# SMART SOLUTIONS FOR RAILWAYS

## A PROJECT REPORT

*Submitted by*


**NAVEEN TR** **(Roll No: 727819TUCS136)**

**NITHINRAJ R** **(Roll No: 727819TUCS137)**

**NITHINRAAJ J** **(Roll No: 727819TUCS138)**

**KAVI S** **(Roll No: 727819TUCS108)**


*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

**IN**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**(Accredited by NBA)**


**SRI KRISHNA COLLEGE OF TECHNOLOGY**

**An Autonomous Institution | Accredited by NAAC with 'A' Grade**

**Affiliated to Anna University | Approved by AICTE**

**KOVAIPUDUR, COIMBATORE 641042**


**NOVEMBER 2022**

**PROJECT REPORT**

# INTRODUCTION

# 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 humanto-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

# LITERATURE SURVEY

# 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. D. Hesse, “Rail Inspection Using Ultrasonic Surface Waves” Thesis, Imperial College of London, 2007.

2. Md. Reya Shad Azim1 , Khizir Mahmud2 and C. K. Das. Automatic railway 6 track switching system, International Journal of Advanced Technology, Volume 54, 2014.

3. S. Somalraju, V. Murali, G. saha and V. Vaidehi, “Title-robust railway crack detection scheme using LED (Light Emitting Diode) - LDR (Light Dependent Resistor) assembly IEEE 2012.

4. S. Srivastava, R. P. Chourasia, P. Sharma, S. I. Abbas, N. K. Singh, “Railway Track Crack detection vehicle”, IARJSET, Vol. 4, pp. 145-148, Issued in 2, Feb 2017.

5. U. Mishra, V. Gupta, S. M. Ahzam and S. M. Tripathi, “Google Map Based Railway Track Fault Detection Over the Internet”, International Journal of Applied Engineering Research, Vol. 14, pp. 20-23, Number 2, 2019.

6. R. A. Raza, K. P. Rauf, A. Shafeeq, “Crack detection in Railway track using Image processing”, IJARIIT, Vol. 3, pp. 489-496, Issue 4, 2017.

7. N. Bhargav, A. Gupta, M. Khirwar, S. Yadav, and V. Sahu, “Automatic Fault Detection of Railway Track System Based on PLC (ADOR TAST)”, International Journal of Recent Research Aspects, Vol. 3, pp. 91-94, 2016.

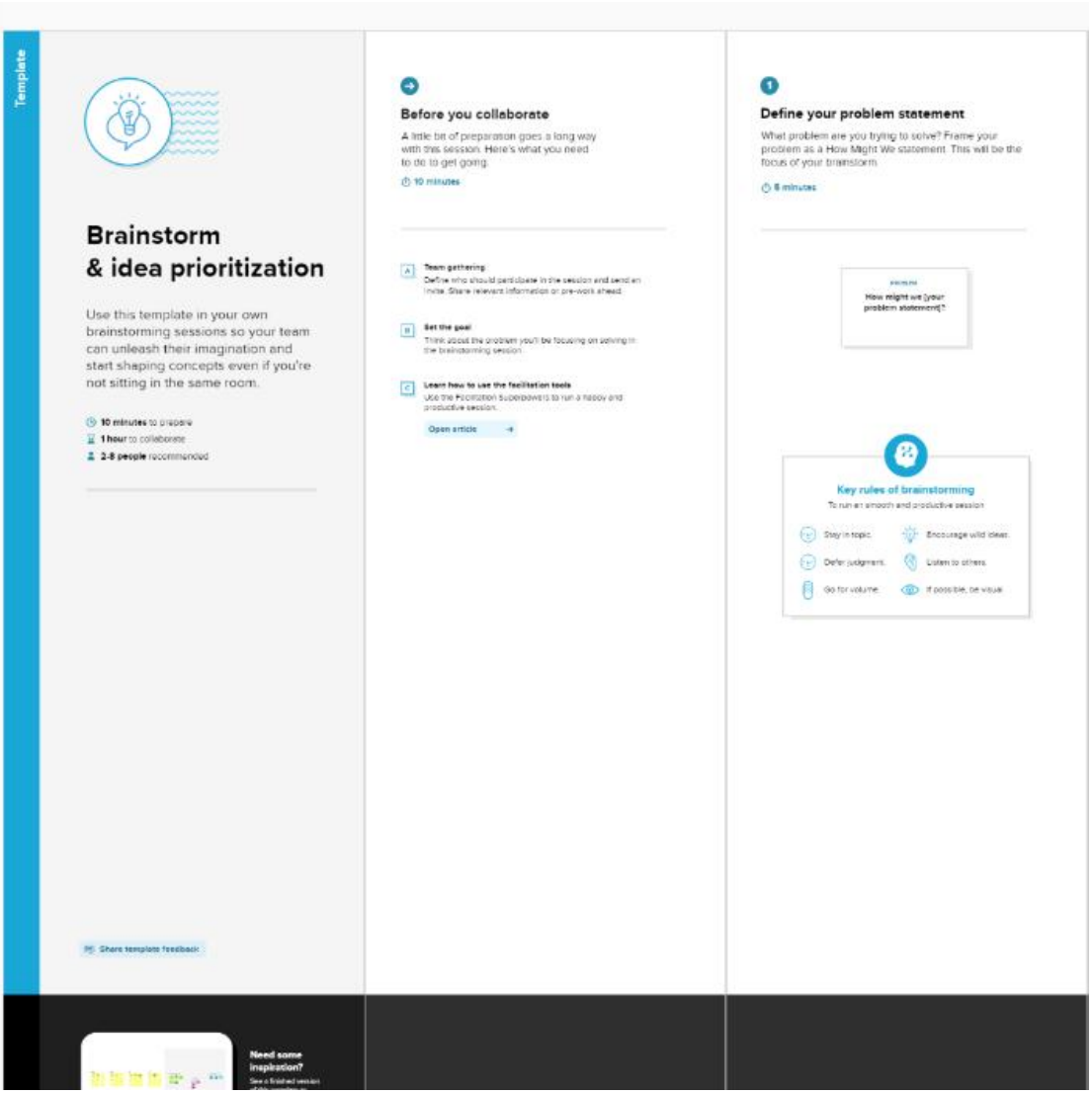
## 2.3 PROBLEM STATEMENT DEFINITION

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”.

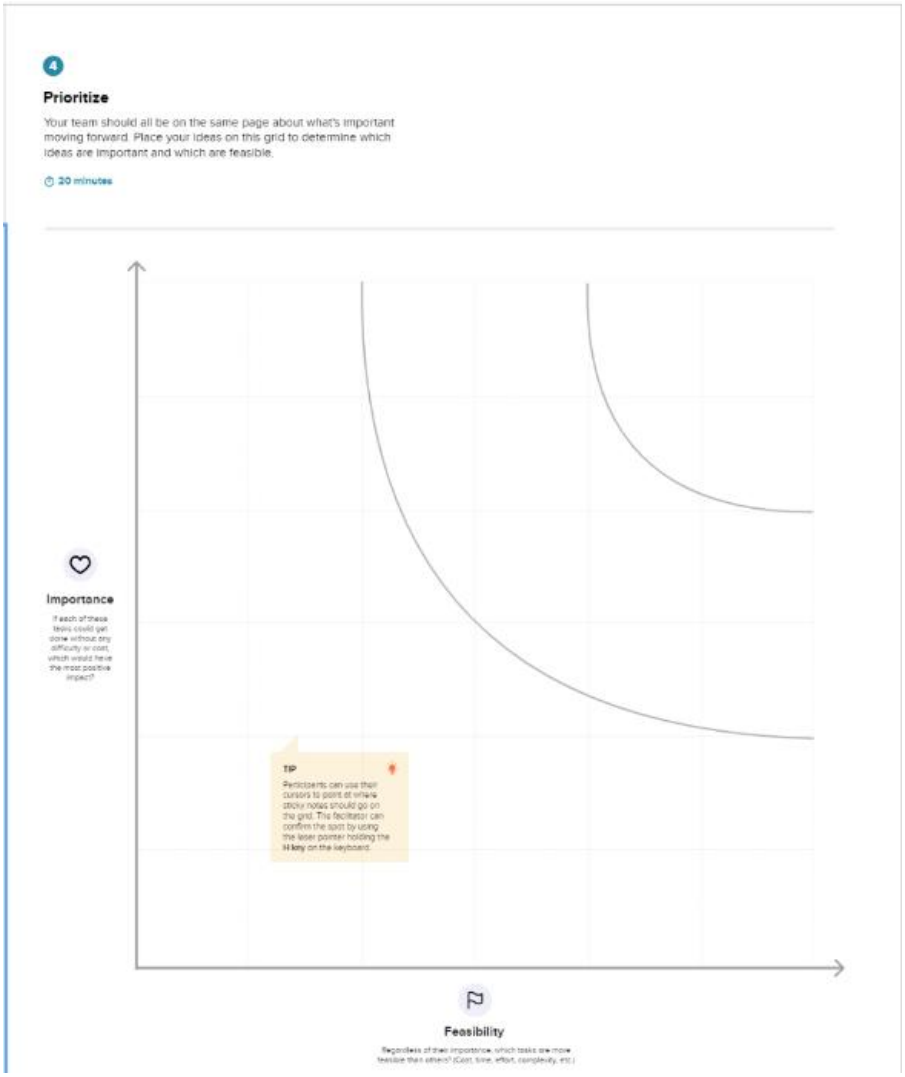
# IDEATION AND PROPOSED SOLUTION

# 3. IDEATION AND PROPOSED SOLUTON

## 3.1 EMPATHY MAP CANVAS



## 3.2 IDEATION & BRAINSTORMING


## 3.3

## PROPOSED SOLUTION

| S.NO | PARAMETER | DESCRIPTION |
|---|---|---|
| 1 | Problem Statement (Problem to be solved) | In order to satisfy the passengers, the Railways provides various services to its passengers But, the passengers can face some problems |
| 2 | Idea / Solution description | The idea is to minimize the ticket booking problems among the passengers by providing Online mode of booking rather than papers. . In queues in front of the ticket counters in railway stations have been drasticallyincreased over the time. |
| 3 | Novelty / Uniqueness | Online mode of booking is most common and so ease of access to everyone that makes more efficient uniqueness of utilizing the technique. People can book their ticket through online and they get a QR code through SMS |

| | | |
|---|---|---|
| **4** | Social Impact / Customer Satisfaction | Customers for sure they get satisfied as they are in the fast roaming world this technique makes more easier for travelling passengers. A web page is designed in which the user can book tickets and will be provided with the QR code, which will be shown to the ticket collector and by scanning the QR code the ticket collector will get the passenger details |
| **5** | Business Model (Revenue Model) | A web page is designed in which the user can book tickets and will be provided with the QR code, which will be shown to the ticket collector and by scanning the QR code the ticket collector will get the passenger details. The booking details of the user will be stored in the database, which can be retrieved any time |
| **6** | Scalability of the Solution | The scalability of this solution is most feasible among the passengers who are willing to travel. No need of taking printout |

| | | |
|---|---|---|
| | | Counter ticket has to be handled with care, but SMS on mobile is enough. No need to taking out wallet and showing your ticket to TTR just tell your name to TTR that you are a passenger with valid proof |

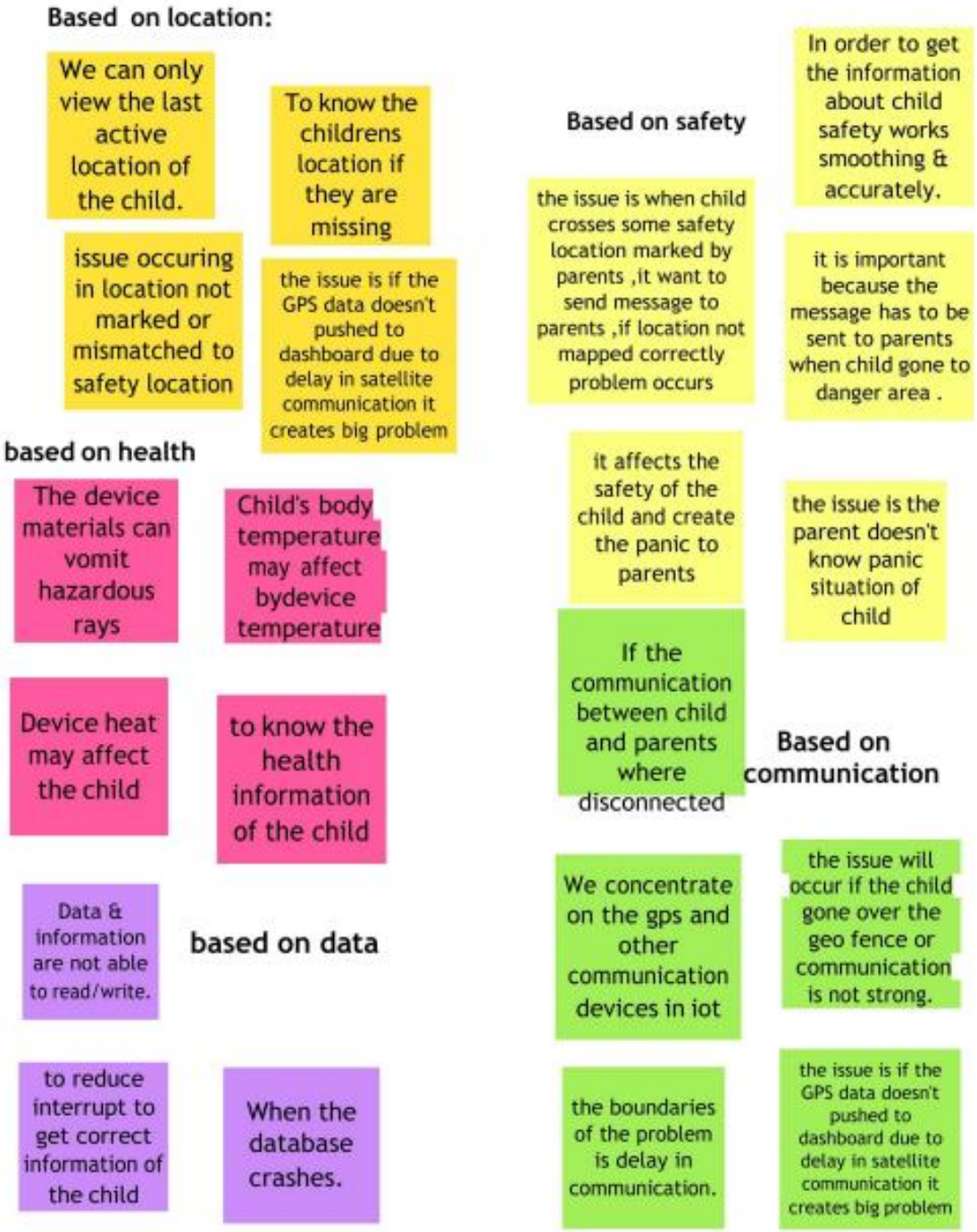
## 3.4 PROBLEM SOLUTION FIT

3

## Group ideas

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

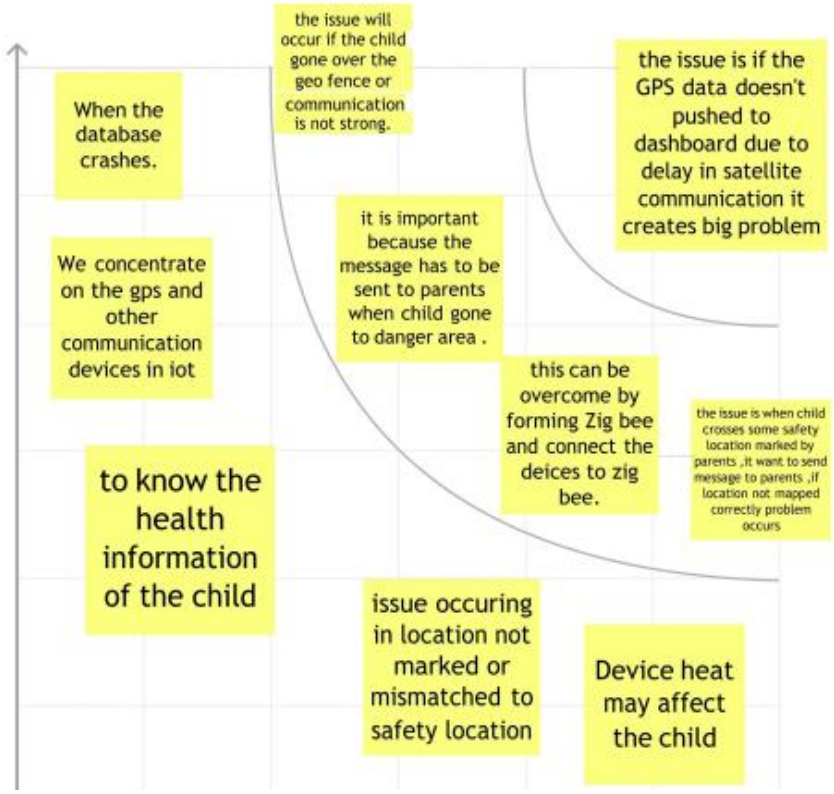
20 minutes



## Prioritize

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

20 minutes



# REQUIREMENT ANALYSIS

# 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<br>• 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 |

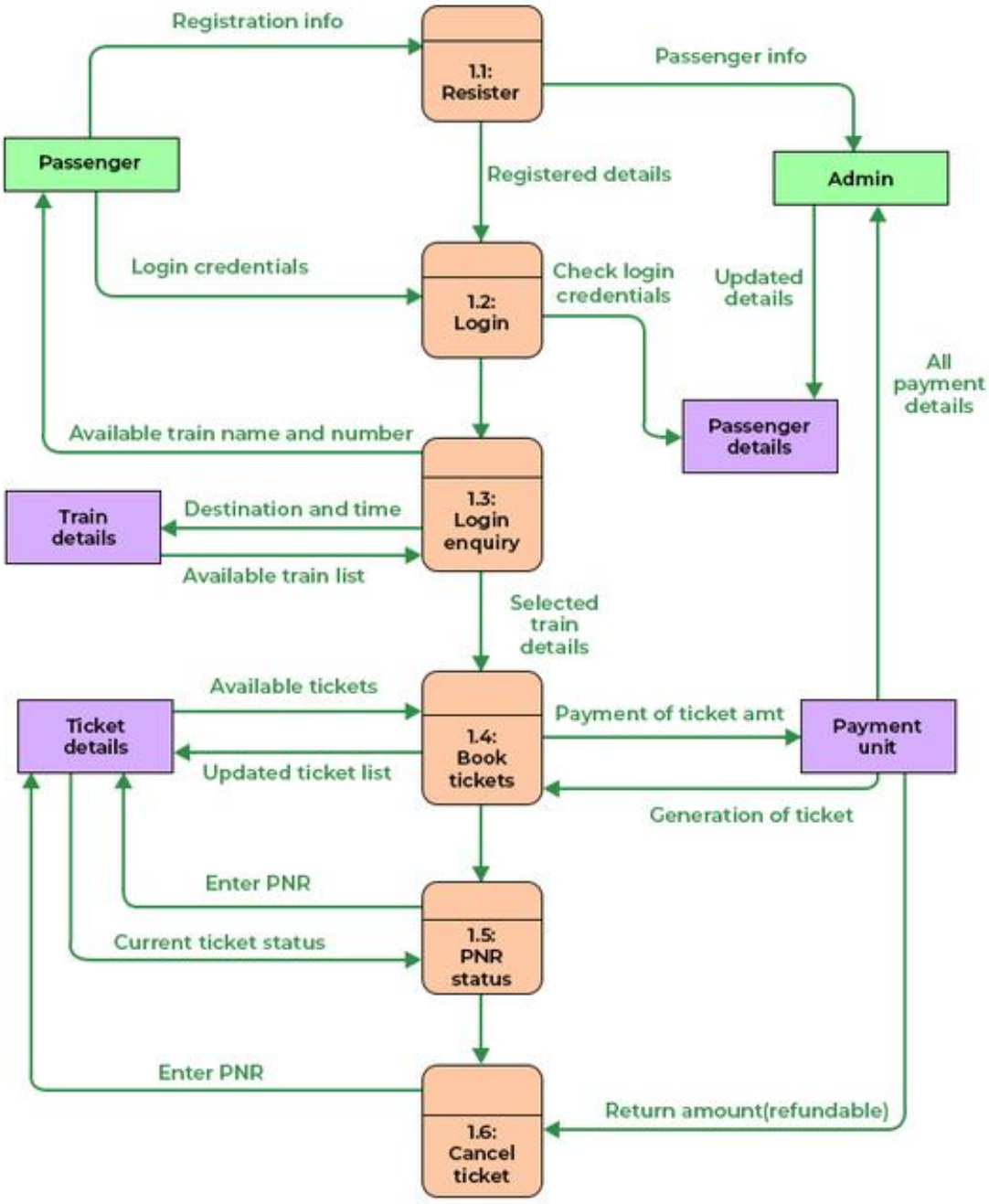
## 4.2 NON-FUNCTIONAL REQUIREMENTS

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | Usability | Search results should populate within acceptable time limits |
| NFR-2 | Security | System should visually confirm as well as send booking confirmation to the user's contact |
| NFR-3 | Reliability | System should accept payments via different payment methods, like<br>PayPal, wallets, cards, vouchers, etc |
| NFR-4 | Performance | Search results should populate within acceptable time limits |
| NFR-5 | Availability | User should be helped appropriately to fill in the mandatory fields, incase of invalid input |
| NFR-6 | Scalability | Use of captcha and encryption to<br>avoid bots from booking tickets |

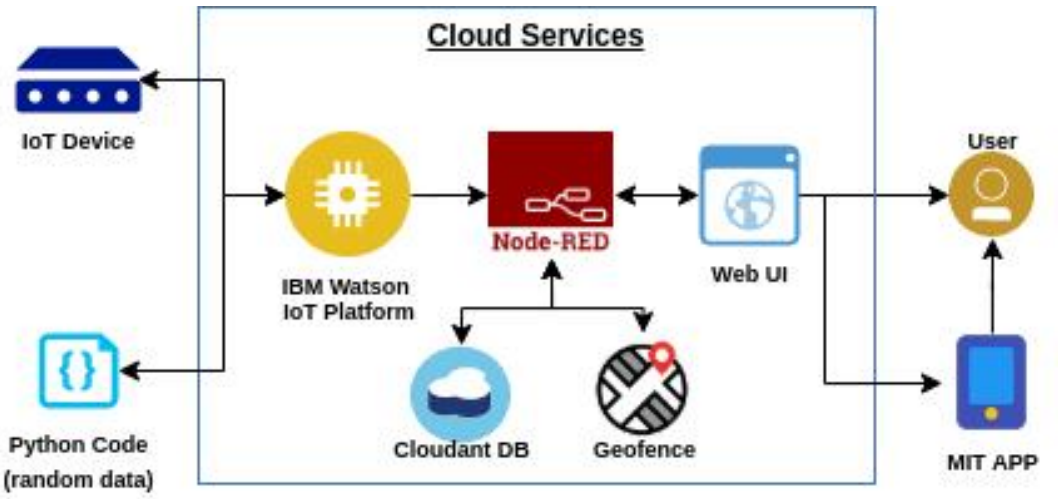
# PROJECT DESIGN

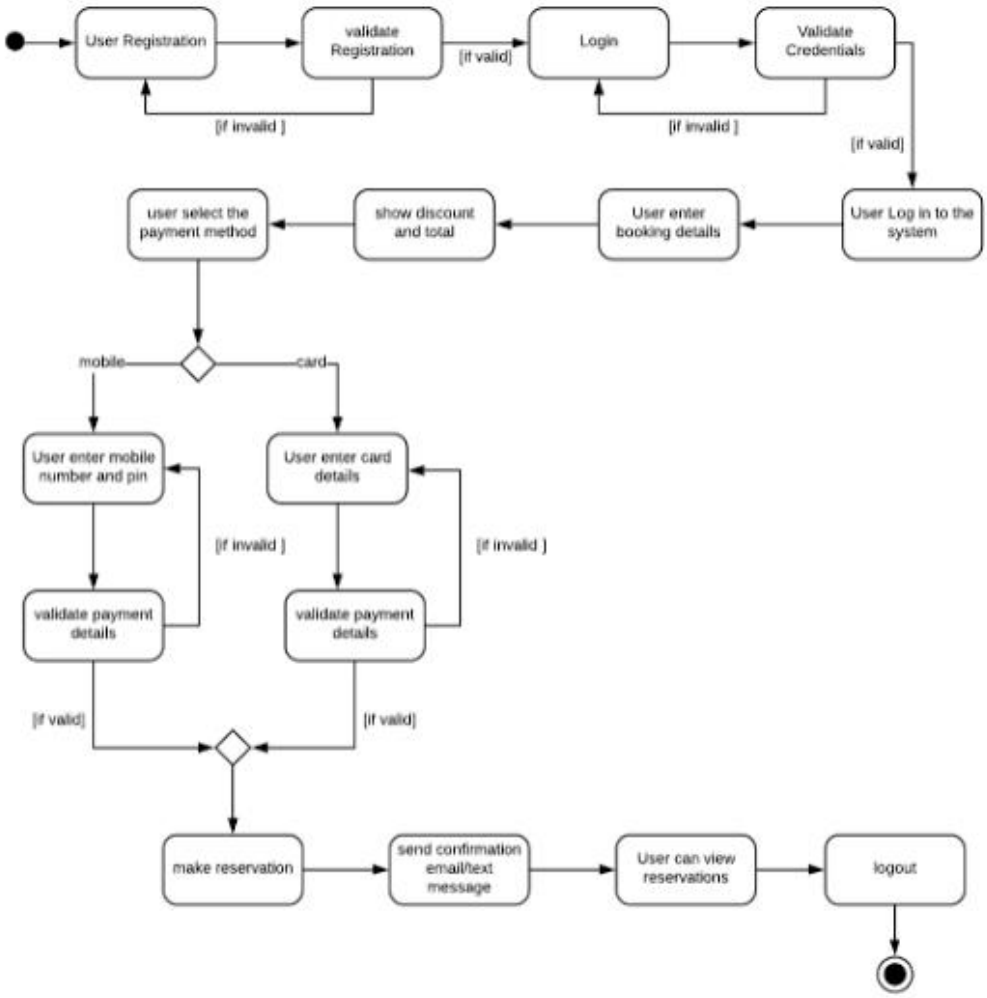
# 5. PROJECT DESIGN

## 5.1 DATA FLOW DIAGRAMS



## 5.2 SOLUTION & TECHNICAL ARCHITECTURE



## 5.3 USER STORIES

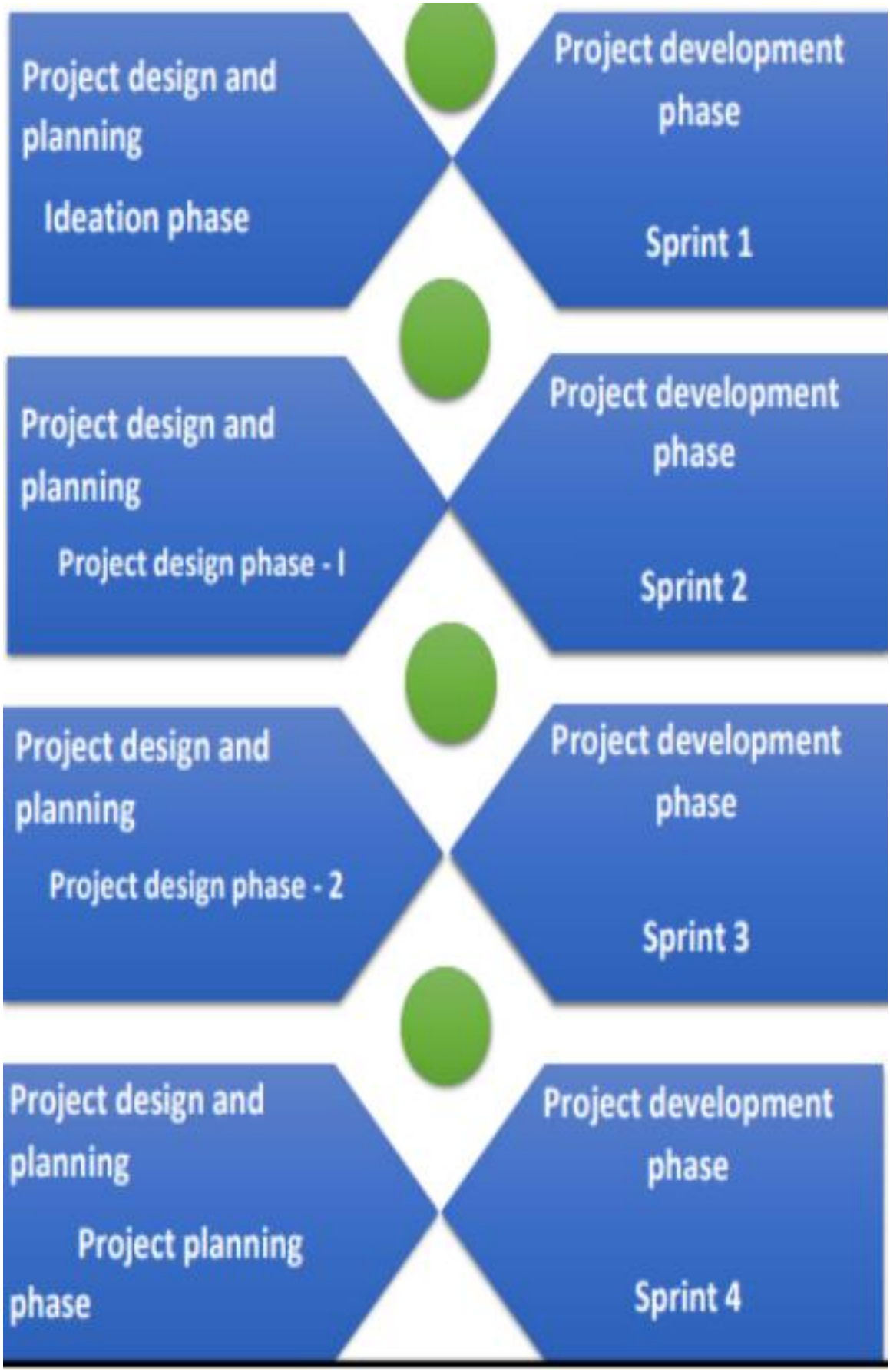
| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|---|---|---|---|---|---|---|
| Customer<br><br>(Mobile user, Web user) | Registration | USN-1 | As a user, I can register through the form by Filling in my details | I can register and create my account / dashboard | High | Sprint-1 |
| | | USN-2 | As a user, I can register through phone numbers, Gmail, Facebook or other social sites | I can register & create my dashboard with Facebook login or other social sites | High | Sprint-2 |
| | Conformation | USN-3 | As a user, I will receive confirmation through email or OTP once registration is successful | I can receive confirmation email & click confirm. | High | Sprint-1 |
| | Authentication/Login | USN-4 | As a user, I can login via login id and password or through OTP received on register phone number | I can login and access my account/dashboard | High | 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 | I can view the train details (name & number), corresponding routes it passes through based on the start and destination entered. | High | Sprint-1 |
| | Booking | USN-6 | As a use, I can provide the basic details such as a name, age, gender etc... | I will view, modify or confirm the details enter. | High | Sprint-1 |
| | | 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. | I will view, modify or confirm the seat/class berth selected | High | Sprint-1 |
| | Payment | USN-8 | As a user, I can choose to pay through credit Card/debit card/UPI. | I can view the payment Options available and select my desirable choice To proceed with the payment | High | Sprint-1 |
| | | USN-9 | As a user, I will be redirected to the selected Payment gateway and upon successful | I can pay through the payment portal and confirm the booking if any changes need to | High | Sprint-1 |

| 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 |

# PROJECT PLANNING AND SCHEDULING

# 6. PROJECT PLANNING AND SCHEDULING

## 6.1. SPRINT PLANNING & ESTIMATION


Project design and planning
Ideation phase
Project development phase
Sprint 1
Project design and planning
Project design phase - I
Project development phase
Sprint 2
Project design and planning
Project design phase - 2
Project development phase
Sprint 3
Project design and planning
Project planning phase
Project development phase
Sprint 4

## 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 |

# CODING AND SOLUTIONING

# 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

## PROGRAMM:

```
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) 30

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() 31

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: ")32

s.sendmail('&&&&&&&&&&&&',emailid,msg)

a = input("Enter Your OTP >>: ")

if a == OTP:

print("Verified")

else:

print("Please Check your OTP again")
```

# TESTING

# 8.1. TEST CASES

## SPRINT - 1

| Testcase ID | Feature Type | Component | TestScenario | Pre-Requisite | Steps To Execute | Test Data | ExpectedResult | Actual Result | Status | Commnets | TC for Automation(Y/N) | BUG ID | Executed By |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | UI | generating OTP | generating the otp for further process | | | | numbers, Gmail, Facebook or other social sites and to get oto number | Working as expected | pass | | | | NAVEEN TR |
| 3 | Functional | OTP verification | Verify user otp using mail | | 1.Enter gmail id and enter password<br>2.click submit | Username: abc@gmail.com<br>password: Testing123 | OTP verified is to be displayed | Working as expected | pass | | | | KAVI S |
| 4 | Functional | Login page | Verify user is able to log into application with Invalid credentials | | 1.Enter into log in page<br>2.Click on My Account dropdown button<br>3.Enter invalid username/email in Email text box<br>4.Enter valid password in password text box<br>5.Click on login button | Username: abc@gmail<br>password: Testing123 | Application should show 'Incorrect email or password' validation message. | Working as expected | pass | | | | NITHINRAJ R |
| 5 | Functional | Display Train details | The user can view about the available train details | | 1.As a user, I can enter the startand destination to get the list of trains available connecting the above | Username: abc@gmail.com<br>password:<br>Testing123678686786876876 | A user can view about the available trains to enter start and destination details | Working as expected | fail | | | | NITHINRAAJ J |

## SPRINT - 2

| Test case ID | Feature Type | Component | Test Scenario | Pre-Requisite | Steps To Execute | Test Data | Expected Result | Actual Result | Status | Commnets | TC for Automation(Y/N) | BUG ID | Executed By |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Functional | Booking | user can provide the basic details such as a name, age, gender etc | | 1.Enter method of reservation<br>2.Enter name,age,gender<br>3.Enter how many tickets wants to be booked<br>4.Also enter the number member's details like name,age,gender | | Tickets booked to be displayed | Working as expected | Pass | | | | NAVEEN T R |
| 2 | 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 | | 1.,known to which the seats are available | | known to which the seats are available | Working as expected | pass | | | | NITHINRAAJ J |
| 3 | Functional | Payment | user, I can choose to pay through credit Card/debit card/UPI. | | 1.user can choose payment method<br>2.pay using tht method | | payment for the booked tickets to be done using payment method through either the following methods credit Card/debit | Working as expected | pass | | | | KAVI S |
| 4 | Functional | Redirection | user can be redirected to the selected | | 1.After payment the usre will be redirected to the previous page | | After payment the usre will be redirected to the previous page | Working as expected | pass | | | | NITHINRAJ R |

## SPRINT - 3

| Test case ID | Feature Type | Component | Test Scenario | Pre-Requisite | Steps To Execute | Test Data | Expected Result | Actual Result | Status | Commnets | TC for Automation(Y/N) | BUG ID | Executed By |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | during my journey. | | 4.Also enter the number member's details like name,age,gender | | | | | | | | |
| 2 | 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 | | | | NAVEEN TR |
| 3 | Functional | Remainder notificatio n | 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 | | | | NITHINRAJ R |
| 4 | Functional | GPS tracking | user can track the train using GPS and can get information such as ETA, Current stop and | | 1.tracking train for getting information | | tracking process through GPS | Working as expected | pass | | | | NITHINRAAJ J |

## SPRINT - 4

| Test case ID | Feature Type | Component | Test Scenario | Pre-Requisite | Steps To Execute | Test Data | Expected Result | Actual Result | Status | Commnets | TC for Automation(Y/N) | BUG ID | Executed By |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Functional | Ticket cancellatio | 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 | | | | NITHINRAJ R |
| 2 | UI | Raise queries | user can raise queries through the query box or via mail. | | 1.raise the queries | | raise the queries | Working as expected | pass | | | | NITHINRAAJ J |
| 3 | 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 | | | | KAVI S |
| 4 | 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 | | | | NAVEEN TR |

# RESULTS

# 9. RESULTS

## 9.1. PERFORMANCE METRICS

# ADVANTAGES &DISADVANTAGES

# 10. ADVANTAGES &DISADVANTAGES

## 10.1 ADVANTAGES

- Openness – compatibility between different system modules, potentially from different vendors;
- Orchestration – ability to manage large numbers of devices, with full visibility over them;
- Dynamic scaling – ability to scale the system according to the application needs, through resource virtualization and cloud operation;
- Automation – ability to automate parts of the system monitoring application, leading to better performance and lower operation costs.

## 10.2.DISADVANTAGES

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

# CONCLUSION

## 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.

# FUTURE SCOPE

## 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.

# APPENDIX

# 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_required46

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, 47

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() :48

# 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 >>: ")49

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)50
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="")51
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():52

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="+fro
m_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 string53
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 = []54
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=4056
)
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
)57
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",
)58
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",59
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,
)60
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,61
},
},
)
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)62
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
)63
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:64
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):65

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]66

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<73:67
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+=168
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.C
ounter)
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.__destination69
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())70

# 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:71

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)72
# 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')73
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"74
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):

print75

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):76
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:77

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"78

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."80

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\tLO
ADI

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 ("."),81

time.sleep(0.5)

print ("."),
```

```
time.sleep(2)

os.system('cls')

print "\n\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")82

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="83

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
print84
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
```

```
print85

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') 86
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")87

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)88
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:89
print
print "="*80
print " \t\t\t\t RAILWAY"
print
print "="*80
print
print "\t\t\t1. **UPDATE TRAIN DETAILS."
print
print "\t\t\t2. TRAIN DETAILS. "
print
print "\t\t\t3. RESERVATION OF TICKETS."
print
print "\t\t\t4. CANCELLATION OF TICKETS. "
print
print "\t\t\t5. DISPLAY PNR STATUS."
```

```
print

print "\t\t\t6. QUIT."

print"** - office use....."

ch=int(raw_input("\t\t\tENTER YOUR CHOICE : "))

os.system('cls')

print

"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\tLO
ADI

NG. .",

time.sleep(1)

print ("."),

time.sleep(0.5)

print (".")

time.sleep(2)

os.system('cls')

if ch==1:90

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:91

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>92

<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:")93

# 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)94
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"] == 200:95

# 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 list97

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 GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-11482-1659330842