Experiment No: 3

Railway Ticket Reservation System using Python

Objective:

To implement a Python program that simulates a basic railway ticket reservation system by loading train and passenger data from CSV files, checking seat availability, booking tickets, updating train data, calculating fare, and generating reports.

Task Description:

You are tasked with developing a railway ticket reservation system for a busy rail network. The system must provide the following core functionalities:

Load Train Data: The program should read the train data from a CSV file named "trains.csv." Each row in the CSV file represents a train with the following information:

- Train ID (a unique alphanumeric code)
- Train Name
- Source Station
- Destination Station
- Total Seats (total number of seats available on the train)

Load Passenger Data: The program should read the passenger data from a CSV file named "passengers.csv." Each row in the CSV file represents a passenger with the following information:

- Passenger Name
- Train ID (the ID of the train the passenger wants to book a ticket on)
- Number of Tickets (the number of tickets the passenger wants to book)

Check Seat Availability: Given the train ID and the number of tickets requested by a passenger, the program should check if there are enough seats available on the specified train for booking. If seats are available, the booking should be confirmed, and the total fare for the booking should be calculated as per the fare rules (you can define fare rules based on distance, class, etc.).

Update Seat Availability: After confirming the booking, the program should update the seat availability for the corresponding train.

Generate Reports:

1. Report 1: The program should generate a report showing the details of all the trains, including their names, source stations, destination stations, and the total number of seats available on each train.
2. Report 2: The program should generate a report showing the total revenue earned from each train based on the total number of confirmed bookings and their respective fares.

Handle Errors: The program should handle various types of errors gracefully, such as invalid train IDs, invalid passenger names, insufficient seats, etc., and provide appropriate error messages.

Note: You can assume that the passenger data in "passengers.csv" will not exceed the available seats on any train. You can design the fare rules based on your preference and mention them clearly in the program.

Steps to Perform the Program:

1. Import required libraries:
    - csv, os, sys
2. Read train data from trains.csv:
    - Store each train's details in a dictionary using Train ID as the key.
3. Read passenger data from passengers.csv:
    - Loop through each passenger request and perform booking validation.
4. Check seat availability:
    - Compare requested tickets with available seats in the train record.
5. Calculate fare:
    - Apply your predefined fare rule (e.g., ₹500 per seat or based on class/distance).
6. Update seat count:
    - Subtract the booked tickets from the total available seats of that train.
7. Track total revenue:
    - Maintain a dictionary to track total earnings per train.
8. Generate Reports:
    - Report 1: Train-wise details and seat availability.
    - Report 2: Train-wise total revenue generated from confirmed bookings.
9. Handle errors:
    - Use try-except blocks to manage:
        - Invalid or missing data
        - Inconsistent CSV format
        - Invalid train IDs
10. Use functions or modular design:
    - Separate functionality using functions like load_trains(), book_ticket(), generate_report() for cleaner code.