Project Report

1. INTRODUCTION

1.1 PROJECT OVERVIEW

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

1.2 PURPOSE

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

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

In the Existing train tracks are manually researched. LED (Light Emitting Diode) and LDR (Light Dependent Resister) sensors cannot be implemented on the block of the tracks]. The input image processing is a clamorous system with high cost and does not give the exact result. The Automated Visual Test Method is a complicated method as the video color inspection is implemented to examine the cracks in rail track which does not give accurate result in bad weather. This traditional system delays transfer of information. Srivastava et al., (2017) proposed a moving gadget to detect the cracks with the help of an array of IR sensors to identify the actual position of the cracks as well as notify to nearest railway station

. Mishra et al., (2019) developed a system to track the cracks with the help of Arduino mega power using solar energy and laser. A GSM along with a GPS module was implemented to get the actual location of the faulty tracks to inform the authorities using SMS via a link to find actual location on Google Maps. Rizvi Aliza Raza presented a prototype in that is capable of capturing photos of the track and compare it with the old database and sends a message to the authorities regarding the crack detected. The detailed analysis of traditional railway track fault detection techniques is explained in table

2.2 REFERENCES

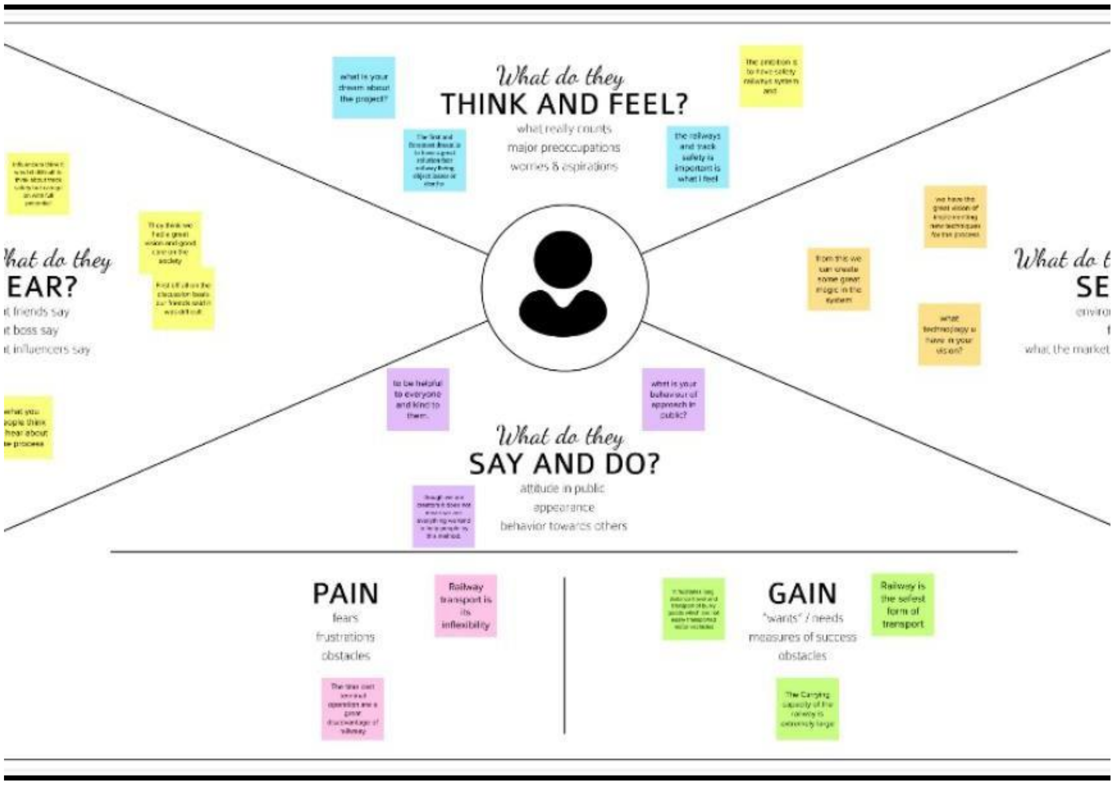
1. D. Hesse, “Rail Inspection Using Ultrasonic Surface Waves” Thesis, Imperial College of London, 2007.
2. Md. Reya Shad Azim1 , Khizir Mahmud2 and C. K. Das. Automatic railway track switching system, International Journal of Advanced Technology, Volume 54, 2014.
3. S. Somalraju, V. Murali, G. saha and 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

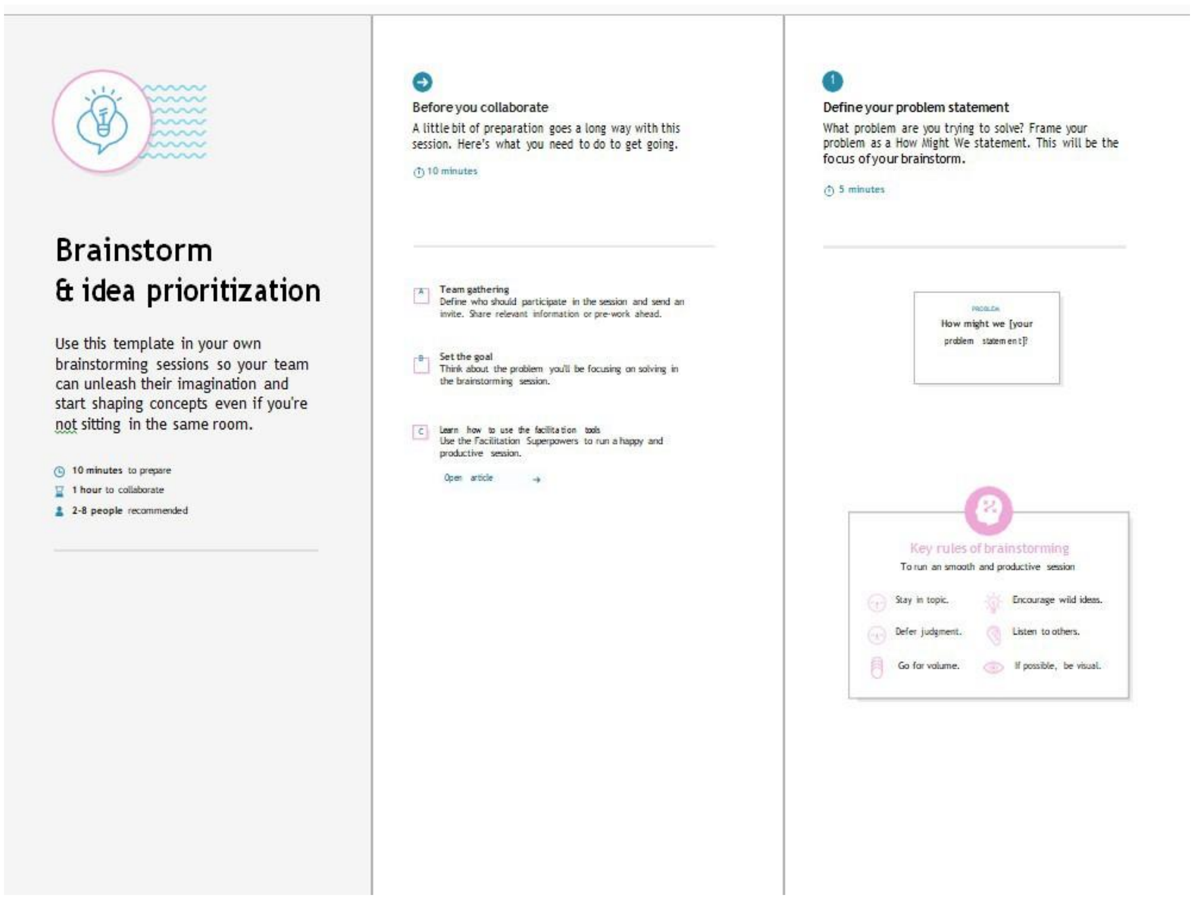
Among the various modes of transport, railways is one of the biggest modes of transport in the world. Though there are competitive threats from airlines, luxury buses, public transports, and personalized transports the problem statement is to answer the question “What are the problems faced by the passengers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

3. IDEATION AND PROPOSED SOLUTON

3.1 EMPATHY MAP CANVAS


what is your dream about the project?
What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
the railways and track safety is important is what I feel
We have the great vision of implementing new techniques for the process
from this we can create some great magic in the system
what technology u have in your vision?
What do they
EAR?
friends say
boss say
influencers say
SE
what the market
to be helpful to everyone and kind to them.
what is your behaviour of approach in public?
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
PAIN
fears
frustrations
obstacles
Railway transport is its inflexibility
GAIN
"wants" / needs
measures of success
obstacles
Railway is the safest form of transport
The Carrying capacity of the railway is extremely large

3.2 IDEATION & BRAINSTORMING



Brainstorm
& idea prioritization

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

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

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

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

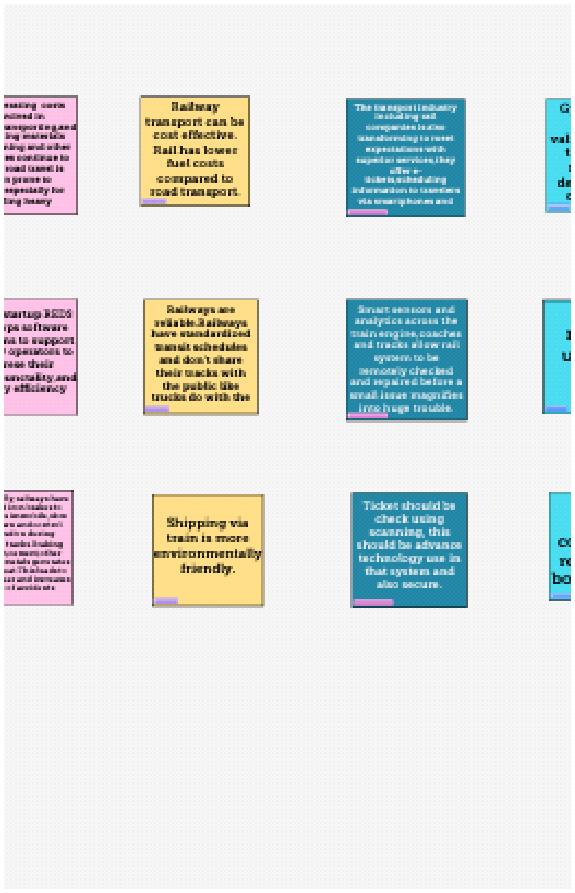
1 **Define your problem statement**
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
How might we [your problem statement]?

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.



easing costs
olved in
ansporting and
ing materials
ning and other
es continue to
road travel is
n prone to
especially for
ling heavy

Railway transport can be cost effective. Rail has lower fuel costs compared to road transport.

The transport industry including rail companies is also transforming to meet expectations with superior services, they offer e-tickets, scheduling information to travelers via smartphones and

startup REDS
rps software
ns to support
operators to
ease their
nctuality and
y efficiency

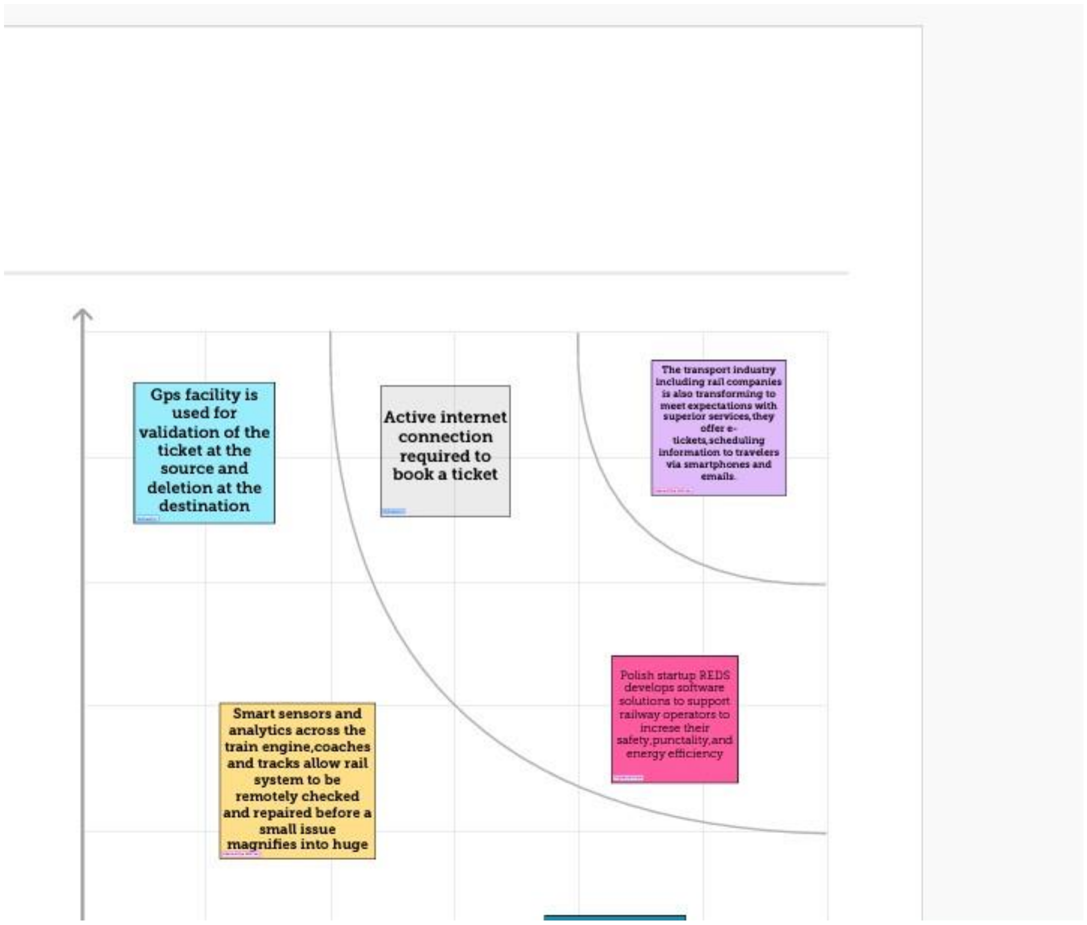
Railways are reliable. Railways have standardized transit schedules and don't share their tracks with the public like trucks do with the

Smart sensors and analytics across the train engine, coaches and tracks allow rail system to be remotely checked and repaired before a small issue magnifies into huge trouble.

[illegible]

Shipping via train is more environmentally friendly.

Ticket should be check using scanning, this should be advance technology use in that system and also secure.

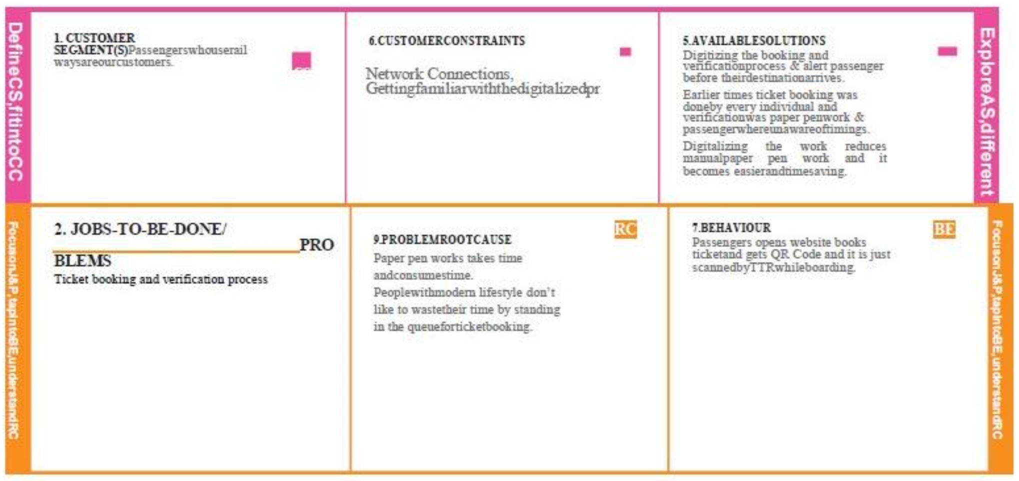

Gps facility is used for validation of the ticket at the source and deletion at the destination
Active internet connection required to book a ticket
The transport industry including rail companies is also transforming to meet expectations with superior services,they offer e-tickets,scheduling information to travelers via smartphones and emails.
Smart sensors and analytics across the train engine,coaches and tracks allow rail system to be remotely checked and repaired before a small issue magnifies into huge
Polish startup REDS develops software solutions to support railway operators to increse their safety,punctality,and energy efficiency

3.2 PROPOSED SOLUTION

.NO	PARAMETERS	DESCRIPTIONS
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 drastically increased 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

3.3 Problem Solution fit



DefineCS,fitintoCC

1. CUSTOMER SEGMENT(S) Passengerswhouserail waysareourcustomers.

6.CUSTOMERCONSTRAINTS

Network Connections, Gettingfamiliarwiththedigitalizedpr

5.AVAILABLESOLUTIONS

Digitizing the booking and verificationprocess & alert passenger before theirdestinationarrives.

Earlier times ticket booking was doneby every individual and verificationwas paper penwork & passengerwhereunawareoftimings.

Digitalizing the work reduces manualpaper pen work and it becomes easierandtimesaving.

ExploreAS,different

FocusonJ&P,tapintoBE,understandRC

2. JOBS-TO-BE-DONE/PROBLEMS

Ticket booking and verification process

9.PROBLEMROOTCAUSE RC

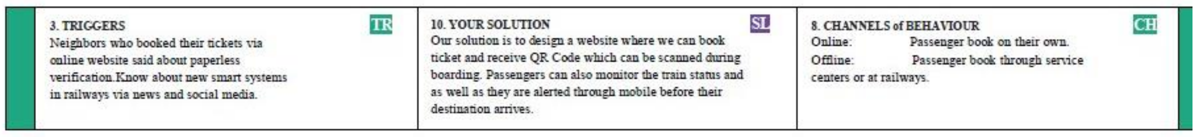
Paper pen works takes time andconsumestime.

Peoplewithmodern lifestyle don't like to wastetheir time by standing in the queueforticketbooking.

7.BEHAVIOUR BE

Passengers opens website books ticketand gets QR Code and it is just scannedbyTTRwhileboarding.

FocusonJ&P,tapintoBE,understandRC



3. TRIGGERS TR

Neighbors who booked their tickets via online website said about paperless verification.Know about new smart systems in railways via news and social media.

10. YOUR SOLUTION SL

Our solution is to design a website where we can book ticket and receive QR Code which can be scanned during boarding. Passengers can also monitor the train status and as well as they are alerted through mobile before their destination arrives.

8. CHANNELS of BEHAVIOUR CH

Online: Passenger book on their own.

Offline: Passenger book through service centers or at railways.

4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENTS

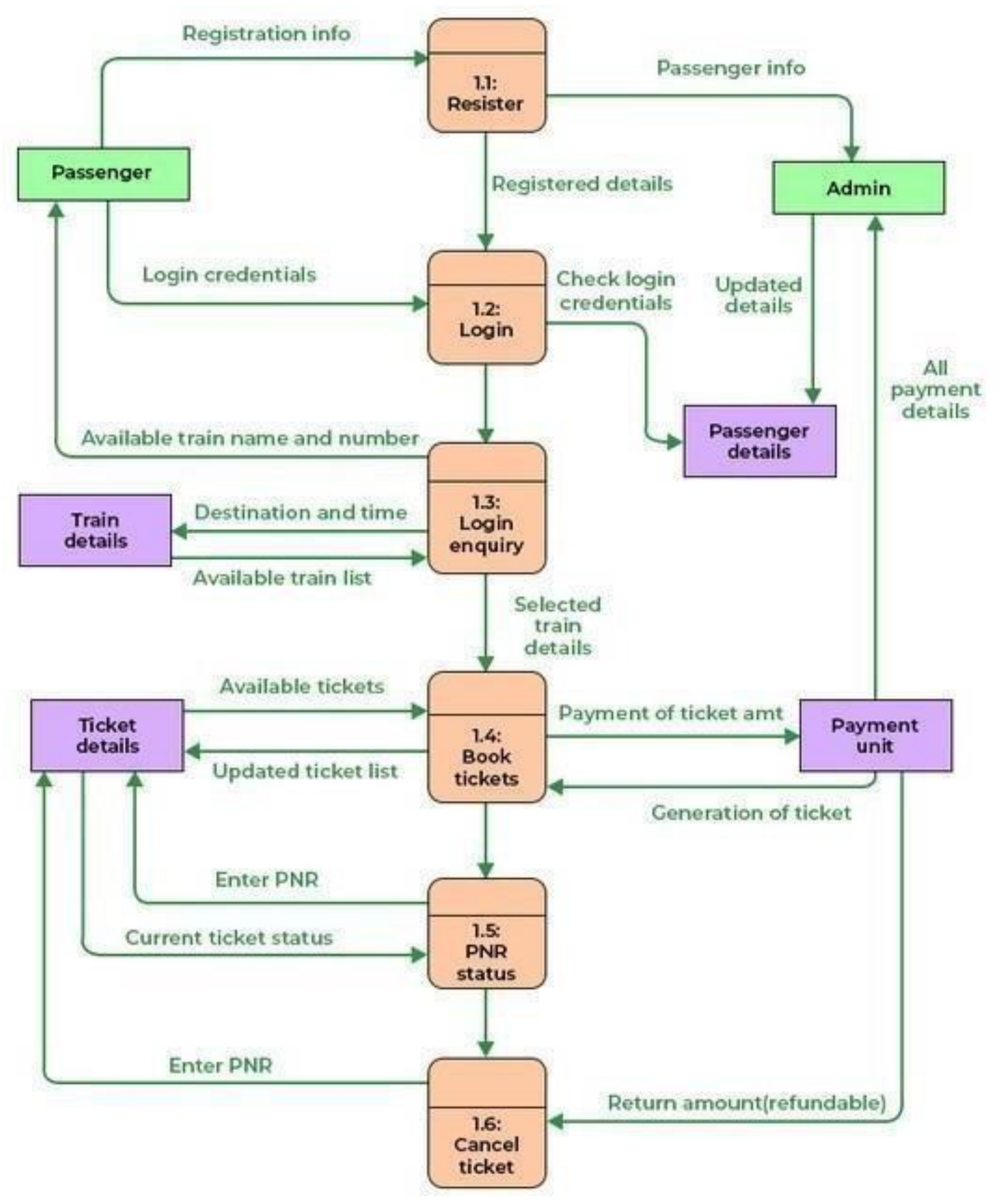
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Unique accounts	• Every online booking needs to be associated with an account • One account cannot be associated with multiple users
FR-2	Booking options	Search results should enable users to find the most recent and relevant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

4.2. NON-FUNCTIONAL REQUIREMENTS

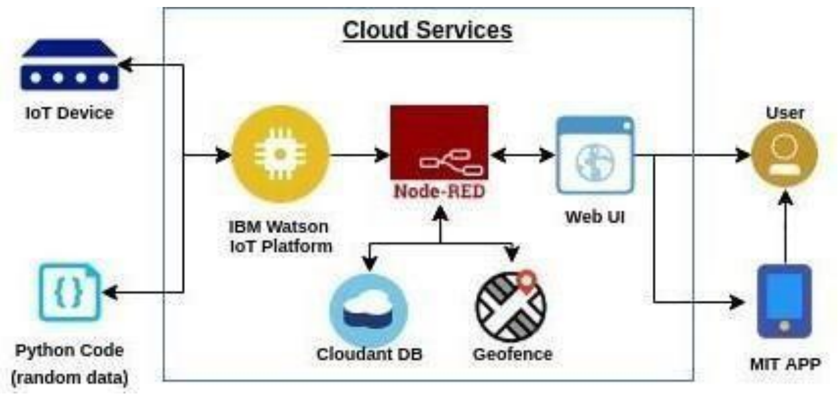
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Search results should populate within acceptable time limits
NFR-2	**Security**	System should visually confirm as well as send booking confirmation to the user's contact
NFR-3	**Reliability**	System should accept payments via different payment methods, like 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 avoid bots from booking tickets

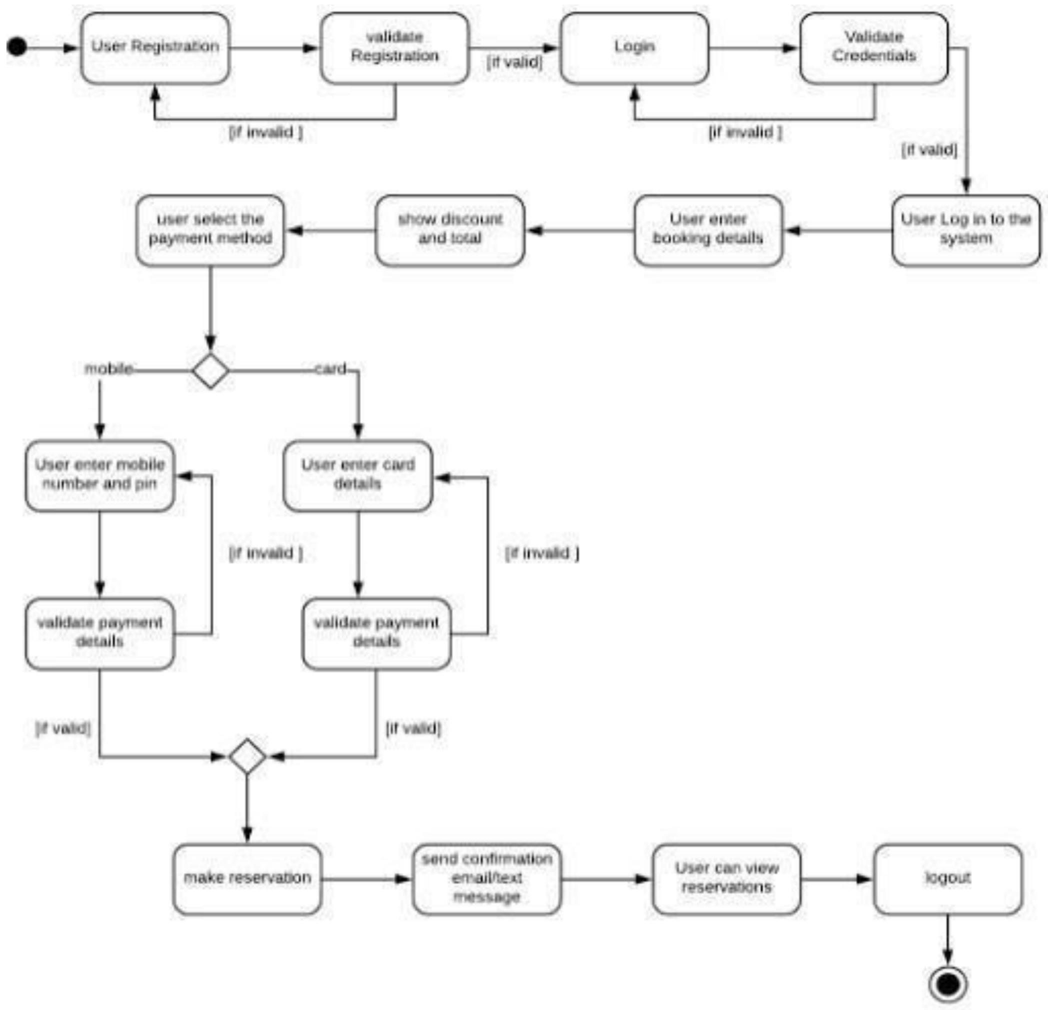
5. PROJECT DESIGN

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 (Mobile user, Web user)	Registration	USN-1	As a user, I can register through theform by Filling in mydetails	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 confirmationemail & clickconfirm.	High	Sprint-1
	Authentication/Login	USN-4	As a user, I can loginvia login id and password or throughOTP 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 thelist of trains availableconnecting 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 choosethe 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 confirmthe 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 Confirme d	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 informationsuch 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 makeit once resolved	Medium	Sprint-2
dministrator	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 informationfed.	High	Sprint-1

6. PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING& ESTIMATION

rint	unctional equirement pic)	ser Story umber	User Story / Task	tory Points	iority	eam Members
rint-1	egistration	SN-1	As a user, I can register through the form by illing in my details	1	High	kohila krishnaveni
rint-1		SN-2	As a user, I can register through hone umbers, Gmail, Facebook or other ocial sites	3	High	pandeeswari rajalakshmi
rint-1	onformation	SN-3	As a user, I will receive confirmation through mail or OTP once registration is uccessful	3	Low	kohila karishna pandeeavari rajalakshmi
rint-1	gin	SN-4	As a user, I can login via login id nd password or through OTP eceived on register phone number	1	Medium	kohilakarishna pandeeavari rajalakshmi sangeetha
rint-1	isplay Train details	SN-5	As a user, I can enter the start and destination to get the list of trains available connecting the bove	2	High	

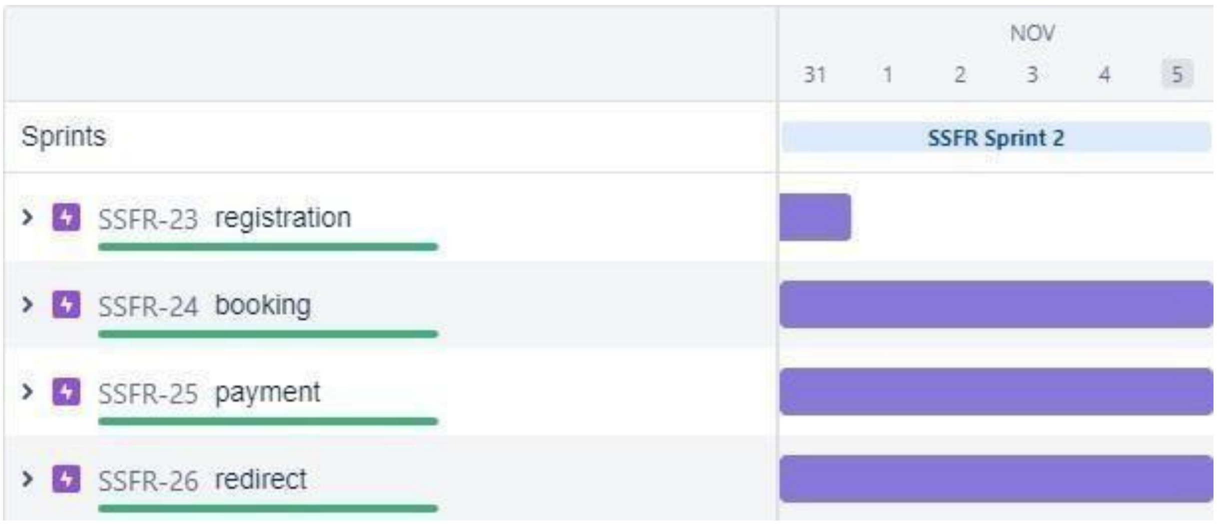
rint- 2	ooking	SN-6	As a use, I can provide the basic details such as name, age, gender etc…	3	Medium	Snge etha
rint- 2		SN-7	As a user, I can choose the class, seat/berth. If a preferred seat/berth isn’t available I can llocated based on the availability	3 rajalakshmi	High	kohila Krishna pandeeswari

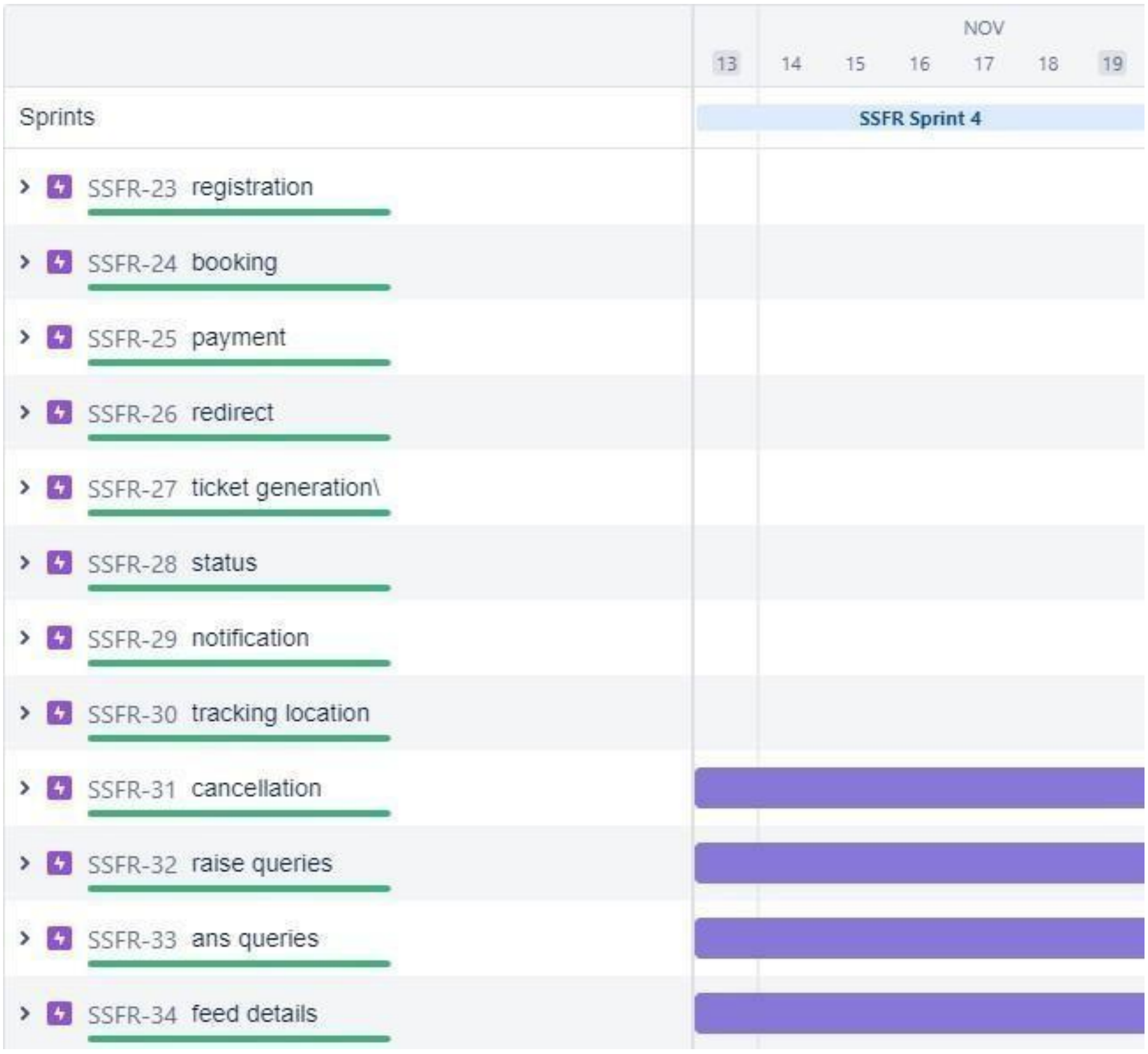
srint- 2	payment	SN-8	As a user, I can choose to pay through credit ard/debit card/UPI.	3	igh	veryone
srint- 2		SN-9	As a user, I will be redirected to the elected	2	edium	kohila Krishna
srint- 3	cket generation	SN-10	As a user, I can download the generated e- ticket for my journey along with the QR code which is used for authentication during my ourney.	1	ow	rajalaks hmi sangeeth a
rint- 3	cket status	SN-11	As a user, I can see the status of my icket Whether it’s confirmed/waiting/RAC.	2	igh	pandee swari rajalaksh mi
rint- 3	emainders notification	SN-12	s a user, I get remainders about my journey A day before my actual ourney.	1	gh	
rint- 3	cket cancellation	SN-13	s a user, I can track the train using GPS and can get information such as ETA, Current stop and elay	2	gh	kohila krishna
rint- 4		SN-14	s a user, I can cancel my tickets if there’s any Change of plan	1	gh	pandeeswari
rint- 4	aise queries	SN-15	s a user, I can raise queries through the query box or via mail.	2	edium	sangeetha rajalaksh mi
rint- 4	nswer the queries	SN-16	s a user, I will answer the questions/doubts aised by the customers.	2	gh	pandeeswari
rint- 4	ed details	SN-17	s a user, I will feed information about the trains delays and add extra seats if a new compartment is dded.	1	gh	rajala kshmi

6.2 SPRINT DELIVERY SCHEDULE

rint	**otal Story Points**	**Duration**	**print Start Date**	**print End Date (Planned)**	**tory Points Completed (as on Planned nd Date)**	**print Release Date (Actual)**
rint-1	0	Days	4 Oct 2022	9 Oct 2022	0	9 Oct 2022
rint-2	0	Days	1 Oct 2022	5 Nov 2022	0	Nov 2022
rint	otal Story Points	Duration	print Start Date	print End Date (Planned)	tory Points Completed (as on Planned End ate)	Sprint Release Date (Actual)
rint-3	0	Days	7 Nov 2022	2 Nov 2022	0	2 Nov 2022
rint-4	0	Days	4 Nov 2022	9 Nov 2022	0	9 Nov2022

6.1. REPORTS FROM JIRA





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

7. CODING AND SOLUTIONING

7.1. FEATURE 1

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

7.2. FEATURE 2

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

PYTHON SCRIPT TO GENERATE RANDOM GPS DATA

```
mport time
 import sys
mport ibmiotf.application
mport ibmiotf.device
mport random
mport requests
mport json
Provide your IBM Watson Device Credentials
rganization = MD07mg"
eviceType = "rasbperrypi"        #Credentials of Watson IoT sensor simulator
eviceId = "12345"
uthMethod = "token"
uthToken = "12345678"
 Initialize the device client.
=0
ry:
      deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-
oken": authToken}
      deviceCli = ibmiotf.device.Client(deviceOptions)
      #..............................................

xcept Exception as e:
      print("Caught exception connecting device: %s" % str(e))
      sys.exit()
 Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times
eviceCli.connect()
hile True:
   overpass_url = "http://overpass-api.de/api/interpreter"
   overpass_query = """
   [out:json];area[name="India"];(node[place="village"](area););out;
   """
   response = requests.get(
   overpass_url,
   params={'data': overpass_query}
   )
   coords = []
```

```
if response.status_code == 200:
        data = response.json()
        places = data.get('elements', [])
        for place in places:
            coords.append((place['lat'], place['lon']))
        print ("Got %s village coordinates!" % len(coords))
        print (coords[0])
    else:
        print("Error")

    i = random.randint(1,100)
    L = coords[i]
    #Send random gprs data to node-red to IBM Watson
    data = {"d":{ 'Latitude' : L[0], 'Longitude' : L[1]}}
    #print data
    def myOnPublishCallback():
        print("Published gprs location = ", L, "to IBM Watson")
    success = deviceCli.publishEvent("Data", "json", data, qos=0, on_publish=myOnPublishCallback)
    time.sleep(12)
    if not success:
        print("Not connected to IoTF")
    time.sleep(1)

deviceCli.disconnect()
```

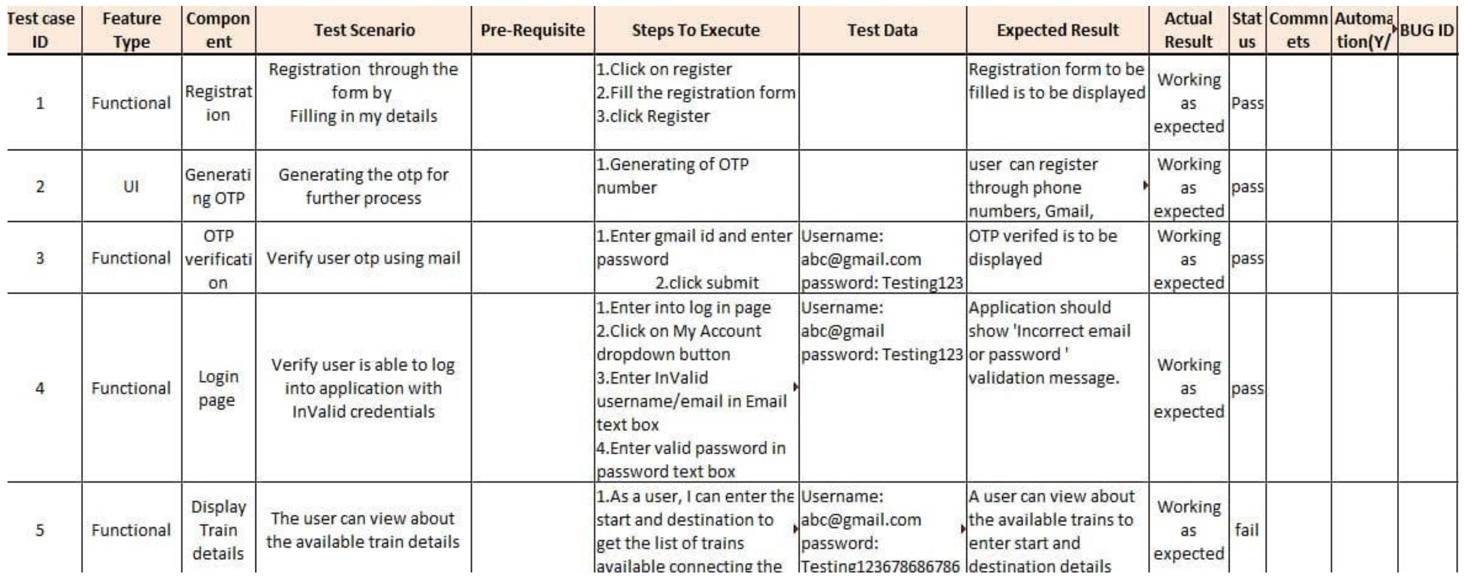
PYTHON CODE FOR TICKET GENERATION

```
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(self.Counter)
            print( "Ticket id will be:",_ticket_id)
        else:
            return False
    def get_ticket_id(self):
        return self.ticket_id
    def get_passenger_name(self):
        return self. passenger_name
    def get_source(self):
        if self._source=="Delhi":
            return self. source
        else:
            print("you have written invalid soure option")
            return None
    def get_destination(self):
        if self._destination=="Pune":
            return self. destination
        elif self._destination=="Mumbai":
            return self. destination
        elif self._destination=="Chennai":
            return self. destination
```

```
elif self._destination=="Kolkata":
    return self. destination
else:
    return None
```

8. TESTING

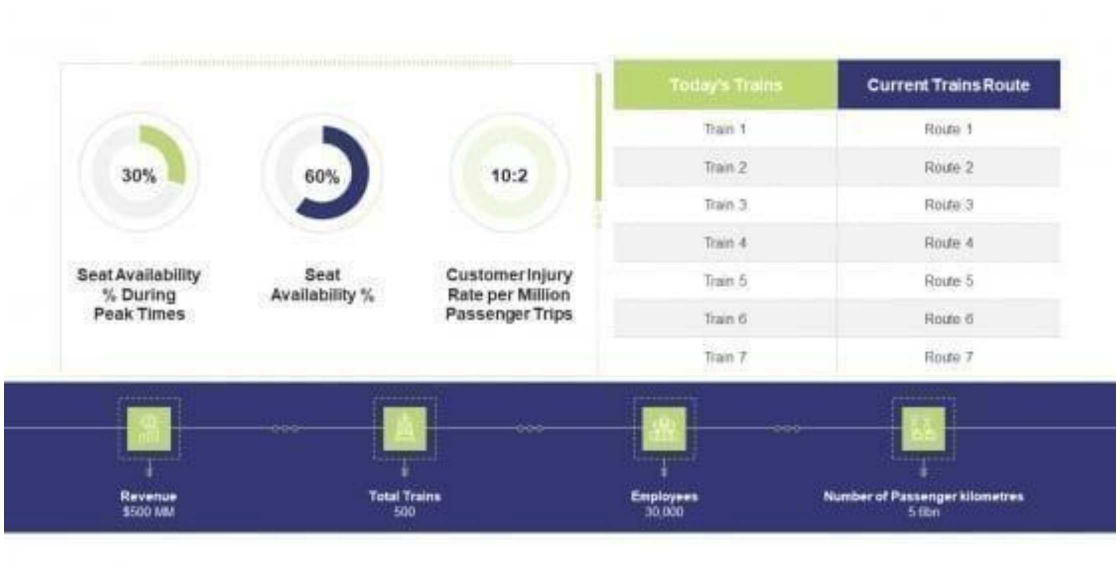
8.1.TEST CASES



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	Automation(Y/	BUG ID
1	Functional	Registration	Registration through the form by Filling in my details		1.Click on register 2.Fill the registration form 3.click Register		Registration form to be filled is to be displayed	Working as expected	Pass			
2	UI	Generating OTP	Generating the otp for further process		1.Generating of OTP number		user can register through phone numbers, Gmail,	Working as expected	pass			
3	Functional	OTP verification	Verify user otp using mail		1.Enter gmail id and enter password 2.click submit	Username: abc@gmail.com password: Testing123	OTP verifed is to be displayed	Working as expected	pass			
4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter into log in page 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box	Username: abc@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	pass			
5	Functional	Display Train details	The user can view about the available train details		1.As a user, I can enter the start and destination to get the list of trains available connecting the	Username: abc@gmail.com password: Testing123678686786	A user can view about the available trains to enter start and destination details	Working as expected	fail			

9. RESULTS

9.1. PERFORMANCE METRICS



30%
Seat Availability % During Peak Times

60%
Seat Availability %

10:2
Customer Injury Rate per Million Passenger Trips

Today's Trains	Current Trains Route
Train 1	Route 1
Train 2	Route 2
Train 3	Route 3
Train 4	Route 4
Train 5	Route 5
Train 6	Route 6
Train 7	Route 7

Revenue
$500 MM

Total Trains
500

Employees
30,000

Number of Passenger kilometres
5.6bn

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.

11. CONCLUSION

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

12. FUTURE SCOPE

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

PYTHON CODE FOR LOGIN

```
from tkinter import *
import sqlite3
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)
#==============================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()
```

PYTHON CODE 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())
```

PYTHON CODE 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")
```

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)

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()
```

START AND DESTINATION

```
# import module
from bs4 import BeautifulSoup

# user define function
# Scrape the data
def getdata(url):
    r = requests.get(url)
    return r.text



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

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

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

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

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

```
# Display the result
for item in result:
    if item != "":
        print(item)
```

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
```

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="Białystok",
)
user1.set_unusable_password()
group_name = "employer"
_profile1, _ = UserProfile.objects.get_or_create(
    user=user1,
      date_of_birth=datetime.now() - timedelta(days=6600),
    group=GroupTypeChoices(group_name).name,
    address="Myśliwska 14",
    postal_code="15-569",
    phone_number="+48100200300",
)
# user2 - employee
user2, _ = User.objects.get_or_create()
    email="bar@foo.com",
    first_name="Employee",
    last_name="Testowy",
    city="Białystok",
)
user2.set_unusable_password()
group_name = "employee"
_profile2, _ = UserProfile.objects.get_or_create()
    user=user2,
      date_of_birth=datetime.now() - timedelta(days=7600),
    group=GroupTypeChoices(group_name).name,
    address="Myśliwska 14",
    postal_code="15-569",
    phone_number="+48200300400",
)
response_customer = stripe.Customer.create()
    email=user.email,
    description=f"EMPLOYER - {user.get_full_name}",
    name=user.get_full_name,
    phone=user.profile.phone_number,
)
user1.stripe_id = response_customer.stripe_id
```

```
user1.save()
mcc_code, url = "1520", "https://www.softserveinc.com/"
response_ca = stripe.Account.create()
    type="custom",
    country="PL",
    email=user2.email,
    default_currency="pln",
    business_type="individual",
    settings={"payouts": {"schedule": {"interval": "manual", }}},
    requested_capabilities=["card_payments", "transfers", ],
    business_profile={"mcc": mcc_code, "url": url},
    individual={
        "first_name": user2.first_name,
        "last_name": user2.last_name,
        "email": user2.email,
        "dob": {
            "day": user2.profile.date_of_birth.day,
             "month": user2.profile.date_of_birth.month,
            "year": user2.profile.date_of_birth.year,
        },
        "phone": user2.profile.phone_number,
        "address": {
            "city": user2.city,
            "postal_code": user2.profile.postal_code,
            "country": "PL",
            "line1": user2.profile.address,
        },
    },
)
user2.stripe_id = response_ca.stripe_id
user2.save()
tos_acceptance = {"date": int(time.time()), "ip": user_ip},
stripe.Account.modify(user2.stripe_id, tos_acceptance=tos_acceptance)
passport_front = stripe.File.create(
    purpose="identity_document",
    file=_file, # ContentFile object
    stripe_account=user2.stripe_id,
)
individual = {
    "verification": {
        "document": {"front": passport_front.get("id"),},
        "additional_document": {"front": passport_front.get("id"),},
    }
}
```

```
stripe.Account.modify(user2.stripe_id, individual=individual)
new_card_source = stripe.Customer.create_source(user1.stripe_id, source=token)
stripe.SetupIntent.create(
    payment_method_types=["card"],
    customer=user1.stripe_id,
    description="some description",
    payment_method=new_card_source.id,
)
payment_method = stripe.Customer.retrieve(user1.stripe_id).default_source
payment_intent = stripe.PaymentIntent.create(
    amount=amount,
    currency="pln",
    payment_method_types=["card"],
    capture_method="manual",
    customer=user1.stripe_id, # customer
    payment_method=payment_method,
    application_fee_amount=application_fee_amount,
    transfer_data={"destination": user2.stripe_id}, # connect account
    description=description,
    metadata=metadata,
)
payment_intent_confirm = stripe.PaymentIntent.confirm(
    payment_intent.stripe_id, payment_method=payment_method
)
stripe.PaymentIntent.capture(
    payment_intent.id, amount_to_capture=amount
)
stripe.Balance.retrieve(stripe_account=user2.stripe_id)
stripe.Charge.create(
    amount=amount,
    currency="pln",
    source=user2.stripe_id,
    description=description
)
stripe.PaymentIntent.cancel(payment_intent.id)
        unique_together = ("user", "group")
```

REDIRECT:

```
import logging
import attr
from flask import Blueprint, flash, redirect, request, url_for
from flask.views import MethodView
from flask_babelplus import gettext as _
from flask_login import current_user, login_required
from pluggy import HookimplMarker
@attr.s(frozen=True, cmp=False, hash=False, repr=True)
class UserSettings(MethodView):
    form = attr.ib(factory=settings_form_factory)
    settings_update_handler = attr.ib(factory=settings_update_handler)
    decorators = [login_required]
    def get(self):
        return self.render()
    def post(self):
        if self.form.validate_on_submit():
            try:
                 self.settings_update_handler.apply_changeset(
                     current_user, self.form.as_change()
                )
            except StopValidation as e:
                self.form.populate_errors(e.reasons)
                return self.render()
            except PersistenceError:
                logger.exception("Error while updating user settings")
                flash(_("Error while updating user settings"), "danger")
                return self.redirect()
            flash(_("Settings updated."), "success")
            return self.redirect()
        return self.render()
    def render(self):
        return render_template("user/general_settings.html", form=self.form)
    def redirect(self):
        return redirect(url_for("user.settings"))
@attr.s(frozen=True, hash=False, cmp=False, repr=True)
class ChangePassword(MethodView):
    form = attr.ib(factory=change_password_form_factory)
    password_update_handler = attr.ib(factory=password_update_handler)
    decorators = [login_required]
    def get(self):
        return self.render()
    def post(self):
        if self.form.validate_on_submit():
            try:
                 self.password_update_handler.apply_changeset(
                     current_user, self.form.as_change()
                )
            except StopValidation as e:
                self.form.populate_errors(e.reasons)
                return self.render()
            except PersistenceError:
                logger.exception("Error while changing password")
                flash(_("Error while changing password"), "danger")
                return self.redirect()
            flash(_("Password updated."), "success")
            return self.redirect()
        return self.render()
    def render(self):
        return render_template("user/change_password.html", form=self.form)
    def redirect(self):
        return redirect(url_for("user.change_password"))
@attr.s(frozen=True, cmp=False, hash=False, repr=True)
class ChangeEmail(MethodView):
    form = attr.ib(factory=change_email_form_factory)
    update_email_handler = attr.ib(factory=email_update_handler)
    decorators = [login_required]
    def get(self):
        return self.render()
    def post(self):
        if self.form.validate_on_submit():
            try:
                 self.update_email_handler.apply_changeset(
                     current_user, self.form.as_change()
                )
            except StopValidation as e:
                self.form.populate_errors(e.reasons)
                return self.render()
            except PersistenceError:
                 logger.exception("Error while updating email")
                flash(_("Error while updating email"), "danger")
                return self.redirect()
            flash(_("Email address updated."), "success")
            return self.redirect()
        return self.render()
    def render(self):
        return render_template("user/change_email.html", form=self.form)
    def redirect(self):
        return redirect(url_for("user.change_email"))
```

SEATS 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
```

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()
```

13.2 GITHUB LINK :

https://github.com/IBM-EPBL/IBM-Project-18449-1659685334