IBM PROJECT REPORT

COLLEGE NAME	**SRM EASWARI ENGINEERING COLLEGE**
TEAM ID	**PNT2022TMID54425**
PROJECT NAME	**SMART SOLUTION FOR RAILWAY SYSTEM**

TEAM MEMBERS:

- **SHIYAM R**
- **SOORAJ I**
- **SRIVIDYA R**
- **VIGNESH B**

Project Report Format

1. **INTRODUCTION**
   1.1 Project Overview
   1.2 Purpose
2. **LITERATURE SURVEY**
   2.1 Existing problem
   2.2 References
   2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   3.1 Empathy Map Canvas
   3.2 Ideation & Brainstorming
   3.3 Proposed Solution
   3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   4.1 Functional requirement
   4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   5.1 Data Flow Diagrams
   5.2 Solution & Technical Architecture
   5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   6.1 Sprint Planning & Estimation
   6.2 Sprint Delivery Schedule
   6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   7.1 Feature 1
   7.2 Feature 2
   7.3 Database Schema (if Applicable)
8. **TESTING**

INTRODUCTION

INTRODUCTION

1.1 PROJECT OVERVIEW

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

1.2. PURPOSE

Internet is basically system of interconnected computers through network. But now its use is changing with changing world and it is not just confined to emails or web browsing. Today's internet also deals with embedded sensors and has led to development of smart homes, smart rural area, e-health care's etc. and this introduced the concept of IoT . Internet of Things refers to interconnection or communication between two or more devices without human- 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

LITERATURE SURVEY

Literature Survey

Title & Author(s)	Year	Technique(s)	Findings/Pros/Cons
Smart Computing Applications in Railway Systems - A case study in Indian Railways Passenger Reservation System, Parag Chatterjee,Asoke Nath	2014	UID-based Reservation System	Using this UID-based technology, it is possible to decentralize the task of reservation from booking clerks (in railway reservation counters) to automatic ticket vending machines (ATVM) also. Since the unreserved tickets are already getting issued through ATVMs, the reservation feature can also be implemented in the ATVMs.
Railway Train Ticket Generation through ATM Machine: A Business Application for Indian Railways, Amit Kumar Gupta, Priyanka Ahlawat Mann	2011	ATM Module for Railway Ticket Generation	A model for the integration of ATM machine and railway ticket booking is proposed. In this model we proposed a change in architecture of ATM machines by adding an option for railway ticket module. This module will work on the railway reservation server and bank server. This model provides user to book railway tickets (general/reservations) through ATM card. This feature will also help user to check the waiting list of railway reservation
The Recent Reliable Advancements In The Indian Railway Ticketing System, Pardeep Kumar	2020	UID-based technology, RFID card	The passengers may travel using these cards and once the amount gets finished in the card they can again recharge it. He suggested this recharge in the card can be done on a monthly basis or on a quarterly basis. proposed the SMS ticketing system.
Finding trend of advanced ticket booking in Indian railways, Anuj Budhkar, Sanhita Das	2017	TransCAD, Cube Voyager	The mobile-based train ticketing system provides a better service to the passengers by enhancing the ticket issuing process. Dynamic QR codes, E-Wallet system, Ticket booking system, Report generating system, and Admin backend panel are the main processes of the mobile-based train ticketing system.
"TrainGo App" -Mobile based Train Ticketing System, Supun Nimesh, Sunesh Hettiarachchi,Samanthi Wickramsinghe	2020	Java and MySQL for App Development	A mobile-based train ticketing system is developed by carefully analyzing the collected requirements. Navigation must be simple, easy to set up and use, make quick the response,

			user training are the expected requirements from the solution.
Railway Online Booking System Design and Implementation, Wang Zongjiang	2012	DBMS ER Model	Seat distribution problems occur when a flexible seat reservation system is implemented in which passengers are allowed to reserve seats by submitting their demands instead of specifying trains.
Smart Ticketing and Seat Reservation System, Sachinthana Virajith, Isuri Gamage	2021	Global System Mobile Communications-Railway, 5G, and Wireless Sensor Networks.	According to that process, passengers need to visit the counters in railway stations, pay for the tickets, and get the tickets. The tickets currently issued by the Railways are valid only from the date of issue and to the given destination only.
A Combinatorial Auction Based Algorithm for Flexible Seat Reservation Systems, Kazutoshi Otomura, Norio Tomii	2005	Adaptive Data fusion	To solve the seat distribution problem, we have formalized it as a winner determination problem of the combinatorial auction mechanism. It should be noted that difficulty of the seat distribution problem varies depending on instances of the problem.
A multiobjective planning model for intercity train seat allocation, Yu-Hern Chang, Chung-Hsing Yeh	2004	Fuzzy Logic programming	The plan determines how many reserved and non-reserved seats are to be allocated at each origin station for all subsequent destination stations on each train run operated within a specified operating period.
Smart Computing Applications in Railway Systems - A case study in Indian Railways Passenger Reservation System, Parag Chatterjee,Asoke Nath	2014	UID-based Reservation System	Basically, this proposed smart model approach for passenger reservation system depends on some requirements, without which the benefits would not be fully enjoyed. This includes the UID registration of all passengers who needs to travel.
Smart ticketing system for railways in smart cities using software as a service architecture, Godson D'silva, Anoop Kunjumon Scariah, Lukose Roy Pannapara, Jessica John Joseph	2017	AWS IOT,AWS Dynamodb	
SMART RAIL RESERVATION AND VERIFICATION SYSTEM WITH UNIQUE IDENTIFICATION IN IoT USING CLOUD DATABASE, Adapa Sri Kumar Satya Ganapathi, S.Praveen Kumar, P.Madhusudhanan, S.Ranjith Kumar, M.Ganesan	2018	IOT using Cloud Database	In the current system there are many disadvantages which are to be rectified. The main thing which comes under is about allocation of lower berths.During Verification there are possibilities for fake identification also.

An optimization model to assign seats in long distance trains to minimize SARS-CoV-2 diffusion, Md Haque, Faiz Hamid	2022	Neural network	The unprecedented spread of SARS-CoV-2 has pushed governmental bodies to undertake stringent actions like travel regulations, localized curfews, curb activity participation, etc. These restrictions assisted in controlling the proliferation of the virus; however, they severely affected major economies.
IoT Based Ticket Checking System, Kirti Dhiman , Er. CK Raina	2017	IOT based System	In this paper we represent main layered architecture of IOT. We make a system that checks the ticket of passenger in trains through IOT based System. After entered in train ticket is checked by a System. We studied the entire layer that is used in IOT. This system describes the whole architecture of IOT.
A new railway line planning model considering multinomial LOGIT-based traffic assignment, B.H. Park, C.-S. Kim, T. Lim, H.-L. Rho	2011	Fuzzy Extension Constraint Algoritm	A system split is a procedure to distribute passengers' demand over an entire network. This distribution is based on a value assigned to a route which is determined primarily from the total travel time, the number of transfers and the maximum headway (or minimum frequency) of a route.

IDEATION AND PROPOSED SOLUTION

BRAIN STORMING:

Smart Computing Applications in Railway Systems - A case study in Indian Railways Passenger Reservation System

The demand for safe, fast, and reliable rail services continues to be the reason for concern in all the countries across the globe. Lack of operational efficiency and reliability, safety and security issues, and aging railway systems and practices are haunting various countries to bring about a change in their existing rail infrastructure. The global rail industry struggles to meet the increasing demand for freight and passenger transportation due to lack of optimized use of rail network and inefficient use of rail assets. This is expected to induce rail executives to build rail systems that are smarter and more efficient. The passenger reservation system of Indian Railways is one of the world's largest reservation models. Daily about one million passengers travel in reserved accommodation with Indian Railways. Another sixteen million travel with unreserved tickets in Indian Railways. In this vast system, it is a herculean task to efficiently handle the passenger data, which is a key point of consideration now-a-days. In this paper, the authors have explored different issues of implementing smart computing in railway systems pertaining to reservation models.

IoT Based Ticket Checking System

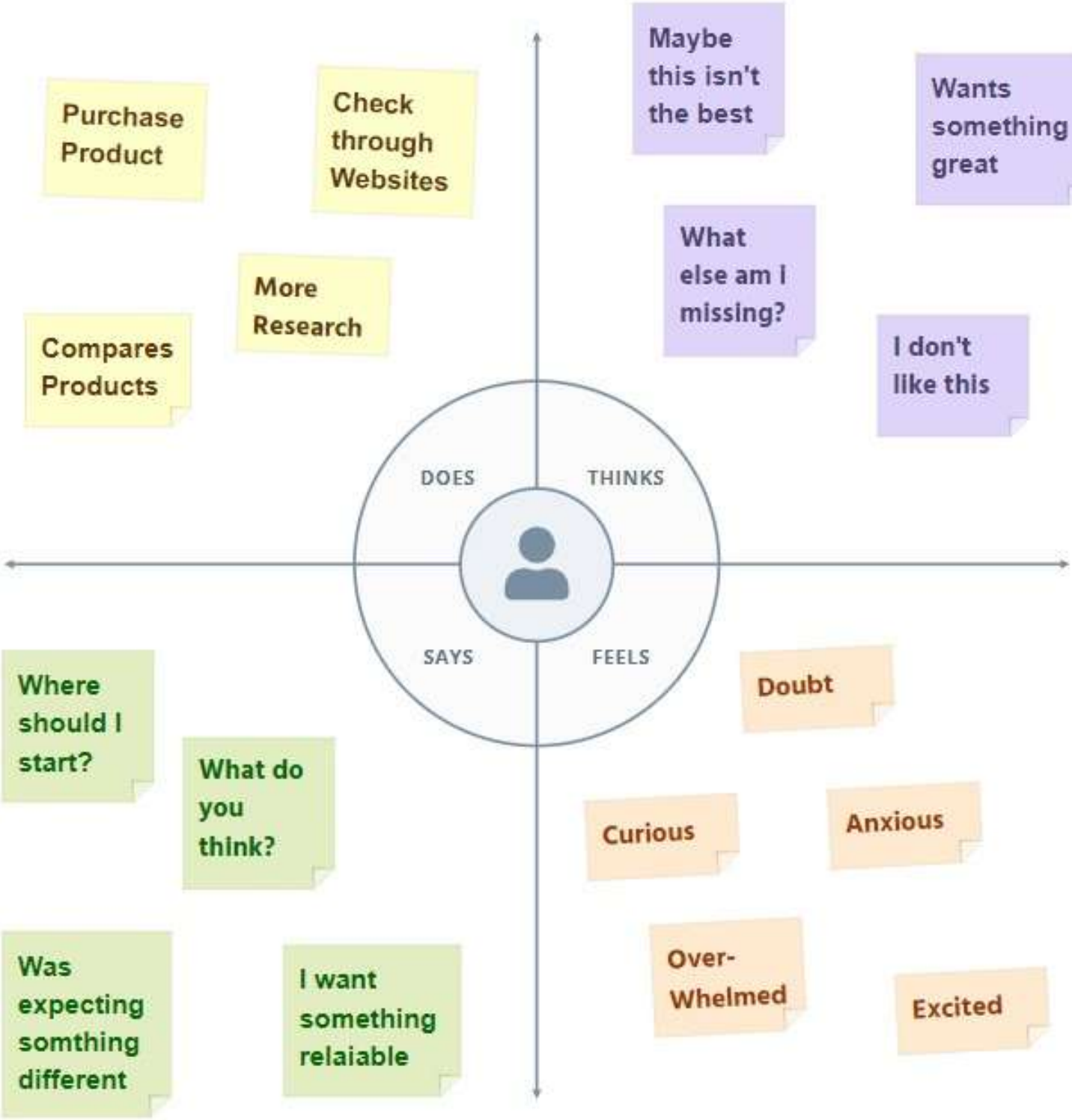
Internet of things. The term Internet of Things was used by Kevin Ashton in 1999. IOT is like a vehicle used to as a “Smart Devices” and other items like Electronics, softwares.IOT words was Invented from a two words “Internet ”and “Things”. Internet is a vast network. The Internet is the global system of interconnected

computer networks that use the protocol to link devices. Internet is used in daily life to communicate, search information etc. things means important information or devices. In recent days a population is gained day by a day and smart cities have gained popularity. In this paper we present a “IOT BASED TICKET CHECKING SYSTEM”. This system is consist of an IOT module that is used to check the tickets of passenger in trains. This system describes the whole architecture of a train system.

SMART RAIL RESERVATION AND VERIFICATION SYSTEM WITH UNIQUE IDENTIFICATION IN IoT USING CLOUD DATABASE

The Internet of Things is inter-networking of physical devices, buildings, and other items which are embedded with electronics, software, sensors, actuator, and network connectivity that enable these objects to collect and exchange data. The devices which are connecting to internet are called IoT Devices. In technical we can say it as the device which has IP Address is called as an IoT Device. Analysts say that by 2020 there will be over 50 billion devices. That's a lot of connections. More over some estimate that it would be 100 billion devices. In general ticket reservation for the Indian railways is quite a complex process. This involves various steps which could be much complicated for illiterates. More over Indian railways is using more than 2 tonnes of paper for booking and verification process. To avoid these problems and more over to move the nation towards digitalization we are proposing this idea. Here in this proposal we are building web-based application for reservation and mobile application for ticket verification process.

EMPATHY MAP- Smart Solutions for Railways



Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	Lack of services provided by which customer can book tickets and ticket collector can verify tickets as paperless entries.
2.	Idea / Solution description	Our solution is to build a one-stop web application that has the following features: • Book train tickets through online • QR code based verification of tickets • Track live location of the train • Alerting the passengers before trainreaching the destination location
3.	Novelty / Uniqueness	Our solution is unique because of thefollowing reasons: ● Verification of tickets is done usingdynamic QR code ● One-stop solution for all of the services mentioned in solution description
4.	Social Impact / CustomerSatisfaction	Customers would definitely be satisfiedwith our solution as it: ● It is hassle free and paperless trainticket booking ● The solution enables fasterverification of tickets ● The solution gives precise or approximate location of trains based oncustomer's requirement
5.	Business Model (Revenue Model)	**1.Key Partners:** Users who use our app for booking tickets.Indian Railways. **2.Key Activities:** Our app is for booking train tickets fasterand easier with QR Code. Our app provides live tracking of trains. Our app alerts passengers before destinationstation.

		3.**Customer Relationships:** Customers can book tickets using onlinepayment like Gpay/Paytm. Customers can verify their ticket with thehelp of QR Code. 4.**Cost Structure:** Actual ticket cost will be displayed on the user account for their journey.
6.	Scalability of the Solution	Proposed solution would have the above mentioned features in the initial version. As we gain many users, the servers will be scaled in the cloud service to accommodatemore users. Some of them could be: ● Improved process of generation andverification of QR ● On demand toilet cleaning services insidetrains ● Automatic ticket checking at doors etc., Our solution is viable as we can use a GPS module in all trains and connect them to a cloud service so that live location tracking isvery easy.

PROBLEM SOLUTION FIT

Define CS, fit into CC

1. CUSTOMER CS

People who travel via train from one place to another.

6. CUSTOMER C

Customers who book tickets from a web portal, will receive booking information and the traveler has totake a copy of it and it has to be shown to the ticket checker for confirmation.

5. AVAILABLE AS

But this can be replaced by electronic way as aQR code where booking information can be stored. The code is viewed in the phone and the code will be scanned by the ticket checkerwith a device code can read the information and details can be verified.

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS J&P

The scanner and devices that installed may have a chance of getting damaged because ofnatural calamities. They is a chance of getting disconnected due to various reasons like network issue, scanner malfunction, etc.

9. PROBLEM ROOT CAUSE RC

In day to day people who uses sub urban trains increased a lot and to for tickets and chart preparation we are using excessive amount of paper and dyes for printing. For the ticket checker manually verify 1000's of passenger ticket is a difficult task and the traveler also should bring a copy to verify it.
For that QR code based electronic scanning device can be given to the checker and once scanned the traveler code he can approve the data and confirm that passenger has boarded the train.

7. BEHAVIOUR BE

Customer has to book the tickets and keep the QRcode generated via booking confirmation till their journey.

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

3. TRIGGERS TR

The customers are able to view very conveniently through their phone and easyto use which triggers even the illiterates totravel by booking in the online portal.

4. EMOTIONS: BEFORE / AFTER EM

Customers will feel some relief even thoughif they receive the mail and details in apps like train details and QR code which can be downloaded to offline where network issues does not arises and stop spending money for printouts and keeping it safe till journey.

10. YOUR SOLUTION SL

Through this project we can get benefited in both the sides: one is passenger and another one is the railwaydepartment as well as ticket checker.

The hard copy of the tickets can be converted in to soft copy with the help of QR code where the booking is storedand sent to the passenger's mobile number and mail id atthe time of booking. The generated code can be viewed only in a smart phone for that issue if a person does not have smart phone he/she can tell the ticket checker a 7 digit /blocked letters which will fetch the data of the passenger.

The ticket checker will be given a small scanning device with a display to view the data and confirm the presencein the train.

For Network issues: (Solution)

For the customer side the code will be sent to his WhatsApp/ mailand they can download it. When it comes to checker side the scanning device will have memory to store the final passenger list where he can verify by scanned data and with the data chart given to the machine and it will compare the data and the checker will click confirm to mark the presence.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

If the Passenger has internet no worries they view the data in the mail as well as log in portal.To view the QR code.

8.2 OFFLINE

If the passenger has no internet or other issues, the can show or tell the 7 digit /blocked letters to the ticket checker and confirm it.

Identify Strong TR & EM

REQUIREMENT ANALYSIS

Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through website Manual Registration
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Payment Options	Net Banking/ UPI/Credit card/Debit card Digital Wallet
FR-4	User Requirements	Smart Phone / Laptop Internet QR code Scanner
FR-5	User Feedback	Feedback via App/website Contact the authority via mail Direct Complaint
FR-6	Installation	Free installation via preferred app store. Directly use via website

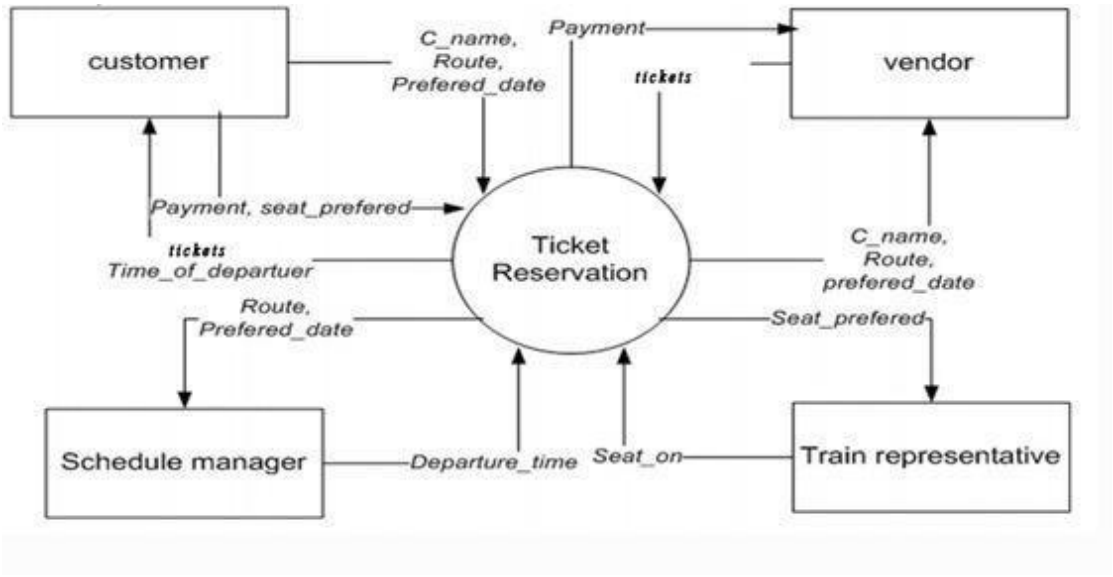
Non-functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Easy to use the application and understand. Even illiterate people can use it easily.
NFR-2	**Security**	Multi-Factor authentication Strong password policy Having a strong encryption
NFR-3	**Reliability**	Periodic updates to fix any bugs in the application. Internet is required only the time of booking and the booking details and the QR code will be made offline the mobile phone by sending the details via mail and SMS and WhatsApp. Offline mode for important features for better reliability to use in place with no internet connectivity.
NFR-4	**Performance**	The user interface should user- friendly and the application can be hosted via light speed server to prevent the loading time for booking and payment for the user.

NFR-5	**Availability**	When the user enters the application, according to the constraints given by the user to move to other destination the availability of trains and database for all other trains has to be up to date with an availability of seats. The database has to be dynamically updated whenever the user reserve a ticket and the availability of seats to be reduced to total seats in coach. Website or application has to 24/7 available for booking with customer care support.
NFR-6	**Scalability**	The database should be able to handle a large volume of data especially during peak times. It should scale automatically to be cost effectively. It should be able to store the data in the server as well as the mirror server at the time of requirement.

PROJECT DESIGN

DATA FLOW DIAGRAMS:

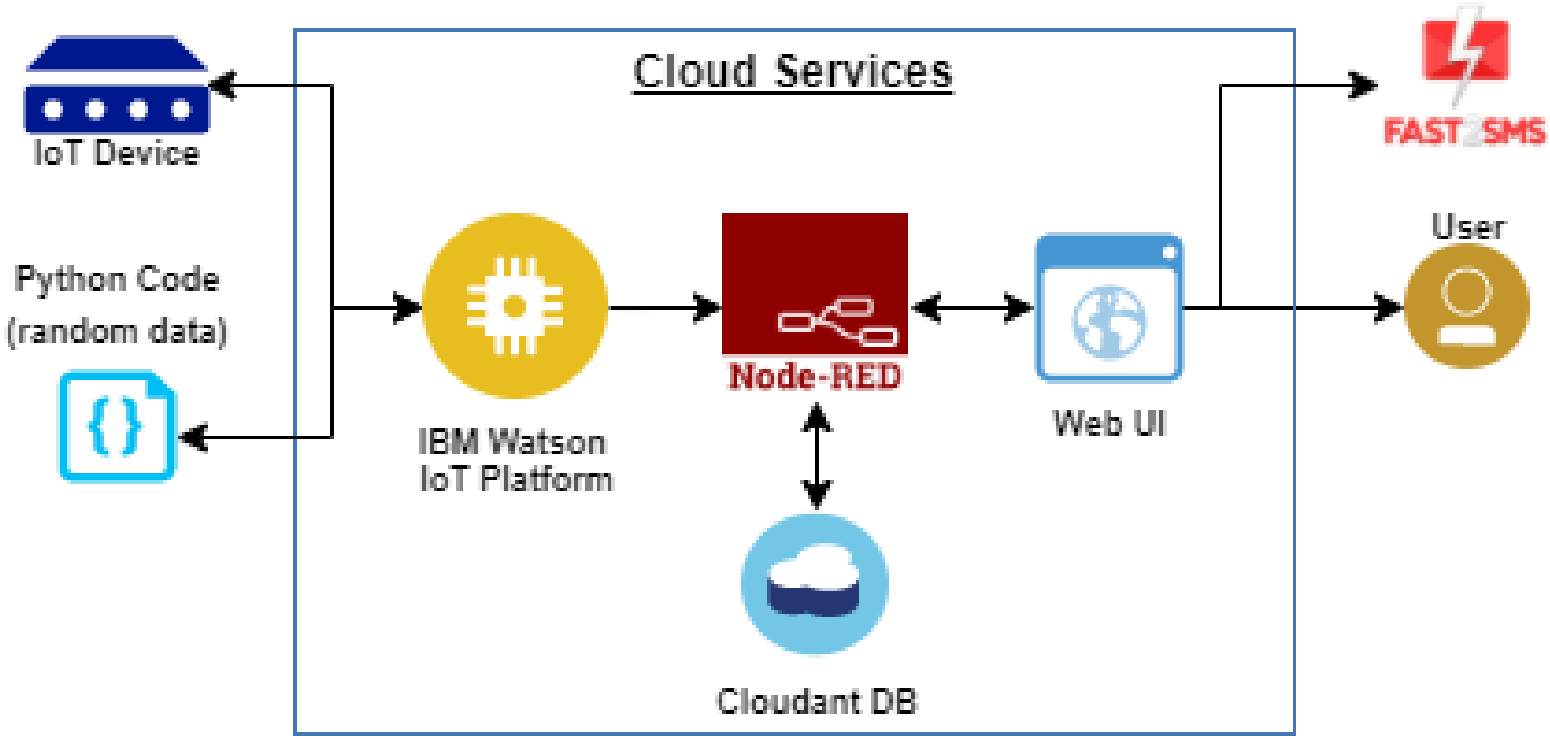


User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Passenger	Online Registration	USN-1	As a passenger, I can register for the application by entering my email, password, and confirming my password and to create alogin credentials so I can securely access myself service online account.	I can access my account /Input data fields to enter: 1.Username/email 2.Password	High	Sprint-1
Passenger	Ticket Conformation	USN-2	As a passenger, I will receive confirmation email once I have registered for the application to check my ticket whether it is conformed or not.	I can receive confirmation email & click confirm	Medium	Sprint-1
Passenger	Payment	USN-3	As a passenger, I can register for the application through Facebook and want to paymy ticket cost in online payment.	I can register & access the dashboard with Facebook Login	High	Sprint-2
Passenger	Booking status	USN-4	As a passenger , I can register for the application through Gmail	E-Ticket	High	Sprint-3
Administrator	Login and update information.	USN-5	As a admin , I can log into the application by entering email & password ,and to check thetrains details	Train Details	High	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Ticket Collector	Dashboard	USN-6	As a Ticket Collector ,I want to check the users whether he/she have tickets or not with scanning theQR Code	QR CODE	High	Sprint-4
Passenger	Knowing Current Location details	USN-7	As a Passenger,I want to know the train current location.	GPS of train	Medium	Sprint-4

Technical Architecture:



Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with website	HTML, CSS, JavaScript
2.	Live location tracking	Tracking live location of trains and updating it in the website.	IBM Watson IOT , IBM cloud
3.	Ticket booking and verification	Ticket is generated as a QR code which can be shown to the Ticket Examiner who scans the code using Scanner and verifies the ticket.	Django, qrcode.js
4.	Food ordering	Passengers can order food in the website and itcan delivered to them .	GrubHub API
5.	Database	Data Type, Configurations etc.	Django
6.	Cloud Database	Database Service on Cloud for storing tickets anduser identification.	IBM DB2, IBM Cloudant
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	Trash segregation	Provide bounties for trash segregation	IBM Watson IoT, node.js
9.	Destination Notification	Alerts passengers just before the destination station	Geofencing API

Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Open-Source Frameworks are used in our application for UI creation, QR code generation, trash segregation	Django, qrcode.js, JavaScript,node.js
2.	Security Implementations	Securing user credentials and ticket information in database	SHA-256, OWASP
3.	Scalable Architecture	To scale the cloud database for storing more tickets as users increase	IBM Auto Scale
4.	Availability	To make the application available to use 24/7	API
5.	Performance	To increase the performance of the application hosted in the high-performance instance	IBM Instance

PROJECT PLANNING AND SCHEDULING

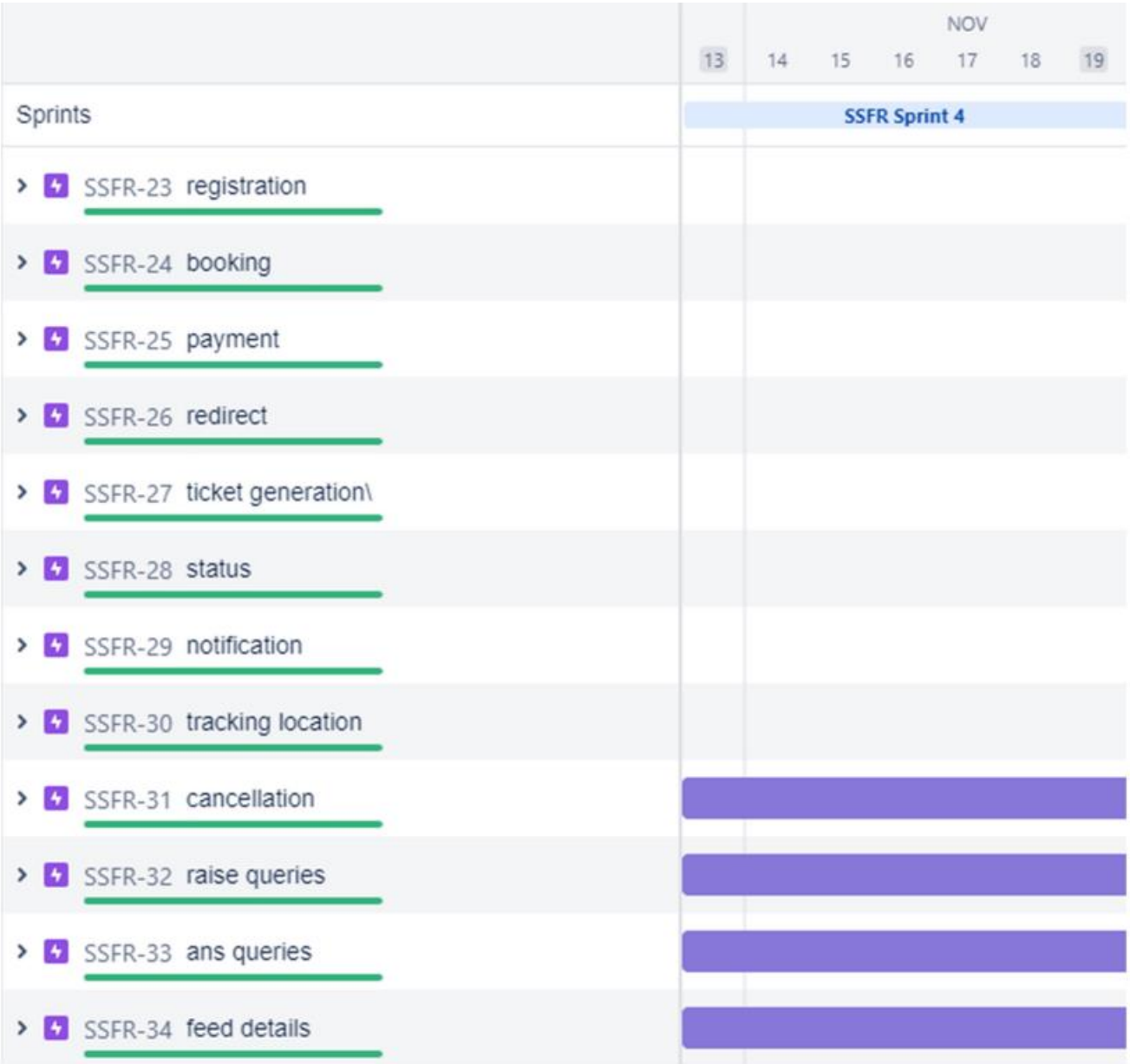
SPRINT PLANNING& ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a passenger, I want to create a login credentials so I can securely book a train ticket online.	15	High	Shiyam R, Sooraj I, Srividya R, Vignesh B
Sprint-1	Ticket Confirmation	USN-2	As a passenger, I want to check my ticket whether it is confirmed or not. And get a ticket conformation message.	5	Medium	Shiyam R, Sooraj I, Srividya R, Vignesh B
Sprint-2	Payment	USN-3	As a passenger, I want to pay my ticket cost in online payment modes like GooglePay,Paytm,etc.	15	High	Shiyam R, Sooraj I, Srividya R, Vignesh B
Sprint-2	Knowing Current Location details	USN-7	As a passenger, I want to track the live location of the train	5	Low	Shiyam R, Sooraj I, Srividya R, Vignesh B
Sprint-3	Booking Status	USN-4	As a passenger, I want to check my ticket once it is confirmed	5	Medium	Shiyam R, Sooraj I, Srividya R, Vignesh B
Sprint-3	Verifying Tickets	USN-6	As a Ticket Collector, I want to check the passengers whether he/she have tickets by scanning the QR Code.	15	High	Shiyam R, Sooraj I, Srividya R, Vignesh B
Sprint-4	Updating Train Information	USN-5	As an admin, I want to check the train's details like when will the train reach stations and update Train information.	10	Medium	Shiyam R, Sooraj I, Srividya R, Vignesh B
Sprint-4	Raise a compliant	USN-8	As a user, I should be able to raise a compliant if something is wrong or give review for the journey travelled in the train.	10	Medium	Shiyam R, Sooraj I, Srividya R, Vignesh B

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	02 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

REPORTS FROM JIRA:



CODING & SOLUTIONING

PROGRAM FOR LED BLINKING:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
cnt = 0
MAIL_CHECK_FREQ = 1
RED_LED = 4
GPIO.setup(RED_LED, GPIO.OUT)
while True:
if cnt == 0 :
GPIO.output(RED_LED, False)
cnt = 1
else:
GPIO.output(RED_LED, True)
cnt = 0
time.sleep(MAIL_CHECK_FREQ)
GPIO.cleanup()
```

PROGRAM FOR TRAFFIC LIGHTS FOR RASPBERRY PI:

```
import RPi.GPIO as GPIO
import time
try:
def lightTraffic(led1, led2, led3, delay ):
GPIO.output(led1, 1)
time.sleep(delay)
GPIO.output(led1, 0)
GPIO.output(led2, 1)
time.sleep(delay)
GPIO.output(led2, 0)
GPIO.output(led3, 1)
time.sleep(delay)
```

```
GPIO.output(led3, 0)
GPIO.setmode(GPIO.BCM)
button = 19
GPIO.setup(button, GPIO.IN,
pull_up_down=GPIO.PUD_UP)
ledGreen = 16
ledYellow = 12
ledRed = 23
GPIO.setup(ledGreen, GPIO.OUT)
GPIO.setup(ledYellow, GPIO.OUT)
GPIO.setup(ledRed, GPIO.OUT)
while True:

input_state = GPIO.input(button)
if input_state == False:
print('Button Pressed')
lightTraffic(ledGreen, ledYellow, ledRed, 1)
else:
GPIO.output(ledGreen, 0)
GPIO.output(ledYellow, 0)
GPIO.output(ledRed, 0)
except KeyboardInterrupt:
print ("You've exited the program")
finally:
GPIO.cleanup()
```

PROGRAM FORWEBSITE LOGIN:

```
from tkinter import *
import sqlite3
root = Tk()
root.title("RAILWAYS LOGIN")
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)
#==============================VARIABLES==========
============================
USERNAME = StringVar()
PASSWORD = StringVar()
#==============================FRAMES============
============================
Top = Frame(root, bd=2, relief=RIDGE)
Top.pack(side=TOP, fill=X)
Form = Frame(root, height=200)
Form.pack(side=TOP, pady=20)
#==============================LABELS=============
============================
lbl_title = Label(Top, text = "Python: Simple Login
Application", font=('arial', 15))
lbl_title.pack(fill=X)
lbl_username = Label(Form, text = "Username:", font=('arial',
14), bd=15)
lbl_username.grid(row=0, sticky="e")

lbl_password = Label(Form, text = "Password:", font=('arial',
14), bd=15)
lbl_password.grid(row=1, sticky="e")
lbl_text = Label(Form)
lbl_text.grid(row=2, columnspan=2)
```

```
#==============================ENTRY
WIDGETS==================================
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)
#==============================METHODS==========
==============================
def Database():
global conn, cursor
conn = sqlite3.connect("pythontut.db")
cursor = conn.cursor()
cursor.execute("CREATE TABLE IF NOT EXISTS `member`
(mem_id INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, username TEXT, password TEXT)")
cursor.execute("SELECT * FROM `member` WHERE
`username` = 'admin' AND `password` = 'admin'")
if cursor.fetchone() is None:
cursor.execute("INSERT INTO `member` (username,
password) VALUES('admin', 'admin')")
conn.commit()
def Login(event=None):
Database()
if USERNAME.get() == "" or PASSWORD.get() == "":
lbl_text.config(text="Please complete the required field!",
fg="red")

else:
cursor.execute("SELECT * FROM `member` WHERE
`username` = ? AND `password` = ?",
(USERNAME.get(), PASSWORD.get()))
```

```
if cursor.fetchone() is not None:
HomeWindow()
USERNAME.set("")
PASSWORD.set("")
lbl_text.config(text="")
else:
lbl_text.config(text="Invalid username or password",
fg="red")
USERNAME.set("")
PASSWORD.set("")
cursor.close()
conn.close()
#==============================BUTTON
WIDGETS==================================
btn_login = Button(Form, text="Login", width=45,
command=Login)
btn_login.grid(pady=25, row=3, columnspan=2)
btn_login.bind('<Return>', Login)
def HomeWindow():
global Home
root.withdraw()
Home = Toplevel()
Home.title("Python: Simple Login Application")
width = 600
height = 500

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.resizable(0, 0)
Home.geometry("%dx%d+%d+%d" % (width, height, x, y))
```

```
lbl_home = Label(Home, text="Successfully Login!",
font=('times new roman', 20)).pack()
btn_back = Button(Home, text='Back',
command=Back).pack(pady=20, fill=X)
def Back():
Home.destroy()
root.deiconify()
```

PROGRAM FOR OTP GENERATION:

```
# import library
import math, random
# function to generate OTP
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())
```

PROGRAM FOR OTP VERIFICATION:

```
import os
import math
import random
```

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

PROGRAM FOR REGISTRATION:

```
from tkinter import*
base = Tk()
base.geometry("500x500")
base.title("registration form")
labl_0 = Label(base, text="Registration
form",width=20,font=("bold", 20))
labl_0.place(x=90,y=53)
lb1= Label(base, text="Enter Name", width=10,
font=("arial",12))
lb1.place(x=20, y=120)
en1= Entry(base)
```

```
en1.place(x=200, y=120)
lb3= Label(base, text="Enter Email", width=10,
font=("arial",12))
lb3.place(x=19, y=160)
en3= Entry(base)
en3.place(x=200, y=160)
lb4= Label(base, text="Contact Number",
width=13,font=("arial",12))
lb4.place(x=19, y=200)
en4= Entry(base)
en4.place(x=200, y=200)
lb5= Label(base, text="Select Gender", width=15,
font=("arial",12))
lb5.place(x=5, y=240)
var = IntVar()
Radiobutton(base, text="Male", padx=5,variable=var,
value=1).place(x=180, y=240)
Radiobutton(base, text="Female", padx =10,variable=var,
value=2).place(x=240,y=240)
Radiobutton(base, text="others", padx=15, variable=var,
value=3).place(x=310,y=240)
```

PROGRAM FOR TICKET BOOKING:

```
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
```

PROGRAM FOR PAYMENT:

```
from django.contrib.auth.base_user import
AbstractBaseUser
from django.db import models
```

```
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="Bialystok",
)
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="Mysliwska 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="Bialystok",
)
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="Mysliwska 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")
```

PROGRAM FOR SEAT BOOKING:

```
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
```

PROGRAM FOR TICKET CONFROMATION:

```
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(se
lf.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
```

PROGRAM FOR GPS TRACKING:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from PIL import Image, ImageDraw
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()
```

PROGRAM FOR NOTIFICATION:

```
import pyttsx3
from plyer import notification
import time
# 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()
```

PROGRAM FOR TICKET GENERATION:

```
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(se
lf.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
```

PROGRAM FOR FEEDBACK:

```
# Python program to find PNR
# status using RAILWAY API
# import required modules
import requests, json
# Enter API key here
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")
```

PROGRAM FOR QUERIES:

```
import email, smtplib, ssl
from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
```

```
from email.mime.text import MIMEText
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)
```

PROGRAM FOR RAISE QUERIES:

```
import smtplib, ssl
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
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()
)
```

PROGRAM FOR TICKET CANCELLATION:

```
from pickle import load,dump
import time
import random
import os
class tickets:
def __init__(self):
self.no_ofac1stclass=0
self.totaf=0
```

```
self.no_ofac2ndclass=0
self.no_ofac3rdclass=0
self.no_ofsleeper=0
self.no_oftickets=0
self.name=''
self.age=''
self.resno=0
self.status=''
def ret(self):
return(self.resno)
def retname(self):
return(self.name)
def display(self):
f=0
fin1=open("tickets.dat","rb")
if not fin1:
print "ERROR"
else:
print
n=int(raw_input("ENTER PNR NUMBER : "))
print "\n\n"
print ("FETCHING DATA . . .".center(80))
time.sleep(1)
print
print('PLEASE WAIT...!!'.center(80))
time.sleep(1)
os.system('cls')
try:
while True:
tick=load(fin1)
if(n==tick.ret()):
```

```
f=1
print "="*80
print("PNR STATUS".center(80))
print"="*80
print
print "PASSENGER'S NAME :",tick.name
print
print "PASSENGER'S AGE :",tick.age
print
print "PNR NO :",tick.resno
print
print "STATUS :",tick.status
print
print "NO OF SEATS BOOKED : ",tick.no_oftickets
print
except:
pass
fin1.close()
if(f==0):

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

```
def confirmation (self):
self.status="CONFIRMED"
print "PNR NUMBER : ",self.resno
print
time.sleep(1.5)
print "STATUS = ",self.status
print
def cancellation(self):
z=0
f=0
fin=open("tickets.dat","rb")
fout=open("temp.dat","ab")
print
r= int(raw_input("ENTER PNR NUMBER : "))
try:
while(True):
tick=load(fin)

z=tick.ret()
if(z!=r):
dump(tick,fout)
elif(z==r):
f=1
except:
pass
fin.close()
fout.close()
os.remove("tickets.dat")
os.rename("temp.dat","tickets.dat")
if (f==0):
print
print "NO SUCH RESERVATION NUMBER FOUND"
```

```
print
time.sleep(2)
os.system('cls')
else:
print
print "TICKET CANCELLED"
print"RS.600 REFUNDED...."
def reservation(self):
trainno=int(raw_input("ENTER THE TRAIN NO:"))
z=0
f=0
fin2=open("tr1details.dat")
fin2.seek(0)
if not fin2:
print "ERROR"
else:
try:
while True:
tr=load(fin2)
z=tr.gettrainno()
n=tr.gettrainname()
if (trainno==z):
print
print "TRAIN NAME IS : ",n
f=1
print
print "-"*80
no_ofac1st=tr.getno_ofac1stclass()
no_ofac2nd=tr.getno_ofac2ndclass()
no_ofac3rd=tr.getno_ofac3rdclass()
no_ofsleeper=tr.getno_ofsleeper()
```

```
if(f==1):
fout1=open("tickets.dat","ab")
print
self.name=raw_input("ENTER THE PASSENGER'S NAME ")
print
self.age=int(raw_input("PASSENGER'S AGE : "))
print
print"\t\t SELECT A CLASS YOU WOULD LIKE TO TRAVEL IN :- "
print "1.AC FIRST CLASS"
print
print "2.AC SECOND CLASS"
print
print "3.AC THIRD CLASS"

print
print "4.SLEEPER CLASS"
print
c=int(raw_input("\t\t\tENTER YOUR CHOICE = "))
os.system('cls')
amt1=0
if(c==1):
self.no_oftickets=int(raw_input("ENTER NO_OF FIRST CLASS
AC SEATS TO BE BOOKED :
"))
i=1
while(i<=self.no_oftickets):
self.totaf=self.totaf+1
amt1=1000*self.no_oftickets
i=i+1
print
print "PROCESSING. .",
time.sleep(0.5)
```

```
print ".",
time.sleep(0.3)
print'.'
time.sleep(2)
os.system('cls')
print "TOTAL AMOUNT TO BE PAID = ",amt1
self.resno=int(random.randint(1000,2546))
x=no_ofac1st-self.totaf
print
if(x>0):
self.confirmation()
dump(self,fout1)

break
else:
self.pending()
dump(tick,fout1)
break
elif(c==2):
self.no_oftickets=int(raw_input("ENTER NO_OF SECOND
CLASS AC SEATS TO BE BOOKED
: "))
i=1
def menu():
tr=train()
tick=tickets()
print
print "WELCOME TO PRAHIT AGENCY".center(80)
while True:
print
print "="*80
print " \t\t\t\t RAILWAY"
```

```
print
print "="*80
print
print "\t\t\t1. **UPDATE TRAIN DETAILS."
print
print "\t\t\t2. TRAIN DETAILS. "
print
print "\t\t\t3. RESERVATION OF TICKETS."
print
print "\t\t\t4. CANCELLATION OF TICKETS. "
print
print "\t\t\t5. DISPLAY PNR STATUS."
print
print "\t\t\t6. QUIT."
print"** - office use....."
ch=int(raw_input("\t\t\tENTER YOUR CHOICE : "))
os.system('cls')
print
"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n
\t\t\t\t\t\t\tLOADING. .",
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\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\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\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
```

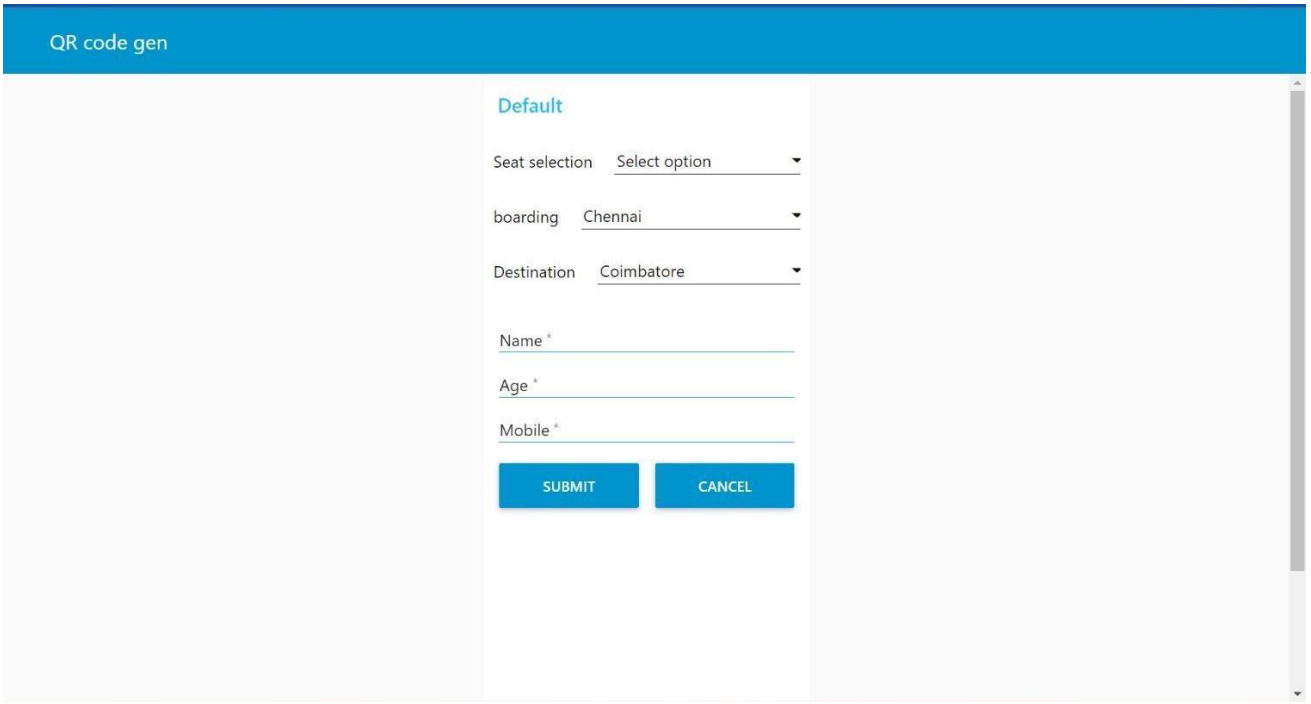
```
print"\t\t\t\tTRAIN DETAILS"
print"*"*80
print
tr=load(fin)
tr.output()
raw_input("PRESS ENTER TO VIEW NEXT TRAIN DETAILS")

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
print
tick.display()
elif ch==6:
quit()
raw_input("PRESS ENTER TO GO TO BACK MENU".center(80))
```

```
os.system('cls')
menu()
```

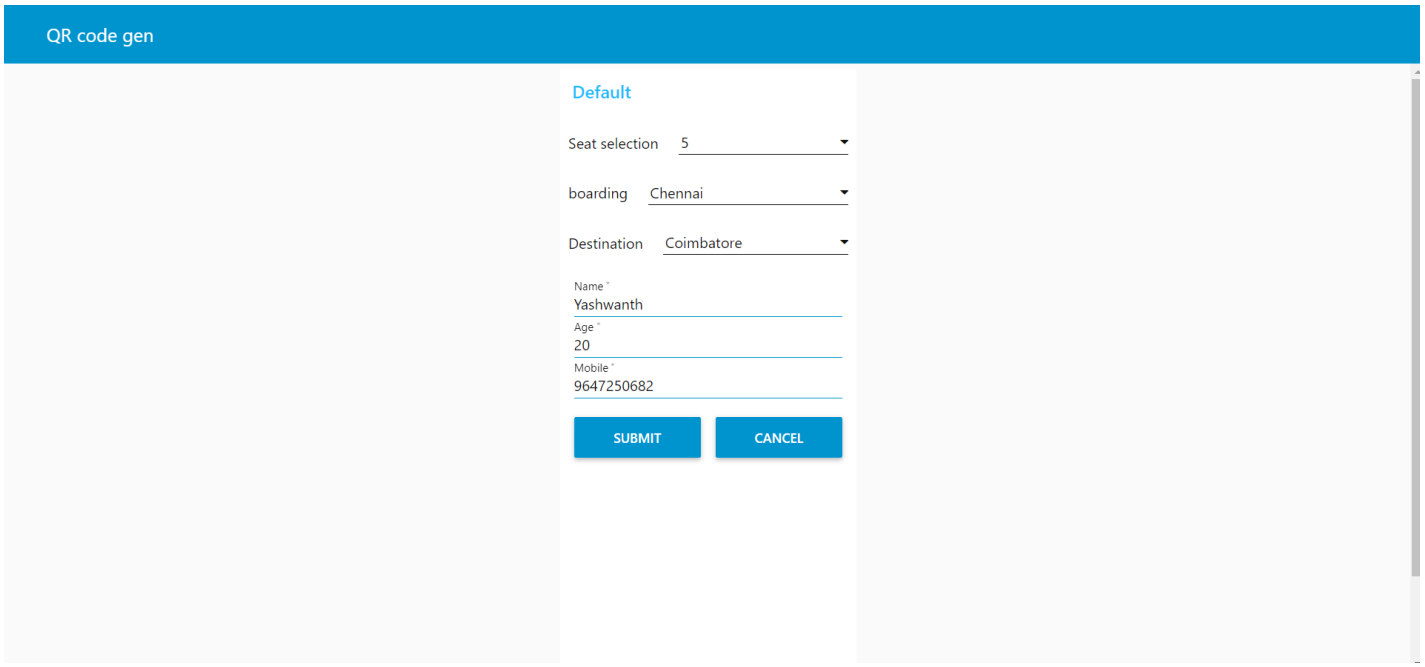
TESTING

TESTING THE WEB UI USING NODE RED

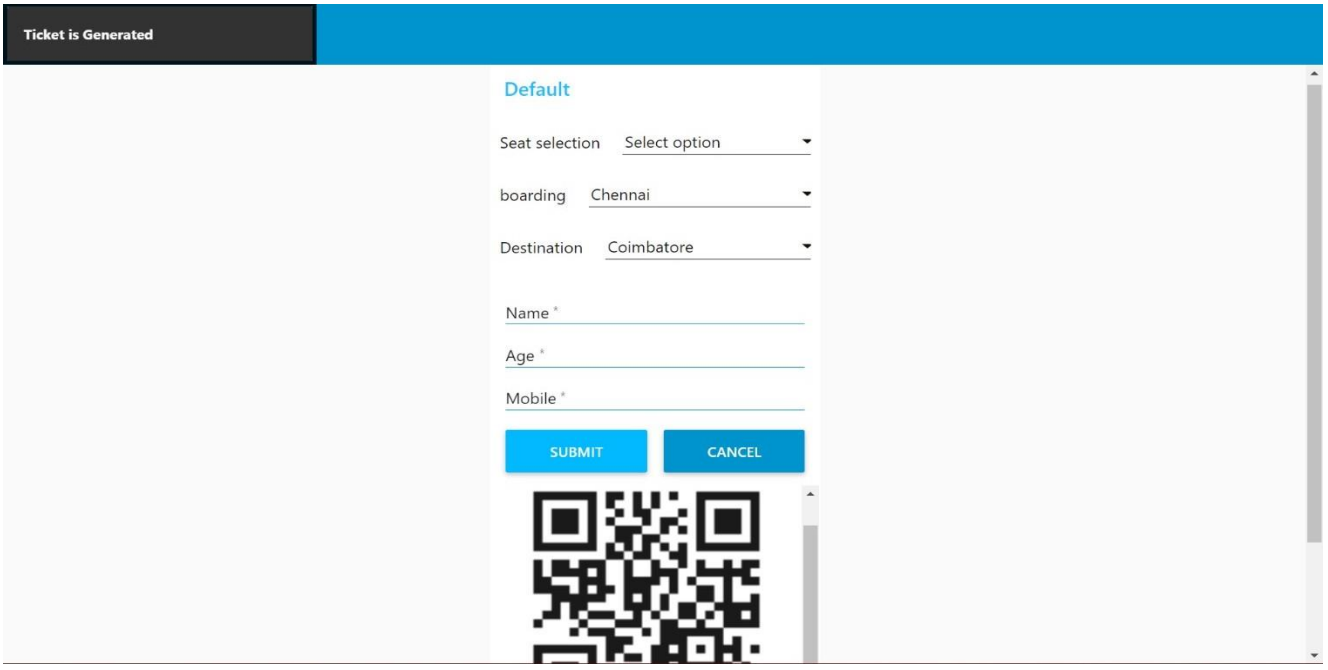
WEB UI:



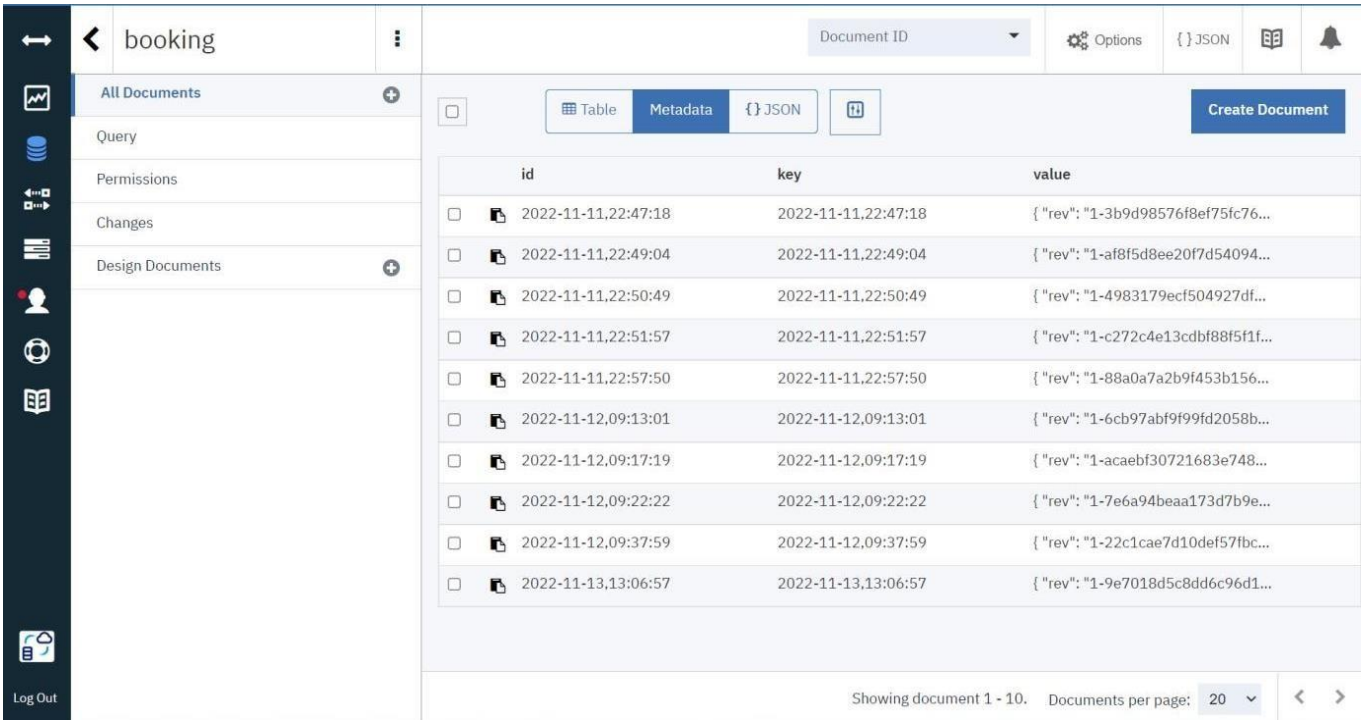
FILL THE REQUIRED DETAILS:



GENERATION OF QRCODE:



DATA STORED IN CLOUDANT:



RESULTS

PERFORMANCE METRICS:

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- **The *Right* Information, Every Time:**
  Many rail operations around the country still manage inventory by writing down railcar information as they pass into and out of the plant or facility, inviting the potential for a “4” to be entered as a “7” in the system, and causing losses of time, resources, and sometimes money
- **Safety, Security, and Savings:**
  Custom alerts tell operators when a car is due for inspection, when it shouldn’t be loaded, and more before they are able to complete its next activity, and volume correction calculators can enable you to maximize your railcar’s image and potentially save thousands long-term.
- **Results for your Operation:**
  Heightening your operation’s tracking-and-tracing and automating your logistics activities can be a proven benefit to the safety and efficiency of any rail-based supply chain strategy.

DISADVANTAGES:

- **Miss a Payment:**

  Unless you have your own billing software, paperless statements actually make it easier for some people to forget to pay on time. This is especially true if you typically use the paper statement as your payment reminder. The alternative to this is just printing off the paperless statement you receive in your email and then use that as your payment reminder.

- **More Passwords are required:**

  This may not seem to be a huge issue for many but think about this for just a moment. If you sign up for paperless statements you now end up with another user name and password to use at another website.This is why billing software for business has to be user friendly from the very first step.

- **Restricted Access to Old Statements:**

  They may date back to long before billing software was commonplace but it was our personal system. The main reason why you would want to be able to produce physical copies of old statements would be for income tax purposes.

CONCLUSION

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

FUTURE SCOPE:

Growing populations and rising congestion in urban centers have made traditional railway infrastructure, which takes up a lot of space, difficult to implement. In densely populated urban locations, constructing new metro rail lines costs too much in terms of land acquisition, inspection and leveling, and eventual construction. These projects also take several years to complete, leaving urban cities in a state of congestion for a prolonged period.

Indian startup designs and develops novel urban transit solutions. Apart from providing existing mainline railway, rapid transit, and urban transportation with services like surveys, inspections, and design, they also develop radical solutions.

GITHUB LINK:

- **https://github.com/IBM-EPBL/IBM-Project-41750-1660644539**