.create_default_context() with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:

server.login(sender_email, password) server.sendmail(sender_email,

receiver_email, message.as_string()

)

subject = "An email with attachment from Python"

body = "This is an email with attachment sent from Python" sender_email = "my@gmail.com" receiver_email = "your@gmail.com"

password = input("Type your password and press enter:")

# Create a multipart message and set headers message = MIMEMultipart() message["From"] = sender_email message["To"] = receiver_email message["Subject"] = subject message["Bcc"] = receiver_email # Recommended for mass emails # Add body to email message.attach(MIMEText(body, "plain")) filename = "document.pdf" # In same directory as script # Open PDF file in binary mode with open(filename, "rb") as attachment:

# Add file as application/octet-stream

# Email client can usually download this automatically as attachment part = MIMEBase("application", "octet-stream") part.set_payload(attachment.read())

# Encode file in ASCII characters to send by email encoders.encode_base64(part)

# Add header as key/value pair to attachment part part.add_header(

"Content-Disposition", f"attachment; filename= {filename}",

)

# Add attachment to message and convert message to string message.attach(part) text = message.as_string()

# Log in to server using secure context and send email context = ssl.create_default_context() with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:

server.login(sender_email, password) server.sendmail(sender_email, receiver_email, text) api_key = "Your_API_key" # base_url variable to store url base_url = "https://api.railwayapi.com/v2/pnr-status/pnr/"

# Enter valid pnr_number pnr_number = "6515483790"

# Stores complete url address complete_url = base_url +

pnr_number + "/apikey/" + api_key + "/" # get method of requests

module # return response object response_ob =

requests.get(complete_url)

# json method of response object convert

# json format data into python format data result = response_ob.json()

# now result contains list # of nested dictionaries if

result["response_code"] == 200:

# train name is extracting

# from the result variable data train_name = result["train"]["name"] #

train number is extracting from # the result variable data

train_number = result["train"]["number"]

# from station name is extracting # from the result variable data from_station

= result["from_station"]["name"]

# to_station name is extracting from # the result variable data to_station

= result["to_station"]["name"]

# boarding point station name is

# extracting from the result variable data boarding_point =
result["boarding_point"]["name"] # reservation upto
station name is

# extracting from the result variable data reservation_upto =

result["reservation_upto"]["name"]

# store the value or data of "pnr" # key in pnr_num variable pnr_num =
result["pnr"]

# store the value or data of "doj" key # in variable date_of_journey variable
date_of_journey = result["doj"]

# store the value or data of

# "total_passengers" key in variable total_passengers =
result["total_passengers"]

# store the value or data of "passengers" # key in variable passengers_list
passengers_list = result["passengers"]

# store the value or data of

# "chart_prepared" key in variable chart_prepared = result["chart_prepared"]

# print following values print(" train

name : " + str(train_name) + "\n train

number : " + str(train_number)

+ "\n from station : " + str(from_station)

+ "\n to station : " + str(to_station)

+ "\n boarding point : " + str(boarding_point)

+ "\n reservation upto : " + str(reservation_upto)

+ "\n pnr number : " + str(pnr_num)

+ "\n date of journey : " + str(date_of_journey)

+ "\n total no. of passengers: " + str(total_passengers)

+ "\n chart prepared : " + str(chart_prepared))

# looping through passenger list for

passenger in passengers_list:

# store the value or data # of "no" key in variable passenger_num

= passenger["no"]

# store the value or data of

# "current_status" key in variable current_status = passenger["current_status"]

# store the value or data of

# "booking_status" key in variable booking_status =

passenger["booking_status"] # print following

values print(" passenger number : " +

str(passenger_num) + "\n current status : " +

str(current_status) + "\n booking_status : " +

str(booking_status)) else:

```
print("Record Not Found")
```

13.2.Git Hub & Project demo link:

Git Hub:

https://github.com/IBM-EPBL/IBM-Project-15534-1659600186

Project demo link:

https://youtu.be/hZ_TZTyGsoM