SMART SOLUTION FOR RAILWAYS

By

JOICE JERLIN RAJI T	Team Leader
BANU V	Team Member 1
DEEPIKA S	Team Member 2
JOSHINIKA V	Team Member 3

A report for the 19CSP14 - PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

ELECTRONICS AND COMMUNICATION ENGINEERING

In partial fulfilment of the award of the degree of

BACHELOR OF ENGINEERING

in

Electronics and Communication Engineering

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

Project Report

1.INTRODUCTION

1.1 ABSTRACT

Current ticket reservation system is based on use of QR Code, which contains the details of the ticket records such as train timings, its arrival timings, departure timings and passenger reservations details. The printed ticket consists of information which includes all train details with QR Code Information. The Ticket reservation system involves three main factors the database, online passenger and dataset. In the proposed system GUI is developed for the users through by which users book their tickets and the ticket generated will be in the form of QR code which is generated after booking confirmation. The QR Code will be generated on the basis of encrypted data entered by the user.

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-to-human and human-to-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

2.1 EXISTING SYSTEM

In the Existing train tracks are manually researched. LED (Light Emitting Diode) and LDR (Light Dependent Resister) sensors cannot be implemented on the block of the tracks]. The input image processing is a clamorous system with high cost and does not give the exact result. The Automated Visual Test Method is a complicated method as the video color inspection is implemented to examine the cracks in rail track which does not give accurate result in bad weather. This traditional system delays transfer of information. Srivastava et al., (2017) proposed a moving gadget to detect the cracks with the help of an array of IR sensors to identify the actual position of the cracks as well as notify to nearest railway station . Mishra et al., (2019) developed a system to track the cracks with the help of Arduino mega power using solar energy and laser. A GSM along with a GPS module was implemented to get the actual location of the faulty tracks to inform the authorities using SMS via a link to find actual location on Google Maps. Rizvi Aliza Raza presented a prototype in that is capable of capturing photos of the track and compare it with the old database and sends a message to the authorities regarding the crack detected. The detailed analysis of traditional railway track fault detection techniques is explained in table

2.2 References

1) *Gangurde, Nirmit, Subendu Ghosh, Akash Giri, and Swapnil Gharat. "Ticketing System Using AES Encryption Based QR Code." In 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 201-206. IEEE, 2022.*

   In this paper GUI is developed for the users through by which users book their tickets and the ticket generated will be in the form of QR code which is generated after booking confirmation. The QR Code will be generated on the basis of encrypted data entered by the user. A mobile application is designed to scan the encrypted QR Code. On decrypting, the information about the passenger can be viewed.

2) *Kazi, Sanam, Murtuza Bagasrawala, Farheen Shaikh, and Anamta Sayyed. "Smart eticketing system for public transport bus." In 2018 International Conference on Smart City and Emerging Technology (ICSCET), pp. 1-7. IEEE, 2018.*

   The user can check the availability of seats, book tickets, get the seat automatically through efficient novel algorithm and the expected waiting time. If seats are not vacant, our algorithm will efficiently allot the seat that will be vacant in shortest time. They will pay digitally through our portal.

3) *Karthick, S., and A. Velmurugan. "Android suburban railway ticketing with GPS as ticket checker." In 2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 63-66. IEEE, 2012.*

   This paper Android Suburban Railway (ASR) ticketing is mainly to buy the suburban tickets. Our ASR ticket can be bought with just a smart phone application, where you can carry your suburban railway tickets in your smart phone as a QR code. It uses the smart phones "GPS" facility to validate and delete your ticket automatically after a specific interval of time once the user reaches the destination.

4) *Alam, Shah, Mahfuzulhoq Chowdhury, and Abu Bakkar Siddique. "A User-friendly Android Application Featuring Smart Ticketing System and Destination Announcement for Metro Rail based Rapid Transport System in Bangladesh." In 2021 3rd International Conference on Electrical & Electronic Engineering (ICEEE), pp. 29-32. IEEE, 2021.*

   This paper presents a user-friendly android application for metro-rail based rapid transport system. It can offer a smart ticketing, users authorization by verifying QR code, and notify the metro-rail passengers when they arrive close to their final destination.

5) *Ariffin, Ahmad Ashraff Bin, Noor Hafizah Abdul Aziz, and Kama Azura Othman. "Implementation of GPS for location tracking." In 2011 IEEE control and system graduate research colloquium, pp. 77-81. IEEE, 2011.*

   This project is aim to design and implement a low cost Global Positioning System suitable to be used for traveling and sailing activities. The function of the GPS is to locate the position of user. The effects of line of sights in relation to different experimented locations are also studied. The GPS modules will generate the coordinates of latitude and longitude as well as the bearing angles between two positions.

2.3 PROBLEM STATEMENT DEFINITION

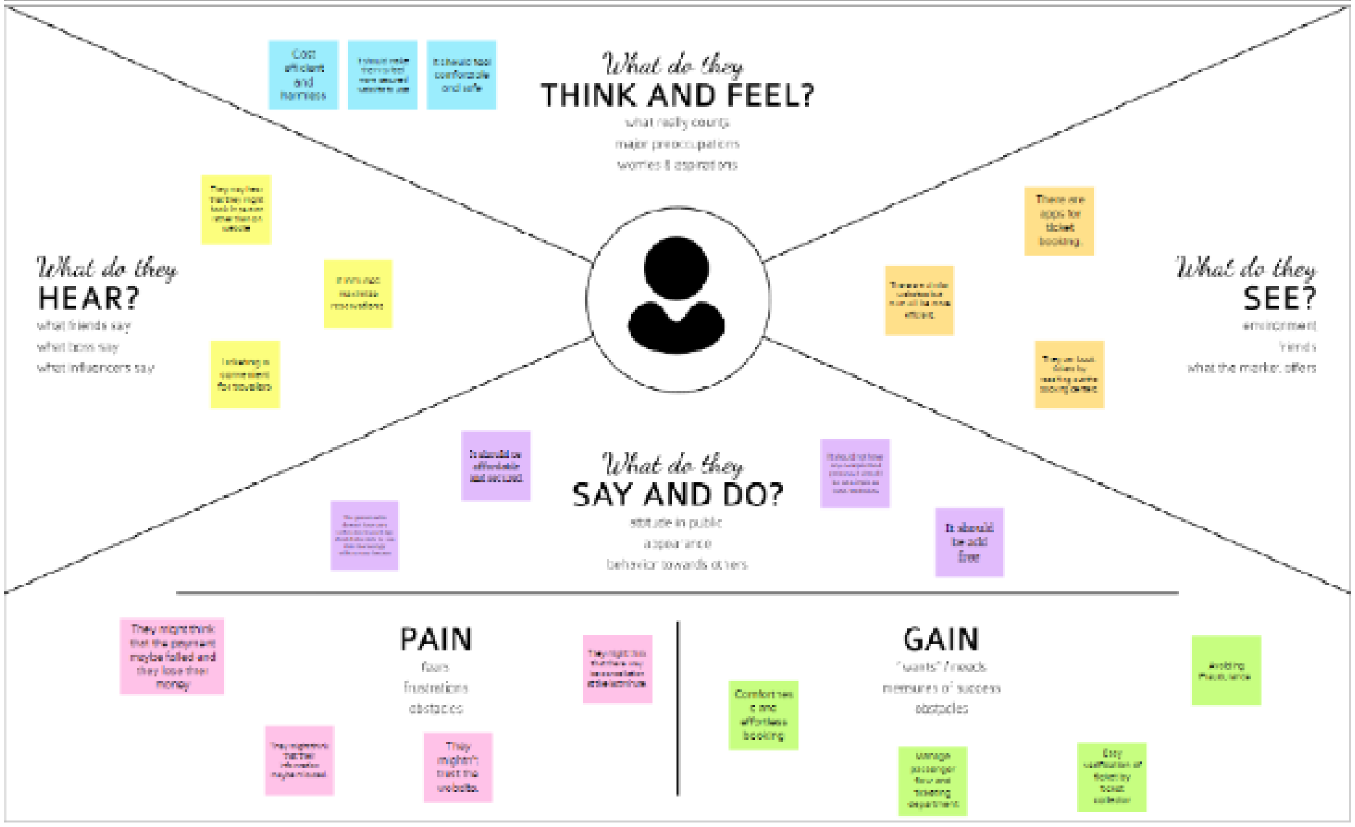
Among the various modes of transport, railways is one of the biggest modes of transport in the world. Though there are competitive threats from airlines, luxury buses, public transports, and personalized transports the problem statement is to answer the question “What are the problems faced by the passengers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

3. IDEATION AND PROPOSED SOLUTON

3.1 EMPATHY MAP CANVAS

Empathize & Discover

Date	25 September 2022
Team ID	PNT2022TMID18248
Project name	Smart solution for railways
Maximum Marks	4 Marks



3.2 IDEATION & BRAINSTORMING

Template

Brainstorm
& idea prioritization

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

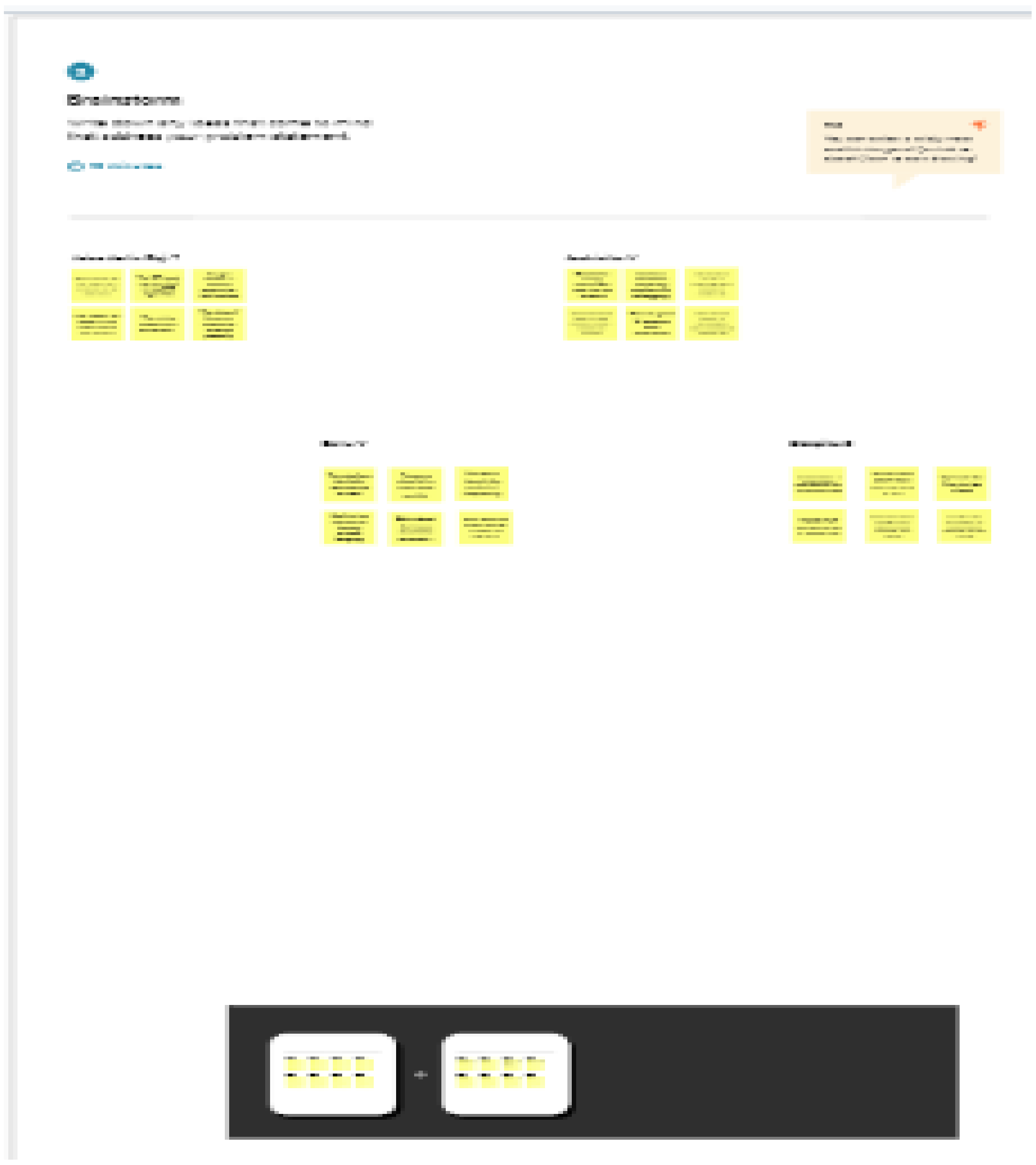
Before you collaborate

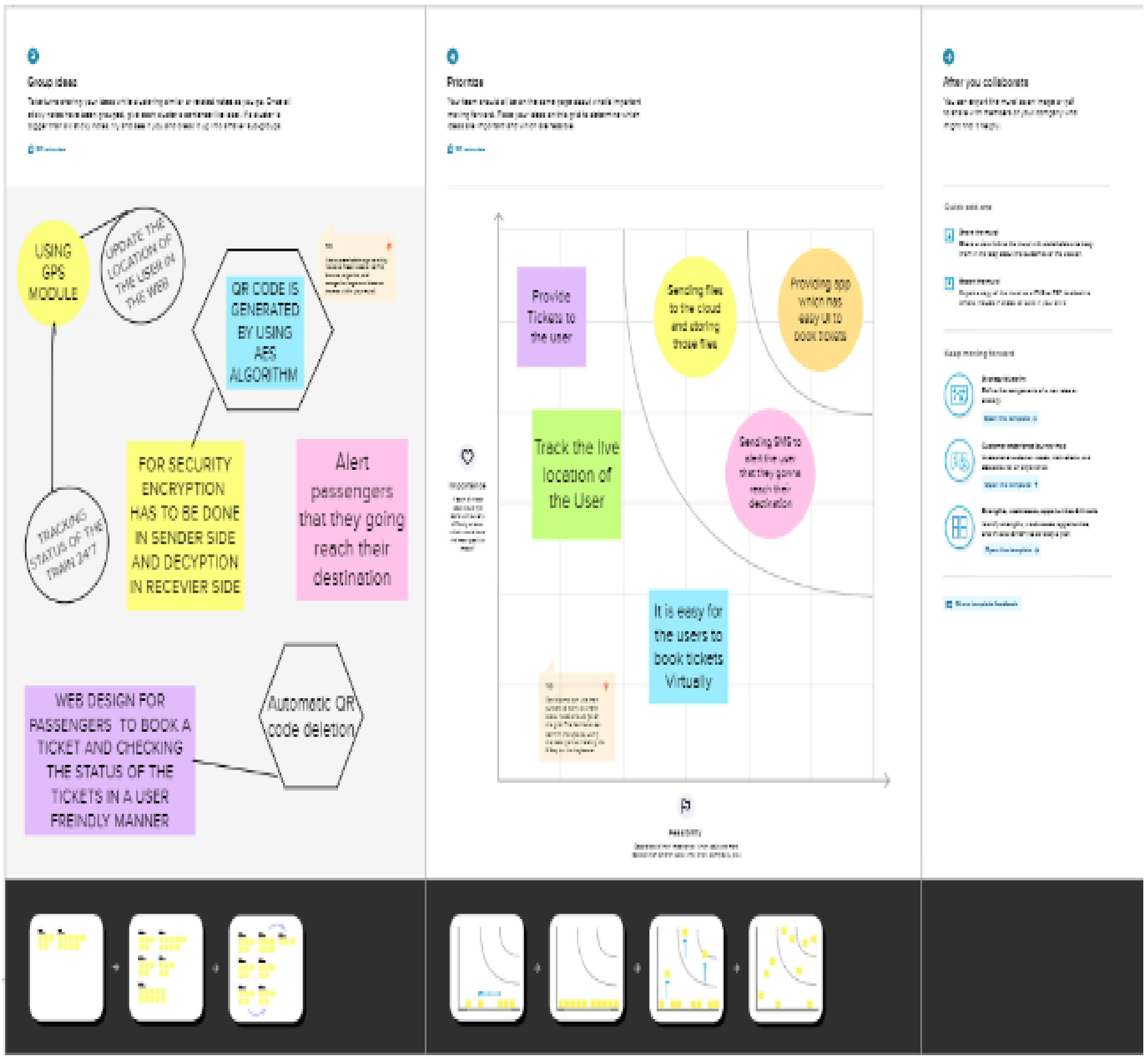
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

Define your problem statement

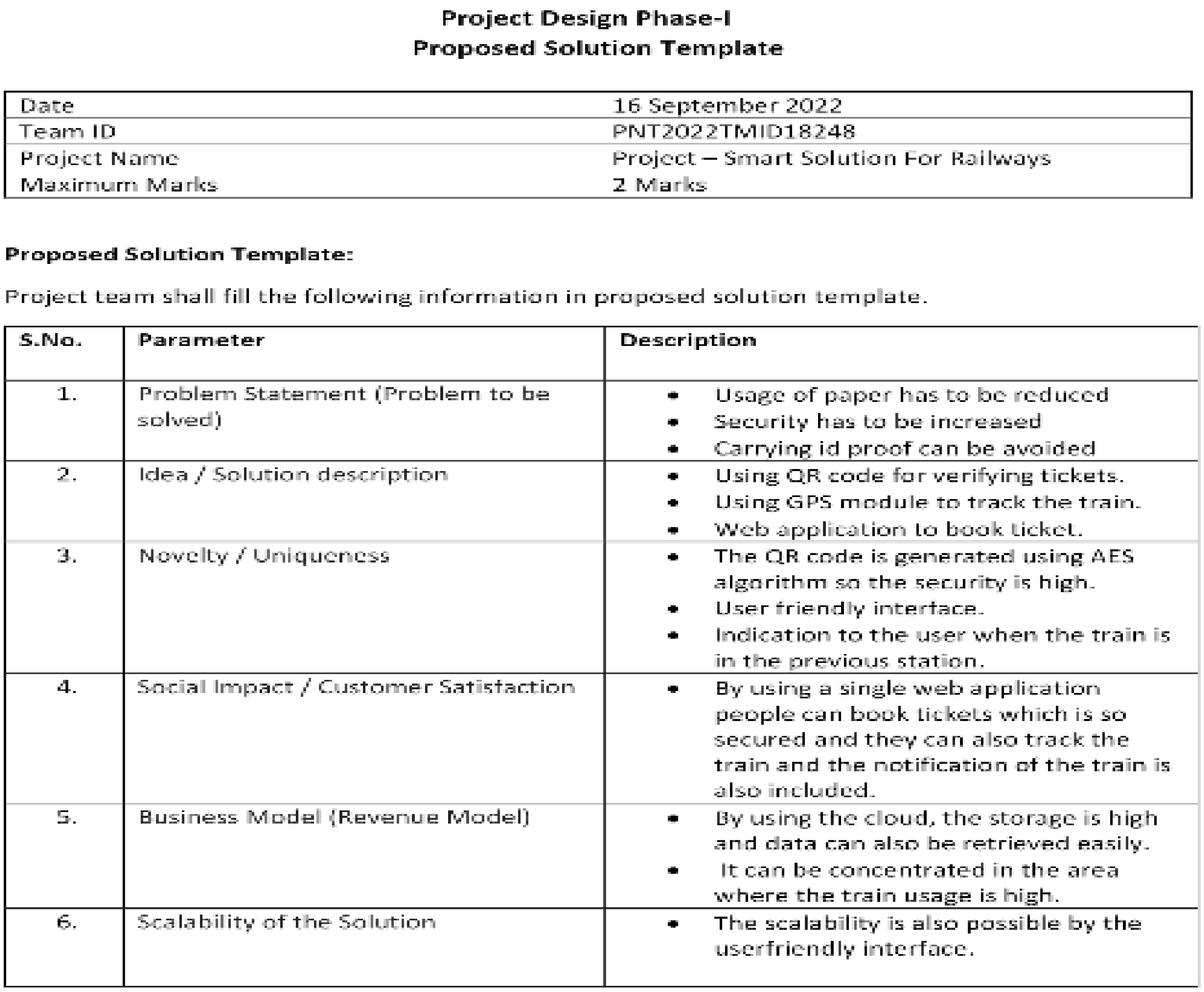
The security is less in formal booking of tickets. The time consumption is also high. The usage of paper is also high to avoid these issues we are going for smart solutions.

Key rules of brainstorming


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


Group ideas
USING GPS MODULE
UPDATE THE LOCATION OF THE USER IN THE WEB
QR CODE IS GENERATED BY USING AES ALGORITHM
TRACKING STATUS OF THE TRAIN 24*7
FOR SECURITY ENCRYPTION HAS TO BE DONE IN SENDER SIDE AND DECYPTION IN RECEVIER SIDE
Alert passengers that they going reach their destination
WEB DESIGN FOR PASSENGERS TO BOOK A TICKET AND CHECKING THE STATUS OF THE TICKETS IN A USER FREINDLY MANNER
Automatic QR code deletion
Prioritize
Provide Tickets to the user
Sending files to the cloud and storing those files
Providing app which has easy UI to book tickets
Track the live location of the User
Sending SMS to alert the user that they gonna reach their destination
It is easy for the users to book tickets Virtually
Importance
Feasibility
After you collaborate

3.3 PROPOSED SOLUTION



Project Design Phase-I
Proposed Solution Template

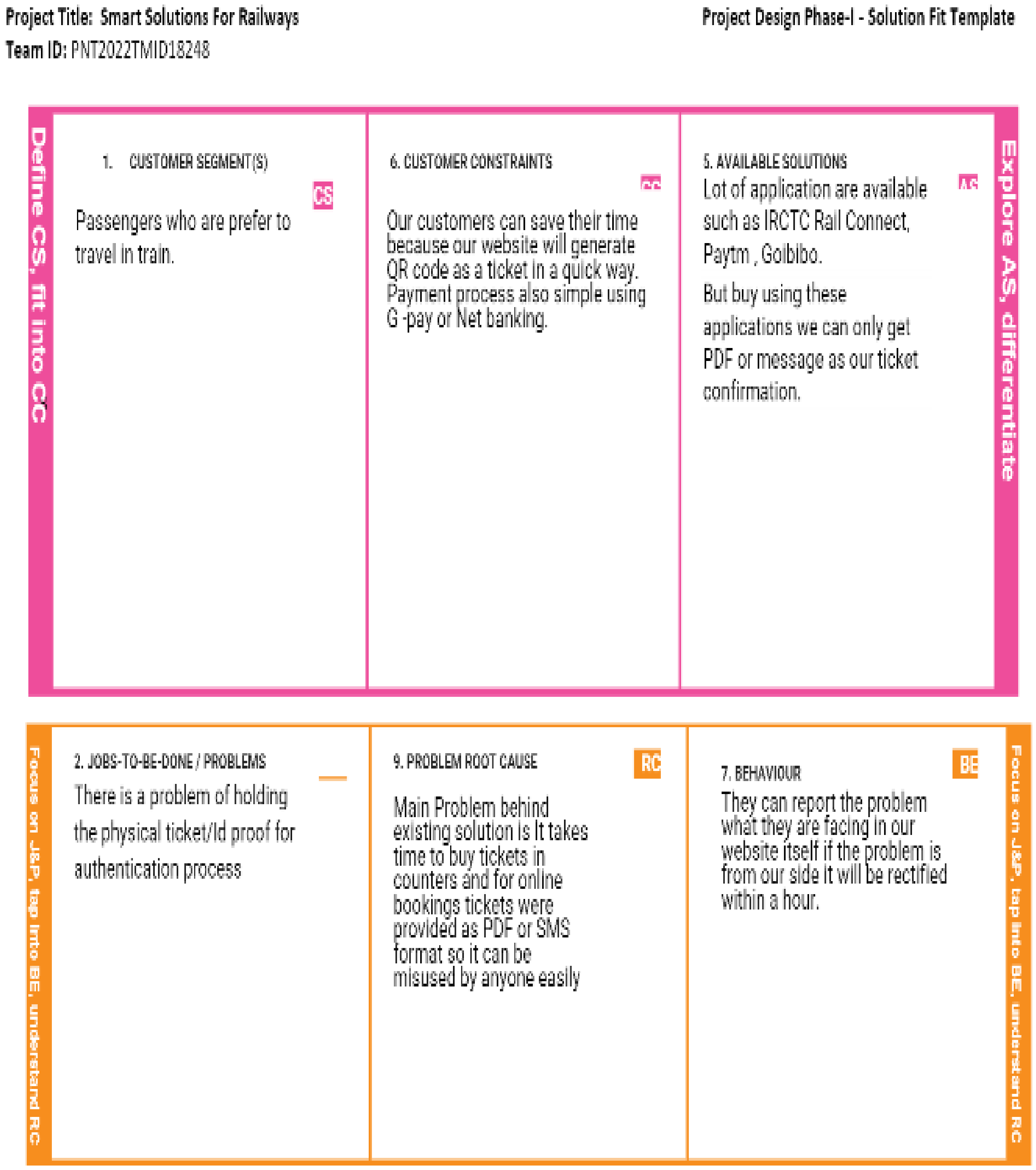
Date	16 September 2022
Team ID	PNT2022TMID18248
Project Name	Project – Smart Solution For Railways
Maximum Marks	2 Marks

Proposed Solution Template:

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• Usage of paper has to be reduced • Security has to be increased • Carrying id proof can be avoided
2.	Idea / Solution description	• Using QR code for verifying tickets. • Using GPS module to track the train. • Web application to book ticket.
3.	Novelty / Uniqueness	• The QR code is generated using AES algorithm so the security is high. • User friendly interface. • Indication to the user when the train is in the previous station.
4.	Social Impact / Customer Satisfaction	• By using a single web application people can book tickets which is so secured and they can also track the train and the notification of the train is also included.
5.	Business Model (Revenue Model)	• By using the cloud, the storage is high and data can also be retrieved easily. • It can be concentrated in the area where the train usage is high.
6.	Scalability of the Solution	• The scalability is also possible by the userfriendly interface.

3.4 PROBLEM SOLUTION FIT



Project Title: Smart Solutions For Railways **Project Design Phase-I - Solution Fit Template**

Team ID: PNT2022TMID18248

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Passengers who are prefer to travel in train.

6. CUSTOMER CONSTRAINTS CC

Our customers can save their time because our website will generate QR code as a ticket in a quick way. Payment process also simple using G -pay or Net banking.

5. AVAILABLE SOLUTIONS AS

Lot of application are available such as IRCTC Rail Connect, Paytm , Goibibo.

But buy using these applications we can only get PDF or message as our ticket confirmation.

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS

There is a problem of holding the physical ticket/Id proof for authentication process

9. PROBLEM ROOT CAUSE RC

Main Problem behind existing solution is It takes time to buy tickets in counters and for online bookings tickets were provided as PDF or SMS format so it can be misused by anyone easily

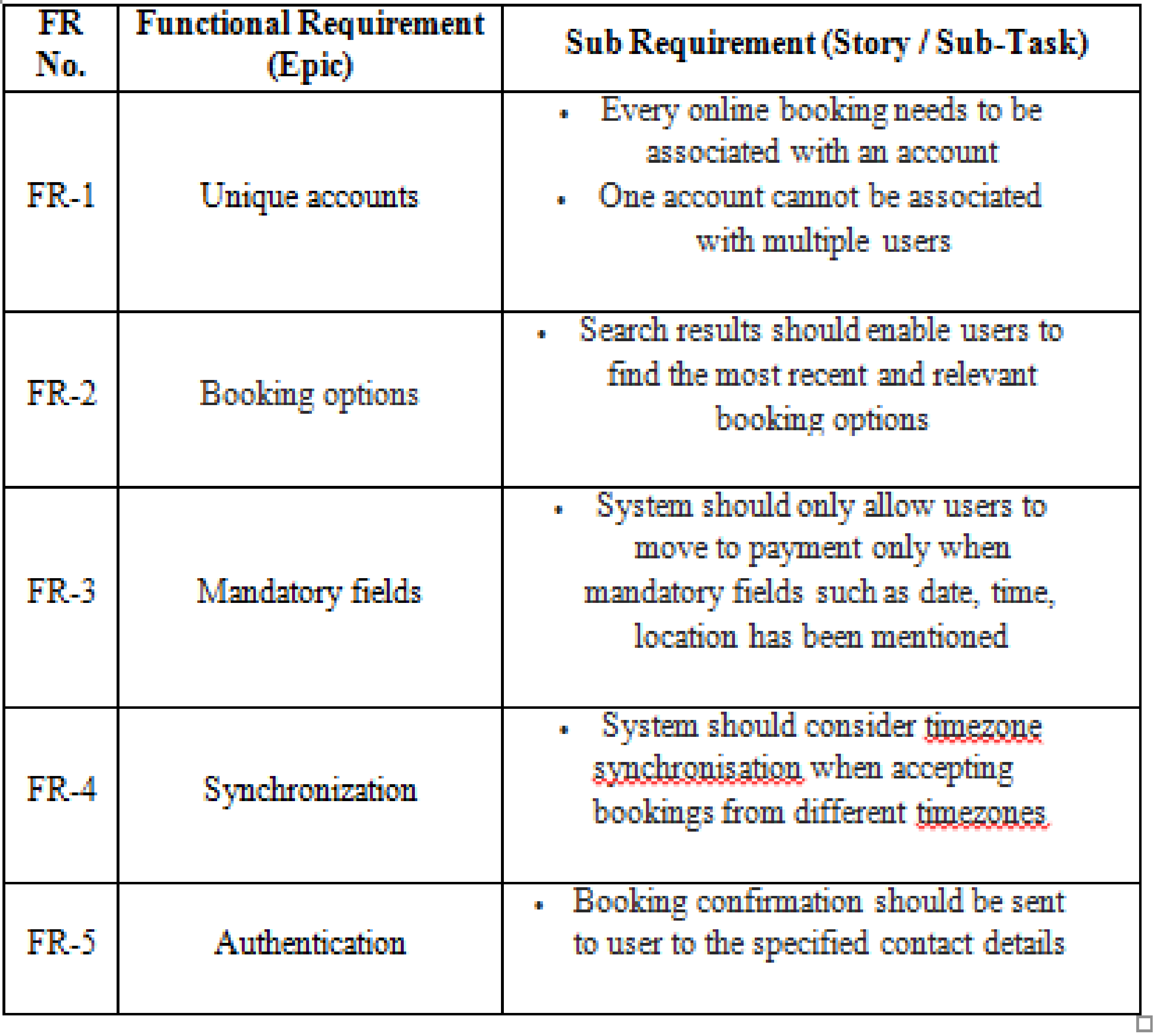
7. BEHAVIOUR BE

They can report the problem what they are facing in our website itself if the problem is from our side it will be rectified within a hour.

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

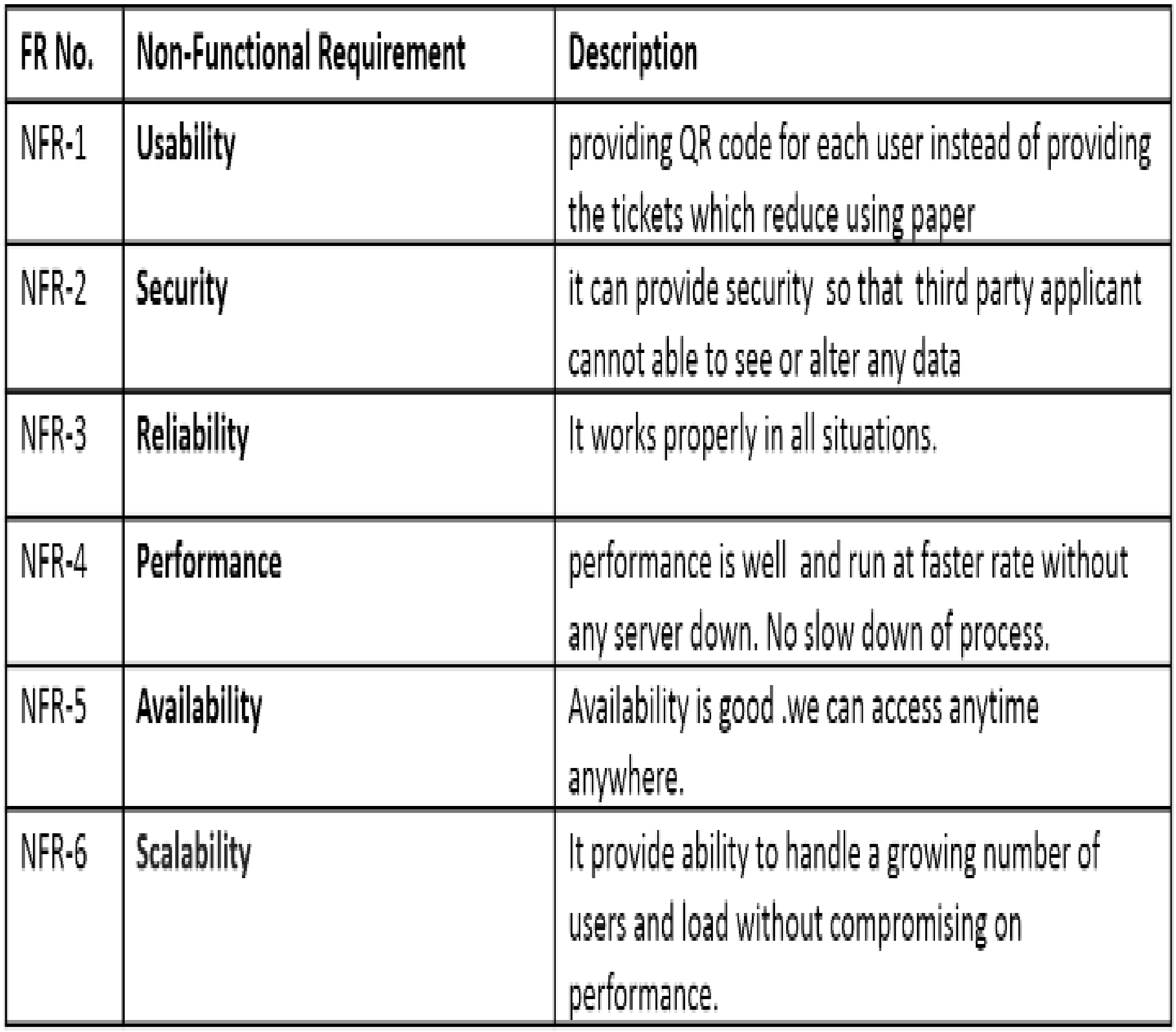
4.REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENTS



FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Unique accounts	• Every online booking needs to be associated with an account • One account cannot be associated with multiple users
FR-2	Booking options	• Search results should enable users to find the most recent and relevant booking options
FR-3	Mandatory fields	• System should only allow users to move to payment only when mandatory fields such as date, time, location has been mentioned
FR-4	Synchronization	• System should consider timezone synchronisation when accepting bookings from different timezones
FR-5	Authentication	• Booking confirmation should be sent to user to the specified contact details

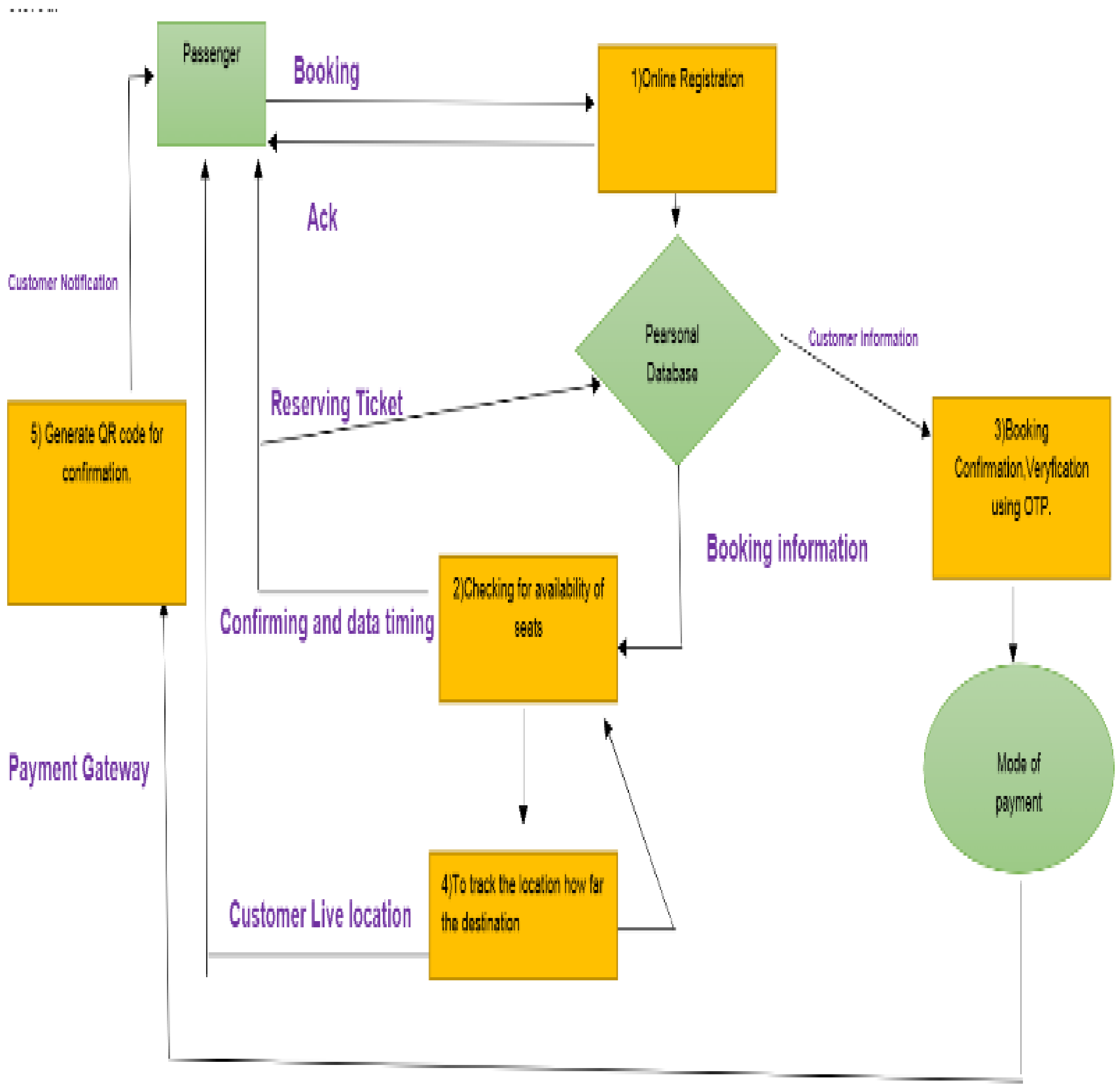
4.2. NON-FUNCTIONAL REQUIREMENTS



FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	providing QR code for each user instead of providing the tickets which reduce using paper
NFR-2	**Security**	it can provide security so that third party applicant cannot able to see or alter any data
NFR-3	**Reliability**	It works properly in all situations.
NFR-4	**Performance**	performance is well and run at faster rate without any server down. No slow down of process.
NFR-5	**Availability**	Availability is good .we can access anytime anywhere.
NFR-6	**Scalability**	It provide ability to handle a growing number of users and load without compromising on performance.

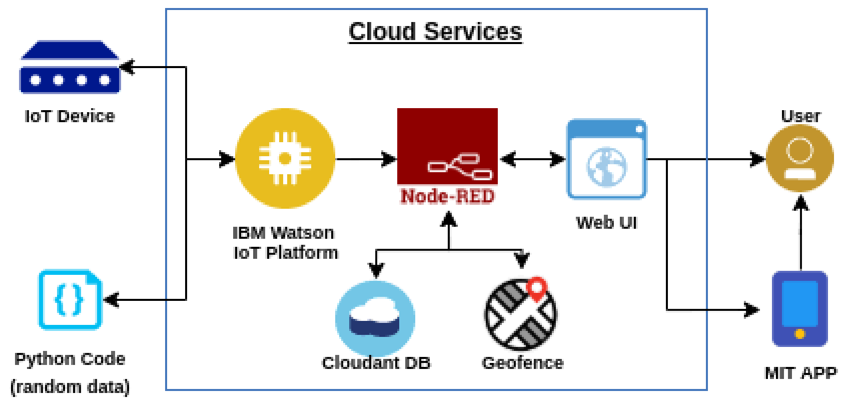
5.PROJECT DESIGN

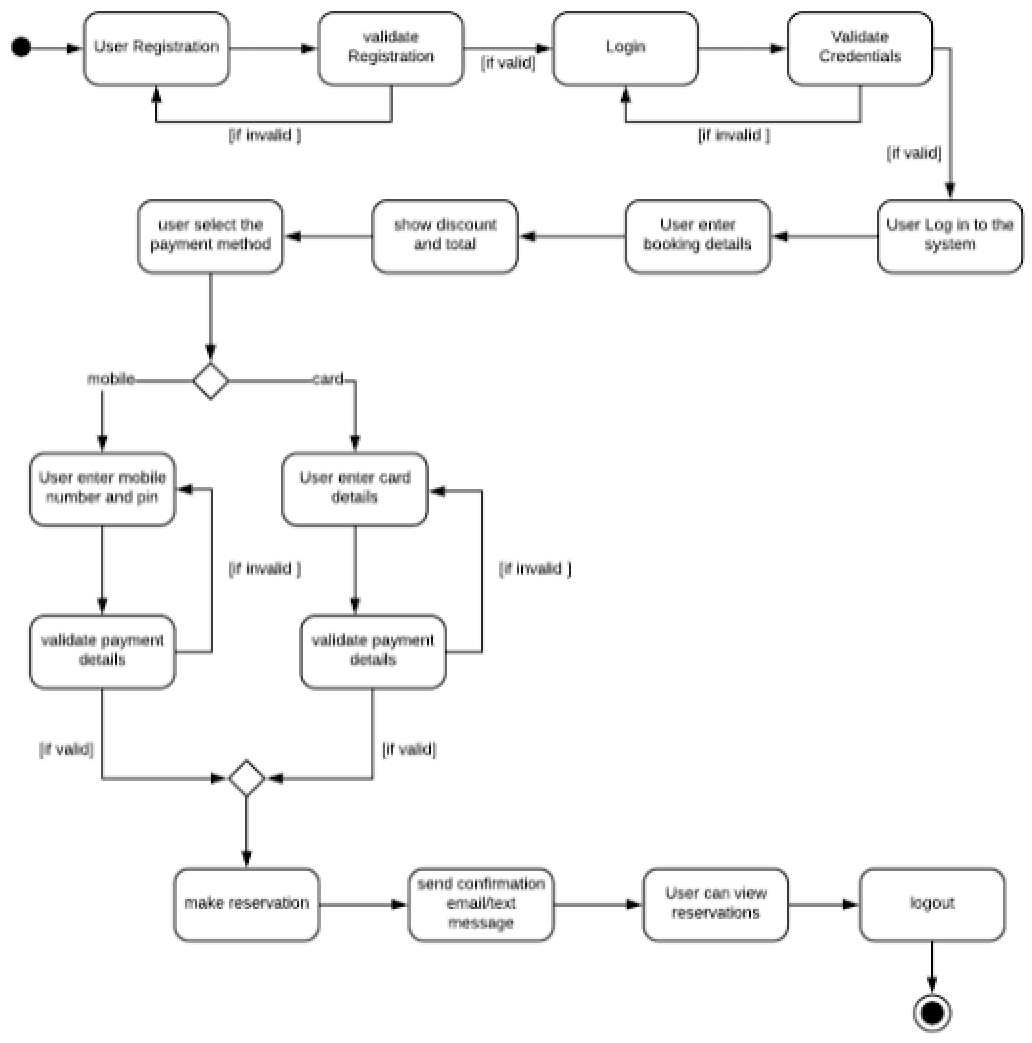
5.1 DATA FLOW DIAGRAMS



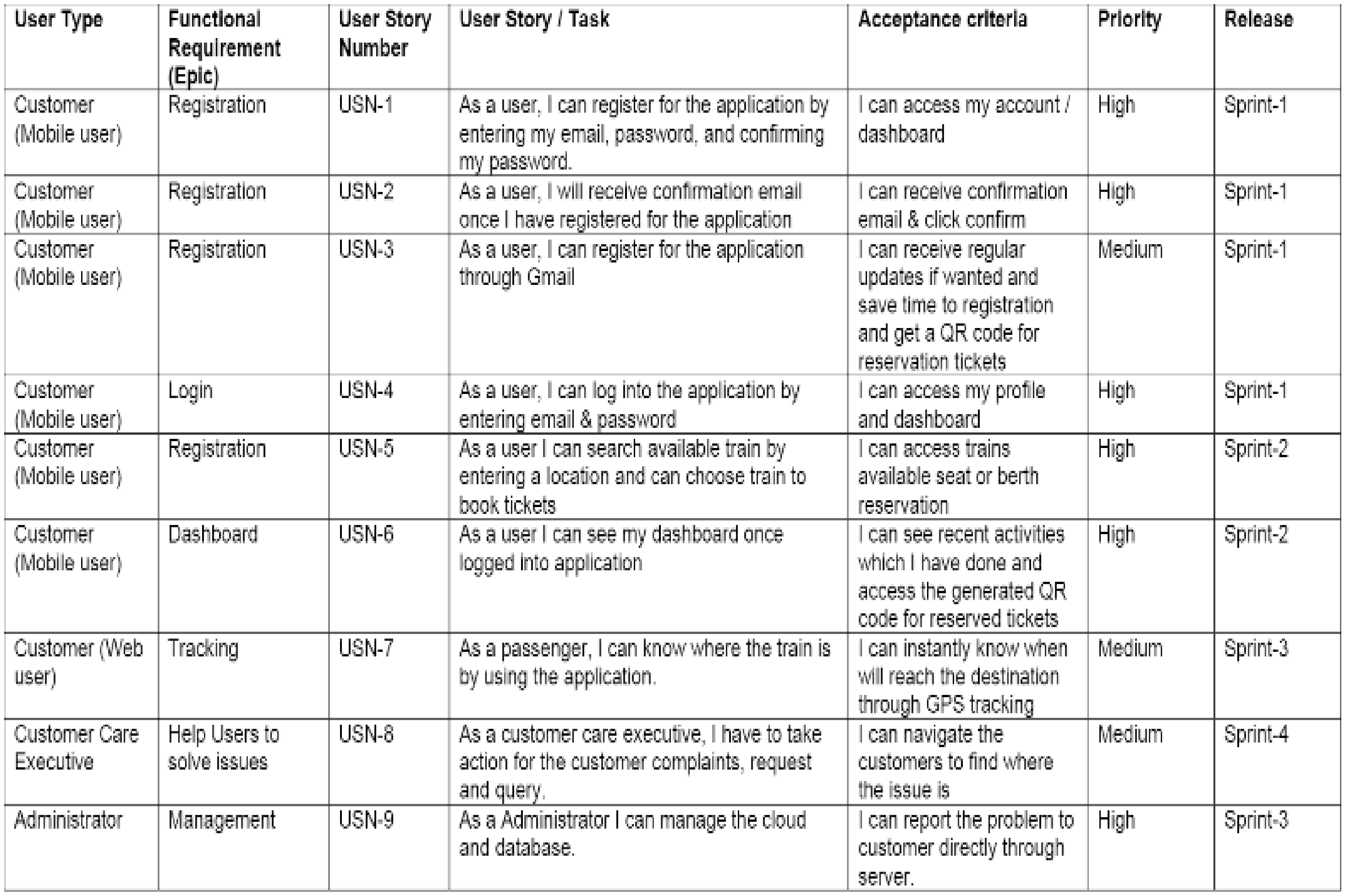
Passenger
Booking
1)Online Registration
Ack
Customer Notification
Pearsonal Database
Customer Information
Reserving Ticket
5) Generate QR code for confirmation.
3)Booking Confirmation,Veryfication using OTP.
Booking information
2)Checking for availability of seats
Confirming and data timing
Mode of payment
Payment Gateway
4)To track the location how far the destination
Customer Live location

5.2 SOLUTION & TECHNICAL ARCHITECTURE





5.3 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
Customer (Mobile user)	Registration	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
Customer (Mobile user)	Registration	USN-3	As a user, I can register for the application through Gmail	I can receive regular updates if wanted and save time to registration and get a QR code for reservation tickets	Medium	Sprint-1
Customer (Mobile user)	Login	USN-4	As a user, I can log into the application by entering email & password	I can access my profile and dashboard	High	Sprint-1
Customer (Mobile user)	Registration	USN-5	As a user I can search available train by entering a location and can choose train to book tickets	I can access trains available seat or berth reservation	High	Sprint-2
Customer (Mobile user)	Dashboard	USN-6	As a user I can see my dashboard once logged into application	I can see recent activities which I have done and access the generated QR code for reserved tickets	High	Sprint-2
Customer (Web user)	Tracking	USN-7	As a passenger, I can know where the train is by using the application.	I can instantly know when will reach the destination through GPS tracking	Medium	Sprint-3
Customer Care Executive	Help Users to solve issues	USN-8	As a customer care executive, I have to take action for the customer complaints, request and query.	I can navigate the customers to find where the issue is	Medium	Sprint-4
Administrator	Management	USN-9	As a Administrator I can manage the cloud and database.	I can report the problem to customer directly through server.	High	Sprint-3

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
			completion of payment I'll be redirected to the booking website.	be done I can move back to the initial payment page		
	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.	I can show the generated QR code so that authentication can be done quickly.	High	Sprint-1
	Ticket status	USN-11	As a user, I can see the status of my ticket Whether it's confirmed/waiting/RAC.	I can confidentially get the Information and arrange alternate transport if the ticket isn't Confirmed	High	Sprint-1
	Remainders notification	USN-12	As a user, I get remainders about my journey A day before my actual journey.	I can make sure that I don't miss the journey because of the constant notifications.	Medium	Sprint-2
		USN-13	As a user, I can track the train using GPS and can get information such as ETA, Current stop and delay.	I can track the train and get to know about the delays pian accordingly	Medium	Sprint-2
	Ticket cancellation	USN-14	As a user, I can cancel my tickets if there's any Change of plan	I can cancel the ticket and get a refund based on how close the date is to the journey.	High	Sprint-1
	Raise queries	USN-15	As a user, I can raise queries through the query box or via mail.	I can view my pervious queries.	Low	Sprint-2
Customer care Executive	Answer the queries	USN-16	As a user, I will answer the questions/doubts Raised by the customers.	I can view the queries and make it once resolved	Medium	Sprint-2
Administrator	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.	I can view and ensure the corrections of the information fed.	High	Sprint-1

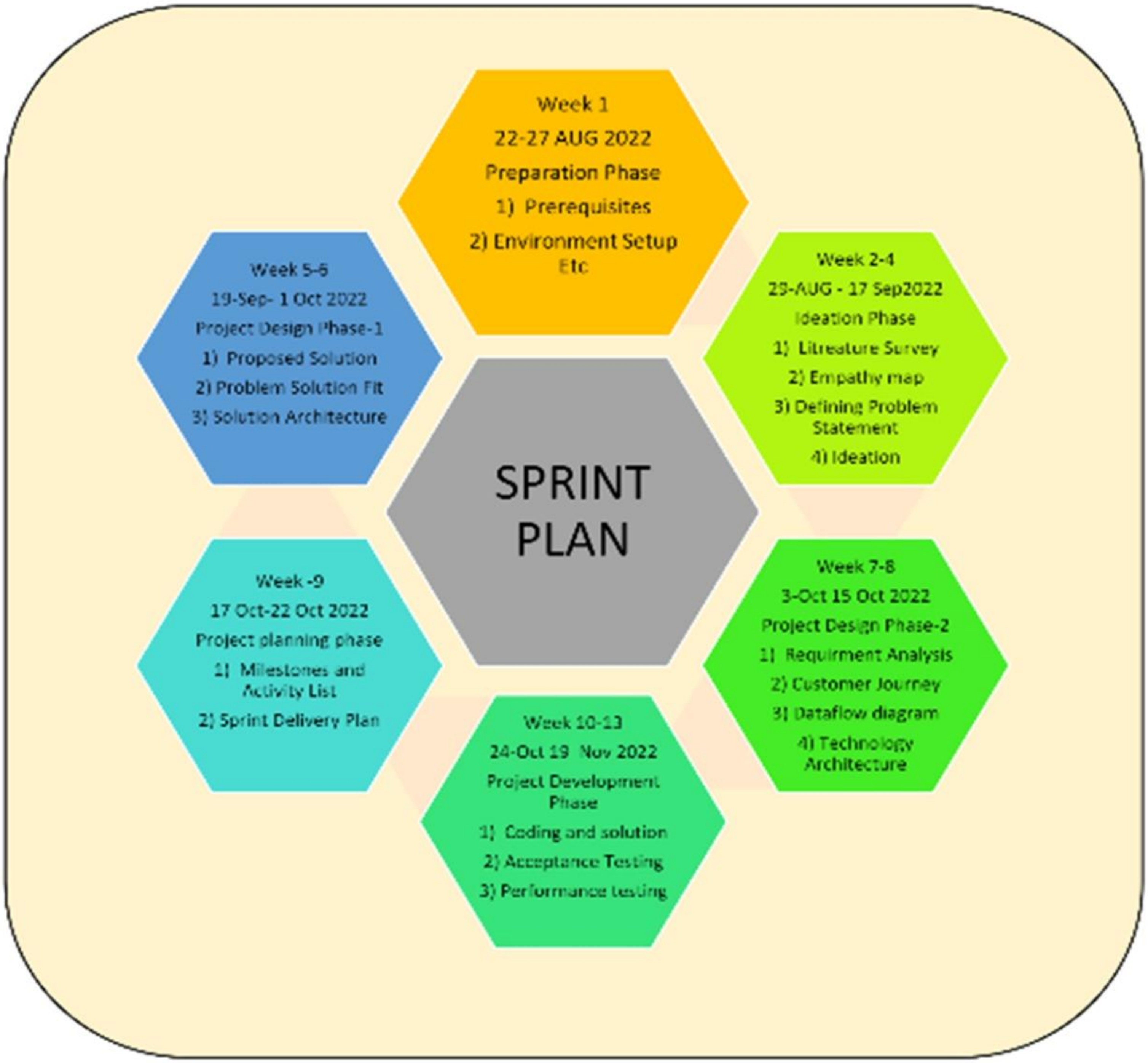
6.PROJECT PLANNING AND SCHEDULING

6.1. SPRINT PLANNING& ESTIMATION

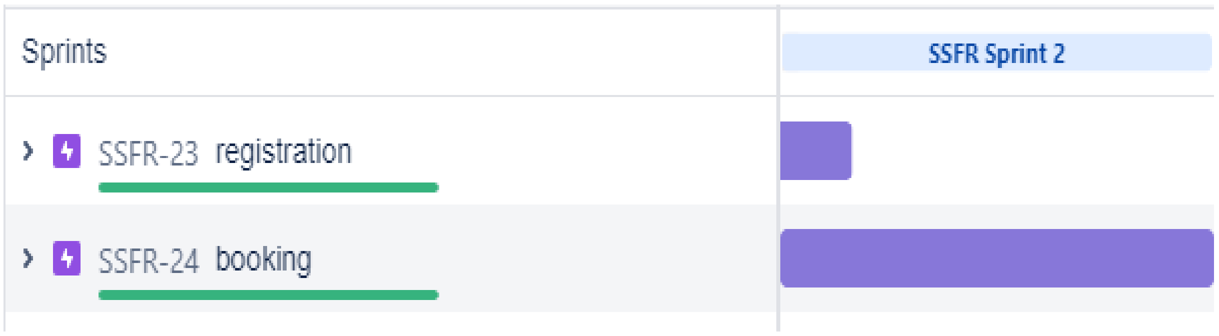
Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**
Sprint-1	Registration	USN-1	As a user, I can register through the form by Filling in my details	2	High
Sprint-1		USN-2	As a user, I can register through phone numbers, Gmail, Facebook or other social sites	1	High
Sprint-1	Conformation	USN-3	As a user, I will receive confirmation through email or OTP once registration is successful	2	Low
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
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
Sprint-2	Booking	USN-6	As a use, I can provide the basic details such as a name, age, gender etc…	2	High
Sprint-2		USN-7	As a user, I can choose the class, seat/berth. If preferred seat/berth isn’t available I can be allocated based on the availability	1	Low
Sprint-2	Payment	USN-8	As a user, I can choose to pay through credit Card/debit card/UPI.	1	High
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-2		USN-9	As a user, I will be redirected to the selected	2	High
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 usedfor authentication during my journey.	1	High
Sprint-3	Ticket status	USN-11	As a user, I can see the status of my ticket.	2	High

Sprint-3	Remainders notification	USN-12	As a user, I get remainders about my journey A day before my actual journey.	1	High
Sprint-3	Ticket cancellation	USN-13	As a user, I can track the train using GPS and can get information such asETA, Current stop and delay	2	High
Sprint-4		USN-14	As a user, I can cancel my tickets if there's any Change of plan	1	High
Sprint-4	Raise queries	USN-15	As a user, I can raise queries throughthe query box or via mail.	2	Medium
Sprint-4	Answer the queries	USN-16	As a user, I will answer the questions/doubts Raised by the customers.	2	High
Sprint-4	Feed details	USN-17	As a user, I will feed information about the trains delays and add extraseats if a new compartment is added.	1	High

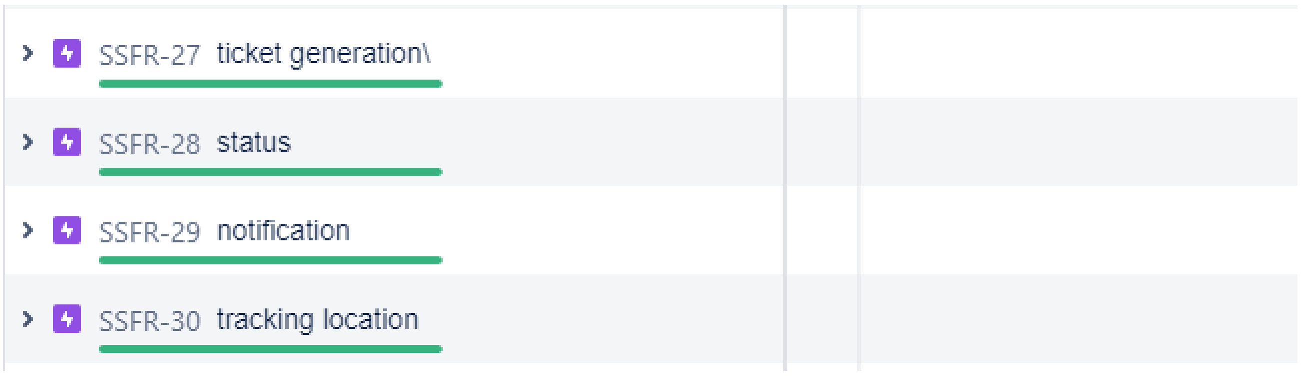
6.2. SPRINT DELIVERY SCHEDULE



6.3. REPORTS FROM JIRA



Sprints	SSFR Sprint 2
SSFR-23 registration	
SSFR-24 booking	



SSFR-27 ticket generation\
SSFR-28 status
SSFR-29 notification
SSFR-30 tracking location

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.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 *
import sqlite3
root = Tk()
root.title("TICK BOOKING")
width = 500
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.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 = "TICK BOOKING", 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` =
    'joice' AND `password` = 'joice'")
        cursor.execute("SELECT * FROM `member` WHERE `username` =
    'banu' AND `password` = 'banu'")
        cursor.execute("SELECT * FROM `member` WHERE `username` =
    'deeps' AND `password` = 'deeps'")
        cursor.execute("SELECT * FROM `member` WHERE `username` =
    'josh' AND `password` = 'josh12'")
        if cursor.fetchone() is None:
            cursor.execute("INSERT INTO `member` (username, password)
    VALUES('joice', 'joice')")
            cursor.execute("INSERT INTO `member` (username, password)
    VALUES('banu', 'banu')")
            cursor.execute("INSERT INTO `member` (username, password)
    VALUES('deeps', 'deeps')")
            cursor.execute("INSERT INTO `member` (username, password)
    VALUES('josh', 'josh12')")
conn.commit()
    def Login(event=None):
        Database()
        if USERNAME.get() == "" or PASSWORD.get() == "":
            lbl_text.config(text="Complete the required field!", fg="blue")
        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 login", fg="blue")
            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("TICK BOOKING")
    width = 500
    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="Login Successfull!", 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 generateOTP() :
     digits = "0123456789"
     OTP = ""
     for i in range(6) :
        OTP += digits[math.floor(random.random() * 8)]
    return OTP

if__name__== "__main__" :

    print(“OTP:”generateOTP())

digits="0123456789"
OTP=""
for i in range(6):
    OTP+=digits[math.floor(random.random()*8)]
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")
base = Tk()
base.geometry("500x500")
base.title("register here")

labl_0 = Label(base,
text="Register
here",width=20,font=("bol
d", 20))
labl_0.place(x=90,y=53)

lb1= Label(base,
text="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="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="Phone Number",
width=13,font=("arial",12)
)
lb4.place(x=19, y=200)
en4= Entry(base)
en4.place(x=200, y=200)

list_of_gender = ("Male",
"Female", "Others")
cv = StringVar()
drplist= OptionMenu(base,
cv, *list_of_gender)
drplist.config(width=15)
cv.set("Select")
lb2= Label(base,
text="Gender",
width=13,font=("arial",12)
)
lb2.place(x=21,y=240)
drplist.place(x=200, y=230)

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

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

Button(base,
text="Register",
width=10).place(x=180,y=3
80)
base.mainloop()
def Back():
   Home.destroy()
   root.deiconify()
def getdata(url):
   get= requests.get(url)
   return get.text
from_S_c = "1"
from_S_n = "Kolkata"

To_s_c = "2"
To_s_n = "Chennai"
url =
"https://www.railyatri.in/booking/trai
ns-between-
stations?from_code="+from_S_c+"&
from_name="+from_S_n+"+JN+&jo
urney_date=+Wed&src=tbs&to_code
=" + \
    To_s_c+"&to_name="+To_s_n + \
    "+JN+&user_id=-
1603228437&user_token=355740&ut
m_source=dwebsearch_tbs_search_tr
ains"
data = getdata(url)
soup = BeautifulSoup(data,
'html.parser')
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 is between
"+from_S_n+" and "+To_s_n)
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
```

7.2. FEATURE 2

```
class User(AbstractBaseUser):
    """
    User model.
    """

    USERNAME_FIELD = "email"

    REQUIRED_FIELDS = ["first_name", "last_name"]

    email = models.EmailField(
        verbose_name="E-mail",
        unique=True
    )

    first_name = models.CharField(
        verbose_name="First name",
        max_length=30
    )

    last_name = models.CharField(
        verbose_name="Last name",
        max_length=40
    )

    city = models.CharField(
        verbose_name="City",
        max_length=40
    )

    stripe_id = models.CharField(
        verbose_name="Stripe ID",
        unique=True,
        max_length=50,
        blank=True,
        null=True
    )

    objects = UserManager()

    @property
    def get_full_name(self):
        return f"{self.first_name} {self.last_name}"

    class Meta:
        verbose_name = "User"
        verbose_name_plural = "Users"


class Profile(models.Model):
    """
    User's profile.
    """

    phone_number = models.CharField(
        verbose_name="Phone  number",
        max_length=15
    )
    date_of_birth = models.DateField(
        verbose_name="Date of birth"
    )

    postal_code = models.CharField(
        verbose_name="Postal  code",
        max_length=10,
        blank=True
    )

    address = models.CharField(
        verbose_name="Address",
        max_length=255,
        blank=True
    )

    class Meta:
        abstract = True


class UserProfile(Profile):
    """
    User's profile model.
    """

    user = models.OneToOneField(
        to=User, on_delete=models.CASCADE, related_name="profile",
    )
    group = models.CharField(
        verbose_name="Group type",
        choices=GroupTypeChoices.choices(),
        max_length=20,
        default=GroupTypeChoices.EMPLOYEE.name,
    )

    def __str__(self):
        return self.user.email

    class Meta:

# user 1 - employer
user1, _ = User.objects.get_or_create(
    email="foo@bar.com",
    first_name="Employer",
    last_name="Testowy",
    city="Białystok",
)

user1.set_unusable_password()

group_name = "employer"

_profile1, _ = UserProfile.objects.get_or_create(
    user=user1,
    date_of_birth=datetime.now() - timedelta(days=6600),
    group=GroupTypeChoices(group_name).name,
    address="Myśliwska 14",
    postal_code="15-569",
    phone_number="+48100200300",
)

# user2 - employee
user2, _ = User.objects.get_or_create()
    email="bar@foo.com",
    first_name="Employee",
    last_name="Testowy",
    city="Białystok",
)

user2.set_unusable_password()

group_name = "employee"

_profile2, _ = UserProfile.objects.get_or_create()
    user=user2,
    date_of_birth=datetime.now() - timedelta(days=7600),
    group=GroupTypeChoices(group_name).name,
    address="Myśliwska 14",
    postal_code="15-569",
    phone_number="+48200300400",
)

response_customer = stripe.Customer.create()
    email=user.email,
    description=f"EMPLOYER - {user.get_full_name}",
    name=user.get_full_name,
    phone=user.profile.phone_number,
)
```

```
user1.stripe_id = response_customer.stripe_id
user1.save()

mcc_code, url = "1520", "https://www.softserveinc.com/"

response_ca = stripe.Account.create()
    type="custom",
    country="PL",
    email=user2.email,
    default_currency="pln",
    business_type="individual",
    settings={"payouts": {"schedule": {"interval": "manual", }}},
    requested_capabilities=["card_payments", "transfers", ],
    business_profile={"mcc": mcc_code, "url": url},
    individual={
        "first_name": user2.first_name,
        "last_name": user2.last_name,
        "email": user2.email,
        "dob": {
            "day": user2.profile.date_of_birth.day,
            "month": user2.profile.date_of_birth.month,
            "year": user2.profile.date_of_birth.year,
        },
        "phone": user2.profile.phone_number,
        "address": {
            "city": user2.city,
            "postal_code": user2.profile.postal_code,
            "country": "PL",
            "line1": user2.profile.address,
        },
    },
)

user2.stripe_id = response_ca.stripe_id
user2.save()

tos_acceptance = {"date": int(time.time()), "ip": user_ip},

stripe.Account.modify(user2.stripe_id, tos_acceptance=tos_acceptance)

passport_front = stripe.File.create(
    purpose="identity_document",
    file=_file, # ContentFile object
    stripe_account=user2.stripe_id,
)

individual = {
    "verification": {
        "document": {"front": passport_front.get("id"),},
        "additional_document": {"front": passport_front.get("id"),},
    }
}


stripe.Account.modify(user2.stripe_id, individual=individual)

new_card_source = stripe.Customer.create_source(user1.stripe_id,
source=token)
```

```
stripe.SetupIntent.create(
    payment_method_types=["card"],
    customer=user1.stripe_id,
    description="some description",
    payment_method=new_card_source.id,
)

payment_method =
stripe.Customer.retrieve(user1.stripe_id).default_source

payment_intent = stripe.PaymentIntent.create(
    amount=amount,
    currency="pln",
    payment_method_types=["card"],
    capture_method="manual",
    customer=user1.stripe_id, # customer
    payment_method=payment_method,
    application_fee_amount=application_fee_amount,
    transfer_data={"destination": user2.stripe_id}, # connect account
    description=description,
    metadata=metadata,
)

payment_intent_confirm = stripe.PaymentIntent.confirm(
    payment_intent.stripe_id, payment_method=payment_method
)

stripe.PaymentIntent.capture(
    payment_intent.id, amount_to_capture=amount
)
```

```
stripe.Balance.retrieve(stripe_account=user2.stripe_id)

stripe.Charge.create(
   amount=amount,
   currency="pln",
   source=user2.stripe_id,
   description=description
)

stripe.PaymentIntent.cancel(payment_intent.id)



      unique_together = ("user", "group")
@attr.s(frozen=True, cmp=False, hash=False, repr=True)
class UserSettings(MethodView):
   form = attr.ib(factory=settings_form_factory)
   settings_update_handler = attr.ib(factory=settings_update_handler)

   decorators = [login_required]

   def get(self):
      return self.render()

   def post(self):
      if self.form.validate_on_submit():
         try:
            self.settings_update_handler.apply_changeset(
               current_user, self.form.as_change()
            )
         except StopValidation as e:
            self.form.populate_errors(e.reasons)
            return self.render()
        except PersistenceError:
            logger.exception("Error while updating user settings")
            flash(_("Error while updating user settings"), "danger")
            return self.redirect()

        flash(_("Settings updated."), "success")
        return self.redirect()
    return self.render()

  def render(self):
      return render_template("user/general_settings.html",
form=self.form)

  def redirect(self):
      return redirect(url_for("user.settings"))


@attr.s(frozen=True, hash=False, cmp=False, repr=True)
class ChangePassword(MethodView):
  form = attr.ib(factory=change_password_form_factory)
  password_update_handler =
attr.ib(factory=password_update_handler)
  decorators = [login_required]

  def get(self):
      return self.render()

  def post(self):
    if self.form.validate_on_submit():
        try:
            self.password_update_handler.apply_changeset(
                current_user, self.form.as_change()
            )
        except StopValidation as e:
            self.form.populate_errors(e.reasons)
            return self.render()
        except PersistenceError:
            logger.exception("Error while changing password")
            flash(_("Error while changing password"), "danger")
            return self.redirect()

        flash(_("Password updated."), "success")
        return self.redirect()
    return self.render()

  def render(self):
    return render_template("user/change_password.html",
form=self.form)

  def redirect(self):
    return redirect(url_for("user.change_password"))


@attr.s(frozen=True, cmp=False, hash=False, repr=True)
class ChangeEmail(MethodView):
  form = attr.ib(factory=change_email_form_factory)
  update_email_handler = attr.ib(factory=email_update_handler)
  decorators = [login_required]
  def get(self):
    return self.render()

  def post(self):
    if self.form.validate_on_submit():
        try:
            self.update_email_handler.apply_changeset(
                current_user, self.form.as_change()
            )
        except StopValidation as e:
            self.form.populate_errors(e.reasons)
            return self.render()
        except PersistenceError:
            logger.exception("Error while updating email")
            flash(_("Error while updating email"), "danger")
            return self.redirect()

        flash(_("Email address updated."), "success")
        return self.redirect()
    return self.render()

  def render(self):
    return render_template("user/change_email.html", form=self.form)

  def redirect(self):
    return redirect(url_for("user.change_email"))
def berth_type(s):

  if s>0 and s<73:
    if s % 8 == 1 or s % 8 == 4:
        print (s), "is lower berth"
    elif s % 8 == 2 or s % 8 == 5:
        print (s), "is middle berth"
    elif s % 8 == 3 or s % 8 == 6:
        print (s), "is upper berth"
    elif s % 8 == 7:
        print (s), "is side lower berth"
    else:
        print (s), "is side upper berth"
  else:
    print (s), "invalid seat number"

# Driver code
s = 10
berth_type(s)      # 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
            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\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\t\t\t\tENTER THE
PASSWORD: ")
            os.system('cls')
            if (j==r):
                x='y'
                while (x.lower()=='y'):
                    fout=open("tr1details.dat","ab")
                    tr.getinput()
                    dump(tr,fout)
                    fout.close()
                    print"\n\n\n\n\n\n\n\n\n\n\n\t\t\tUPDATING TRAIN LIST
PLEASE WAIT . .",
                    time.sleep(1)
                    print ("."),
                    time.sleep(0.5)
                    print ("."),
                    time.sleep(2)
                    os.system('cls')
                    print "\n\n\n\n\n\n\n\n\n\n\n"
                    x=raw_input("\t\tDO YOU WANT TO ADD ANY MORE
TRAINS DETAILS ? ")
                    os.system('cls')
                continue
            elif(j<>r):
                print"\n\n\n\n\n"
                print "WRONG PASSWORD".center(80)
        elif ch==2:
            fin=open("tr1details.dat",'rb')
            if not fin:
                print "ERROR"
            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\t\t\t\t\t\tLOADI
NG. .",
        time.sleep(1)
        print ("."),
        time.sleep(0.5)
        print (".")
        time.sleep(2)
        os.system('cls')
        if ch==1:
          j="*****"
          r=raw_input("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\tENTER THE
PASSWORD: ")
          os.system('cls')
          if (j==r):
            x='y'
            while (x.lower()=='y'):
              fout=open("tr1details.dat","ab")
              tr.getinput()
              dump(tr,fout)
              fout.close()
              print"\n\n\n\n\n\n\n\n\n\n\n\t\t\tUPDATING TRAIN LIST
PLEASE WAIT . .",
              time.sleep(1)
              print ("."),
              time.sleep(0.5)
              print ("."),
              time.sleep(2)
              os.system('cls')
              print "\n\n\n\n\n\n\n\n\n\n\n"
              x=raw_input("\t\tDO YOU WANT TO ADD ANY MORE
TRAINS DETAILS ? ")
              os.system('cls')
            continue
          elif(j<>r):
            print"\n\n\n\n\n"
            print "WRONG PASSWORD".center(80)
        elif ch==2:
          fin=open("tr1details.dat",'rb')
          if not fin:
                print "ERROR"
            tick.display()
        elif ch==6:
            quit()

        raw_input("PRESS ENTER TO GO TO BACK
MENU".center(80))
        os.system('cls')

menu()
sender_email = "my@gmail.com"
receiver_email = "your@gmail.com"
password = input("Type your password and press enter:")

message = MIMEMultipart("alternative")
message["Subject"] = "multipart test"
message["From"] = sender_email
message["To"] = receiver_email

# Create the plain-text and HTML version of your message
text = """\
Hi,
How are you?
Real Python has many great tutorials:
www.realpython.com"""
html = """\
<html>
  <body>
    <p>Hi,<br>
       How are you?<br>
```

```
      <a href="http://www.realpython.com">Real Python</a>
      has many great tutorials.
    </p>
  </body>
</html>
"""

# Turn these into plain/html MIMEText objects
part1 = MIMEText(text, "plain")
part2 = MIMEText(html, "html")

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

# Create secure connection with server and send email
context = ssl.create_default_context()
with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as
server:
    server.login(sender_email, password)
    server.sendmail(
        sender_email, receiver_email, message.as_string()
    )
subject = "An email with attachment from Python"
body = "This is an email with attachment sent from Python"
sender_email = "my@gmail.com"
receiver_email = "your@gmail.com"
password = input("Type your password and press enter:")
```

```
# Create a multipart message and set headers
message = MIMEMultipart()
message["From"] = sender_email
message["To"] = receiver_email
message["Subject"] = subject
message["Bcc"] = receiver_email  # Recommended for mass emails

# Add body to email
message.attach(MIMEText(body, "plain"))

filename = "document.pdf"  # In same directory as script

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

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

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

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

```
text = message.as_string()

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

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

# Enter valid pnr_number
pnr_number = "6515483790"

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

# get method of requests module
# return response object
response_ob = requests.get(complete_url)

# json method of response object convert
# json format data into python format data
result = response_ob.json()

# now result contains list
# of nested dictionaries
if result["response_code"] == 200:
```

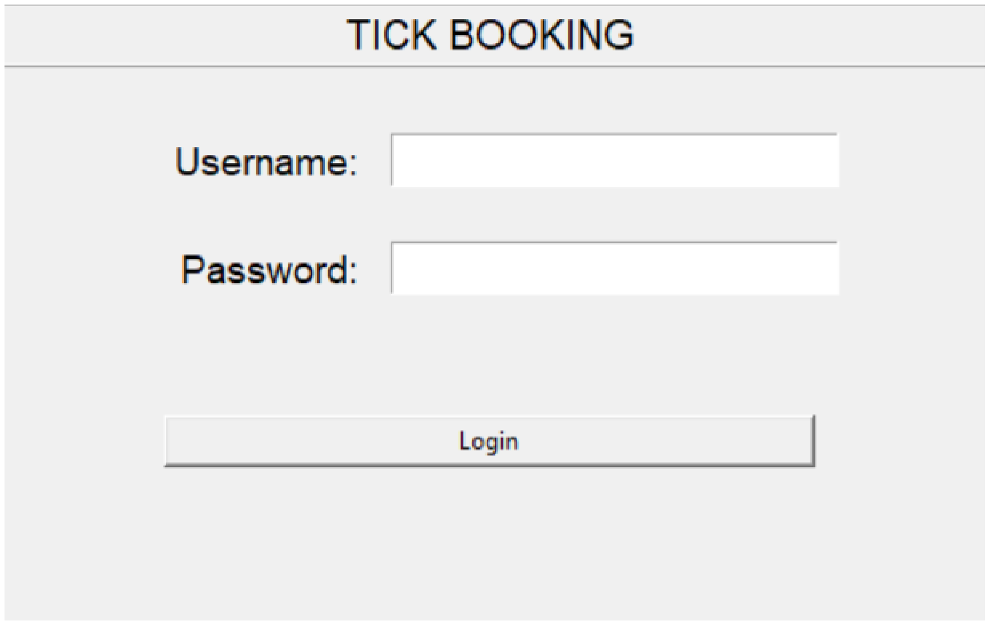
```
                    # train name is extracting
                    # from the result variable data
                    train_name = result["train"]["name"]

                    # train number is extracting from
                    # the result variable data
                    train_number = result["train"]["number"]

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

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

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

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

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

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

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

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

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

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

# looping through passenger list
```

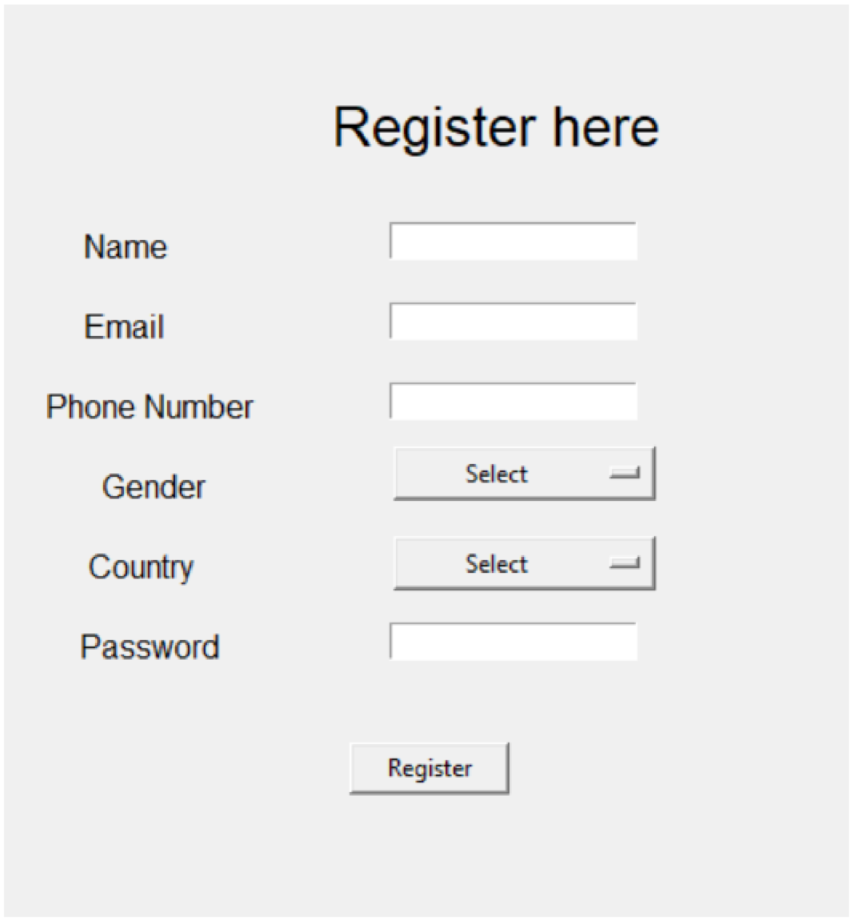
```
                    for passenger in passengers_list:

                        # store the value or data
                        # of "no" key in variable
                        passenger_num = passenger["no"]

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

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

                        # print following values
                        print(" passenger number : " + str(passenger_num)
                              + "\n current status : " + str(current_status)
                              + "\n booking_status : " + str(booking_status))

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

8.RESULTS

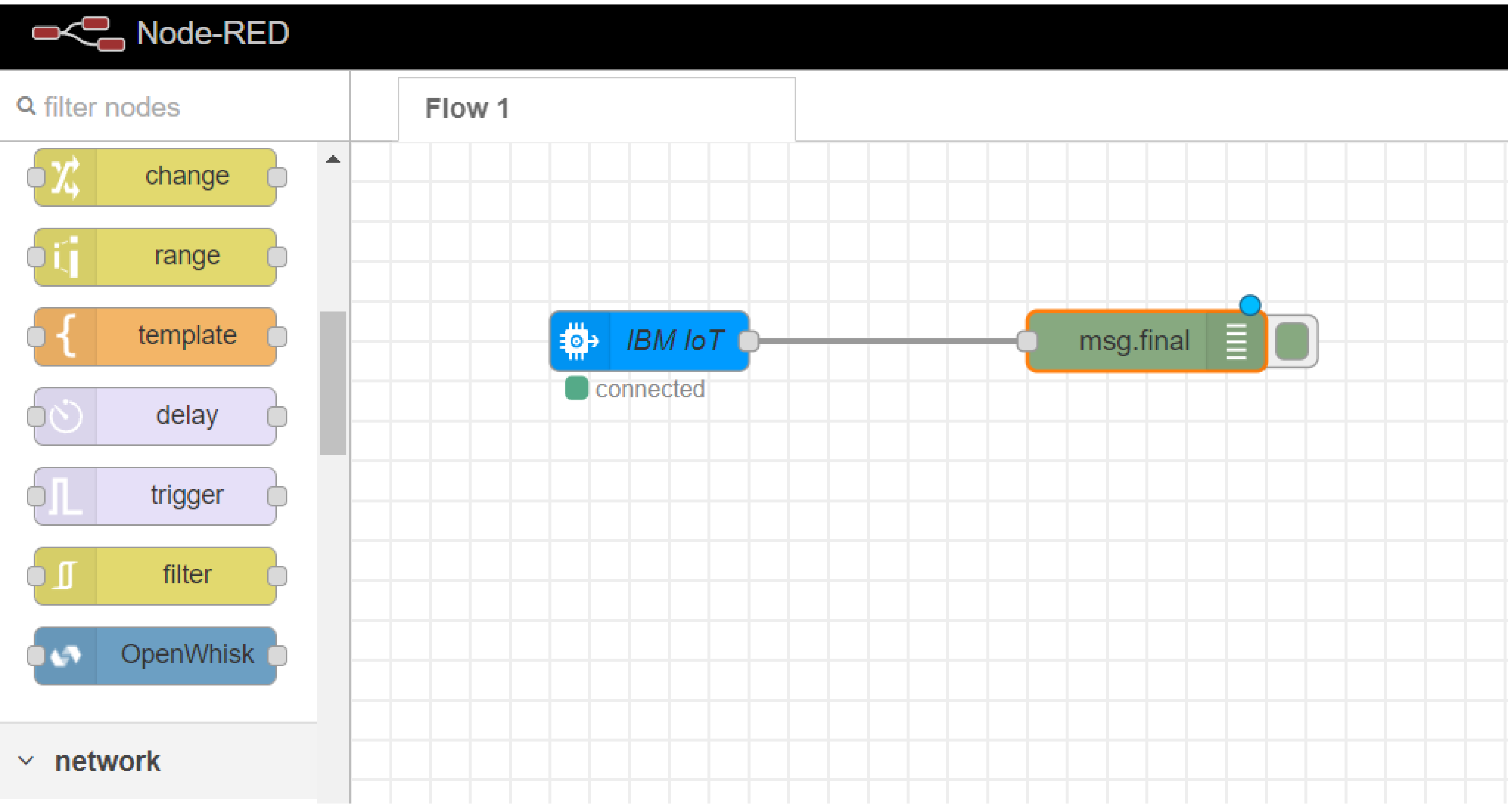
LOGIN PAGE:



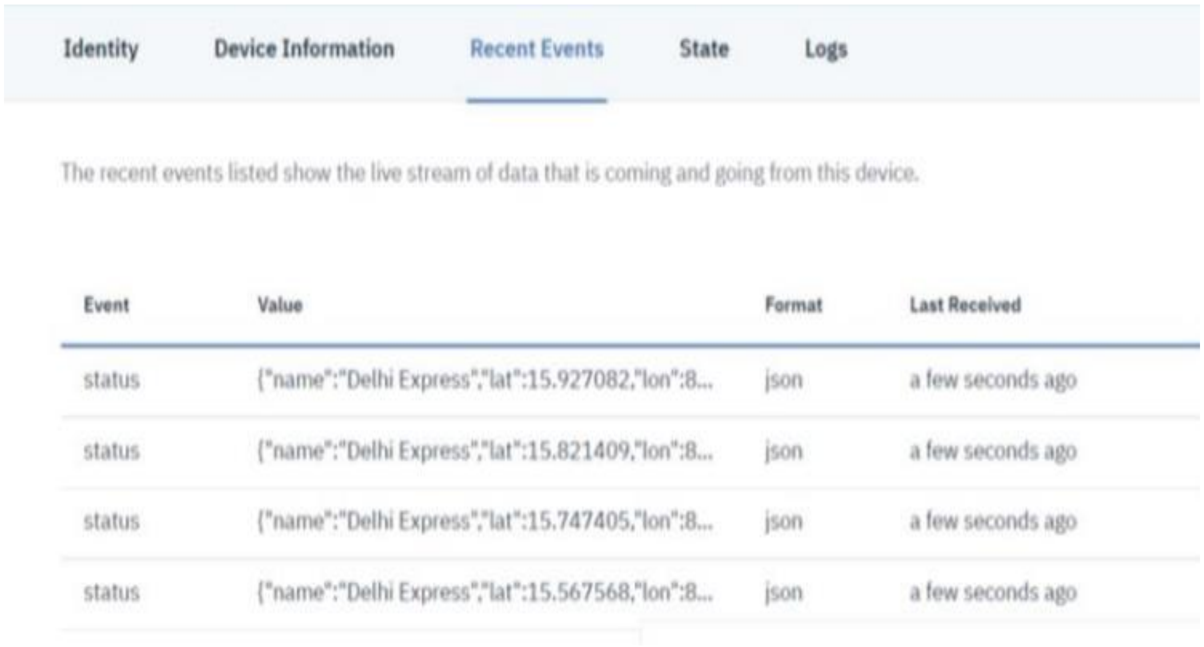
REGISTRATION PAGE:



TRAIN LOCATION:

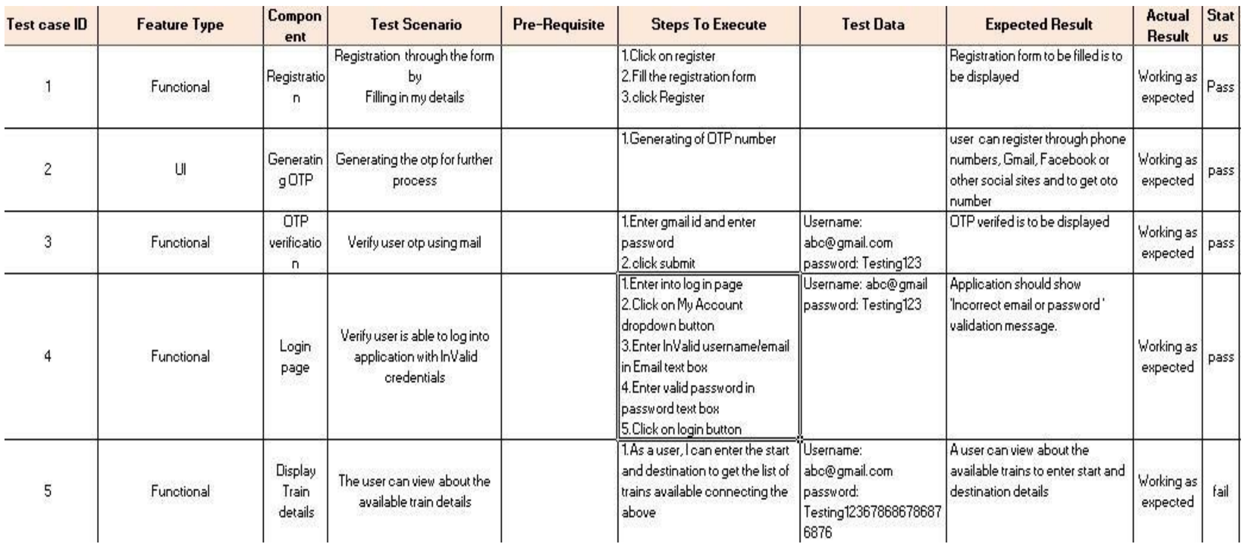


IBM IOT WATSON:

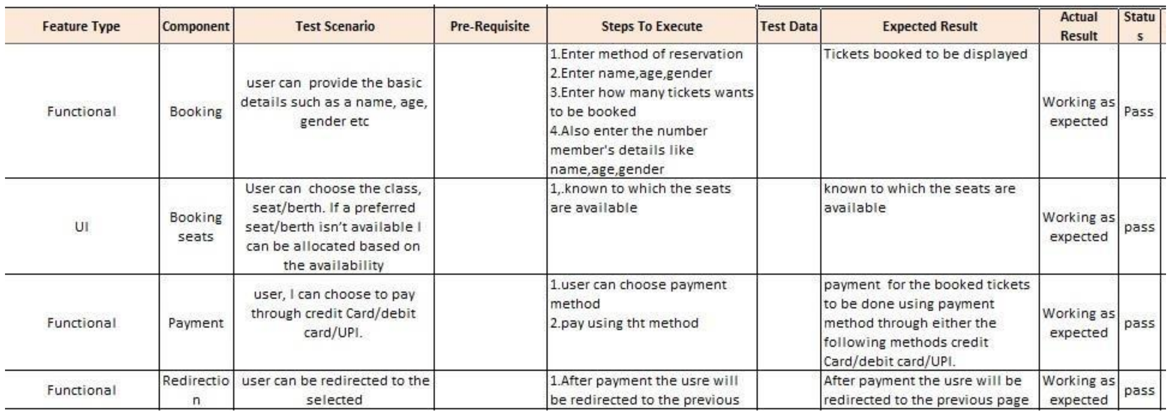


The recent events listed show the live stream of data that is coming and going from this device.

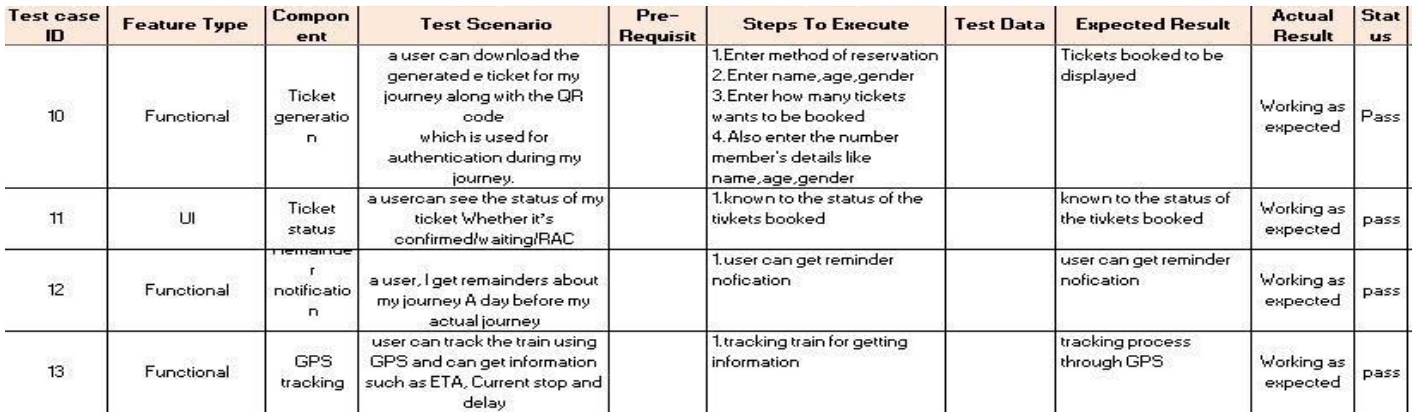
Event	Value	Format	Last Received
status	{"name":"Delhi Express","lat":15.927082,"lon":8...	json	a few seconds ago
status	{"name":"Delhi Express","lat":15.821409,"lon":8...	json	a few seconds ago
status	{"name":"Delhi Express","lat":15.747405,"lon":8...	json	a few seconds ago
status	{"name":"Delhi Express","lat":15.567568,"lon":8...	json	a few seconds ago



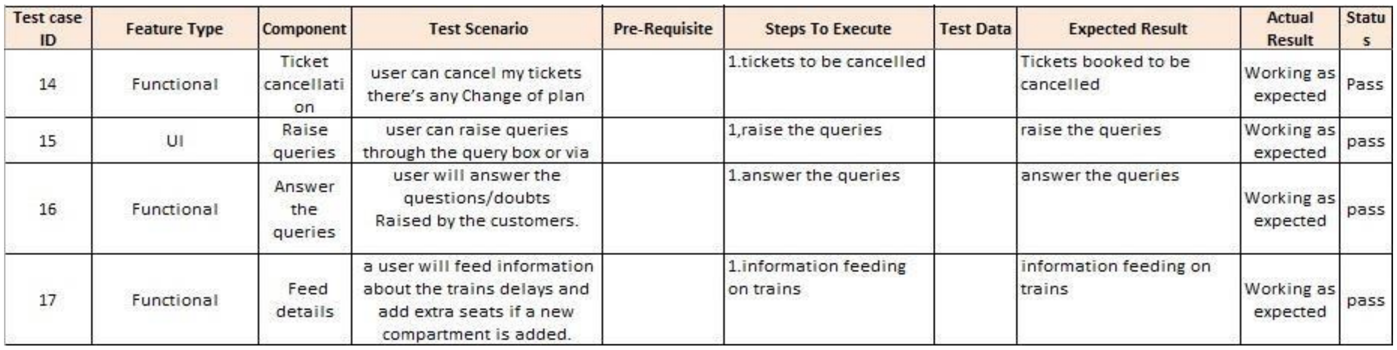
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
1	Functional	Registration	Registration through the form by Filling in my details		1.Click on register 2.Fill the registration form 3.click Register		Registration form to be 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: Testing123678686786876876	A user can view about the available trains to enter start and destination details	Working as expected	fail



Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
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	Redirection	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
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



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
14	Functional	Ticket cancellation	user can cancel my tickets there's any Change of plan		1.tickets to be cancelled		Tickets booked to be cancelled	Working as expected	Pass
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
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
17	Functional	Feed details	a user will feed information about the trains delays and add extra seats if a new compartment is added.		1.information feeding on trains		information feeding on trains	Working as expected	pass

9.CONCLUSION

Android application which will be used for the checking a ticket and it will make it easy for ticket checker to check whether ticket is valid or invalid. With the use of QR codes the problems for ticket reservations system are overcome. The implementation of this project enables to develop a QR code based ticketing system which will make verification become easy. The Ticket details will be encrypted and stored in the database. The ticket checker can verify the passenger's ticket easily.