SMART SOLUTIONS FOR RALIWAYS -IoT

A PROJECT REPORT

Submitted by

BENCY BETHEES	**961419104017**
BRINESH BP	**961419104022**
BINEX BIJU SAMUEL	**961419104021**
ATHIL ANIL	**961419104016**

TEAM ID: PNT2022TMID51648

MAR EPHRAEM COLLEGE OF ENGINEERING AND TECHNOLOGY, ELAVUVILAI, MARTHANADAM

INDEX

S.NO	TITLE
1	**INTRODUCTION**
1.1	Project Overview
1.2	Purpose
2	**LITERATURE SURVEY**
2.1	Existing problem
2.2	References
2.3	Problem Statement Definition
3	**IDEATION & PROPOSED SOLUTION**
3.1	Empathy Map Canvas
3.2	Ideation & Brainstorming
3.3	Proposed Solution
3.4	Problem Solution Fit
4	**REQUIREMENT ANALYSIS**
4.1	Functional requirements
4.2	Non-Functional requirements
5	**PROJECT DESIGN**
5.1	Data Flow Diagrams

5.2	Solution &Technical Architecture
5.3	User Stories
6	**PROJECT PLANNING & SCHEDULING**
6.1	Sprint Planning & Estimation
6.2	Sprint Delivery Schedule
6.3	Reports from JIRA
7	**CODING & SOLUTIONING**
7.1	Feature 1
7.2	Feature 2
7.3	Database Schema
8	**TESTING**
8.1	Test Cases
8.2	User Acceptance Testing
9	**RESULTS**
9.1	Performance Metrics
10	**ADVANTAGES & DISADVANTAGES**
11	**CONCLUSION**
12	**FUTURE SCOPE**
13	**APPENDIX**

CHAPTER 1
INTRODUCTION

1.1 PROJECT SUMMARY

The largest railway system in Asia is operated by Indian Railways. Additionally, it is the second-largest network in the world run by a single organization.

This project includes, The access of the train, the price of the ticket, the departure and arrival times of the train, and the ability to book a ticket using a debit, credit, or master card or an upi-id are all available to users in this. If a user wants to cancel a ticket after booking it, they can do so with convenience as well. It has a QR code that holds the customer's information and is useful for the ticket inspector to verify. It could track trains using a gps system. Online ticket booking has many benefits, including cutting down on paper use and electricity use at the train ticket counter. It also saves time by eliminating the need to wait in a lengthy line. Our railroad reservation system is employed in order to prevent manual errors.

IoT technologies help railways successfully manage passenger safety, operational efficiency, and the passenger experience. Snart sensors can be used to track important assets, manage passengers flow, and enable predictive maintenance.

1.2. PURPOSE

A train passenger needs to be aware of the availability of their tickets, the status of their reservations for a certain train or location, the arrival and departure times of their train, any special trains, and the location of their train. Such inquiries cannot be answered during peak hours by the customer information centers at train stations. Less reservation counters are available for customers and travelers. On many reservation systems, it takes a while for anyone to make a reservation. The traveler's cannot get assistance from call centers. The goal of the online railway ticket reservation system is to provide a web application that offers customers the option to book tickets online as well as information about available trains.

CHAPTER 2
LITERATURE SURVEY

TITLE: TRACKWARN-AN AI-DRIVEN WARNING SYSTEM FOR RAILWAY TRACK WORKERS

AUTHOR: M.I.M.Amjath

YEAR OF PUBLICATION: 2021

he analyzed and looked at the train mishaps, hospitalization keep, and so on. It gets in to extra portrayal of insights. The peril of huge injury, in light of separation cosmopolitan, is multiple times greater for travelers travel via car contrasted and travelers going by rail. The mean length of keep in clinic for a transport mishap including a railroad train was four days that were longer than the mean length of save for all External reasons for injury. A train is a set of vehicles, empty or loaded worked by locomotive, or any other self-propelled unit, including light engine/engines or rail-motor vehicles or a single rail-motor vehicle, empty or conveying passengers, live-stock, parcels or goods, which cannot be readily lifted off the track and running under a particular number or a distinct name from fixed point of departure to a fixed destination. Part of a train shall also be treated as a train for the purpose of these definition, classification and statistics. The train engine or any other vehicle once put on the train continues to be a part of the train until the station is reached beyond which it is not required to go on the same train.

DISADVANTAGE:

Slows down the study process.

TITLE: MONITORING OF THE OPERATING PARAMETERS OF RAILWAY SYSTEMS THROUGH THE USE OF SMARTPHONE DETECTION TECHNOLOGIES

AUTHOR: Francesco Apicella

YEAR OF PUBLICATION: 2021

Our cities are becoming increasingly smart thanks to information and communication technologies, sustainable solutions for human activities and innovative mobility frameworks. In this context, one of the most promising approaches is relying on the Internet of Things (IoT) which allows objects of everyday life to become computing devices exchanging useful data. In particular, the paper proposes to adopt such technologies for monitoring railway systems, thus obtaining information concerning, on one hand, service performance and, on the other, travellers behaviour. A numerical application has been performed in a real rail context, thus pointing out the feasibility of the proposed methodology. Improving safety by early warning of distress in or impending failures in wheels and wheel bearings. Using the vibration signature of the same sensors that are strategically placed on unsprung mass to do track condition monitoring too to indicate deterioration in the health of tracks thereby avoiding sudden failures in service.

DISADVANATGE:

Rail transport cannot provide door to door service as it is tied to a particular track. Intermediate loading or unloading[5] involves greater cost, more wear and tear and wastage of time.

2.1 EXISTING SYSTEM

A GSM and GPS module were used to pinpoint the exact location of the defective tracks so that the authorities could be notified via SMS and sent a link to view the area on Google Maps. A prototype that can take pictures of the track, compare them to an older database, and alert the authorities of a breach in the surface was presented by Rizvi Aliza Raza. Table provides a thorough examination of conventional railway track defect detecting methods.

2.2 REFERENCES

1. D. Hesse, “Rail Inspection Using Ultrasonic Surface Waves” Thesis, Imperial College of London, 2007.
2. Md. Reya Shad Azim1 , Khizir Mahmud2 and C. K. Das. Automatic railwaytrack switching system, International Journal of Advanced Technology, Volume 54, 2014.
3. S. Somalraju, V. Murali, G. saha and V. Vaidehi, “Title-robust railway crack detection scheme using LED (Light Emitting Diode) - LDR (Light Dependent Resistor) assembly IEEE 2012.
4. S. Srivastava, R. P. Chourasia, P. Sharma, S. I. Abbas, N. K. Singh, “Railway Track Crack detection vehicle”, IARJSET, Vol. 4, pp. 145-148, Issued in 2, Feb 2017.
5. U. Mishra, V. Gupta, S. M. Ahzam and S. M. Tripathi, “Google Map Based Railway Track Fault Detection Over the Internet”, International Journal of Applied Engineering Research, Vol. 14, pp. 20-23, Number 2, 2019.
6. R. A. Raza, K. P. Rauf, A. Shafeeq, “Crack detection in Railway track using Image processing”, IJARIIT, Vol. 3, pp. 489-496, Issue 4, 2017.
7. Khekare G S, Sakhare A V. A smart city framework for intelligent traffic system using VANET[C]// International Multi-conference on Automation. IEEE, 2013.
8. COOPER Dave E. Intelligent transportation systems for smart cities: a progress review[J]. Science China (Information Sciences), 2012, 55(12):2908-2914.

9. Stefansson G, Lumsden K. Performance issues of Smart Transportation Management systems[J]. International Journal of Productivity & Performance Management, 2009, 58(1):55-70.

10. Huang X. Smart Antennas for Intelligent Transportation Systems[C]// International Conference on ITS Telecommunications Proceedings. IEEE, 2006:426-429.

11. Li X, Song J. The Top Design Methodology of Smart City in China[C]// International Conference on Intelligent Computation Technology and Automation. IEEE, 2014:861-864.

12. Jianbo, Cheng, Peng. Top-Level Design of Smart City Based on "Integration of Four Plans"[J]. ZTE Communications, 2015, 13(4):34-39.

13. Lanke N, Koul S, Lanke N, et al. Smart Traffic Management System[J]. International Journal of Computer Applications, 2014, 75(7):19-22.

14. Bouhedda M, Bellatreche S, Ahmed-Serier R. Smart traffic signal controller design and hardware implementation-based ant colony system[C]// International Conference on Modelling, Identification and Control. IEEE, 2017:1110-1116

2.3 PROBLEM STATEMENT DEFINITION

In a train a ticket is issued by a railway operator that allows users to travel on the railway. User can use tickets to travel on a specific route at a specific time. The introduction of manual ticketing has always made it easier for authorities to keep track of various events involving financial matters and people's travelling patterns. These activities, however, had to be carried out manually, requiring a large amount of manpower and resources. The use of electronic equipment such as computers, printing machines, paper, and ink are the primary and most important resources required for paper ticketing systems. Smart railway solutions are mainly concentrating on passengers as well as the reliability of the trains. The solution helps in saving the time of the passengers It also helps in long-term running of the train as it become well maintained due to regular inspection Keeping trains running safely, economically and on time.

CHAPTER 3

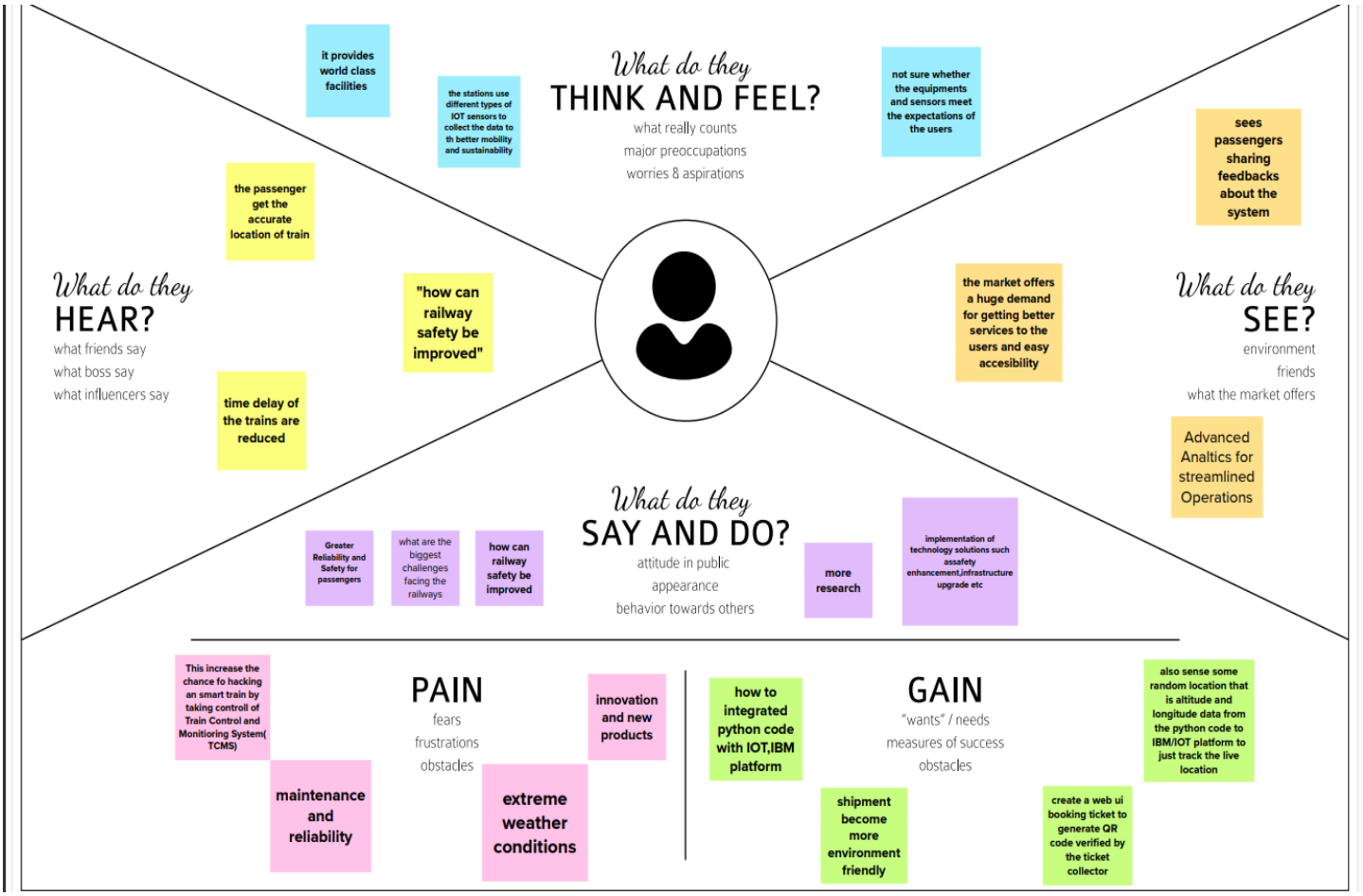
IDEATION AND PROPOSED SOLUTON

3.1 EMPATHY MAP CANVAS

An Empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user person, an empathy map can represent a group of users, such as a customer segment .

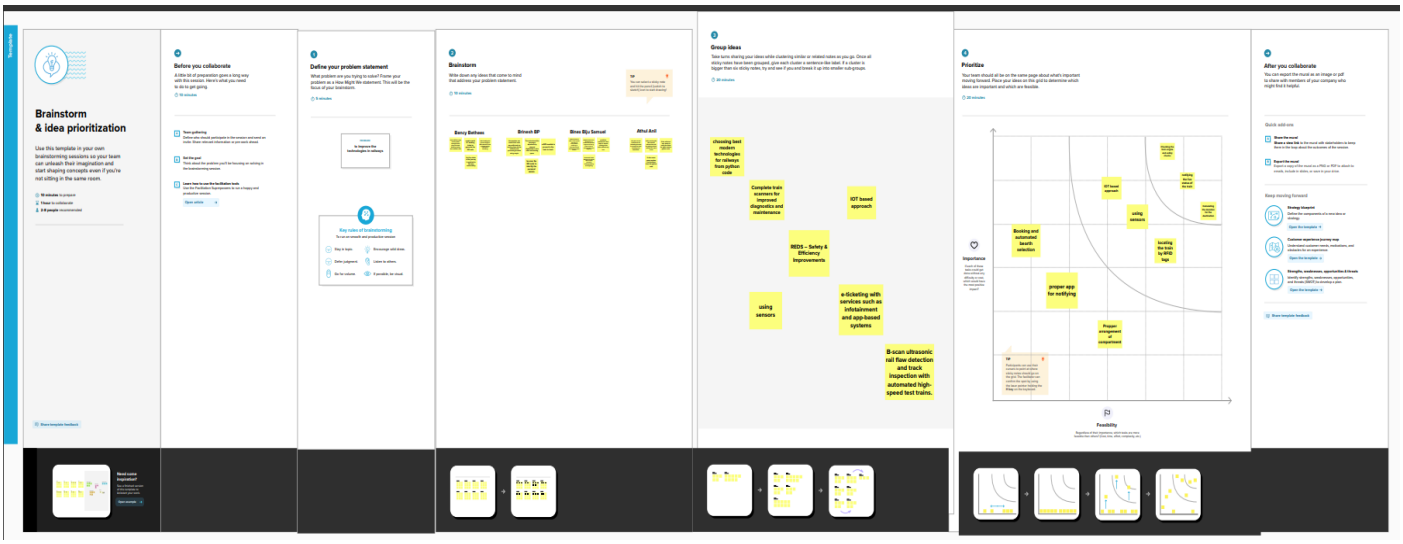
The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.

In this activity you are expected to prepare the empathy map canvas to capture the user Pains & Gains, Prepare list of problem statements



3.2 IDEATION & BRAINSTORMING

Brainstorming is one of the primary methods employed during the Ideation stage of a typical Design Thinking process. Ideation refers to the whole creative process of coming up with and communicating new ideas. It can take many different forms, from coming up with a totally new idea to combining multiple existing ideas to create a new process or organizational system. Ideation is similar to a practice known as brainstorming. In this activity you are expected to list the ideas by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance

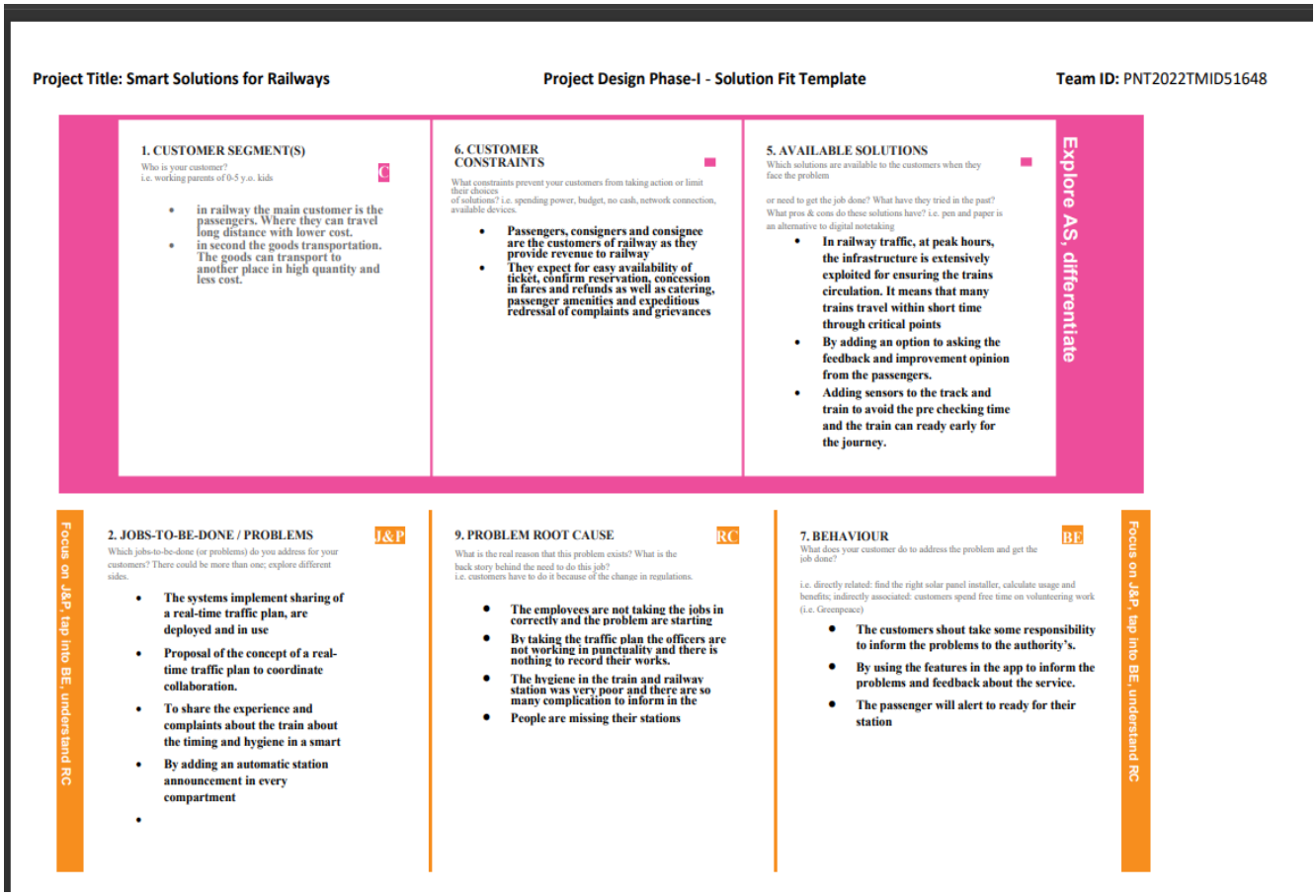


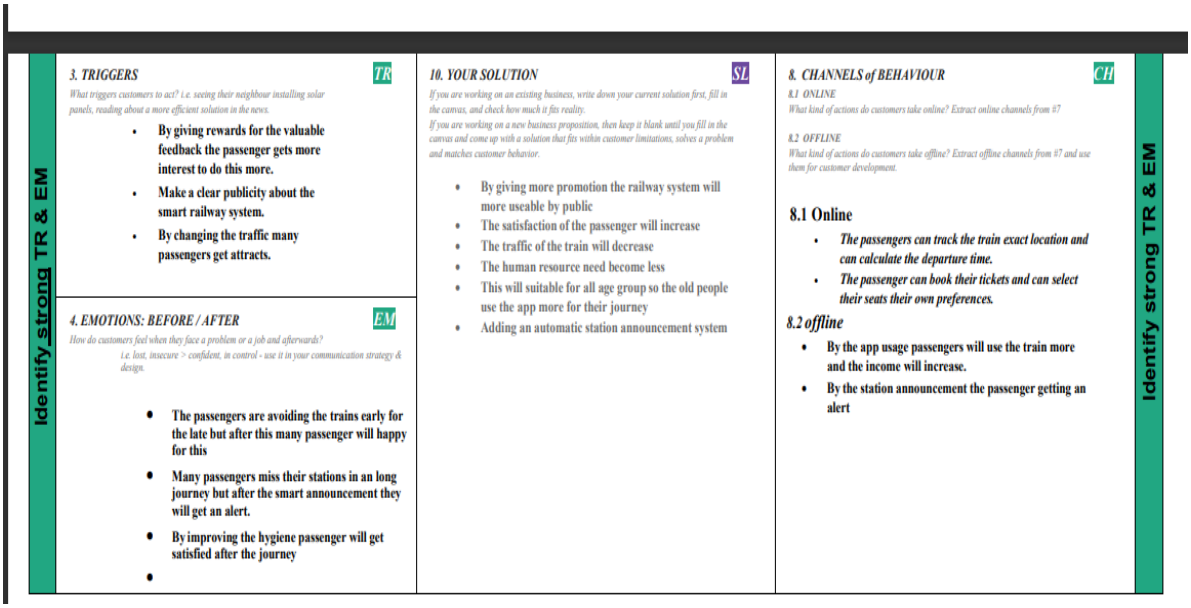
3.3 PROPOSED SOLUTIONS

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Smart Solutions for railways is designed to reduced the work load of the user and also the use of paper
2.	Idea / Solution description	A Web UI is designed to enable online ticket booking and a QR code is generated for the user who has booked the ticket to verify it with Travelling Ticket Examiner (TTE) which is done using Cloud Service .
3.	Novelty / Uniqueness	• Unique QR code for each user to verify their Tickets by Ticket Checker. • GPS Module to track the location of Train and live status is updated in the Web app .
4.	Social Impact / Customer Satisfaction	• GPS Module to track the location of Train and live status is updated in the Web app . • The location of the train is updated periodically in the web app so customers can easily track the status of the train which may help the customer to arrive on time .
5.	Business Model (Revenue Model)	• Selling a Product which enables online booking platform and automatic verification system would fetch more revenues to the Organisation , also the Online Platform has to be maintained continuously and so , the business will sustain and can be improved if required .
6.	Scalability of the Solution	• a QR code is generated to hold the unique data of user in cloud , and a GPS module is attached to lot platform inorder to monitor the train location which gives an additional tracking mechanism to ensure reliability . • We propose a solution which works on the SaaS (Software as a Service) cloud model wherein all necessities are deployed in the web browser which gives an easy access and also the cost is minimal which gives a scaled solution approach .

3.3 PROBLEM SOLUTION FIT

The Problem-Solution Fit canvas is based on the principles of Lean Startup, LUM (Lazy User Model) and User Experience design. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why. Technologically advanced approach to reduce the work load of the users and also the use of paper





CHAPTER 4

REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENTS

Functional requirements describe the desired end function of a system operating within normal parameters, so as to assure the design is adequate to make the desired product and the end product reaches its potential of the design in order to meet user expectations

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Before the user registration there will be language selector .All the language is applicable .When user enter in to the website they can see the page which shows enter the email-id ,mobile number and name. After that in screen it shows the verification code which will be sent to the email-id.
FR-2	User verification	The verification code is send to the registered email id.
FR-3	User confirmation	The verification code is entered in to the website. After finishing that home page is opened.
FR-4	Process of booking	When the home page is opened there will be a From and To options. We must enter the details then after that we can able to see the number of trains availability and seats availability. We can select the particular train and particular seats which we need and click the confirm option.
FR-5	Payment process	After entering all the details select the payment option like UPI apps , Net-banking , etc., When we select the comfortable method then it process through selected payment option then payment can be done carefully and securely, then the ticket will be confirmed. After confirmation it will return to the page and we can see the details of booking.
FR-6	Confirmation message	After all the process has been completed the QR code will be send to both mobile number(via SMS) and email id. QR code will be shown to the ticket collector where all the booking details can be viewed by scanning the QR code.

4.2. NON-FUNCTIONAL REQUIREMENTS

Non-Functional Requirements ensure the software system follows legal and adherence rules, specify the quality attribute of the software, ensure the reliability, availability, performance, and scalability of the software system and help in constructing the security policy of the software system.

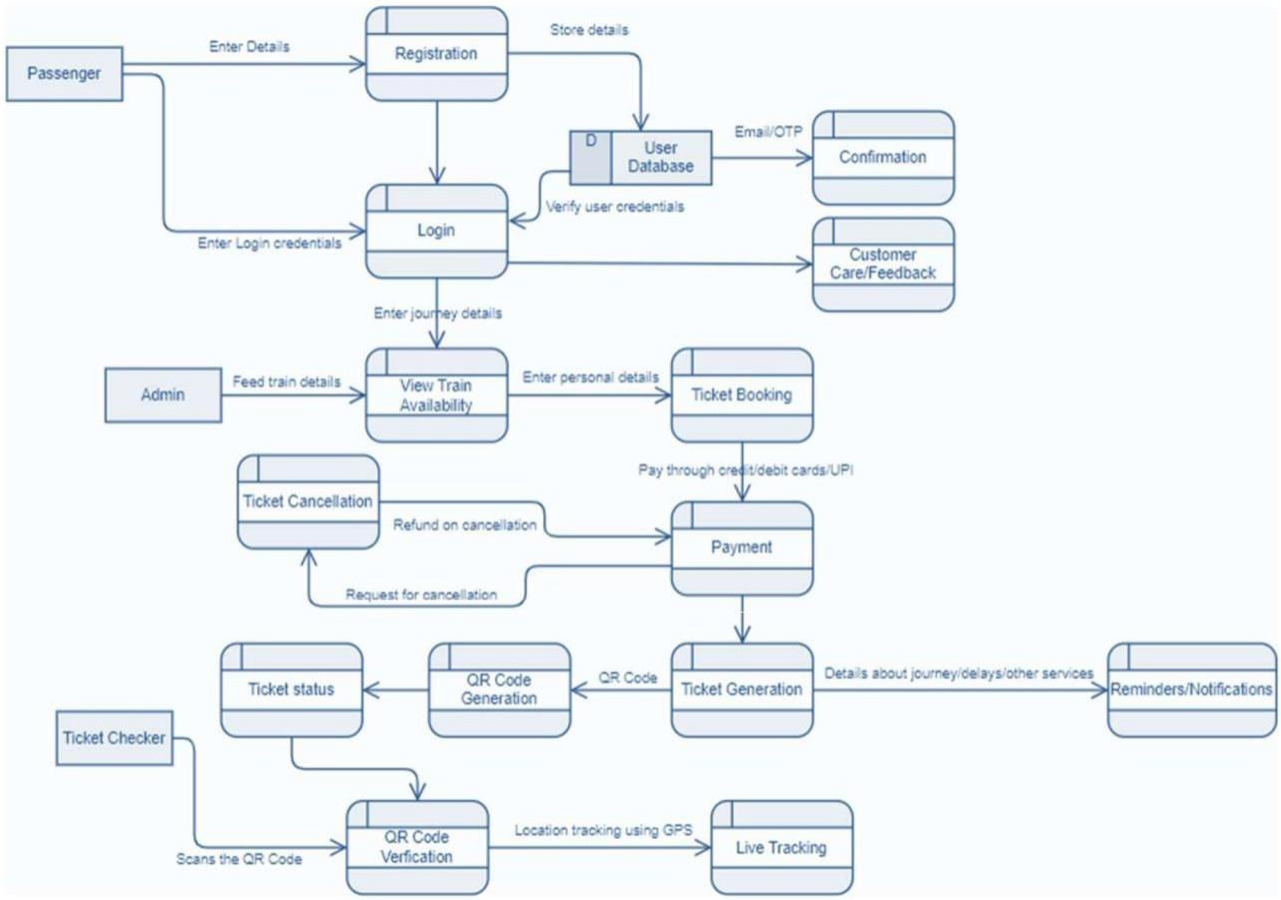
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	A new system should empower the railway industry to better leverage the potential given by modern communication technology. Unlike ERTMS, FRMCS will decouple applications, services and transport to allow independence and transport bearer flexibility.
NFR-2	**Security**	Integrated the Simple Modeling Language for Embedded Systems (SMoLES) with the Security Model Analysis Language (SMAL). SMAL provides security extensions to the composition meta-model of the Domain Specific Modeling Language (DSML)and can express access control policies for IoT applications. The resulting framework is called SMoLES Security (SMoLES-SEC).
NFR-3	**Reliability**	Achieving an increase in reliability and safety parametersby even a few percentage points is a rare statistical event 1. Monitoring of failure-prone systems on locomotives, such as the engine or electrical systems can increase the reliability significantly 2. Monitoring of bridges regarding material stress ordynamic behavior to detect changes indicating future failure
NFR-4	**Performance**	• Speed acceleration • Improvement of operations, efficiency of scheduling and increased energy efficiency
NFR-5	**Scalability**	• This model can reduce the man power

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

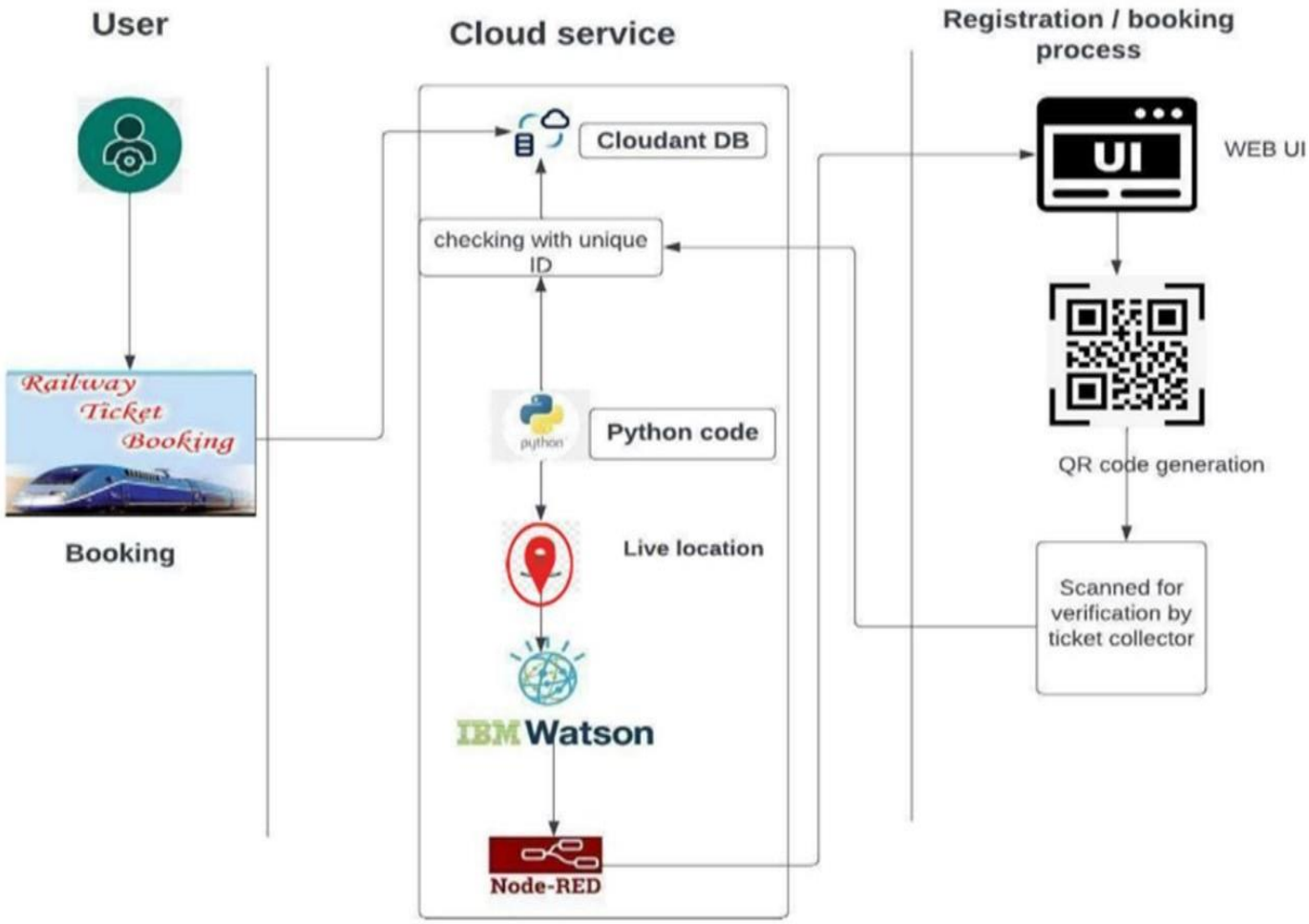
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. The objective of a DFD is to show the scope and boundaries of a system as a whole.



5.2 SOLUTION & TECHNICAL ARCHITECTURE

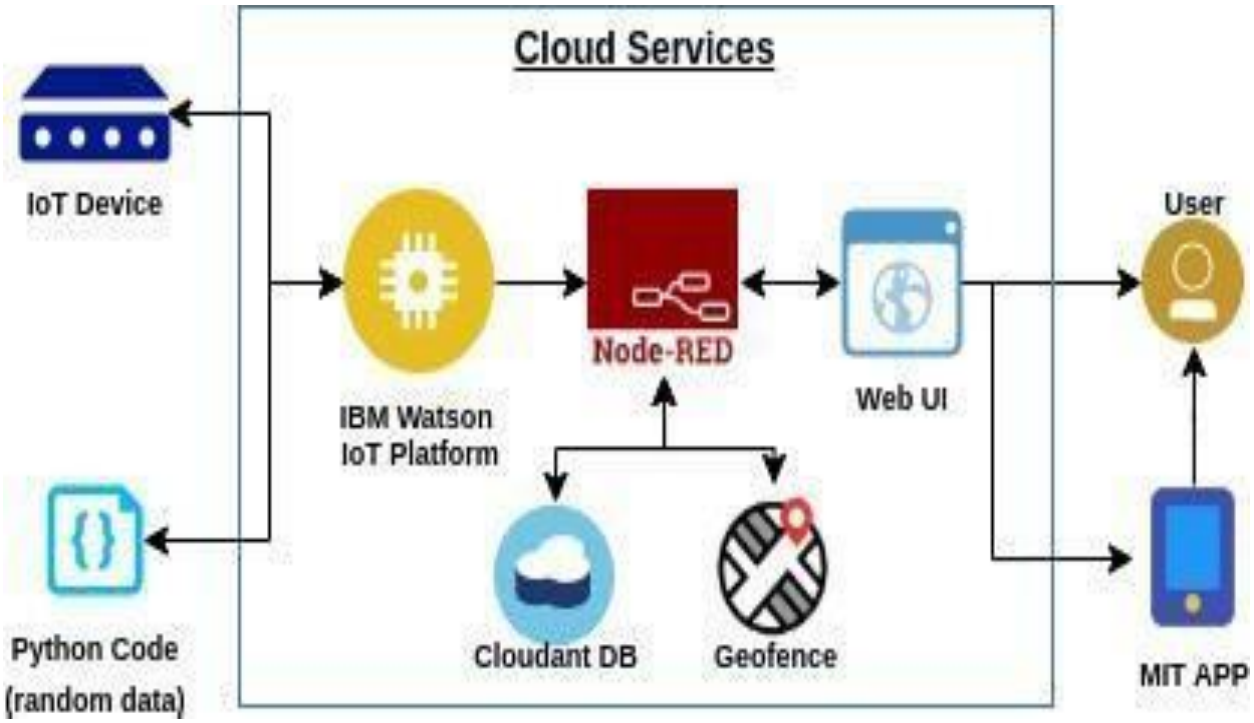
Solution Architecture

Solution architecture provides the ground for software development projects by tailoring IT solutions to specific business needs and defining their functional requirements and stages of implementation



Technical Architecture

Technical Architecture ensures that technology fits into existing computer systems by specifying its hardware, access methods, protocols and more.



5.3 USER STORIES

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile User)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Confirmation	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		High	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user I can enter my dashboard by entering the username and the password	I can access the website	High	Sprint-1
Customer (Web User)	Booking	USN-7	As a user I can access the ticket booking section	I can get about the available tickets/seats	High	Sprint-1
Customer Care Executive	Customer Queries	USN-8	As a customer care executive, I can check the customer queries they posted in the website	I can improve the customer satisfaction	High	Sprint-1
Administrator	Maintaining website	USN-9	As an administrator, I can maintain the website and enhance the online presence	I can improve the website appearance and usability	High	Sprint-2
		USN-10	As an administrator, I can maintain the issues in the ticket booking section	I can improve the issues in ticket booking section	High	Sprint-2
		USN-11	As an administrator, I can update website content	I can ensure the content is in harmony with the customers overall objectives	Medium	Sprint-2
		USN-12	As an administrator, I can improve website	I can enhance the user experience	High	Sprint-2

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING &ESTIMATION

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	25 SEPTEMBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	19 SEPTEMBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	25 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	24 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	30 OCTOBER 2022
Solution Architecture	Prepare solution architecture document.	19 SEPTEMBER 2022

Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	31 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	16 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	31 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram.	31 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	01 NOVEMBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	Sprint 1- 08 November 2022 Sprint 2- 10 November 2022 Sprint 3- 14 November 2022 Sprint 4- 17 November 2022

6.2.PRINT DELIVERY SCHEDULE

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Monitor the Speed of Train	USN-1	The Railway must take care of passengers and peoples.In the train there are so many families we should secure them.	2	High	Bency Bethees Brinesh B.P Binex Biju Samuel Athul Anil
Sprint-2	Avoid From Accidents	USN-2	If any accident occurs their technical team will take care of it and save the passengers.	1	High	Bency Bethees Brinesh B.P Binex Biju Samuel Athul Anil
Sprint-3	Detect the Motions	USN-3	We have monitor the motions and delays by 24/7 hrs. To avoid the accidents,and delays by using only sensors. The railway must takecare of what are the necessary process to avoid the train accidents and delays.	2	Low	Bency Bethees Brinesh B.P Binex Biju Samuel Athul Anil
Sprint-4	The model is trained and tested by sample dataset.	USN-4	The programmer design the model to detect the Train Details.	2	Medium	Bency Bethees Brinesh B.P Binex Biju Samuel Athul Anil

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-5	Warning message	USN-5	In case any accident or delay occur, the device give the alarm and alert message to concerned department within a minute.	1	High	Bency Bethees Brinesh B.P Binex Biju Samuel Athul Anil

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

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	3 Days	8 Nov 2022	10 Nov 2022	20	10 Nov 2022
Sprint-2	20	3 Days	11 Nov 2022	13 Nov 2022	20	13 Nov 2022
Sprint-3	20	3 Days	14 Nov 2022	16 Nov 2022	20	16 Nov 2022
Sprint-4	20	3 Days	17 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

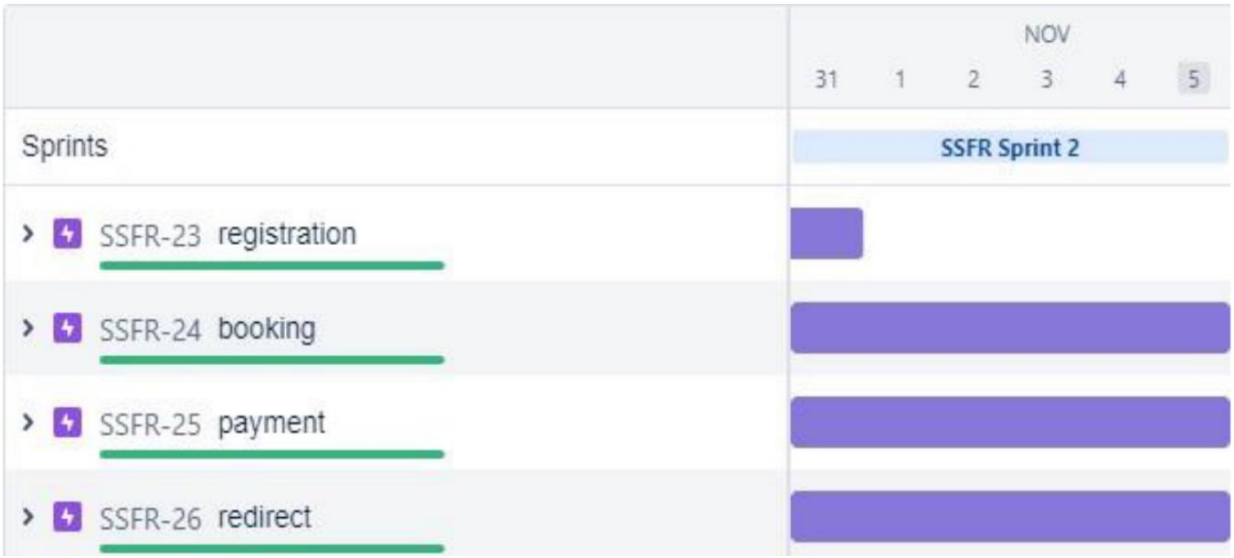
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let us calculate the team's average velocity (AV) per iteration unit (story points per day)

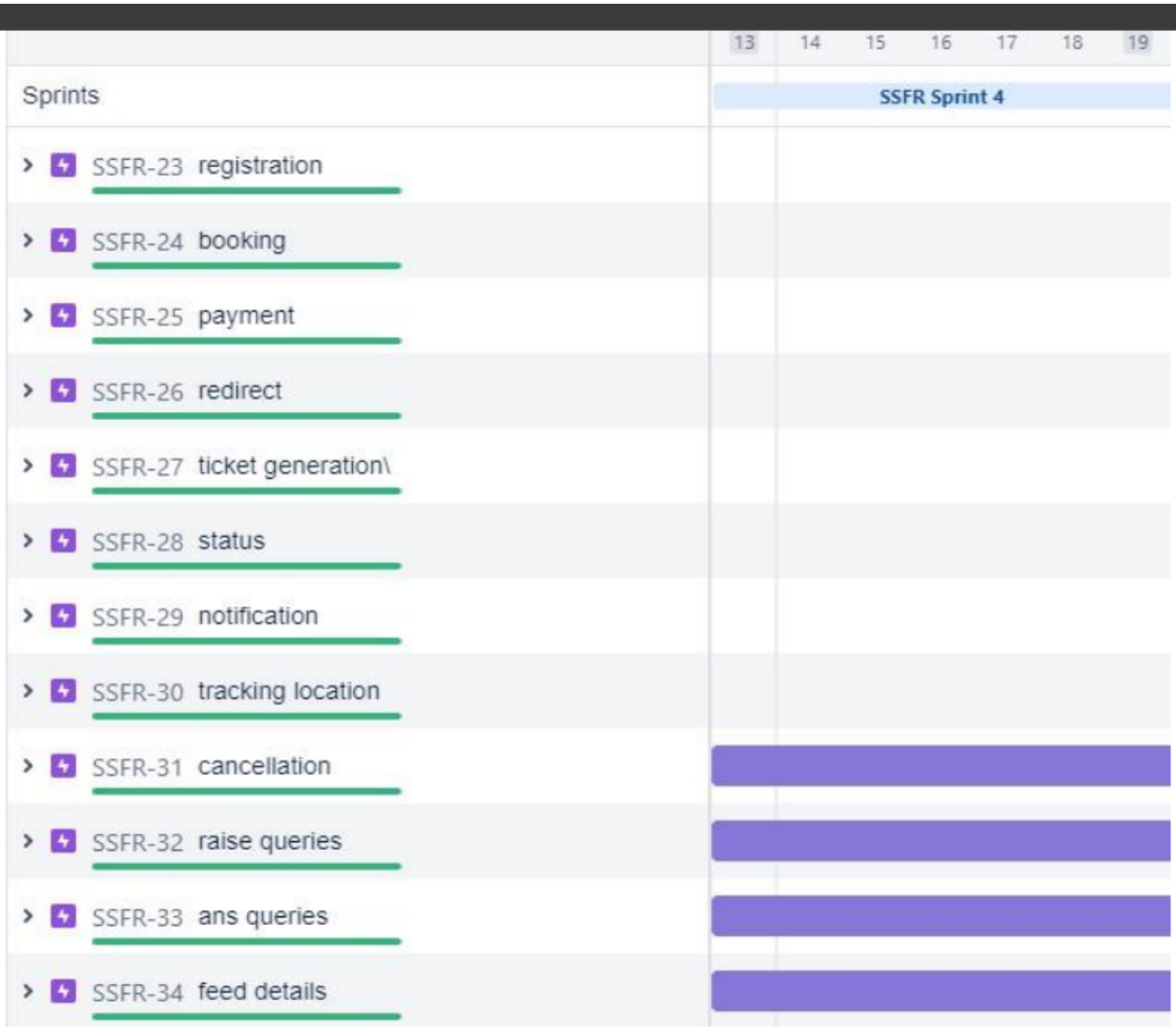
$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

6.3. REPORTS FROM JIRA

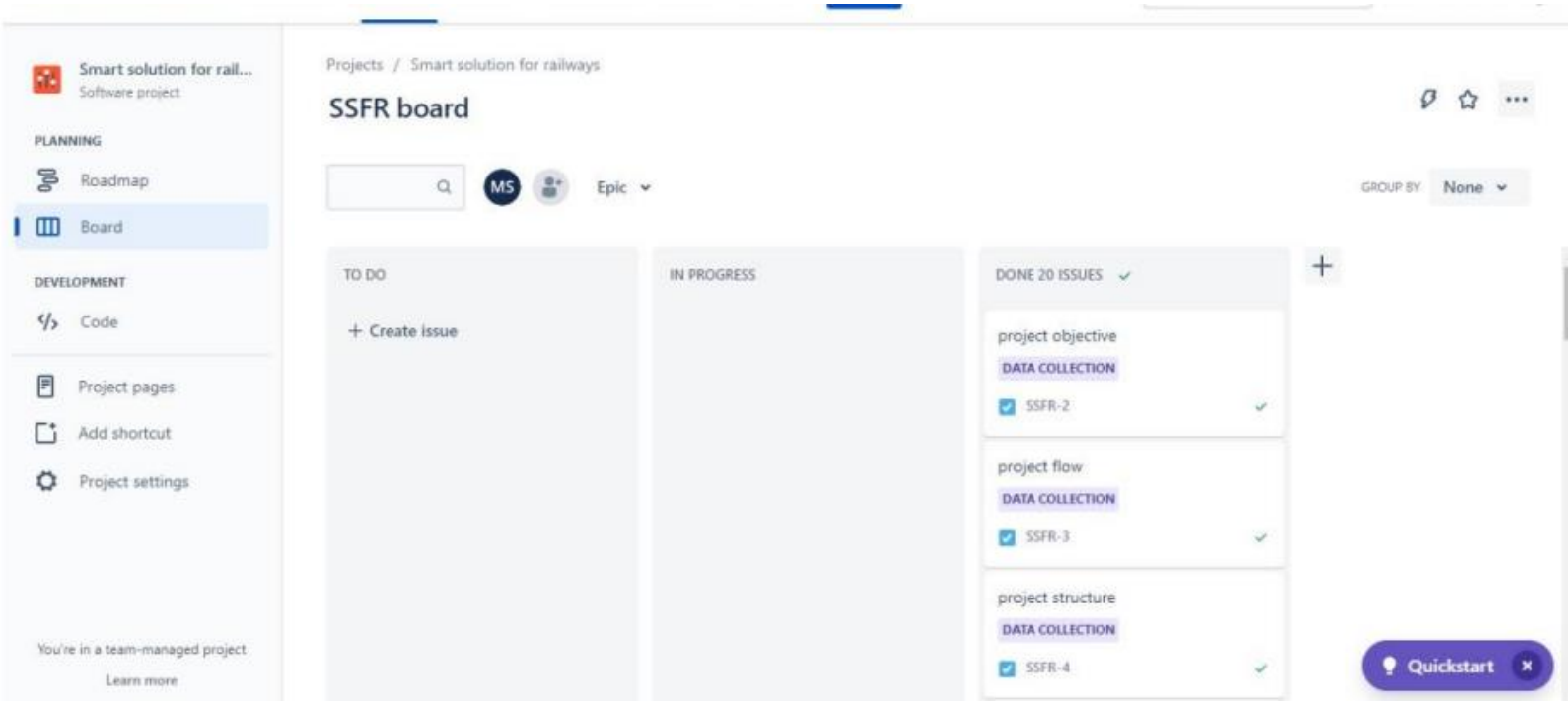
JIRA is a software testing tool developed by the Australian Company Atlassian. It is a bug tracking tool that reports all the issues related to your software or mobile apps. JIRA is based on the Agile methodology.

a) CREATING ROAD MAP





b) CREATE ASSFC BOARD IN JIRA SOFTWARE



CHAPTER 7

CODING AND SOLUTIONING

7.1. FEATURE 1

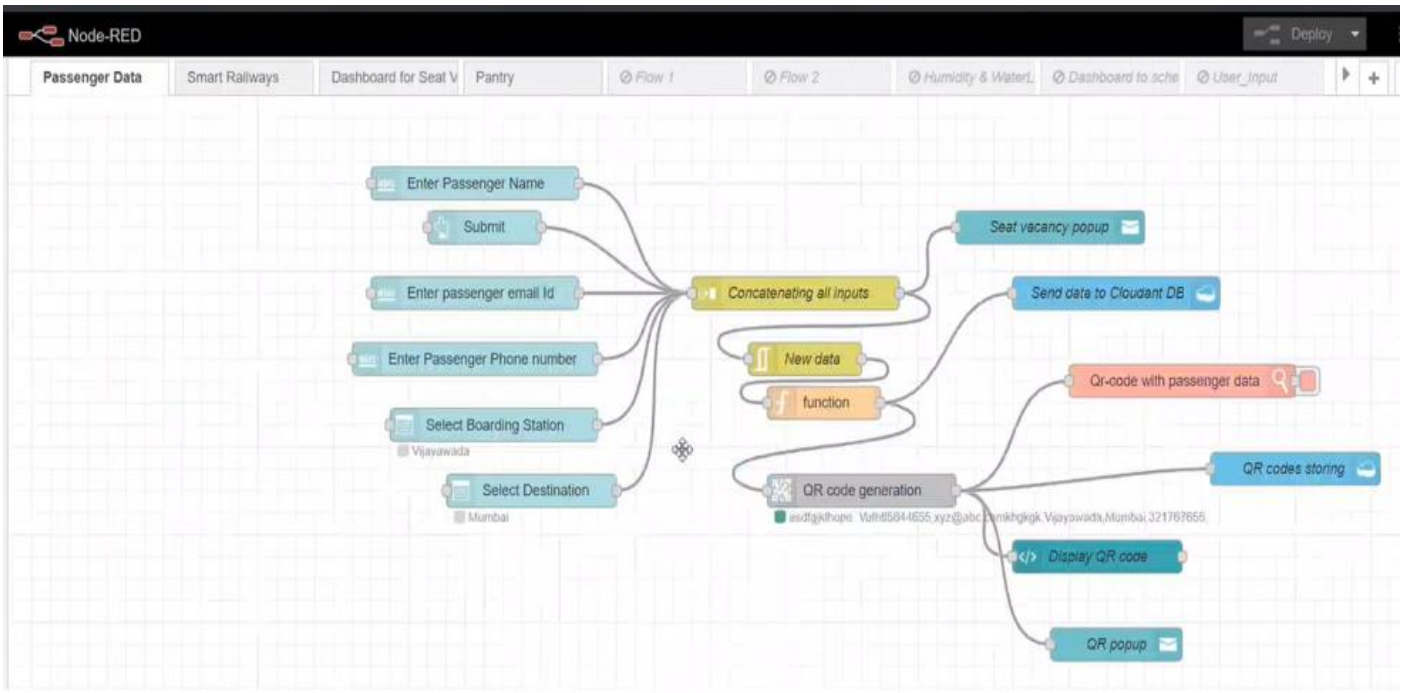
- IoT Device
- IBM Watson platform
- Node-Red
- Cloudant DB
- Web UI
- Geo fence
- MIT App
- Python code

7.2. FEATURE 2

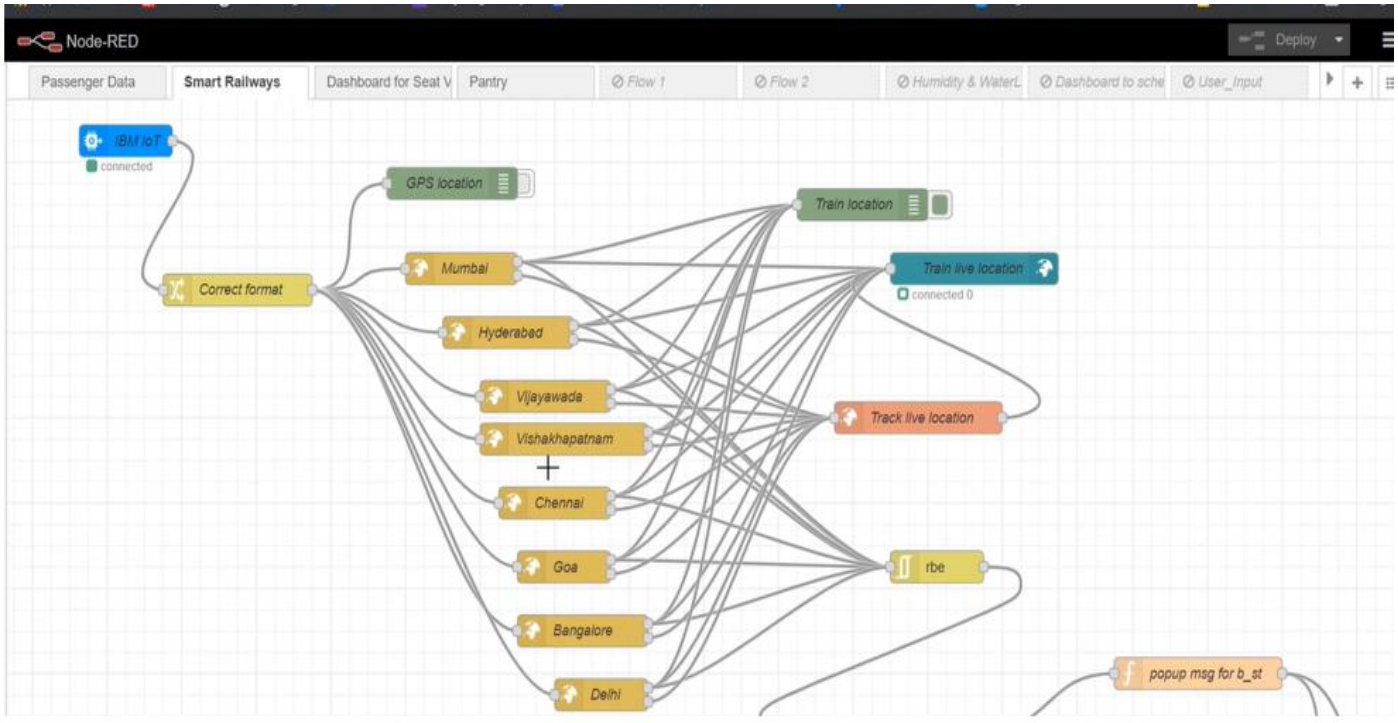
- Registration
- Login
- Verification
- Ticket booking
- Payment
- Ticket cancellation
- Adding queries

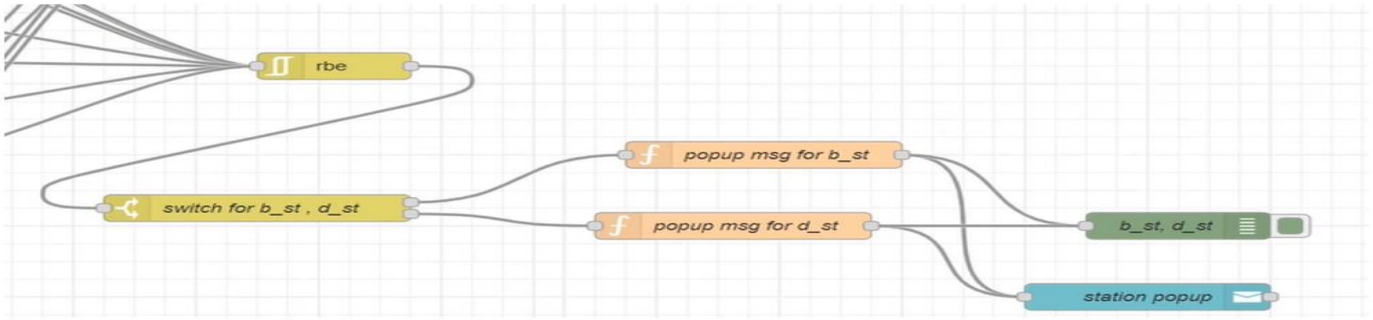
Node-Red

Passenger Data

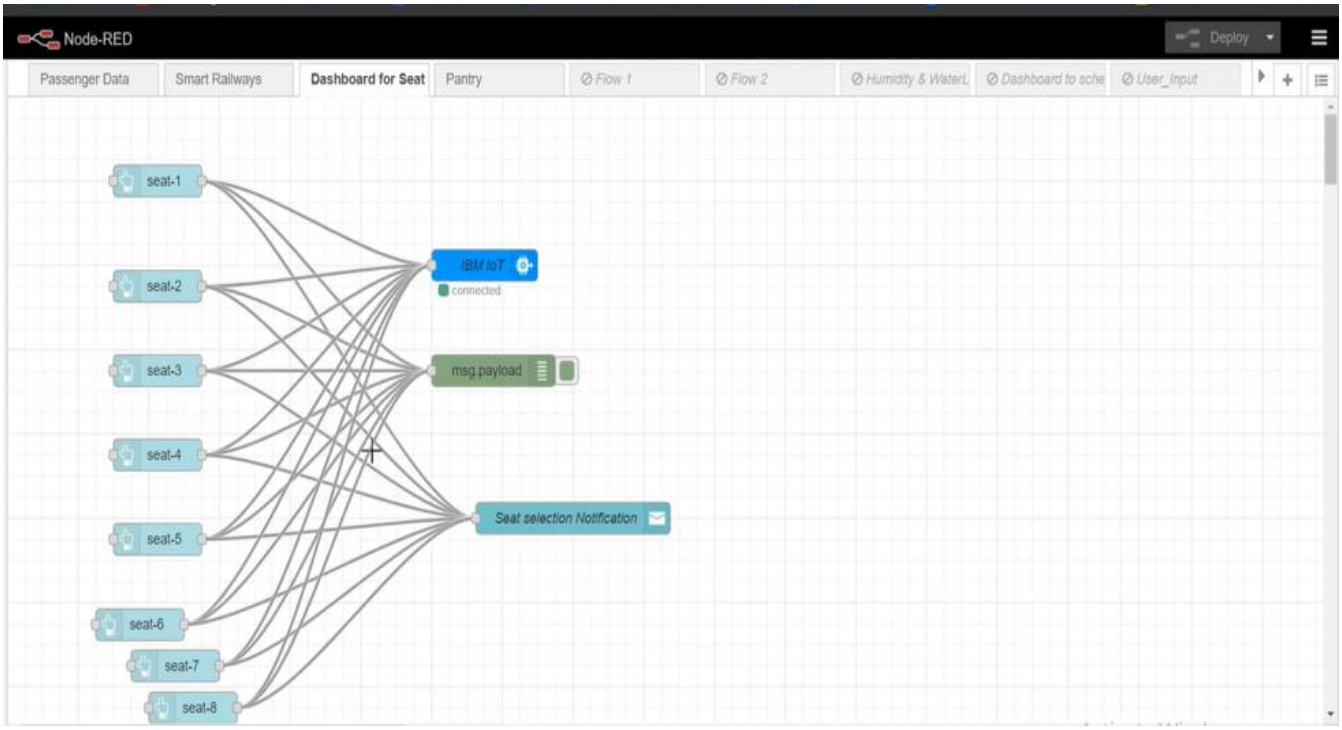


Smart Railway

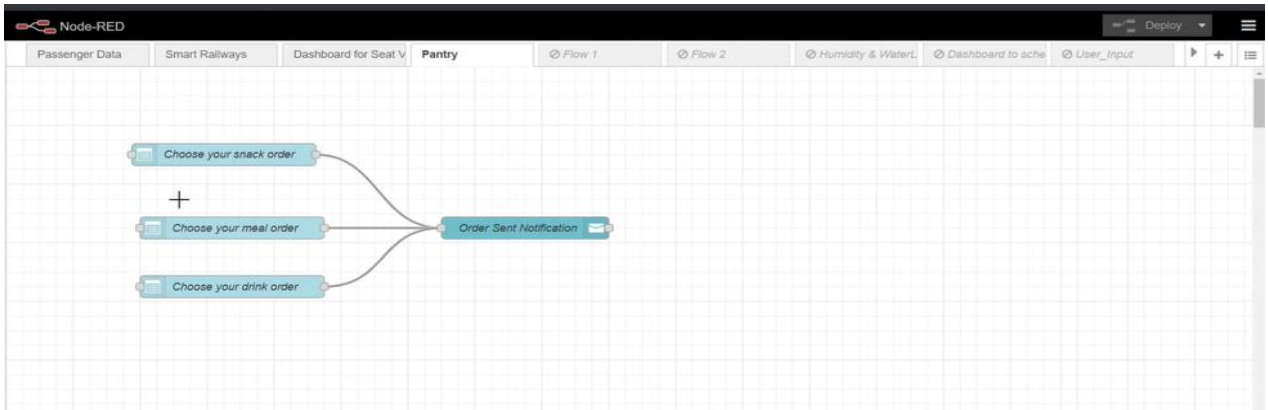




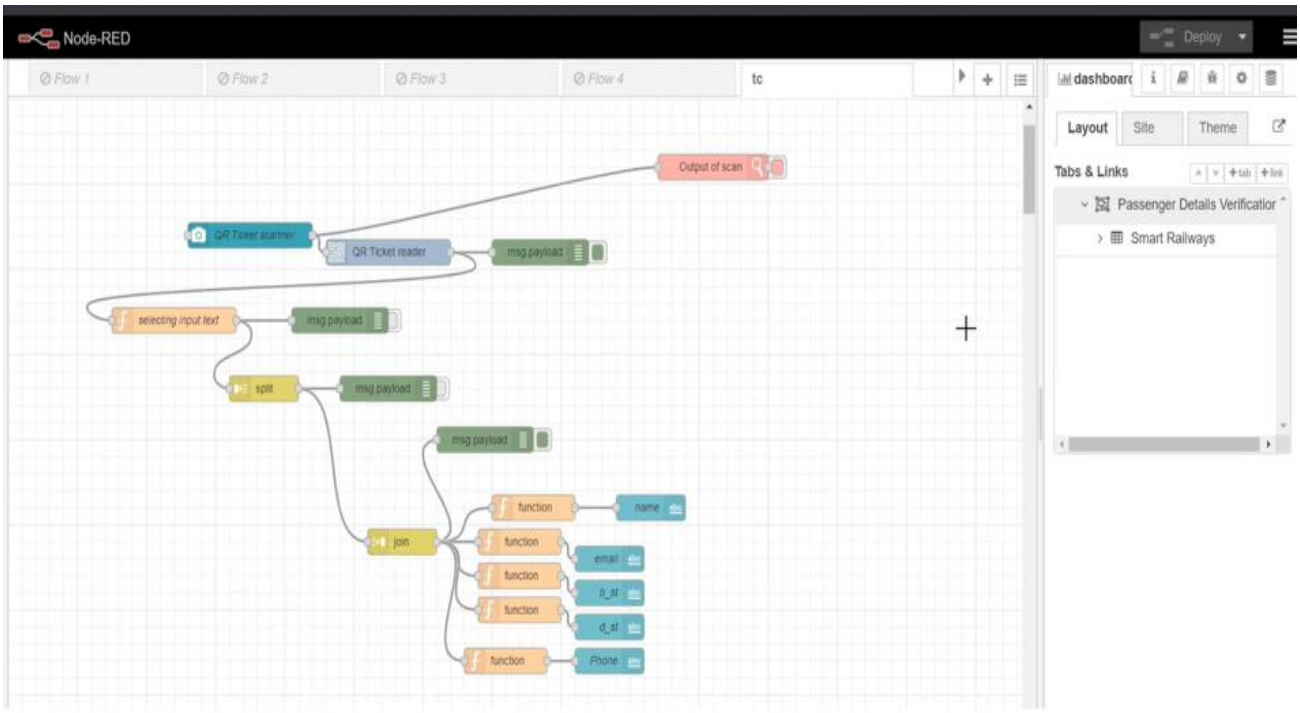
Dashboard for seat booking



Pantry



TC



QR CODE SCANNER:

```
import cv2 as cv import
numpy as np import time
import pyzbar.pyzbar as pyzbar
from ibmcloudant. cloudant_v1 import CloudantV1 from ibmcloudant
import CouchDbSessionAuthenticator from
ibm_cloud_sdk_core.authenticators import BasicAuthenticator
import wiotp.sdk.device
    authenticator=BasicAuthenticator('apikey-v2- 2ji0x00sov1b6clf61hctelp07os2c41mauy6mk7a3ot',
'6866a033c311b4968d996ca9fa217206') service=CloudantV1(authenticator=authenticator)
service.set_service_url('https://apikey-v2-
2ji0x00sov1b6clf61hctelp07os2c41mauy6mk7a3ot:6866a033c311b496d996ca9fa
217206@53e4077b-d008-4545-8ea1-
1d70926b1b71bluemix.cloudantnosqldb.appdomain.cloud')
  cap = cv.VideoCapture(0)
 font = cv.FONT_HERSHEY_PLAIN
 if not cap.isOpened():
      print("Cannot open camera")
exit()

myConfig = {
        "identity" :{
            "orgId":"u3neop",
```

```
                "typeId":"qrcode",
                "deviceId":"1234567"
},
 "auth":{
        "token":"1234567890"
         }         }   def
 myCommandCallback(cmd):
            print("Message received fromIBM 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=response, qos=0, onPublish=None)
    print("Published data Successfully: %s",response)
 print("\n")
   while
True:
 ret,
frame=cap.r
ead()
       decodedObjects = pyzbar.decode(frame)          if not ret:
print("Can't receive frame (stream end?). Exiting ...")
 break          for obj in decodedObjects:
 a=obj.data.decode('UTF-8')
              cv.putText(frame, "Ticket", (50,50),font,2, 17 (255 ,0, 0),3)
 try:
              response=service.get_document( db='bookingdetails', doc_id = a ) .get_result()
print(response)
print("\n\n")              pub(response)
time.sleep(5)             except Exception as
 e:
               response={'Error':'Not a Valid Ticket'}
 pub(response)          print("Not a Valid Ticket")          print("\n\n")
time.sleep(5)
        cv.imshow("Frame" ,frame)          if
 cv.waitKey(1) & 0xFF == ord('q'):
 break
         client.commandCallback = myCommandCallback cap.release()
 cv.destroyAllWindows()
 client.disconnect()
```

Python Code for ticket verification:

```
import cv2 as cv
import numpy as np
import time
import pyzbar.pyzbar as pyzbar
from ibmcloudant.cloudant_v1 import CloudantV1
from ibmcloudant import CouchDbSessionAuthenticator
```

```
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator

import wiotp.sdk.device

authenticator=BasicAuthenticator('apikey-v2-
1w8tqt2prt3j7qz9d1rgrxhar3w9v43i2359u79ut5jb','86181a38eca19ae487f512b10aca0c80')
service=CloudantV1(authenticator=authenticator)

service.set_service_url('https://apikey-v2-
1w8tqt2prt3j7qz9d1rgrxhar3w9v43i2359u79ut5jb:86181a38eca19ae487f512b10aca0c80@9163f25
a-

10b8-4374-a8de-cb92e4357567-bluemix.cloudantnosqldb.appdomain.cloud')

cap = cv.VideoCapture(0)

font = cv.FONT_HERSHEY_PLAIN

if not cap.isOpened():
print("Cannot open camera")

exit()


myConfig = {

"identity" :{

"orgId":"ryc4pr",
"typeId":"QR_Reads",

"deviceId":"876543"

},
"auth":{

"token":"GGHvsi!XL-i7x0mC6B"

}

}
def myCommandCallback(cmd):

print("Message received fromIBM 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=response, qos=0,
onPublish=None)
print("Published data Successfully: %s",response)
```

```
print("\n")

while True:

ret, frame=cap.read()

decodedObjects = pyzbar.decode(frame)
if not ret:

print("Can't receive frame (stream end?). Exiting ...")

break

for obj in decodedObjects:
a=obj.data.decode('UTF-8')

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

(255 ,0, 0),3)

try:

response=service.get_document(
db='bookingdetails',

doc_id = a

) .get_result()

print(response)
print("\n\n")

pub(response)

time.sleep(5)
except Exception as e:

response={'Error':'Not a Valid Ticket'}

pub(response)

print("Not a Valid Ticket")
print("\n\n")

time.sleep(5)

cv.imshow("Frame" ,frame)

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

client.commandCallback = myCommandCallback

cap.release()

cv.destroyAllWindows()
client.disconnect()
```

Python code for train tracking:

```
import wiotp.sdk.device
import time

import random

myConfig = {

"identity" :{
"orgId":"ytluse",

"typeId":"2702",
"deviceId":"12345"

},
"auth":{

"token":"O+n)Eh+lNX0y3?rG!8"
}

}
def myCommandCallback(cmd):

print("Message received fromIBM 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':'Delhi Express','lat':13.344279,'lon':80.214367}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':13.515254,'lon':80.093518}
pub(myData)
```

```
time.sleep(3)
myData = {'name':'Delhi Express','lat':13.728799,'lon':80.005627}

pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':13.910160,'lon':79.906750}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':14.102035,'lon':79.851819}
pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':14.261807,'lon':79.862805}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':14.623537,'lon':79.950695}
pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':15.111987,'lon':79.994641}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':15.313413,'lon':80.005627}
pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':15.567568,'lon':80.104504}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':15.747405,'lon':80.269299
}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':15.821409,'lon':80.302258}
pub(myData)
```

```
time.sleep(3)

myData = {'name':'Delhi Express','lat':15.927082,'lon':80.445080}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':16.022141,'lon':80.554943}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':17.033801,'lon':80.295512}
pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':18.383088,'lon':18.383088}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':19.074762,'lon':79.487698}
pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':20.179065,'lon':79.001439}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':21.306421,'lon':78.789356}
pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':22.518024,'lon':77.829404}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':23.264139,'lon':77.429333}
pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':24.509723,'lon':78.330212}
```

```
pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':25.668840,'lon':78.451062}
pub(myData)

time.sleep(3)

myData = {'name':'Delhi Express','lat':26.177704,'lon':78.170910}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':27.505914,'lon':77.676526}

pub(myData)
time.sleep(3)

myData = {'name':'Delhi Express','lat':28.302041,'lon':77.308484}
pub(myData)

time.sleep(3)
client.commandCallback = myCommandCallback

client.disconnect()
```

Train 2:

```
import wiotp.sdk.device

import time
import random

myConfig = {
"identity" :{

"orgId":"ytluse",
"typeId":"train2",

"deviceId":"mysore"
},

"auth":{
"token":"-BCOX+hMk?*@xc@AV9"

}
}

def myCommandCallback(cmd):
```

```
print("Message received fromIBM 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':'Mysuru SF Express','lat':11.024938,'lon':76.982315} pub(myData)
time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':11.220325,'lon':77.570083} pub(myData)

time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':11.564960,'lon':77.993057} pub(myData)
time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':11.780142,'lon':78.037002} pub(myData)

time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.134824,'lon':78.130386} pub(myData)
time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.226105,'lon':78.091934} pub(myData)

time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.344187,'lon':78.037002} pub(myData)
time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.489034,'lon':78.009536} pub(myData)

time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.655239,'lon':77.866714} pub(myData)
time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.735622,'lon':77.756851} pub(myData)

time.sleep(3)
```

```
myData = {'name':'Mysuru SF Express','lat':12.907020,'lon':77.696426} pub(myData)

time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.987323,'lon':77.646988} pub(myData)
time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.955205,'lon':77.509659} pub(myData)

time.sleep(3)
myData = {'name':'Mysuru SF Express','lat':12.665958,'lon':77.136123

}
pub(myData)

time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.548022,'lon':76.921890} pub(myData)
time.sleep(3)

myData = {'name':'Mysuru SF Express','lat':12.336809,'lon':76.644485} pub(myData)

time.sleep(3)
client.commandCallback = myCommandCallback

client.disconnect()
```

Train 3:

```
import wiotp.sdk.device

import time
import random

myConfig = {
"identity" :{

"orgId":"ytluse",
"typeId":"2702",

"deviceId":"678910"

},
"auth":{

"token":"4skuC*xkhDH&la(CUR"
}
```

```
}
def myCommandCallback(cmd):

print("Message received fromIBM 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':'Goa Express','lat':11.688572, 'lon':78.098877}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':11.711433, 'lon':78.076905}
pub(myData)

time.sleep(3)
myData = {'name':'Goa Express','lat':11.978226,  'lon':78.116730}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':12.085676, 'lon': 78.119477}
pub(myData)

time.sleep(3)
myData = {'name':'Goa Express','lat':12.402400, 'lon':78.023347}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':12.884795, 'lon':77.707490}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':13.018630,'lon':77.614106}
pub(myData)
```

```
time.sleep(3)
myData = {'name':'Goa Express','lat':13.334194, 'lon':77.086762}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':13.299448, 'lon':76.858796}
pub(myData)

time.sleep(3)
myData = {'name':'Goa Express','lat':13.344884,'lon': 76.205109}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':13.619985, 'lon':75.966157}
pub(myData)

time.sleep(3)
myData = {'name':'Goa Express','lat':13.974739, 'lon':76.119965}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':14.423398, 'lon':75.949677}
pub(myData)

time.sleep(3)
myData = {'name':'Goa Express','lat':14.922914, 'lon':75.389374}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':15.119216, 'lon':75.389374}
pub(myData)

time.sleep(3)
myData = {'name':'Goa Express','lat':15.449980, 'lon':74.406230}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':15.352006,'lon':74.307353}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':15.314922, 'lon':74.218089}
pub(myData)
```

```
time.sleep(3)
myData = {'name':'Goa Express','lat':15.283131, 'lon':74.146678}

pub(myData)
time.sleep(3)

myData = {'name':'Goa Express','lat':15.276839, 'lon':74.129855}
pub(myData)

time.sleep(3)
time.sleep(3)

myData = {'name':'Goa Express','lat':15.282800, 'lon':74.125392}
pub(myData)

time.sleep(3)
time.sleep(3)

myData = {'name':'Goa Express','lat':15.296378,'lon':74.135692}
pub(myData)

time.sleep(3)
client.commandCallback = myCommandCallback

client.disconnect()
```

CHAPTER-8

TESTING

8.1.TEST CASES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement. A test case has components that describe input, action, and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on “HOW” to validate test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

TYPES OF TESTS

1. Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome

of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3. Functional test

   Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items: Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions: identified functions must be exercised. Output : identified classes of application outputs must be exercised. Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for 19 testing. Before functional testing is complete, additional tests are identified and the effective value of current test s is determined.

4. System Test

   System testing ensures that the entire integrated software system meets requirements . It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5. White Box Testing

   White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6. Black Box Testing

   Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7. Unit Testing:

   Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

8.2. User Acceptance Testing

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automati
1	Functional	Registration	Registration through the form by Filling in my details		1.Click on register 2.Fill the registration form 3.click Register		Registration form to be filled is to be displayed	Working as expected	Pass		
2	UI	Generating OTP	Generating the otp for further process		1.Generating of OTP number		user can register through phone numbers, Gmail, Facebook or other social sites and to get oto number	Working as expected	pass		
3	Functional	OTP verification	Verify user otp using mail		1.Enter gmail id and enter password 2.click submit	Username: abc@gmail.com password: Testing123	OTP verifed is to be displayed	Working as expected	pass		
4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter into log in page 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: abc@gmail password: Testing123	Application should show 'Incorrect email or password' validation message.	Working as expected	pass		
5	Functional	Display Train details	The user can view about the available train details		1.As a user, I can enter the start and destination to get the list of trains available connecting the above	Username: abc@gmail.com password: Testing12367868678687 6876	A user can view about the available trains to enter start and destination details	Working as expected	fail		

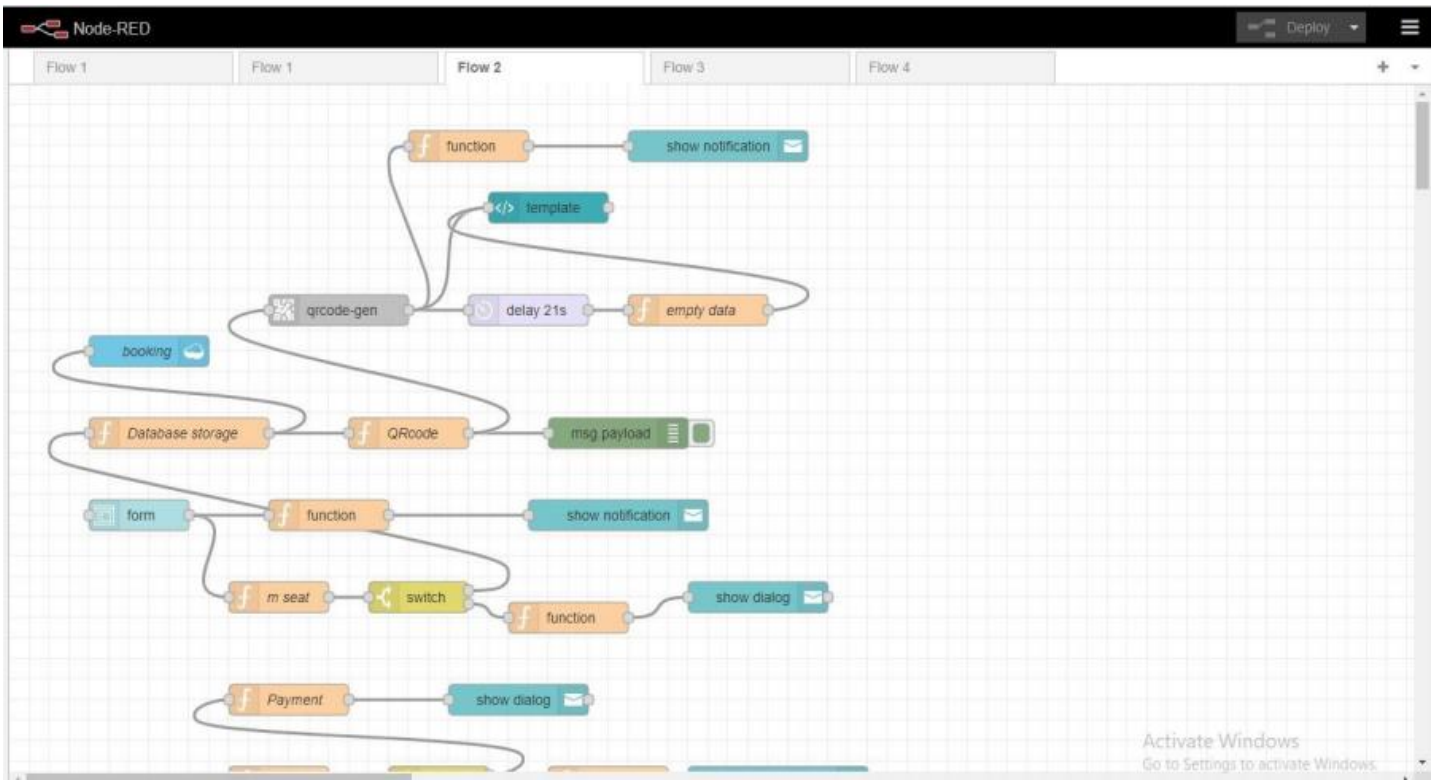
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements. Test Results: All the test cases mentioned above passed successfully. No defects encountered.

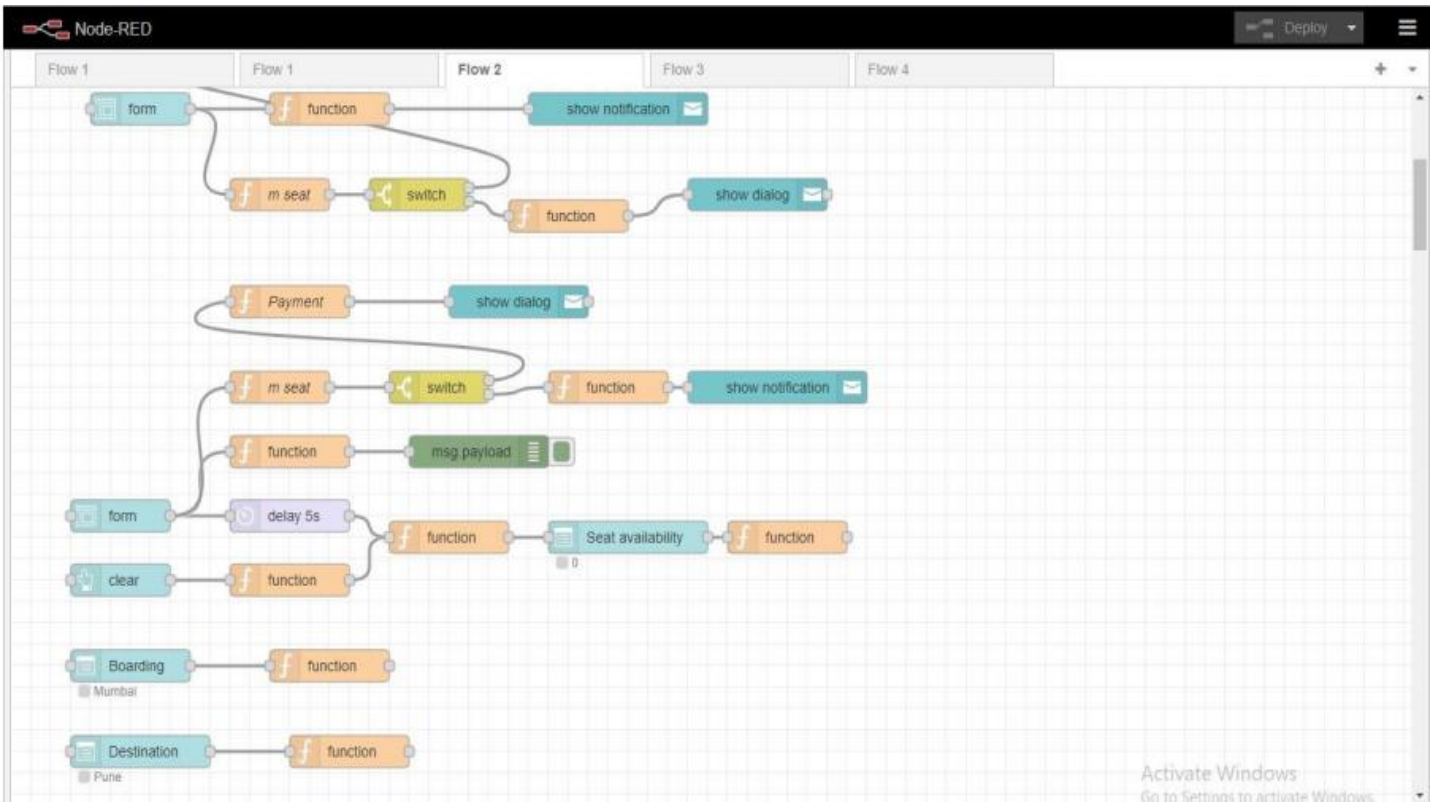
Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Statu s	Commnets	TC for Automation(Y/N	BUG ID
Functional	Booking	user can provide the basic details such as a name, age, gender etc		1.Enter method of reservation 2.Enter name,age,gender 3.Enter how many tickets wants to be booked 4.Also enter the number member's details like name,age,gender		Tickets booked to be displayed	Working as expected	Pass			
UI	Booking seats	User can choose the class, seat/berth. If a preferred seat/berth isn't available I can be allocated based on the availability		1,.known to which the seats are available		known to which the seats are available	Working as expected	pass			
Functional	Payment	user, I can choose to pay through credit Card/debit card/UPI.		1.user can choose payment method 2.pay using tht method		payment for the booked tickets to be done using payment method through either the following methods credit Card/debit card/UPI.	Working as expected	pass			
Functional	Redirectio n	user can be redirected to the selected		1.After payment the usre will be redirected to the previous		After payment the usre will be redirected to the previous page	Working as expected	pass			

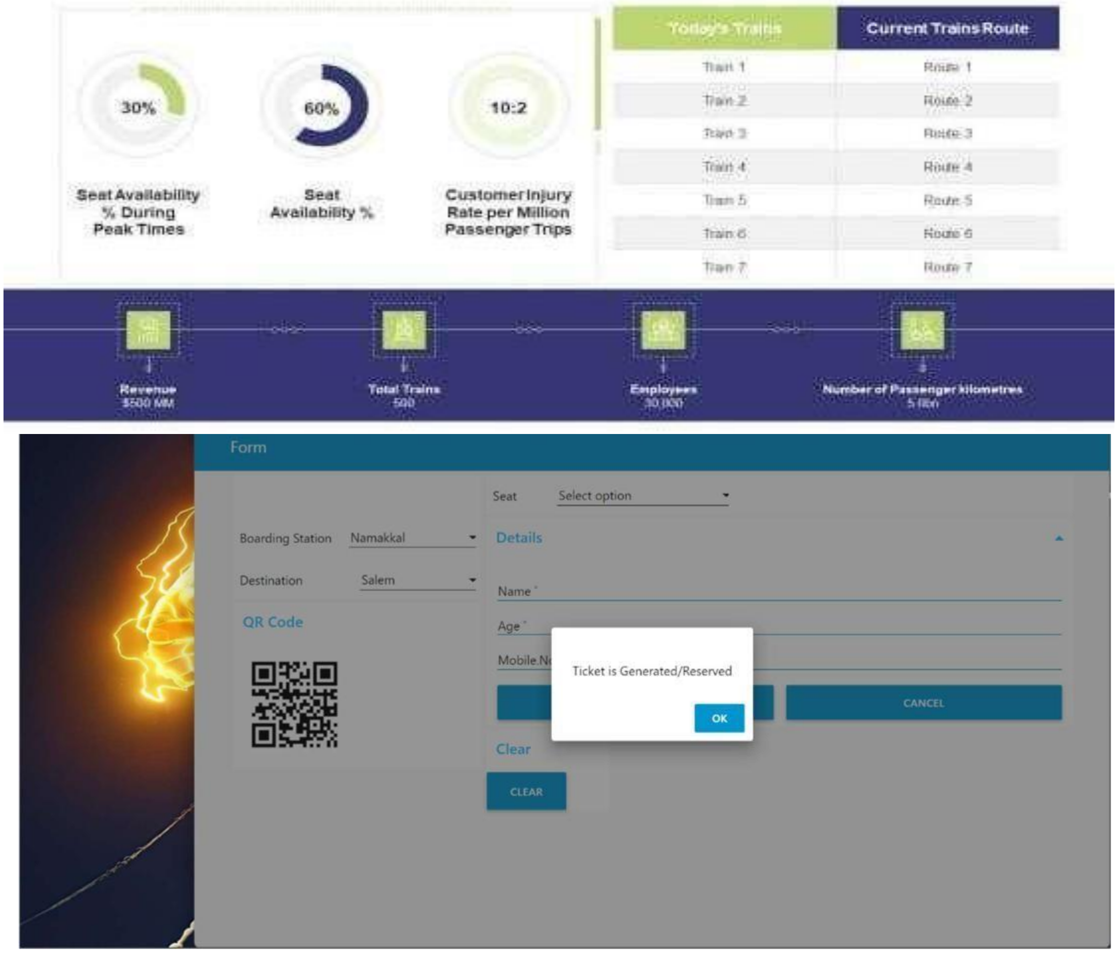
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisit	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Autom	BUG ID
10	Functional	Ticket generation	a user can download the generated e ticket for my journey along with the QR code which is used for authentication during my journey.		1.Enter method of reservation 2.Enter name,age,gender 3.Enter how many tickets wants to be booked 4.Also enter the number member's details like name,age,gender		Tickets booked to be displayed	Working as expected	Pass			
11	UI	Ticket status	a usercan see the status of my ticket Whether it's confirmed/waiting/RAC		1.known to the status of the tivkets booked		known to the status of the tivkets booked	Working as expected	pass			
12	Functional	Remainder notification	a user, I get remainders about my journey A day before my actual journey		1.user can get reminder nofication		user can get reminder nofication	Working as expected	pass			
13	Functional	GPS tracking	user can track the train using GPS and can get information such as ETA, Current stop and delay		1.tracking train for getting information		tracking process through GPS	Working as expected	pass			

CHAPTER 9
RESULTS

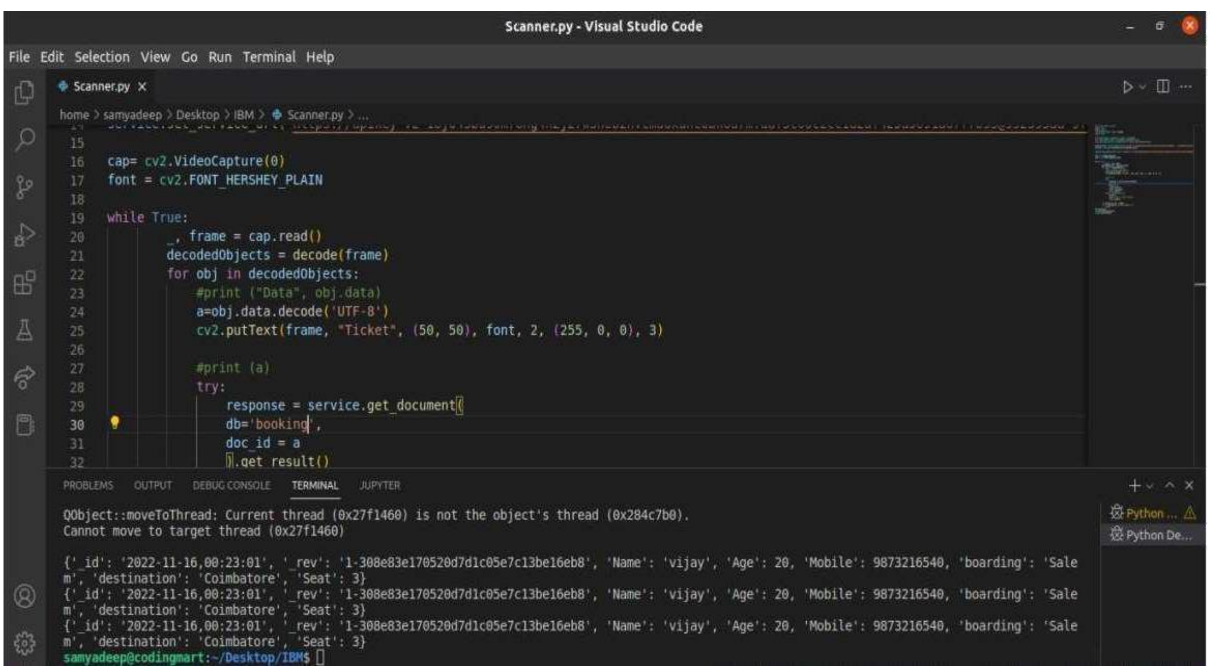
9.1. PERFORMANCE METRICS



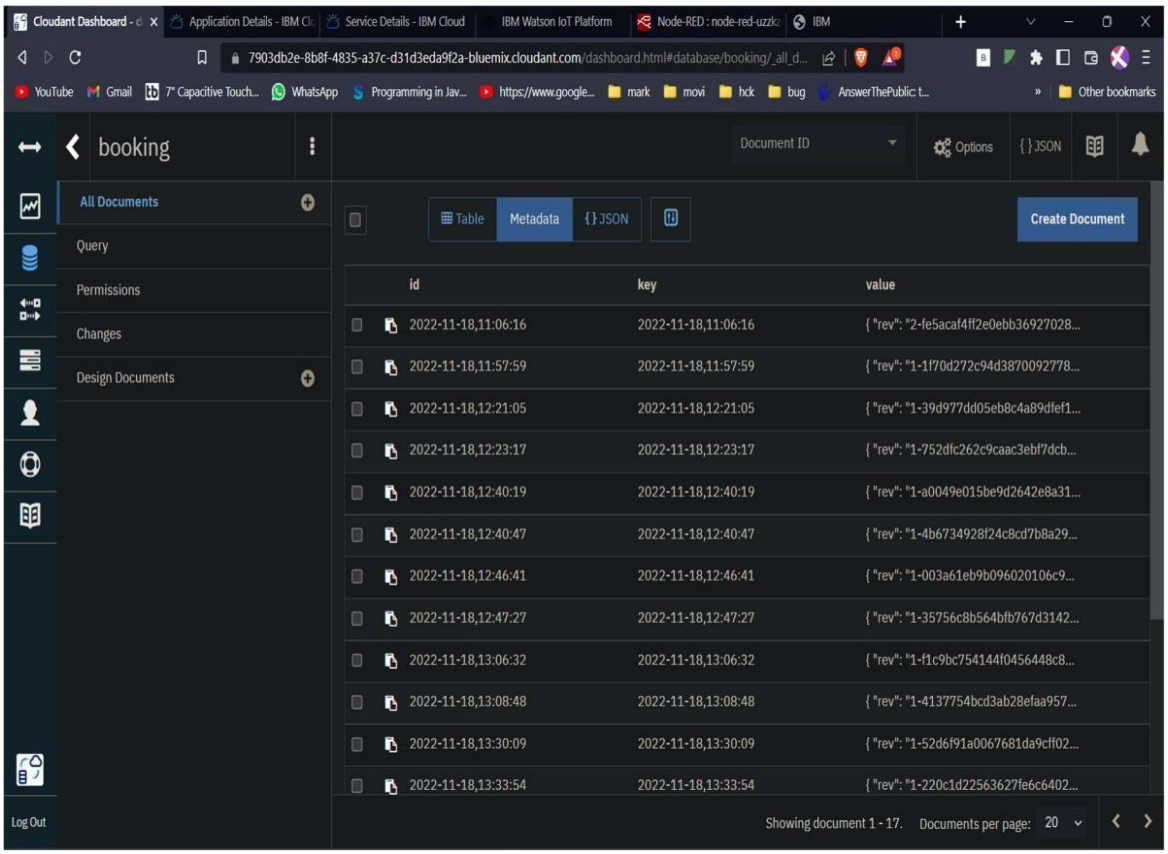




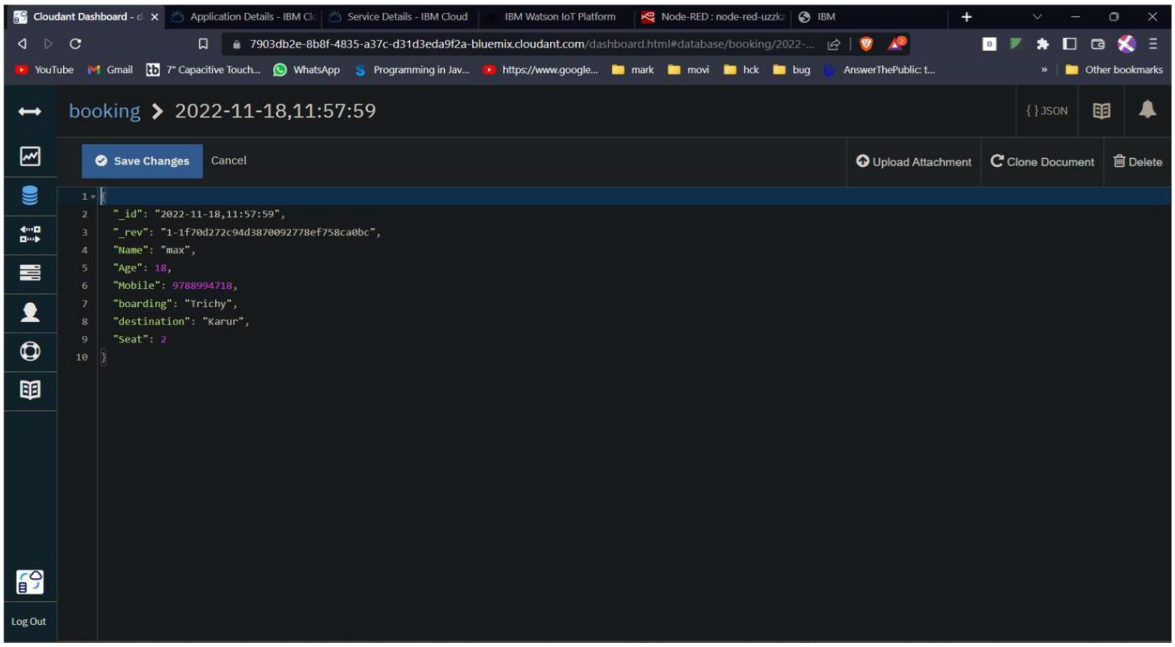
Ticket generated



QR code verified

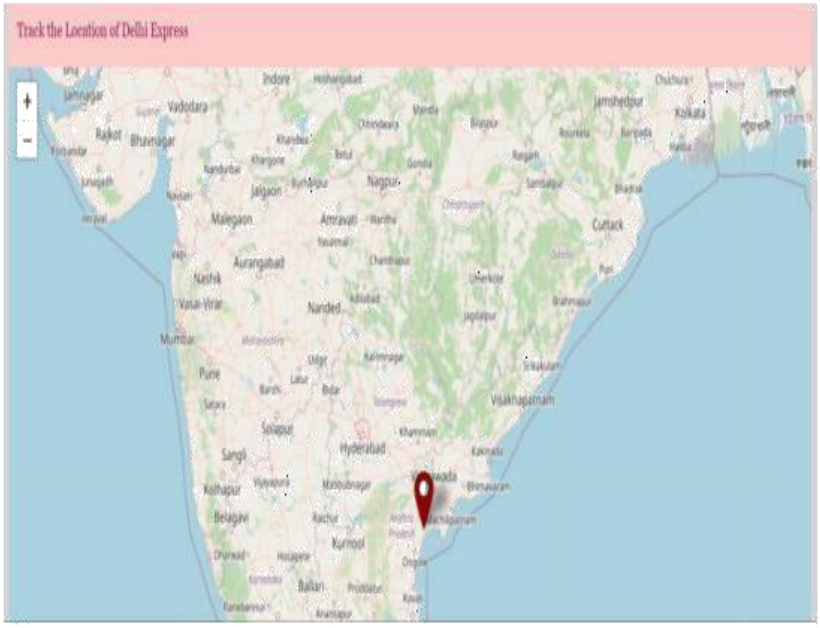


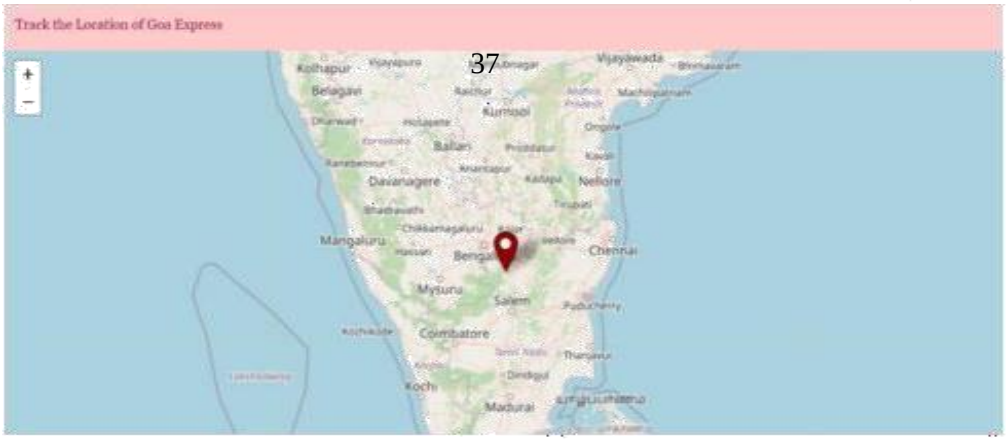
Cloud database

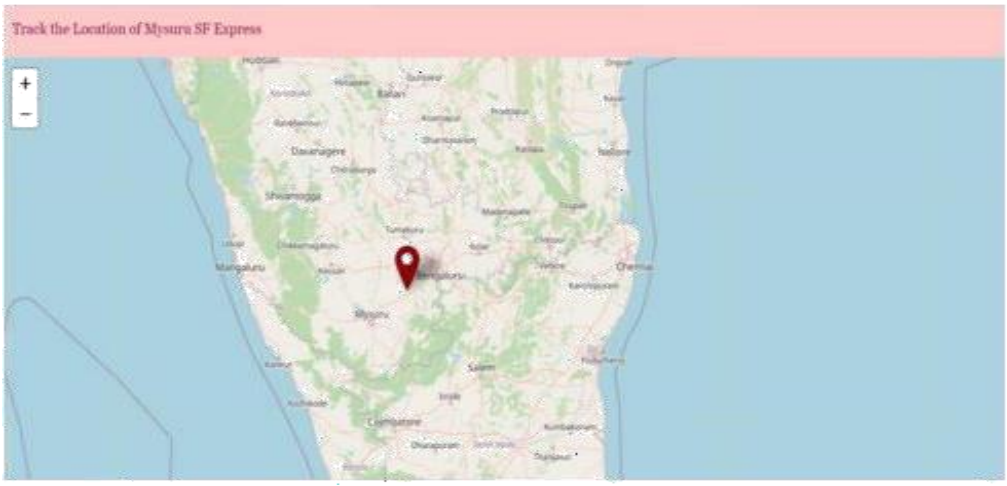


Ticket confirmation

Train Tracking:







CHAPTER 10

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

CHAPTER 11

CONCLUSION

Online railway ticket reservation system was successfully designed and developed as per the specifications. We have created a project to make the user and the worker to reduce the time consumption for the issue and buy a ticket. It is also reducing the fear of lost the ticket. from this project we will generate the qrcode it is more helpful to have all details in the qrcode itself. While ticket checker scan the qrcode,it will show all detail about the passenger and the details of the ticket .

At present the current framework is physically and human controlled framework once the train leaves the station. The station ace advises the guard about the appearance regarding the train through the phone. When the watchman gets the data then he shuts the entryway relying upon the planning at which the train shows up. Thus, if the train is late because of specific reasons, at that point entryway stay shut for quite a while causing traffic close to the doors. There is no unified framework is accessible by and by signals are control by mean of interlocking and wrong signals and sign gadget which is self-loader framework. The programmed railroad entryway control at the level intersection and hostile to crash gadget. The ideal opportunity for which it is shut is less contrasted with the physically worked entryways and furthermore lessens the human work. This kind of entryways can be utilized in an unmanned level intersection where the odds of mishaps are higher and solid activity is required. Since the activity is programmed mistake because of manual activity is forestalled. Also, executing the work railroad framework can be brought together which can control the train crash mishaps. Another methodology for improving wellbeing at LCs and train crash on IR has been proposed. Organizations have been given to keep up records of LC inventories mishap/episode reports. A normal appraisal of wellbeing execution ought to be finished. This methodology ought to have the option to cut down the rising pattern in mishaps at LCs and train impact mishap. This undertaking utilizes the

current framework of railroads for example present flagging strategy and meets all the necessities to have a programmed controlling of the railroad traffic. It gives the management and control framework give the intend to constant investigation survey and information assortment for the reason for upkeep on the versatile and fixed offices for the assurance of activity wellbeing and support 60 effectiveness just as the security examination dynamic framework dependent on the portion of wellbeing information. The extraordinary accomplishment of present-day advancements in each important field and the mechanical improvement of the railroad business itself have furnished rail route with practicality to win higher help quality and quicker speed.

CHAPTER 12

FUTURE SCOPE

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

CHAPTER 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_entry = ("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(", 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 42 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-
betweenstations?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

 last_name = models.CharField( verbose_name="Last name", max_length=40 )
 city = models.CharField( verbose_name="City", max_length=40 )
 stripe_id = models.CharField( 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, 46 "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)
# fxn call for berth type

 s = 7 berth_type(s)   # fxn call for berth type

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

                    if "no" in question:
                   Speak("We will automatically start after 5 Mins Sir.”)

                    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  mins”,
                       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 as you have”
                     "been continuously using it for 45 mins and it will
                     affect your eyes",
                                  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:
                    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 NAME")
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\n\n\n\t\t\t\t\t\tLOADI 63 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\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" 64 print "WRONG PASSWORD".center(80)
 elif ch==2:
                fin=open("tr1details.dat",'rb')
if not fin:
                print "ERROR"
else:           try:
while True:
              print"*"*80
print"\t\t\t\tTRAIN DETAILS"
                         print"*"*80
print                                tr=load(fin)
tr.output()


                            raw_input("PRESS ENTER TO VIEW NEXT TRAIN DETAILS")
                             os.system('cls') except
EOFError:
          Pass
 elif ch==3:
print'='*80
             print "\t\t\t\tRESERVATION OF TICKETS"
 print'='*80          print           tick.reservation()
elif ch==4:
                print"="*80
                print"\t\t\t\tCANCELLATION OF TICKETS"
print                    print"="*80                  print
tick.cancellation()                elif ch==5:
                print "="*80
print("PNR STATUS".center(80))


              print"="*80
 printclass 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:

           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 NAME ")
                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\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\tENT ER 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=”””\<html>

    <body>

   <p>Hi,</p>

   How are yoy?<br>

  <a href=http://www.realpython.com>Real Python</a>

  Has many 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")

Git hub link

https://github.com/IBM-EPBL/IBM-Project-49024-1660815222