**AP LAB\_3**

**~ Prof. Ishan Maheta**

✅ **Title 3: Railway Ticket Reservation System**

🎯 **Objective:**

* To implement a Python program that simulates a basic railway ticket reservation system.
* To load train and passenger data from CSV files, check seat availability, book tickets, update train data, calculate fare, and generate reports.

📘 **Task Description:**  
You are tasked with developing a **railway ticket reservation system** for a busy rail network. The system must:

1. Load Train Data from trains.csv (Train ID, Train Name, Source, Destination, Total Seats, Distance).
2. Load Passenger Data from passengers.csv (Passenger Name, Train ID, Number of Tickets).
3. Check Seat Availability and confirm/reject bookings.
4. Calculate Fare based on distance:
   * ≤ 500 km → ₹1.5 per km
   * 501–1000 km → ₹1.2 per km
   * 1000 km → ₹1.0 per km
5. Update seat availability after successful bookings.
6. Generate Reports:
   * **Report 1:** Train details (name, route, available seats).
   * **Report 2:** Revenue summary (successful/failed bookings, fare collected).
7. Handle errors (invalid Train IDs, insufficient seats, invalid ticket numbers).

💻 **Python Code:**

import numpy as np

import pandas as pd

from typing import Literal

import traceback

from reportlab.lib.pagesizes import A4

from reportlab.pdfgen import canvas

# Centralize file paths here

TRAINS\_FILE = r"D:\Ghanu Study\Sem - 5\Advance Python\Lab\Code\files\trains.csv"

PASSENGERS\_FILE = r"D:\Ghanu Study\Sem - 5\Advance Python\Lab\Code\files\passengers.csv"

# Function to print errors

def print\_error(e: Exception) -> None:

    tb = traceback.extract\_tb(e.\_\_traceback\_\_)

    for filename, line, funcname, text in tb:

        print("-" \* 100)

        print(f"Error -> {e}")

        print(f"File -> {filename}")

        print(f"Function -> {funcname}, Line -> {line}")

        print(f"Code -> {text}\n")

        print("-" \* 100)

# Fare calculation based on distance

def fare(distance):

    try:

        if distance <= 0:

            raise ValueError(f"Invalid distance {distance}")

        if distance <= 500:

            return distance \* 1.5

        elif distance <= 1000:

            return distance \* 1.2

        else:

            return distance \* 1.0

    except Exception as e:

        print\_error(e)

# Booking function

def book(name: str, passengers: pd.DataFrame, trains: pd.DataFrame) -> Literal[0, 1]:

    try:

        passenger\_index = np.where(passengers["Passenger Name"] == name)[0]

        train\_id = passengers.loc[passenger\_index, "Train ID"].iloc[0]

        num = passengers.loc[passenger\_index, "Number of Tickets"].iloc[0]

        if num <= 0:

            raise ValueError(f"Invalid number of tickets: {num}")

        if train\_id not in trains["Train ID"].values:

            raise ValueError(f"No train with ID {train\_id}")

        train\_index = np.where(trains["Train ID"] == train\_id)[0]

        available = trains.loc[train\_index, "Total Seats"].iloc[0]

        if num > available:

            raise ValueError(f"Not enough seats in train {train\_id}")

        trains.loc[train\_index, "Total Seats"] -= num

        passengers.loc[passenger\_index, "Fare"] = num \* trains.loc[train\_index, "Single Fare"].iloc[0]

        passengers.loc[passenger\_index, "Status"] = "Success"

        return 0

    except Exception as e:

        print\_error(e)

        passengers.loc[passenger\_index, "Status"] = "Failed"

        return 1

# Report 1: Train details

def print\_report1(trains: pd.DataFrame, filename: str) -> None:

    try:

        c = canvas.Canvas(filename, pagesize=A4)

        width, height = A4

        margin = 60

        c.setFont("Helvetica-Bold", 20)

        c.drawString(100, height - 30, "Details of Trains")

        y = height - margin

        c.setFont("Helvetica", 12)

        for i, train in trains.iterrows():

            c.drawString(50, y, f"{i+1}) {train['Train Name']}")

            y -= 20

            c.drawString(70, y, f"Source -> {train['Source Station']}")

            y -= 20

            c.drawString(70, y, f"Destination -> {train['Destination Station']}")

            y -= 20

            c.drawString(70, y, f"Seats Available -> {train['Total Seats']}")

            y -= 30

            if y < margin:

                c.showPage()

                y = height - margin

        c.save()

    except Exception as e:

        print\_error(e)

# Report 2: Revenue and booking summary

def print\_report2(passengers: pd.DataFrame, trains: pd.DataFrame, filename: str) -> None:

    try:

        c = canvas.Canvas(filename, pagesize=A4)

        width, height = A4

        margin = 60

        c.setFont("Helvetica-Bold", 20)

        c.drawString(100, height - 30, "Fare Collection Report")

        y = height - margin

        c.setFont("Helvetica", 12)

        for i, train in trains.iterrows():

            train\_id = train["Train ID"]

            success = np.where((passengers["Train ID"] == train\_id) & (passengers["Status"] == "Success"))[0]

            failed = np.where((passengers["Train ID"] == train\_id) & (passengers["Status"] == "Failed"))[0]

            c.drawString(50, y, f"{i+1}) {train['Train Name']}")

            y -= 20

            c.drawString(70, y, f"Successful Bookings -> {success.shape[0]}")

            y -= 20

            c.drawString(70, y, f"Failed Bookings -> {failed.shape[0]}")

            y -= 20

            c.drawString(70, y, f"Total Fare Collected -> {np.sum(passengers.loc[success, 'Fare'])}")

            y -= 30

            if y < margin:

                c.showPage()

                y = height - margin

        c.save()

    except Exception as e:

        print\_error(e)

# Main

def main():

    try:

        # Always load from files folder

        trains = pd.read\_csv(TRAINS\_FILE)

        passengers = pd.read\_csv(PASSENGERS\_FILE)

        passengers["Status"] = "Pending"

        passengers["Fare"] = -1

        trains["Single Fare"] = trains["Distance"].apply(fare)

        passengers["Passenger Name"].apply(book, args=(passengers, trains,))

        print\_report1(trains, "report1.pdf")

        print\_report2(passengers, trains, "report2.pdf")

        print("✅ Reports generated successfully!")

    except Exception as e:

        print\_error(e)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

🧪 **Sample Input Files:**

📄 trains.csv

Train ID,Train Name,Source Station,Destination Station,Total Seats,Distance

T001,Shatabdi Express,Mumbai,Delhi,300,1384

T002,Rajdhani Express,Delhi,Kolkata,350,1530

T003,Duronto Express,Chennai,Bangalore,200,350

T004,Garib Rath,Ahmedabad,Jaipur,400,650

T005,Jan Shatabdi,Pune,Nagpur,250,820

📄 passengers.csv

Passenger Name,Train ID,Number of Tickets

Aarav,T001,2

Isha,T002,4

Rohan,T003,1

Meera,T004,3

Vikram,T005,2000

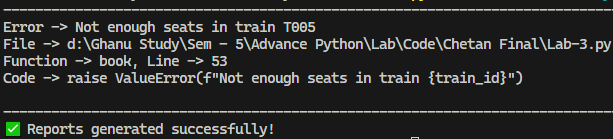
Priya,T001,1

Ananya,T003,4

Siddharth,T004,1

Pooja,T005,3

🧪 **Sample Output (Reports):**



✅ **Conclusion:**  
This experiment successfully simulates a **railway reservation system** using Python. It demonstrates reading structured data from CSV files, handling bookings with validations, calculating fares dynamically, updating seat availability, and generating professional PDF reports. With error handling and modular design, it efficiently automates essential parts of railway operations.