Project Report

1.INTRODUCTION

1.1 PROJECT OVERVIEW

SMART SOLUTIONS FOR RAILWAYS is to manage Indian Railways is the largest railway network in Asia and additionally world's second largest network operated underneath a single management. Due to its large size, it is difficult to monitor the cracks in tracks manually. This paper deals with this problem and detects cracks in tracks with the help of ultrasonic sensor attached to moving assembly with help of stepper motor. Ultrasonic sensor allows the device to moves back and forth across the track and if there is any fault, it gives information to the cloud server through which railway department is informed on time about cracks and many lives can be saved. This is the application of IoT, due to this it is cost effective system. This effective methodology of continuous observation and assessment of rail tracks might facilitate to stop accidents. This methodology endlessly monitors the rail stress, evaluate the results and provide the rail break alerts such as potential buckling conditions, bending of rails and wheel impact load detection to the concerned authorities.

1.2 PURPOSE

Internet is basically system of interconnected computers through network. But now its use is changing with changing world and it is not just confined to emails or web browsing. Today's internet also deals with embedded
sensors and has led to development of smart homes, smart rural area, e-health care's etc. and this introduced the concept of IoT . Internet of Things refers to interconnection or communication between two or more devices without human-tohuman and humanto-computer interaction. Connected devices are equipped with sensors or actuators perceive their surroundings. IOT has four major components which include sensing the device, accessing the device, processing the information of the device, and provides application and services. In addition to this it also provides security and privacy of data . Automation has affected every aspect of our daily lives. More improvements are being

introduced in almost all fields to reduce human effort and save time. Thinking of the same is trying to introduce automation in the field of track testing. Railroad track is an integral part of any company's asset base, since it provides them with the necessary business functionality. Problems that occur due to problems in railroads need to be overcome. The latest method used by the Indian railroad is the tracking of the train track which requires a lot of manpower and is time-consuming

2.LITERATURE SURVEY

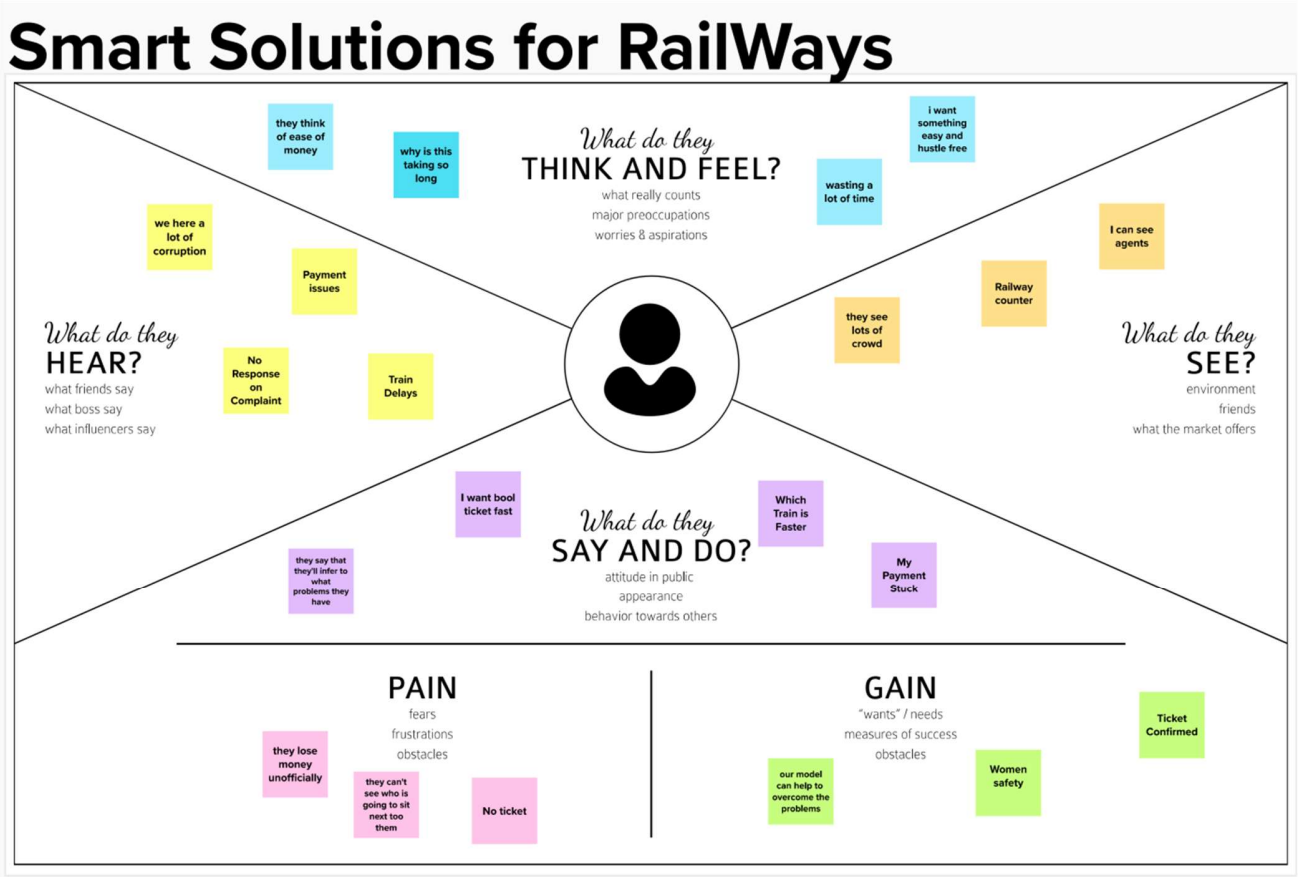
PAPER NAME	AUTHOR	YEAR	METHODOLOGY	MERITS	DEMERITS
Passenger Monitoring Model for easily Accessible Public City Trams/Trains.	**Roman Khoeblal, Teeravisit Laohapensaeng, Roungsan Chaisricharoen**	**2015**	**Passenger monitoring, passenger control RFID distance reading, ticket control, RFID ticket inspection.**	**It is possible to travel cross country with a single public transportation card, using transport systems of several transport operators.**	**Applicable only for passenger monitoring**
Application of smart computing in Indian Railway Systems.	**Parag Chatterjee, Asoke Nath**	**2014**	**By Interlinking unique identification system with train ticket reservation system by using video surveillance, rail sensors, biometric input devices and multimedia displays**	**Reduces manual effort in passenger data entry.** **Provides security verification**	**Significant investment is needed** **Risk of database.**

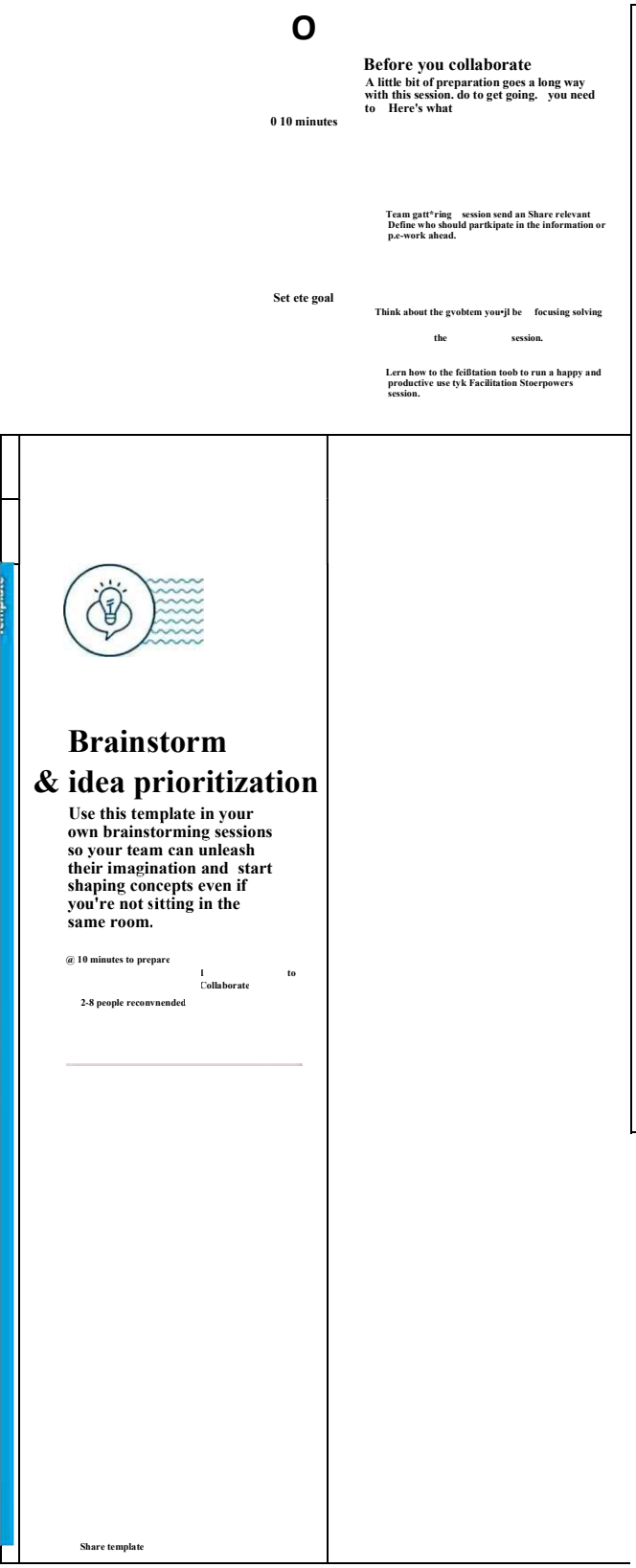
Android Suburban Railway Ticketing with GPS as Ticket Checker.	**Sana Khoja, Maithili Kadam**	**2012**	**Android, SQ lite, Cloud Database, ASR, QR Code.**	**E-Ticket facility, enabling reuse and replacement of components.**	**QR Codes before the user enters or leaves the station, where the user can have access which**
					is risk in ticket booking.
Novel Approach for Smart Indian Railways.	**Sujith Kumar, K.M.Yathe endra Parvan, V.Sumathy , Thejeswari C.K**	**2017**	**Digitalization, Smart Railways, Aadhar Card, Smartphone, Identity Verification.**	**Employ a mobile application through which passengers can access various ticketing options in user friendly and efficient manner.**	**Biometric database is risk of hacking.**

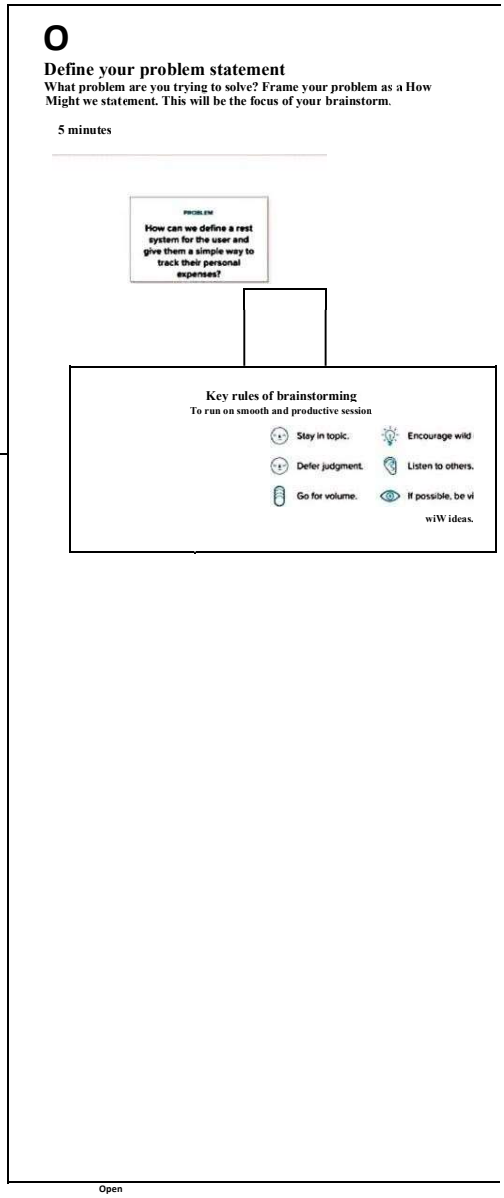
A Review on IOT based automated seat allocation and verification using QR code.	**Sarvath Saba, Sharon Philip, Shriharsha, Mukund Naik, Sudeep Sherry**	**2022**	**The system lets the passenger to have a comfortable journey by checking the temperature first for normal and then the count for avoid crowd using the QR Code.**	**This model proposes a radical change in train operation and passenger experience. One of the many steps towards a more digitized society as a part of the "Digital India" movement proposed in 2015 by the Prime Minister.**	**The system is not fool-proof and requires a dramatic change in the existing system in terms of the people allowed on platforms, etc. but baby steps matter.**

3. IDEATION AND PROPOSED SOLUTON

3.1 EMPATHY MAP CANVAS



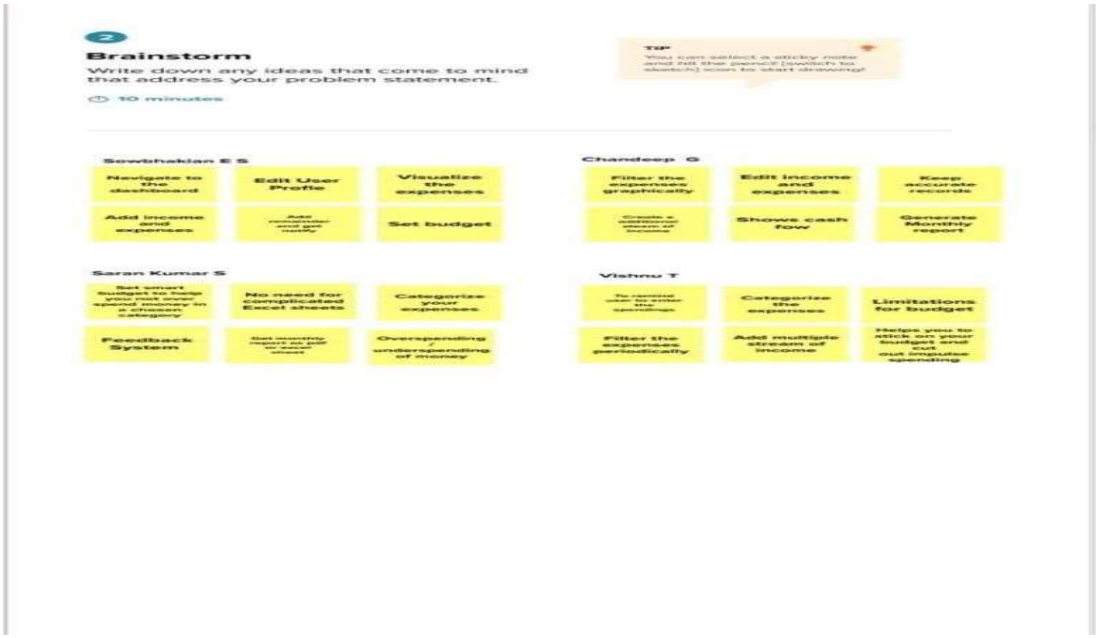




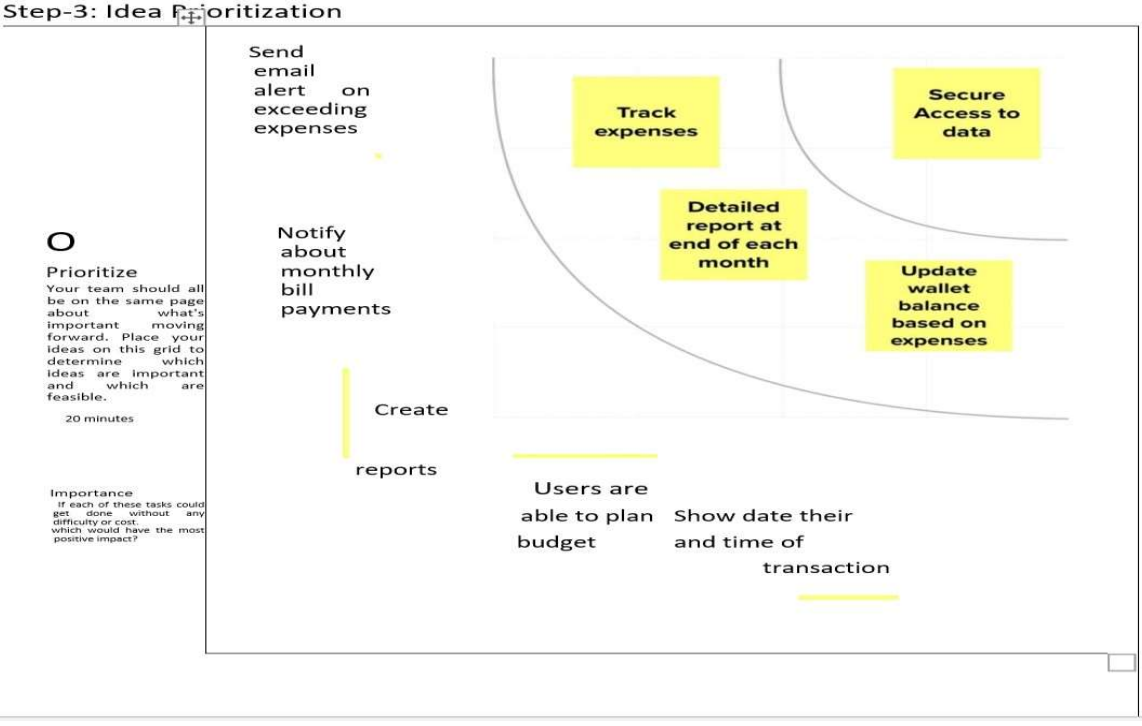
3.2 IDEATION & BRAINSTORMING

Step1:

Step 2:



Step-3: Idea Prioritization



3.3 PROPOSED SOLUTION:

S.NO	Parameters	Description1
1.	Problem statement (problem to be solved)	• To design a webpage where public can view and book tickets and to enable proper less ticket verification • To track the live location of all the trains • To increase smart facilities in railways to ensure passenger safety and comfort
2.	Idea/solution description	• GPS tracker is placed in the train so that the passengers can track the location of the train even it is delayed. • Passengers can book their tickets using the website which is possible at anytime, anywhere. • Smart ticketing to avail seasons so that physical work is eradicated.
3.	novelty/uniqueness	• Automated waiting list clearance
		• Health monitoring to loco pilot • Qr based entry and exit into stations

4.	**Social Impact / Customer Satisfaction**	• **No Queuing to get tickets and burdenless because of e-tickets.** • **Elimination of dilemma whether the train has left or yet to arrive.** • **Can get the status and avail of eseasons** • **instead of visiting the station physically every time.**
5.	**Business model (Revenue model)**	**Transaction Revenue Model**
6.	**Scalability of the solution**	**The booking and tracking software can support a large number of customers The automations can be implemented in a large scale**

3.4 PROBLEM SOLUTION FIT

1.Customer Segment The passengers travelling in the train	6. Customer Limitations • Health Concern • Safety and Comfort • Timing Concerns	5. Available Solution Emergency train stopping Location updation in stations
2. Problems/Pains • Existing ticket checking methods must be made contactless • The train location tracking must be made more accurate • More automations can be brought in trains	9. Problem Root Cause The investment in improving railway sectors is less and also research in this area is limited	7. Behaviour Directly related: The comfort and safety of people. Saves a lot of waiting time Indirectly related: Reduces the manpower involved and makes railways computer based

3. Triggers to Act 10. Your Solution 8. Channels

Seeing people without tickets. Making people aware of the best of automation

- To enable QR based ticket verification
- To track and update the live location of all the trains using GPS module
- To increase smart facilities in railways

During their journey in the train

4. Emotions

Before: Frustration, Unsatisfied

After: Happy, feeling safe and secure

4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENTS

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

4.2. NON-FUNCTIONAL REQUIREMENTS

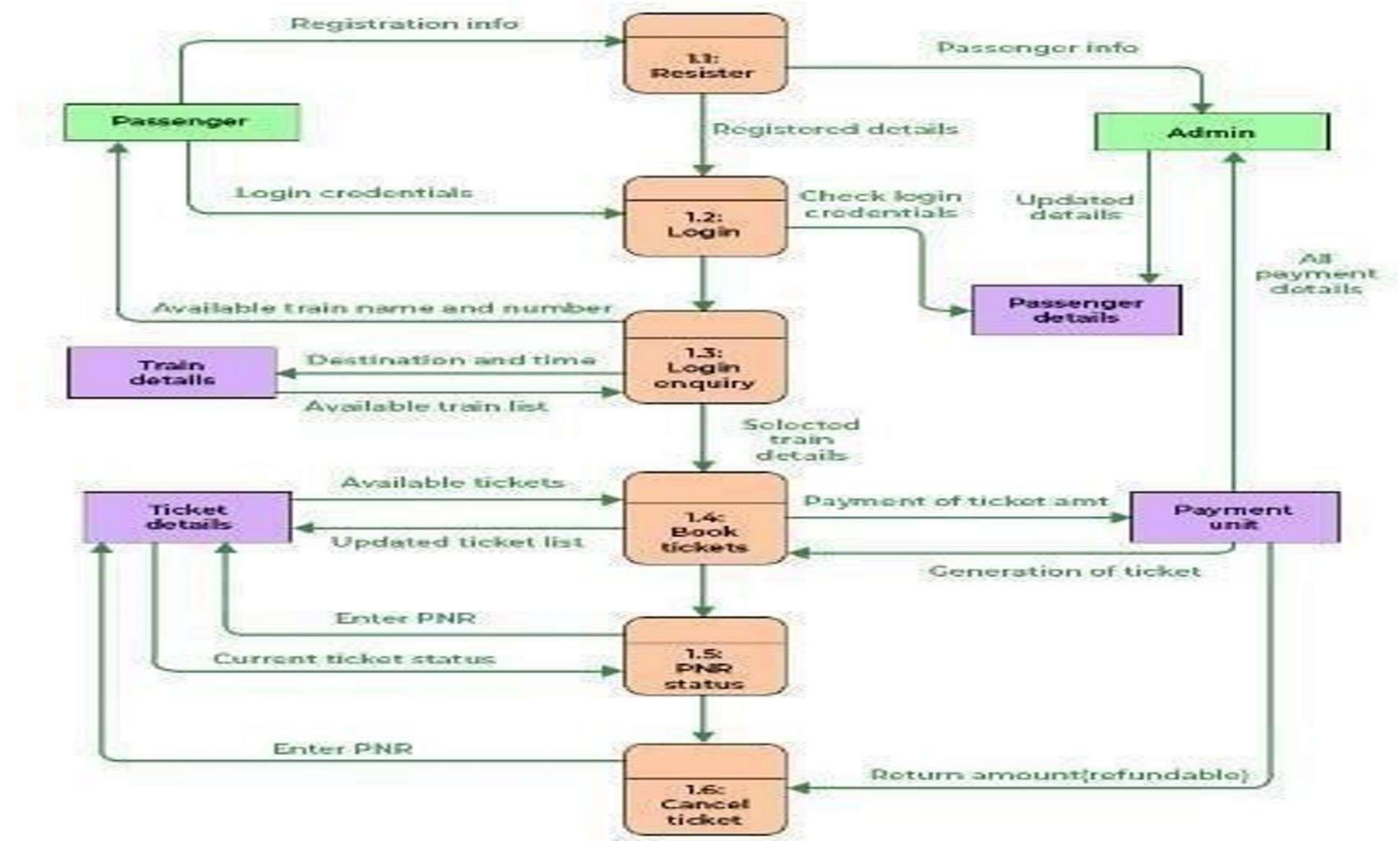
FR No.	Non-Functional Requirement	Description

NFR-1	**Usability**	**Within periodic maintenance, we can detect cracks in the railway track. which will be highly usable on remote railway tracks.**
NFR-2	**Security**	**Accidents and property damage can be prevented with the help of our smart sensors which immediately send the fault to the pilot and administration.**
NFR-3	**Reliability**	**Traffic lights and signalling can be made accurately with the help of sensors. so it is more reliable.**
NFR-4	**Performance**	**Communication plays a vital role in transferring the crack-detected signal to the responsible authority so that they can take appropriate measures within a short span.**
NFR-5	**Availability**	**Our idea is to make the crack alert to all the trains passing through that fault-prone area.**

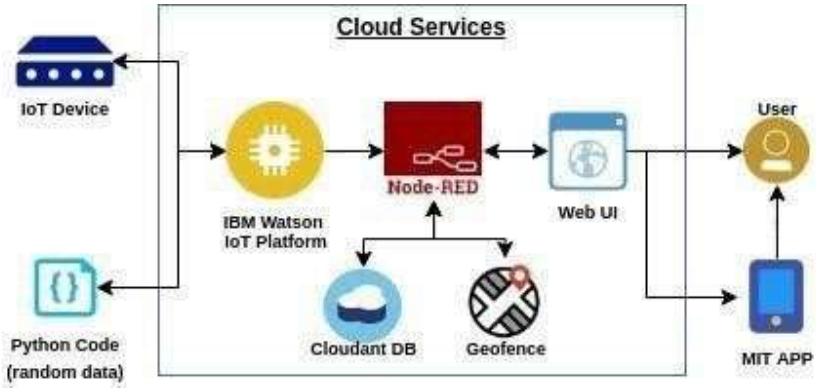
NFR-6	Scalability	Our project is based on IoT & cloud, which makes the pilot and authority updated every single sec. Adhoc is easy to handle.

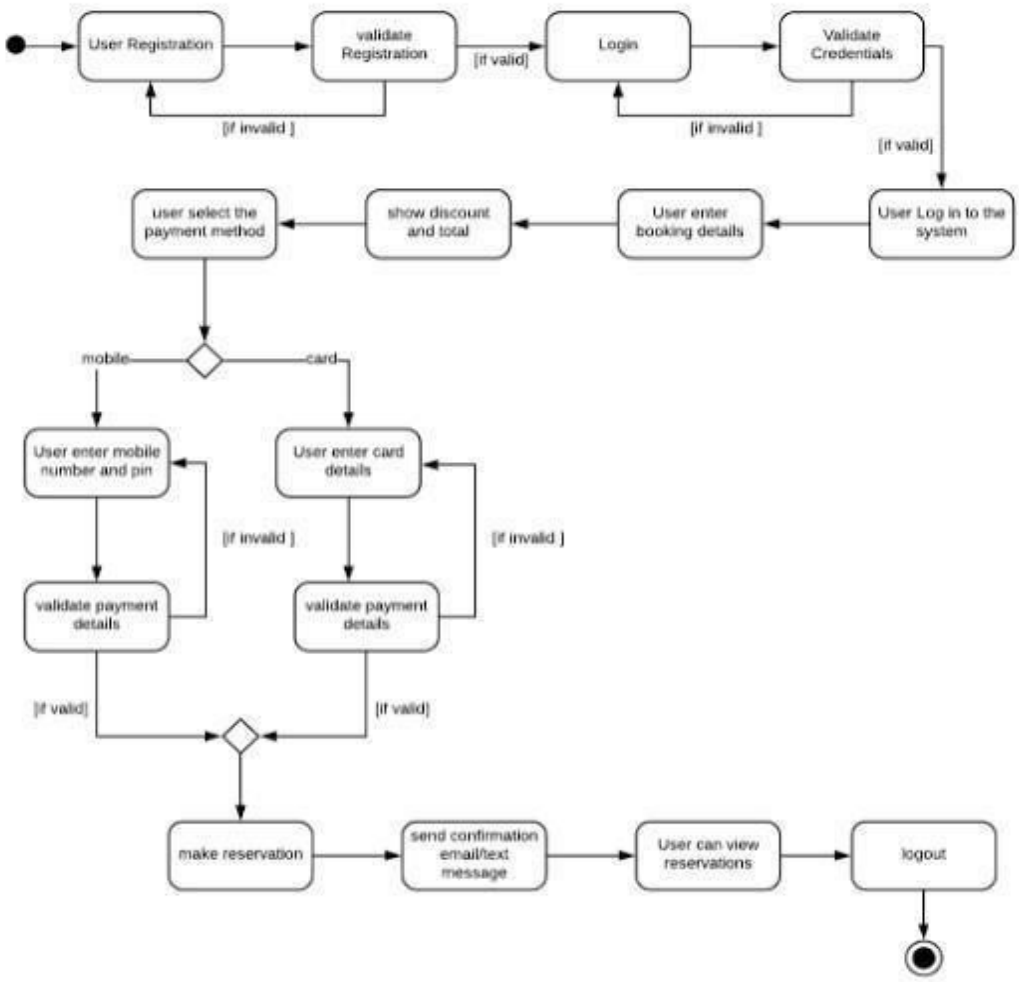
5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



5.2 SOLUTION & TECHNICAL ARCHITECTURE


Cloud Services
IoT Device
IBM Watson
IoT Platform
Node-RED
Web UI
User
Python Code
(random data)
Cloudant DB
Geofence
MIT APP



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
Customer (Mobile user)	Reserving ticket	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard
Customer (Mobile user)	Reserving ticket	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm

Customer (Mobile user)	Reserving ticket	USN-3	As a user, I can register for the application and enter the details for reserving the ticket	I can register & access the dashboard
				with Facebook Login
Customer (Mobile user)	Dashboard	Users	The details will be stored safely	I can access it using database
Customer (Web user)	Reserving ticket	user	Enter the details and click submit button to book ticket	I can use the QR code which is been generated
Customer Care Executive	Connecting the service provider	customer	Connects with the service by logging in	Can get connected with the server
Administrator	Provides the services	admin	The data is given by the user	Can add or update the data provided by the user

6. PROJECT PLANNING AND SCHEDULING

6.1. SPRINT PLANNING& ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register through the form by Filling in my details	2	High	Nivetha
Sprint-1		USN-2	As a user, I can register through phone numbers, Gmail, Facebook or other social sites	1	High	snekha
Sprint-1	Conformation	USN-3	As a user, I will receive confirmation through email or OTP once registration is successful	2	Low	archana
Sprint-1	login	USN-4	As a user, I can login via login id and password or through OTP received on register phone number	2	Medium	gowri
Sprint-1	Display Train details	USN-5	As a user, I can enter the start and destination to get the list of trains available connecting the above	1	High	Nivetha
Sprint-2	Booking	USN-6	As a use, I can provide the basic details such as a name, age, gender etc…	2	High	snekha
Sprint-2		USN-7	As a user, I can choose the class, seat/berth. If a preferred seat/berth isn't available I can be allocated based on the availability	1	Low	archana
Sprint-2	Payment	USN-8	As a user, I can choose to pay through credit Card/debit card/UPI.	1	High	gowri

Sprint-2		USN-9	As a user, I will be redirected to the selected	2	High	nivetha

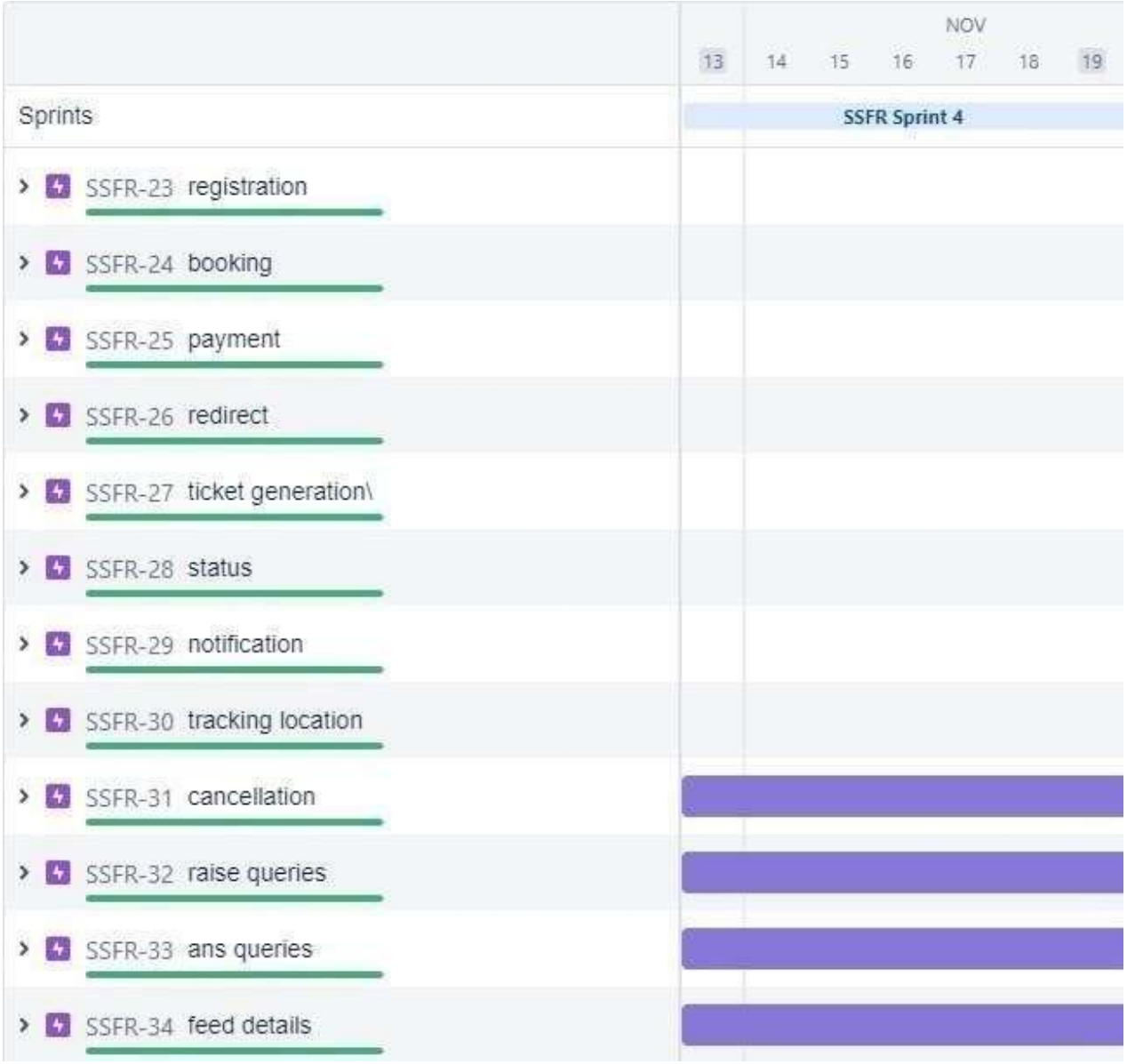
Sprint-3	Ticket generation	USN-10	As a user, I can download the generated e- ticket for my journey along with the QR code which is used for authentication during my journey.	1	High	gowri
Sprint-3	Ticket status	USN-11	As a user, I can see the status of my ticket	2	High	archana
			Whether it's confirmed/waiting/RAC.			
Sprint-3	Remainders notification	USN-12	As a user, I get remainders about my journey A day before my actual journey.	1	High	snekha
Sprint-3	Ticket cancellation	USN-13	As a user, I can track the train using GPS and can get information such as ETA, Current stop and delay	2	High	nivetha
Sprint-4		USN-14	As a user, I can cancel my tickets if there's any Change of plan	1	High	archana
Sprint-4	Raise queries	USN-15	As a user, I can raise queries through the query box or via mail.	2	Medium	gowri
Sprint-4	Answer the queries	USN-16	As a user, I will answer the questions/doubts Raised by the customers.	2	High	nivetha
Sprint-4	Feed details	USN-17	As a user, I will feed information about the trains delays and add extra seats if a new compartment is added.	1	High	snekha

6.2. SPRINT DELIVERY SCHEDULE

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

6.3. REPORTS FROM JIRA


NOV
31
1
2
3
4
5
Sprints
SSFR Sprint 2
SSFR-23 registration
SSFR-24 booking
SSFR-25 payment
SSFR-26 redirect


NOV
13
14
15
16
17
18
19
Sprints
SSFR Sprint 4
SSFR-23 registration
SSFR-24 booking
SSFR-25 payment
SSFR-26 redirect
SSFR-27 ticket generation\
SSFR-28 status
SSFR-29 notification
SSFR-30 tracking location
SSFR-31 cancellation
SSFR-32 raise queries
SSFR-33 ans queries
SSFR-34 feed details

7.CODING AND SOLUTIONING

7.1. FEATURE 1

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

7.2. FEATURE 2

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

7.3. DATABASE SCHEMA

```
labl_0 = Label(base, text="Registration form",width=20,font=("bold",
20))    labl_0.place(x=90,y=53)


lb1= Label(base, text="Enter Name", width=10, font=("arial",12))
lb1.place(x=20, y=120)   en1= Entry(base)     en1.place(x=200, y=120)

lb3= Label(base, text="Enter Email", width=10, font=("arial",12))
lb3.place(x=19, y=160)   en3= Entry(base)     en3.place(x=200, y=160)

lb4= Label(base, text="Contact Number", width=13,font=("arial",12))
lb4.place(x=19, y=200)   en4= Entry(base)     en4.place(x=200, y=200)

lb5= Label(base, text="Select Gender", width=15, font=("arial",12))
lb5.place(x=5, y=240)     var = IntVar()

Radiobutton(base, text="Male", padx=5,variable=var,
value=1).place(x=180, y=240)

Radiobutton(base, text="Female", padx =10,variable=var,
value=2).place(x=240,y=240)

Radiobutton(base, text="others", padx=15, variable=var,
value=3).place(x=310,y=240)

list_of_cntry = ("United States", "India", "Nepal", "Germany")   cv =
```

```
StringVar()   drplist= OptionMenu(base, cv, *list_of_cntry)
drplist.config(width=15)   cv.set("United States")   lb2= Label(base,
text="Select Country", width=13,font=("arial",12))
lb2.place(x=14,y=280)
drplist.place(x=200, y=275)

lb6= Label(base, text="Enter Password", width=13,font=("arial",12))
lb6.place(x=19, y=320)   en6= Entry(base, show='*')     en6.place(x=200,
y=320)

lb7= Label(base, text="Re-Enter Password",
width=15,font=("arial",12))    lb7.place(x=21,
y=360)   en7 =Entry(base, show='*')
en7.place(x=200, y=360)

Button(base, text="Register", width=10).place(x=200,y=400)
base.mainloop()




def generateOTP() :

   # Declare a digits variable
# which stores all digits     digits
= "0123456789"
   OTP = ""

  # length of password can be changed    #
by changing value in range     for i in
range(4) :
      OTP += digits[math.floor(random.random() * 10)]
```

```
    return OTP

# Driver code if __name__ == "__main__"
:
    print("OTP of 4 digits:", generateOTP())

digits="0123456789" OTP="" for i in
range(6):
    OTP+=digits[math.floor(random.random()*10)] otp
= OTP + " is your OTP" msg= otp s = smtplib.SMTP('smtp.gmail.com',
587)
s.starttls()
s.login("Your Gmail Account", "You app password") emailid
= input("Enter your email: ")
s.sendmail('&&&&&&&&&&&',emailid,msg
) a = input("Enter Your OTP >>: ") if a == OTP:      print("Verified")
else:
    print("Please Check your OTP again") roo
```

8.TESTING

8.1.TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Pre-requisite	Steps to Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation	BUG	Executed By
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				Nikhila
2	UI	Generation 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	Working as expected	Pass				Preethiha
3	Functional	OTP verification	Varify user otp using mail		1.Enter gmail id and enter password 2.Click submit	Username: abc@gmail.com Password: Testing123	OTP verified this to be displayed	Working as expected	Pass				Kishokkumar
4	Functional	Login page	Verify user is able to log into application within Valid credentials		1.Enter into login page 2. Click on My Account dropdown button 3. Enter invalid user name/email text box 4.Enter valid password in password and text box 5.Click on login button	Username: abc@gmail.com Password: Testing123	Application should show incorrect email or password validation message	Working as expected	Pass				Raguram

Test case ID	Feature Type	Component	Test Scenario	Pre-requisite	Steps to Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation	BUG	Executed By
5	Functional	Display Train details	The user can view about the available train details		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: 123678686786876	A user can view about the available trains to enter start and destination details	Working as expected	Fail				Nikhila
6	Functional	Booking	user can provide the basic details such as a name, age, gender, etc.,		1. Enter method of reservation 2. Enter name, age, sender 3. Enter how many tickets want to be booked 4. Also enter the number members details like		Tickets booked to be displayed	Working as expected	Pass				Kishokkumar
7	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 or available		known to the status of the tickets booked	Working as expected	Pass				Preethiha
			user, I can choose to pay through credit Card/debit card/UPI.		1. User can choose payment method 2. Pay using the method		payment for the booked tickets to be done using payment method	Working as					

Test case ID	Feature Type	Component	Test Scenario	Pre-requisite	Steps to Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation	BUG	Executed By
11	UI	Ticket status	a uercan see the status of my ticket whether it's confirmed/waiting/RAC.		1. Known to the status of the tickets booked		known to the status of the tickets booked	Working as expected	Pass				Preethiha
12	Functional	Remainder notification	a User, I get remainders about my journey A day before my actual journey.		1. User can get reminder notification		user can get reminder notication	Working as expected	Pass				Kishokkumar
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				Raguram
14	Functional	Tickect cancelling	user can cancel my tickets		1. Tickets to be cancelled		Tickets booked to be concelled	Working as expected	Pass				Nikhila
15	UI	Raise queries	user can raise queries through the query box or via.		1. Raise the queries		raise the queries	Working as expected	Pass				Preethiha
16	Functional	Answer the queries	user will answer the questions/doubts Raised by the customers.		1. Answer the queries		answer the queries	Working as expected	Pass				Kishokkumar
17	Functional	Feed details	a user will feed information about the trains delays		1. Information feeding on trains		information feeding on trains	Working as expected	Pass				Raguram

Test case ID	Feature Type	Component	Test Scenario	Pre-requisite	Steps to Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation	BUG	Executed By
8	Functional	Payment	user, I can choose to pay through credit Card/debit card/UPI.		1. User can choose payment method 2. Pay using the 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				Raguram
9	Functional	Redirection	user can be redirected to the selected.		1. After payment the user will be redirected to the previous		After payment the usre will be redirected to the previous page	Working as expected	Pass				Kishokkumar
10	Functional	Ticket generation	A user can downloaded the generated e-ticket for my journey along with the QR code which is used for authentication during my		1. Enter method of reservation 2. Enter name, age, sender 3. Enter how many tickets want to be booked 4. Also enter the number members details like		Tickets booked to be displayed	Working as expected	Pass				Nikhila
11	UI	Ticket status	a uercan see the status of my ticket whether it's		1. Known to the status of the tickets booked		known to the status of the tickets booked	Working as	Pass				Preethiha

9.RESULTS

9.1.PERFORMANCE METRICS

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; o 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.

11.CONCLUSION

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

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 broken

13. APPENDIX

13.1.SOURCE PROGRAM

```
import math, random
import os                    import
smtplib                  import sqlite3
import
requests
from bs4
import
BeautifulSoup
from
```

```
django.contrib.auth.base_user import AbstractBaseUser
from django.db import models
            import logging
import pandas as pd
import pyttsx3
            from plyer import notification
import time              import   numpy as np
import matplotlib.pyplot
as plt              from PIL import Image,
ImageDraw              from
pickle import load,dump
            import smtplib, ssl
            from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import email

           from email import encoders
           from email.mime.base import MIMEBase



             import attr
             from flask import Blueprint, flash, redirect, request,
url_for                 from flask.views import MethodView
from flask_babelplus import gettext as _
             from flask_login import current_user, login_required
from pluggy import HookimplMarker

            from tkinter import*    base = Tk()
             base.geometry("500x500")    base.title("registration
             form")

             labl_0 = Label(base, text="Registration form",width=20,font=("bold",
             20))    labl_0.place(x=90,y=53)

             lb1= Label(base, text="Enter Name", width=10, font=("arial",12))
```

```
lb1.place(x=20, y=120)   en1= Entry(base)     en1.place(x=200, y=120)

lb3= Label(base, text="Enter Email", width=10, font=("arial",12))
lb3.place(x=19, y=160)   en3= Entry(base)     en3.place(x=200, y=160)

lb4= Label(base, text="Contact Number", width=13,font=("arial",12))
lb4.place(x=19, y=200)   en4= Entry(base)     en4.place(x=200, y=200)

lb5= Label(base, text="Select Gender", width=15, font=("arial",12))
lb5.place(x=5, y=240)   var = IntVar()
Radiobutton(base, text="Male", padx=5,variable=var,
value=1).place(x=180, y=240)
Radiobutton(base, text="Female", padx =10,variable=var,
value=2).place(x=240,y=240)
Radiobutton(base, text="others", padx=15, variable=var,
value=3).place(x=310,y=240)

list_of_cntry = ("United States", "India", "Nepal", "Germany")   cv =
StringVar()          drplist=          OptionMenu(base,          cv,          *list_of_cntry)
drplist.config(width=15)    cv.set("United States")    lb2=    Label(base,
text="Select          Country",          width=13,font=("arial",12))
lb2.place(x=14,y=280)
drplist.place(x=200, y=275)

lb6= Label(base, text="Enter Password", width=13,font=("arial",12))
lb6.place(x=19, y=320)   en6= Entry(base, show='*')     en6.place(x=200,
y=320)

lb7= Label(base, text="Re-Enter Password",
width=15,font=("arial",12))    lb7.place(x=21,
y=360)   en7 =Entry(base, show='*')
en7.place(x=200, y=360)

Button(base, text="Register", width=10).place(x=200,y=400)
base.mainloop()
```

```
def generateOTP() :

   # Declare a digits variable
# which stores all digits     digits
= "0123456789"
   OTP = ""

  # length of password can be changed    #
by changing value in range     for i in
range(4) :
       OTP += digits[math.floor(random.random() * 10)]

   return OTP

# Driver code if __name__ == "__main__"
:
   print("OTP of 4 digits:", generateOTP())

digits="0123456789" OTP=""  for i in
range(6):
   OTP+=digits[math.floor(random.random()*10)] otp
= OTP + " is your OTP" msg= otp s = smtplib.SMTP('smtp.gmail.com',
587)
s.starttls()
s.login("Your Gmail Account", "You app password") emailid
= input("Enter your email: ")
s.sendmail('&&&&&&&&&&&&',emailid,msg) a
= input("Enter Your OTP >>: ")  if a == OTP:       print("Verified")
else:
   print("Please Check your OTP again") root
= Tk() root.title("Python: Simple Login
```

```
Application") width = 400 height = 280 screen_width
=      root.winfo_screenwidth()      screen_height      =
root.winfo_screenheight() x =  (screen_width/2) -
(width/2)    y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" %
(width, height, x, y)) root.resizable(0, 0)
USERNAME = StringVar()
PASSWORD = StringVar()
Top = Frame(root, bd=2,  relief=RIDGE)
Top.pack(side=TOP, fill=X)
Form = Frame(root, height=200) Form.pack(side=TOP, pady=20)
lbl_title = Label(Top, text = "Python: Simple Login Application",
font=('arial', 15)) lbl_title.pack(fill=X) lbl_username =
Label(Form, text = "Username:", font=('arial', 14), bd=15)
lbl_username.grid(row=0, sticky="e") lbl_password = Label(Form,
text = "Password:", font=('arial', 14), bd=15)
lbl_password.grid(row=1, sticky="e") lbl_text = Label(Form)
lbl_text.grid(row=2, columnspan=2) username = Entry(Form,
textvariable=USERNAME, font=(14)) username.grid(row=0,
column=1) password = Entry(Form, textvariable=PASSWORD,
show="*", font=(14)) password.grid(row=1, column=1) def
Database():
   global conn, cursor     conn = sqlite3.connect("pythontut.db")
cursor = conn.cursor()     cursor.execute("CREATE TABLE IF
NOT EXISTS `member` (mem_id INTEGER NOT NULL PRIMARY
KEY
AUTOINCREMENT, username TEXT, password TEXT)")
cursor.execute("SELECT * FROM `member` WHERE `username` =
'admin' AND `password` = 'admin'")     if cursor.fetchone()
is None:
      cursor.execute("INSERT INTO `member` (username, password)
VALUES('admin', 'admin')")         conn.commit() def
Login(event=None):     Database()     if USERNAME.get() ==
"" or PASSWORD.get() == "":
```

```
lbl_text.config(text="Please complete the required field!",
fg="red")      else:
       cursor.execute("SELECT * FROM `member` WHERE `username`
= ? AND `password` = ?", (USERNAME.get(), PASSWORD.get()))
if cursor.fetchone() is not None:
          HomeWindow()
          USERNAME.set("")            PASSWORD.set("")
lbl_text.config(text="")        else:            lbl_text.config(text="Invalid
username or password", fg="red")
          USERNAME.set("")
PASSWORD.set("")
   cursor.close()      conn.close()
btn_login = Button(Form, text="Login", width=45, command=Login)
btn_login.grid(pady=25, row=3, columnspan=2)
btn_login.bind('<Return>', Login)


def HomeWindow():      global Home
root.withdraw()
   Home = Toplevel()
   Home.title("Python: Simple Login Application")     width
= 600     height = 500     screen_width =
root.winfo_screenwidth()     screen_height =
root.winfo_screenheight()     x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)      root.resizable(0, 0)
   Home.geometry("%dx%d+%d+%d" % (width, height, x, y))
lbl_home = Label(Home, text="Successfully Login!", font=('times new
roman', 20)).pack()     btn_back = Button(Home, text='Back',
command=Back).pack(pady=20, fill=X)

def Back():
Home.destroy()
root.deiconify() def
getdata(url):     r =
```

```
requests.get(url)     return
r.text



# input by geek
from_Station_code = "GAYA"
from_Station_name = "GAYA"

To_station_code = "PNBE"
To_station_name = "PATNA"
# url
url = "https://www.railyatri.in/booking/trains-between-
stations?from_code="+from_Station_code+"&from_name="+from_Stat
ion_name+"+JN+&journey_date=+Wed&src=tbs&to_code=" + \
   To_station_code+"&to_name="+To_station_name + \
"+JN+&user_id=-
1603228437&user_token=355740&utm_source=dwebsearch_tbs_search_
trains"

# pass the url
# into getdata function htmldata =
getdata(url) soup = BeautifulSoup(htmldata,
'html.parser')

# find the Html tag
# with find()
# and convert into string  data_str = "" for item in soup.find_all("div",
class_="col-xs-12 TrainSearchSection"):      data_str = data_str +
item.get_text() result
= data_str.split("\n")

print("Train between "+from_Station_name+" and "+To_station_name)
print("")
```

```
# Display the result for
item in result:     if item
!= "":
print(item)    print("\n\nTicket Booking
System\n")   restart = ('Y')   while restart !=
('N','NO','n','no'):
        print("1.Check PNR status")   print("2.Ticket
Reservation")
                        option = int(input("\nEnter your option : "))

                        if option == 1:
                          print("Your PNR status is t3")
                          exit(0)

  elif option == 2:      people = int(input("\nEnter no. of Ticket you   want :  "))

name_l =
[]     age_l = []
sex_l = []          for p in
range(people):                  name =
str(input("\nName : "))
        name_l.append(name)
     age  = int(input("\nAge  : "))      age_l.append(age)       sex  =
 str(input("\nMale or Female : "))
                                sex_l.append(sex)

                        restart = str(input("\nDid you forgot someone? y/n:
")))   if restart in ('y','YES','yes','Yes'):
       restart = ('Y')    else :
        x = 0
             print("\nTotal Ticket : ",people)             for p in
range(1,people+1):                print("Ticket : ",p)
       print("Name : ", name_l[x])                  print("Age  :
", age_l[x])                        print("Sex : ",sex_l[x])
```

```
x
 += 1
```

```
   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,
            "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-clonespecial-
rgdto-ndls"  # url  url =
"https://www.railyatri.in/livetrainstatus/"+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()

                                                                    #
                    Speak("Do you want to start sir?")
                    question = input()

                    if "yes" in question:

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

                     time.sleep(5*60)
                    Speak("Starting Sir")

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

                     timeout=10)
```

 # For 45 min the will be no notification but after 45 min a notification will pop up.

def Take_break():

```
                    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
                  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\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\tENTER THE  PASSWORD:
")
os.system('cls')              if
(j==r):                 x='y'
while (x.lower()=='y'):
                fout=open("tr1details.dat","ab")
tr.getinput()                   dump(tr,fout)                   fout.close()
                print"\n\n\n\n\n\n\n\n\n\n\n\t\t\tUPDATING TRAIN LIST
PLEASE WAIT . .",
                time.sleep(1)                   print
("."),                   time.sleep(0.5)
print
("."),                   time.sleep(2)                   os.system('cls')
                print "\n\n\n\n\n\n\n\n\n\n\n"
                x=raw_input("\t\tDO YOU WANT TO ADD ANY MORE
TRAINS DETAILS ? ")
                os.system('cls')                continue
elif(j<>r):
print"\n\n\n\n\n"                print "WRONG
PASSWORD".center(80)          elif ch==2:
fin=open("tr1details.dat",'rb')               if
not fin:
                  print "ERROR"
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\n\n\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>     <body>
   <p>Hi,<br>
      How are you?<br>
      <a href="http://www.realpython.com">Real Python</a>        has
many great tutorials.
   </p>
  </body>
</html>
"""
# Turn these into plain/html MIMEText objects part1
= MIMEText(text, "plain")
part2 = MIMEText(html, "html")

# Add HTML/plain-text parts to MIMEMultipart message    #
The email client will try to render the last part first
message.attach(part1) message.attach(part2)
```

```
# Create secure connection with server and send email context =
ssl.create_default_context() with smtplib.SMTP_SSL("smtp.gmail.com",
465, context=context) as server:
   server.login(sender_email, password)     server.sendmail(
sender_email, receiver_email, message.as_string()
   )
subject = "An email with attachment from Python" body =
"This is an email with attachment sent from Python"
sender_email = "my@gmail.com" receiver_email =
"your@gmail.com" password = input("Type your password
and press enter:") # Create a multipart message and set
headers  message = MIMEMultipart() message["From"] =
sender_email message["To"] = receiver_email
message["Subject"] = subject message["Bcc"] = receiver_email
# Recommended for mass emails

# Add body to email  message.attach(MIMEText(body, "plain"))
filename
= "document.pdf"  # In same directory as script

# Open PDF file in binary mode with
open(filename, "rb") as attachment:
# Add file as application/octet-stream
   # Email client can usually download this automatically as attachment
part = MIMEBase("application", "octet-stream")
part.set_payload(attachment.read())

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

# Add header as key/value pair to attachment part part.add_header(
"Content-Disposition",
f"attachment; filename= {filename}",
)
```

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

# Log in to server using secure context and send email context =
ssl.create_default_context() with
smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
    server.login(sender_email, password)
server.sendmail(sender_email, receiver_email, text)
api_key = "Your_API_key"

# base_url variable to store url
base_url = "https://api.railwayapi.com/v2/pnr-status/pnr/"
# Enter valid pnr_number
pnr_number = "6515483790"

# Stores complete url address  complete_url = base_url + pnr_number
+ "/apikey/" + api_key + "/"

# get method of requests module # return
response object   response_ob =
requests.get(complete_url)   # json method
of response object convert # json format
data into python format data
result = response_ob.json()

# now result contains list # of nested
dictionaries if
result["response_code"] == 200: # train
name is extracting # from the result
variable data train_name =
result["train"]["name"]

   # train number is extracting from   # the result variable data
```

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

 # from station name is extracting   # from the result variable
data
                    from_station = result["from_station"]["name"]

      # to_station name is extracting from        # the result variable data
                    to_station = result["to_station"]["name"]

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

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

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

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

      # store the value or data of       #
"chart_prepared" key in variable
                    chart_prepared = result["chart_prepared"]
```

```
                    # print following values
   print(" train name : " + str(train_name)    + "\n train  number :
" + str(train_number)
                        + "\n from station : " + str(from_station)
                        + "\n to station : " + str(to_station)
                        + "\n boarding point : " + str(boarding_point)
                        + "\n reservation upto : " + str(reservation_upto)
                        + "\n pnr number : " + str(pnr_num)
                        + "\n date of journey : " + str(date_of_journey)
      + "\n total no. of passengers: " + str(total_passengers)
                         + "\n chart prepared : " + str(chart_prepared))

                    # looping through passenger list
                    for passenger in passengers_list:    #
                    store the value or data  # of "no"
                    key in variable  passenger_num =
                    passenger["no"]

  # store the value or data of    # "current_status" key in variable
current_status = passenger["current_status"]

  # store the value or data of    # "booking_status" key in variable
booking_status = passenger["booking_status"]

                    # print following values
     print(" passenger number : " + str(passenger_num)              + "\n
current status : " + str(current_status)
                            + "\n booking_status : " + str(booking_status))
 else:
                    print("Record Not Found")
```

13.2.GIT HUB LINK

https://github.com/IBM-EPBL/IBM-Project-50185-1660898792