SMART SOLUTION FOR RAILWAYS

NALAIYA THIRAN PROJECT BASED LEARNING

on

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

A PROJECT REPORT

MANOJ KUMAR S	720719106070
MANISH VIGRAM S K	720719106069
KARTHICK RAJA K	720719106054
RAMPRASAD S	720719106093

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHOLOGY

Approved by AICTE, New Delhi, Accredited with 'A' Grade by NAAC

(An Autonomous Institution, Affiliated to Anna University, Chennai)

COIMBATORE – 641 032

November 2022

1.INTRODUCTION

1.1PROJECT OVERVIEW

As technology is growing fast, so we need to update ourselves to be in touch with new technology. The current process of metro ticket booking ticketing is very slow and tedious process. Customer needs to stand in long queue for issuing metro ticket at metro stations which is time consuming and this process is hectic to employees in the stations as well as passengers. Existing train ticket booking system has some drawbacks, like ticket is regenerated every time and is in paper printout format. This is a vapid process, which require to reprint the ticket every time. And existence system does not provide any security options. QR-code (abbreviated from Quick Response Code) is the trademark for a type of matrix barcode (or two dimensional barcode) first designed for the automotive industry in Japan. A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR-code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and kanji).The QR-code system became popular outside the automotive industry due to its fast readability and greater storage capacity compared to standard UPC barcodes. Applications include product tracking, item identification, time tracking, document management, and general marketing. A QR-code consists of black squares arranged in a square grid on a white background, which can be read by an imaging device such as a camera, and processed using Reed–Solomon error correction until the image can be appropriately interpreted. The required data is then extracted from patterns that are present in both horizontal and vertical components of the image. The Purpose of proposed system is to provide use of new technology in travel sector. To develop an android application that is cost efficient. To make an efficient use of QR-code technique. Provide solution without extra hardware requirement. To make system easy to handle. This system provides effective software for maintaining metro tickets. Digital metro ticket generating system is useful for peoples to get their metro ticket online, anytime and from anywhere instead of standing in long queues to get their tickets. This system reduces paperwork, time consumption and makes the process of issuing ticket in simpler and faster way. Passengers can book ticket very fast as within two or three click he / she can book metro ticket on app, just need to recharge their account of digital ticketing. No need to print the ticket every time. This system performs functionalities like accessing basic information of user authentication. The admin or the ticket checker would be able to verify the authenticity of the passenger's ticket by scanning QR-code which is provided on the recommended device like android mobile and after scanning it will notify to user when ticket is accessed. In our proposed system once the ticket number and time of buy is generated the details saved in the MySQL database are sent to Google Chart API engine in order to generate the QR-code. here all the personal and ticket information are converted into QR-code and sent back to the user mobile as HTTP response and saved in the application memory. This service checks the user's current location in accordance with the destination geo points, after which the ticket type is checked and accordingly the ticket is deleted if two is single or updated if type is return. In this module the checker will have QR-code reader and scan the QR-code.

1.2 PURPOSE

1. Reducing waiting time of passenger in que.

2. Develop an android application that is cost efficient.

3. Use of QR-code technique which provide better solution without extra hardware requirement.

4. Generating quick and easy to use android application.

5. Improving security of passenger’s ticket by providing

Registration and login to android application

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

- A QR-code [1] (it stands for “Quick Response”) is a mobile phone readable barcode that can store website URL‟s, plain text, phone number, Email addresses and pretty much any other alphanumeric data. The Quick Response (QR) code first used in automotive industry has now become popular due to its large storage capacity and extremely less response time here QR-code is used to store user information in encoded form. QR-code can be used in Android, Blackberry OS, Nokia Symbian as well as Apple iOS devices. The browser supports URL redirection which allows QR-code to send metadata to existing applications on the device

- In paper [2] Vrijendra Singh, Man Mohan Swarup, Abhiram Dwivedi, Rajendra Prasad, Chanchal Sonkar , Monark Bag, proposed a system in which the Dynamic Seat Allocation (DSA) system consider the International Journal of Scientific Research in Science and Technology (www.ijsrst.com) | Volume 7 | Issue 1 Prof. Ravindra Jogekar, et al Int J Sci Res Sci Technol. January-February-2020 ; 7 (1) : 103-107 105 advantage of QR-code processing along with one of the standards of wireless communication. Their approach is to make fair processing in seat reservation or allocation in Indian Railway

- In paper [3] Gayatri Shinde Sadaf Sheikh, Tazeen Shaikh, Mayuri Potghan, authors proposed an android application in which ticket can carry in the form of QR-code but it is difficult to passenger to understand the buying ticket is correct or not. Because most of the people are unaware of QR-code technology.

- In paper [4] Akshay Babar, Tushar Dongare introduced a model which provide various techniques for buying tickets through their mobile application through GPS facility of android mobile so that user can easily get the list of station and he can easily buy tickets, but Sometimes GPS signals are not accurate due to some obstacles to the signals

- As pointed out by Sadaf Shaikh et al. [7], this QRcode can be used to transfer between mobiles and can be shown to the ticket checker for validation. QRcodes are the 2D barcode that can store more than 4,000 alphanumeric characters in a limited horizontal and vertical space. A traditional linear (1D) barcode can hold roughly 20 horizontal characters. QR-codes are also easy to use and can be easily read from any direction with a simple Smartphone application or dedicated barcode scanner. ATVMs and CVM machines technologies are already installed in the Mumbai Suburban Railways. On October 2007 ATVM technology was introduced in the MSR in order to decrease long queues for tickets. The major drawback with existing ATVM system is the scalability issue. Only 3-4 tickets can be bought per minute through ATVM. Another issue with the system is the cost of installing the machine. Each machine costs around 17500 INR excluding the maintenance costs which vary according to the usage intensity

2.2 REFERNCES

[1] Vinay Maheshwar, Kalpesh Patil, Azim Maredia, Apeksha Waghmare "Android Application on ETicketing Railway System Using Qr-Code", IOSRJEN, ISSN (e): 2250-3021, ISSN (p): 2278- 8719 Volume 13, PP 33-38

[2] Man Mohan Swarup, Abhiram Dwivedi, Chanchal Sonkar, Rajendra Prasad, Monark Bag, Vrijendra Singh, ―A QR-code Based Processing For Dynamic and Transparent Seat Allocation in

Indian Railway‖, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1, May 2012.

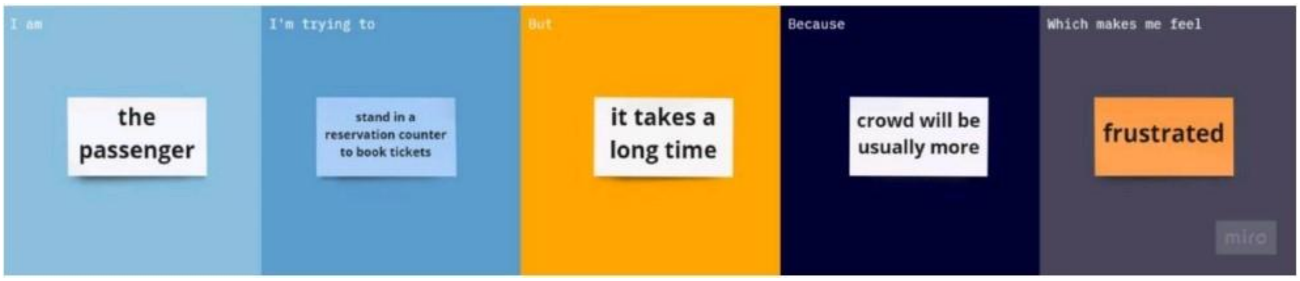
[3] Sadaf Sheikh, Gayatri Shinde, Mayuri Potghan, Tazeen Shaikh, ―Urban railway ticketing International Journal of Scientific Research in Science and Technology (www.ijsrst.com) | Volume 7 | Issue 1 Prof. Ravindra Jogekar, et al Int J Sci Res Sci Technol. January-February-2020 ; 7 (1) : 103-107 107 application‖, International Journal Of Advance Research In Computer Science And Software Engineering Vol. 4, Issue 1.

[4] Tushar Dongare, Akshay Babar, Et Al., Android Application For Ticket Reservation With GPS As Ticket Validation International Journal Of Emerging Research In Management And Technology ISSN: 2278-9359, Vol-3, Issue-3, March 2014.

[7] Sadaf Shaikh, Gayatri Shinde, Mayuri Potghan, Tazzen Shaikh, Ranjeetsingh Suryawanshi "Urban Railway Ticketing Appion", International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), pp. 130-132, January-2014.

2.3 PROBLEM STATEMENT DEFINITION

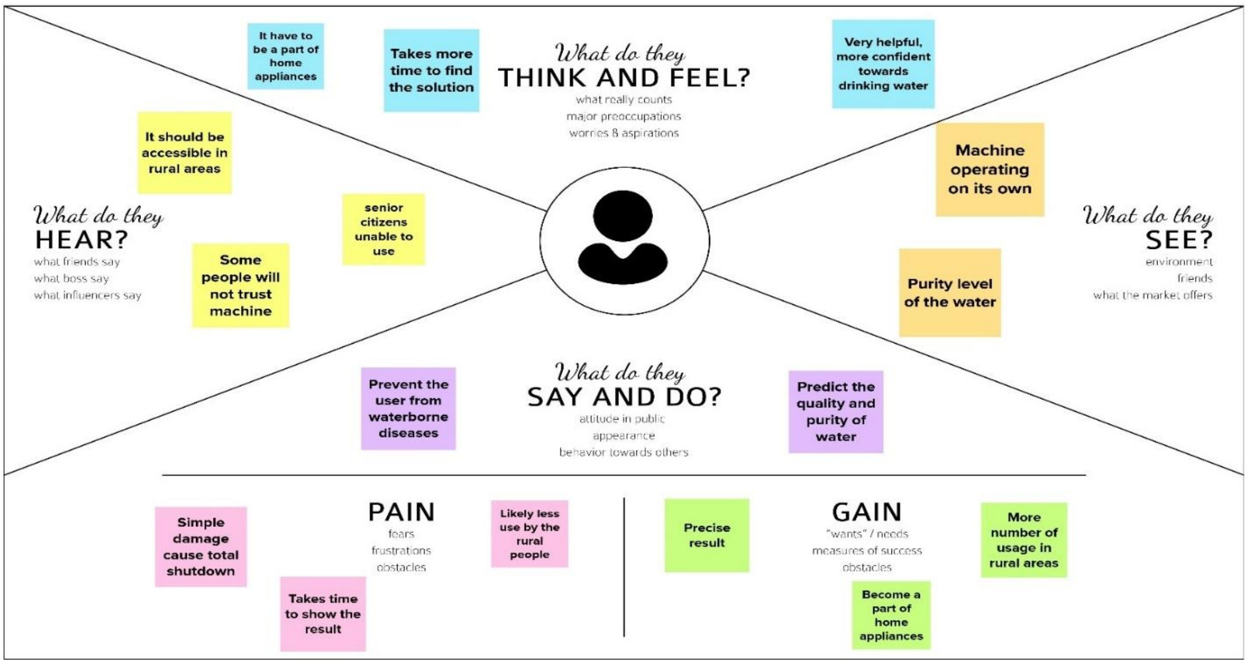
Date	19 September 2022
Team ID	PNT2022TMID10155
Project Name	SMART SOLUTION FOR RAILWAYS
Maximum Marks	2 Marks



Problem Statement	Problem Statement Passengers waiting in a long queue to book ticket is totally a waste of time in todays world. Passengers wants to save their time and physical activity. In order to solve this problem, we need a solution.
I am	The passenger
I'm trying to	Stand in reservation counter to book tickets
But	It takes a long time
Because	Crowd will be usually more
Which makes me feel	frustrated

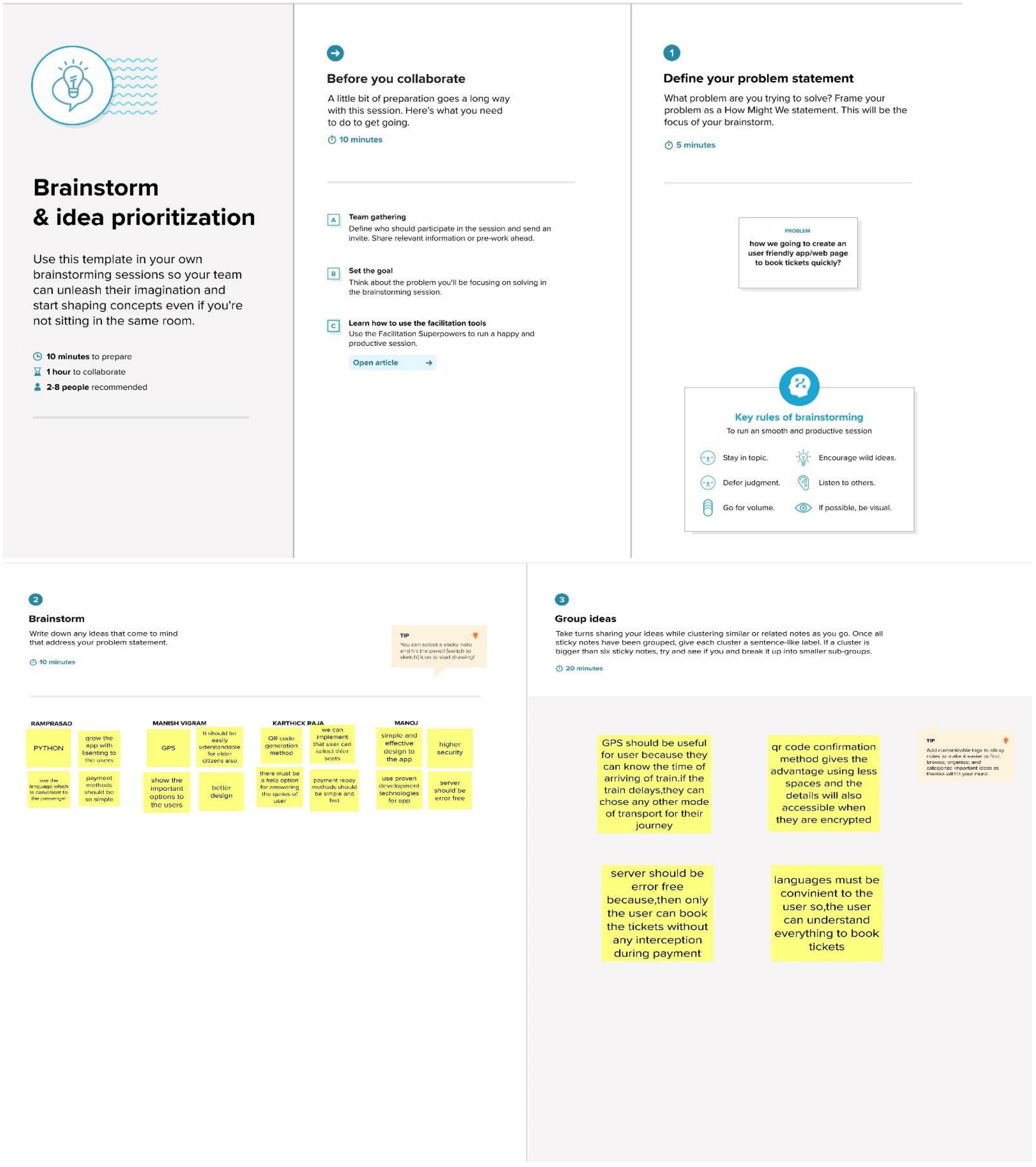
3.IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



It have to be a part of home appliances
Takes more time to find the solution
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Very helpful, more confident towards drinking water
It should be accessible in rural areas
Machine operating on its own
What do they HEAR?
what friends say
what boss say
what influencers say
senior citizens unable to use
Some people will not trust machine
What do they SEE?
environment
friends
what the market offers
Purity level of the water
Prevent the user from waterborne diseases
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Predict the quality and purity of water
Simple damage cause total shutdown
PAIN
fears
frustrations
obstacles
Likely less use by the rural people
Takes time to show the result
Precise result
GAIN
"wants" / needs
measures of success
obstacles
More number of usage in rural areas
Become a part of home appliances

3.2 IDEATION AND BRAINSTROMING



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 we going to create an user friendly app/web page to book tickets quickly?

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.

2 **Brainstorm**

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

10 minutes

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

RAMPRASAD
- PYTHON
- grow the app with lisenting to the users
- use the language which is convinient to the passenger
- payment methods should be so simple

MANISH VIGRAM
- GPS
- It should be easily uderstandable for older citizens also
- show the important options to the users
- better design

KARTHICK RAJA
- QR code generation method
- we can implement that user can select thier seats
- there must be a help option for answering the quries of user
- payment repay methods should be simple and fast

MANOJ
- simple and effective design to the app
- higher security
- use proven development technologies for app
- server should be error free

3 **Group ideas**

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

20 minutes

- GPS should be useful for user because they can know the time of arriving of train.if the train delays,they can chose any other mode of transport for their journey
- qr code confirmation method gives the advantage using less spaces and the details will also accessible when they are encrypted
- server should be error free because,then only the user can book the tickets without any interception during payment
- languages must be convinient to the user so,the user can understand everything to book tickets

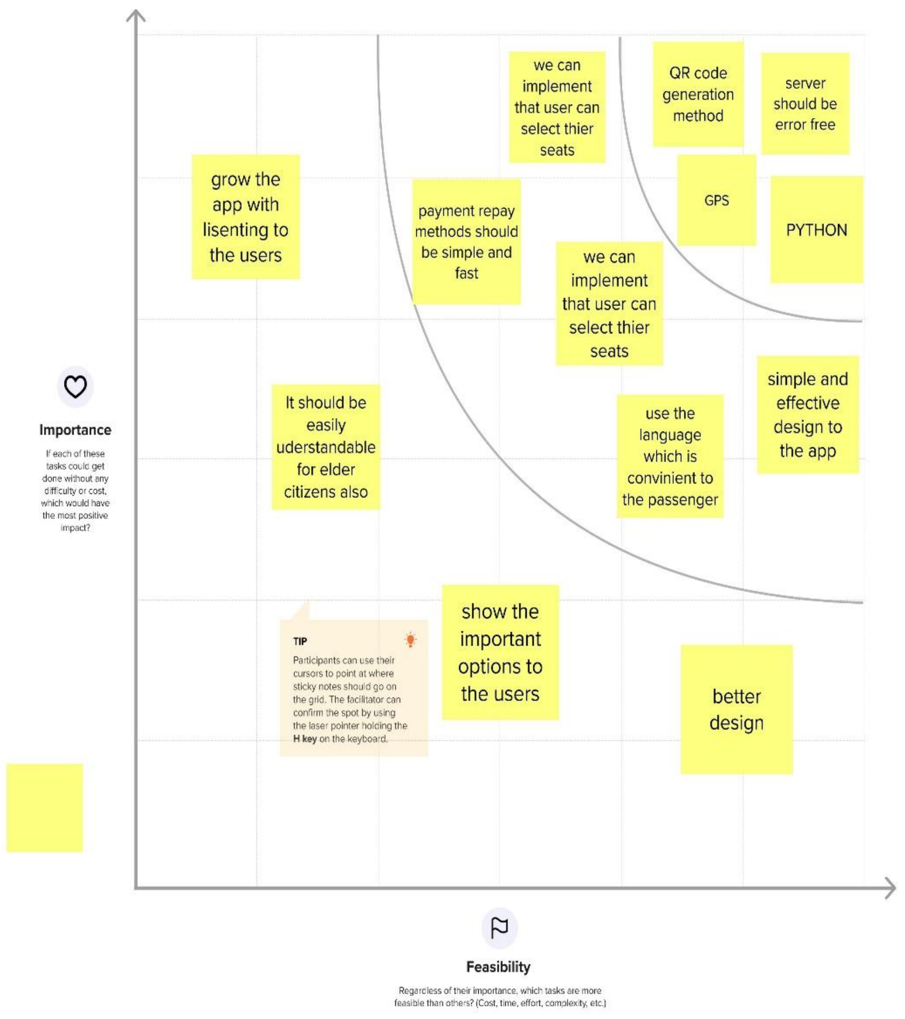
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4

Prioritize

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

20 minutes



we can implement that user can select thier seats
QR code generation method
server should be error free
grow the app with lisenting to the users
payment repay methods should be simple and fast
GPS
PYTHON
we can implement that user can select thier seats
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
It should be easily uderstandable for elder citizens also
simple and effective design to the app
use the language which is convinient to the passenger
show the important options to the users
TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.
better design
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback

3.3 PROPOSED SOLUTION

Date	30 September 2022
Team ID	PNT2022TMID10155
Project Name	Smart Solution for Railways
Maximum Marks	2 Marks

Proposed Solution:

S.NO	Parameter	Description
1.	Problem statement (problem to be solved)	Passengers waiting in a long queue to book ticket is totally a waste of time in todays world. Passengers wants to save their time and physical activity. We want to reduce the paper use for tickets
2.	Idea/solution description	It is not possible for passenger to book tickets in station everytime. So an app or web page to book tickets in online with android phone or pc. It will generate a QR code for booking tickets.
3.	Novelty/Uniqueness	• User friendly • Reduce the usage of paper • Save time for people • Database storage with QR verification
4.	Social impact/Customer satisfaction	customer satisfaction is an important factor to consider. In order to achieve this goal, it is important that this app is user friendly and easily understandable for all people in both rural and urban areas.
5.	Business Model (Revenue Model)	Reduction of paper usage, simple and understandable process and unique database storage with QR verification.
6.	Scalability of the solution	It can be used for all modes transport including the public and private sectors

3.4 PROBLEM SOLUTION FIT

TEAM ID: PNT2022TMID10155

1.CUSTOMER SEGMENTS Customers are: • Functional Traveller • passenger	6.**CUSTOMER LIMITATION**: Customer can access only to book a seat through application using mobile phones or pc from anywhere.	5.**AVAILABLE SOLUTION**: 1. Onlinewebapplication tobookthetraintickets QRcodeisgenerated onceticketisbooked. 2. In web application we canbeabletotrackthe live location and arriving time of the train
2.PROBLEMS/PAINS: 1.passengers wasting a lot of time by booking tickets in counter 2.TTE has to process huge paperwork tO verify passenger tickets	**9.PROBLEM** ROOT/CAUSE: 1.To spend longtime to book a ticket in station 2.Passengersarenot properly verified before enteringinto train	7.**BEHAVIOR**: Detects themotion ofthe passenger andtallythatcount with the Number of tickets booked
3.TRIGGERS TO ACT: Railway passengers see their neighbours easily booking tickets without having to wait in long lines 4.EMOTIONS (BEFORE/AFTER): Previously, passenger sees the ticket booking as time- consuming. After using an online method, passenger feeling it as timeconvenience.	**10.YOUR SOLUTION:** 1.Passenger can book ticketin online 2.They have the unique generated QR code, by that they can verify thatQRcode ticket collector	**8.CHANNEL OF BEHAVIOR:** ONLINE: Theycanabletotrackthe location oftrain. OFFLINE: ScantheQRcodeinthe ticket to verify the information. Based on the passenger counts the automated doors are opened.

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

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 Web page
FR-2	User Confirmation	Confirmation via Message Confirmation via OTP
FR-3	IBM cloud	All details of users are available in cloud
FR-4	Web application	Contains details like boarding station, destination, seat options, name, age, mobile number etc
FR-5	QR code generator	QR code contains the registration details of the user that will be sent via message

4.2 NON-FUNCTIONAL REQUIREMENT

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

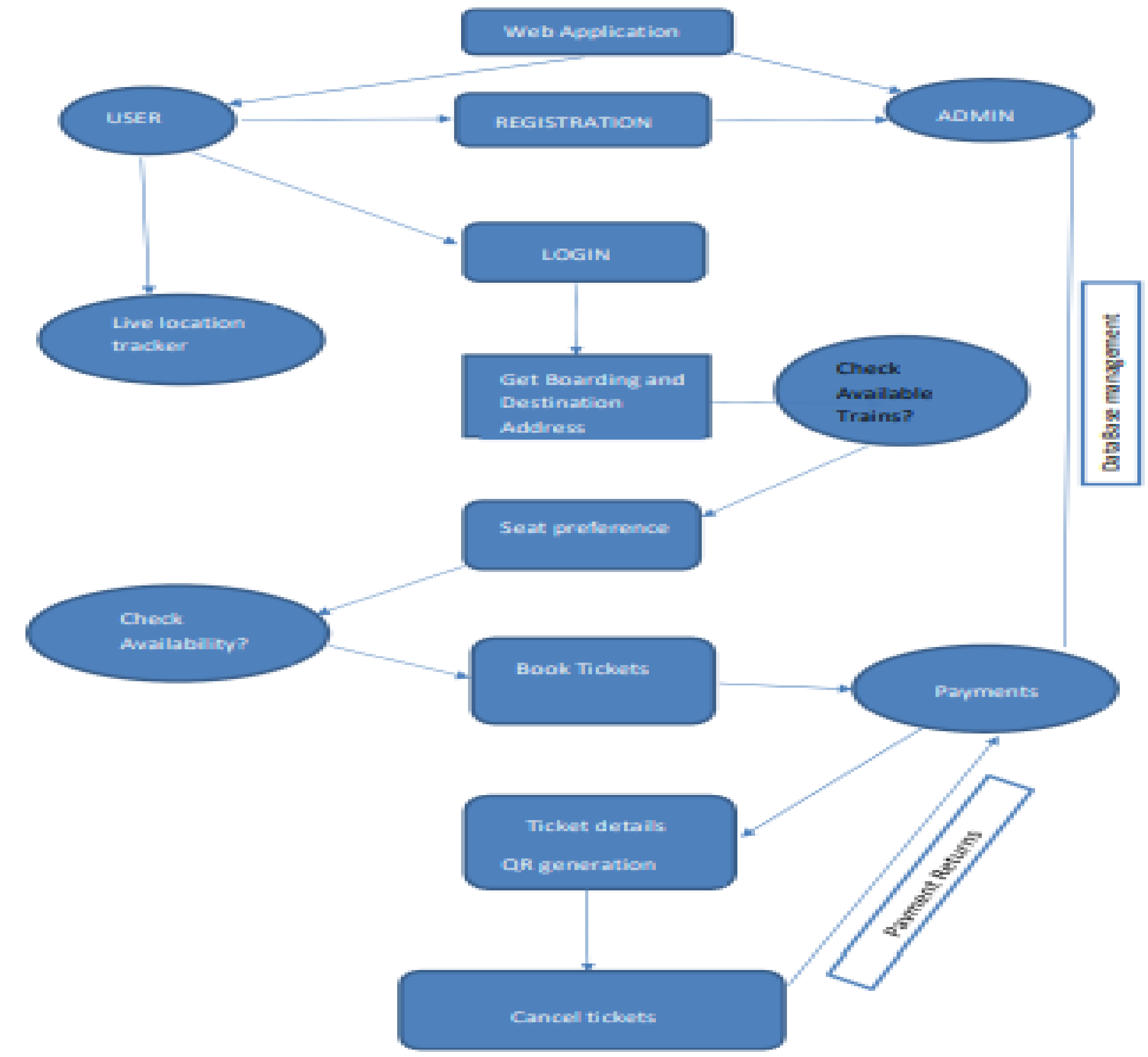
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User can use the Web application to book tickets and can check the status of the trains.
NFR-2	**Security**	The user details are stored in the cloud with high encryption and these details are non-shareable.
NFR-3	**Reliability**	The confirmation of the tickets will be sent immediately to the user after the confirmation of payment. The payment details of the users are also well secured and confirm payment through OTP.

NFR-4	**Performance**	Everything will be done quickly. Since it is online booking, Users can book their tickets comfortably in their places without going to railway station
NFR-5	**Availability**	The Web application will be available for all the users to book tickets and all the boarding details of the users are also available in the web application. Users can also check the availability of the trains through the Web application.

NFR6	**Scalability**	This idea can also be upgraded by adding some additional features in the future.

5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAM



5.2 SOLUTION & TECHNICAL ARCHITECTURE

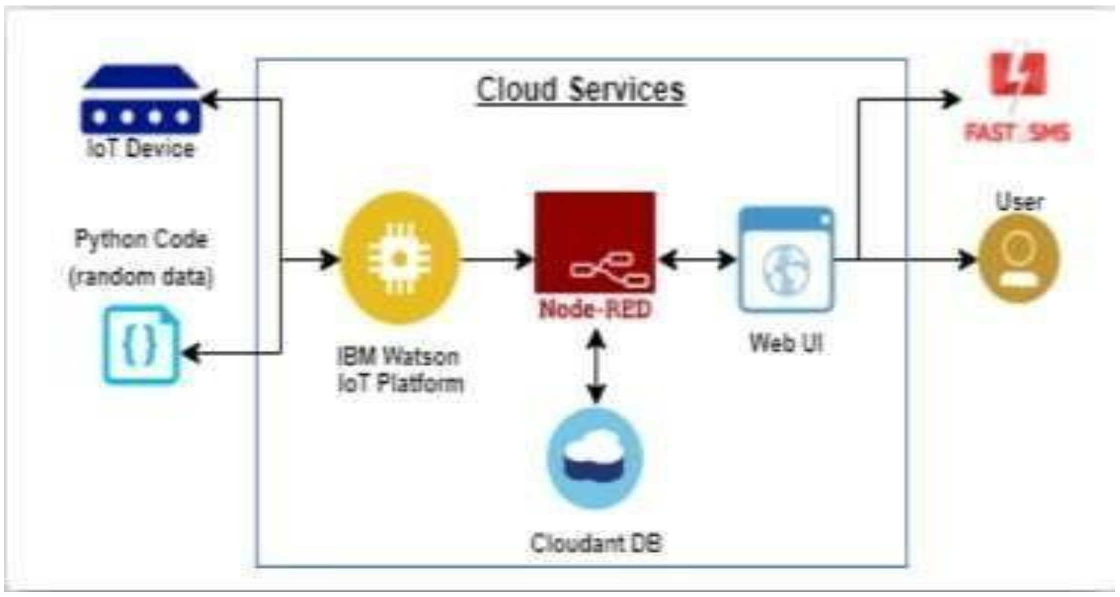


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	IoT Device	How user interacts with application	Cloud technology
2.	Python code	python code for publishing the location (latitude and longitude) data to the IBM IoT Platform and the other python code to read the QR Code and fetch the data from Cloudant DB.	Python
3.	IBM Watson IoT Platform	IBM Watson IoT platform acts as the mediator to connect the web application to IoT device, so create the IBM Watson IoT platform.	Analytics and information retrival
4.	Node Red	Connect to IBM IoT platform and get the location, store the data in Cloudant DB.	Java script , cloud technology
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	Web UI	we get the expected output by providing the desired user input where the QR Code is generated and the same data is stored in the form of json in Cloudant DB.	python
8.	Fast SMS	To confirm ticket booking	python
9.	user	Take user input (Basic Information) for booking a seat on the train	Python

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Usability	User can use the Web application to book tickets and can check the status of the trains.	QR code
2.	Security	The user details are stored in the cloud with high encryption and these details are non-shareable.	Encryptions, QR code
3.	Reliability	The confirmation of the tickets will be sent immediately to the user after the confirmation of payment. The payment details of the users are also well secured and confirm payment through OTP.	IBM Watson IoT Platform, Node Red
4.	Availability	The Web application will be available for all the users to book tickets and all the boarding details of the users are also available in the web application. Users can also check the availability of the trains through the Web application.	IBM Watson IoT Platform, Node Red
5.	Performance	Everything will be done quickly. Since it is online booking, Users can book their tickets comfortably in their places without going to railway station	IBM Watson IoT Platform, Node Red
6.	Scalability	This idea can also be upgraded by adding some additional features in the future.	IBM Watson IoT Platform, Node Red

5.3 USER STORIES

User Stories

User Type	Functional Requirement (epic)	User Story number	User Story / Task	Acceptance criteria	Priority	Release
CUSTOMER (MOBILE USER)	Reserving ticket	USN-1	As a user, I canregister for the application by entering my email, password,and confirming my password.	I can access my account / dashboard	High	Sprint-1
CUSTOMER (MOBILE USER	Reserving ticket	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 register & access the dashboard with Facebook Login	Low	Sprint-2
CUSTOMER (MOBILE USER	Dashboard	Users	The details will be stored safely	I can accessit using database	Medium	Sprint-3
CUSTOMER (WEB USER)	Reserving ticket	Users	Enter the details and click submit button to book ticket	I can use the QR code which is been generated	High	Sprint-1
CustomerCare Executive	Connectingthe service provider	Customer	Connects with theservice 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 AND ESTIMATION

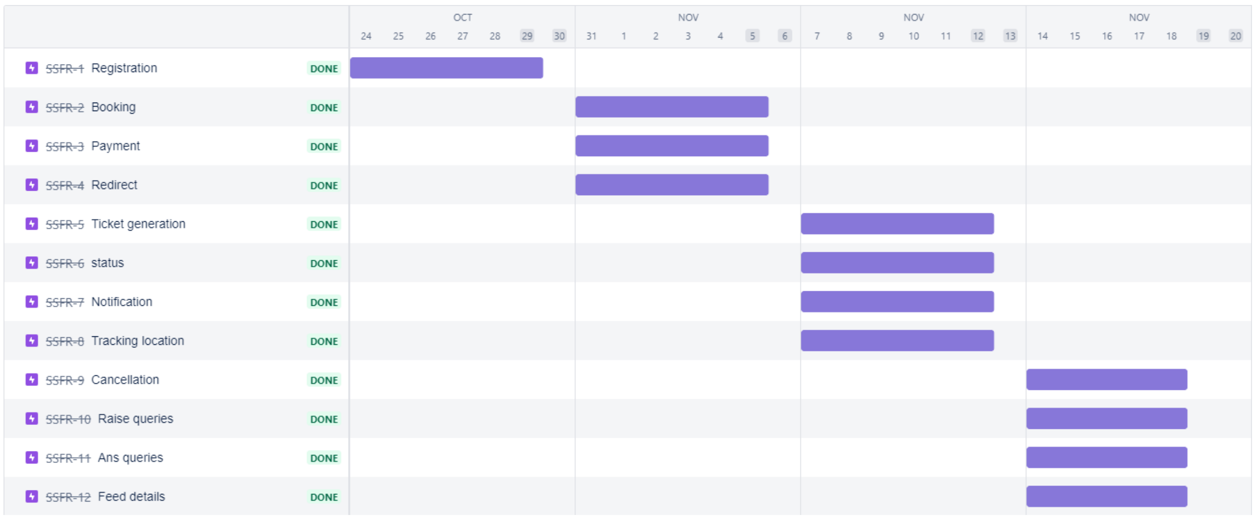
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register through the form by Filling in my details	2	High	Manoj kumar
Sprint-1		USN-2	As a user, I can register through phone numbers, Gmail, Facebook or other social sites	1	High	Ramprasad
Sprint-1	Conformation	USN-3	As a user, I will receive confirmation through email or OTP once registration is successful	2	Low	Karthick raja
Sprint-1	login	USN-4	As a user, I can login via login id and password or through OTP received on register phonenumber	2	Medium	Manish vigram
Sprint-1	Display Train details	USN-5	As a user, I can enter the start and destination to get the list of trains available connecting the above	1	High	Manoj kumar
Sprint-2	Booking	USN-6	As a use, I can provide the basic details such as a name, age, gender etc…	2	High	Ramprasad
Sprint-2	Booking seats	USN-7	As a user, I can choose the class, seat/berth. If apreferred seat/berth isn't available I can be allocated based on the availability	1	Low	Karthick raja
Sprint-2	Payment	USN-8	As a user, I can choose to pay through credit Card/debit card/UPI.	1	High	Manish vigram
Sprint	Functional Requiremen t (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Member s
Sprint-2	Redirection	USN-9	As a user, I will be redirected to the selected	2	High	Manoj kumar
Sprint-3	Ticket generation	USN-10	As a user, I can download the generated e- ticket for my journey along with the QR code which is used for authentication during my journey.	1	High	Ramprasad
Sprint-3	Ticket status	USN-11	Whether it's confirmed/waiting/RAC.	2	High	Karthick raja

Sprint-3	Remainders notification	USN-12	As a user, I get remainders about my journey A day before my actual journey.	1	High	Manish vigram
Sprint-3	GPS tracking	USN-13	As a user, I can track the train using GPS and can get information such as ETA, Current stop and delay	2	High	Manoj kumar
Sprint-4	Ticket cancellation	USN-14	As a user, I can cancel my tickets if there's any Change of plan	1	High	Ramprasad
Sprint-4	Raise queries	USN-15	As a user, I can raise queries through the query box or via mail.	2	Medium	Karthick raja
Sprint-4	Answer the queries	USN-16	As a user, I will answer the questions/doubts Raised by the customers.	2	High	Manish vigram
Sprint-4	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.	1	High	Manoj kumar

6.2 SPRINT DELIVERY SCHEDULE

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date(Actual)**
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	5 Nov 2022
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov2022

6.3 REPORTS FROM JIRA



~~SSFR-1~~ Registration DONE
~~SSFR-2~~ Booking DONE
~~SSFR-3~~ Payment DONE
~~SSFR-4~~ Redirect DONE
~~SSFR-5~~ Ticket generation DONE
~~SSFR-6~~ status DONE
~~SSFR-7~~ Notification DONE
~~SSFR-8~~ Tracking location DONE
~~SSFR-9~~ Cancellation DONE
~~SSFR-10~~ Raise queries DONE
~~SSFR-11~~ Ans queries DONE
~~SSFR-12~~ Feed details DONE

7.CODING AND SOLUTIONING

7.1 FEATURE 1

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

7.2 FEATURE 2

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

```
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()
```

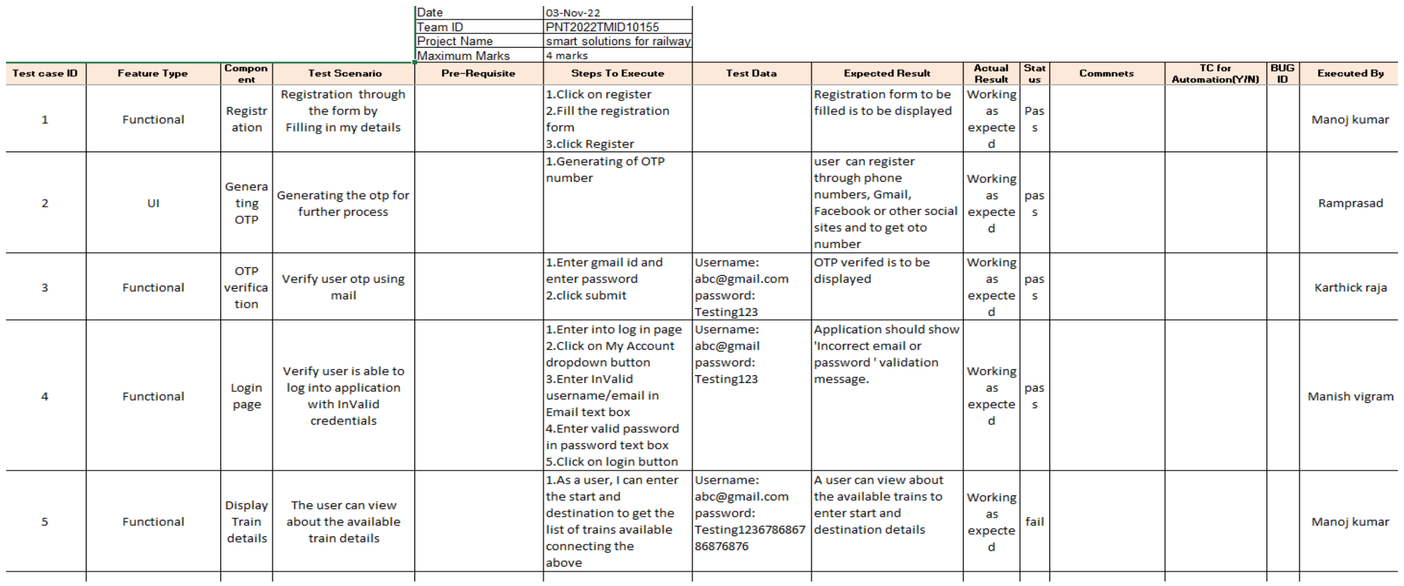
```
def generateOTP() :

    # Declare a digits variable #
        which stores all digits
        digits = "0123456789"
        OTP = ""

   # length of password can be changed # by
       changing value in range
    for i in range(4) :
            OTP += digits[math.floor(random.random() * 10)]

        return OTP

# Driver code
if_name_______== "__main__" :

        print("OTP of 4 digits:", generateOTP())

    digits="0123456789"
OTP=""
    for i in range(6):
        OTP+=digits[math.floor(random.random()*10)]
otp = OTP + " is your OTP"
    msg= otp
s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
s.login("Your Gmail Account", "You app password")
emailid = input("Enter your email: ")
```

```
s.sendmail('&&&&&&&&&&&&',emailid,msg)   a = input("Enter Your OTP >>: ")
if a == OTP: print("Verified")
else:
print("Please Check your OTP again")
roo
```

8. TESTING

8.1 TEST CASES

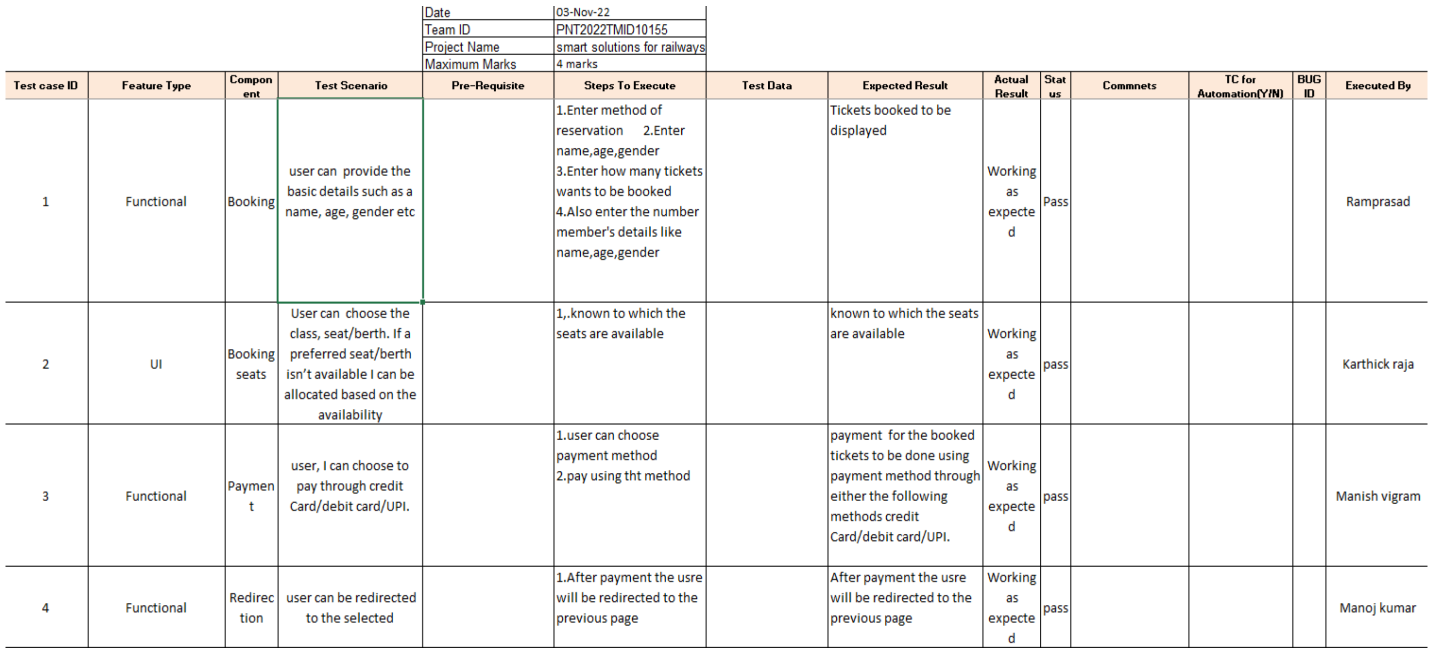
SPRINT-1



Date	03-Nov-22
Team ID	PNT2022TMID10155
Project Name	smart solutions for railway
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
1	Functional	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				Manoj kumar
2	UI	Generating OTP	Generating the otp for further process		1.Generating of OTP number		user can register through phone numbers, Gmail, Facebook or other social sites and to get oto number	Working as expected	pass				Ramprasad
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				Karthick raja
4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter into log in page 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: abc@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	pass				Manish vigram
5	Functional	Display Train details	The user can view about the available train details		1.As a user, I can enter the start and destination to get the list of trains available connecting the above	Username: abc@gmail.com password: Testing123678686786876876	A user can view about the available trains to enter start and destination details	Working as expected	fail				Manoj kumar

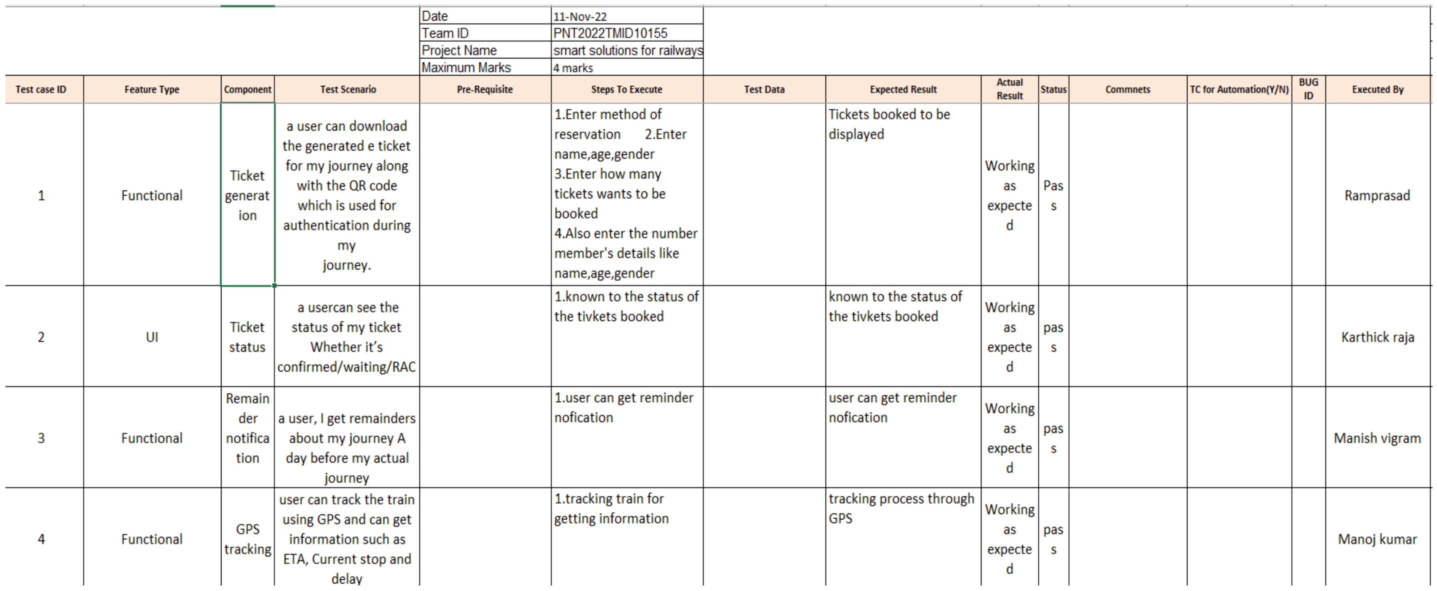
SPRINT-2



Date	03-Nov-22
Team ID	PNT2022TMID10155
Project Name	smart solutions for railways
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
1	Functional	Booking	user can provide the basic details such as a name, age, gender etc		1.Enter method of reservation 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				Ramprasad
2	UI	Booking seats	User can choose the class, seat/berth. If a preferred seat/berth isn't available I can be allocated based on the availability		1,.known to which the seats are available		known to which the seats are available	Working as expected	pass				Karthick raja
3	Functional	Payment	user, I can choose to pay through credit Card/debit card/UPI.		1.user can choose payment method 2.pay using tht method		payment for the booked tickets to be done using payment method through either the following methods credit Card/debit card/UPI.	Working as expected	pass				Manish vigram
4	Functional	Redirection	user can be redirected to the selected		1.After payment the usre will be redirected to the previous page		After payment the usre will be redirected to the previous page	Working as expected	pass				Manoj kumar

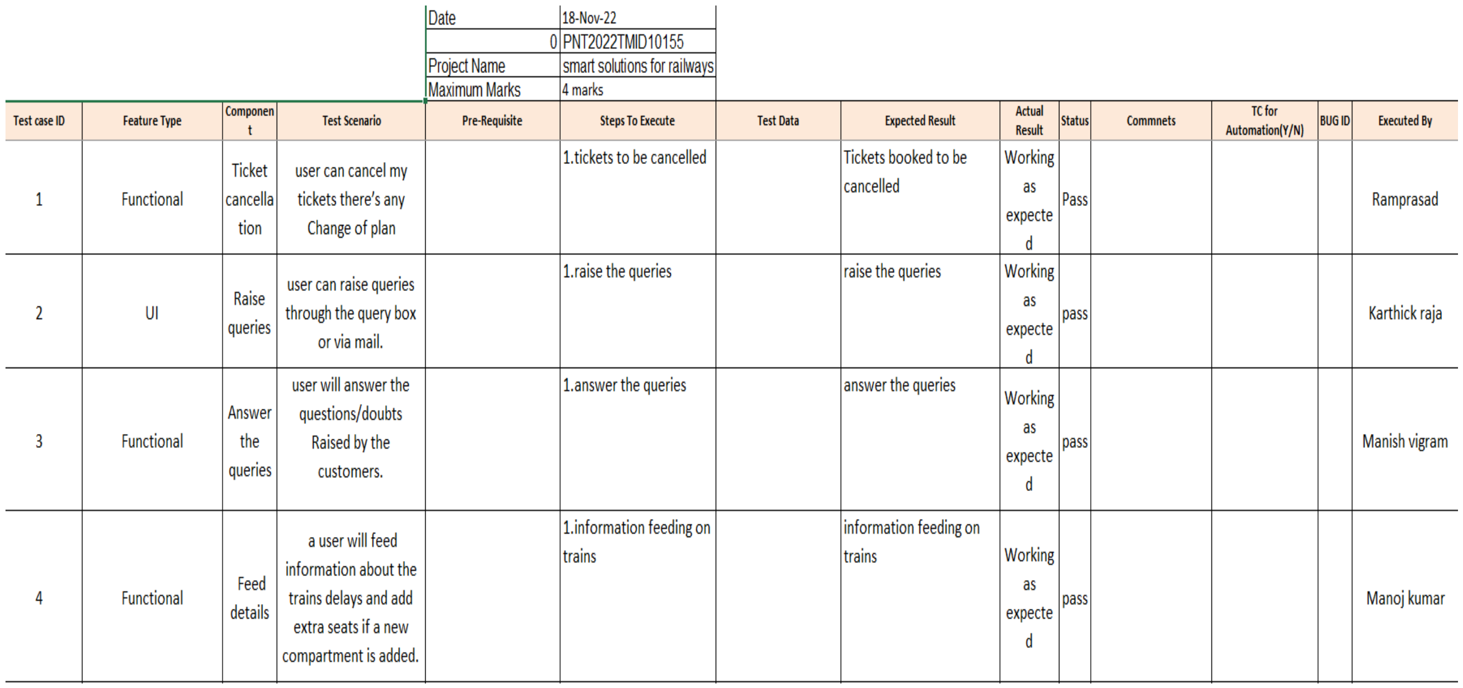
SPRINT-3



Date	11-Nov-22
Team ID	PNT2022TMID10155
Project Name	smart solutions for railways
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
1	Functional	Ticket 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				Ramprasad
2	UI	Ticket status	a usercan see the status of my ticket Whether it's confirmed/waiting/RAC		1.known to the status of the tivkets booked		known to the status of the tivkets booked	Working as expected	pass				Karthick raja
3	Functional	Remainder notification	a user, I get remainders about my journey A day before my actual journey		1.user can get reminder nofication		user can get reminder nofication	Working as expected	pass				Manish vigram
4	Functional	GPS tracking	user can track the train using GPS and can get information such as ETA, Current stop and delay		1.tracking train for getting information		tracking process through GPS	Working as expected	pass				Manoj kumar

SPRINT-4

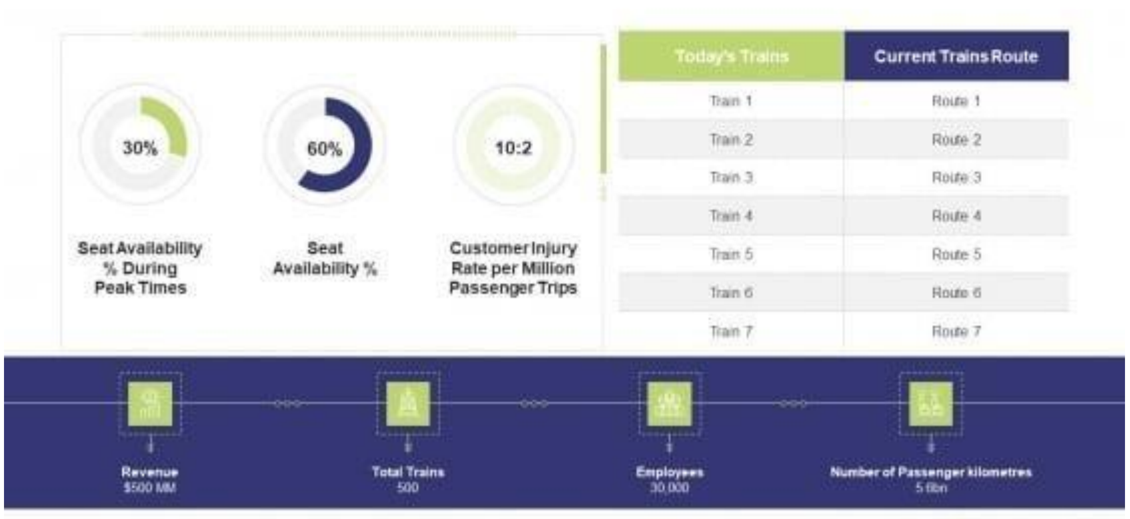


Date	18-Nov-22
0	PNT2022TMID10155
Project Name	smart solutions for railways
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
1	Functional	Ticket cancellation	user can cancel my tickets there's any Change of plan		1.tickets to be cancelled		Tickets booked to be cancelled	Working as expected	Pass				Ramprasad
2	UI	Raise queries	user can raise queries through the query box or via mail.		1.raise the queries		raise the queries	Working as expected	pass				Karthick raja
3	Functional	Answer the queries	user will answer the questions/doubts Raised by the customers.		1.answer the queries		answer the queries	Working as expected	pass				Manish vigram
4	Functional	Feed details	a user will feed information about the trains delays and add extra seats if a new compartment is added.		1.information feeding on trains		information feeding on trains	Working as expected	pass				Manoj kumar

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

13.APPENDIX

13.1SOURCE PROGRAM

```
    import math, random import os
 import smtplib import
      sqlite3 import requests
 from bs4 import BeautifulSoup
 from django.contrib.auth.base_user import AbstractBaseUser from django.db import models
 import logging import pandas as pd
      import pyttsx3
from plyer import notification import time
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw from pickle import
     load,dump import smtplib, ssl
from email.mime.text import MIMEText
     from email.mime.multipart import MIMEMultipart import email

from email import encoders
from email.mime.base import MIMEBase


  import attr
  from flask import Blueprint, flash, redirect, request, url_for from flask.views import MethodView
  from flask_babelplus import gettext as _
  from flask_login import current_user, login_required
from pluggy import HookimplMarker

   from tkinter import* 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()


def generateOTP() :
    # Declare a digits variable # which stores all
        digits digits = "0123456789" OTP = ""


   # length of password can be changed # by changing value in
       range
    for i in range(4) :
            OTP += digits[math.floor(random.random() * 10)] return OTP

# Driver code
if__name_== "______main____" :


        print("OTP of 4 digits:", generateOTP()) digits="0123456789"
OTP=""
    for i in range(6): OTP+=digits[math.floor(random.random()*10)]
otp = OTP + " is your OTP" msg= otp
s = smtplib.SMTP('smtp.gmail.com', 587) s.starttls()
s.login("Your Gmail Account", "You app password") emailid = input("Enter your email: ")
    s.sendmail('&&&&&&&&&&&',emailid,msg)
a = input("Enter Your OTP >>: ")
```

```
    if a == OTP: print("Verified")
else:
        print("Please Check your OTP again") root = Tk()
root.title("Python: Simple Login Application") width = 400
height = 280
screen_width = root.winfo_screenwidth() screen_height =
    root.winfo_screenheight() x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2) root.geometry("%dx%d+%d+%d" % (width, height, x, y))
    root.resizable(0, 0)
USERNAME = StringVar() PASSWORD =
    StringVar()
Top = Frame(root, bd=2, relief=RIDGE) Top.pack(side=TOP, fill=X)
Form = Frame(root, height=200) Form.pack(side=TOP, pady=20)
lbl_title = Label(Top, text = "Python: Simple Login Application", font=('arial', 15))
lbl_title.pack(fill=X)
lbl_username = Label(Form, text = "Username:", font=('arial', 14), bd=15)
lbl_username.grid(row=0, sticky="e")
lbl_password = Label(Form, text = "Password:", font=('arial', 14), bd=15)
lbl_password.grid(row=1, sticky="e") lbl_text = Label(Form)
```

```
lbl_text.grid(row=2, columnspan=2)
username = Entry(Form, textvariable=USERNAME, font=(14)) username.grid(row=0, column=1)
password = Entry(Form, textvariable=PASSWORD, show="*", font=(14))
password.grid(row=1, column=1) def Database():
    global conn, cursor
    conn = sqlite3.connect("pythontut.db") cursor = conn.cursor()
         cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER NOT NULL
     PRIMARY KEY
     AUTOINCREMENT, username TEXT, password TEXT)") cursor.execute("SELECT * FROM `member` WHERE
         `username` =
'admin' AND `password` = 'admin'")
    if cursor.fetchone() is None:
             cursor.execute("INSERT INTO `member` (username, password) VALUES('admin', 'admin')")
        conn.commit()
     def Login(event=None): Database()
    if USERNAME.get() == "" or PASSWORD.get() == "":
             lbl_text.config(text="Please complete the required field!", fg="red") else:
        cursor.execute("SELECT * FROM `member` WHERE `username`
     = ? AND `password` = ?", (USERNAME.get(), PASSWORD.get())) if cursor.fetchone() is not None:
                 HomeWindow()
                 USERNAME.set("")
             PASSWORD.set("")
             lbl_text.config(text="")
```

```
            else:
                lbl_text.config(text="Invalid username or password", fg="red") USERNAME.set("")
                PASSWORD.set("")
        cursor.close()
            conn.close()
btn_login = Button(Form, text="Login", width=45, command=Login) btn_login.grid(pady=25, row=3,
    columnspan=2) btn_login.bind('<Return>', Login)



    def HomeWindow(): global
        Home root.withdraw()
        Home = Toplevel()
        Home.title("Python: Simple Login Application") width = 600
    height = 500
    screen_width = root.winfo_screenwidth() screen_height =
        root.winfo_screenheight() x = (screen_width/2) - (width/2)
        y = (screen_height/2) - (height/2) root.resizable(0,
        0)
        Home.geometry("%dx%d+%d+%d" % (width, height, x, y)) lbl_home = Label(Home, text="Successfully
        Login!", font=('times new
roman', 20)).pack()
        btn_back = Button(Home, text='Back', command=Back).pack(pady=20, fill=X)


def Back():
    Home.destroy()
        root.deiconify()
def getdata(url):
    r = requests.get(url) return r.text




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


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


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


# find the Html tag # with find()
# and convert into 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)
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
```

FEATURE 2

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

USERNAME_FIELD = "email"

REQUIRED_FIELDS = ["first_name", "last_name"] email = models.EmailField(
    verbose_name="E-mail",
    unique=True
)

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

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

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


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


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


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



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


    phone_number           =           models.CharField(
        verbose_name="Phone number", max_length=15
)
```

```
    date_of_birth = models.DateField(
        verbose_name="Date of birth"
)


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


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


    class Meta: abstract = True



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


user = models.OneToOneField(
    to=User, on_delete=models.CASCADE, related_name="profile",
)
```

```
        group = models.CharField( verbose_name="Group type",
            choices=GroupTypeChoices.choices(), max_length=20,
            default=GroupTypeChoices.EMPLOYEE.name,
    )


        def____str____(self): return
            self.user.email


    class Meta:


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


user1.set_unusable_password() group_name = "employer"


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


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


user2.set_unusable_password() group_name = "employee"


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

)


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

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


mcc_code, url = "1520", "https://www.softserveinc.com/" response_ca = stripe.Account.create()

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

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

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

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

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

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

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

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


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


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


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


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

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


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


stripe.PaymentIntent.cancel(payment_intent.id)




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


    decorators = [login_required] def get(self):

        return self.render()


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


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


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


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



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


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


    def post(self):
            if self.form.validate_on_submit(): try:
                    self.password_update_handler.apply_changeset( current_user,
                        self.form.as_change()

                )
                except StopValidation as e:
                    self.form.populate_errors(e.reasons) return self.render()
            except PersistenceError:
                    logger.exception("Error while changing password") flash(_("Error while changing
                    password"), "danger") return self.redirect()


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


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


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



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

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


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


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


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


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


if s>0 and s<73:
```

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


# Driver code s = 10
berth_type(s)                   # fxn call for berth type


s = 7
berth_type(s)                  # fxn call for berth type


s = 0
berth_type(s)                   # fxn call for berth type class
   Ticket:

      counter=0
          def____init_(self,passenger_name,source,destination): self.
              ____passenger_name=passenger_name
          self.________source=source
          self.________destination=destination
              self.Counter=Ticket.counter
              Ticket.counter+=1
```

```
def validate_source_destination(self):
            if (self.______source=="Delhi" and (self.____________destination=="Pune" or self.
 ________destination=="Mumbai" or self.__________________destination=="Chennai" or self.
 ________destination=="Kolkata")):
                  return True else:
           return False


      def generate_ticket(self ): if True:

 ___ticket_id=self.__source[0]+self.__destination[0]+"0"+str(self.Counter) print( "Ticket id will be:",
                  ____________________________________ticket_id)
     else:
           return False
      def get_ticket_id(self): return
            self.ticket_id
      def get_passenger_name(self): return
            self.__passenger_name
def get_source(self):
            if self.______source=="Delhi": return
                  self.____source
     else:
           print("you have written invalid soure option") return None
def get_destination(self):
            if self.______destination=="Pune": return self.
                  ________destination
            elif self.______destination=="Mumbai": return self.
                  __________destination
```

```
                elif self.______destination=="Chennai": return self.
                    __________destination
                elif self.______destination=="Kolkata": return self.
                    __________destination


            else:
                return None
                # user define function # Scrape
        the data
def getdata(url):
                                r = requests.get(url) return r.text


# input by geek
train_name = "03391-rajgir-new-delhi-clone-special-rgd-to-ndls"


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


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


# traverse the live status from # this Html code
data = []
for item in soup.find_all('script', type="application/ld+json"):
                                data.append(item.get_text())
```

```
# convert into dataframe df =
    pd.read_json(data[2])

# display this column of # dataframe
print(df["mainEntity"][0]['name'])
print(df["mainEntity"][0]['acceptedAnswer']['text'])
Speak method
def Speak(self, audio):

                    # Calling the initial constructor # of pyttsx3
                    engine = pyttsx3.init('sapi5')

                    # Calling the getter method
                    voices = engine.getProperty('voices')

                    # Calling the setter method engine.setProperty('voice', voices[1].id)

                    engine.say(audio) engine.runAndWait()


def Take_break():

                    Speak("Do you want to start sir?") question = input()

                    if "yes" in question:
```

```
                    Speak("Starting Sir")

            if "no" in question:
                Speak("We will automatically start after 5 Mins

Sir.")
                time.sleep(5*60) Speak("Starting
                    Sir")

            # A notification we will held that
            # Let's Start sir and with a message of # will tell you to take a
                 break after 45 # mins for 10 seconds
                 while(True): notification.notify(title="Let's Start sir",
                message="will tell you to take a break after 45 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")

mins",          notification.notify(title="Break Notification", message="Please do use your device after
                    sometime

                "been continuously using it for 45 mins and it will timeout=10)

as you have" affect your eyes",
# Driver's Code
if__name_== '__main_':
                                        Take_break() data_path =
    'data.csv'
data = pd.read_csv(data_path, names=['LATITUDE', 'LONGITUDE'], sep=',')
gps_data = tuple(zip(data['LATITUDE'].values, data['LONGITUDE'].values))


image = Image.open('map.png', 'r') # Load map image. img_points = []
for d in gps_data:
        x1, y1 = scale_to_img(d, (image.size[0], image.size[1])) # Convert GPS coordinates to image coordinates.
        img_points.append((x1, y1)) draw =
    ImageDraw.Draw(image)
draw.line(img_points, fill=(255, 0, 0), width=2) # Draw converted records to the map image.


image.save('resultMap.png')
x_ticks = map(lambda x: round(x, 4), np.linspace(lon1, lon2, num=7)) y_ticks = map(lambda x: round(x, 4),
    np.linspace(lat1, lat2, num=8))
y_ticks = sorted(y_ticks, reverse=True) # y ticks must be reversed due to conversion to image coordinates.


fig, axis1 = plt.subplots(figsize=(10, 10))
axis1.imshow(plt.imread('resultMap.png')) # Load the image to matplotlib plot.
axis1.set_xlabel('Longitude')
```

```
axis1.set_ylabel('Latitude')
      axis1.set_xticklabels(x_ticks)
      axis1.set_yticklabels(y_ticks) axis1.grid()
plt.show() class tickets:
            def_____init_____(self):
                  self.no_ofac1stclass=0 self.totaf=0
                  self.no_ofac2ndclass=0
                  self.no_ofac3rdclass=0
                  self.no_ofsleeper=0
                  self.no_oftickets=0 self.name=''
            self.age='' self.resno=0
                  self.status=''
            def ret(self): return(self.resno)
            def retname(self):
                  return(self.name)
            def display(self): f=0
            fin1=open("tickets.dat","rb") if not fin1:
                        print "ERROR" else:
                  print
                  n=int(raw_input("ENTER PNR NUMBER : ")) print "\n\n"
```

```
print ("FETCHING DATA . . .".center(80)) time.sleep(1)
print
print('PLEASE WAIT...!!'.center(80)) time.sleep(1)
os.system('cls') try:
            while True: tick=load(fin1)
                 if(n==tick.ret()):
                 f=1
                 print "="*80
                 print("PNR STATUS".center(80)) print"="*80
                 print
                 print "PASSENGER'S NAME :",tick.name print
                 print "PASSENGER'S AGE :",tick.age print
                 print "PNR NO :",tick.resno print
                 print "STATUS :",tick.status print
                 print "NO OF SEATS BOOKED : ",tick.no_oftickets print
except:
            pass
      fin1.close()
      if(f==0):
     print
```

```
            print "WRONG PNR NUMBER..!!"
            print
 def pending(self): self.status="WAITING LIST" print
        "PNR NUMBER :",self.resno print
  time.sleep(1.2)
  print "STATUS = ",self.status print
  print "NO OF SEATS BOOKED : ",self.no_oftickets print
 def confirmation (self): self.status="CONFIRMED"
print "PNR NUMBER : ",self.resno print
time.sleep(1.5)
print "STATUS = ",self.status print
 def cancellation(self): z=0
f=0 fin=open("tickets.dat","rb")
       fout=open("temp.dat","ab") print
r= int(raw_input("ENTER PNR NUMBER : ")) try:
             while(True):
                   tick=load(fin)
                   z=tick.ret() if(z!=r):
                    dump(tick,fout)
                  elif(z==r):
                  f=1
   except:
         pass
       fin.close()
       fout.close()
   os.remove("tickets.dat") os.rename("temp.dat","tickets.dat")
       if (f==0):
      print
      print "NO SUCH RESERVATION NUMBER FOUND"
      print time.sleep(2)
         os.system('cls')
   else:
      print
      print "TICKET CANCELLED" print"RS.600
         REFUNDED ................................................. "
def reservation(self):
       trainno=int(raw_input("ENTER THE TRAIN NO:")) z=0
   f=0 fin2=open("tr1details.dat") fin2.seek(0)
   if not fin2:
         print "ERROR" else:
      try:
         while True:
```

```
                    tr=load(fin2) z=tr.gettrainno()
                       n=tr.gettrainname() if
                       (trainno==z):
                      print
                      print "TRAIN NAME IS : ",n
                      f=1 print
                      print "-"*80 no_ofac1st=tr.getno_ofac1stclass()
                         no_ofac2nd=tr.getno_ofac2ndclass()
                         no_ofac3rd=tr.getno_ofac3rdclass()
                         no_ofsleeper=tr.getno_ofsleeper()
                       if(f==1): fout1=open("tickets.dat","ab") print
                      self.name=raw_input("ENTER THE PASSENGER'S

                      print
                      self.age=int(raw_input("PASSENGER'S AGE : ")) print
                      print"\t\t SELECT A CLASS YOU WOULD LIKE TO

NAME ")

TRAVEL IN :- "
                      print "1.AC FIRST CLASS"
                      print
                      print "2.AC SECOND CLASS"
                      print
                      print "3.AC THIRD CLASS"
                      print
                      print "4.SLEEPER CLASS"
                    print
                    c=int(raw_input("\t\t\tENTER YOUR CHOICE = ")) os.system('cls')
                    amt1=0 if(c==1):
                            self.no_oftickets=int(raw_input("ENTER NO_OF FIRST CLASS AC
SEATS TO BE BOOKED : "))
                            i=1 while(i<=self.no_oftickets):
                            self.totaf=self.totaf+1 amt1=1000*self.no_oftickets
                                i=i+1
                        print
                        print "PROCESSING. .",
                            time.sleep(0.5) print
                            ".", time.sleep(0.3)
                            print'.' time.sleep(2)
                            os.system('cls')
                        print "TOTAL AMOUNT TO BE PAID = ",amt1
                        self.resno=int(random.randint(1000,2546)) x=no_ofac1st-self.totaf
                            print
                            if(x>0):
                            self.confirmation()
                                dump(self,fout1) break
                        else:
                            self.pending()
```

```
                                                dump(tick,fout1) break
                                            elif(c==2):  self.no_oftickets=int(raw_input("ENTER NO_OF
SECOND CLASS AC SEATS TO BE BOOKED :   "))
                                            i=1




     def menu(): tr=train()
          tick=tickets() print
     print "WELCOME TO PRAHIT AGENCY".center(80)
     while True:
               print
               print "="*80
               print " \t\t\t\t RAILWAY" print
               print "="*80 print
               print "\t\t\t1. **UPDATE TRAIN DETAILS." print
               print "\t\t\t2. TRAIN DETAILS. " print
               print "\t\t\t3. RESERVATION OF TICKETS." print
               print "\t\t\t4. CANCELLATION OF TICKETS. "
               print
               print "\t\t\t5. DISPLAY PNR STATUS."
```

```
                print
                print "\t\t\t6. QUIT." print"** - office use
                    ......................................................... "
                ch=int(raw_input("\t\t\tENTER YOUR CHOICE : ")) os.system('cls')
                    print "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\tLOADI NG. .",
                time.sleep(1) print ("."),
                    time.sleep(0.5) print
                    (".") time.sleep(2)
                    os.system('cls') if
                    ch==1:
                    j="*****" r=raw_input("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\tENTER THE
PASSWORD: ")
                    os.system('cls') if (j==r):
                        x='y'
                            while (x.lower()=='y'): fout=open("tr1details.dat","ab") tr.getinput()
                            dump(tr,fout) fout.close()
                                print"\n\n\n\n\n\n\n\n\n\n\n\n\t\t\tUPDATING TRAIN LIST PLEASE WAIT . .",
                            time.sleep(1) print
                                ("."),
      time.sleep(0.5) print ("."),
             time.sleep(2)
             os.system('cls')
      print "\n\n\n\n\n\n\n\n\n\n\n"
             x=raw_input("\t\tDO YOU WANT TO ADD ANY MORE TRAINS DETAILS ? ")
             os.system('cls')
       continue
 elif(j<>r): print"\n\n\n\n\n"
       print "WRONG PASSWORD".center(80) elif ch==2:
 fin=open("tr1details.dat",'rb') if not fin:
       print "ERROR" else:
try:
             while True:
                   print"*"*80
            print"\t\t\t\tTRAIN DETAILS" print"*"*80
            print tr=load(fin)
                   tr.output()




            raw_input("PRESS ENTER TO VIEW NEXT TRAIN
DETAILS")
```

```
                                os.system('cls') except
                          EOFError:
                               pass
                elif ch==3:

                print'='*80
                print "\t\t\t\tRESERVATION OF TICKETS" print'='*80
                print tick.reservation()
                elif ch==4:
                   print"="*80
                print"\t\t\t\tCANCELLATION OF TICKETS" print
                print"="*80 print
                   tick.cancellation()
                elif ch==5: print "="*80
                print("PNR STATUS".center(80)) print"="*80
                   printclass tickets: def
       ________init_____(self):
self.no_ofac1stclass=0 self.totaf=0
   self.no_ofac2ndclass=0
   self.no_ofac3rdclass=0
   self.no_ofsleeper=0
   self.no_oftickets=0 self.name="
self.age="
self.resno=0 self.status=''
 def ret(self): return(self.resno)
 def retname(self):
       return(self.name)
 def display(self): f=0
fin1=open("tickets.dat","rb") if not fin1:
             print "ERROR" else:
      print
      n=int(raw_input("ENTER PNR NUMBER : ")) print "\n\n"
      print ("FETCHING DATA . . .".center(80)) time.sleep(1)

      print
      print('PLEASE WAIT...!!'.center(80)) time.sleep(1)
      os.system('cls') try:
                   while True: tick=load(fin1)
                         if(n==tick.ret()):
                        f=1
                        print "="*80
                        print("PNR STATUS".center(80)) print"="*80
                        print
```

```
                    print "PASSENGER'S NAME :",tick.name print
                    print "PASSENGER'S AGE :",tick.age print
                    print "PNR NO :",tick.resno print
                    print "STATUS :",tick.status print
                    print "NO OF SEATS BOOKED : ",tick.no_oftickets print
      except:
                 pass
          fin1.close()
          if(f==0):
          print
          print "WRONG PNR NUMBER..!!"
          print
 def pending(self): self.status="WAITING LIST" print
       "PNR NUMBER :",self.resno print
 time.sleep(1.2)
 print "STATUS = ",self.status print
 print "NO OF SEATS BOOKED : ",self.no_oftickets print
 def confirmation (self): self.status="CONFIRMED"
print "PNR NUMBER : ",self.resno print
```

```
time.sleep(1.5)
print "STATUS = ",self.status print
 def cancellation(self): z=0
f=0 fin=open("tickets.dat","rb")
    fout=open("temp.dat","ab") print
r= int(raw_input("ENTER PNR NUMBER : ")) try:
        while(True):
            tick=load(fin)
            z=tick.ret() if(z!=r):
                dump(tick,fout)
            elif(z==r):
             f=1
except:
        pass
    fin.close()
    fout.close()
os.remove("tickets.dat") os.rename("temp.dat","tickets.dat")
    if (f==0):
   print
   print "NO SUCH RESERVATION NUMBER FOUND"
   print time.sleep(2)
        os.system('cls')
    else:
        print
        print "TICKET CANCELLED" print"RS.600
            REFUNDED ................................................. "
def reservation(self):
         trainno=int(raw_input("ENTER THE TRAIN NO:")) z=0
    f=0 fin2=open("tr1details.dat") fin2.seek(0)
    if not fin2:
            print "ERROR" else:
        try:
                while True: tr=load(fin2)
                    z=tr.gettrainno()
                    n=tr.gettrainname() if
                    (trainno==z):
                    print
                    print "TRAIN NAME IS : ",n
                    f=1 print
                    print "-"*80 no_ofac1st=tr.getno_ofac1stclass()
                        no_ofac2nd=tr.getno_ofac2ndclass()
                        no_ofac3rd=tr.getno_ofac3rdclass()
                        no_ofsleeper=tr.getno_ofsleeper()
                    if(f==1): fout1=open("tickets.dat","ab")
                    print
                    self.name=raw_input("ENTER THE PASSENGER'S

NAME ")
                    print
                    self.age=int(raw_input("PASSENGER'S AGE : ")) print
                    print"\t\t SELECT A CLASS YOU WOULD LIKE TO
TRAVEL IN :- "
                    print "1.AC FIRST CLASS"
                    print
                    print "2.AC SECOND CLASS"
                    print
                    print "3.AC THIRD CLASS"
                    print
                    print "4.SLEEPER CLASS"
                    print
                    c=int(raw_input("\t\t\tENTER YOUR CHOICE = ")) os.system('cls')
                    amt1=0 if(c==1):
                        self.no_oftickets=int(raw_input("ENTER NO_OF FIRST CLASS AC
  SEATS TO BE BOOKED : "))
                        i=1 while(i<=self.no_oftickets):
                        self.totaf=self.totaf+1 amt1=1000*self.no_oftickets
                          i=i+1
                      print
                      print "PROCESSING. .",
                      time.sleep(0.5)
```

```
                                        print ".",
                                        time.sleep(0.3) print'.'
                                        time.sleep(2)
                                        os.system('cls')
                                    print "TOTAL AMOUNT TO BE PAID = ",amt1
                                    self.resno=int(random.randint(1000,2546)) x=no_ofac1st-self.totaf
                                        print
                                        if(x>0):
                                       self.confirmation()
                                           dump(self,fout1) break
                                    else:
                                       self.pending() dump(tick,fout1)
                                           break
                                    elif(c==2):  self.no_oftickets=int(raw_input("ENTER NO_OF
SECOND CLASS AC SEATS TO BE BOOKED :  "))
                                    i=1




    def menu(): tr=train()
        tick=tickets() print
    print "WELCOME TO PRAHIT AGENCY".center(80)
    while True:
```

```
print
print "="*80
print " \t\t\t\t RAILWAY" print
print "="*80 print
print "\t\t\t1. **UPDATE TRAIN DETAILS." print
print "\t\t\t2. TRAIN DETAILS. " print
print "\t\t\t3. RESERVATION OF TICKETS." print
print "\t\t\t4. CANCELLATION OF TICKETS. "
print
print "\t\t\t5. DISPLAY PNR STATUS." print
print "\t\t\t6. QUIT." print"** - office use
    ........................................................ "
ch=int(raw_input("\t\t\tENTER YOUR CHOICE : ")) os.system('cls')
    print "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\tLOADI NG. .",
time.sleep(1) print ("."),
    time.sleep(0.5) print
    (".") time.sleep(2)
    os.system('cls') if
    ch==1:
            j="*****" r=raw_input("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\tENTER THE
PASSWORD: ")
            os.system('cls') if (j==r):
               x='y'
                 while (x.lower()=='y'): fout=open("tr1details.dat","ab") tr.getinput()
                 dump(tr,fout) fout.close()
                   print"\n\n\n\n\n\n\n\n\n\n\n\t\t\tUPDATING TRAIN LIST PLEASE WAIT . .",
                 time.sleep(1) print ("."),
                   time.sleep(0.5) print
                   ("."), time.sleep(2)
                   os.system('cls')
                 print "\n\n\n\n\n\n\n\n\n\n\n"
                   x=raw_input("\t\tDO YOU WANT TO ADD ANY MORE TRAINS DETAILS ? ")
                   os.system('cls')
                 continue
               elif(j<>r): print"\n\n\n\n\n"
                 print "WRONG PASSWORD".center(80) elif ch==2:
               fin=open("tr1details.dat",'rb') if not fin:
```

```
                        print "ERROR"
                    tick.display()
                elif ch==6:
                    quit()


            raw_input("PRESS ENTER TO GO TO BACK

MENU".center(80))
            os.system('cls')


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


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


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

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

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

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

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

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


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


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


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


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


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

```
text = message.as_string()


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


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


# Enter valid pnr_number pnr_number =
    "6515483790"


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


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


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


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

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

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

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

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

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

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

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


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


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


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


# looping through passenger list
```

```
for passenger in passengers_list:


# store the value or data # of "no" key in variable

passenger_num = passenger["no"]


# store the value or data of

# "current_status" key in variable current_status = passenger["current_status"]


# store the value or data of

# "booking_status" key in variable booking_status = passenger["booking_status"]


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


else:

print("Record Not Found")
```

13.1. GIT HUB LINK

IBM-EPBL/IBM-Project-37684-1660317330

13.2 DEMO VIDEO LINK

https://drive.google.com/file/d/1ebZbFN6V9luZbbEABc2aNtmuKtUo7s-V/view?usp=drivesdk

https://youtu.be/6vV0g0giqTE