Smart Solutions for Railways

Category: ***Internet of Things***

PROJECT REPORT

SUBMITTED BY

Team ID: PNT2022TMID31318

NAME	**REGISTER NUMBER**
1. GIRIJA K	710719104033
2. AARTHI S	710719104002
3. ANU SANKARI S	710719104013
4. KIRTHIVARSINI M	710719104054

INPARTIAL FULFILLMENT FOR THE

AWARD OF THE DEGREE

Of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND

ENGINEERING

DR.N.G.P. INSTITUTE OF TECHNOLOGY

COIMBATORE-641402

ANNA UNIVERSITY: CHENNAI - 600025

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

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

1.1 Project Overview

As trains are one of the most preferred modes of transportation among middle class and impoverished people as it attracts for its amenities.
Simultaneously there is an increase at risk from thefts and accidents like chain snatching, derailment, fire accident. In order to avoid or in better words to stop all such brutality we came up with a solution by providing an application which can be accessed by the user after booking their tickets. With a single click this app addresses issues by sending a text message to TC and RPF as an alert. In our project we use Node-Red service, app-development, IBM cloud platform to store passenger data.

1.2 Purpose

The purpose of this project is to report and get relived from the issues related to trains.

2. LITERATURE SURVEY

2.1 Existing problem

A Web page is designed for the public where they can book tickets by seeing the available seats.
After booking the train, the person will get a QR code which has to be shown to the Ticket Collector while boarding the train.
The ticket collectors can scan the QR code to identify the personal details.
A GPS module is present in the train to track it. The live status of the journey is updated in the Web app continuously
All the booking details of the customers will be stored in the database with a unique ID and they can be retrieved back when the Ticket Collector scans the QR Code.

2.2 References

S.NO	TITLE	AUTHOR	YEAR	KEY TECHNOLOGY
1	Main geotechnical problems of railways and roads in kriolitozone and their solutions.	Kondratiev, Valentin G	2017	Main problems in railways
2	Construction and Building Materials	Sañudo, Roberto, Marina Miranda, Carlos García, and David García-Sanchez	2019	Drainage in railways
3	Problems of Indian Railways	Benjamin	2021	Common problems in Indian railways
4	A comparative study of Indian and worldwide railways.	Sharma,Sunil Kumar, and Anil Kumar	2014	Study of Indian railways
5	Ticketing solutions for Indian railways using RFID technology	Prasanth,Venugopal, and K.P. Soman	2009	Solution for ticketing using RFID

2.3 Problem Statement Definition

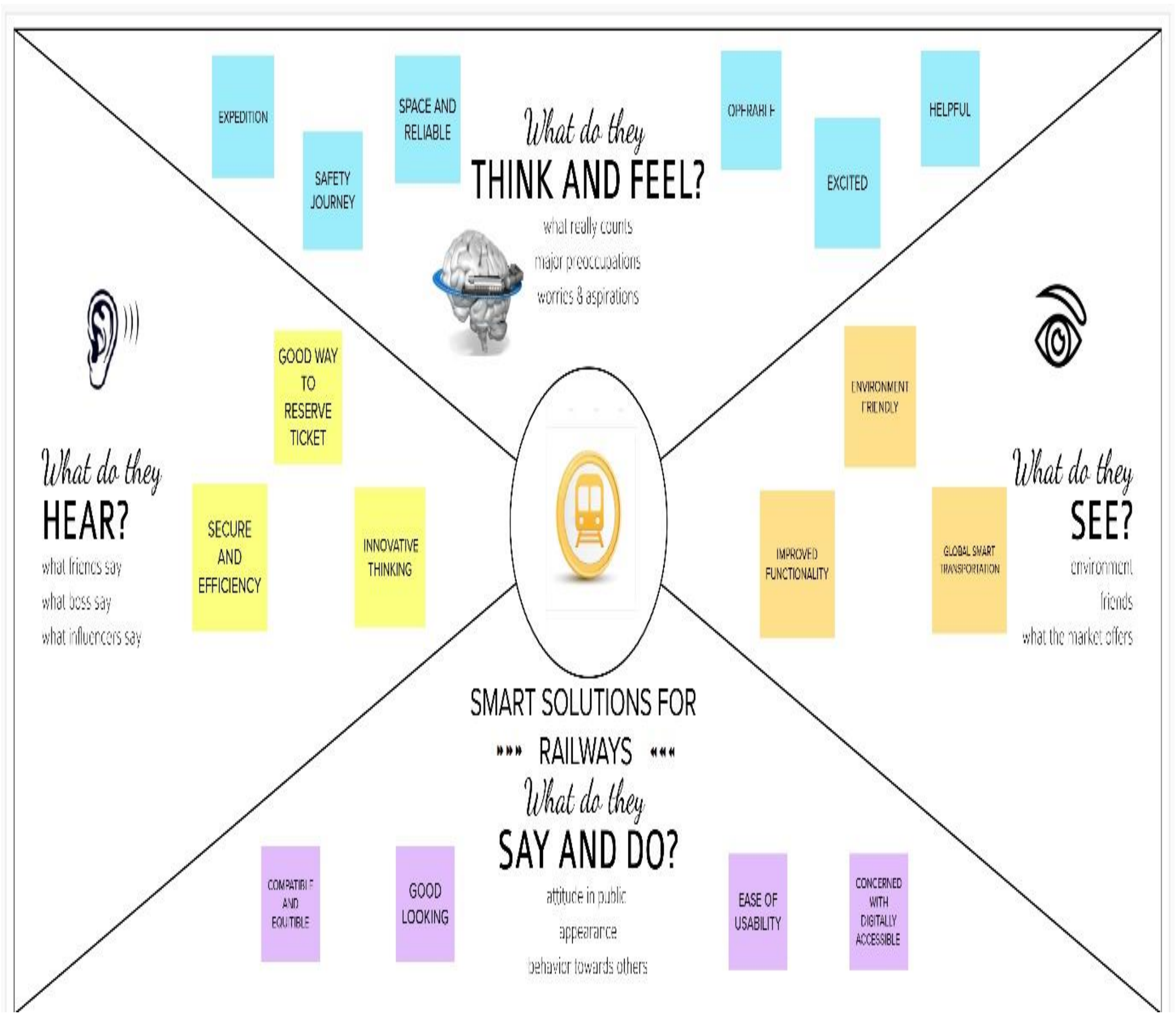
Smart Solutions for railways are designed to reduce the work load of the user and the use of paper.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map

CanvasOnline Ticket

Booking:



3.2 Ideation & Brainstorming

- Creating an Application for passengers
- Digital Railway solution
- Secure Access to data
- Notify about monthly bill payments
- Track expenses
- The IOT connected trains
- Send email alert on exceeding Bill expenses
- Detailed report at end of each month

Idea prioritization:

- To protect from:
- Ticket booking Jamming
- Fire accident
- Theft
- Robbery

Include Features like:

- Tracking management
- QR code

3.3 Proposed Solution

S.No	Parameter	Description
1	Problem Statement (Problem to be solved)	*Smart Solutions for railways is designed to reduced the work load of the user and also the use of paper and also provides the live location of the train.

		*In their busy schedule as fast roaming world public in need of online booking process. The queues in front of the ticket counters in railway stations have been drastically increased over the period of time. *Ticket reservation through counter is not sufficient and convenient for the passengers. The passengers are struggling to get tickets in the time from ticket counters. So they like to switch over online ticket booking.
2	Idea/ Solution description	*A webpage is designed in which the user can book tickets and will be provided with a QR code which will be shown to the ticket collector and the ticket collector will be scanning the QR code to get the passenger details. * The webpage also shows the live locations of the train by placing a GPS module in the train. The location of the journey will be updated continuously in the webpage. * The booking details of the user will be stored in the database which can be retrieved anytime
3	Novelty/ Uniqueness	*A QR code will be provided by the webpage to the user which will reduce the paper work. *All the booking details of the customers will be stored in the database with a unique ID and they can be retrieved back when the Ticket Collector scans the QR Code.You can also view interactive seat map.

4	Social Impact/ Customer Satisfaction	*The booking tickets is made easy to use and it is also reliable and no need to go to station for booking tickets and the transaction process is also made easy. *One can manage online ticket booking and apply for a cancellation in case of any change in plans . *The customer will be notified on email as well as cell phone on all confirmation and cancellations
5	Business Model (Revenue Model)	*With this solution - By using this application, the customer can schedule their destination, view availability of the seat, view interactive seat map and select their seat for their convenience. Moreover, it enables your customers to organize trips and daily shuttles effortlessly and it also reduces the carrying of tickets. The customer can also watch the current location of the train. *without this solution – they have to travel to the station to book tickets and also have to carry their tickets to show to ticket collector.
6	Scalability of the solution	1. No need of taking print out. 2. Counter ticket has to be handled with care, but SMS on mobile is more than enough. 3. You are becoming environment friendly and contributing for greener planet by ignoring printout. 4. No need of taking out wallet and showing your ticket to TTR, just tell your name to TTR that you are passenger with a valid proof. 5. While booking counter ticket you had to carry cash and while booking E- ticket you are paying through online directly from bank which makes work more easy for you.

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through form Registration through Gmail Registration through LinkedIn Registration through Mobile number
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP Confirmation via call Confirmation via message
FR-3	Journey details	Provides From and To information and date of travel and seat.
FR-4	Select Trains	Select the appropriate trains among the list and also based on the seat availability, time, date of travel.
FR-5	Book and add passenger	Fill the essential details such as name, contact details and age, government ID.
FR-6	Proceed to pay	Select an appropriate payment options among UPI, Internet banking, credit card, debit card.
FR-7	Ticket confirmation and Invoices	Ticket confirmation status is send to their registered email id or phone number.
FR-8	Database management	Entire Journey details will be stored in the server.
FR-9	Food Service	Foods are available for the registered passengers in an effective manner.

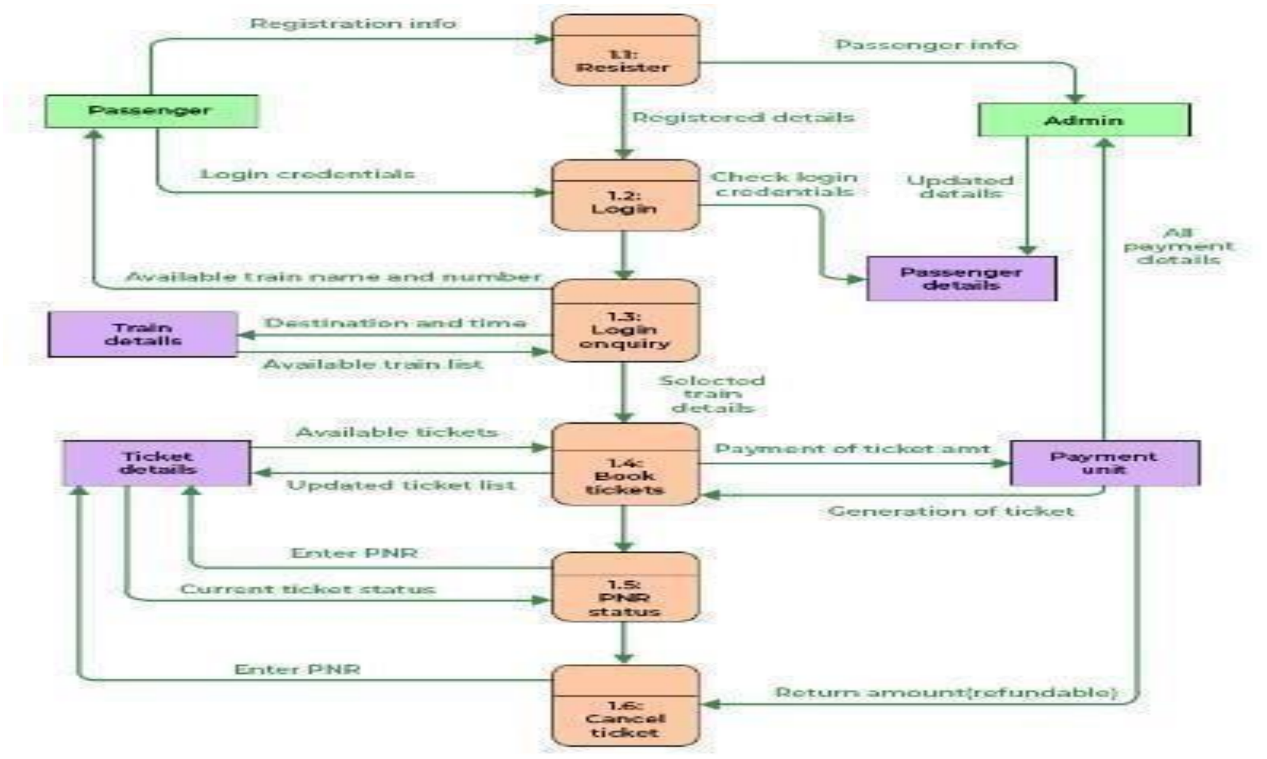
4.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Availability of e-tickets with QR generation instead of physical one.
NFR-2	**Security**	It protects the details of a passenger against man in the middle and denial of service attacks.
NFR-3	**Reliability**	It enables the user to securely use the app which provides maximum trust to the user.
NFR-4	**Performance**	No server down problem.
NFR-5	**Availability**	Accessibility through website or application anytime and from anywhere.
NFR-6	**Scalability**	Number of users concurrently interacting with our web application with higher reliability.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution Architecture

As trains are one of the most preferred modes of transportation amongmiddle class and impoverished people as it attracts for its amenities. Simultaneously there is an increase at risk from thefts and accidents like chain-snatching, derailment, fire accident. In order to avoid or in better words to stop all such brutality we came up with a solution by providing an application which can be accessed by the user after booking their tickets. With a single click this app addresses issues by sending a text message to TC and RPF as an alert. In our project we use Node-Red service, app-development, IBM cloud platform to store passenger data.

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	UI	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
Customer (Mobile user)	Login Page	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
Customer (Mobile user)	Reserving ticket	USN-3	As a user, I can register for the application and enter the details for reserving the ticket.	I can view, modify the details	High	Sprint-2
Customer (Webuser)	Reserving ticket	User	Enter the details and click submit button to book ticket	I can book tickets and get QR code	High	Sprint-2
Customer (Mobile user)	Dashboard	Users	The details can be stored and retrieved	I can change the informati on when required	Medium	Sprint-3
Customer Care Executive	Connecting the service provider	Custome r	Connects with the service by logging in	Can get connected with the server	Medium	Sprint-3
Administrator	Provides the services	Admin	The data is given by the user	Can add or update the data provided by the user	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

STEP 1	Identify the problem
STEP 2	Draft the problem statement and abstract
STEP 3	List the requirement
STEP 4	Write the appropriate code
STEP 5	Run in the suitable form
STEP 6	Test the created code and check the designed prototype
STEP 7	The solution is established

6.2 Reports from

JIRASPRINT 1

```
#include <LiquidCrystal.h>

LiquidCrystal 1cd(5,6,8,9,10,11); int red1ed = 2; int green1ed = 3;
int buzzer = 4; int sensor = A0;
int sensorThresh = 400;
void setup()
{
pinMode(red1ed, OUTPUT); pinMode(green1ed,OUTPUT);
pinMode(buzzer,OUTPUT); pinMode(sensor,INPUT); serial.begin(9600);
1cd.begin(16,2);
}
Void loop()
{
 int analogValue = analogRead(sensor); Serial.print(analogvalue);
if(analogValue>sensorThresh)
      {
         digitalWrite(red1ed,HIGH); digit1Weite(green1ed,LOW);
tone(buzzer,1000,10000);

        1cd.clear();

        1cd.setCursor(0,1);

        1cd.print("RAILWAYS"); delay(1000);

        1cd.clear();

        1cd.setCursor(0,1);

        1cd.print("SMART SOLUTION"); delay(1000);

        } else {
         digitalWrite(greenlad,HIGH); digitalWrite(red1ed,LOW);

         noTone(buzzer); 1cd.clear(); 1cd.setCursor(0,0);
```

```
            1cd.print("SAFE"); delay(1000);

              1cd.clear();

              1cd.setCursor(0,1);

            1cd.print("ALL CLEAR"); delay(1000);

          }

     }
```

SPRINT 2

Main Program:

```
importwiotp.sdk.device
importtime importrandom
myConfig={
"identity":{
"orgId":"gagtey",
"typeId":"GPS",
"deviceId":"12345"
},
"auth":{
"token":"12345678"
}
}
defmyCommandcallback(cmd):
print("messagereceivedfromIBMIOTPlatform:%s"%cmd.data['command'])
m=cmd.data['command']
client=wiotp.sdk.device.deviceclient(config=myConfig,logHandlers=None)
client.connect()
defpub(data):


client.publishEvent(eventId="status",msgFormat="json",data=mydata,qos=0,
print("publishedatasuccessfully:%s",mydata)
whileTrue:
mydata={'name':'Train1','lat':17.6387448,'lon':78.4754336}
```

```
pub(myData)
time.sleep(3)
#mydata={'name':'Train2','lat':17.6387448,'lon':78.4754336)
#pub(myData)
#time.sleep(3)
mydata={'name':'Train1','lat':17.6341908,'lon':78.4744722)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6340889,'lon':78.4745052)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6248626,'lon':78.4720259)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6188577,'lon':78.4698726)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6132382,'lon':78.4707318)
pub(myData)
time.sleep(3)
client.commandCallback=mycommanCallbak
client.disconnect()
```

Code:

```
importcv2
importnumpyasnp
importtime
importpyzbar.pyzbaraspuzbar
fromibmcloudant.cloudant_v1importcloudantv1

fromibmcloudantimportcouchDbsessionAuthenticator
fromibm_cloud_sdk_core.AuthenticatorsimportBasicAuhtenticator
authenticator=BasicAuthenticator('apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz','b0ab119f45d3e6255eabb978)
service=cloudantv1(authenticator=authenticator)
```

```
service.set_service_url('https:/ apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb978
cap=cv2.videoCapture(0)
font=cv2.FONT_HERSHEY_PLAIN
whileTrue:
_,frame=cap.read(0)
decodeObjects=pyzbar.decode(frame)
forobjindecodeObjects:
#print("Data",obj.data)
a=obj.data.decode('UTF-8')
cv2.putText(frame,"Ticket",(50,50),font,2,(255,0,0),3)
#print(a)
try:
responce=service.get_document(db='booking',doc_id=a).get_result()
print(response)
time.sleep(5)
exceptExceptionase:
print("NotvalidTicket")
time.sleep(5)
cap.imshow("Frame",frame)
ifcv2.waitKey{1}&0XFF==ord('q'):
break
cap.release()
cv2.destroyAllWindows()
client.disconnect()
```

SPRINT 3

- This project presents its first ever digital event dedicated to rail transport, the “Smart Mobility Experience” which will take place on March 24th. This event will be the occasion for clients and partners of the rail ecosystem, to discover new products and major innovations, as well as to exchange about the digitalization and future of rail.
- for improved service performance and energy efficiency, and to boost the

attractiveness for users.

- It helps transporting passengers safely, and with best possible experience, supervises operations with accurate situation awareness, and optimizes transport service efficiency.
- Using digital technologies such as IoT, cloud and web IT, data analytics , it designs innovative solutions such as digital signalling, train autonomy, mobile ticketing, passenger flow analytics, data driven operation control, smart maintenance, which will drastically impact the way we all travel.
- Provide real-time passenger density insights to public transport operators
- The solution helps alleviate crowding by reducing busy times, and consequently enhances overall passenger safety, comfort, and travel experience.
- The targeted performances of density accuracy are above 90%.

In Hand's Connectivity Solution for Rail

Transit:MAIN:

```
importwiotp.sdk.device
importtime importrandom
myConfig={
"identity":{
"orgId":"gagtey",
"typeId":"GPS",
"deviceId":"12345"
},
"auth":{
"token":"12345678"
}
}
defmyCommandcallback(cmd):
print("messagereceivedfromIBMIOTPlatform:%s"%cmd.data['command'])
m=cmd.data['command']
client=wiotp.sdk.device.deviceclient(config=myConfig,logHandlers=None)
```

```
client.connect()
defpub(data):
client.publishEvent(eventId="status",msgFormat="json",data=mydata,qos=0,
print("publishedatasuccessfully:%s",mydata)
whileTrue:
mydata={'name':'Train1','lat':17.6387448,'lon':78.4754336)
pub(myData)
time.sleep(3)
#mydata={'name':'Train2','lat':17.6387448,'lon':78.4754336)
#pub(myData)
#time.sleep(3)
mydata={'name':'Train1','lat':17.6341908,'lon':78.4744722)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6340889,'lon':78.4745052)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6248626,'lon':78.4720259)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6188577,'lon':78.4698726)
pub(myData)
time.sleep(3)

mydata={'name':'Train1','lat':17.6132382,'lon':78.4707318)
pub(myData)
time.sleep(3)
client.commandCallback=mycommanCallbak
client.disconnect()
```

PROGRAM:

```
importcv2
importnumpyasnp
importtime
```

```
importpyzbar.pyzbaraspuzbar
fromibmcloudant.cloudant_v1importcloudantv1
fromibmcloudantimportcouchDbsessionAuthenticator
fromibm_cloud_sdk_core.AuthenticatorsimportBasicAuhtenticator
authenticator=BasicAuthenticator('apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz','b0ab119f45d3e6255eabb978)
service=cloudantv1(authenticator=authenticator)
service.set_service_url('https:/ apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb978
cap=cv2.videoCapture(0)
font=cv2.FONT_HERSHEY_PLAIN
whileTrue:
_,frame=cap.read(0)
decodeObjects=pyzbar.decode(frame)
forobjindecodeObjects:
#print("Data",obj.data)
a=obj.data.decode('UTF-8')
cv2.putText(frame,"Ticket",(50,50),font,2,(255,0,0),3)
#print(a)
try:
responce=service.get_document(db='booking',doc_id=a).get_result()


print(response)
time.sleep(5)
exceptExceptionase:
print("NotvalidTicket")
time.sleep(5)
cap.imshow("Frame",frame)
ifcv2.waitKey{1}&0XFF==ord('q'):
break
cap.release()
cv2.destroyAllWindows()
client.disconnect()
```

SPRINT 4

Main:

```
importwiotp.sdk.device
importtime importrandom
myConfig={
"identity":{
"orgId":"gagtey",
"typeId":"GPS",
"deviceId":"12345"
},
"auth":{
"token":"12345678"
}
}
defmyCommandcallback(cmd):
print("messagereceivedfromIBMIOTPlatform:%s"%cmd.data['command'])
m=cmd.data['command']
client=wiotp.sdk.device.deviceclient(config=myConfig,logHandlers=None)
client.connect()
defpub(data):
client.publishEvent(eventId="status",msgFormat="json",data=mydata,qos=0,
print("publishedatasuccessfully:%s",mydata)
whileTrue:
mydata={'name':'Train1','lat':17.6387448,'lon':78.4754336)
pub(myData)
time.sleep(3)
#mydata={'name':'Train2','lat':17.6387448,'lon':78.4754336)
#pub(myData)
#time.sleep(3)
mydata={'name':'Train1','lat':17.6341908,'lon':78.4744722)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6340889,'lon':78.4745052)
pub(myData)
```

```
time.sleep(3)
mydata={'name':'Train1','lat':17.6248626,'lon':78.4720259)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6188577,'lon':78.4698726)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6132382,'lon':78.4707318)
pub(myData)
time.sleep(3)
client.commandCallback=mycommanCallbak
client.disconnect()
```

Program:

```
importcv2
importnumpyasnp
importtime
importpyzbar.pyzbaraspuzbar

fromibmcloudant.cloudant_v1importcloudantv1
fromibmcloudantimportcouchDbsessionAuthenticator
fromibm_cloud_sdk_core.AuthenticatorsimportBasicAuhtenticator
authenticator=BasicAuthenticator('apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz','b0ab119f45d3e6255eabb978)
service=cloudantv1(authenticator=authenticator)
service.set_service_url('https:/ apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb978
cap=cv2.videoCapture(0)
font=cv2.FONT_HERSHEY_PLAIN
whileTrue:
_,frame=cap.read(0)
decodeObjects=pyzbar.decode(frame)
forobjindecodeObjects:
#print("Data",obj.data)
```

```
a=obj.data.decode('UTF-8')
cv2.putText(frame,"Ticket",(50,50),font,2,(255,0,0),3)
#print(a)
try:
responce=service.get_document(db='booking',doc_id=a).get_result()
print(response)
time.sleep(5)
exceptExceptionase:
print("NotvalidTicket")
time.sleep(5)
cap.imshow("Frame",frame)
ifcv2.waitKey{1}&0XFF==ord('q'):
break
cap.release()
cv2.destroyAllWindows()
client.disconnect()
```

7. CODING & SOLUTIONING

7.1 Feature 1

1. IoT device
2. IBM Watson Platform
3. Node red
4. Cloudant DB
5. Web UI
6. MIT App Inventor
7. Python code

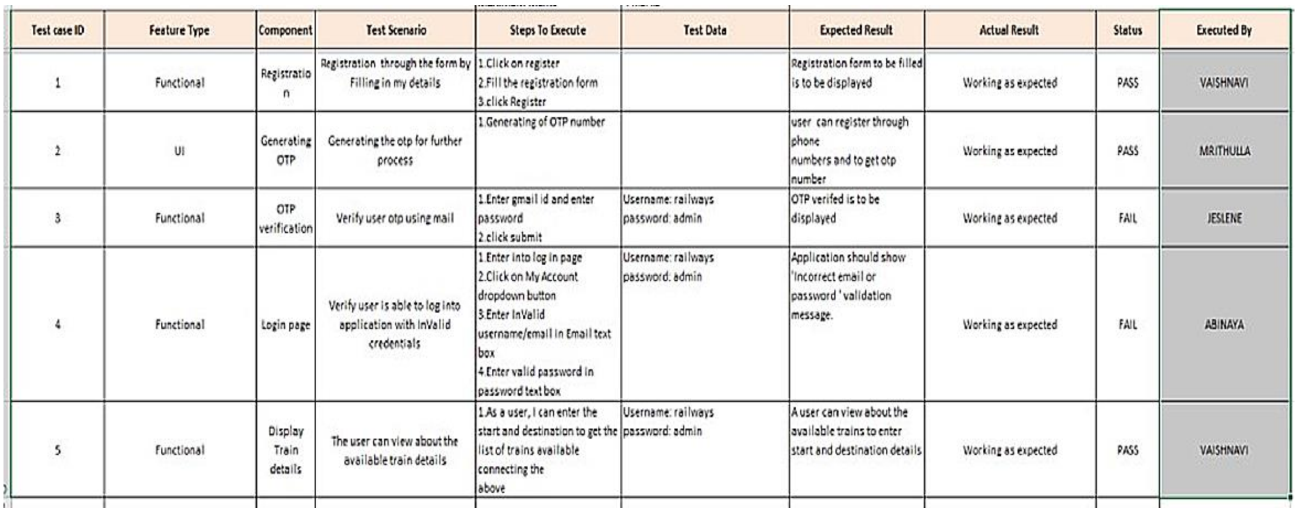
7.2 Feature 2

1. Login
2. Verification
3. Ticket Booking

4. Adding rating

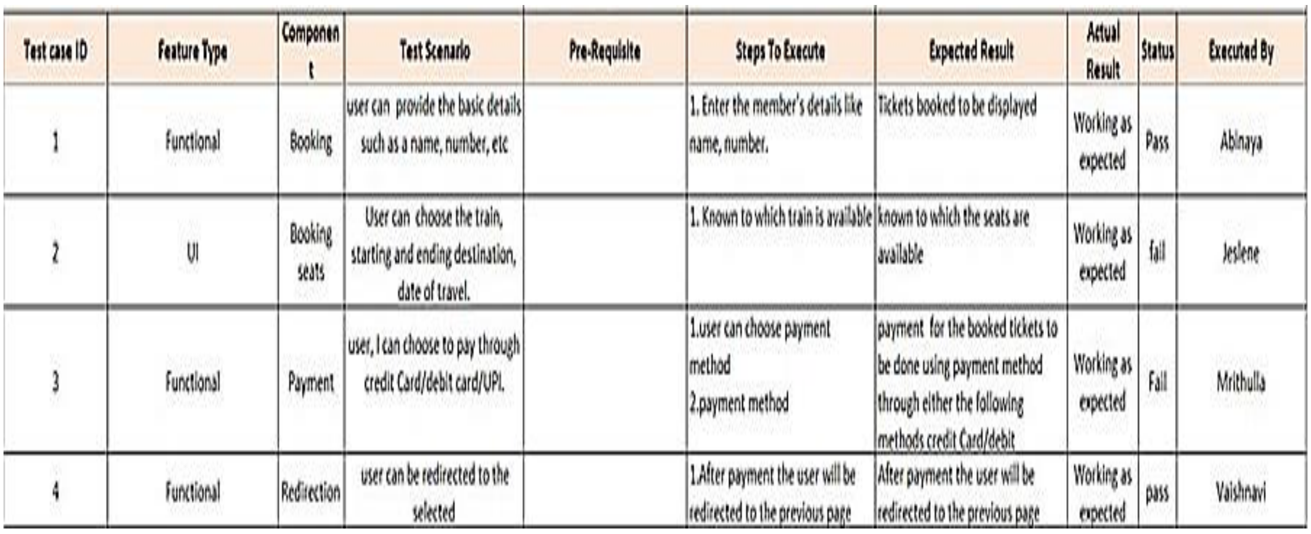
8. TESTING AND RESULTS

8.1 Test CasesTest Case 1



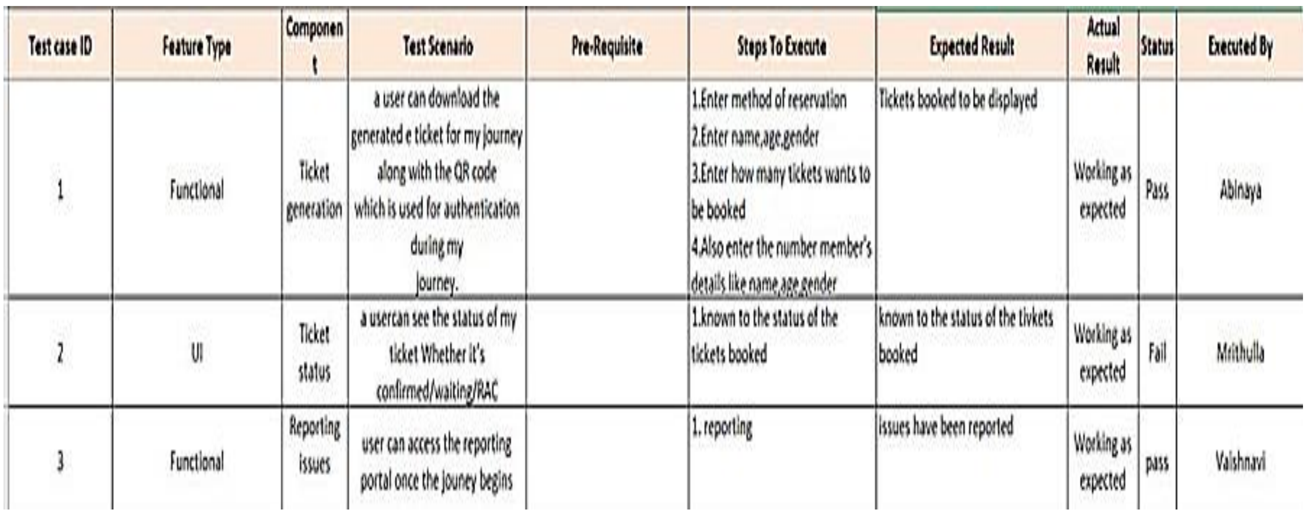
Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Executed By
1	Functional	Registration	Registration through the form by Filling in my details	1.Click on register 2.Fill the registration form 3.click Register		Registration form to be filled is to be displayed	Working as expected	PASS	VAISHNAVI
2	UI	Generating OTP	Generating the otp for further process	1.Generating of OTP number		user can register through phone numbers and to get otp number	Working as expected	PASS	MRITHULLA
3	Functional	OTP verification	Verify user otp using mail	1.Enter gmail id and enter password 2.click submit	Username: railways password: admin	OTP verifed is to be displayed	Working as expected	FAIL	JESLENE
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: railways password: admin	Application should show 'Incorrect email or password ' validation message.	Working as expected	FAIL	ABINAYA
5	Functional	Display Train details	The user can view about the available train details	1.As a user, I can enter the start and destination to get the list of trains available connecting the above	Username: railways password: admin	A user can view about the available trains to enter start and destination details	Working as expected	PASS	VAISHNAVI

Test Case 2



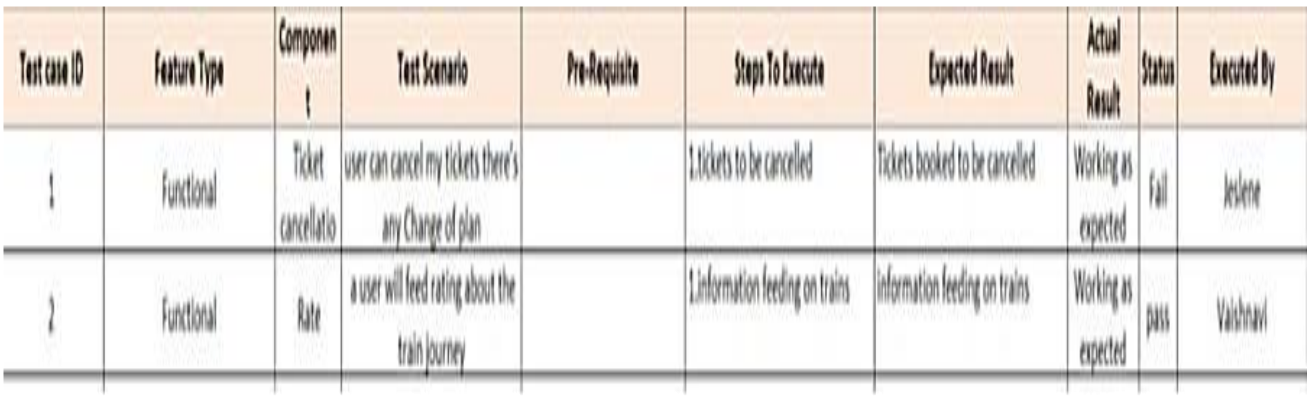
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Expected Result	Actual Result	Status	Executed By
1	Functional	Booking	user can provide the basic details such as a name, number, etc		1. Enter the member's details like name, number.	Tickets booked to be displayed	Working as expected	Pass	Abinaya
2	UI	Booking seats	User can choose the train, starting and ending destination, date of travel.		1. Known to which train is available	known to which the seats are available	Working as expected	fail	Jeslene
3	Functional	Payment	user, I can choose to pay through credit Card/debit card/UPI.		1.user can choose payment method 2.payment method	payment for the booked tickets to be done using payment method through either the following methods credit Card/debit	Working as expected	Fail	Mrithulla
4	Functional	Redirection	user can be redirected to the selected		1.After payment the user will be redirected to the previous page	After payment the user will be redirected to the previous page	Working as expected	pass	Vaishnavi

Test Case 3



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Expected Result	Actual Result	Status	Executed By
1	Functional	Ticket generation	a user can download the generated e ticket for my journey along with the QR code which is used for authentication during my journey.		1.Enter method of reservation 2.Enter name,age,gender 3.Enter how many tickets wants to be booked 4.Also enter the number member's details like name,age,gender	Tickets booked to be displayed	Working as expected	Pass	Abinaya
2	UI	Ticket status	a usercan see the status of my ticket Whether it's confirmed/waiting/RAC		1.known to the status of the tickets booked	known to the status of the tivkets booked	Working as expected	Fail	Mrithulla
3	Functional	Reporting issues	user can access the reporting portal once the jouney begins		1. reporting	issues have been reported	Working as expected	pass	Vaishnavi

Test Case 4



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Expected Result	Actual Result	Status	Executed By
1	Functional	Ticket cancellatio	user can cancel my tickets there's any Change of plan		1.tickets to be cancelled	Tickets booked to be cancelled	Working as expected	Fail	Jeslene
2	Functional	Rate	a user will feed rating about the train journey		1.information feeding on trains	information feeding on trains	Working as expected	pass	Vaishnavi

9. ADVANTAGES

1. The passengers can use this application, while they are travelling alone to ensure their safety.
2. It is easy to use.
3. It has minimized error rate.

10. DISADVANTAGES

Network issues may arise.

11. CONCLUSION

Almost all the countries across the globe strive to meet the demand for safe, fast, and reliable rail services. Lack of operational efficiency and reliability, safety, and security issues, besides 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. Often, they suffer from the lack in smart technologiesand latest technological updates to provide the most efficient passenger services.This is expected to induce rail executives to build rail systems that are smarter andmore 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. But the implementation of the latest technological updates in this system gradually turns inevitable due to increasing demand for providing the most efficient passenger services. Handling the passenger data efficiently backed by intelligent processing and timely retrieval would help backing up the security breaches. Here we've explored different issues of implementing smart computing in railway systems pertaining to reservation models besides pointing out some future scopesof advancement. Most significant improvements have been evidenced by more informative and user-friendly websites, mobile applications for real-time information about vehicles in motion, and e-ticket purchases and timetable information implemented at stations and stops. With the rise of Industry, railway companies can now ensure that they are prepared to avoid the surprise of equipment downtime. Like above mentioned, the developed application of our project can lead the passenger who travel can travel safely without any fear.

12. FUTURE SCOPE

This application is ensured for safety for the passengers while they are travelling alone as well as they travel with their family or friends.

In future, this application may also be used by passengers who travel through bus. By further enhancement of the application the passengers can explore more features regarding their safety.

13. APPENDIX

13.1 Source

Code 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
   INTEGERNOT 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()
```

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 import requests
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+&j ourney_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)
```

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
```

SEATS BOOKING

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

CONFIRMATION

# import module import requests from bs4 import BeautifulSoup importpandas as pd

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

return r.text

# input by geek

train_name = "03391-rajgir-new-delhi-clone-special-rgd-to-ndls"

# url

url = "https:/ www.railyatri.in/live-train-status/"+train_name

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

# traverse the live status from # this Html code data = [] for item in

```
soup.find_all('script', type="application/ld+json"):
  data.append(item.get_text())

  # convert into dataframe
  df = pd.read_json(data[2])
  # display this column of # dataframe
  print(df["mainEntity"][0]['name'])

  print(df["mainEntity"][0]['acceptedAnswer']['text'])
```

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
```

OTP GENERATION

```
import os import math import randomimport
smtplib

digits = "0123456789"
OTP = ""

for i in range (6):
   OTP += digits[math.floor(random.random()*10)]
otp = OTP + " is your OTP" message =
otps = 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")
```

OTP VERIFICATION

```
import os import math import randomimport
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")
```

13.2 GitHu

bGitHub

link:

https://github.com/IBM-EPBL/IBM-Project-38566-1660382479.git

Demo Video Link

https://drive.google.com/file/d/1oPyce_JFyGFfgpokoM6S8l5IKdYOQjda/view?usp=drivesdk