Presentation

By Chhatarpal Yadav

2501350075

Introduction

Railways are one of the most commonly used modes of transportation. Managing ticket reservations efficiently ensures convenience for passengers and reduces human error. This project simulates a Railway Ticket Reservation System, where users can view available trains, book tickets, and cancel them. It demonstrates computational thinking by applying abstraction, decomposition, and pattern recognition to model a real-world system into structured algorithms and working code

Problem Analysis

- a) Abstraction We focus only on essential details and ignore real-world complexities like online payments or seat classes. Essential elements: Train details (Train No., Name, Source, Destination, Seats, Fare) User information (Name, Ticket ID) Operations: View Trains, Book Ticket, Cancel Ticket, View Booked Tickets Ignored elements: Payment gateway Real-time train schedule Multi-user concurrency

- b) Decomposition Subsystem Description 1. Display Module Show available trains and their details 2. Booking Module Take user input, check seat availability, confirm booking 3. Cancellation Module Cancel a booked ticket and update seats 4. Storage Module Store ticket data using JSON file for persistence 5. Main Menu Connects all modules and handles user interactio

- c) Pattern Recognition Booking and cancellation both follow a “check-update-save” pattern. Similar to other reservation systems (e.g., airline or movie ticket booking). Data is always stored and retrieved in key-value pairs. Common use of input validation and looping menus in all interactive systems

Flowchart

START

↓

Show Main Menu

↓

1. View Trains
2. Book Ticket
3. View Booked Tickets
4. Exit

↓

User enters a choice

↓

View Trains	Book Ticket	View Tickets
Display list of trains	Ask name & train number Check seats If available → Book ticket	Show all booked tickets

↓

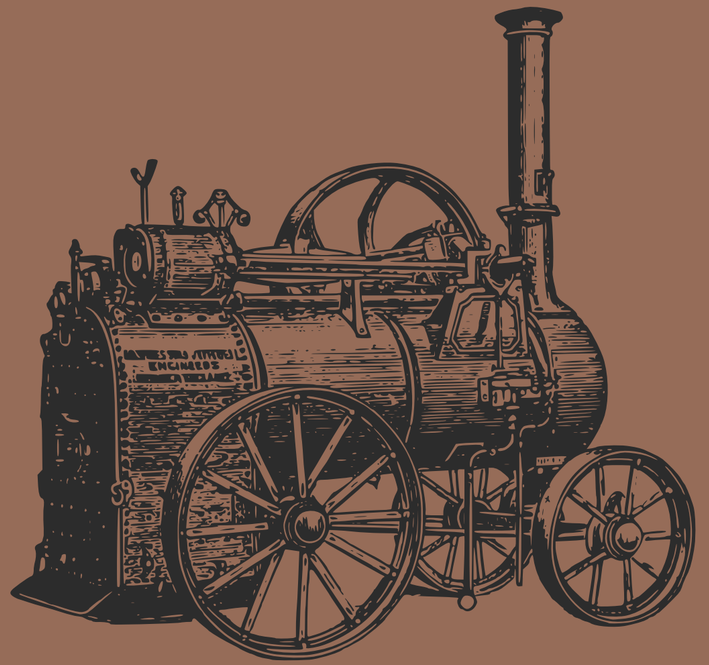
Return to Main Menu

↓

Exit
Display Thank You msg

↓

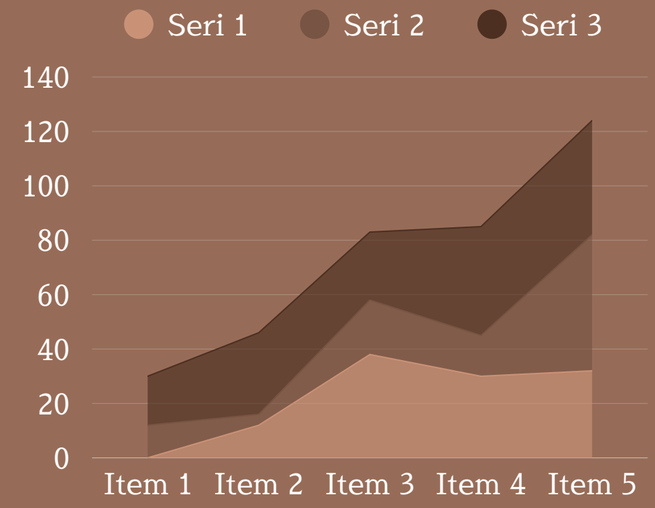
END



Pseudocode

```
BEGIN
  DISPLAY menu
  ASK user for choice

  IF choice = 1 THEN
     SHOW available trains
  ELSE IF choice = 2 THEN
     ASK user name and train number
     BOOK ticket if seat available
  ELSE IF choice = 3 THEN
     SHOW booked tickets
  ELSE IF choice = 4 THEN
     EXIT program
  ENDIF
END
```



python implementation

```
# Railway Ticket Reservation System (Beginner Version)

# Predefined train list
trains = {
101: {"name": "Rajdhani Express", "from": "Delhi", "to": "Mumbai", "seats": 3, "fare": 1200},
102: {"name": "Shatabdi Express", "from": "Delhi", "to": "Chandigarh", "seats": 3, "fare": 800}
}

# Empty list to store bookings
booked_tickets = []

while True:
print("\n===== RAILWAY TICKET BOOKING SYSTEM =====")
print("1. View Trains")
print("2. Book Ticket")
print("3. View Booked Tickets")
print("4. Exit")

choice = input("Enter your choice: ")

if choice == "1":
print("\nAvailable Trains:")
for num, info in trains.items():
print(f"{num}: {info['name']} ({info['from']} → {info['to']}) | Seats: {info['seats']} | Fare: ₹{info['fare']}")

elif choice == "2":
name = input("Enter your name: ")
train_no = int(input("Enter train number to book: "))
if train_no in trains and trains[train_no]["seats"] > 0:
trains[train_no]["seats"] -= 1
booked_tickets.append({"name": name, "train": trains[train_no]["name"]})
print("\n✅ Ticket booked successfully!")
else:
print("\n❌ No seats available or invalid train number.")

elif choice == "3":
print("\nBooked Tickets:")
if not booked_tickets:
print("No tickets booked yet.")
else:
for ticket in booked_tickets:
print(f"Name: {ticket['name']} | Train: {ticket['train']}")

elif choice == "4":
print("🚆 Thank you for using the system!")
break

else:
print("Invalid choice, please try again.")
```

Reflection

CHALLENGES FACED DESIGNING A PROPER DATA STRUCTURE FOR TRAIN AND TICKET STORAGE. MANAGING DATA PERSISTENCE USING FILES. AVOIDING SEAT DUPLICATION DURING BOOKING. INSIGHTS GAINED LEARNED HOW TO APPLY COMPUTATIONAL THINKING IN A REAL-WORLD SYSTEM. UNDERSTOOD HOW ABSTRACTION SIMPLIFIES COMPLEX SYSTEMS. IMPROVED UNDERSTANDING OF PYTHON FILE HANDLING AND STRUCTURED PROGRAMMING. POTENTIAL IMPROVEMENTS ADD SEAT CLASS (SLEEPER, AC, ETC.) IMPLEMENT FARE CALCULATION BASED ON DISTANCE. USE A DATABASE (LIKE SQLITE) FOR BETTER SCALABILITY. ADD A GUI (TKINTER OR WEB INTERFACE)

Thank You