DEPARTMENT OF COMPUTER ENGINEERING

A. Y. 2025-26 Semester-I

MINI PROJECT REPORT

Subject: object oriented programming

Group No. : SY2 G5

Title of the Project: Railway Reservation System

Group Members:

Sr. No.	PRN No.	Name of the Student
	B24CE1064	Aman Gotad
	B24CE1069	Shruti Deshmukh
	B24CE1079	Srushti Salgar
	B24CE1085	Swapnil Waghmare

Object Oriented Programming Used:

The following and OOP concepts are used in the project:

- Classes and Objects – Used to represent real world entities like Train, Passenger, and Ticket.
- Inheritance – Passenger class inherits from Person to reuse attributes (name, age).
- Vectors (from STL) – Used to store trains, tickets, and passenger lists dynamically.
- Strings – To store names, station details, and times.
- File Handling – For storing booking data permanently in a text file.
- Pointers and Dynamic Memory Allocation – Used to create passengers and tickets at runtime.
- Loops and Conditional Statements – For menu navigation and decision-making.

Mini-Project Idea:

The Railway Reservation System simulates the process of booking, viewing, and canceling train tickets through an interactive console-based C++ program.

Input:

- Train details (train code, name, route, timings)
- Passenger details (name, age, number of passengers)
- PNR number for cancellations
- Menu choices from the user

Output:

- Displays available trains and booking confirmation
- Shows PNR, seat numbers, and passenger details
- Displays current bookings and train status
- Confirms ticket cancellations

Techniques Used:

- Object-Oriented Programming (Classes, Inheritance, Encapsulation)
- File Handling for storing booking information
- Dynamic memory allocation and destructors
- Menu-driven interaction using loops and conditions

Program:

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
using namespace std;

// Class having train info
class Train {
public:
    int code;              // train code
    string name;           // Train name
    string from, to;       // Source and destination stations
    string departure, arrival; // Departure and arrival times
    int totalSeats;        // Total capacity of the train
    int seatsBooked;       // Seats already booked
    int price;             // Ticket price per passenger

    Train(int c, string n, string f, string t, string dep, string arr, int seats, int p) {
        code = c; name = n; from = f; to = t;
        departure = dep; arrival = arr;
        totalSeats = seats;
        seatsBooked = 0;      // Initially no seats booked
        price = p;
    }

    // Display basic info of the train
    void showShort() {
        cout << code << " | " << name << " | " << from << " -> " << to
             << " | Dep: " << departure << " | Arr: " << arrival << "\n";
```

```
    }

    // Display full info
    void showFull() {
        showShort();
        cout << "Total Seats: " << totalSeats << " | Booked: " << seatsBooked
             << " | Available: " << totalSeats - seatsBooked
             << " | Price: " << price << " Rs\n";
    }
};

// Class having person info
class Person {
public:
    string name;
    int age;

    Person(string n, int a) {
        name = n; age = a;
    }
};

// Passenger class derived from Person
class Passenger : public Person {
public:
    int seatNo;
    int pnr;        // Unique passenger reservation number

    Passenger(string n, int a, int seat, int p) : Person(n, a) {
        seatNo = seat;
        pnr = p;
    }

    // Display passenger details
    void showInfo() {
        cout << "PNR: " << pnr << " | Name: " << name << " | Age: " << age
             << " | Seat No: " << seatNo << " |\n";
    }
};

// Ticket class to group multiple passengers for a train booking
class Ticket {
public:
    int trainIndex;              // Index of train booked from trains vector
    vector<Passenger*> passengers; // pointer to objects of passanger class (used for passenger list)
```

```
    Ticket(int index) {
        trainIndex = index;
    }

    ~Ticket() {
        for (int i = 0; i < passengers.size(); i++) {
            delete passengers[i];
        }
    }

    // Display full ticket info: train details + passenger list
    void showDetails(const Train& train) {
        cout << "-------------------------\n";
        cout << "Train: " << train.name << " (" << train.code << ")\n";
        cout << "Route: " << train.from << " -> " << train.to << "\n";
        cout << "Dep: " << train.departure << " | Arr: " << train.arrival << "\n";
        cout << "Passengers:\n";
        for (int i = 0; i < passengers.size(); i++) {
            passengers[i]->showInfo();
        }
        cout << "-------------------------\n";
    }
};

//function to find train index by train code
int findTrain(vector<Train>& trains, int code) {
    for (int i = 0; i < trains.size(); i++) {
        if (trains[i].code == code) return i;
    }
    return -1;
}

int main() {
    vector<Train> trains = {
        Train(111, "Deccan Express", "Pune", "Mumbai", "10:30", "14:10", 100, 1500),
        Train(523, "Konkan Express", "Goa", "Pune", "08:00", "17:40", 80, 1200),
        Train(987, "Shatabdi", "Mumbai", "Delhi", "17:00", "07:00", 120, 2000),
        Train(245, "Duronto Express", "Nagpur", "Mumbai", "21:00", "10:45", 110, 1800),
        Train(412, "Intercity Express", "Ahmedabad", "Surat", "06:15", "09:20", 90, 1000),
        Train(334, "Rajdhani Express", "Delhi", "Kolkata", "16:45", "10:15", 130, 2500),
        Train(278, "Garib Rath", "Bangalore", "Chennai", "07:30", "13:00", 95, 900),
        Train(395, "Jan Shatabdi", "Lucknow", "Varanasi", "05:40", "10:30", 85, 800),
        Train(409, "Goa Express", "Delhi", "Goa", "15:00", "12:00", 100, 2200),
        Train(516, "Tejas Express", "Mumbai", "Goa", "06:00", "14:00", 120, 2000),
        Train(628, "Chennai Mail", "Chennai", "Kolkata", "20:00", "09:30", 100, 2100),
        Train(742, "Punjab Mail", "Mumbai", "Amritsar", "19:00", "18:15", 110, 2300),
```

```
    Train(863, "Tamil Nadu Express", "Delhi", "Chennai", "22:00", "18:00", 125, 2600)
  };

  vector<Ticket*> bookings; // List of all active tickets stored as pointers
  int pnrCounter = 10000;   // Generates unique PNR numbers for passengers

  cout << "=== RAILWAY RESERVATION SYSTEM ===\n";
  cout << "Available trains:\n";

  // Display all trains briefly for user reference
  for (int i = 0; i < trains.size(); i++) {
    trains[i].showShort();
  }

  int choice;
  do {
    // Display the menu options to the user
    cout << "\nMenu:\n1. Book Ticket\n2. View Bookings\n3. Cancel Ticket\n4. View Train Status\n5.
Exit\nEnter choice: ";
    cin >> choice;

    if (choice == 1) {
      // Booking flow

      cout << "Enter train code: ";
      int code; cin >> code;

      int tIndex = findTrain(trains, code);
      if (tIndex == -1) {
        cout << "Train not found!\n";
        continue;
      }

      Train &train = trains[tIndex]; // Reference to selected train to update its seats directly
      int available = train.totalSeats - train.seatsBooked;

      if (available <= 0) {
        cout << "No seats available on this train.\n";
        continue;
      }

      cout << "Enter number of passengers to book (max " << available << "): ";
      int pCount; //passanger count - pCount
      cin >> pCount;

      if (pCount <= 0 || pCount > available) {
```

```
            cout << "NO Sufficient seats.\n";
            continue;
        }

        Ticket* ticket = new Ticket(tIndex); // Create new ticket pointer for this booking

        // Opening file in append mode to save booking data (tikctes)
        ofstream out("bookings.txt", ios::app);
        out << "Train: " << train.name << " (" << train.code << ")\n";
        out << "Route: " << train.from << " -> " << train.to << "\n";
        out << "Dep: " << train.departure << " | Arr: " << train.arrival << "\n";
        out << "Passengers:\n";

        for (int i = 0; i < pCount; i++) {
            string pname;
            int age;

            cout << "Passenger " << i+1 << " name: ";
            cin.ignore();
            getline(cin, pname);

            cout << "Passenger " << i+1 << " age: ";
            cin >> age;

            int seatNo = train.seatsBooked + 1;
            Passenger* p = new Passenger(pname, age, seatNo, pnrCounter++);
            ticket->passengers.push_back(p);
            train.seatsBooked++;       // updates new booking in train (means if booking is done,increments the no
of booked seats)

            cout << "Seat Confirmed for " << pname << ", Seat No: " << seatNo << ", PNR: " << p->pnr << "\n";

            // Write passenger info to file
            out << "PNR: " << p->pnr << " | Name: " << p->name << " | Age: " << p->age << " | Seat No: " << p-
>seatNo << " |\n";
        }
        out << "------------------------\n";
        out.close();

        bookings.push_back(ticket);
        cout << "Booking successful.\n";

    } else if (choice == 2) {
        // View all bookings saved in file

        ifstream in("bookings.txt");
```

```
    if (!in) {
        cout << "No bookings yet.\n";
    } else {
        string line;
        while (getline(in, line)) {
            cout << line << endl;
        }
        in.close();
    }

} else if (choice == 3) {
    // Cancel ticket flow
    cout << "Enter PNR number to cancel: ";
    int pnr; cin >> pnr;
    bool found = false;

    for (int i = 0; i < bookings.size(); i++) {
        Ticket* t = bookings[i];
        for (int j = 0; j < t->passengers.size(); j++) {
            if (t->passengers[j]->pnr == pnr) {
                found = true;
                Passenger* pass = t->passengers[j];
                int trainIdx = t->trainIndex;

                Train& train = trains[trainIdx];
                train.seatsBooked--;  // Release seat

                delete pass;        // Free passenger object memory
                t->passengers.erase(t->passengers.begin() + j);  // Remove from vector

                cout << "Cancelled ticket for " << pass->name << "\n";

                if (t->passengers.empty()) {
                    // Delete entire ticket if no passengers remain
                    delete t;
                    bookings.erase(bookings.begin() + i);
                    i--; // Fix index after erase
                }
                break;
            }
        }
        if (found) break;
    }

    if (!found) {
        cout << "PNR not found!\n";
```

```
            }

        } else if (choice == 4) {
            // View train booking status
            cout << "Enter Train Code to check status: ";
            int tc; cin >> tc;
            int index = findTrain(trains, tc);
            if (index == -1) {
                cout << "Train not found!\n";
            } else {
                trains[index].showFull();
            }

        } else if (choice == 5) {

            cout << "Exiting system...\n";
            for (int i = 0; i < bookings.size(); i++) {
                delete bookings[i];  // Calls Ticket destructor to delete passengers
            }
            bookings.clear();

        } else {
            cout << "Invalid choice, please try again.\n";
        }

    }
    while (choice != 5);

    return 0;
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Gotad\OneDrive\Desktop\oops mini project> g++ oop.cpp -o oop.exe
PS C:\Users\Gotad\OneDrive\Desktop\oops mini project> ./oop.exe
=== RAILWAY RESERVATION SYSTEM ===
Available trains:
111 | Deccan Express | Pune -> Mumbai | Dep: 10:30 | Arr: 14:10
523 | Konkan Express | Goa -> Pune | Dep: 08:00 | Arr: 17:40
987 | Shatabdi | Mumbai -> Delhi | Dep: 17:00 | Arr: 07:00
245 | Duronto Express | Nagpur -> Mumbai | Dep: 21:00 | Arr: 10:45
412 | Intercity Express | Ahmedabad -> Surat | Dep: 06:15 | Arr: 09:20
334 | Rajdhani Express | Delhi -> Kolkata | Dep: 16:45 | Arr: 10:15
278 | Garib Rath | Bangalore -> Chennai | Dep: 07:30 | Arr: 13:00
395 | Jan Shatabdi | Lucknow -> Varanasi | Dep: 05:40 | Arr: 10:30
409 | Goa Express | Delhi -> Goa | Dep: 15:00 | Arr: 12:00
516 | Tejas Express | Mumbai -> Goa | Dep: 06:00 | Arr: 14:00
628 | Chennai Mail | Chennai -> Kolkata | Dep: 20:00 | Arr: 09:30
742 | Punjab Mail | Mumbai -> Amritsar | Dep: 19:00 | Arr: 18:15
863 | Tamil Nadu Express | Delhi -> Chennai | Dep: 22:00 | Arr: 18:00

Menu:
1. Book Ticket
2. View Bookings
3. Cancel Ticket
4. View Train Status
5. Exit
Enter choice: 1
Enter train code: 409
Enter number of passengers to book (max 100): 2
Passenger 1 name: Aman
Passenger 1 age: 19
Seat Confirmed for Aman, Seat No: 1, PNR: 10000
Passenger 2 name: Rudra
Passenger 2 age: 21
Seat Confirmed for Rudra, Seat No: 2, PNR: 10001
Booking successful.
```

```
Menu:
1. Book Ticket
2. View Bookings
3. Cancel Ticket
4. View Train Status
5. Exit
Enter choice: 2

Train: Intercity Express (412)
Route: Ahmedabad -> Surat
Dep: 06:15 | Arr: 09:20
Passengers:
PNR: 10000 | Name: suraj | Age: 19 | Seat No: 1 |
-------------------------
Train: Goa Express (409)
Route: Delhi -> Goa
Dep: 15:00 | Arr: 12:00
Passengers:
PNR: 10000 | Name: Aman | Age: 19 | Seat No: 1 |
PNR: 10001 | Name: Rudra | Age: 21 | Seat No: 2 |
-------------------------
```

```
Menu:
1. Book Ticket
2. View Bookings
3. Cancel Ticket
4. View Train Status
5. Exit
Enter choice: 4
Enter Train Code to check status: 409
409 | Goa Express | Delhi -> Goa | Dep: 15:00 | Arr: 12:00
Total Seats: 100 | Booked: 2 | Available: 98 | Price: 2200 Rs
```

```
Menu:
1. Book Ticket
2. View Bookings
3. Cancel Ticket
4. View Train Status
5. Exit
Enter choice: 3
Enter PNR number to cancel: 10001
Ticket Cancelled  Aman
```

Github Link :
https://github.com/AmansWork-24-28/OOPS/blob/main/Railway%20Reservation%20System

Analysis :

Overall Program Complexity

Time Complexity (per operation):

1. Display Trains / Bookings: O(n)
2. Search Train / PNR: O(n)
3. Cancel Ticket: $O(n^2)$
4. View Train Status: O(1)
5. Space Complexity: O(n) for storing n trains, tickets, and passenger details in vectors; other operations use constant extra memory.