PROJECT REPORT

INTRODUCTION

1.1PROJECT OVERVIEW:

A smart railway station is a station area that uses different types of electronic Internet of things (IOT) sensors to collect data and use that data to better improve efficiency, mobility and sustainability. It mainly includes. smart management, smart infrastructure, and smart mobility. Railways have been an essential mode of transportation to people all over the world for centuries. They were critical to the industrial revolution and played a major role in creating thriving, innovative societies. Today, railways are more important than ever as country and city governments are being asked to find innovative ways to safely get back to business post-COVID, meet the changing needs of their citizens, address urban population increases, and reduce their environmental impact.

To meet these challenges and position themselves for future success, many forward-thinking governments and railway operators are looking for smart, intelligent IoT technologies to modernize their railways.

1.2 PURPOSE:

Its application increases safety, efficiency and ease of use with train management systems. Control and surveillance systems reduce the risk of collisions and regulate speed. Advanced consumer technologies help maximize connectivity and allow passengers to continue their activities on smart devices while traveling. Rail transport (also known as train transport) is a means of transport that transfers passengers and goods on wheeled vehicles running on rails, which are incorporated in tracks.

The Corporate aim of the Indian Railways is to commit itself to ensuring that all its activities are managed to the highest level of safety which is pragmatic and reasonably practicable to achieve. In terms of the economy, railways played a major role in integrating markets and increasing trade. In terms of politics, railways shaped the finances of the colonial government and the Princely States.

LITERATURE SURVEY

2.1 EXISTING PROBLEM:

The Internet of Things seems to be created for use in the Railways; even the acronym “IoT”
might be decrypted as the Internet of Trains. Really, IoT sensors measuring speed, vibration,
telemetry, brakes, and more have made it much easier to monitor the schedule, detect route
issues, and eliminate human mistakes while operating a train.
Smart railway is a technologically advanced approach to efficiently manage railway operations
through sharing of rail data across rail infrastructure components, such as passengers, control
centers, ticketing department, and freight.

2.2 REFERENCES:

- Internet of Things for Smart Railway: Feasibility and Applications

Author: Ohyun Jo, Graduate Student Member, IEEE, Yong-Kyu Kim, Member, IEEE, and Juyeop Kim, Member, IEEE 2018

- Internet of Things in the Railway Domain: Edge Sensing System Based on Solid-State LIDAR and Fuzzy Clustering for Virtual Coupling

Author: GABRIEL MUJICA , (Member, IEEE), JAVIER HENCHE, AND JORGE PORTILLA , (Senior Member, IEEE)

- Robust Railway Crack Detection Scheme (RRCDS) Using LED-LDR Assembly 2012 Internet of things

Author: Gourav saha, vaidehi,vigneshwar murali

- Review on railway track crack detection using IR transmitter and receiver

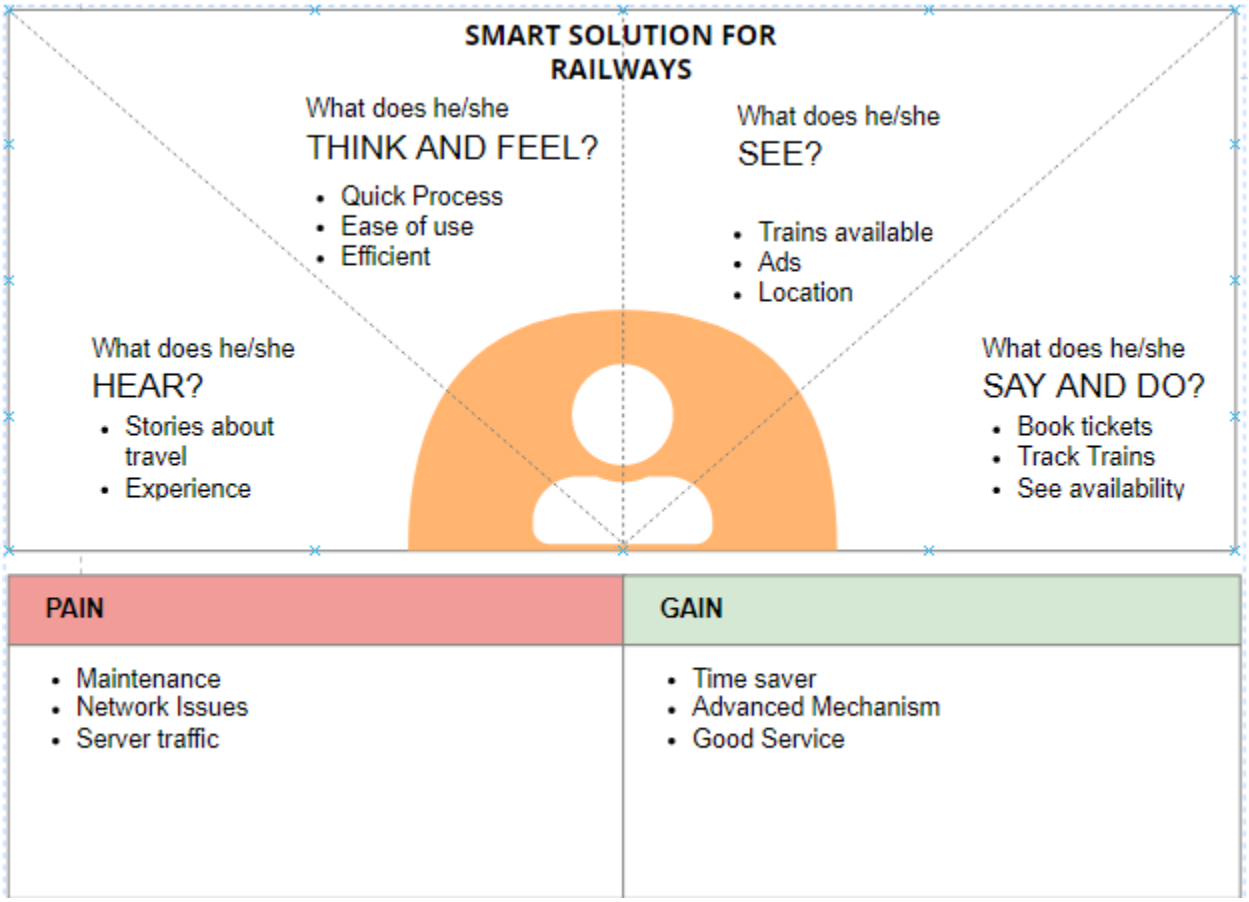
Author: Rakesh V. Pise1, Parag D. Nikhar2, Prof. Avinash H.Shelar

2.3 PROBLEM STATEMENT DEFINITION:

- We don’t need to spend time in entering into application directly
- we can use QR scanner.
- It improves encryption to avoid misleading of data.
- Automatic gate systems can be employed.
- Track fault and barrier object detection can save many lives.

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:

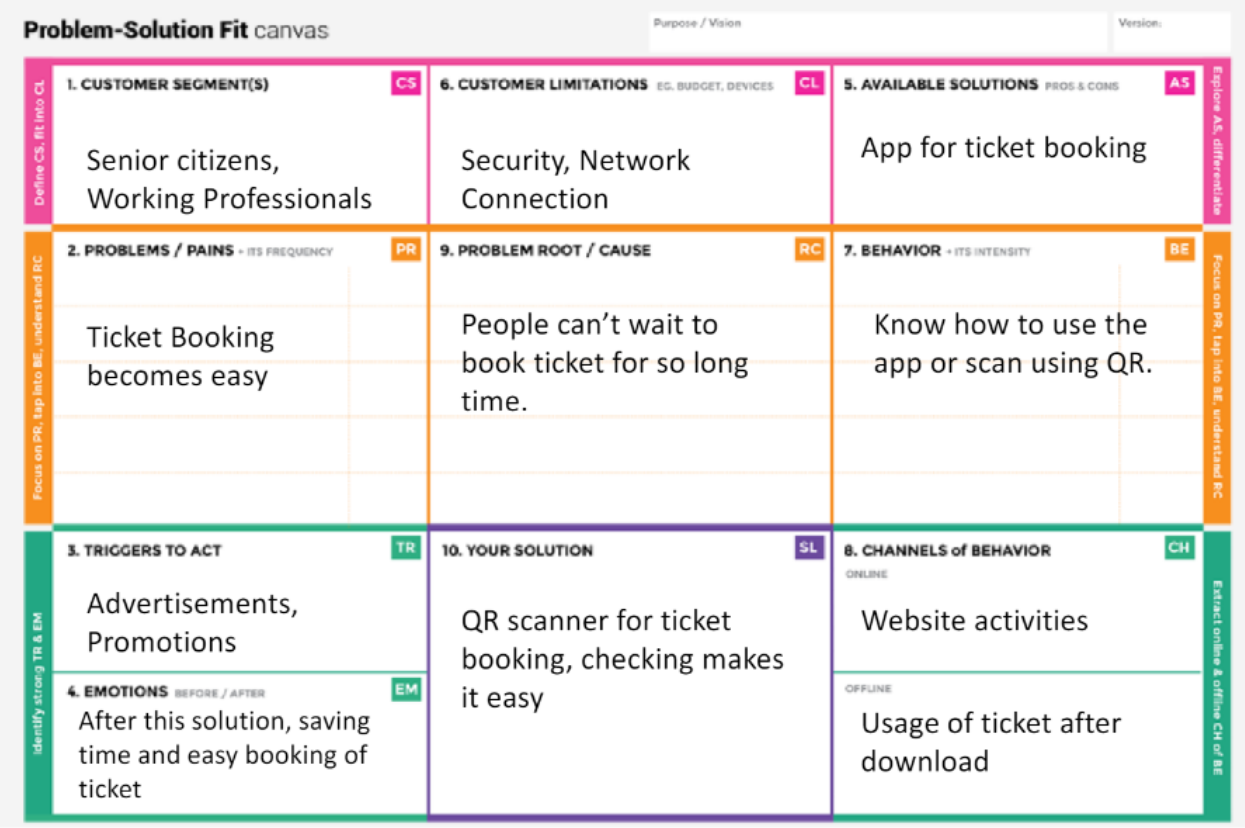


3.2 IDEATION AND BRAINSTORMING:

3.3 PROPOSED SOLUTION:

S No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Passengers often have various issues in booking ticket.
2.	Idea / Solution description	Ticket is provided to passenger as in bus booking system. It is easy to confirm the train ticket. If we find the availability of ticket, ticket can be booked and confirmed.
3.	Novelty / Uniqueness	If there is availability of ticket, seats are booked. QR code can be scanned to know the status.
4.	Social Impact / Customer Satisfaction	Customer can book ticket on the go and it is easy for them compared to the traditional way of booking.
5.	Business Model (Revenue Model)	E-ticket, Drinking water, Tourism, Catering are providing revenue for railways.
6.	Scalability of the Solution	Providing various food delivery options from nearby restaurants to the railway stations.

3.4 PROBLEM SOLUTION FIT:



REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT:

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.

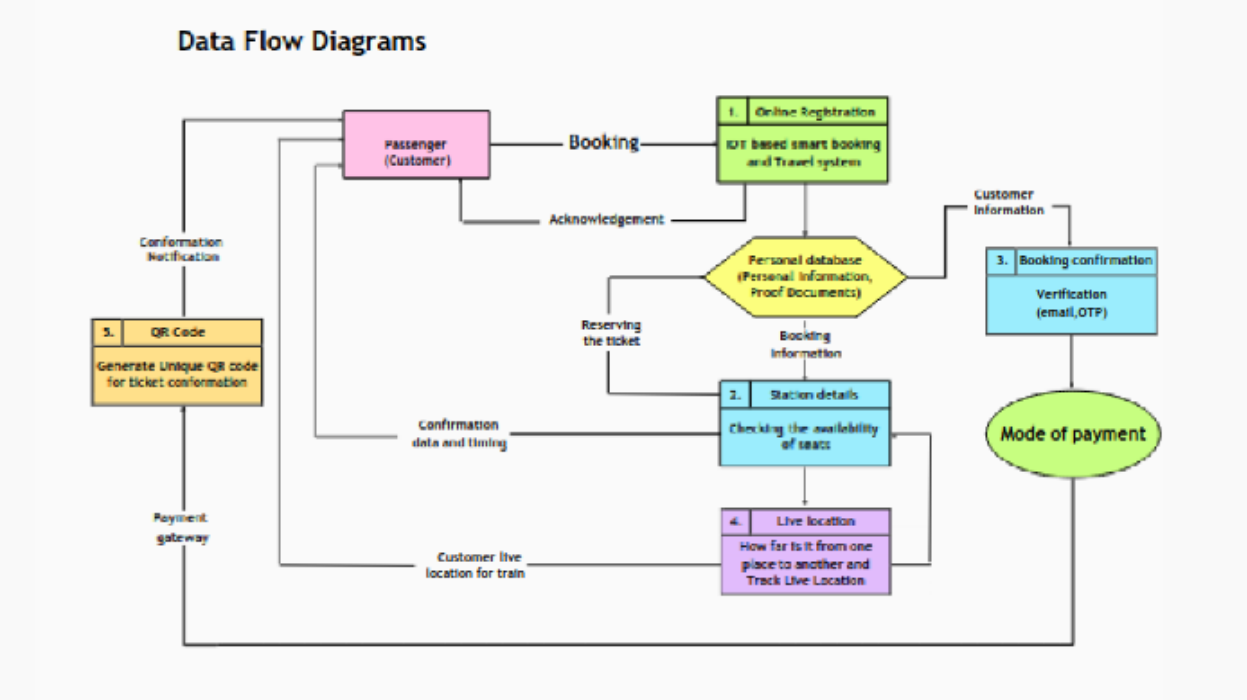
4.2 NON-FUNCTIONAL REQUIREMENT:

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.

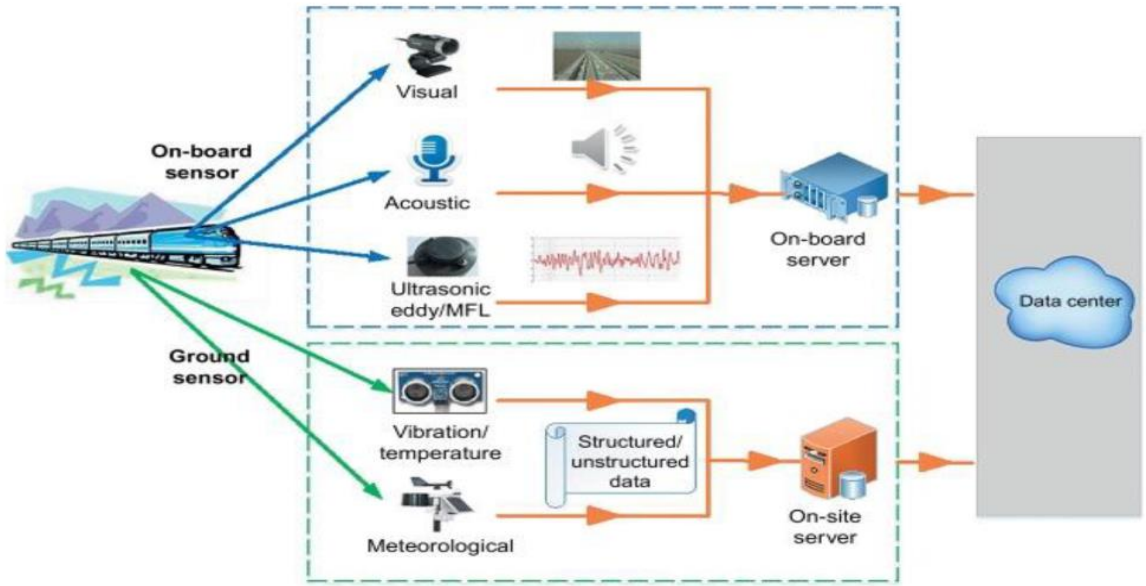
NFR-3	**Reliability**	It enables the user to securely use the app which provides maximum trust to the user.
NFR-4	**Performance**	No server down problem.
NFR-5	**Availability**	Accessibility through website or application anytime and from anywhere.
NFR-6	**Scalability**	Number of users concurrently interacting with our web application with higher reliability.

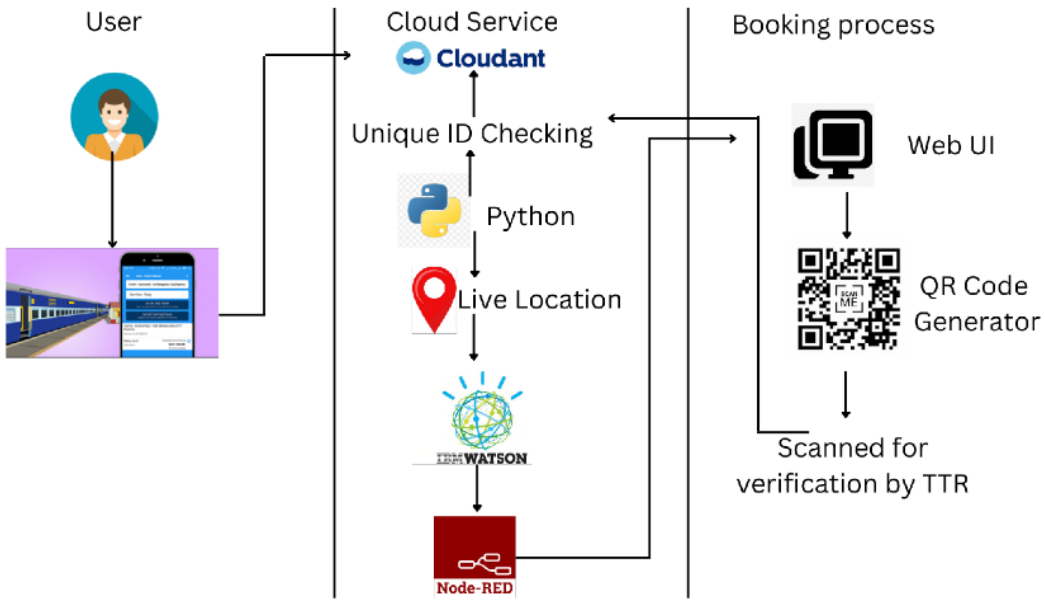
PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS:



5.2 SOLUTION AND TECHNICAL ARCHITECTURE:





5.3 USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Customer	Registration	USN-1	As a user, I can register for the application by entering my email, password, and	I can register successfully.	High	Sprint-1

			confirming my password.			
	Conformation	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can confirm the registration.	High	Sprint-1
	login	USN-3	As a user, I can register for the application through Facebook	I can login and register via facebook.	Low	Sprint-1
	Display train details	USN-4	As a user, I can register for the application through Gmail	I can view registered applications.	Medium	Sprint-1
	Booking	USN-5	As a user, I can log into the application by	I can log into the application	High	Sprint-2

			entering email & password	and enter my login.		
		USN-7	As a user, I can choose the class, seat/berth. If a preferred seat/berth isn't available I can be allocated based on the availability	I can choose the seat which is available.	Low	Sprint-2
	Payment	USN-8	As a user, I can choose to pay through credit Card/debit card/UPI.	I can pay via any card.	High	Sprint-2
	Functional Requirement	USN-9	User Story / Task	Story/task	High	Sprint-2

	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 journey	I can view e-ticket.	High	Sprint-3
	Ticket Status	USN-11	As a user, I can see the status of my ticket	I can see the status.	Medium	Sprint-3
	Reminder Notification	USN-12	As a user, I get remainders about my journey A day before my actual journey.	I can get reminders.	Low	Sprint-3
	Ticket Cancellation	USN-13	As a user, trains can be tracked via GPS.	I can track train details.	Medium	Sprint-4

	Raise Queries	USN-14	As a user, queries can be raised via mail.	I can raise queries.	Low	Sprint-4
Customer Care	Answer Queries	USN-15	As a user, queries can be answered via mail.	I can answer the queries.	Low	Sprint-4
Admin	Feed Details	USN-16	As a user, information can be sent back as feedback.	I can send the feedback.	Low	Sprint-4

PROJECT PLANNING AND

SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION:

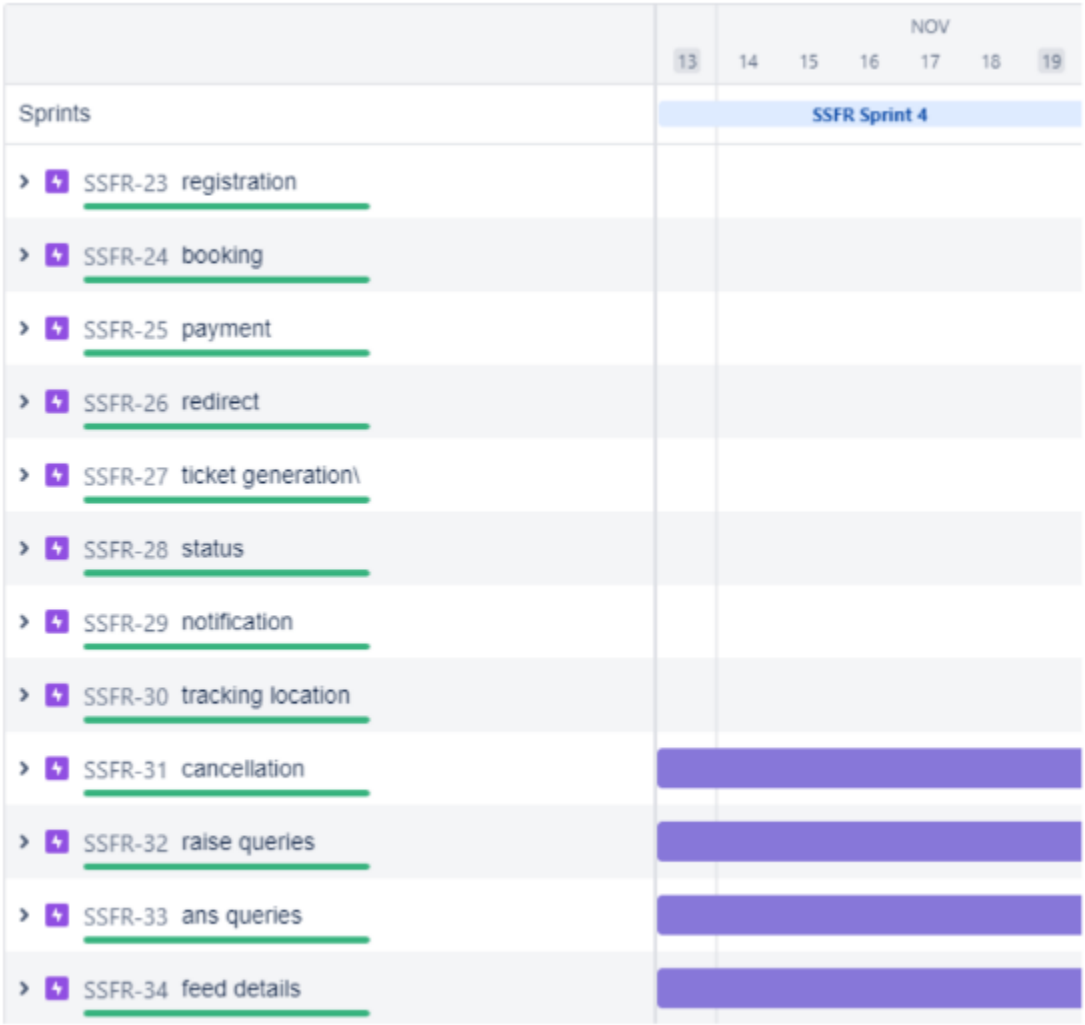
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Preethi
Sprint-1	Conformation	USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Pavithra
Sprint-1	login	USN-3	As a user, I can register for the application through Facebook	2	Low	Preethi
Sprint-1	Display train details	USN-4	As a user, I can register for the application through Gmail	2	Medium	Pavithra
Sprint-2	Booking	USN-5	As a user, I can log into the application by entering email & password	1	High	Irfana
Sprint-2		USN-7	As a user, I can choose the class, seat/berth. If a preferred seat/berth isn't available I can be allocated based on the availability	2	Low	Pavithra
Sprint-2	Payment	USN-8	As a user, I can choose to pay through credit Card/debit card/UPI.	1	High	Pavithra
Sprint-2	Functional Requirement	USN-9	User Story / Task	2	High	Irfana
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 journey	1	High	Leo
Sprint-3	Ticket Status	USN-11	As a user, I can see the status of my ticket	1	Medium	Preethi
Sprint-3	Reminder Notification	USN-12	As a user, I get remainders about my journey A day before my actual journey.	2	Low	Leo
Sprint-4	Ticket Cancellation	USN-13	As a user, trains can be tracked via GPS.	2	Medium	Irfana
Sprint-4	Raise Queries	USN-14	As a user, queries can be raised via mail.	1	Low	Leo
Sprint-4	Answer Queries	USN-15	As a user, queries can be answered via mail.	1	Low	Leo
Sprint-4	Feed Details	USN-16	As a user, information can be sent back as feedback.	1	Low	Irfana

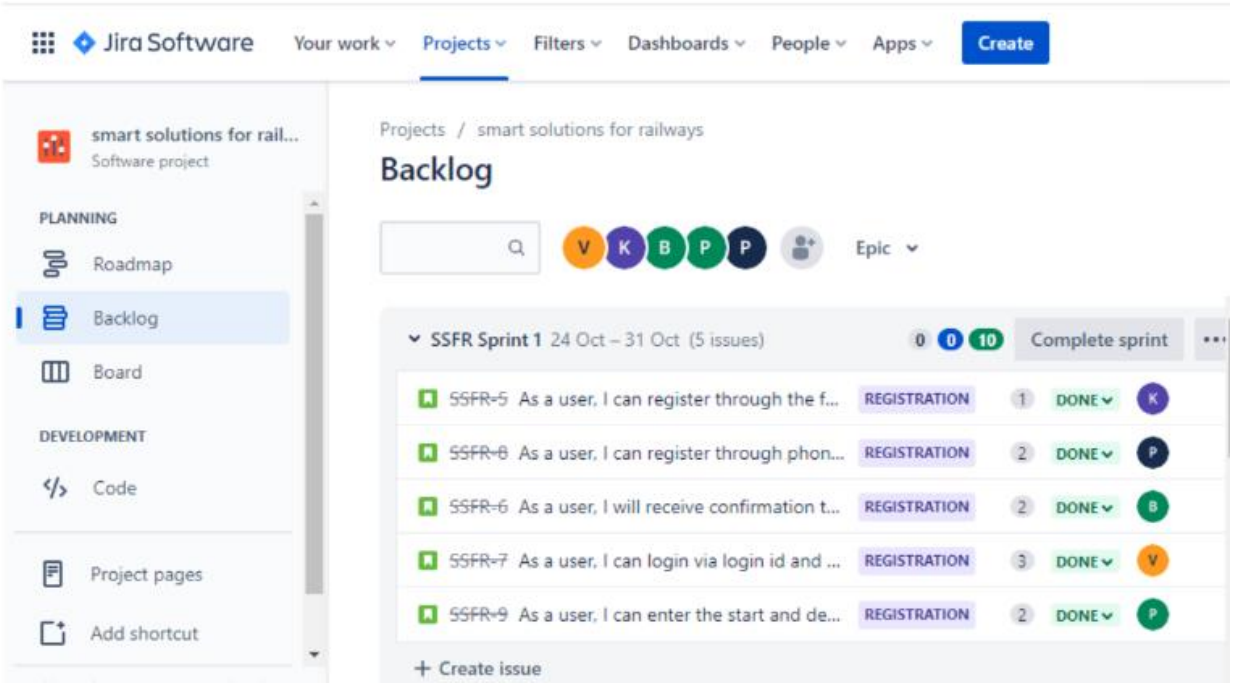
6.2 SPRINT DELIVERY SCHEDULE

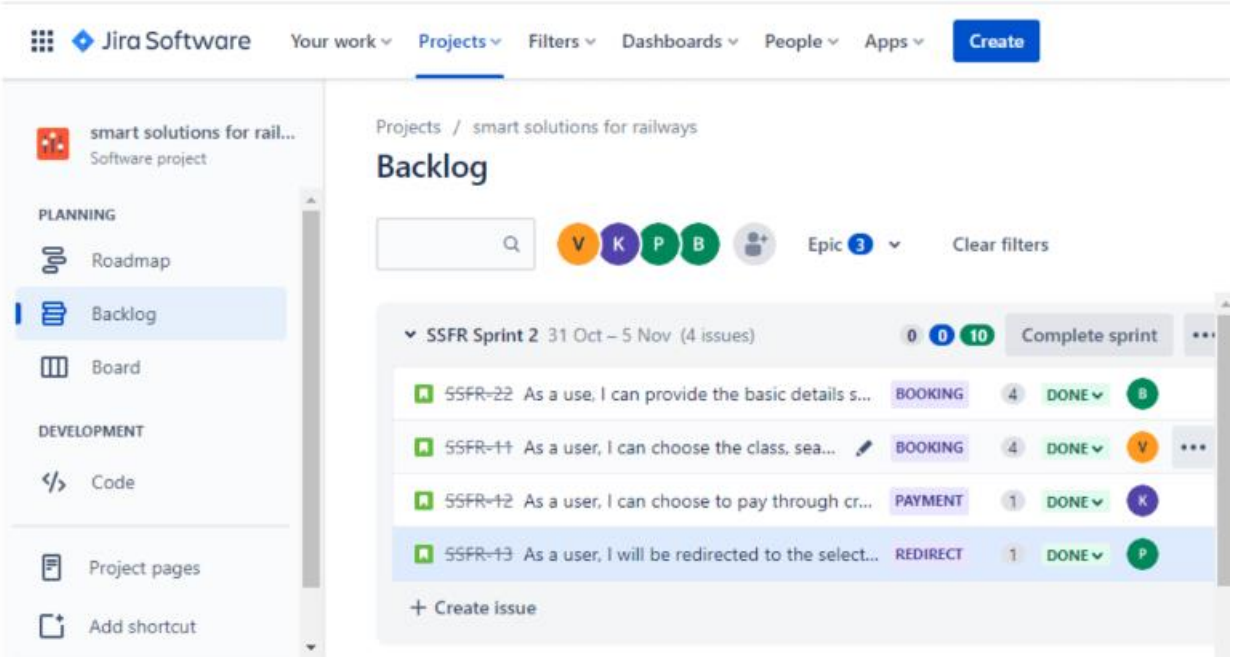
Project Tracker, Velocity & Burndown Chart:

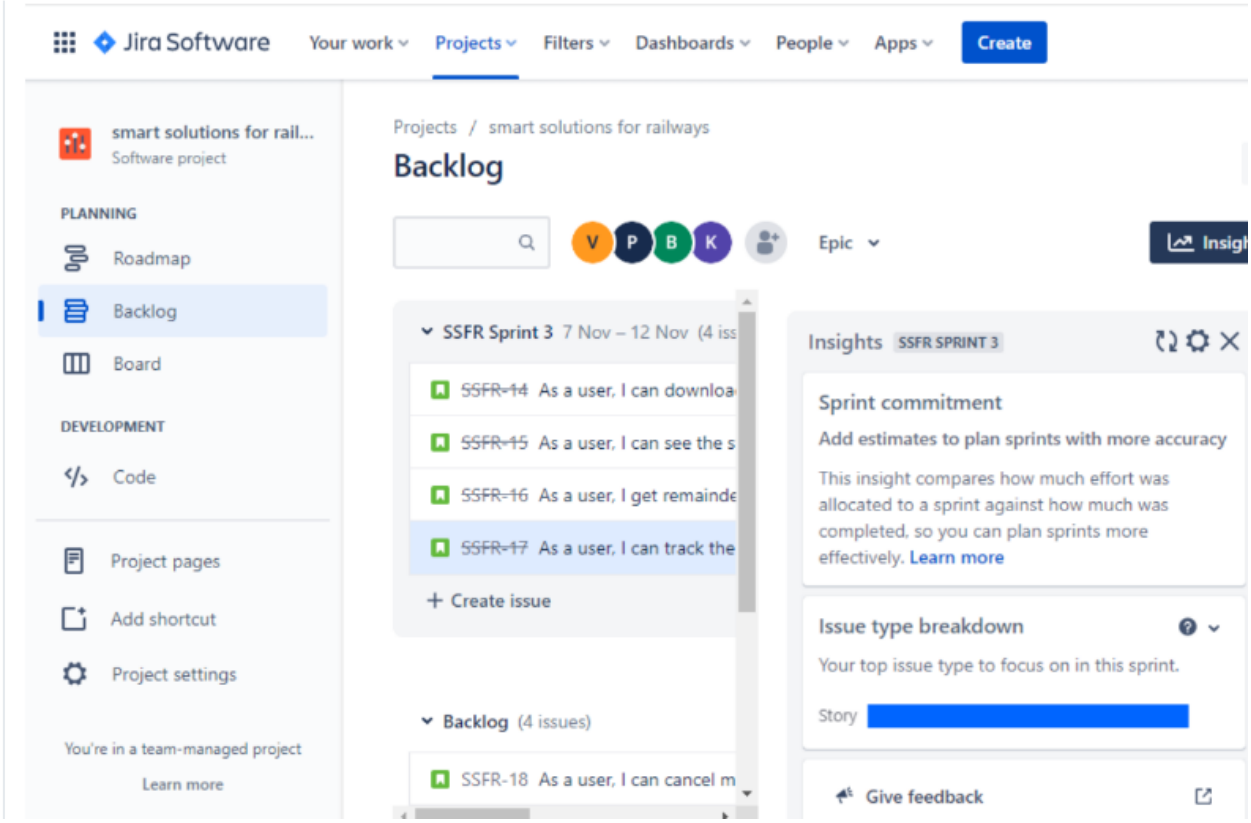
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	17 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	17 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	17 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	18 Nov 2022

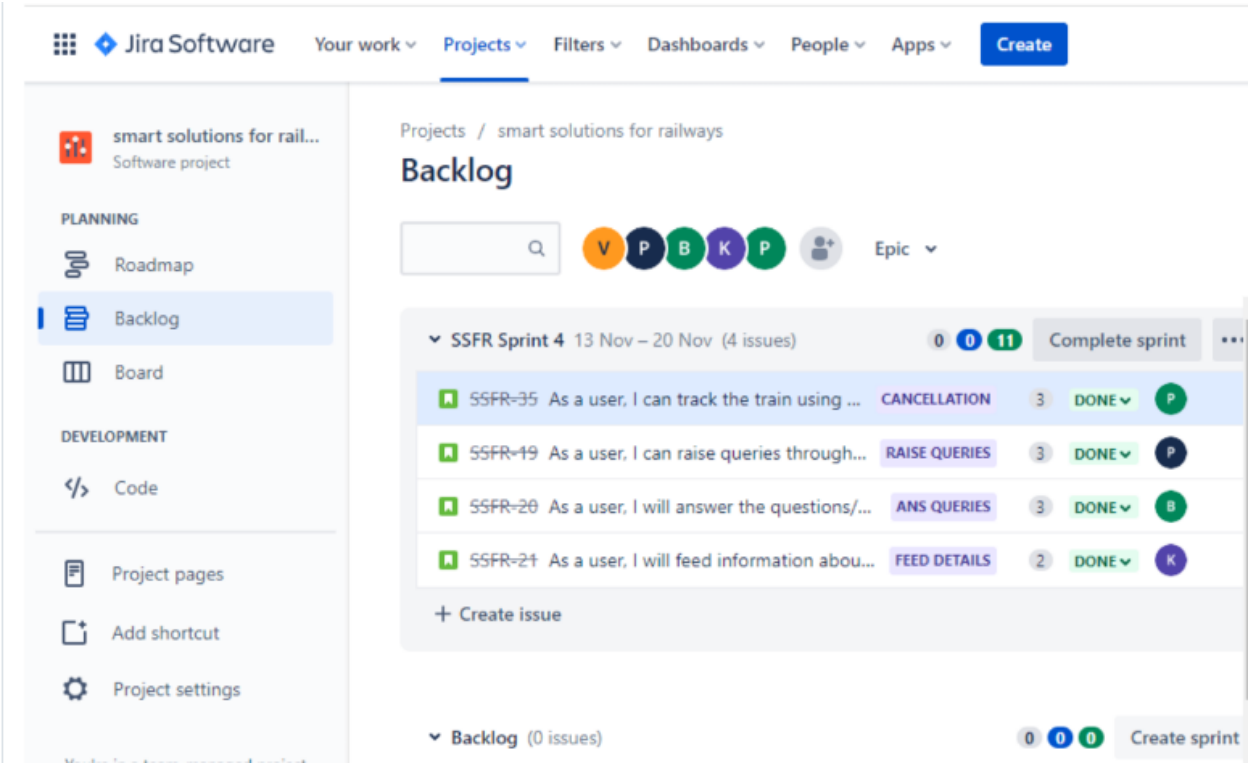
6.3 REPORTS FROM JIRA:




Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
smart solutions for rail...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Projects / smart solutions for railways
Backlog
Epic
SSFR Sprint 1 24 Oct – 31 Oct (5 issues)
0 0 10
Complete sprint
SSFR-5 As a user, I can register through the f... REGISTRATION 1 DONE
SSFR-8 As a user, I can register through phon... REGISTRATION 2 DONE
SSFR-6 As a user, I will receive confirmation t... REGISTRATION 2 DONE
SSFR-7 As a user, I can login via login id and ... REGISTRATION 3 DONE
SSFR-9 As a user, I can enter the start and de... REGISTRATION 2 DONE
Create issue


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
smart solutions for rail...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Projects / smart solutions for railways
Backlog
Epic 3
Clear filters
SSFR Sprint 2 31 Oct – 5 Nov (4 issues)
0 0 10
Complete sprint
SSFR-22 As a use, I can provide the basic details s... BOOKING 4 DONE
SSFR-11 As a user, I can choose the class, sea... BOOKING 4 DONE
SSFR-12 As a user, I can choose to pay through cr... PAYMENT 1 DONE
SSFR-13 As a user, I will be redirected to the select... REDIRECT 1 DONE
Create issue


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
smart solutions for rail...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / smart solutions for railways
Backlog
Epic
SSFR Sprint 3 7 Nov – 12 Nov (4 iss
SSFR-14 As a user, I can downloa
SSFR-15 As a user, I can see the s
SSFR-16 As a user, I get remainde
SSFR-17 As a user, I can track the
Create issue
Backlog (4 issues)
SSFR-18 As a user, I can cancel m
Insights SSFR SPRINT 3
Sprint commitment
Add estimates to plan sprints with more accuracy
This insight compares how much effort was allocated to a sprint against how much was completed, so you can plan sprints more effectively. Learn more
Issue type breakdown
Your top issue type to focus on in this sprint.
Story
Give feedback


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
smart solutions for rail...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
Projects / smart solutions for railways
Backlog
Epic
SSFR Sprint 4 13 Nov – 20 Nov (4 issues)
Complete sprint
SSFR-35 As a user, I can track the train using ... CANCELLATION DONE
SSFR-19 As a user, I can raise queries through... RAISE QUERIES DONE
SSFR-20 As a user, I will answer the questions/... ANS QUERIES DONE
SSFR-21 As a user, I will feed information abou... FEED DETAILS DONE
Create issue
Backlog (0 issues)
Create sprint

CODING AND SOLUTIONING

7.1 FEATURE 1:

- IOT device
- IBM Watson Platform
- Node red
- Cloudant DB
- Web UI
- Geofence
- MIT App
- Python Code

7.2 FEATURE 2:

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

Sample code for features:

login.py:

```
from tkinter import *
import sqlite3

root = Tk()
root.title("Python: Simple Login Application")
width = 400
height = 280
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))
root.resizable(0, 0)
```

```
    #==============================VARIABLES==========================
    USERNAME = StringVar()
    PASSWORD = StringVar()

    #==============================FRAMES============================
    Top = Frame(root, bd=2,  relief=RIDGE)
    Top.pack(side=TOP, fill=X)
    Form = Frame(root, height=200)
    Form.pack(side=TOP, pady=20)

    #==============================LABELS============================
    lbl_title = Label(Top, text = "Python: Simple Login Application", font=('arial',
15))
    lbl_title.pack(fill=X)
    lbl_username = Label(Form, text = "Username:", font=('arial', 14), bd=15)
    lbl_username.grid(row=0, sticky="e")
    lbl_password = Label(Form, text = "Password:", font=('arial', 14), bd=15)
    lbl_password.grid(row=1, sticky="e")
    lbl_text = Label(Form)
    lbl_text.grid(row=2, columnspan=2)

    #==============================ENTRY
WIDGETS=================================
    username = Entry(Form, textvariable=USERNAME, font=(14))
    username.grid(row=0, column=1)
    password = Entry(Form, textvariable=PASSWORD, show="*", font=(14))
    password.grid(row=1, column=1)




    #==============================METHODS============================
    def Database():
        global conn, cursor
        conn = sqlite3.connect("pythontut.db")
```

```
        cursor = conn.cursor()
        cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id
INTEGER NOT NULL PRIMARY KEY  AUTOINCREMENT, username TEXT,
password TEXT)")
        cursor.execute("SELECT * FROM `member` WHERE `username` = 'admin'
AND `password` = 'admin'")
        if cursor.fetchone() is None:
            cursor.execute("INSERT INTO `member` (username, password)
VALUES('admin', 'admin')")
            conn.commit()
    def Login(event=None):
        Database()
        if USERNAME.get() == "" or PASSWORD.get() == "":
            lbl_text.config(text="Please complete the required field!", fg="red")
        else:
            cursor.execute("SELECT * FROM `member` WHERE `username` = ?
AND `password` = ?", (USERNAME.get(), PASSWORD.get()))
            if cursor.fetchone() is not None:
                HomeWindow()
                USERNAME.set("")
                PASSWORD.set("")
                lbl_text.config(text="")
            else:
                lbl_text.config(text="Invalid username or password", fg="red")
                USERNAME.set("")
                PASSWORD.set("")
        cursor.close()
        conn.close()


    #==============================BUTTON
WIDGETS===============================
    btn_login = Button(Form, text="Login", width=45, command=Login)
    btn_login.grid(pady=25, row=3, columnspan=2)
    btn_login.bind('<Return>', Login)


    def HomeWindow():
        global Home
        root.withdraw()
```

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

    def Back():
        Home.destroy()
   root.deiconify()
```

TESTING

8.1 TEST CASES:

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data
1	Functional	Registration	Registration through the form by Filling in my details		1.Click on register 2.Fill the registration form 3.click Register	
2	UI	Generating OTP	Generating the otp for further process		1.Generating of OTP number	
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
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
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

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data
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	
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	
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	
4	Functional	Redirection	user can be redirected to the selected		1.After payment the usre will be redirected to the previous page	

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data
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	
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	
3	Functional	Remainder notification	a user, I get remainders about my journey A day before my actual journey		1.user can get reminder nofication	
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	

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data
1	Functional	Ticket cancellation	user can cancel my tickets there's any Change of plan		1.tickets to be cancelled	
2	UI	Raise queries	user can raise queries through the query box or via mail.		1,raise the queries	
3	Functional	Answer the queries	user will answer the questions/doubts Raised by the customers.		1.answer the queries	
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	

8.2 USER ACCEPTANCE TESTING:

Test Case #2	Select arrival railway station
Time/Date	15:30 / 15th February 2009
Actions	1. Click the drop down list next to 'To' 2. Select the station 'Solihull'
Expected Result	Application displays 'Solihull' station as selected in the drop down list next to the text 'To'
Result	(Enter actual result here)
Pass/Fail	(Enter Test pass or fail here)

RESULTS

9.1 PERFORMANCE METRICS:

ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- Better organized
- Suitable for longer journey
- Promotes tourism
- Generates employment

DISADVANTAGES:

- Highly inflexible
- Costly if the routes are small
- Trains parts are pretty old
- Unsuitable for perishable and fragile items.

CONCLUSION

The railway industry is on its way to integrate predictive maintenance and Big Data. Recent advancements in sensors and condition monitoring technologies have led to continuous data collection and evaluation, significantly minimising the number and cost of unscheduled maintenance.

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 4.0, railway companies can now ensure that they are prepared to avoid the surprise of equipment downtime.

FUTURE SCOPE

The contribution of railways to sustainability is to provide efficient services, transferring traffic from roads and airplanes offering a real alternative to less sustainable transport modes. Rail is a vital part of the solution to the global challenge of climate change.

These track improvements could include junction rearrangements, curve easing, deviations, passing loops and level crossing removals. There are also opportunities for new technology and train options that may reduce journey times.

APPENDIX

SOURCE CODE:

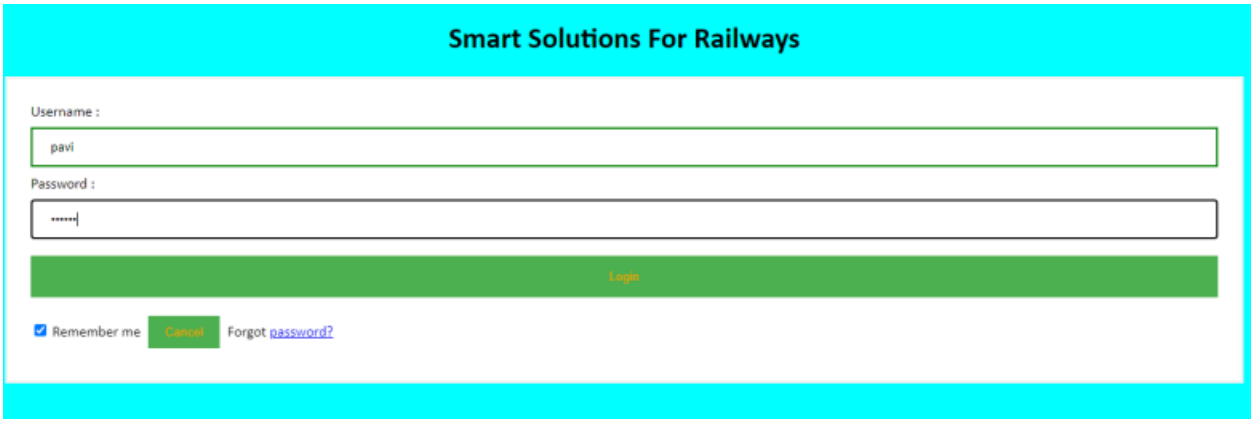
SPRINT 1:

login.py:

```
from tkinter import *
import sqlite3

root = Tk()
root.title("Python: Simple Login Application")
width = 400
height = 280
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))
root.resizable(0, 0)

#==============================VARIABLES=========================
USERNAME = StringVar()
PASSWORD = StringVar()

#==============================FRAMES============================
Top = Frame(root, bd=2,  relief=RIDGE)
Top.pack(side=TOP, fill=X)
Form = Frame(root, height=200)
Form.pack(side=TOP, pady=20)

#==============================LABELS============================
lbl_title = Label(Top, text = "Python: Simple Login Application", font=('arial', 15))
lbl_title.pack(fill=X)
lbl_username = Label(Form, text = "Username:", font=('arial', 14), bd=15)
lbl_username.grid(row=0, sticky="e")
```

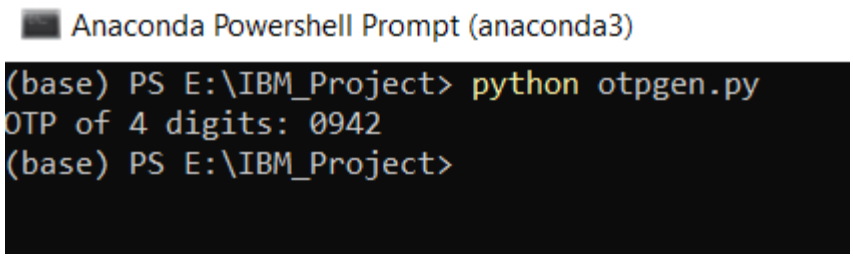
```
    lbl_password = Label(Form, text = "Password:", font=('arial', 14), bd=15)
    lbl_password.grid(row=1, sticky="e")
    lbl_text = Label(Form)
    lbl_text.grid(row=2, columnspan=2)

    #==============================ENTRY
WIDGETS==================================
    username = Entry(Form, textvariable=USERNAME, font=(14))
    username.grid(row=0, column=1)
    password = Entry(Form, textvariable=PASSWORD, show="*", font=(14))
    password.grid(row=1, column=1)




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

```
            PASSWORD.set("")
            lbl_text.config(text="")
        else:
            lbl_text.config(text="Invalid username or password", fg="red")
            USERNAME.set("")
            PASSWORD.set("")
    cursor.close()
    conn.close()


#==============================BUTTON WIDGETS==================================
btn_login = Button(Form, text="Login", width=45, command=Login)
btn_login.grid(pady=25, row=3, columnspan=2)
btn_login.bind('<Return>', Login)


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

def Back():
    Home.destroy()
    root.deiconify()
```



otpgen.py

```
# import library
import math, random

# function to generate OTP
def generateOTP() :

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

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

    return OTP

# Driver code
if __name__ == "__main__" :

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



otpveri.py:

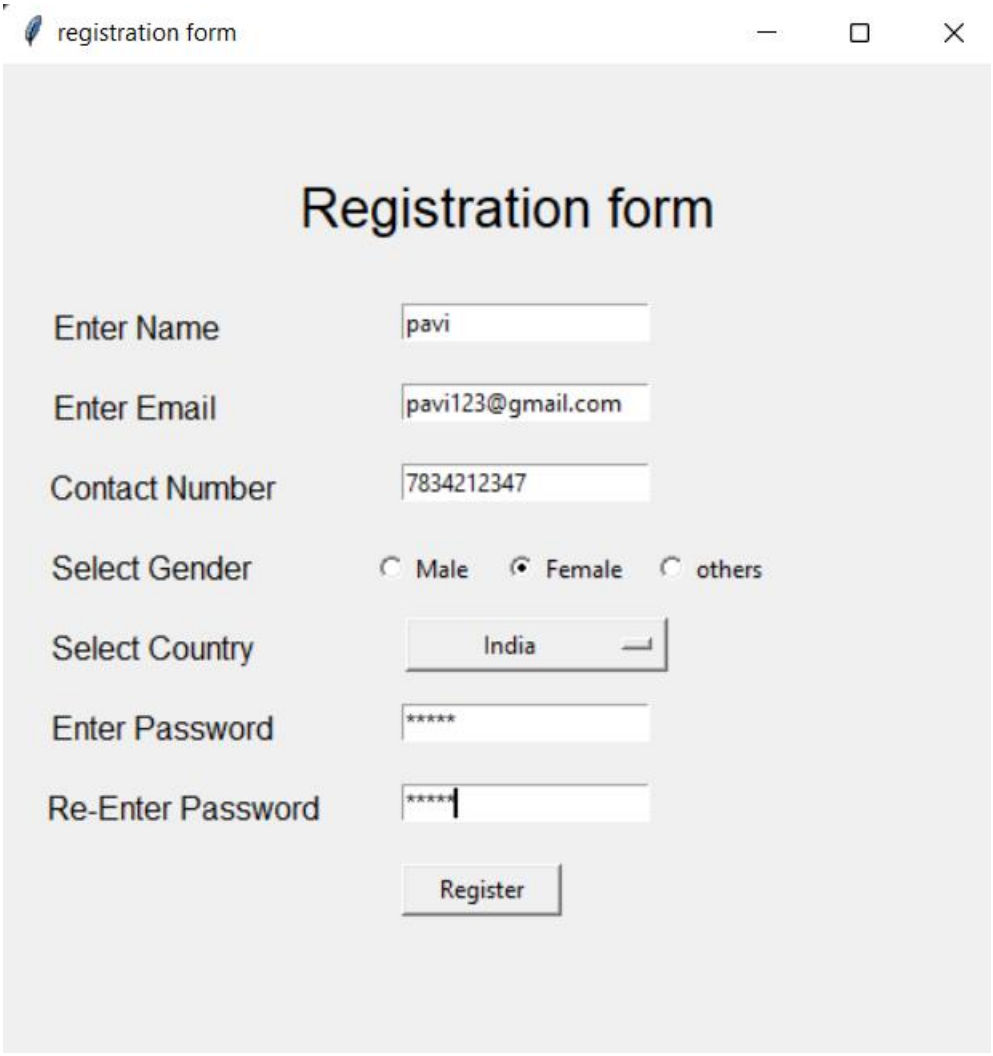
```
import os
import math
import random
import smtplib

digits = "0123456789"
OTP = ""

for i in range (6):
    OTP += digits[math.floor(random.random()*10)]

otp = OTP + " is your OTP"
message = otp
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()

emailid = input("Enter your email: ")
s.login("YOUR Gmail ID", "YOUR APP PASSWORD")
s.sendmail('&&&&&&',emailid,message)

a = input("Enter your OTP >>: ")
if a == OTP:
    print("Verified")
else:
    print("Please Check your OTP again")
```

reg.py:

```
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_des.py:

```
# 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+&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)
```

```
(base) PS E:\IBM_Project\Sprint_1> python start_des.py
Train between GAYA and PATNA

(base) PS E:\IBM_Project\Sprint_1>
```

SPRINT 2:

Booking.py:

```
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
```

```
(base) PS E:\IBM_Project\Sprint_2> python booking.py


Ticket Booking System

1.Check PNR status
2.Ticket Reservation

Enter your option : 2

Enter no. of Ticket you want : 2

Name : Pavi

Age  : 21

Male or Female : Female

Name : Binu

Age  : 21

Male or Female : Female

Did you forgot someone? y/n: n

Total Ticket :  2
Ticket :  1
Name :  Pavi
Age  :  21
Sex :  Female
Ticket :  2
Name :  Binu
Age  :  21
Sex :  Female
1.Check PNR status
2.Ticket Reservation

Enter your option : 1
Your PNR status is t3
(base) PS E:\IBM_Project\Sprint_2>
```

Payment.py:

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


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

    USERNAME_FIELD = "email"

    REQUIRED_FIELDS = ["first_name", "last_name"]

    email = models.EmailField(
        verbose_name="E-mail",
        unique=True
    )

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

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

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

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

```
    )

    objects = UserManager()

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

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


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

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

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

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

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

    class Meta:
```

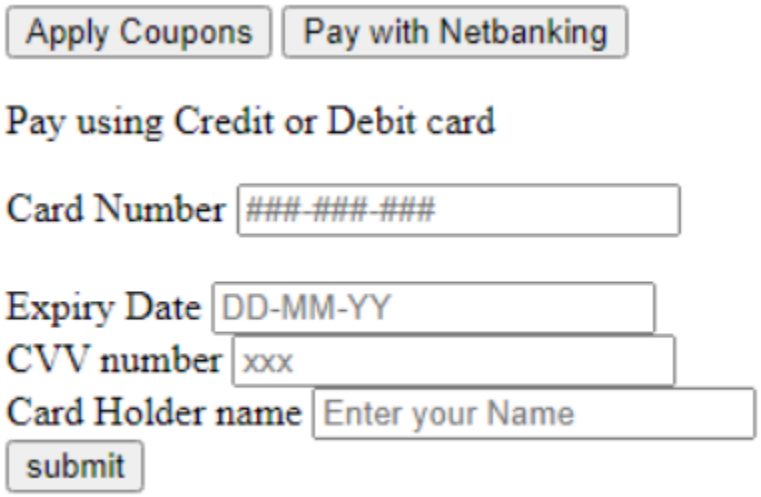
```
        abstract = True


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

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

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

    def __str__(self):
        return self.user.email

    class Meta:

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

user1.set_unusable_password()

group_name = "employer"

_profile1, _ = UserProfile.objects.get_or_create(
    user=user1,
    date_of_birth=datetime.now() - timedelta(days=6600),
    group=GroupTypeChoices(group_name).name,
```

```
    address="Myśliwska 14",
    postal_code="15-569",
    phone_number="+48100200300",
)

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

user2.set_unusable_password()

group_name = "employee"

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

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

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

mcc_code, url = "1520", "https://www.softserveinc.com/"

response_ca = stripe.Account.create()
    type="custom",
```

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

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

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

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

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

individual = {
```

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


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

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

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

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

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

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

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

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

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

stripe.PaymentIntent.cancel(payment_intent.id)


        unique_together = ("user", "group")
```



redirect.py:

```
import logging

import attr
from flask import Blueprint, flash, redirect, request, url_for
```

```
from flask.views import MethodView
from flask_babelplus import gettext as _
from flask_login import current_user, login_required
from pluggy import HookimplMarker

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

    decorators = [login_required]

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

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

seatsbook.py:

```
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
```

```
(base) PS E:\IBM_Project\Sprint_2> python seatsbook.py
10
7
0
(base) PS E:\IBM_Project\Sprint_2>
```

SPRINT 3:

ticketgen.py:

```
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
```

confirmation.py:

```
# import module
import requests
from bs4 import BeautifulSoup
import pandas 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 is booked successfully

Happy journey!!!

Latitude: 13.0261299
Longitude: 80.223446

gpstrack.py:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw

data_path = 'data.csv'
data = pd.read_csv(data_path, names=['LATITUDE', 'LONGITUDE'], sep=',')
gps_data = tuple(zip(data['LATITUDE'].values, data['LONGITUDE'].values))

image = Image.open('map.png', 'r')  # Load map image.
img_points = []
for d in gps_data:
    x1, y1 = scale_to_img(d, (image.size[0], image.size[1]))  # Convert GPS
coordinates to image coordinates.
```

```
        img_points.append((x1, y1))
    draw = ImageDraw.Draw(image)
    draw.line(img_points, fill=(255, 0, 0), width=2)  # Draw converted records to
the map image.

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

    fig, axis1 = plt.subplots(figsize=(10, 10))
    axis1.imshow(plt.imread('resultMap.png'))  # Load the image to matplotlib plot.
    axis1.set_xlabel('Longitude')
    axis1.set_ylabel('Latitude')
    axis1.set_xticklabels(x_ticks)
    axis1.set_yticklabels(y_ticks)
    axis1.grid()
    plt.show()
```

notification.py:

```
import pyttsx3
from plyer import notification
import time


# Speak method
def Speak(self, audio):

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

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

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

```
    engine.say(audio)
    engine.runAndWait()


def Take_break():

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

    if "yes" in question:
        Speak("Starting Sir")

    if "no" in question:
        Speak("We will automatically start after 5 Mins Sir.")
        time.sleep(5*60)
        Speak("Starting Sir")

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

        # For 45 min the will be no notification but
        # after 45 min a notification will pop up.
        time.sleep(0.5*60)

        Speak("Please Take a break Sir")

        notification.notify(title="Break Notification",
        message="Please do use your device after sometime as you have"
        "been continuously using it for 45 mins and it will affect your eyes",
        timeout=10)


# Driver's Code
```

```
if __name__ == '__main__':
    Take_break()
```

SPRINT 4:

ansqueries.py:

```
import email, smtplib, ssl

from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

subject = "An email with attachment from Python"
body = "This is an email with attachment sent from Python"
sender_email = "my@gmail.com"
receiver_email = "your@gmail.com"
password = input("Type your password and press enter:")

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

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

filename = "document.pdf"  # In same directory as script

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

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

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

# Add attachment to message and convert message to string
message.attach(part)
text = message.as_string()

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

feedinfo.py:

```
# Python program to find PNR
# status using RAILWAY API

# import required modules
import requests, json

# Enter API key here
api_key = "Your_API_key"

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

# Enter valid pnr_number
pnr_number = "6515483790"

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

# get method of requests module
# return response object
```

```
response_ob = requests.get(complete_url)

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

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

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

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

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

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

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

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

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

    # store the value or data of "doj" key
    # in variable date_of_journey variable
```

```
date_of_journey = result["doj"]

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

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

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

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

# looping through passenger list
for passenger in passengers_list:

        # store the value or data
        # of "no" key in variable
        passenger_num = passenger["no"]

        # store the value or data of
        # "current_status" key in variable
        current_status = passenger["current_status"]

        # store the value or data of
        # "booking_status" key in variable
        booking_status = passenger["booking_status"]
```

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

else:
    print("Record Not Found")
raisequeries.py:
import smtplib, ssl
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

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

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

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

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

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

# Create secure connection with server and send email
context = ssl.create_default_context()
with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
    server.login(sender_email, password)
    server.sendmail(
        sender_email, receiver_email, message.as_string()
    )
```

ticketcanc.py:

```
from pickle import load,dump
import time
import random
import os
class tickets:
    def __init__(self):
        self.no_ofac1stclass=0
        self.totaf=0
        self.no_ofac2ndclass=0
        self.no_ofac3rdclass=0
        self.no_ofsleeper=0
        self.no_oftickets=0
        self.name=''
        self.age=''
        self.resno=0
        self.status=''
    def ret(self):
        return(self.resno)
    def retname(self):
        return(self.name)
```

```
def display(self):
    f=0
    fin1=open("tickets.dat","rb")
    if not fin1:
        print "ERROR"
    else:
        print
        n=int(raw_input("ENTER PNR NUMBER : "))
        print "\n\n"
        print ("FETCHING DATA . . .".center(80))
        time.sleep(1)
        print
        print('PLEASE WAIT...!!'.center(80))
        time.sleep(1)
        os.system('cls')
        try:
            while True:
                tick=load(fin1)
                if(n==tick.ret()):
                    f=1
                    print "="*80
                    print("PNR STATUS".center(80))
                    print"="*80
                    print
                    print "PASSENGER'S NAME :",tick.name
                    print
                    print "PASSENGER'S AGE :",tick.age
                    print
                    print "PNR NO :",tick.resno
                    print
                    print "STATUS :",tick.status
                    print
                    print "NO OF SEATS BOOKED : ",tick.no_oftickets
                    print
        except:
            pass
        fin1.close()
        if(f==0):
            print
            print "WRONG PNR NUMBER..!!"
```

```
            print
    def pending(self):
        self.status="WAITING LIST"
        print "PNR NUMBER :",self.resno
        print
        time.sleep(1.2)
        print "STATUS = ",self.status
        print
        print "NO OF SEATS BOOKED : ",self.no_oftickets
        print
    def confirmation (self):
        self.status="CONFIRMED"
        print "PNR NUMBER : ",self.resno
        print
        time.sleep(1.5)
        print "STATUS = ",self.status
        print
    def cancellation(self):
        z=0
        f=0
        fin=open("tickets.dat","rb")
        fout=open("temp.dat","ab")
        print
        r= int(raw_input("ENTER PNR NUMBER : "))
        try:
            while(True):
                tick=load(fin)
                z=tick.ret()
                if(z!=r):
                    dump(tick,fout)
                elif(z==r):
                    f=1
        except:
            pass
        fin.close()
        fout.close()
        os.remove("tickets.dat")
        os.rename("temp.dat","tickets.dat")
        if (f==0):
            print
```

```
        print "NO SUCH RESERVATION NUMBER FOUND"
        print
        time.sleep(2)
        os.system('cls')
    else:
        print
        print "TICKET CANCELLED"
        print"RS.600 REFUNDED...."
def reservation(self):
    trainno=int(raw_input("ENTER THE TRAIN NO:"))
    z=0
    f=0
    fin2=open("tr1details.dat")
    fin2.seek(0)
    if not fin2:
        print "ERROR"
    else:
        try:
            while True:
                tr=load(fin2)
                z=tr.gettrainno()
                n=tr.gettrainname()
                if (trainno==z):
                    print
                    print "TRAIN NAME IS : ",n
                    f=1
                    print
                    print "-"*80
                    no_ofac1st=tr.getno_ofac1stclass()
                    no_ofac2nd=tr.getno_ofac2ndclass()
                    no_ofac3rd=tr.getno_ofac3rdclass()
                    no_ofsleeper=tr.getno_ofsleeper()
                if(f==1):
                    fout1=open("tickets.dat","ab")
                    print
                    self.name=raw_input("ENTER THE PASSENGER'S NAME ")
                    print
                    self.age=int(raw_input("PASSENGER'S AGE : "))
                    print
```

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

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



    def menu():
        tr=train()
        tick=tickets()
        print
        print "WELCOME TO PRAHIT AGENCY".center(80)
        while True:
            print
            print "="*80
            print " \t\t\t\t  RAILWAY"
            print
            print "="*80
            print
            print "\t\t\t1. **UPDATE TRAIN DETAILS."
            print
            print "\t\t\t2. TRAIN DETAILS. "
            print
            print "\t\t\t3. RESERVATION OF TICKETS."
            print
            print "\t\t\t4. CANCELLATION OF TICKETS. "
            print
            print "\t\t\t5. DISPLAY PNR STATUS."
            print
            print "\t\t\t6. QUIT."
            print"** - office use....."
            ch=int(raw_input("\t\t\tENTER YOUR CHOICE : "))
            os.system('cls')
            print
"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\tLOADING. .",
            time.sleep(1)
            print ("."),
            time.sleep(0.5)
            print (".")
```

```
            time.sleep(2)
            os.system('cls')
            if ch==1:
               j="*****"
               r=raw_input("\n\n\n\n\n\n\n\n\n\n\n\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"
               else:
                  try:
                     while True:
                        print"*"*80
                        print"\t\t\t\tTRAIN DETAILS"
                        print"*"*80
```

```
                    print
                    tr=load(fin)
                    tr.output()



                    raw_input("PRESS ENTER TO VIEW NEXT TRAIN
DETAILS")
                    os.system('cls')
                except EOFError:
                    pass
            elif ch==3:
                print'='*80
                print "\t\t\t\tRESERVATION OF TICKETS"
                print'='*80
                print
                tick.reservation()
            elif ch==4:
                print"="*80
                print"\t\t\t\tCANCELLATION OF TICKETS"
                print
                print"="*80
                print
                tick.cancellation()
            elif ch==5:
                print "="*80
                print("PNR STATUS".center(80))
                print"="*80
                print
                tick.display()
            elif ch==6:
                quit()

            raw_input("PRESS ENTER TO GO TO BACK MENU".center(80))
            os.system('cls')
             menu()
```

PROJECT DEMO LINK:

https://www.youtube.com/watch?v=E5eG8pQdMys