Railway Reservation System

UCS2265 – Fundamentals and Practice of Software Development

A PROJECT REPORT

Submitted By

MOKITHA S 3122 23 5001 083
MALATHI G 3122 23 5001 076
MUTHU ANUSHREE E 3122 23 5001 088

Department of Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)
Kalavakkam – 603110

June 2024

Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report titled "**Railway Reservation System**" is the bonafide work of "MOKITHA S (3122 23 5001 083), MUTHU ANUSHREE E (3122 23 5001 088) and MALATHI G (3122 23 5001 076)" who carried out the project work in the UCS2265 – Fundamentals and Practice of Software Development during the academic year 2023-24.

Internal Examiner   External Examiner

Date:

TABLE OF CONTENTS

INTRODUCTION:

A Railway Reservation System is a software application that facilitates the booking and management of railway tickets. It provides a platform for passengers to book tickets, check train schedules, seat availability, and receive confirmation of their reservations. This system aims to streamline the ticketing process and improve the overall passenger experience.

PROBLEM STATEMENT :

Develop a software system for doing reservation in trains. The Reservation system should contain The following features:

1. If a passenger wants to reserve ticket(s), firstly, he/she has to log in to the Railway System with valid credentials. Then, the passenger has to provide his/her details with the date of the journey, names of the passengers and their details, origin station Details, destination station details, and the class type of the required ticket(s).

2. The Railway Reservation System will provide the available Train-list, and Seat/berth Availability.

3. To book a ticket, passengers can pay through online mode. After successful payment Of the ticket fare, the System will generate the ticket and PNR no. will be given to the Passenger. The System also keeps the payment details and sends them to the system Admin.

4. The Passenger can check PNR status (confirmed, RAC, waiting list) by entering the PNR no. into the Reservation system.

5. The Reservation system should store all train details, fare details (by class, and date Wise), PNR no, date of trains, etc. This maintenance should be controlled by the Admin.

6. The System also has refund rules which have a date of reservation, ticket fare, and refundable percentage. The passenger can simply cancel the ticket(s) by entering the PNR no and a cancel ticket request. After cancelation, the Admin will pass the refundable amount to the passenger.

Constraints

For the night trains, the following constraints hold:

- There will be 12 sleeper class coaches each with 72 berths.
- There will be 4 AC 3 tier coaches each with 64 berths.

- There will be 3 AC 2 tier coaches each with 32 berths.
- There will be 1 AC First which will have 16 berths

For the day trains, the following constraints hold:

- There are 7 coaches each with 120 seats.
- There are 5 AC chair cars each with 60 seats.
- There are 2 executive chair cars each with 45 seats.

EXTENDED EXPLORATION OF PROBLEM STATEMENT:

The objective is to develop a software system for the reservation of railway tickets. The system will contain the following features:

(1) User Authentication and Details Submission:

Passenger logs in with valid credentials. Passenger provides journey details, passenger names and details, origin and destination stations, and class type.

(2) Train and Seat Availability:

System provides an available train list and seat/berth availability based on the provided details.

(3) Ticket Booking and Payment:

Passenger pays through online mode. After successful payment, the system generates the ticket and assigns a PNR number. Payment details are stored and sent to the system admin.

(4) PNR Status Checking:

Passengers can check PNR status (confirmed, RAC, waiting list) by entering the PNR number.

(5) Data Storage and Maintenance:

The system stores train details, fare details, PNR numbers, and other relevant information. Maintenance of data is controlled by the admin.

(6) Ticket Cancellation and Refund:

The system has refund rules based on the date of reservation, ticket fare, and refundable percentage. Passengers can cancel ticket(s) by entering the PNR number and a cancellation request. After cancellation, the admin processes the refundable amount to the passenger.

EXECUTIVE SUMMARY :

The Railway Registration System aims to simulate the train ticket booking online process used by the IRCTC. This project is significant as solving it involves handling real life challenges and it helps to simulate other similar booking and registration processes.

Steps involved:

1. User logs into the registration software

2. Passenger travel details are collected

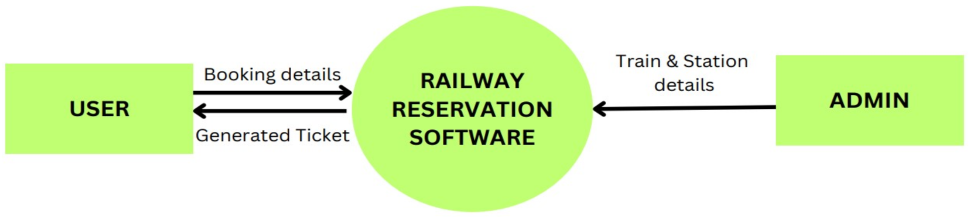
3. Computerized Booking

(i) Payment of fare

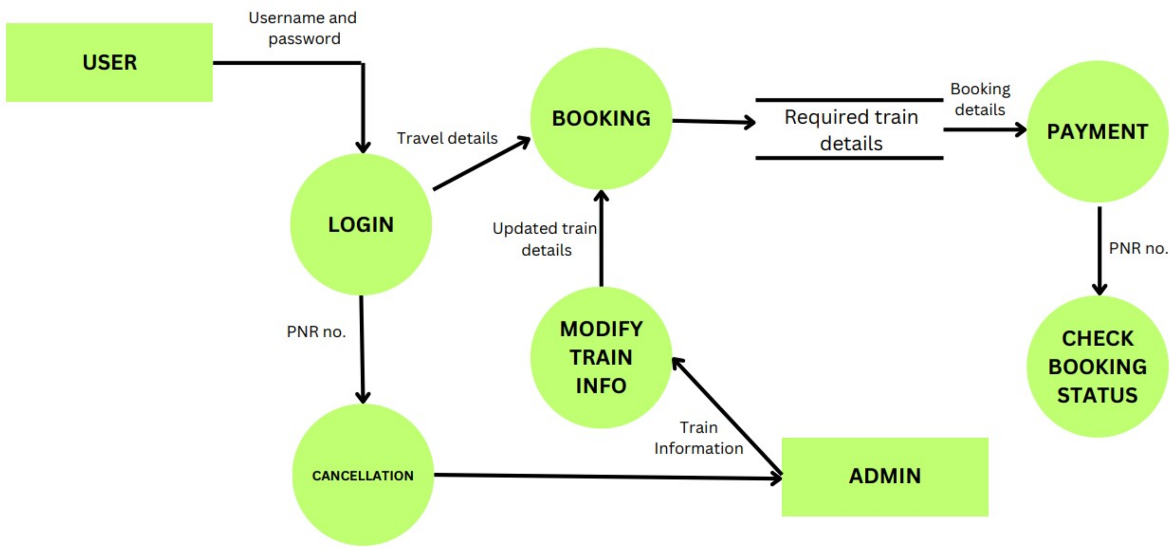
4. In case of ticket cancelation, payment details are sent to admin for verifying refundable amount

DATA FLOW DIAGRAM :

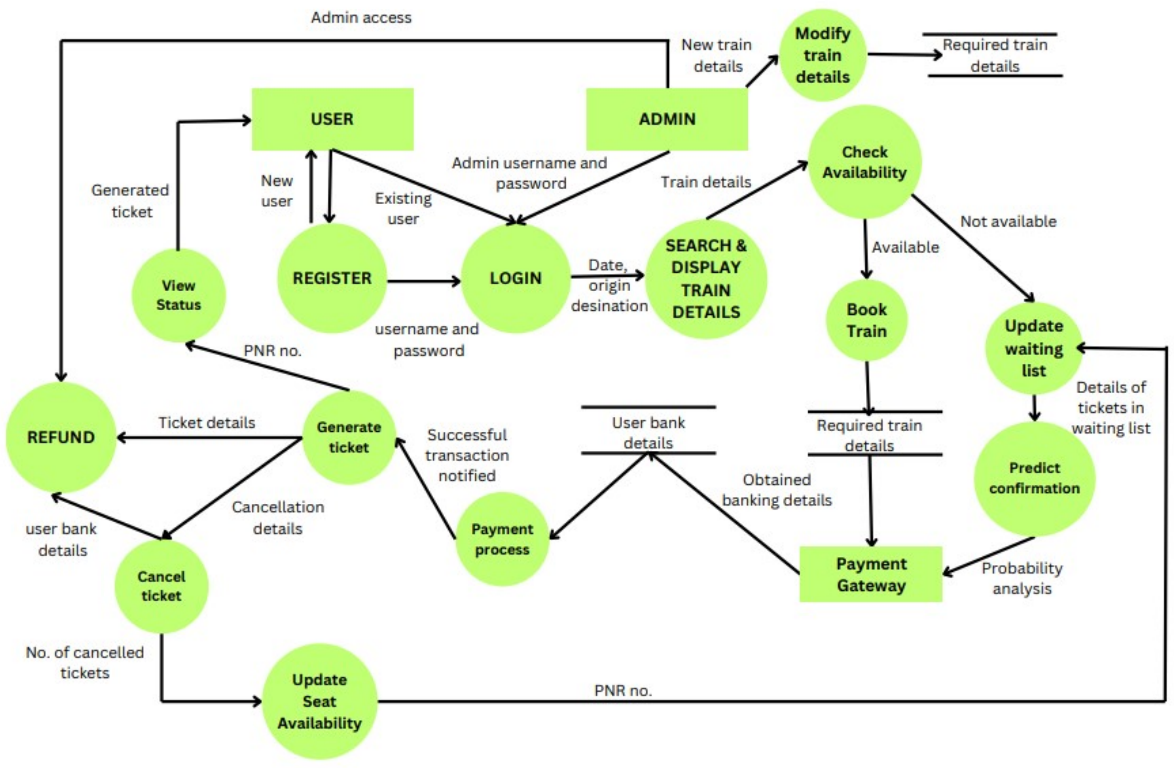
Level 0:



Level 1:



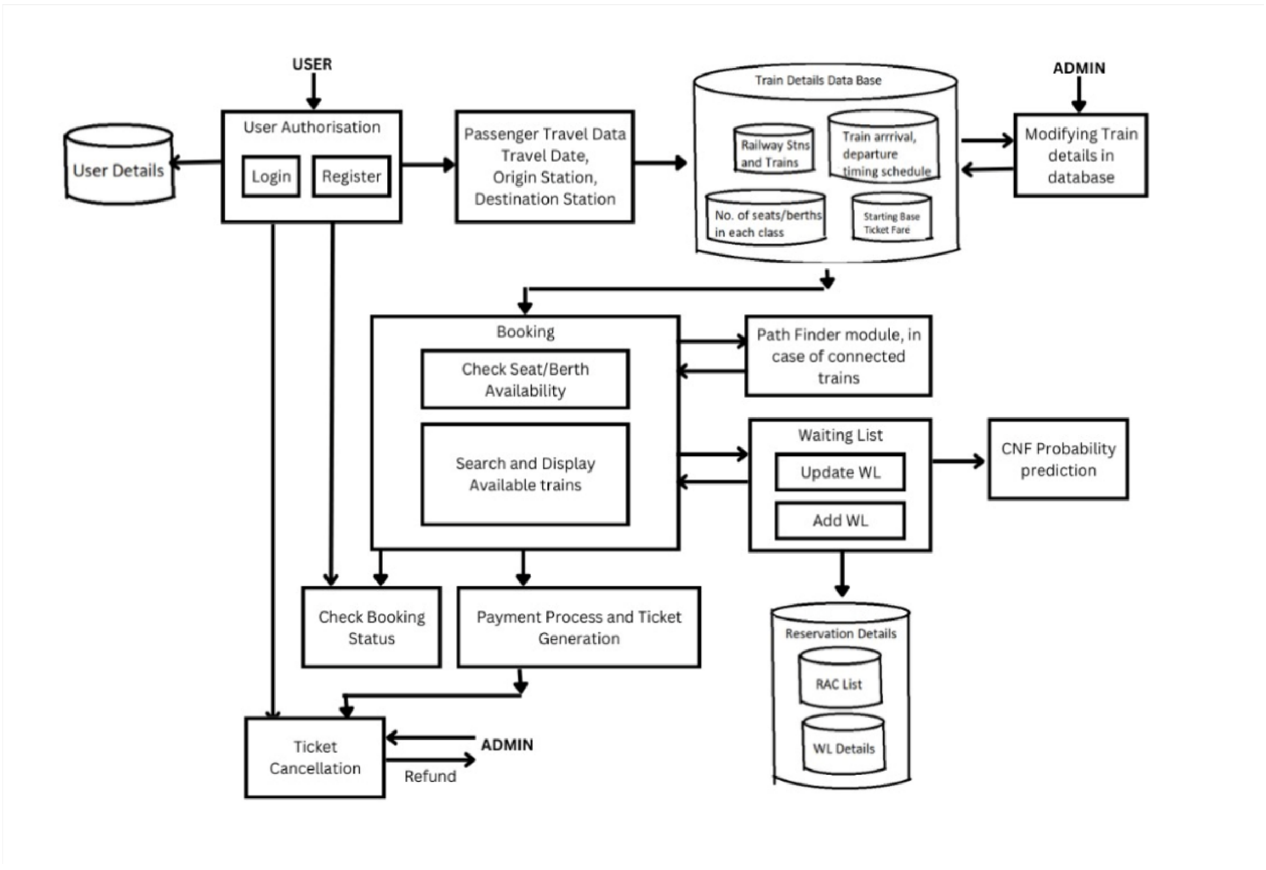
Level 2:



The project begins with a login page where users can sign up or log in securely. Admin registers with their credentials, stored in a login function. The booking stage follows, allowing users to search for trains, select preferences, and book tickets. If seats are unavailable, users can join a waiting list or modify their preferences. Ticket cancellations update the waiting list, and users can check the probability of confirmation. Payment is processed online, and successful transactions generate tickets and PNR numbers. Admin can modify train and station details. CheckBookingStatus allows users to view booking status and cancel tickets if necessary. Refunds are handled by admins based on cancellation timing. Overall, the system simulates seat allocation, booking, and cancellation processes in the Indian railway system.

DETAILED DESIGN:

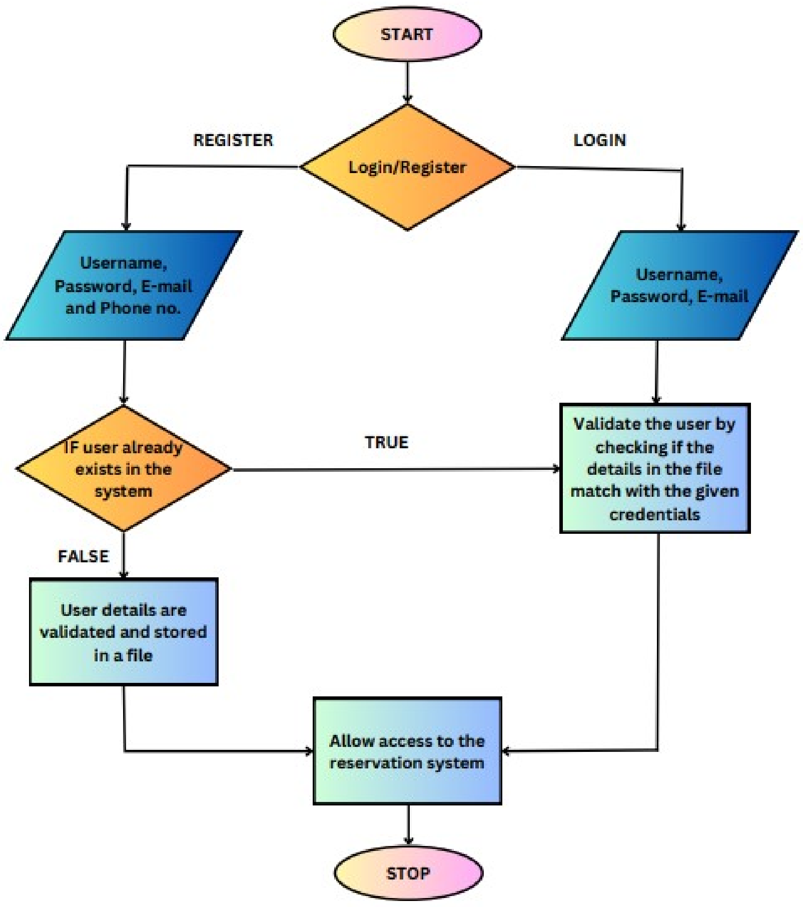
System Architecture:



USER
User Details
User Authorisation
Login
Register
Passenger Travel Data Travel Date, Origin Station, Destination Station
Train Details Data Base
Railway Stns and Trains
Train arrrival, departure timing schedule
No. of seats/berths in each class
Starting Base Ticket Fare
ADMIN
Modifying Train details in database
Booking
Check Seat/Berth Availability
Search and Display Available trains
Path Finder module, in case of connected trains
Waiting List
Update WL
Add WL
CNF Probability prediction
Check Booking Status
Payment Process and Ticket Generation
Reservation Details
RAC List
WL Details
Ticket Cancellation
ADMIN
Refund

MODULE FLOWCHART AND DESCRIPTION

ENTERING THE APPLICATION (entry.c)


START
REGISTER
Login/Register
LOGIN
Username, Password, E-mail and Phone no.
Username, Password, E-mail
IF user already exists in the system
TRUE
Validate the user by checking if the details in the file match with the given credentials
FALSE
User details are validated and stored in a file
Allow access to the reservation system
STOP

The user will be able to sign up as a new customer or log in and log out securely depending on the user's information existence in the system.

(1) Registration (Register() inside entry.c)

Precondition: None

Post-condition: The user will be registered in the system.

Flow:

- User enters the information in all the fields or atleast the mandatory fields.
- Checks if user already exists on the system.
- If not, the user is validated and allowed or denied access to the database.

(2) Login (login() inside entry.c)

Precondition: The user must already be registered in the system.

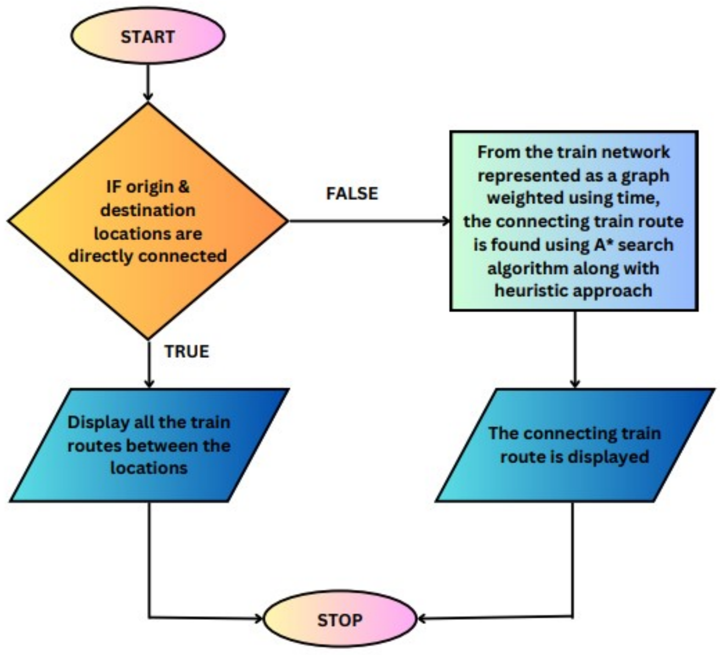
Post-condition: The user will be successfully logged in.

Flow:

- User enters username and password.
- Check if user exists in the database.

If yes, the user is validated and allowed or denied access to the database

PATH FINDER (PathFinder.c)

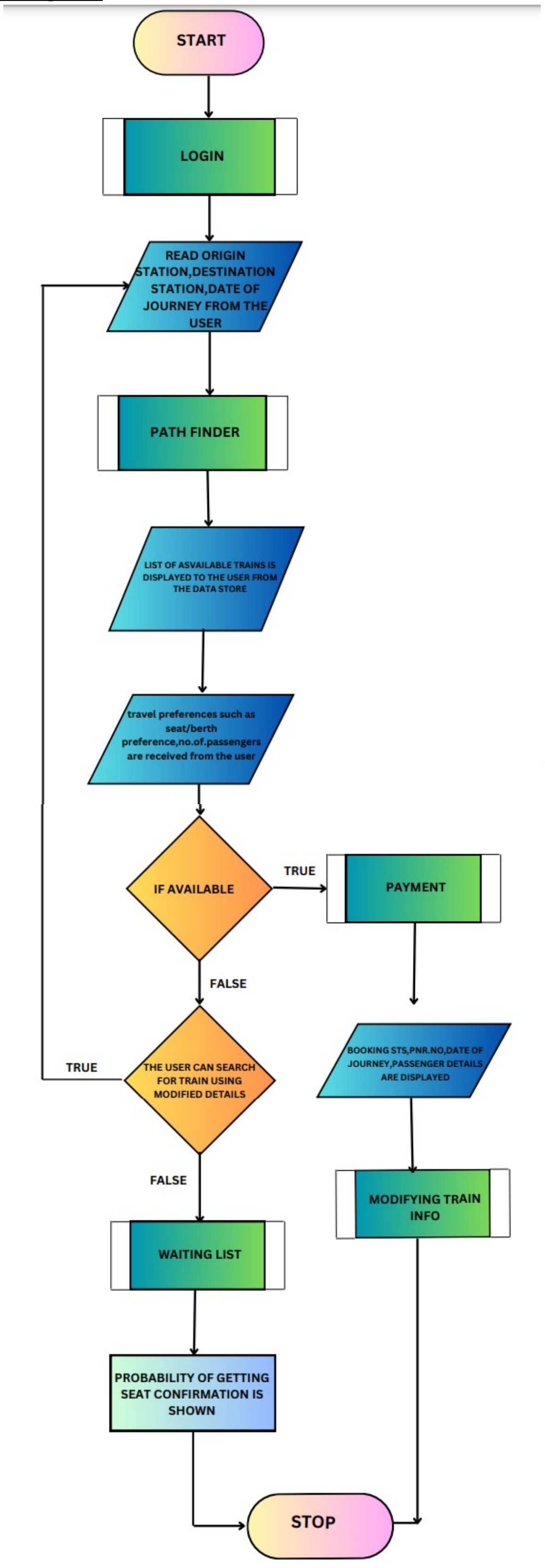

START
IF origin & destination locations are directly connected
FALSE
From the train network represented as a graph weighted using time, the connecting train route is found using A* search algorithm along with heuristic approach
TRUE
Display all the train routes between the locations
The connecting train route is displayed
STOP

Precondition: No direct route is available between the source and destination station.

Postcondition: Connecting trains between the stations are displayed.

We will implement the BFS algorithm as part of our train reservation system to efficiently find connecting trains between two stations, particularly in cases where there isn't a direct connection available. The BFS algorithm intelligently explores possible routes in the train network while considering factors such as distance, time, and availability, enabling us to provide passengers with optimal routes and minimize travel time.

BOOKING (booking.c)


START
LOGIN
READ ORIGIN STATION,DESTINATION STATION,DATE OF JOURNEY FROM THE USER
PATH FINDER
LIST OF ASVAILABLE TRAINS IS DISPLAYED TO THE USER FROM THE DATA STORE
travel preferences such as seat/berth preference,no.of.passengers are received from the user
IF AVAILABLE
TRUE
PAYMENT
FALSE
BOOKING STS,PNR.NO,DATE OF JOURNEY,PASSENGER DETAILS ARE DISPLAYED
TRUE
THE USER CAN SEARCH FOR TRAIN USING MODIFIED DETAILS
FALSE
MODIFYING TRAIN INFO
WAITING LIST
PROBABILITY OF GETTING SEAT CONFIRMATION IS SHOWN
STOP

Precondition: The user is logged into the system

Post-condition: Train ticket will be booked or user will be placed in waiting list.

Flow: Search() > DisplayTrainDetails() > CheckingAvailability() > UpdateWaitingList() > Book()

(1) Search() inside booking.c

- User will search for the required train by supplying details such as origin station, destination station and date of journey.

(2) DisplayTrainDetails() inside booking.c:

- List of available trains corresponding to the given origin station, destination station and date of journey.
- User will be able to choose a train.

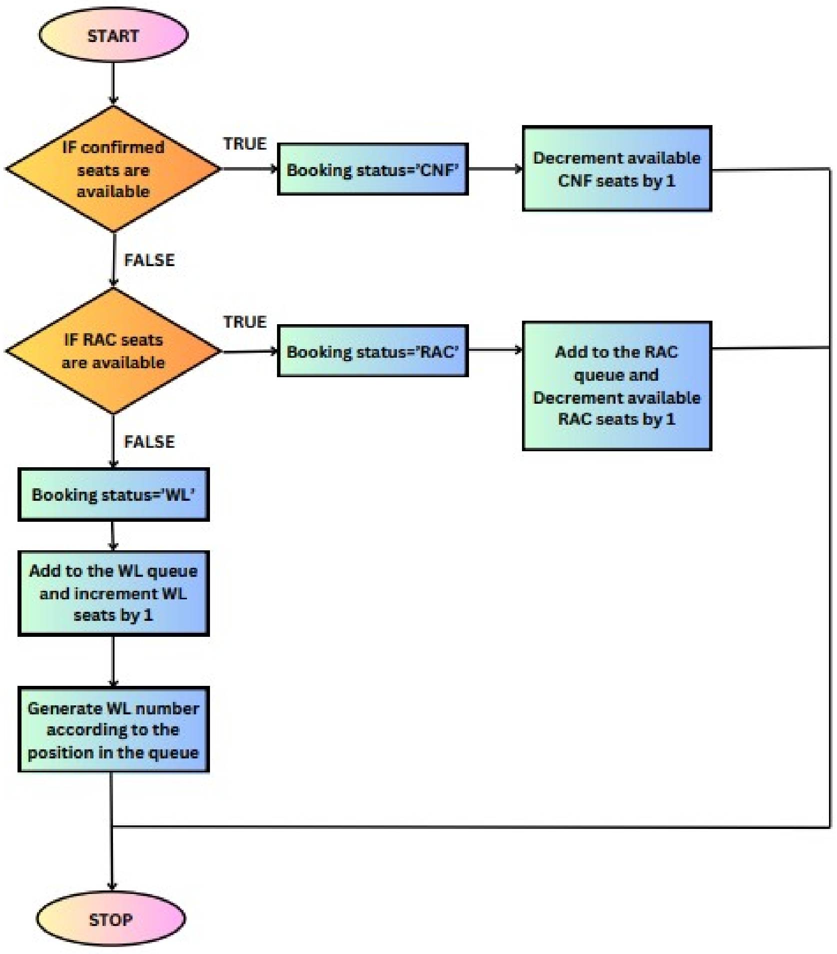
(3) CheckingAvailability() inside booking.c:

- User travel preferences such as Seat/berth, Seat preference(aisle, window) and Berth preference(lower, up) and additional travel details like number of passengers are received from user as input.
- Number of seats available, corresponding to user preferences in the selected train is checked
- If the required number of seats is available in the chosen train, Book() executes.
- If the required number of seats is not available in the chosen train, the user is placed in waiting list, AddWaitingList() gets executed.

(4) Book() inside booking.c:

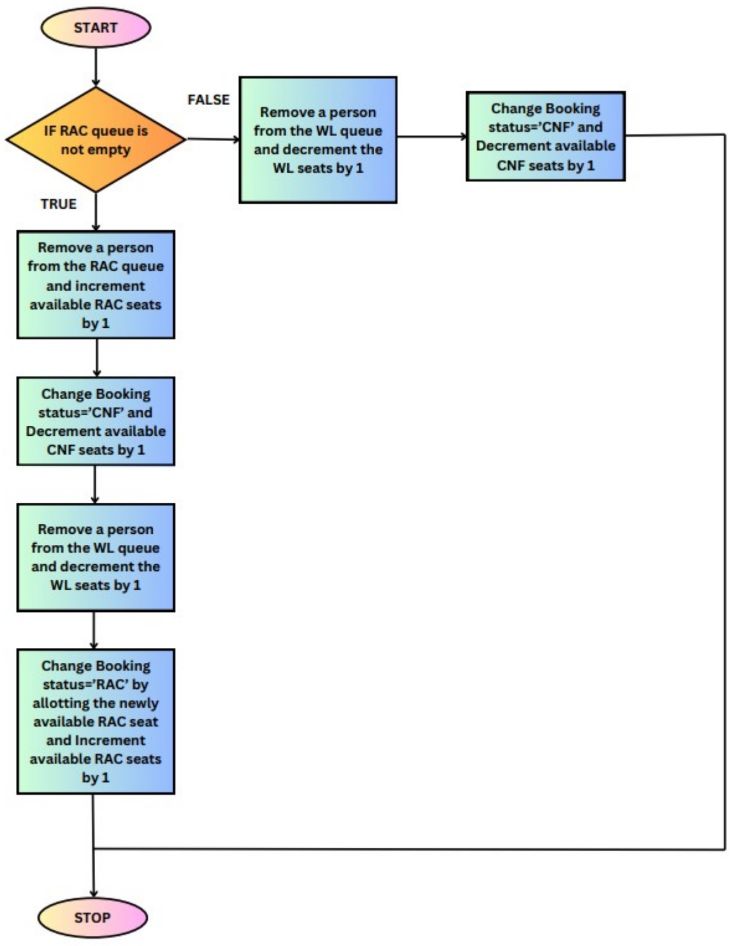
- The user can choose to pay through the online mode which will be processed by payment() in payment.c.
- If payment is successful, a ticket and PNR number will be generated which can be checked using booking status() in booking.c
- If payment is unsuccessful, refund is initiated by the admin.

<u>ADD WAITING LIST</u>


START
IF confirmed seats are available
TRUE
Booking status='CNF'
Decrement available CNF seats by 1
FALSE
IF RAC seats are available
TRUE
Booking status='RAC'
Add to the RAC queue and Decrement available RAC seats by 1
FALSE
Booking status='WL'
Add to the WL queue and increment WL seats by 1
Generate WL number according to the position in the queue
STOP

This train reservation module handles confirmed bookings by decrementing available seats and assigning a 'CNF' status. For unconfirmed bookings, it checks for RAC (Railway Reservation Against Cancellation) seats. If available, the passenger is added to the RAC queue, the seat count is decremented, and a 'RAC' status is assigned. If no RAC seats are available, the passenger is added to the waitlist, the waitlist count is incremented, and a 'WL' status is assigned. Regardless of confirmation, waitlist, or RAC, a confirmation message with the corresponding number (if applicable) is provided to the passenger.

UPDATE WAITING LIST


START
IF RAC queue is not empty
FALSE
Remove a person from the WL queue and decrement the WL seats by 1
Change Booking status='CNF' and Decrement available CNF seats by 1
TRUE
Remove a person from the RAC queue and increment available RAC seats by 1
Change Booking status='CNF' and Decrement available CNF seats by 1
Remove a person from the WL queue and decrement the WL seats by 1
Change Booking status='RAC' by allotting the newly available RAC seat and Increment available RAC seats by 1
STOP

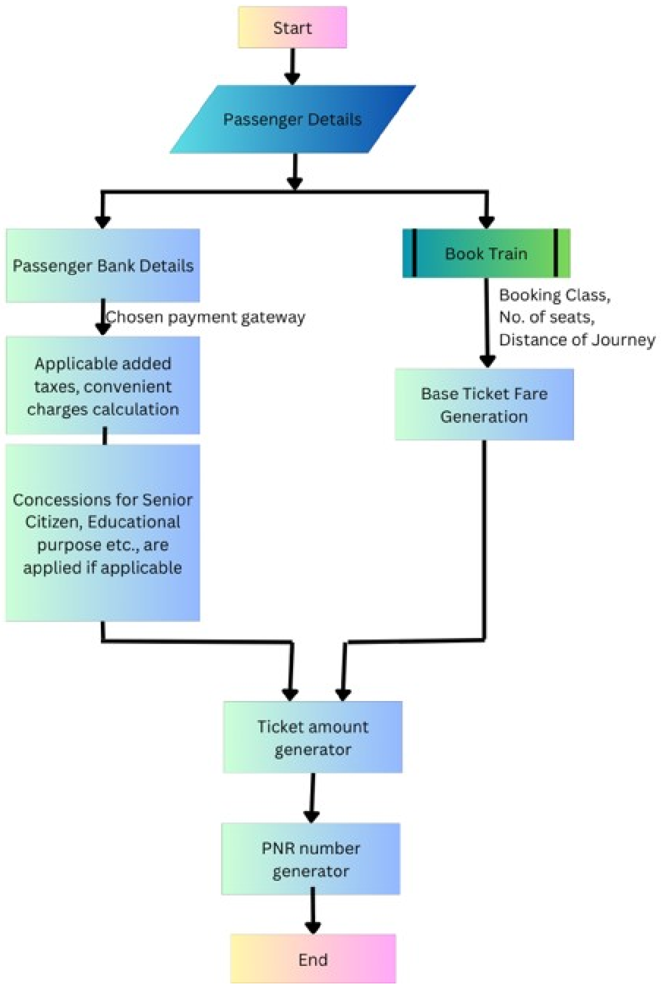
Precondition: A seat is canceled by the user

Postcondition: CNF, RAC and WL queues are updated accordingly.

Working:

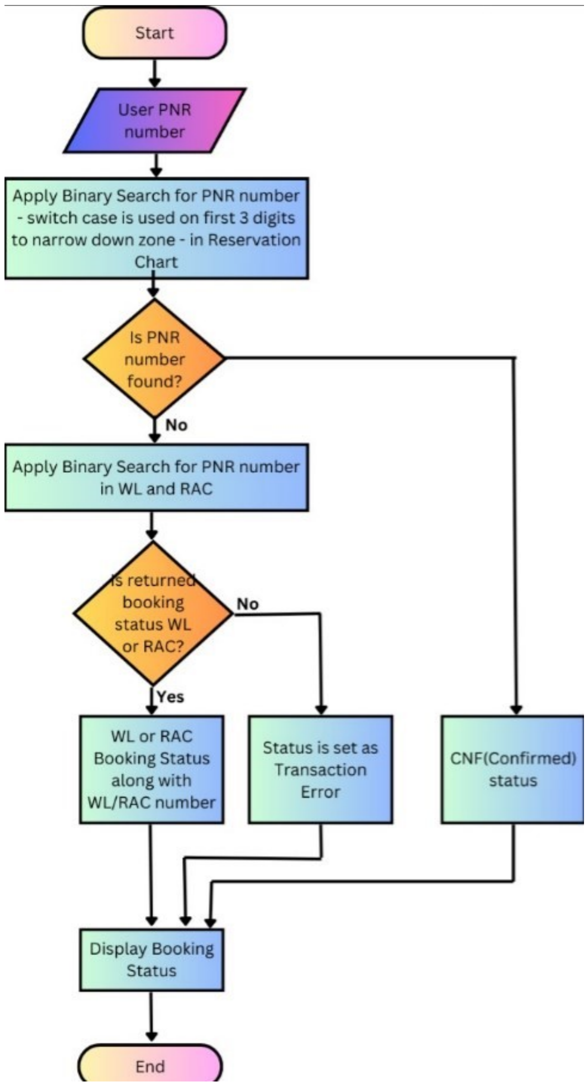
Update waiting list process checks if the RAC is not empty. If not, it removes the first person on the waitlist and assigns them a confirmed seat if available (decrementing confirmed seats). If no confirmed seats are available, it checks for RAC seats (seats allocated from cancellations). If an RAC seat is available, the passenger is assigned the seat and its count is decremented. If no confirmed or RAC seats are available, the removal process ends with the passenger remaining on the waitlist.

PAYMENT (payment.c)


Start
Passenger Details
Passenger Bank Details
Chosen payment gateway
Applicable added taxes, convenient charges calculation
Concessions for Senior Citizen, Educational purpose etc., are applied if applicable
Book Train
Booking Class, No. of seats, Distance of Journey
Base Ticket Fare Generation
Ticket amount generator
PNR number generator
End

The passenger chooses a preferred payment gateway for the transaction. Common options might include credit cards, debit cards, or e-wallets. The system calculates any applicable taxes and convenience charges based on the chosen booking class, number of seats, and journey distance. The base ticket fare is generated based on the chosen booking class, number of seats, and journey distance. If applicable, the system checks for any concession eligibility (discounts) for senior citizens, students, or other categories. The final ticket amount is generated by combining the base ticket fare (step 5) with any applicable taxes, convenience charges (step 4), and concessions (step 6). Upon successful payment, a PNR (Passenger Name Record) number is generated, which is a unique reference code for the passenger's reservation.

CHECK BOOKING STATUS (BookingStatus.c)

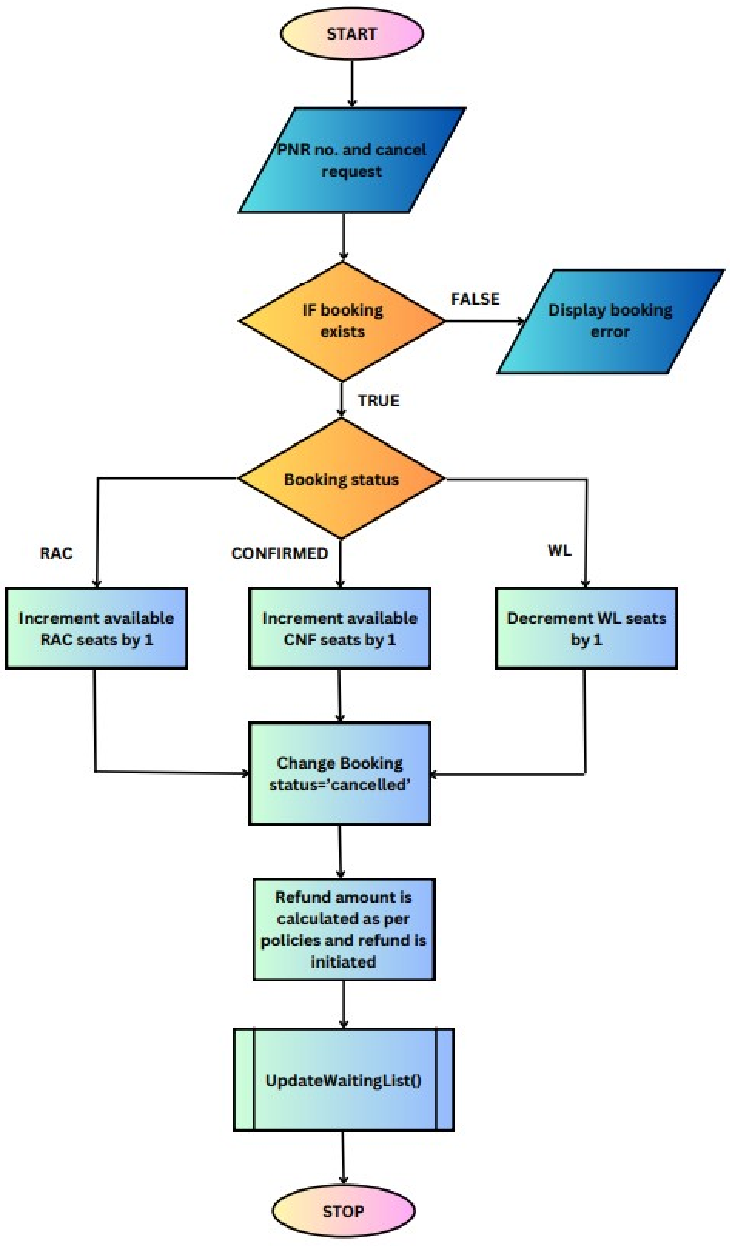

Start
User PNR number
Apply Binary Search for PNR number - switch case is used on first 3 digits to narrow down zone - in Reservation Chart
Is PNR number found?
No
Apply Binary Search for PNR number in WL and RAC
Is returned booking status WL or RAC?
No
Yes
WL or RAC Booking Status along with WL/RAC number
Status is set as Transaction Error
CNF(Confirmed) status
Display Booking Status
End

Precondition: User should have registered in the software and should have booked a ticket

Post-condition: Booking status is displayed without any discrepancies

Check booking status utilizes the PNR number for searching. It first narrows down the search zone and then employs binary search for efficient retrieval. If the PNR is found, the booking status is displayed. Otherwise, the system searches the waitlist and RAC sections, providing details or a "Not Found" message depending on the outcome.

CANCELLATION (cancel.c)

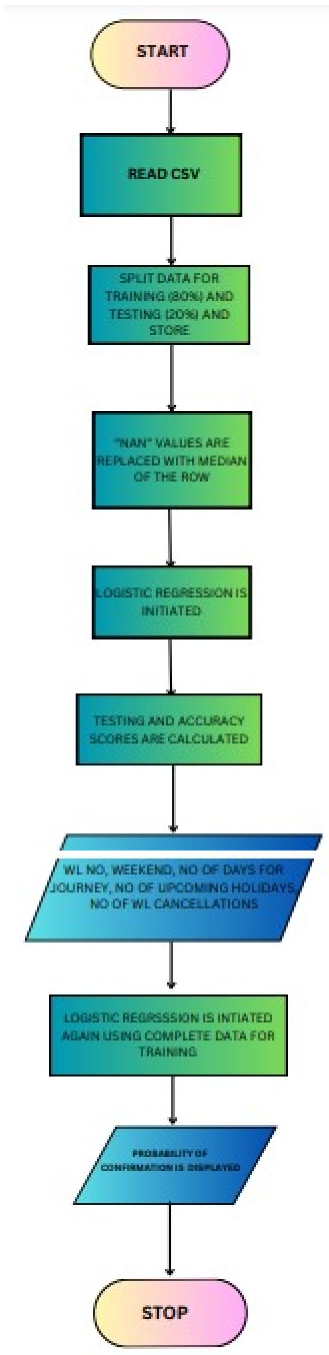

START
PNR no. and cancel request
IF booking exists
FALSE
Display booking error
TRUE
Booking status
RAC
CONFIRMED
WL
Increment available RAC seats by 1
Increment available CNF seats by 1
Decrement WL seats by 1
Change Booking status='cancelled'
Refund amount is calculated as per policies and refund is initiated
UpdateWaitingList()
STOP

Precondition: The user must have initiated a ticket cancelation or the booking must be unsuccessful.

Post-condition: Refund will be initiated

This train ticket cancellation flowchart starts with the passenger entering their PNR number. If valid, the system checks if the reservation is eligible for cancellation based on predefined rules (e.g., cancellation deadline). If eligible, a refund amount is calculated based on cancellation charges. The passenger then receives confirmation with details like PNR, cancellation charges and the final refund amount. If the PNR is invalid or the reservation isn't eligible for cancellation, the process ends accordingly.

PREDICTION MODULE:


START
READ CSV
SPLIT DATA FOR TRAINING (80%) AND TESTING (20%) AND STORE
"NAN" VALUES ARE REPLACED WITH MEDIAN OF THE ROW
LOGISTIC REGRESSION IS INITIATED
TESTING AND ACCURACY SCORES ARE CALCULATED
WL NO, WEEKEND, NO OF DAYS FOR JOURNEY, NO OF UPCOMING HOLIDAYS, NO OF WL CANCELLATIONS
LOGISTIC REGRSSSION IS INTIATED AGAIN USING COMPLETE DATA FOR TRAINING
PROBABILITY OF CONFIRMATION IS DISPLAYED
STOP

To predict the likelihood of a passenger's waitlist confirmation, the system will employ logistic regression. This statistical method is well-suited for classification problems, allowing me to analyze historical railway waiting list data and estimate the probability of confirmation based on features like initial waitlist position, travel date proximity, and seasonality.

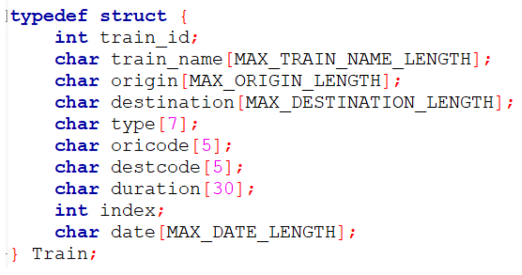
DATA ORGANIZATION AND RATIONALE BEHIND THE SELECTION:

STRUCTURES:

The use of structures in C allows for efficient grouping of related data. This helps in organizing complex data more intuitively.

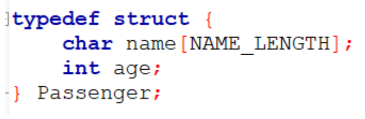
Manage collections of users and their attributes efficiently.

1.Train Structure: This structure holds all necessary information about the train, including its ID, name, origin, destination, type, seat availability, and other details. Using a structure for train information encapsulates all related data in a single unit, making the code more readable and manageable.



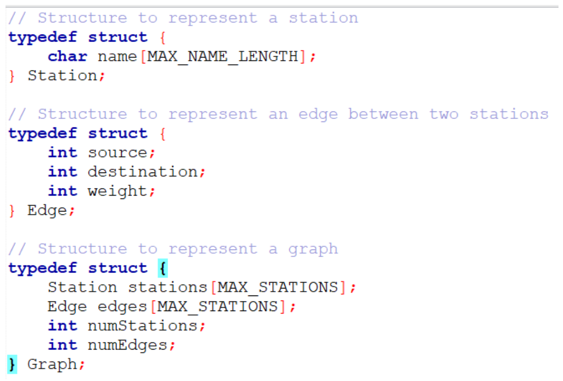
```c
typedef struct {
    int train_id;
    char train_name[MAX_TRAIN_NAME_LENGTH];
    char origin[MAX_ORIGIN_LENGTH];
    char destination[MAX_DESTINATION_LENGTH];
    char type[7];
    char oricode[5];
    char destcode[5];
    char duration[30];
    int index;
    char date[MAX_DATE_LENGTH];
} Train;
```

2.Passenger Structure: This structure is used to store information about passengers. It includes fields for the passenger's name and age. An array of this structure type is used to manage the waiting list.



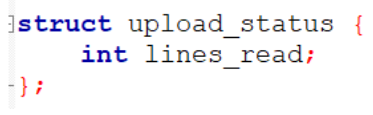
```c
typedef struct {
    char name[NAME_LENGTH];
    int age;
} Passenger;
```

3.Graph structure: This structure represents a graph of the train network, including arrays of stations and edges. It is used for finding connecting train routes when a direct train is not available.



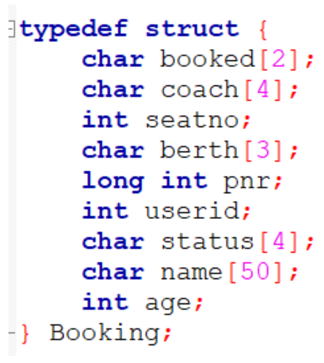
```c
// Structure to represent a station
typedef struct {
    char name[MAX_NAME_LENGTH];
} Station;

// Structure to represent an edge between two stations
typedef struct {
    int source;
    int destination;
    int weight;
} Edge;

// Structure to represent a graph
typedef struct {
    Station stations[MAX_STATIONS];
    Edge edges[MAX_STATIONS];
    int numStations;
    int numEdges;
} Graph;
```

4.Upload Status Structure: The `upload_status` structure tracks the number of lines read from the payload during the upload process for the mail.



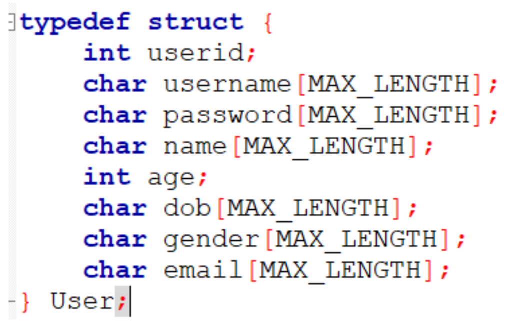
```
struct upload_status {
    int lines_read;
};
```

5. Booking structure: This is used to store the contents of the csv file for each individual train.



```
typedef struct {
    char booked[2];
    char coach[4];
    int seatno;
    char berth[3];
    long int pnr;
    int userid;
    char status[4];
    char name[50];
    int age;
} Booking;
```

6. Use structure: This is used to store the details of a newly registered user.



```
typedef struct {
    int userid;
    char username[MAX_LENGTH];
    char password[MAX_LENGTH];
    char name[MAX_LENGTH];
    int age;
    char dob[MAX_LENGTH];
    char gender[MAX_LENGTH];
    char email[MAX_LENGTH];
} User;
```

ARRAYS AND ARRAYS OF STRUCTURES:

Arrays and arrays of structures are used to handle multiple entities efficiently.

1. **Login array :**.Store and organize user data, ensuring easy access and manipulation

2. **Array of Structures**: An array of `Train` structures is used to store multiple train records. This makes it easy to iterate over all trains and apply filters based on user input.

3. **Waiting List Array**: An array of `Passenger` structures is used to manage the waiting list. This approach allows for easy addition and retrieval of passenger data by indexing into the array.

4. **Upload status structure:** This structure is used to track the status of an email upload operation in the CURL email sending process. It keeps track of the number of lines read from the payload text.

5. **RAC List Array**: An array of `Passenger` structures is used to manage the RAC list.

6. **Connectingtrains array** :

    - Arrays are used to manage fixed-size collections of data, such as the lists of visited stations and the queue for BFS in the `FindConnectingTrains` function.

    - The code uses an array of structures to represent both stations and edges in the graph. Each station and edge is stored in a structure, and arrays of these structures manage the entire collection of stations and edges.

Rationale: Using arrays simplifies the management of fixed-size lists. The fixed sizes (`WAITING_LIST_SIZE` and `RAC_LIST_SIZE`) ensure the program handles known maximum numbers of passengers.

FILE OPERATIONS:

File operations are crucial for persisting train data between program executions. The use of binary files allows efficient reading and writing of complex data structures.

CSV (Comma-Separated Values) files are a popular format for storing tabular data in plain text. In the context of a railway reservation system, CSV files can be used for various purposes, such as storing and retrieving data related to users, trains, bookings, and more.

THE USAGE OF FILES IN THE PROJECT IS EXPLAINED BELOW:

User Details Storage:

- **File**: `user_details.csv`
- **Purpose**: This file stores the details of registered users, including their ID, username, password, and email address. It is used for login, registration, and password recovery functionalities.

Train Data Storage:

- **File**: `modified_trains4.csv`
- **Purpose**: This file contains details of all available trains, including train name, origin, destination, date, number, and type. It is used to filter and select trains based on user input for booking purposes.

Seat Data storage:

- **File**: `day.csv`
- **File:** `night.csv`
- **Purpose**: This file contains details of seat count of all type of seats such as 1AC,2AC,3AC,SL in night.csv and the seat count of AC ,EXECUTIVE CHAIR of the day trains in day.csv. It is used to filter, select and book trains based on user input for booking purposes.

Booking Data Storage:

- **File**: `trains_v1.csv`
- **Purpose**: This file stores booking details, including booked status, coach type, seat number, berth type, PNR number, user ID, booking status, passenger name, and age. It is used for managing and storing booking information.

Graph Data Storage for Pathfinding:

- **File**: `modified_trains4.csv` (example)
- **Purpose**: This file stores the data for the graph representation of the train network, which includes stations and edges. It is used for finding paths between stations when a direct train is not available

EXPLANATION OF APIs OR LIBRARIES :

Standard Libraries Used

1. **stdio.h (Standard Input/Output Library)**

   Functions Used: `printf`, `scanf`, `fgets`, `fopen`, `fclose`, `fprintf`

2. **stdlib.h (Standard Library)**

   Functions Used: `malloc`, `free`, `atoi`, `atof`, `realloc`

3. **string.h (String Handling Library)**

   Functions Used: `strcmp`, `strcpy`, `strlen`, `strtok`, `strdup`, `strcspn`

4. **ctype.h (Character Handling Library)**

   Functions Used: `isdigit`

5. Booking system header file(booking_v1.h):

- Defines data structures (`Train`) that other parts of the program can use.
- Declares functions (`readLine`, `parseTrainData`, etc.) that other parts of the program can call.

These declarations define an interface. The implementation of these functions (presumably in `booking_v1.c`) provides the behavior, while the header file provides the API.

Explanation:

This header file defines the structures and functions used in the booking system. It includes necessary standard libraries and defines constants, data structures, and function prototypes

6. **Payment System Header File (`payment_v1.h`)**

Explanation:

This header file manages the payment processing aspects of the booking system. It includes necessary standard libraries and defines constants, functions, and validation routines.

7. **daytrain_wl.h**

Explanation:

The `daytrain_wl.h` header file manages the waiting list for passengers in a day train. It defines the necessary data structures, constants, and functions required to handle the addition of passengers to the waiting list, display the waiting list, and update the waiting list in a CSV file.

8. **daytrain_book.h**

Explanation:

The `daytrain_book.h` header file handles the booking process for train seats and updates the corresponding CSV file to reflect seat availability. It interacts with the waiting list management functions from `daytrain_wl.h` when necessary.

9. **Login_v1.h**

Explanation:

The `login_v1.h` header file provides the declarations and necessary includes for implementing user login functionality in a C program. This header file includes function prototypes, macro definitions, and structure declarations related to user login, registration, and password management.

10. **Mail_v1.h**

Explanation:

The `mail_v1.h` header file provides the declarations and necessary includes for sending an email using the libcurl library in a C program. This header file includes constant definitions, a structure declaration, and function prototypes related to email functionality

11.Night_train_book.h

Explanation:

The `night_train_book.h` header file includes function declarations and necessary includes for managing train seat reservations and updating the available seat counts in a CSV file. The main functionalities provided by this header file include checking RAC (Reservation Against Cancellation) status and updating seat counts for different classes.

12.Night_train_rac_wl.h

Explanation:

The `night_rac_wl.h` header file is used to manage the Reservation Against Cancellation (RAC) and Waiting List (WL) for train reservations. It includes data structures and functions to add passengers to the RAC or waiting list, display these lists, and handle train reservations.

13.path_v1.h

Explanation:

The path_v1.h header file encapsulates functionalities for managing a graph representation of train stations and routes. It defines essential data structures such as Station and Edge within the overarching Graph structure. Key functions like AddStation and AddEdge enable the addition of stations and edges to the graph, ensuring uniqueness through checks against existing entries. Additionally, PrintGraph outputs a formatted representation of the graph to the console, detailing station names and associated connections. Overall, path_v1.h provides a comprehensive framework for managing and analyzing train station networks efficiently.

14.Check status.h

Explanation:

It introduces functionalities like reading booking data from a CSV file based on a train number, printing booking details in a formatted manner, checking booking status by Passenger Name Record (PNR), and generating payload text containing booking details. The header guard ensures the contents are included only once, preventing duplication during compilation. Overall, this header file forms a crucial component of a system managing train bookings, status inquiries, and payload generation for communication or reporting purposes

USER INTERFACE DESIGN :

The user interface design in the railway reservation system project employs ANSI color codes to enhance visual presentation and improve user experience. Color codes are strategically utilized throughout the program to differentiate between various elements and provide a more engaging interface for users interacting with the system through the console.

Color Code Definitions:

- ANSI color codes such as `CYAN`, `BRIGHT_CYAN`, `GREEN`, `BRIGHT_GREEN`, etc., are defined at the beginning of the code using `#define` statements. These codes are used consistently to display text in different colors, adding visual appeal to the user interface.

The use of ANSI color codes in the user interface design of the railway reservation system contributes to a visually appealing and user-friendly experience, improving navigation, readability, and engagement for users interacting with the system.

PLATFORM USED FOR CODE DEVELOPMENT :

- **Code::blocks:**

  Code::Blocks is an open-source integrated development environment (IDE) that supports multiple compilers, making it versatile for C/C++ development.

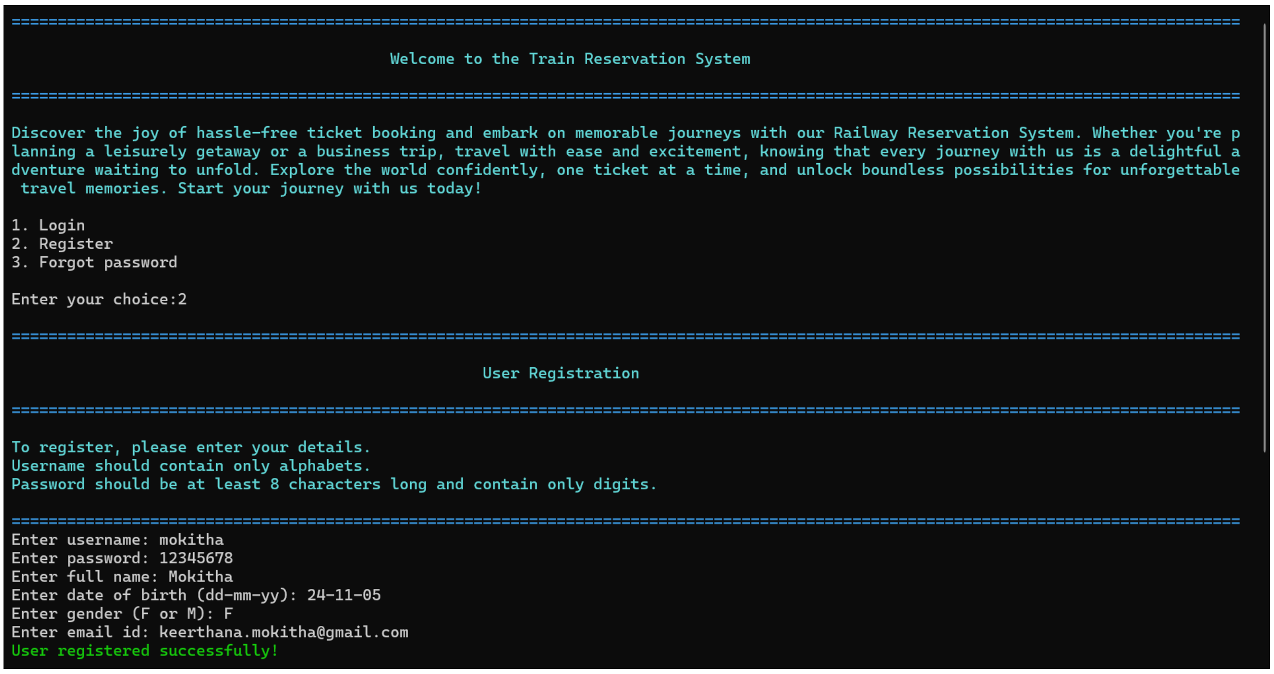
- **Replit:**

  Replit is an online coding platform that allows for collaborative coding, especially useful for team projects where multiple developers need to work on the same codebase simultaneously.

TEST CASES(under different scenarios):

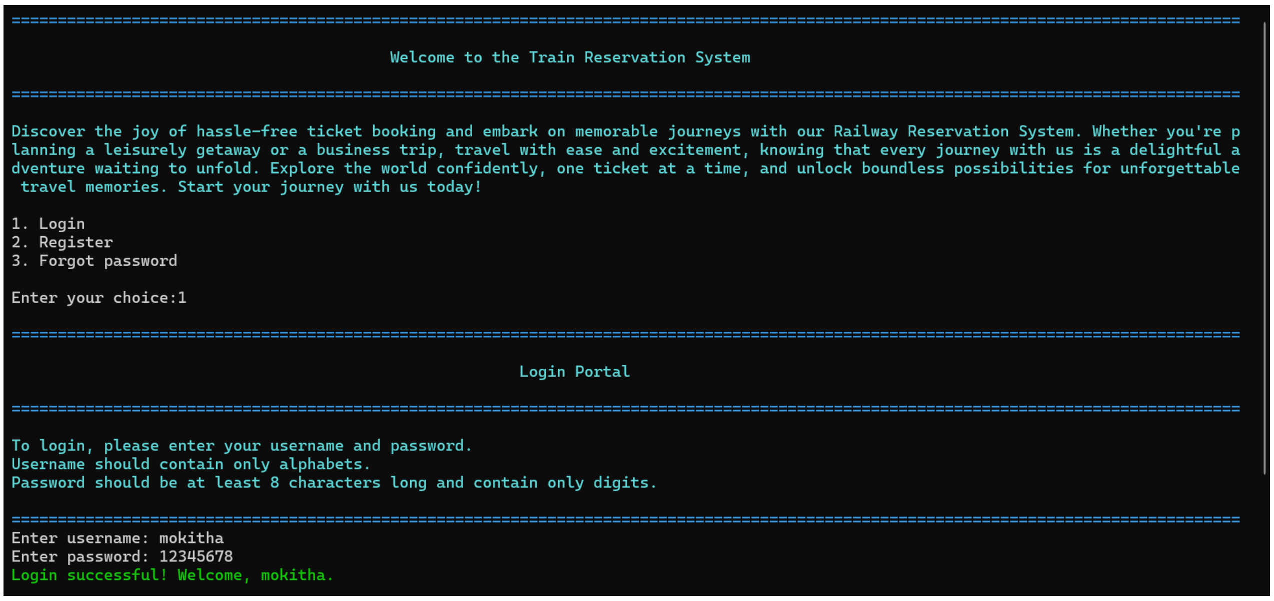
Logging into the system

Test case: 1) Register



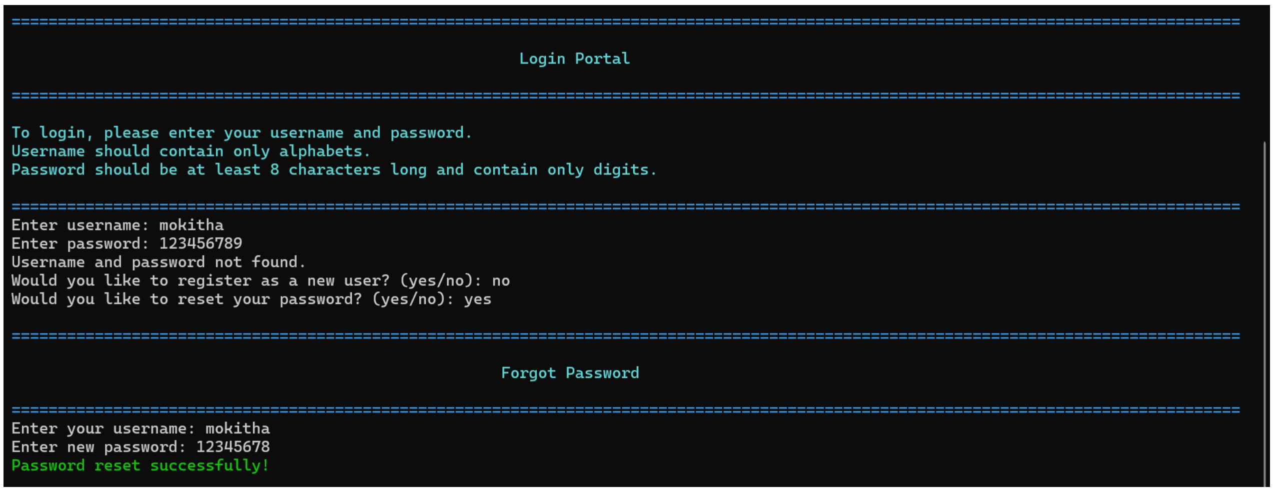
```
===========================================================================================================================

                                        Welcome to the Train Reservation System

===========================================================================================================================

Discover the joy of hassle-free ticket booking and embark on memorable journeys with our Railway Reservation System. Whether you're p
lanning a leisurely getaway or a business trip, travel with ease and excitement, knowing that every journey with us is a delightful a
dventure waiting to unfold. Explore the world confidently, one ticket at a time, and unlock boundless possibilities for unforgettable
 travel memories. Start your journey with us today!

1. Login
2. Register
3. Forgot password

Enter your choice:2

===========================================================================================================================

                                                  User Registration

===========================================================================================================================

To register, please enter your details.
Username should contain only alphabets.
Password should be at least 8 characters long and contain only digits.

===========================================================================================================================
Enter username: mokitha
Enter password: 12345678
Enter full name: Mokitha
Enter date of birth (dd-mm-yy): 24-11-05
Enter gender (F or M): F
Enter email id: keerthana.mokitha@gmail.com
User registered successfully!
```

Test case: 2) Login



```
===========================================================================================================================

                                        Welcome to the Train Reservation System

===========================================================================================================================

Discover the joy of hassle-free ticket booking and embark on memorable journeys with our Railway Reservation System. Whether you're p
lanning a leisurely getaway or a business trip, travel with ease and excitement, knowing that every journey with us is a delightful a
dventure waiting to unfold. Explore the world confidently, one ticket at a time, and unlock boundless possibilities for unforgettable
 travel memories. Start your journey with us today!

1. Login
2. Register
3. Forgot password

Enter your choice:1

===========================================================================================================================

                                                      Login Portal

===========================================================================================================================

To login, please enter your username and password.
Username should contain only alphabets.
Password should be at least 8 characters long and contain only digits.

===========================================================================================================================
Enter username: mokitha
Enter password: 12345678
Login successful! Welcome, mokitha.
```

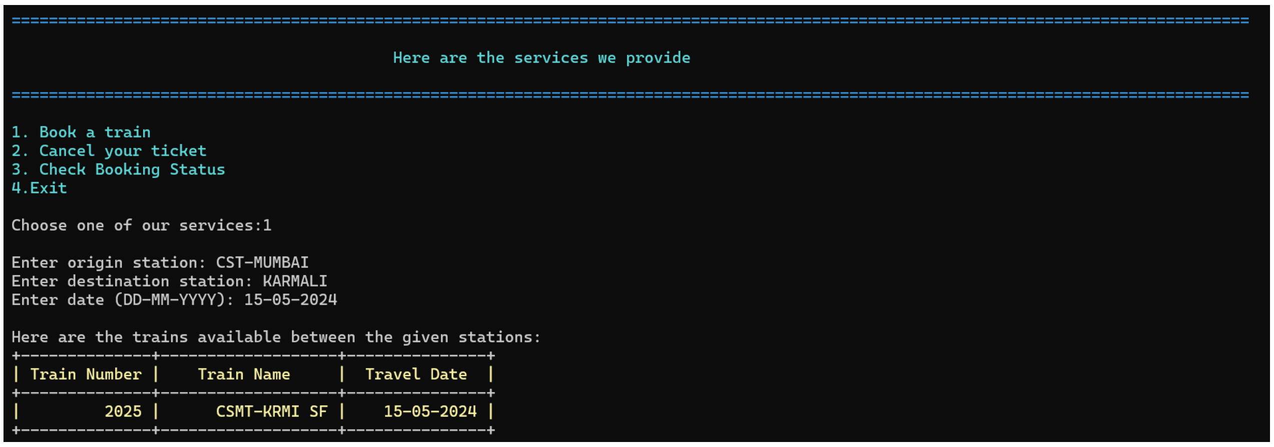
Test case: 3) Forgot password


```
==================================================================================================================================

                                                    Login Portal

==================================================================================================================================

To login, please enter your username and password.
Username should contain only alphabets.
Password should be at least 8 characters long and contain only digits.

==================================================================================================================================
Enter username: mokitha
Enter password: 123456789
Username and password not found.
Would you like to register as a new user? (yes/no): no
Would you like to reset your password? (yes/no): yes

==================================================================================================================================

                                                  Forgot Password

==================================================================================================================================
Enter your username: mokitha
Enter new password: 12345678
Password reset successfully!
```

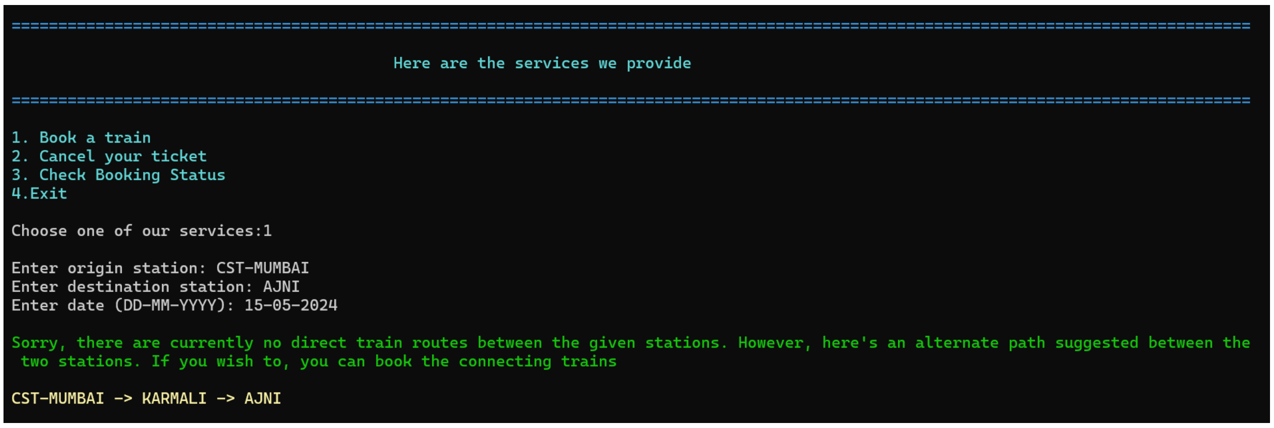
Service 1:Booking a train

Finding train routes:

Test case:1) Direct train route available

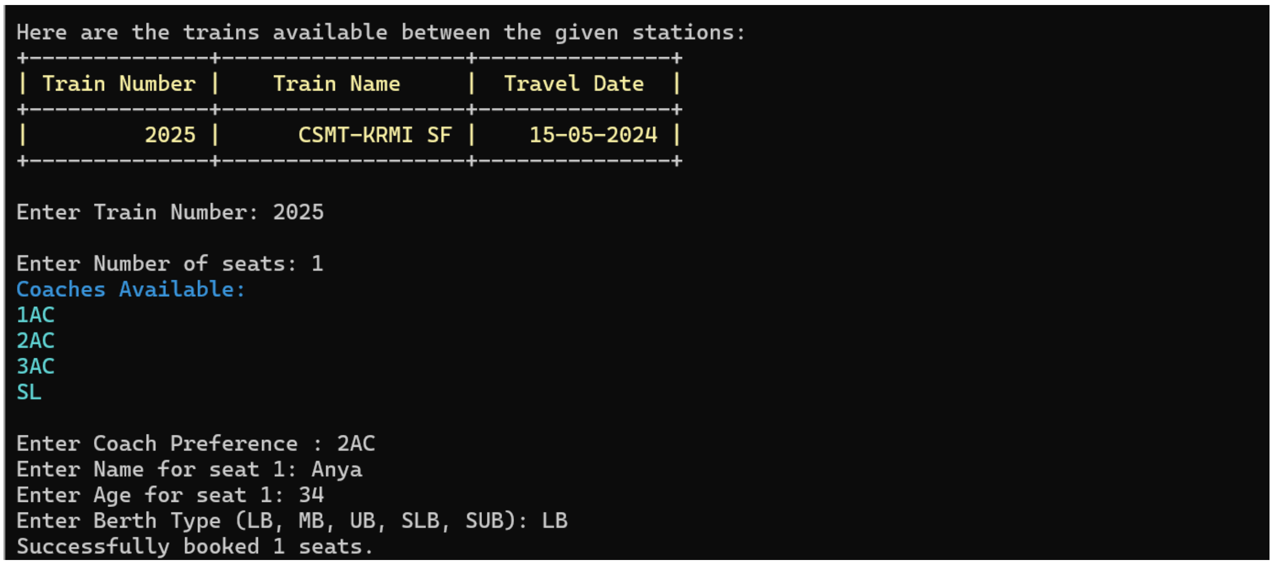

```
===================================================================================================================================

                                         Here are the services we provide

===================================================================================================================================

1. Book a train
2. Cancel your ticket
3. Check Booking Status
4.Exit

Choose one of our services:1

Enter origin station: CST-MUMBAI
Enter destination station: KARMALI
Enter date (DD-MM-YYYY): 15-05-2024

Here are the trains available between the given stations:
+--------------+-------------------+---------------+
| Train Number |     Train Name    |  Travel Date  |
+--------------+-------------------+---------------+
|         2025 |      CSMT-KRMI SF |    15-05-2024 |
+--------------+-------------------+---------------+
```

Test case:2) No direct train route available


```
===================================================================================================================================

                                         Here are the services we provide

===================================================================================================================================

1. Book a train
2. Cancel your ticket
3. Check Booking Status
4.Exit

Choose one of our services:1

Enter origin station: CST-MUMBAI
Enter destination station: AJNI
Enter date (DD-MM-YYYY): 15-05-2024

Sorry, there are currently no direct train routes between the given stations. However, here's an alternate path suggested between the
 two stations. If you wish to, you can book the connecting trains

CST-MUMBAI -> KARMALI -> AJNI
```

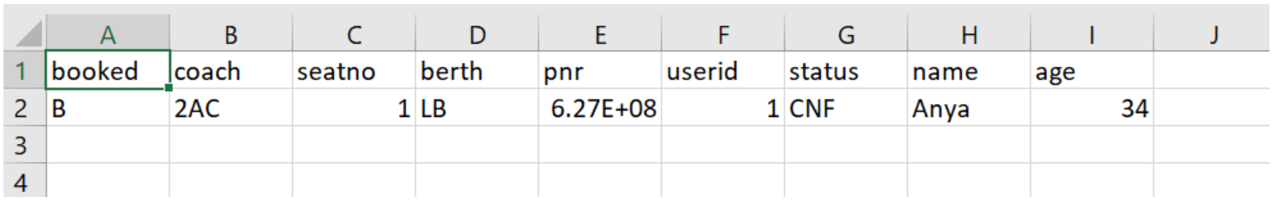
Booking a train:

Test case:1) CNF seat available



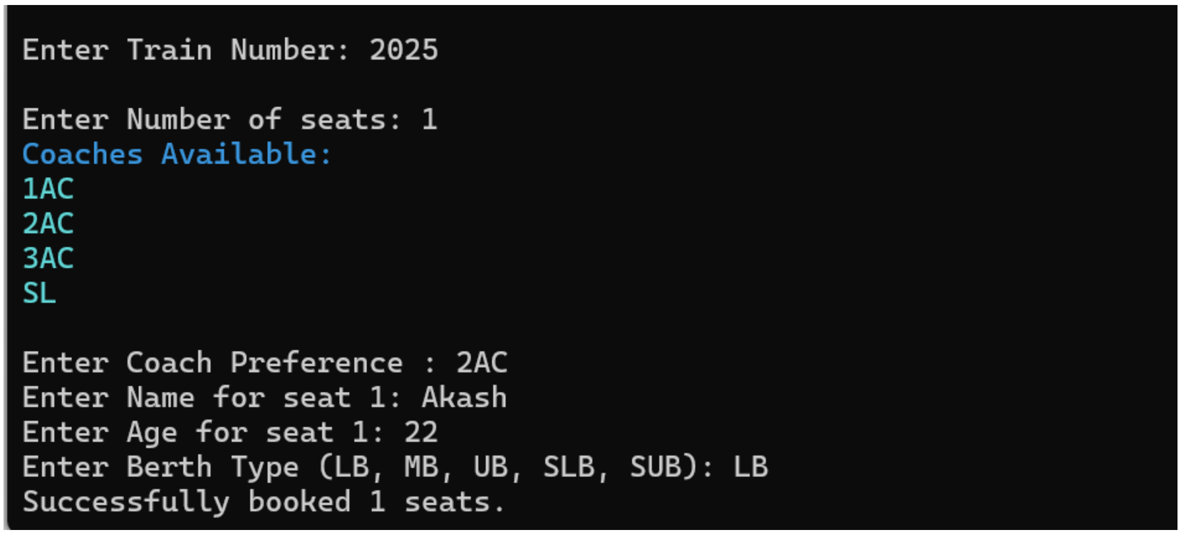
```
Here are the trains available between the given stations:
+--------------+------------------+---------------+
| Train Number |     Train Name   |  Travel Date  |
+--------------+------------------+---------------+
|         2025 |      CSMT-KRMI SF |    15-05-2024 |
+--------------+------------------+---------------+

Enter Train Number: 2025

Enter Number of seats: 1
Coaches Available:
1AC
2AC
3AC
SL

Enter Coach Preference : 2AC
Enter Name for seat 1: Anya
Enter Age for seat 1: 34
Enter Berth Type (LB, MB, UB, SLB, SUB): LB
Successfully booked 1 seats.
```

A new csv file 2025.csv will be created with the following contents

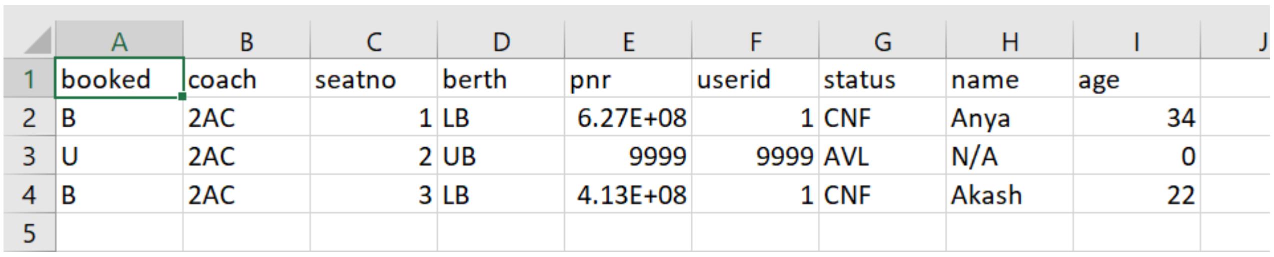


	A	B	C	D	E	F	G	H	I	J
1	booked	coach	seatno	berth	pnr	userid	status	name	age	
2	B	2AC	1	LB	6.27E+08	1	CNF	Anya	34	
3										
4										

In 2AC, the seat numbers for LB and UB keep alternating i.e if seat 1 is LB, the seat 2 will be UB. On booking another LB seat in 2AC successively, the seat gets allotted accordingly as follows.



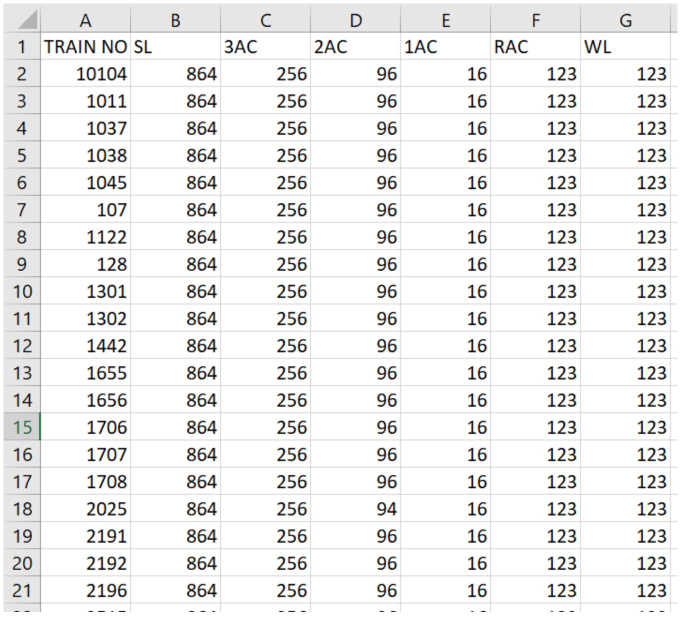
```
Enter Train Number: 2025

Enter Number of seats: 1
Coaches Available:
1AC
2AC
3AC
SL

Enter Coach Preference : 2AC
Enter Name for seat 1: Akash
Enter Age for seat 1: 22
Enter Berth Type (LB, MB, UB, SLB, SUB): LB
Successfully booked 1 seats.
```



	A	B	C	D	E	F	G	H	I	J
1	booked	coach	seatno	berth	pnr	userid	status	name	age	
2	B	2AC	1	LB	6.27E+08	1	CNF	Anya	34	
3	U	2AC	2	UB	9999	9999	AVL	N/A	0	
4	B	2AC	3	LB	4.13E+08	1	CNF	Akash	22	
5										

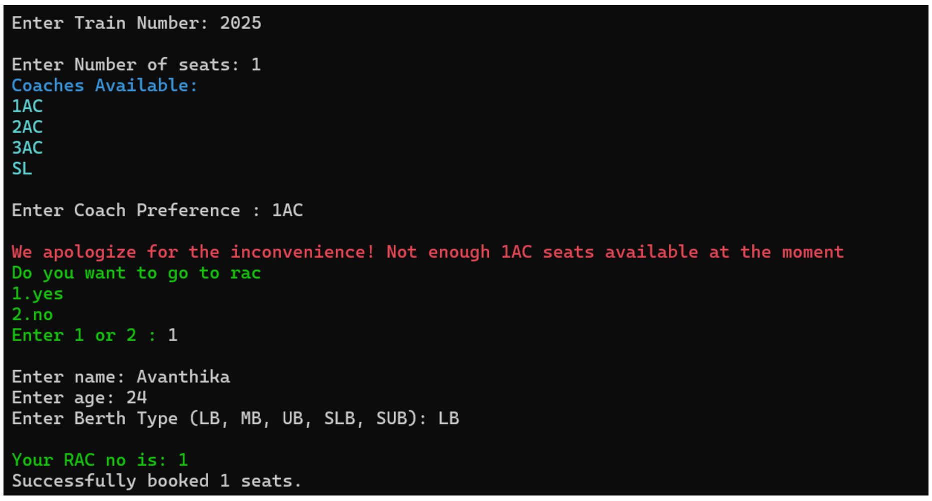
Similarly, dynamic seat allocation is done for all the coaches.

night.csv containing the night trains after booking the above seats.

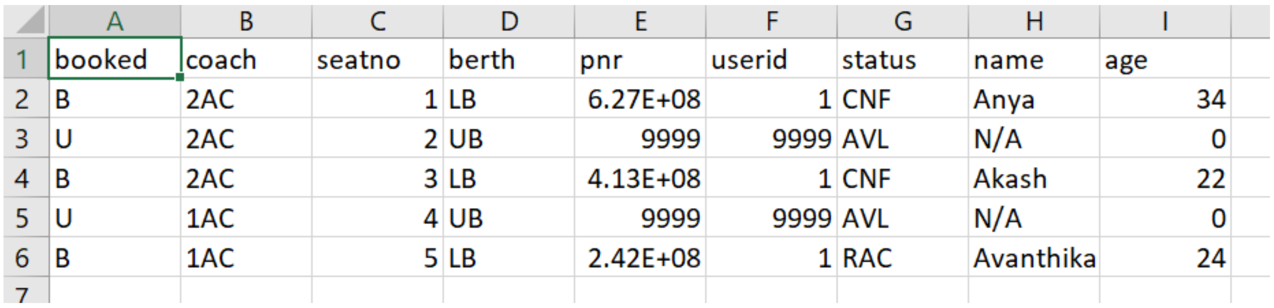


	A	B	C	D	E	F	G
1	TRAIN NO	SL	3AC	2AC	1AC	RAC	WL
2	10104	864	256	96	16	123	123
3	1011	864	256	96	16	123	123
4	1037	864	256	96	16	123	123
5	1038	864	256	96	16	123	123
6	1045	864	256	96	16	123	123
7	107	864	256	96	16	123	123
8	1122	864	256	96	16	123	123
9	128	864	256	96	16	123	123
10	1301	864	256	96	16	123	123
11	1302	864	256	96	16	123	123
12	1442	864	256	96	16	123	123
13	1655	864	256	96	16	123	123
14	1656	864	256	96	16	123	123
15	1706	864	256	96	16	123	123
16	1707	864	256	96	16	123	123
17	1708	864	256	96	16	123	123
18	2025	864	256	94	16	123	123
19	2191	864	256	96	16	123	123
20	2192	864	256	96	16	123	123
21	2196	864	256	96	16	123	123

<u>Test case:2)</u> RAC available

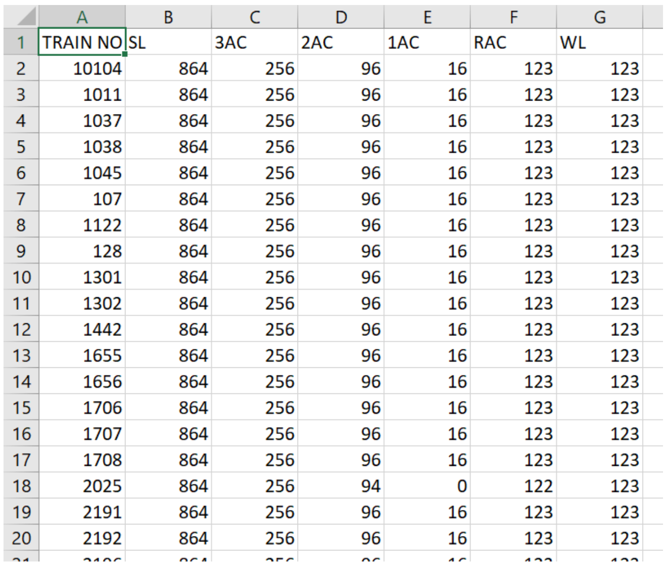


```
Enter Train Number: 2025

Enter Number of seats: 1
Coaches Available:
1AC
2AC
3AC
SL

Enter Coach Preference : 1AC

We apologize for the inconvenience! Not enough 1AC seats available at the moment
Do you want to go to rac
1.yes
2.no
Enter 1 or 2 : 1

Enter name: Avanthika
Enter age: 24
Enter Berth Type (LB, MB, UB, SLB, SUB): LB

Your RAC no is: 1
Successfully booked 1 seats.
```



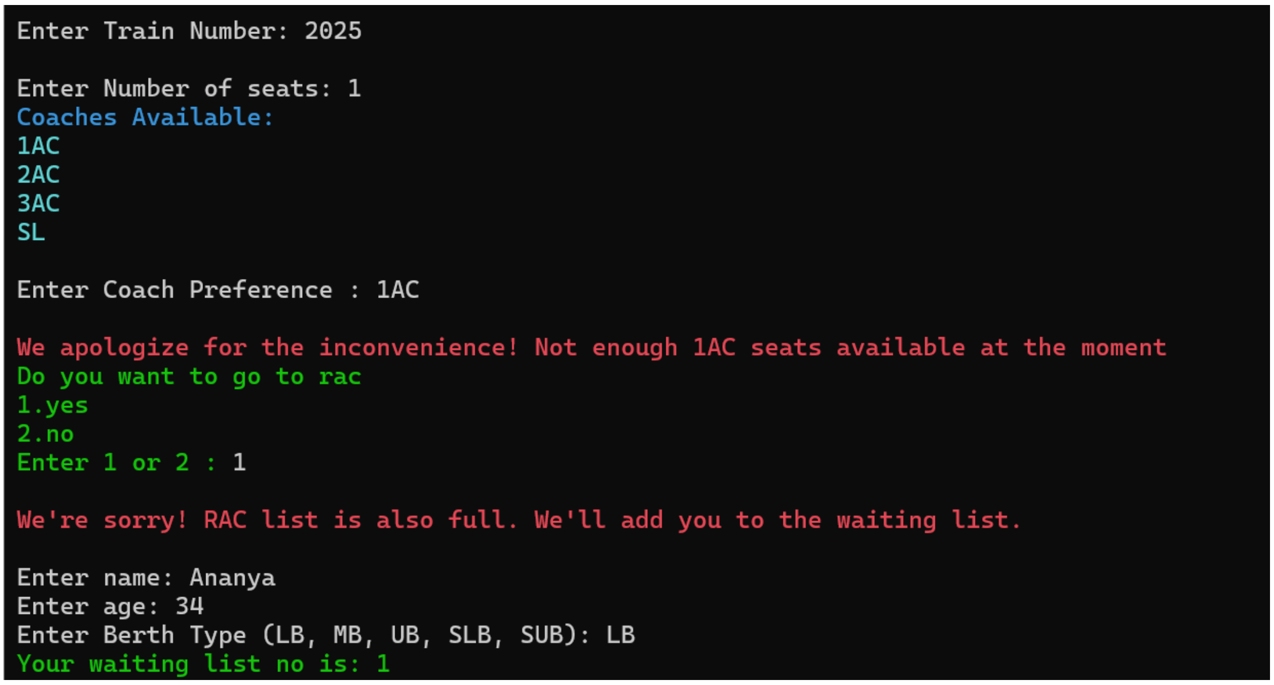
booked	coach	seatno	berth	pnr	userid	status	name	age
B	2AC	1	LB	6.27E+08	1	CNF	Anya	34
U	2AC	2	UB	9999	9999	AVL	N/A	0
B	2AC	3	LB	4.13E+08	1	CNF	Akash	22
U	1AC	4	UB	9999	9999	AVL	N/A	0
B	1AC	5	LB	2.42E+08	1	RAC	Avanthika	24

night.csv containing the night trains after booking the above seat.

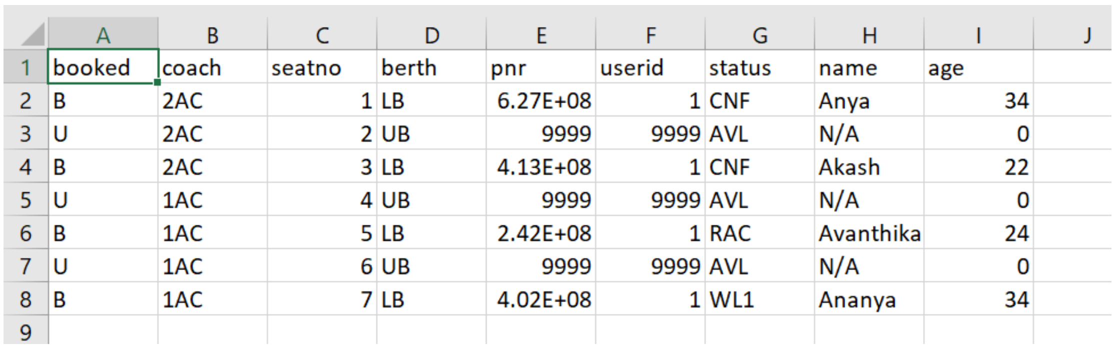


TRAIN NO	SL	3AC	2AC	1AC	RAC	WL
10104	864	256	96	16	123	123
1011	864	256	96	16	123	123
1037	864	256	96	16	123	123
1038	864	256	96	16	123	123
1045	864	256	96	16	123	123
107	864	256	96	16	123	123
1122	864	256	96	16	123	123
128	864	256	96	16	123	123
1301	864	256	96	16	123	123
1302	864	256	96	16	123	123
1442	864	256	96	16	123	123
1655	864	256	96	16	123	123
1656	864	256	96	16	123	123
1706	864	256	96	16	123	123
1707	864	256	96	16	123	123
1708	864	256	96	16	123	123
2025	864	256	94	0	122	123
2191	864	256	96	16	123	123
2192	864	256	96	16	123	123

Test case:3) WL available

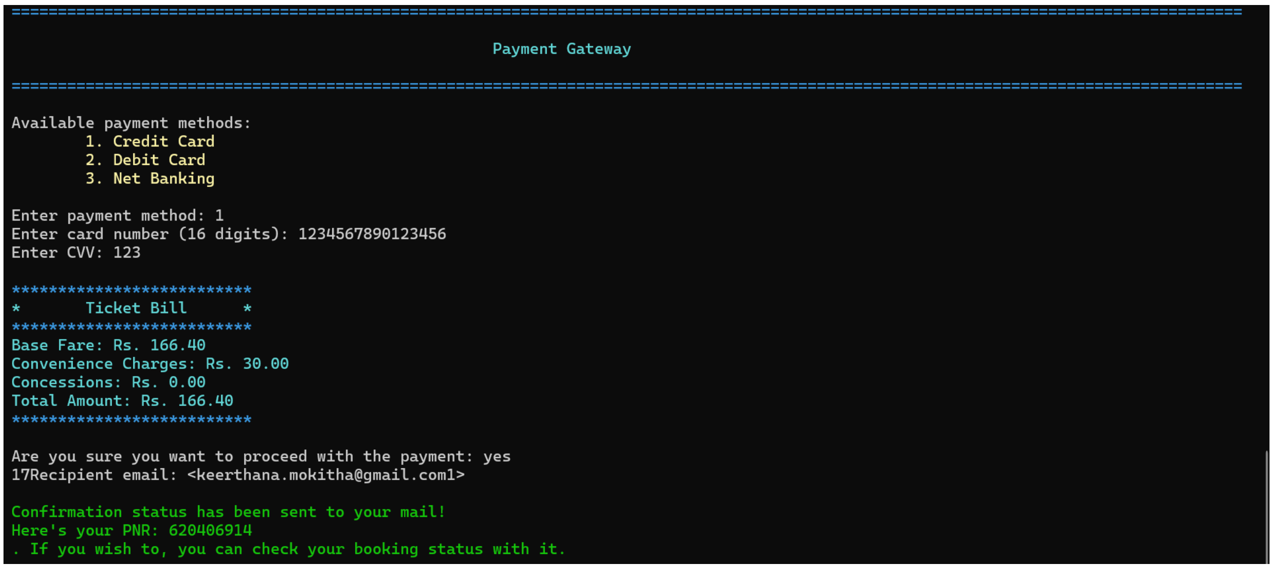


```
Enter Train Number: 2025

Enter Number of seats: 1
Coaches Available:
1AC
2AC
3AC
SL

Enter Coach Preference : 1AC

We apologize for the inconvenience! Not enough 1AC seats available at the moment
Do you want to go to rac
1.yes
2.no
Enter 1 or 2 : 1

We're sorry! RAC list is also full. We'll add you to the waiting list.

Enter name: Ananya
Enter age: 34
Enter Berth Type (LB, MB, UB, SLB, SUB): LB
Your waiting list no is: 1
```



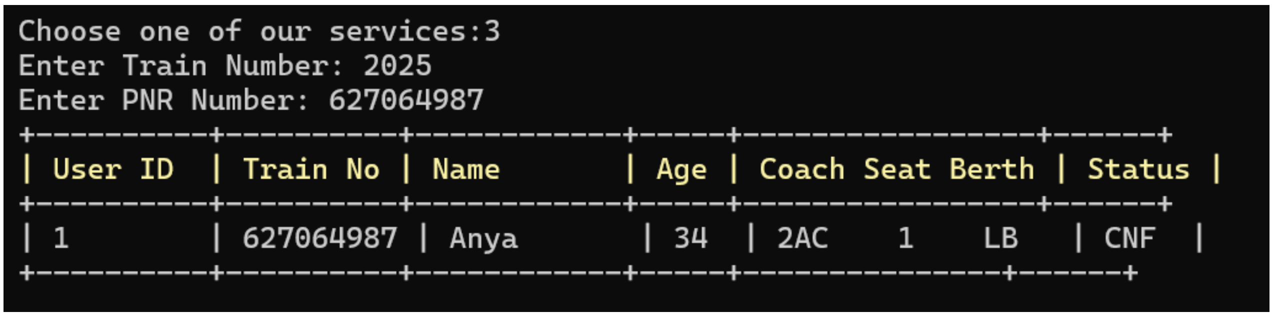
	A	B	C	D	E	F	G	H	I	J
1	booked	coach	seatno	berth	pnr	userid	status	name	age	
2	B	2AC	1	LB	6.27E+08	1	CNF	Anya	34	
3	U	2AC	2	UB	9999	9999	AVL	N/A	0	
4	B	2AC	3	LB	4.13E+08	1	CNF	Akash	22	
5	U	1AC	4	UB	9999	9999	AVL	N/A	0	
6	B	1AC	5	LB	2.42E+08	1	RAC	Avanthika	24	
7	U	1AC	6	UB	9999	9999	AVL	N/A	0	
8	B	1AC	7	LB	4.02E+08	1	WL1	Ananya	34	
9										

Payment gateway:



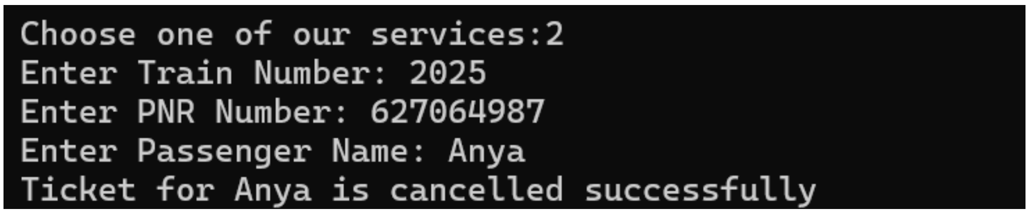
```
==========================================================================================================================

                                                  Payment Gateway

==========================================================================================================================

Available payment methods:
        1. Credit Card
        2. Debit Card
        3. Net Banking

Enter payment method: 1
Enter card number (16 digits): 1234567890123456
Enter CVV: 123

**************************
*       Ticket Bill      *
**************************
Base Fare: Rs. 166.40
Convenience Charges: Rs. 30.00
Concessions: Rs. 0.00
Total Amount: Rs. 166.40
**************************

Are you sure you want to proceed with the payment: yes
17Recipient email: <keerthana.mokitha@gmail.com1>

Confirmation status has been sent to your mail!
Here's your PNR: 620406914
. If you wish to, you can check your booking status with it.
```

Check booking status



```
Choose one of our services:3
Enter Train Number: 2025
Enter PNR Number: 627064987
+----------+----------+------------+-----+-----------------+-------+
| User ID  | Train No | Name       | Age | Coach Seat Berth | Status |
+----------+----------+------------+-----+-----------------+-------+
| 1        | 627064987 | Anya       | 34  | 2AC     1     LB   | CNF    |
+----------+----------+------------+-----+-----------------+-------+
```

Cancel



```
Choose one of our services:2
Enter Train Number: 2025
Enter PNR Number: 627064987
Enter Passenger Name: Anya
Ticket for Anya is cancelled successfully
```

LIMITATIONS:

- **Security Vulnerabilities**:

  The system may have security vulnerabilities, such as lack of input validation, potential buffer overflow issues, or inadequate protection against SQL injection attacks. This could compromise user data or system integrity.

- **Data Handling and Storage**:

  Data handling and storage methods may be simplistic, leading to potential inefficiencies in data retrieval or manipulation. For example, storing user details in a CSV file may not be scalable for large user databases.

- **Dependency on External Libraries**:

  The system relies on external libraries like CURL for functionalities such as email confirmation, which may introduce dependencies and compatibility issues, especially when deploying on different platforms or environments.

- **Limited Functionality**:

  While the system allows basic functions like booking tickets, canceling tickets, and checking booking status, it may lack advanced features found in commercial railway reservation systems. For example, dynamic seat mapping, real-time train tracking, or integration with payment gateways.

- **User Interface:**

  C lacks native support for graphical user interfaces (GUIs). Creating an intuitive and user-friendly interface is cumbersome. The system may rely on a command-line interface (CLI), which can be less appealing to users accustomed to modern GUIs.

OBSERVATIONS FROM SOCIETAL,LEGAL , ENVIRONMENTAL AND ETHICAL PERSPECTIVES :

1. **Societal Perspective**:

   User Convenience: The system's simplicity may be appreciated by users who prefer straightforward interactions without complex graphical interfaces. However, it may also deter users who expect more modern and intuitive booking experiences.

2. **Legal Perspective**:

   Data Privacy: The system's handling of user data, such as storing user details in a CSV file, may raise concerns about data privacy and compliance with data protection regulations. Implementing robust data encryption, access controls, and privacy policies would be essential to address these legal considerations.

3. **Environmental Perspective**:

   Digital Footprint: The system's reliance on online communication, email confirmations, and digital transactions contributes to its digital footprint. Implementing measures to reduce energy consumption, optimize data storage, and promote eco-friendly practices in system operations can contribute to environmental sustainability.

4. **Ethical Perspective**:

Accessibility for Vulnerable Groups: Ensuring that the system is accessible and accommodating to vulnerable groups such as senior citizens, persons with disabilities, or individuals with limited access to technology.

LEARNING OUTCOMES:

- Improved proficiency in programming languages such as C/C++ and familiarity with development environments like Code::Blocks and Replit.
- Acquired knowledge of data structures, file handling, input/output operations, and integration of external libraries (e.g., CURL) into software projects.
- Developed effective communication skills, teamwork abilities, and project management techniques through collaborative work with team members.
- Enhanced problem-solving skills by identifying, analyzing, and addressing technical challenges, bugs, and system issues encountered during the project
- Embraced challenges, learned from mistakes, and demonstrated resilience in overcoming obstacles to achieve project objectives and deliver high-quality software solutions.

REFERENCES:

- https://www.trainman.in/faqs#google_vignette
- https://www.kaggle.com/code/vivank/train-waiting-list-clearing-prediction
- https://www.cleartrip.com/flight-booking/first-air-airlines-flight-pnr-status.html#:~:text=The%20PNR%20number%20may%20be,to%20bookings%2C%20not%20individual%20passengers
- IRCTC Next Generation eTicketing System
- Indian Railways Enquiry PNR Status Live Status IRCTC Reservation Seats (erail.in)