Railway Reservation System
UCS2265 – Fundamentals and Practice of Software Development

PROJECT REPORT
RAILWAY RESERVATION SYSTEM

Submitted By
Shailesh S - 3122 23 5001 121
Saipranav M - 3122 23 5001 110
Rahul V S - 3122 23 5001 104

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

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 "SAIPRANAV M (3122235001110),SHAILESH S (3122235001121) and RAHUL V S (3122235001104)" 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: 19/06/2024

Problem Statement:

Develop a software system for reservation of railway tickets. The Reservation system should contain the following features:

- **PNR Status Check:** Passengers can check PNR status (confirmed, RAC, waiting list).
- **Data Storage:** Store train, fare, PNR, and reservation details.
- **Cancellation:** Cancel tickets with PNR number and request.
- **Refund Rules:** Admin controls refund based on reservation date and fare.
- **Admin Control:** Manages system maintenance, including train and fare details.

Constraints Night Trains:

- 12 sleeper class coaches with 72 berths each.
- 4 AC 3 tier coaches with 64 berths each.
- 3 AC 2 tier coaches with 32 berths each.
- 1 AC First class with 16 berths.

Day Trains:

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

Extended Exploration of the Problem Statement

We have explored the problem statement and have implemented safety measures in traveling for women passengers, feedback from users with sentimental analysis, sending e-ticket to users via registered email, layout of the berths are shown for convenient booking.

Data Flow Diagrams:

Level 0:

Components and Data Flow:

1. **User:**
    - PNR Status Check: Users can send requests to the railway reservation software to check the status of their PNR (Passenger Name Record). The software processes the request and returns the PNR status to the user.
    - Data Storage: Users interact with the railway reservation software to store and retrieve train, fare, PNR, and reservation details.
    - Cancellation: Users can request to cancel tickets using their PNR number. The software processes the cancellation request and confirms the cancellation to the user.
2. **Railway Reservation Software:**
    - Processing Requests: The software handles various user requests such as PNR status check, data storage/retrieval, and ticket cancellation.

- Admin Control: The admin can interact with the software to manage system maintenance tasks, including updating train and fare details, and setting refund rules.

3. **Admin:**
   - System Maintenance: The admin has the authority to manage the system, ensuring it operates smoothly. This includes updating train schedules, fare details, and handling refund policies.

Data Flow:

- **User to Railway Reservation Software:**
  - Users send requests for PNR status checks, data storage, and ticket cancellation to the railway reservation software.
  - The software processes these requests and sends back the required information or confirmation to the user.
- **Railway Reservation Software to User:**
  - The software provides responses to user requests, such as PNR status updates, confirmation of stored data, and cancellation confirmations.
- **Admin to Railway Reservation Software:**
  - The admin sends commands to the software for system maintenance tasks, including updates to train and fare details and management of refund rules.
- **Railway Reservation Software to Admin:**
  - The software may provide feedback or confirmation to the admin regarding the execution of maintenance tasks.

Level 1:

Components and Data Flow:

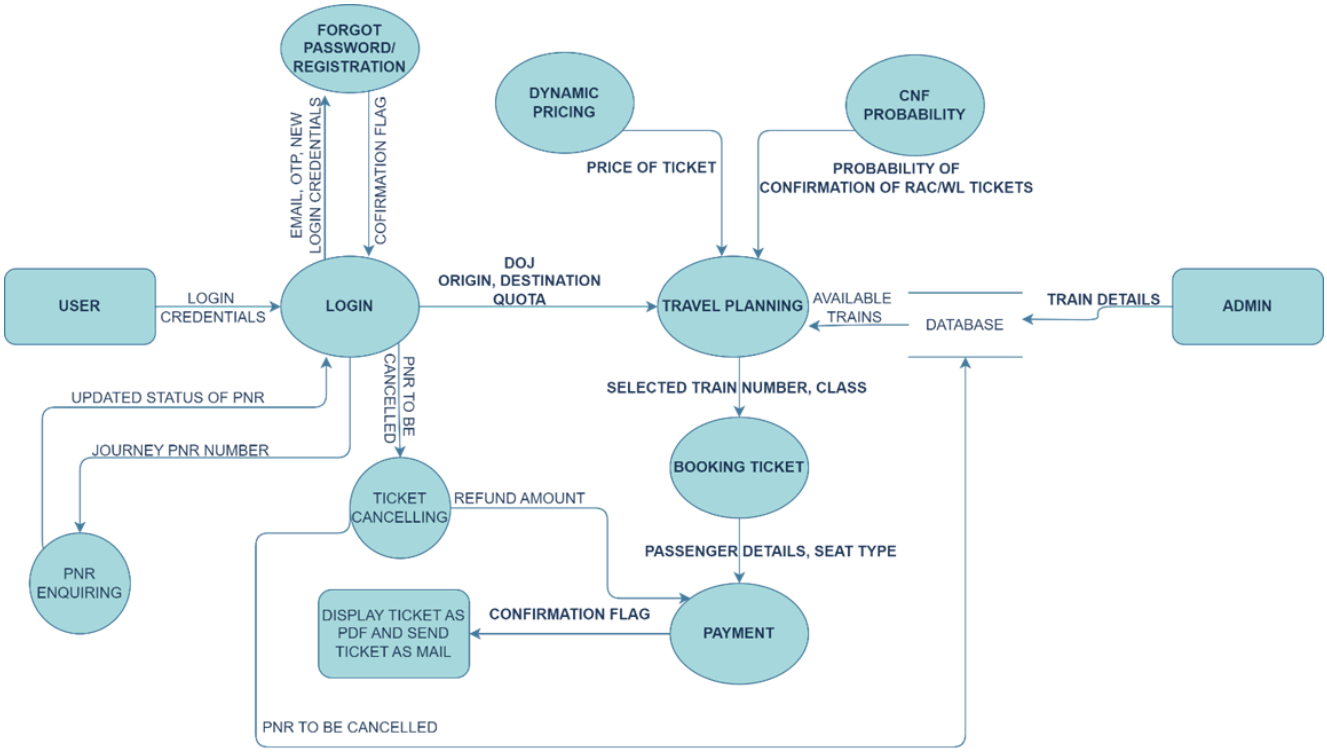
1. **User**:
    - **Login Credentials**: Users log in to the system using their credentials.
    - **PNR Enquiring**: Users can check the status of their PNR (Passenger Name Record) after logging in.
    - **Journey PNR Number**: Users input their PNR number to inquire about its status.
    - **Ticket Cancelling**: Users can cancel tickets by providing the PNR number.
    - **Refund Amount**: Upon cancellation, users receive information about the refund amount.
    - **Email and Ticket**: Users receive their tickets via email in PDF format after booking.
2. **Admin**:
    - **Train Details**: Admins provide and update train details in the database, which the system uses for travel planning and booking.
3. **Forgot Password/Registration**:
    - **Email, OTP, New Login Credentials**: Users who forgot their password or need to register can use this process to get a confirmation flag to log in.

Processes:

1. **Login**:
    - Users enter their credentials to log in.
    - Users can reset their password or register if they don't have an account.
    - Upon successful login, users can proceed with other actions like ticket booking, PNR inquiry, or ticket cancellation.

2. **Travel Planning**:
    - **DOJ, Origin, Destination, Quota**: Users provide the date of journey (DOJ), origin, destination, and quota to plan their travel.
    - **Available Trains**: The system retrieves available trains based on the provided details.
    - **Price of Ticket**: The system calculates ticket prices using dynamic pricing.
    - **Probability of Confirmation**: The system checks the probability of RAC/WL ticket confirmation.
3. **Booking Ticket**:
    - **Selected Train Number, Class**: Users select the train number and class for booking.
    - **Passenger Details, Seat Type**: Users provide passenger details and select the seat type.
    - **Confirmation Flag**: The system confirms the booking and sends the confirmation flag to the user.
4. **Payment**:
    - Users proceed with the payment after selecting the train and providing passenger details.
    - Upon successful payment, the system confirms the booking.
5. **Ticket Cancelling**:
    - **PNR to be Cancelled**: Users provide the PNR number of the ticket to be cancelled.
    - **Refund Amount**: The system calculates the refund amount based on cancellation rules and provides this information to the user.

6. **Display Ticket as PDF and Send Ticket as Mail**:
    - After booking, the system generates the ticket as a PDF and sends it to the user's email.


FORGOT PASSWORD/ REGISTRATION
DYNAMIC PRICING
CNF PROBABILITY
EMAIL, OTP, NEW LOGIN CREDENTIALS
COFIRMATION FLAG
PRICE OF TICKET
PROBABILITY OF CONFIRMATION OF RAC/WL TICKETS
USER
LOGIN CREDENTIALS
LOGIN
DOJ ORIGIN, DESTINATION QUOTA
TRAVEL PLANNING
AVAILABLE TRAINS
DATABASE
TRAIN DETAILS
ADMIN
UPDATED STATUS OF PNR
PNR TO BE CANCELLED
SELECTED TRAIN NUMBER, CLASS
JOURNEY PNR NUMBER
BOOKING TICKET
TICKET CANCELLING
REFUND AMOUNT
PASSENGER DETAILS, SEAT TYPE
PNR ENQUIRING
DISPLAY TICKET AS PDF AND SEND TICKET AS MAIL
CONFIRMATION FLAG
PAYMENT
PNR TO BE CANCELLED

Level 2:

User Registration and Login

1. **Registering**:
    - User details and credentials are input during the registration process.
    - The system verifies the user's email through an OTP (One-Time Password).
    - Once the email is verified, the credentials are saved in the login database.
2. **Logging In**:
    - Users input their email and password.
    - The system verifies the credentials against the login database.
    - If successful, a login flag is set; otherwise, a fail login flag is triggered.
    - Users can reset their password if needed, updating their credentials in the database.

Travel Planning and Ticket Booking

3. **Travel Planning**:
    - Users provide details like Date of Journey (DOJ), origin, and destination.
    - The system checks the available train details from the train details database.
    - The user selects a train based on the provided options.
4. **Booking**:
    - Users provide passenger details (name, age, gender, birth preference).
    - The system calculates the ticket cost and proceeds to payment.

Payment and Ticket Confirmation

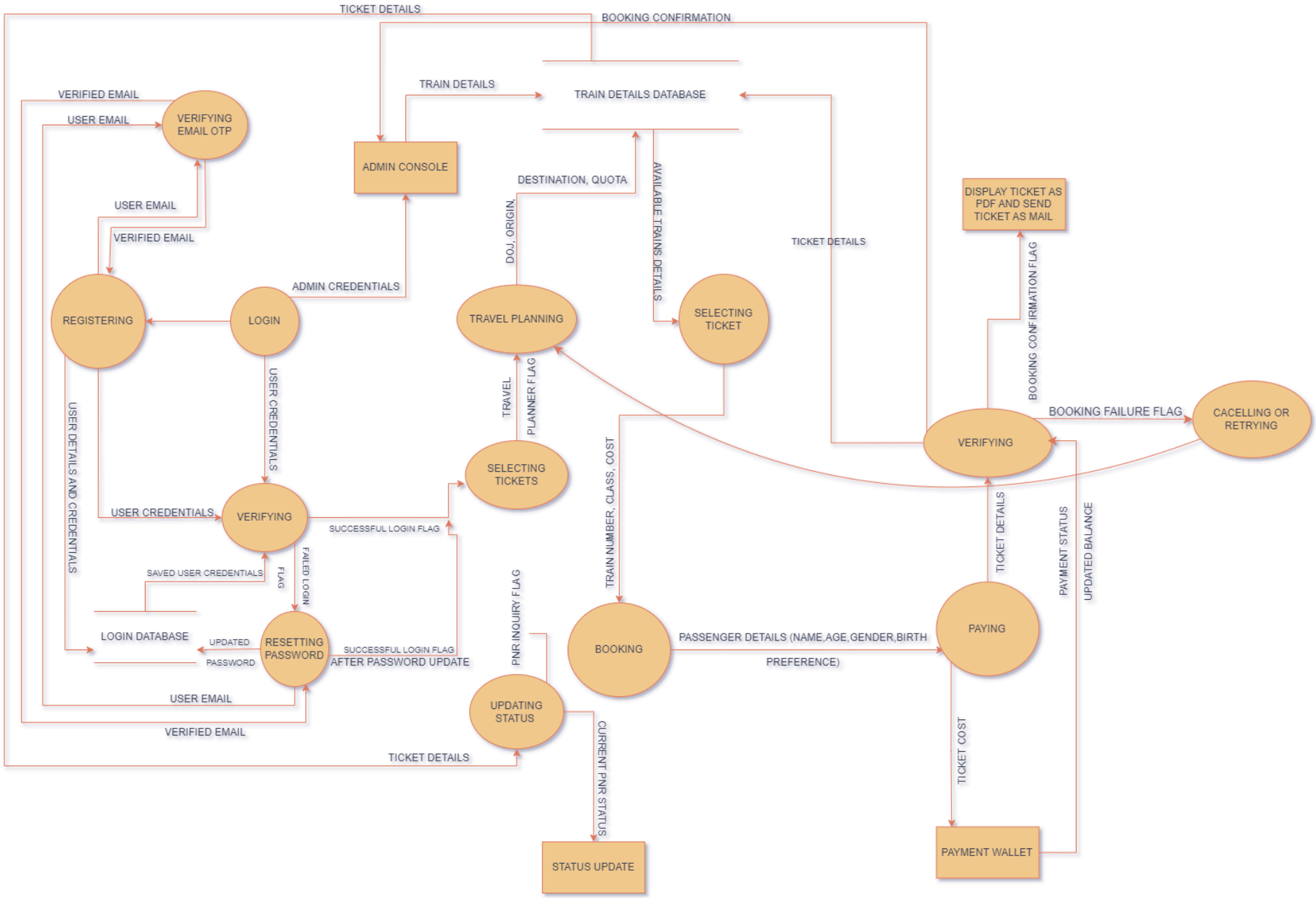
5. **Ticket Paying**:
    - Users make payments via a payment wallet.
    - If there are insufficient funds, an updating balance process is initiated.
    - Upon successful payment, the ticket details are verified, and the system confirms the booking.
6. **Verifying Ticket**:
    - The system verifies the ticket details and PNR (Passenger Name Record) of the booked ticket.
    - Once verified, the ticket is displayed as a PDF and sent via email.

Admin Console and Status Checking

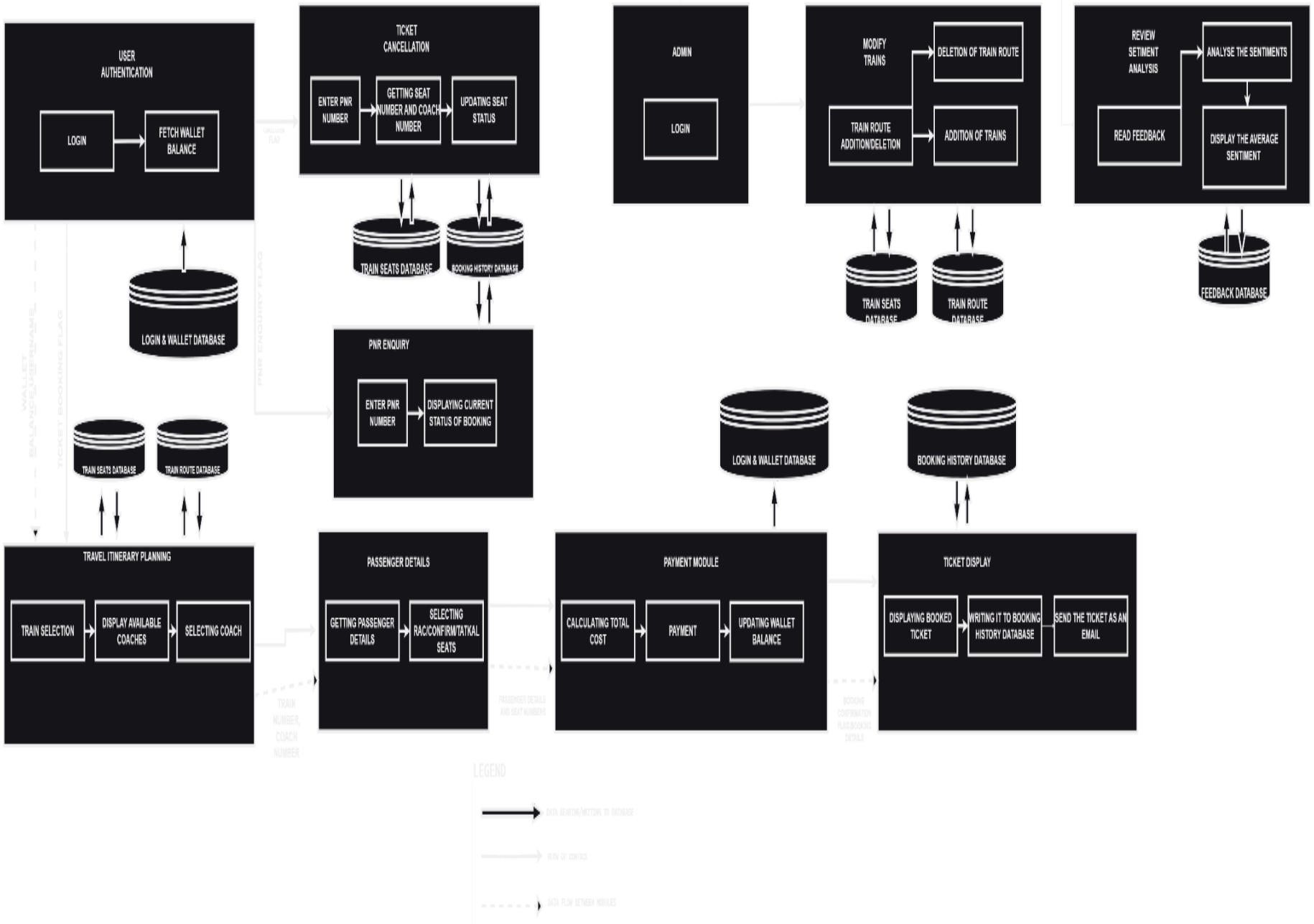
7. **Admin Console**:
    - Admins can access train details, update booking confirmation, and manage train schedules.
8. **PNR Status Checking**:
    - Users can check the status of their PNR.
    - The system updates the status based on current data and provides confirmation probabilities.

Ticket Cancellation

9. **Ticket Cancelling**:
    - Users can cancel their tickets by providing the PNR number.
    - The system processes the cancellation and refunds the money.


TICKET DETAILS
BOOKING CONFIRMATION
VERIFIED EMAIL
USER EMAIL
VERIFYING EMAIL OTP
TRAIN DETAILS
TRAIN DETAILS DATABASE
ADMIN CONSOLE
DESTINATION, QUOTA
DOJ, ORIGIN
AVAILABLE TRAINS DETAILS
DISPLAY TICKET AS PDF AND SEND TICKET AS MAIL
TICKET DETAILS
USER EMAIL
VERIFIED EMAIL
ADMIN CREDENTIALS
REGISTERING
LOGIN
TRAVEL PLANNING
SELECTING TICKET
BOOKING CONFIRMATION FLAG
USER DETAILS AND CREDENTIALS
USER CREDENTIALS
TRAVEL
PLANNER FLAG
BOOKING FAILURE FLAG
CACELLING OR RETRYING
SELECTING TICKETS
USER CREDENTIALS
VERIFYING
SUCCESSFUL LOGIN FLAG
TRAIN NUMBER, CLASS, COST
TICKET DETAILS
PAYMENT STATUS
UPDATED BALANCE
SAVED USER CREDENTIALS
FLAG
FAILED LOGIN
LOGIN DATABASE
UPDATED
PASSWORD
RESETTING PASSWORD
SUCCESSFUL LOGIN FLAG
AFTER PASSWORD UPDATE
PNR INQUIRY FLAG
BOOKING
PASSENGER DETAILS (NAME,AGE,GENDER,BIRTH PREFERENCE)
PAYING
USER EMAIL
UPDATING STATUS
CURRENT PNR STATUS
TICKET COST
VERIFIED EMAIL
TICKET DETAILS
STATUS UPDATE
PAYMENT WALLET

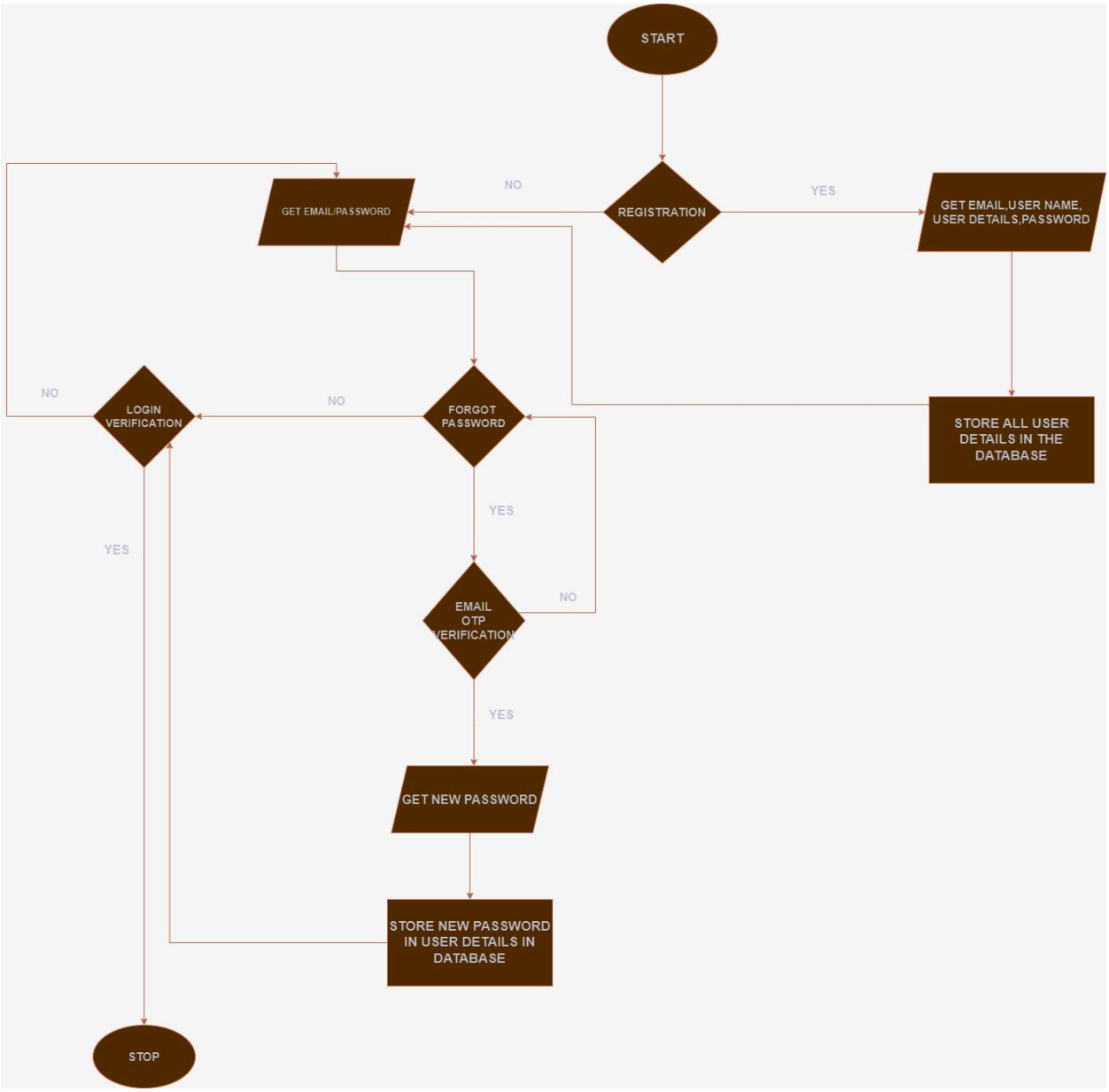
Architecture Diagram:


USER AUTHENTICATION
LOGIN
FETCH WALLET BALANCE
TICKET CANCELLATION
ENTER PNR NUMBER
GETTING SEAT NUMBER AND COACH NUMBER
UPDATING SEAT STATUS
ADMIN
LOGIN
MODIFY TRAINS
DELETION OF TRAIN ROUTE
TRAIN ROUTE ADDITION/DELETION
ADDITION OF TRAINS
REVIEW SETIMENT ANALYSIS
ANALYSE THE SENTIMENTS
READ FEEDBACK
DISPLAY THE AVERAGE SENTIMENT
LOGIN & WALLET DATABASE
TRAIN SEATS DATABASE
BOOKING HISTORY DATABASE
TRAIN SEATS DATABASE
TRAIN ROUTE DATABASE
FEEDBACK DATABASE
PNR ENQUIRY
ENTER PNR NUMBER
DISPLAYING CURRENT STATUS OF BOOKING
TRAIN SEATS DATABASE
TRAIN ROUTE DATABASE
LOGIN & WALLET DATABASE
BOOKING HISTORY DATABASE
TRAVEL ITINERARY PLANNING
TRAIN SELECTION
DISPLAY AVAILABLE COACHES
SELECTING COACH
PASSENGER DETAILS
GETTING PASSENGER DETAILS
SELECTING RAC/CONFIRM/TATKAL SEATS
PAYMENT MODULE
CALCULATING TOTAL COST
PAYMENT
UPDATING WALLET BALANCE
TICKET DISPLAY
DISPLAYING BOOKED TICKET
WRITING IT TO BOOKING HISTORY DATABASE
SEND THE TICKET AS AN EMAIL
LEGEND

Module Flowcharts:

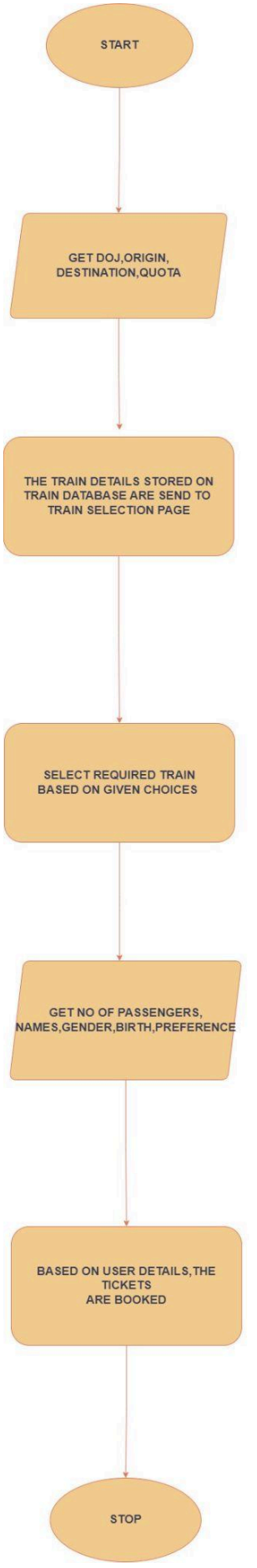
1. User authentication module:

authenticateUser(const char *username, const char *password):



2.Itinerary planning Module:

- **readTrainData(struct Availability *trains, const char *filename)**
- **showAvailableTrains(struct Availability *trains, int num_trains, char *from, char *to, char *date)**

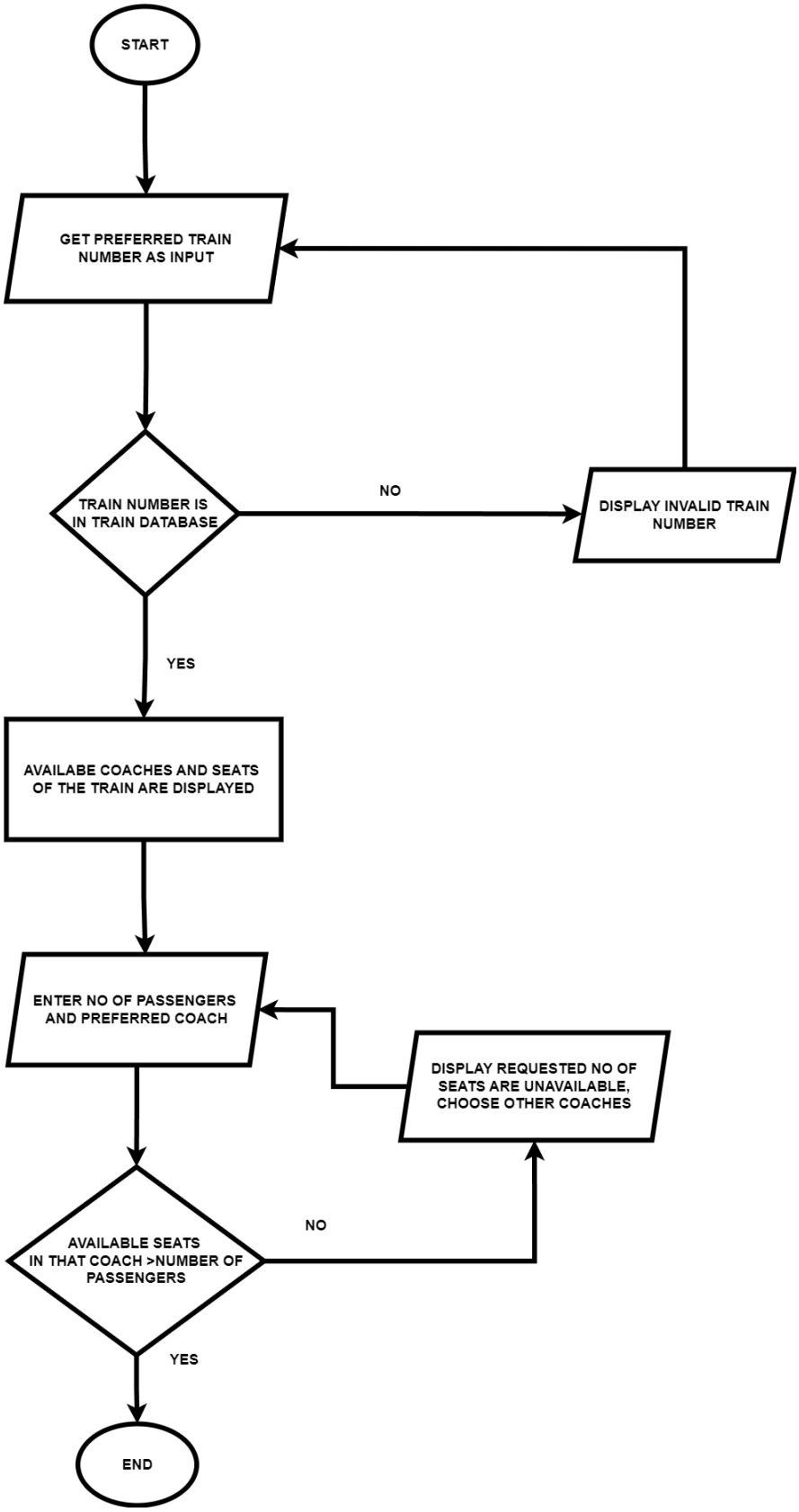

START
GET DOJ,ORIGIN,
DESTINATION,QUOTA
THE TRAIN DETAILS STORED ON
TRAIN DATABASE ARE SEND TO
TRAIN SELECTION PAGE
SELECT REQUIRED TRAIN
BASED ON GIVEN CHOICES
GET NO OF PASSENGERS,
NAMES,GENDER,BIRTH,PREFERENCE
BASED ON USER DETAILS,THE
TICKETS
ARE BOOKED
STOP

3.Seat planning Module:

- **calculateDaysDifference(struct tm date1, struct tm date2)**

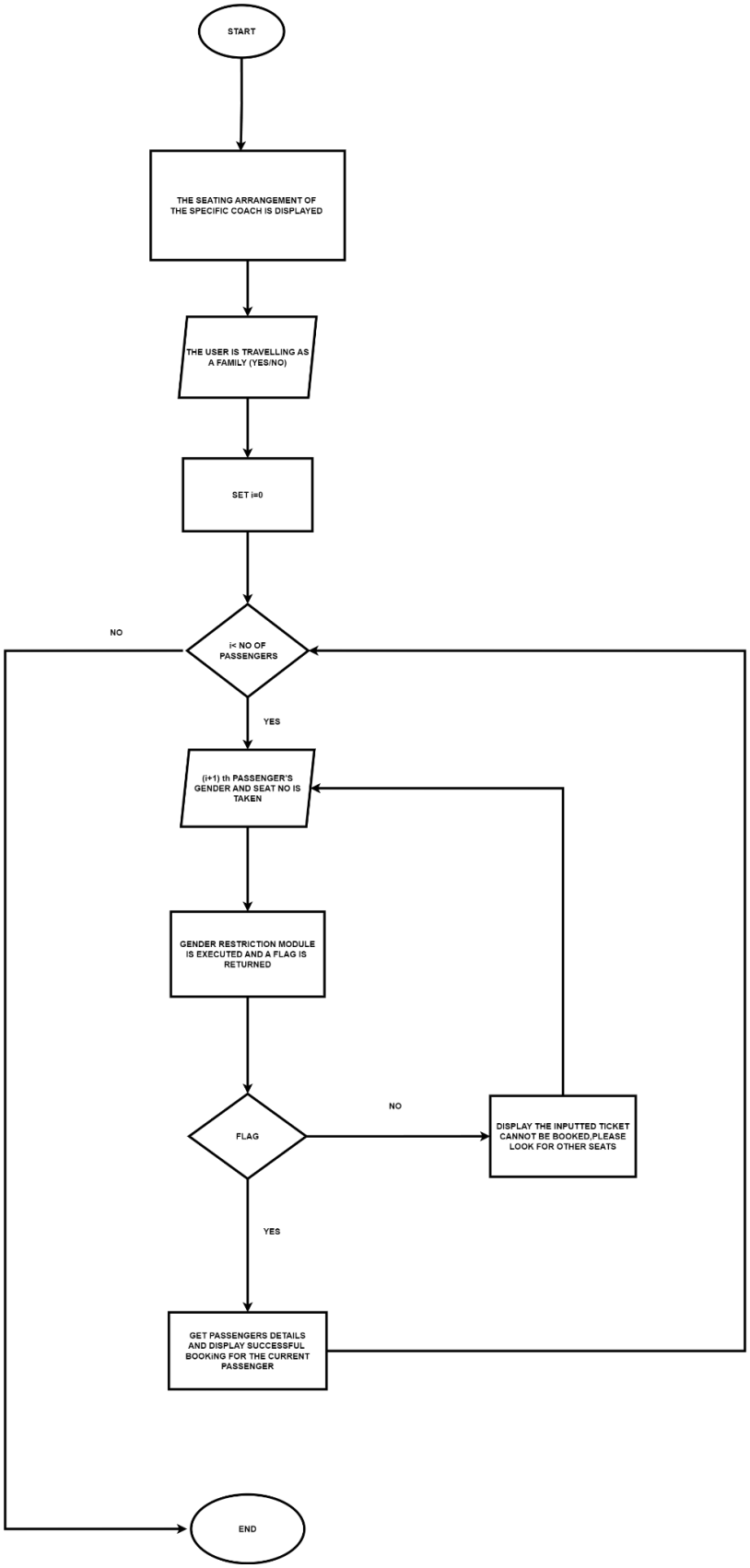
- **displayChart(int coach, int seats[TOTAL_COACHES][SEATS_S], int genderSeats[TOTAL_COACHES][SEATS_S], int seatCount, int rows, int columnsLeft, int columnsRight, struct tm journeyDate)**

- **countAvailableSeats(int coach, int seats[TOTAL_COACHES][SEATS_S], int seatCount, struct tm journeyDate)**


START
GET PREFERRED TRAIN NUMBER AS INPUT
TRAIN NUMBER IS IN TRAIN DATABASE
NO
DISPLAY INVALID TRAIN NUMBER
YES
AVAILABE COACHES AND SEATS OF THE TRAIN ARE DISPLAYED
ENTER NO OF PASSENGERS AND PREFERRED COACH
DISPLAY REQUESTED NO OF SEATS ARE UNAVAILABLE, CHOOSE OTHER COACHES
AVAILABLE SEATS IN THAT COACH >NUMBER OF PASSENGERS
NO
YES
END

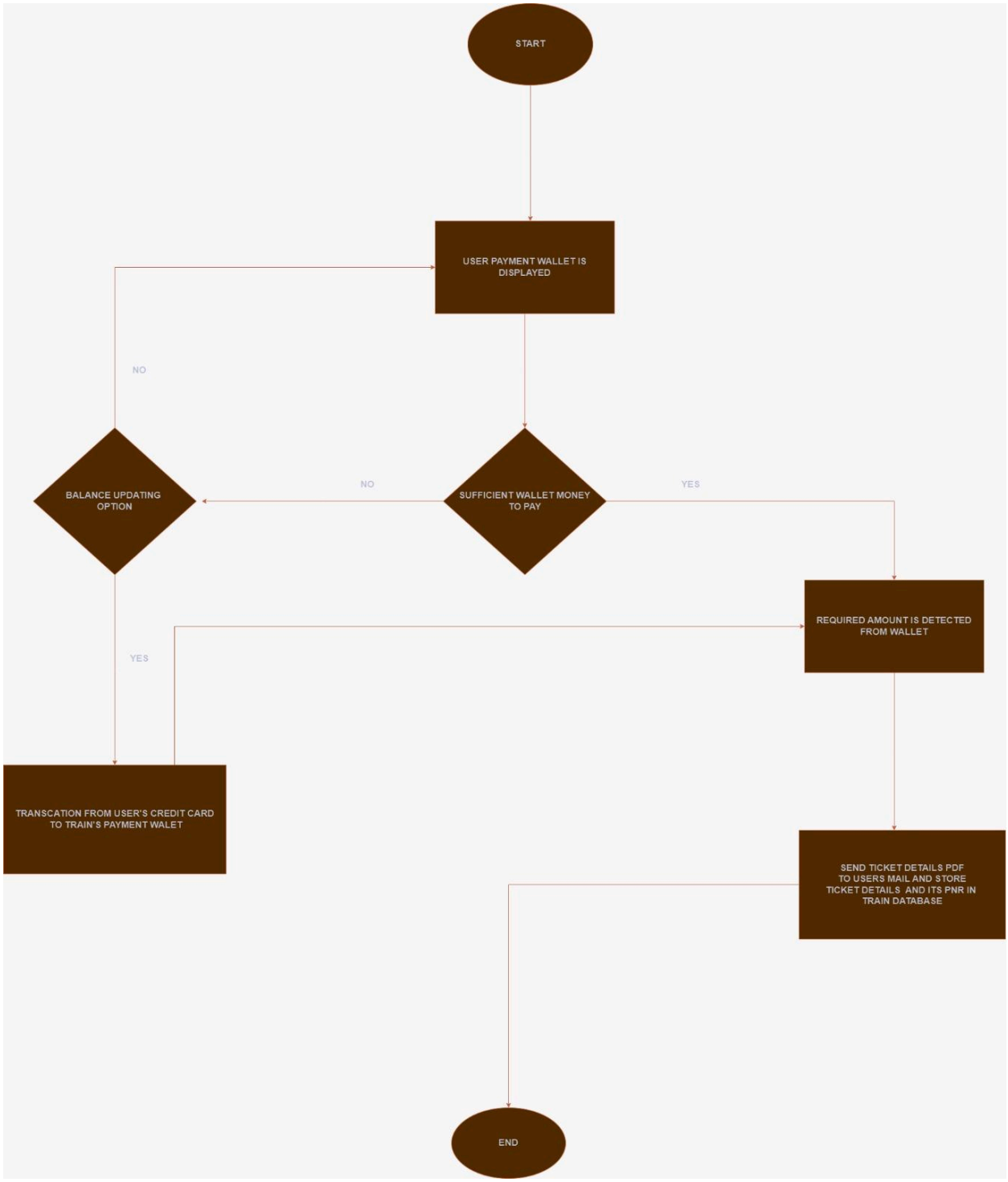
4. Passenger details Module:

- **canBookSeat(int coach, int seatIndex, int gender, int genderSeats[TOTAL_COACHES][SEATS_S], int seatCount, struct tm journeyDate, char fam[])**


START
THE SEATING ARRANGEMENT OF THE SPECIFIC COACH IS DISPLAYED
THE USER IS TRAVELLING AS A FAMILY (YES/NO)
SET i=0
i< NO OF PASSENGERS
NO
YES
(i+1) th PASSENGER'S GENDER AND SEAT NO IS TAKEN
GENDER RESTRICTION MODULE IS EXECUTED AND A FLAG IS RETURNED
FLAG
NO
DISPLAY THE INPUTTED TICKET CANNOT BE BOOKED,PLEASE LOOK FOR OTHER SEATS
YES
GET PASSENGERS DETAILS AND DISPLAY SUCCESSFUL BOOKING FOR THE CURRENT PASSENGER
END

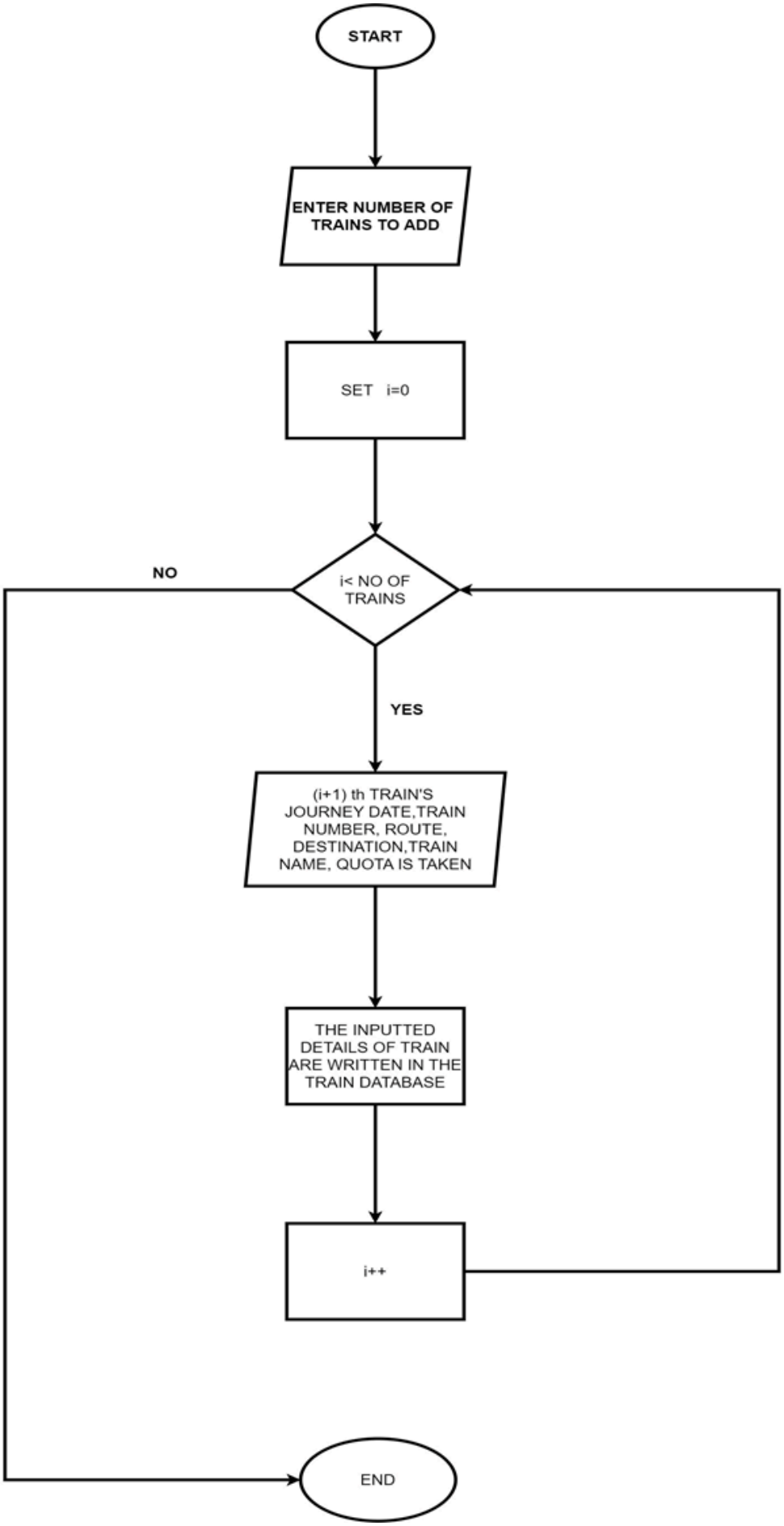
5. Payment Module:

- **stationsCrossed(char *route, char *from, char *to)**
- **updateWalletAmount(const char *username, int newWalletAmount)**
- **calculateTotalCost(int stationsCrossed, int coachType)**
- **void send_email(TravelPlan *plan, const char *to_email)**


START
USER PAYMENT WALLET IS DISPLAYED
NO
BALANCE UPDATING OPTION
NO
SUFFICIENT WALLET MONEY TO PAY
YES
REQUIRED AMOUNT IS DETECTED FROM WALLET
YES
TRANSCATION FROM USER'S CREDIT CARD TO TRAIN'S PAYMENT WALET
SEND TICKET DETAILS PDF TO USERS MAIL AND STORE TICKET DETAILS AND ITS PNR IN TRAIN DATABASE
END

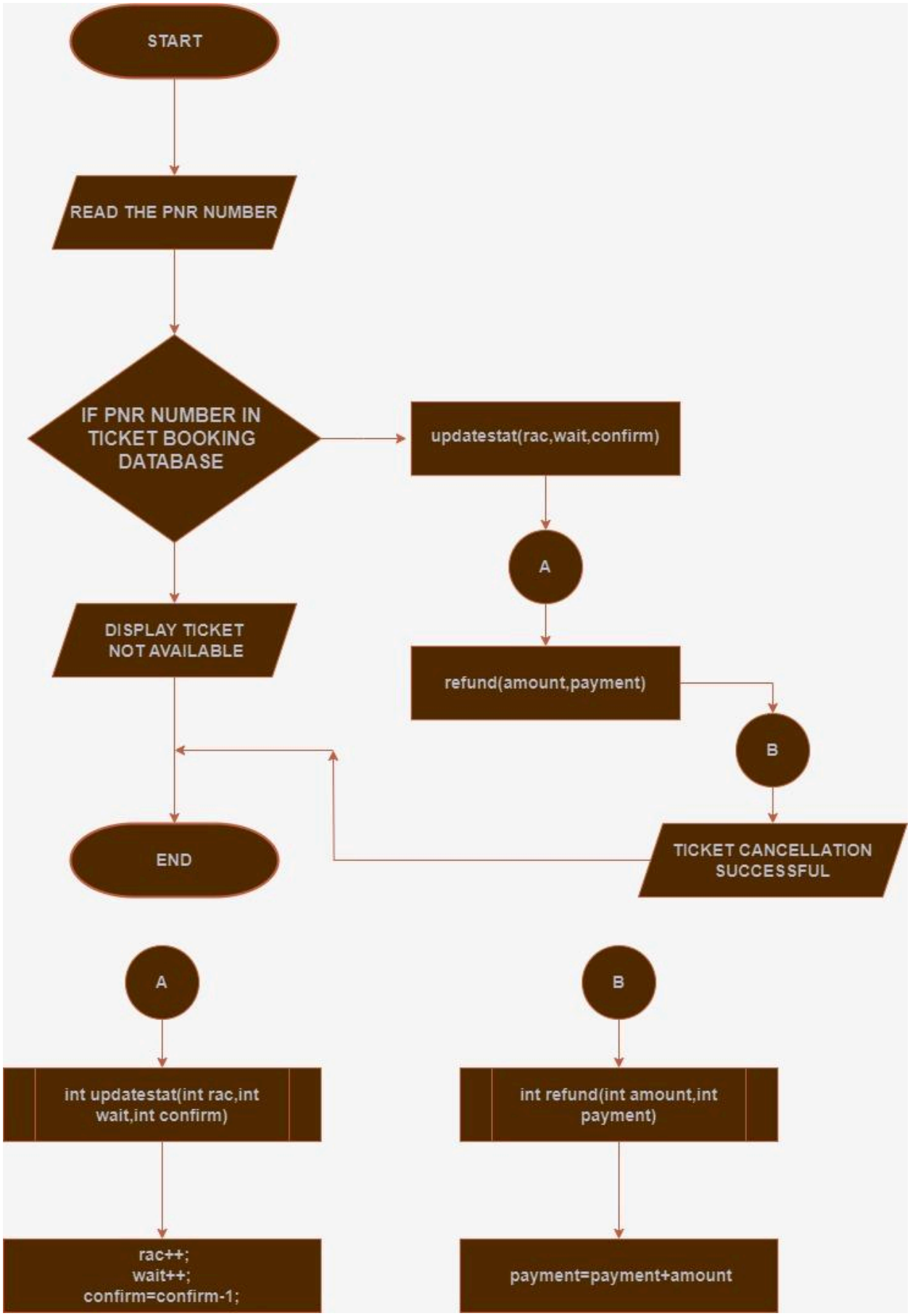
6.Adding a new train Module:

- **getSeatLimit(int coach)**
- **initializeSeats(struct seats *s)**
- **writeTrainDetailsToCSV(struct seats *s, const char *route, const char *destination, const char *trainName, char quota[])**


START
ENTER NUMBER OF TRAINS TO ADD
SET i=0
i< NO OF TRAINS
NO
YES
(i+1) th TRAIN'S JOURNEY DATE,TRAIN NUMBER, ROUTE, DESTINATION,TRAIN NAME, QUOTA IS TAKEN
THE INPUTTED DETAILS OF TRAIN ARE WRITTEN IN THE TRAIN DATABASE
i++
END

7.Cancellation Module:

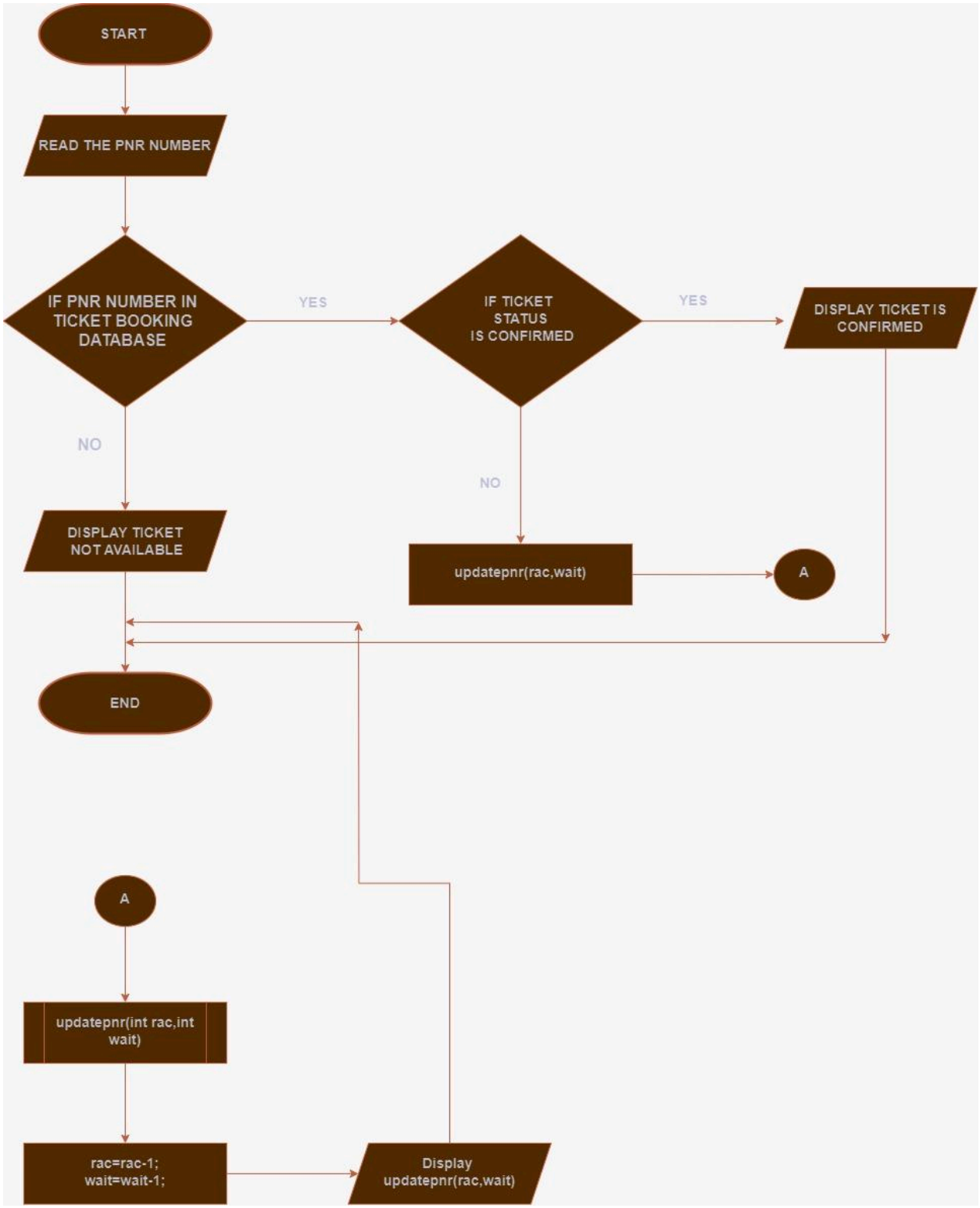
- **unbookSeat(int trainno, int coach, int seat)**


START
READ THE PNR NUMBER
IF PNR NUMBER IN TICKET BOOKING DATABASE
updatestat(rac,wait,confirm)
A
refund(amount,payment)
B
TICKET CANCELLATION SUCCESSFUL
DISPLAY TICKET NOT AVAILABLE
END
A
int updatestat(int rac,int wait,int confirm)
rac++;
wait++;
confirm=confirm-1;
B
int refund(int amount,int payment)
payment=payment+amount

8.PNR enquiry Module:

- **display_booking_details(struct Booking booking, struct Passenger passengers[], int passenger_count)**

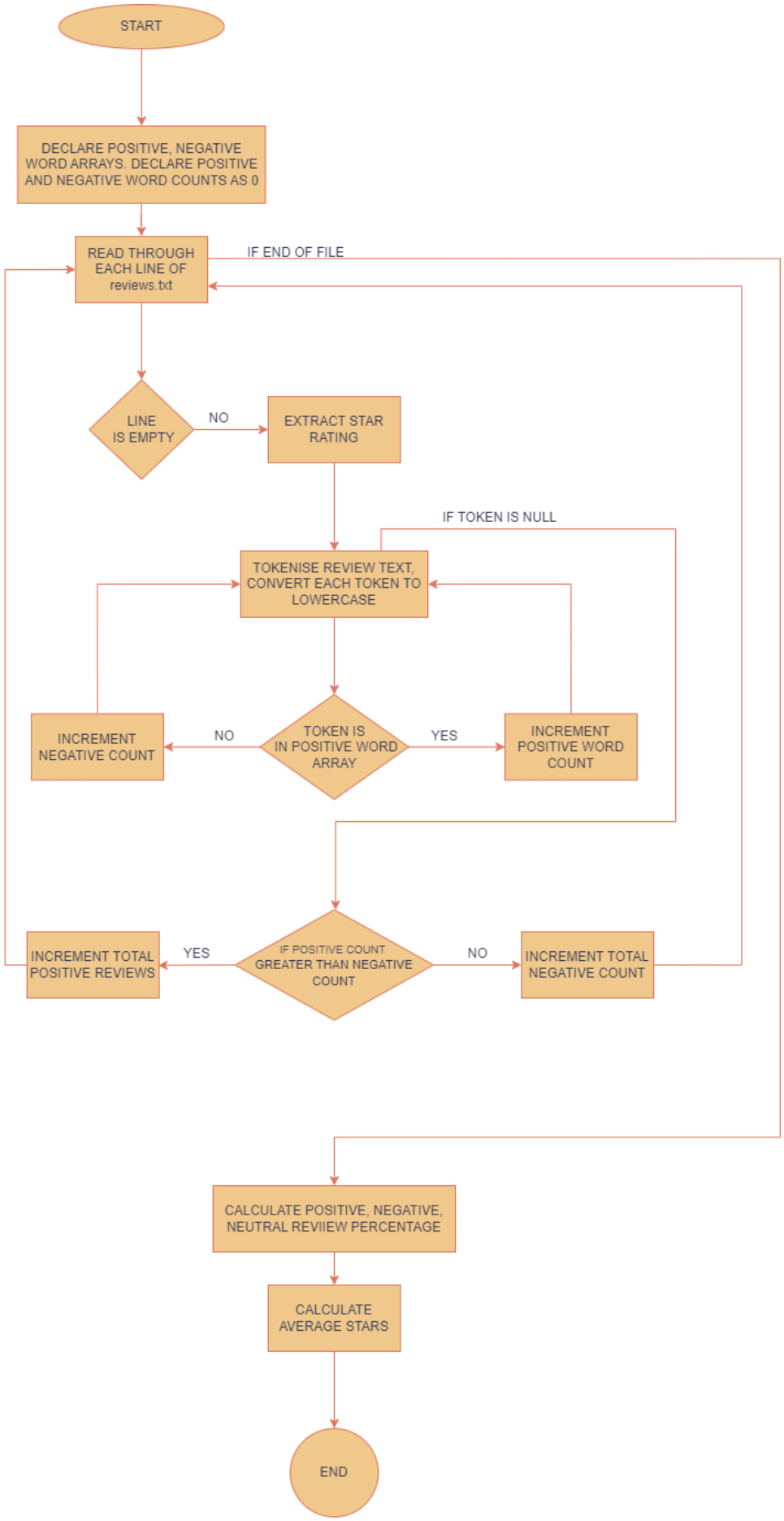
- **pnr_enquiry(const char* filename, const char* pnr_input):**


START
READ THE PNR NUMBER
IF PNR NUMBER IN TICKET BOOKING DATABASE
YES
IF TICKET STATUS IS CONFIRMED
YES
DISPLAY TICKET IS CONFIRMED
NO
NO
DISPLAY TICKET NOT AVAILABLE
updatepnr(rac,wait)
A
END
A
updatepnr(int rac,int wait)
rac=rac-1; wait=wait-1;
Display updatepnr(rac,wait)

9. Feedback Module:

- **void count_word_occurrences(const char *word, const char *positive_words[], int positive_size, const char *negative_words[], int negative_size, int *positive_counts, int *negative_counts)**

- **void add_review(const char *filename)**


START
DECLARE POSITIVE, NEGATIVE WORD ARRAYS. DECLARE POSITIVE AND NEGATIVE WORD COUNTS AS 0
READ THROUGH EACH LINE OF reviews.txt
IF END OF FILE
LINE IS EMPTY
NO
EXTRACT STAR RATING
IF TOKEN IS NULL
TOKENISE REVIEW TEXT, CONVERT EACH TOKEN TO LOWERCASE
INCREMENT NEGATIVE COUNT
NO
TOKEN IS IN POSITIVE WORD ARRAY
YES
INCREMENT POSITIVE WORD COUNT
INCREMENT TOTAL POSITIVE REVIEWS
YES
IF POSITIVE COUNT GREATER THAN NEGATIVE COUNT
NO
INCREMENT TOTAL NEGATIVE COUNT
CALCULATE POSITIVE, NEGATIVE, NEUTRAL REVIIEW PERCENTAGE
CALCULATE AVERAGE STARS
END

Modules used in this Application

1. User authentication Module
2. Itinerary planning Module
3. Seat planning Module
4. Passenger details Module
5. Payment Module
6. Adding a new train Module
7. Cancellation Module
8. PNR enquiry Module
9. Feedback Module

Description of each module

1. User authentication module:

a. To enter a username

b. To enter a password

c. To check if the inputs already exist in the login.csv file . If details are not already present in the file, add it.

d. To set the wallet amount to global wallet amount

2. Itinerary Planning Module:

a. To enter the date of journey , from place, destination and quota

b. It matches with the available trains and displays the available trains

3. Seat Planning Module:

a.To enter the preferred class, berth and family or single option

b.To display the available berths as per the details entered

4. Passenger Details Module:

a.To enter the name, age, gender and preferred berth of the passenger
b.To check if the selected seat can be booked as per the details entered

5. Payment Module:

a.To calculate the number of stations crossed
b.To update the wallet amount
c. To display the total cost and deduct the amount from the wallet
d. To send a copy of ticket to user via email

6. Adding New Train Module:

a.To enter the details of a train like coaches, types of coaches, seats, etc…
b.To add the details in train.csv file

7. Cancellation Module:

a.To enter the PNR number and verify the respective seat
b.To remove the particular ticket from seatsf.bin file

8. PNR Enquiry Module:

a.To enter the PNR number of the ticket
b.To display the booked ticket details

9. Feedback Module:

a.To get the review from the user and perform sentiment analysis

Implementation

1. User authentication module:

- **authenticateUser(const char *username, const char *password):**
  This function checks the provided username and password against records in the "login.csv" file. It reads each line, parses the username, password, email, and wallet amount, and verifies the credentials. If successful, it sets the global walletAmount and returns 1; otherwise, it returns 0.

2.Itinerary planning Module:

- **readTrainData(struct Availability *trains, const char *filename):**
  This function reads train data from a CSV file ("trains.csv") and populates an array of struct Availability with the data.It processes each line to fill in the route, destination, date of journey, train name, train number, and seat availability details.

- **showAvailableTrains(struct Availability *trains, int num_trains, char *from, char *to, char *date):**
  This function displays available trains for a given journey date and route specified by from and to. It checks each train's route and date of journey and prints relevant details if the criteria match.

3.Seat planning Module:

- **calculateDaysDifference(struct tm date1, struct tm date2):**
  This function calculates the difference in days between two struct tm dates. It is used to determine how many days are left before a journey.

- **displayChart(int coach, int seats[TOTAL_COACHES][SEATS_S], int genderSeats[TOTAL_COACHES][SEATS_S], int seatCount, int rows, int columnsLeft, int columnsRight, struct tm journeyDate):**
  This function prints a seating chart for a specified coach. It visually represents seat availability and gender information, considering booking restrictions based on days before the journey.

- **countAvailableSeats(int coach, int seats[TOTAL_COACHES][SEATS_S], int seatCount, struct tm journeyDate):**
  This function counts the number of available seats in a specified coach. It considers booking restrictions for seats close to the journey date.

- **displayAvailableSeats(int seats[TOTAL_COACHES][SEATS_S], struct tm journeyDate):**
  This function displays the number of available seats in each coach of a train. It provides an overview of seat availability across different coach types.

4. Passenger details Module:

This module gets the Name, Gender, Age as input and the makes sure that male and female passenger do not travel together unless they are a family using this following function:

- **canBookSeat(int coach, int seatIndex, int gender, int genderSeats[TOTAL_COACHES][SEATS_S], int seatCount, struct tm journeyDate, char fam[])*:**
  This function checks if a seat can be booked based on the gender of the passenger, adjacent seat gender, and booking restrictions for seats close to the journey date.

5. Payment Module:

- **stationsCrossed(char *route, char *from, char *to):**
  This function calculates the number of stations crossed between two stations specified by from and to in a given route string. It locates the positions of the stations in the route and computes the difference.

- **updateWalletAmount(const char *username, int newWalletAmount):**
  This function updates the wallet amount of a specific user in the "login.csv" file. It reads the file line by line, updates the wallet amount for the matching username, and writes the updated data to a temporary file, which then replaces the original.

- **calculateTotalCost(int stationsCrossed, int coachType):**
  This function calculates the total cost of a journey based on the number of stations crossed and the type of coach. It selects a base cost for the coach type and multiplies it by the number of stations crossed.

- **void send_email(TravelPlan *plan, const char *to_email):**
  The send_email function creates and sends an email itinerary for a travel plan. It first writes details from the plan struct (destination, date, train info, etc.) to a temporary file. Then, it loops to write details for up to 10 passengers, including name, age, gender, and seat number (skipping empty entries). After closing the file, it constructs a command that calls a Python script (emaily.py) with the recipient email and temporary file as arguments. This script reads the content and sends the email. The function checks for execution errors and cleans up by removing the temporary file after successful sending.

6.Adding a new train Module:

- **getSeatLimit(int coach):**

This function returns the seat limit for a given coach type based on the coach index. It helps in determining the valid range of seat numbers for any coach, which is necessary for validating seat numbers during booking and rebooking operations.

- **initializeSeats(struct seats *s):**
  This function initializes the seat configurations for different types of coaches within a train. It sets up the seat numbers for S (Sleeper), AC3 (3-tier AC), AC2 (2-tier AC), and AC1 (1-tier AC) coaches, filling the seats array s1 with seat numbers and gS with zero, indicating unbooked seats. It ensures that each coach is properly configured with the correct number of seats for future booking operations.

- **writeTrainDetailsToCSV(struct seats *s, const char *route, const char *destination, const char *trainName, char quota[]):**
  This function appends train details to a CSV file (trains.csv). It records information such as the train's route, destination, date of journey, train number, and the number of seats available in different coach types. This is useful for maintaining a log of train schedules and seat availability in a human-readable format.

7.Cancellation Module:

- **void removePNR(char *pnrList, const char *pnr) :**
  This function handles the removal of a specific PNR from a list of PNRs. It takes a list of PNRs separated by commas and the specific PNR to be removed as input. The function iterates through the PNR list, comparing each PNR with the one to be removed. If a match is found, it skips that PNR, effectively removing it from the list. The remaining PNRs are concatenated back into a single string. If the removed PNR is the last in the list, the function ensures the resulting string is properly terminated. This function allows for the management and updating of a PNR list by removing specific entries.

- **void updateWalletAmountAndRemovePNR(const char *username, int newWalletAmount, const char *pnrToRemove):**
  This function updates the wallet amount for a specified user and removes a specified PNR from the user's PNR list. It reads user data from a CSV file ("login.csv"), processes each line to locate the user, updates the wallet amount, and removes the specified PNR from the user's list. The updated data is written to a temporary file ("temp.csv"). After processing all lines, the original file is replaced with the updated file. This function ensures that user wallet balances and PNR lists are accurately maintained.
- **int cancelBookingByPNR(struct Planning *bookings, int *num_Confirm, int *num_RAC, char *pnr, const char* filename, const char *username):**
  This function handles the cancellation of a booking. It searches through the bookings to find the booking with the specified PNR and removes it. If the booking is a confirmed booking, it updates the seat details in the train's seating arrangement and shifts other confirmed bookings. If the booking is a RAC (Reservation Against Cancellation) booking, it updates the RAC seats and reassigns them to other bookings if applicable.
  The function also reads seat information from a binary file ("seatsf.bin") to locate the specific train and update its seating details. The refund amount is calculated based on the journey date and the cost of the cancelled booking.
  If a confirmed booking is cancelled, it checks if there are any RAC bookings that can be moved to confirmed status. It reassigns the seats accordingly and updates the train's seat details.
  After processing the cancellation, the user's wallet amount is updated with the refund, and the PNR is removed from the user's PNR list in the "login.csv" file. The function ensures that train seat arrangements, booking lists, and user wallet balances are accurately updated.

8.PNR enquiry Module:

- **display_booking_details(struct Booking booking, struct Passenger passengers[], int passenger_count):**

  Purpose: This function is used to display the details of a booking, including the travel information and the details of each passenger.
  Explanation: When a PNR is found in the file, this function is called to print the booking details stored in the Booking struct and the array of Passenger structs. It formats the output to display each passenger's name, age, gender, and seat number, making it easy to review all relevant information at once.

- **pnr_enquiry(const char* filename, const char* pnr_input):**

  Purpose: This is the main function that searches for a booking by PNR in a file, parses the booking and passenger details, and displays the information if a match is found.
  Explanation: This function opens the specified file, reads it line by line, and searches for the booking details that match the given PNR. It uses the sscanf function to parse the booking details and populate the Booking and Passenger structs. If a matching PNR is found, it calls display_booking_details to print the booking details. The function reads through the file, processes each booking sequentially, and stops once the required PNR is found or the end of the file is reached.

9. Feedback Module:

- **void count_word_occurrences(const char *word, const char *positive_words[], int positive_size, const char *negative_words[], int negative_size, int *positive_counts, int *negative_counts):**

The function count_word_occurrences takes a word and two lists of words (positive and negative) along with their sizes. It iterates through each list and compares the input word with each word in the list. If there's a match, it increments the corresponding count in the positive or negative counts array. This function essentially checks how many times a word appears in positive and negative lists.

- **void add_review(const char *filename):**
  The function add_review allows users to add a review to a file. It opens the file in append mode, prompts the user for a star rating and the review text, and then writes both the rating and review to the file separated by a newline character. Finally, it closes the file and informs the user that the review was added successfully.

Dataset schema and instances:

Login Database:

login.csv

This database is used to store the information of all the users and the new users who register. It stores their username, password, email and wallet balance.

	A	B	C	D	E
1	Username	Password	Email	Balance	PNRs
2	Sai	saisai	sai.saip2005@gmail.	99830	PNR1,PNR2,PNR3,1
3	Rahul	rahulrahul	rahul2310239@ssn.e	2850	PNR4,PNR5
4	Shailesh	shaileshshailesh	shailesh2310592@ss	3000	PNR6
5	test	test	saipranav2310324@	30850	PNR7,PNR8,PNR9,1

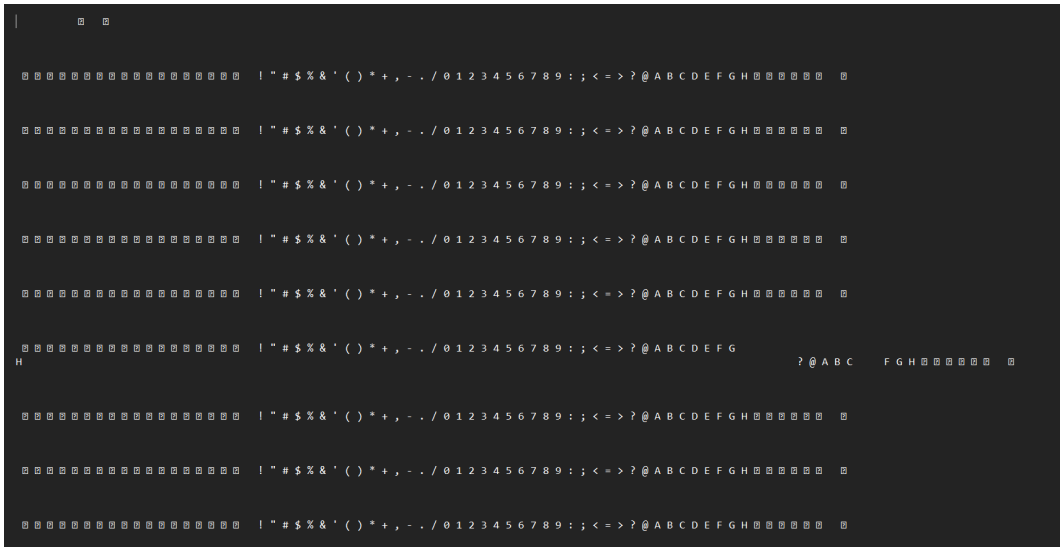
Train route Database:

This database stores the routes of the trains, Destination, Date of journey, Train name, Train number, types of coaches (AC1,AC2,AC3,sleeper) and quota.

	A	B	C	D	E	F	G	H	I	J
1	Route	Destination	Date of Journey	Train Name	TrainNumber	AC1	AC2	AC3	Sleeper	Quota
2	1Chennai-2Katpadi-<	Banglore	2024-06-22	BangloreExp-1500	12	16	96	256	864	night
3	1Chennai-2Coimbat	Palakkad	2024-06-21	PgtExp-1500	13	16	96	256	864	night
4	1Chennai-2Coimbat	Trivandrum	2024-11-11	Capitalexp-1200	14	16	96	256	864	night
5	1Chennai-2Humsafa	Humsafar	2024-06-30	Saipexp-1500	17	16	96	256	864	night

Train seats Database:

This is a binary file containing a structure of Seats for each train, defined with 2 arrays for each coach. The first array contains the seat numbers for each coach (eg-1 to 72 for Sleeper coaches). If a particular seat is booked, the value changes to -1 indicating it has been booked. The second array contains the gender details of passengers. Unbooked seats will be mapped to 0, whilst if it is a Male, the value mapped to the seats in the array will be 100, and 200 for a female. This is used to display the booked seat status in the displaying of the seat chart for each coach.



Booking Database:

The booking database is a text file named book.txt that stores the ticket booking history of passengers. This file serves as a record of all the bookings made by passengers, capturing essential details about each journey and the passengers involved. The file is structured to ensure that each entry is comprehensive and easily retrievable.

- From Station: The starting point of the journey.
- To Station: The destination of the journey.
- Date of Journey (DOJ): The date on which the journey is scheduled.
- Quota: The type of ticket quota (e.g., General, Tatkal).
- Train Number: The unique identifier for the train.
- Coach Type: The type of coach (e.g., AC1, AC2, Sleeper).
- Seat Number: The allocated seat number for the passenger.
- Cost: The cost of the ticket.

From: Chennai
To: Banglore
Date of Journey: 2024-06-22
Train Number: 12
Cost: 200
PNR: 1207BV0ET
Coach No: 7
Passenger 1 Name: rem3
Passenger 1 Age: 25
Passenger 1 Gender: Male
Passenger 1 Seat: 60

From: Chennai
To: Banglore
Date of Journey: 2024-06-22
Train Number: 12
Cost: 600
PNR: 12074K7Y9
Coach No: 7
Passenger 1 Name: test1
Passenger 1 Age: 12
Passenger 1 Gender: Female
Passenger 1 Seat: 58
Passenger 2 Name: Test
Passenger 2 Age: 12
Passenger 2 Gender: Female
Passenger 2 Seat: 59
Passenger 3 Name: test3
Passenger 3 Age: 12
Passenger 3 Gender: Male
Passenger 3 Seat: 61

From: Chennai
To: Banglore
Date of Journey: 2024-06-22
Train Number: 12
Cost: 200
PNR: 120772005
Coach No: 7
Passenger 1 Name: test4
Passenger 1 Age: 12

Reviews.txt:

Gives the reviews and feedbacks of the customers

```
5 This software is incredibly easy to use and very efficient. I love how fast the booking process is.
1 The app is slow and crashes frequently. Very frustrating experience.
4 User interface is intuitive and the app is reliable. Great job!
2 Had a horrible time booking tickets. The app is buggy and unresponsive.
5 Excellent app, very user-friendly and fast.
1 Terrible experience, the app is too complicated and crashes often.
5 Smooth and efficient. Booking tickets has never been easier.
2 Confusing interface and slow performance. Not satisfied.
4 Fast and reliable. Highly recommended.
1 Awful app, it always shows errors and is very slow.
5 Perfect for booking tickets. Very intuitive and quick.
1 Horrible app. Difficult to use and crashes all the time.
5 Love this app! It's fast and easy to use.
2 Disappointing. The app is full of bugs and often unresponsive.
4 Very convenient and easy to navigate. Great for frequent travelers.
1 Unreliable app, it fails to book tickets and often freezes.
5 Fantastic! Very smooth booking process and reliable.
1 Worst app ever. Extremely slow and full of glitches.
4 Satisfied with the app. It's efficient and user-friendly.
```

Admin.csv

This is admin database

	A	B
1	username	password
2	admin1	password123
3	admin2	pass456
4	admin3	admin789

Rac File:

Contains the tickets booked in Rac seats

```
PNR: 1707YHHZ4, Name: Saipu, Age: 22, Gender: Male, Train: 17, RAC No: 1, From: Chennai, To: Humsafar, DOJ: 2024-06-30, Coach: 7, Seat: 63,
Cost: 100
PNR: 1707YHHZ4, Name: Sahilllonu, Age: 22, Gender: Female, Train: 17, RAC No: 2, From: Chennai, To: Humsafar, DOJ: 2024-06-30, Coach: 7,
Seat: 63, Cost: 100
PNR: 1701KYG92, Name: a, Age: 1, Gender: Male, Train: 17, RAC No: 3, From: Chennai, To: Humsafar, DOJ: 2024-06-30, Coach: 1, Seat: 67,
Cost: 50
PNR: 1701D095N, Name: a, Age: 1, Gender: Male, Train: 17, RAC No: 4, From: Chennai, To: Humsafar, DOJ: 2024-06-30, Coach: 1, Seat: 67,
Cost: 50
```

Output for test cases:

C:\Users\prana\Desktop\OneDrive - SSN-Institute\C Lab\FORREVIEW CSEPROJECT\cmake-build-debug\CFinal.exe"
Enter username:test
Enter password:test
Login successful! Wallet amount: 70850
Welcome to the Railway Reservation System
Enter 2 to Access Admin
Enter 3 to Cancel Ticket
Enter 4 to Book Ticket
Enter 5 Enquire PNR
Enter 6 to Quit.
Enter your choice::2
Enter username:admin2
Enter password:pass456
Login successful!
Enter (1) to add a train
Enter (2) to read the sentiment analysis of feedback.
Enter (3) for Clearing Seats
1
Enter the number of trains:1
Enter the journey date (YYYY MM DD):2024 06 24
Enter [Train Number]:20

Do you want to populate a coach f
or demonstration?(YES=1/NO=0)1
Enter the route (e.g.,
1CityA-2CityB):1Chennai-2Katpadi-3Salem-4Erode-5Palakkad
Enter the destination:Palakkad
Enter the [Train Name-Time o
f departure](No Space)PGTEXP-1200
Welcome to the Railway Reservation System
Enter 2 to Access Admin
Enter 3 to Cancel Ticket
Enter 4 to Book Ticket
Enter 5 Enquire PNR
Enter 6 to Quit.
Enter your choice::2
Enter username:admin2
Enter password:pass456
Login successful!
Enter (1) to add a train
Enter (2) to read the sentiment analysis of feedback.
Enter (3) for Clearing Seats
2

Total positive reviews: 49.12%
Total negative reviews: 42.11%
Total neutral reviews: 8.77%
Average star rating: 3.26

Most used positive words:
easy: 8
convenient: 2
fast: 10
quick: 3

smooth: 3
efficient: 8
reliable: 9
user-friendly: 4
intuitive: 7
helpful: 1
excellent: 3
great: 5
amazing: 1
fantastic: 2
perfect: 3
satisfied: 3
happy: 2
impressed: 2
love: 4
awesome: 1

Most used negative words:
difficult: 2
slow: 11
complicated: 2
unreliable: 4
confusing: 2
frustrating: 2
buggy: 3
poor: 2
terrible: 4
bad: 3
worst: 3
horrible: 5
awful: 3
glitchy: 1

laggy: 2
unintuitive: 2
Welcome to the Railway Reservation System
Enter 2 to Access Admin
Enter 3 to Cancel Ticket
Enter 4 to Book Ticket
Enter 5 Enquire PNR
Enter 6 to Quit.
Enter your choice::4

Enter from station:Chennai
Enter to station:Palakkad
Enter date of journey (YYYY-MM-DD):2024-06-24
Available trains between Chennai and Palakkad
on 2024-06-24:

Train Name and Departure Time: PGTEXP-1200
Train Number: 20
Date of Journey: 2024-06-24
Route: 1Chennai-2Katpadi-3Salem-4Erode-5Palakkad
Available Coaches and Seats:
AC1: 16
AC2: 96
AC3: 256
Sleeper: 864

Enter the train number to search:20
Enter number of passengers:2
Available seats in each coach:
Coach S- 1: 67 available seats
Coach S- 2: 67 available seats
Coach S- 3: 67 available seats

Coach S- 4: 67 available seats
Coach S- 5: 67 available seats
Coach S- 6: 67 available seats
Coach S- 7: 67 available seats
Coach S- 8: 67 available seats
Coach S- 9: 67 available seats
Coach S-10: 67 available seats
Coach S-11: 67 available seats
Coach S-12: 67 available seats
Coach AC3-13: 7 available seats
Coach AC3-14: 7 available seats
Coach AC3-15: 7 available seats
Coach AC3-16: 7 available seats
Coach AC2-17: 27 available seats
Coach AC2-18: 27 available seats
Coach AC2-19: 27 available seats
Coach AC1-20: 11 available seats

Enter coach number (1-20):13
Number of available seats in Coach 13: 7

Train Coach Seat Selection Chart for Coach 13:

Disclaimer: Last 5 seats are tatkal and open only one day before departure.
They have Rs.300 Addon Fare.

The next 5 seats before tatkal open only when any coach has been fully booked.

SU	SL	LL	LM	LU	RL	RM	RU
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F

F F F F F F F F
F F F F F F F F
F F F F F F F F
F F F F F F F F
F F F F 53 54 55 56
57 58 59 -- -- -- -- --
Are you travelling as a family? (adjacent gender restriction doesn't apply [Yes/No]):Yes
The current rac no is 0
Enter your gender (1 for Male, 2 for Female):2
Enter seat number to book (1-64):53
Enter details for passenger:
Name:Shailesh
Age:19
Seat 53 in Coach 13 booked successfully.
Enter your gender (1 for Male, 2 for Female):1
Enter seat number to book (1-64):54
Enter details for passenger:
Name:Harjit
Age:19
Seat 54 in Coach 13 booked successfully.
Enter recipient email address for ticket:sai.saip2005@gmail.com
Email sent successfully.

The details of ticket booked are:
From: Chennai
To: Palakkad
DOJ: 2024-06-24
TrainNo: 20
Cost: 400
PNR: 201346UA6

Program successfully quit.

Would you like to add a new review? (y/n): Welcome to the Railway Reservation System
Enter 2 to Access Admin
Enter 3 to Cancel Ticket
Enter 4 to Book Ticket
Enter 5 Enquire PNR
Enter 6 to Quit.
Enter your choice::4
4

Enter from station:Chennai
Chennai
Enter to station:Palakkad
Palakkad
Enter date of journey (YYYY-MM-DD):2024-06-24
2024-06-24
Available trains between Chennai and Palakkad on 2024-06-24:
--
Train Name and Departure Time: PGTEXP-1200
Train Number: 20
Date of Journey: 2024-06-24
Route: 1Chennai-2Katpadi-3Salem-4Erode-5Palakkad
Available Coaches and Seats:
AC1: 16
AC2: 96
AC3: 256
Sleeper: 864

Enter the train number to search:20

20
Enter number of passengers:2
2
Available seats in each coach:
Coach S- 1: 67 available seats
Coach S- 2: 67 available seats
Coach S- 3: 67 available seats
Coach S- 4: 67 available seats
Coach S- 5: 67 available seats
Coach S- 6: 67 available seats
Coach S- 7: 67 available seats
Coach S- 8: 67 available seats
Coach S- 9: 67 available seats
Coach S-10: 67 available seats
Coach S-11: 67 available seats
Coach S-12: 67 available seats
Coach AC3-13: 15 available seats
Coach AC3-14: 7 available seats
Coach AC3-15: 7 available seats
Coach AC3-16: 7 available seats
Coach AC2-17: 27 available seats
Coach AC2-18: 27 available seats
Coach AC2-19: 27 available seats
Coach AC1-20: 11 available seats

Enter coach number (1-20):13
13
Number of available seats in Coach 13: 15

Train Coach Seat Selection Chart for Coach 13:

Disclaimer: Last 5 seats are tatkal and open only one day before departure.
They have Rs.300 Addon Fare.

The next 5 seats before tatkal open only when any coach has been fully booked.
2 people travel in one seat.
The legend for those seats are:
MR:One Male has booked the seat. It is open for further booking.
FR:One Female has booked the seat. It is open for further booking.
MM/MF/FM/FF: 2 males/1 male 1 female/1 female one male/2 females have booked the seat. It cant be booked further.

SU	SL	LL	LM	LU	RL	RM	RU
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	M	55	56
57	58	59	--	--	--	--	--

Are you travelling as a family? (adjacent gender restriction doesn't apply [Yes/No]):Yes
Yes
The current rac no is 0
Enter your gender (1 for Male, 2 for Female):1
1
Enter seat number to book (1-64):55
55

You are trying to book an rac seat.
Book this only if all the other seats are full.

2 people travel in one seat.
If any confirm seat gets cancelled, you will be upgraded free of cost.
Do you want to continue? (Yes / No):Yes
Yes
Enter details for passenger:
Name:Sai
Sai
Age:19
19
55 IS YOUR SEAT NUMBER
Seat 55 in Coach 13 booked successfully.
Enter your gender (1 for Male, 2 for Female):2
2
Enter seat number to book (1-64):55
55

You are trying to book an rac seat.
Book this only if all the other seats are full.
2 people travel in one seat.
If any confirm seat gets cancelled, you will be upgraded free of cost.
Do you want to continue? (Yes / No):Yes
Yes
Enter details for passenger:
Name:Rahul
Rahul
Age:19
19
55 IS YOUR SEAT NUMBER
Seat 55 in Coach 13 booked successfully.
Enter recipient email address for ticket:sai.saip2005@gmail.com
sai.saip2005@gmail.com
Email sent successfully.

The details of ticket booked are:
From: Chennai
To: Palakkad
DOJ: 2024-06-24
TrainNo: 20
Cost: 400
PNR: 2013025A2

Would you like to add a new review? (y/n): Welcome to the Railway Reservation System
Enter 2 to Access Admin
Enter 3 to Cancel Ticket
Enter 4 to Book Ticket
Enter 5 Enquire PNR
Enter 6 to Quit.
Enter your choice::3
3
Enter username:test
test
Enter the PNR number to cancel the booking:201346UA6
201346UA6
Data written to rac_20.txt successfully.
100"PNR7
201346UA6
PNR8
201346UA6
PNR9
201346UA6
201346UA6

201346UA6
2013025A2"
201346UA6
"PNR7,PNR8,PNR9,2013025A2"
Program successfully quit.
100"PNR7
201346UA6
PNR8
201346UA6
PNR9
201346UA6
2013025A2"
201346UA6
"PNR7,PNR8,PNR9,2013025A2"
Program successfully quit.
Booking cancelled successfully!Welcome to the Railway Reservation System
Enter 2 to Access Admin
Enter 3 to Cancel Ticket
Enter 4 to Book Ticket
Enter 5 Enquire PNR
Enter 6 to Quit.
Enter your choice::4
4

Enter from station:Chennai
Chennai
Enter to station:Palakkad
Palakkad
Enter date of journey (YYYY-MM-DD):2024-06-24
2024-06-24
Available trains between Chennai and Palakkad on 2024-06-24:

--
Train Name and Departure Time: PGTEXP-1200
Train Number: 20
Date of Journey: 2024-06-24
Route: 1Chennai-2Katpadi-3Salem-4Erode-5Palakkad
Available Coaches and Seats:
AC1: 16
AC2: 96
AC3: 256
Sleeper: 864

Enter the train number to search:20
20
Enter number of passengers:1
1
Available seats in each coach:
Coach S- 1: 67 available seats
Coach S- 2: 67 available seats
Coach S- 3: 67 available seats
Coach S- 4: 67 available seats
Coach S- 5: 67 available seats
Coach S- 6: 67 available seats
Coach S- 7: 67 available seats
Coach S- 8: 67 available seats
Coach S- 9: 67 available seats
Coach S-10: 67 available seats
Coach S-11: 67 available seats
Coach S-12: 67 available seats
Coach AC3-13: 7 available seats
Coach AC3-14: 7 available seats
Coach AC3-15: 7 available seats
Coach AC3-16: 7 available seats

Coach AC2-17: 27 available seats
Coach AC2-18: 27 available seats
Coach AC2-19: 27 available seats
Coach AC1-20: 11 available seats

Enter coach number (1-20):13
13
Number of available seats in Coach 13: 7

Train Coach Seat Selection Chart for Coach 13:

Disclaimer: Last 5 seats are tatkal and open only one day before departure.
They have Rs.300 Addon Fare.

The next 5 seats before tatkal open only when any coach has been fully booked.
2 people travel in one seat.
The legend for those seats are:
MR:One Male has booked the seat. It is open for further booking.
FR:One Female has booked the seat. It is open for further booking.
MM/MF/FM/FF: 2 males/1 male 1 female/1 female one male/2 females have booked the seat. It cant be booked further.

SU	SL	LL	LM	LU	RL	RM	RU
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F
F	F	F	F	53	54	55	56
57	58	59	--	--	--	--	--

The current rac no is 0
Enter your gender (1 for Male, 2 for Female):1
1
Enter seat number to book (1-64):53
53
Cannot book seat 53 due to adjacent gender restriction or last 5 seats restriction.
Enter your gender (1 for Male, 2 for Female):2
2
Enter seat number to book (1-64):53
53
Enter details for passenger:
Name:Earl
Earl
Age:20
20
Seat 53 in Coach 13 booked successfully.
Enter recipient email address for ticket:sai.saip2005@gmail.com
sai.saip2005@gmail.com
Email sent successfully.

The details of ticket booked are:
From: Chennai
To: Palakkad
DOJ: 2024-06-24
TrainNo: 20
Cost: 200
PNR: 2013VOO5I

Program successfully quit.

Would you like to add a new review? (y/n): Welcome to the Railway Reservation System
Enter 2 to Access Admin
Enter 3 to Cancel Ticket
Enter 4 to Book Ticket
Enter 5 Enquire PNR
Enter 6 to Quit.
Enter your choice::6
6

Process finished with exit code 0

Function Prototypes:

1. authenticateUser:

- Function Prototype: int authenticateUser(const char *username, const char *password);
- Inputs: const char *username, const char *password - Credentials to be authenticated.
- Outputs: int - Returns 1 if authentication is successful, otherwise 0.
- Description: Verifies the provided username and password against records in "login.csv", setting the global walletAmount if successful.

2. stationsCrossed:

- Function Prototype: int stationsCrossed(char *route, char *from, char *to);
- Inputs: char *route, char *from, char *to - The route string and station names to calculate the distance between.
- Outputs: int - The number of stations crossed.

- Description: Calculates the number of stations crossed between two stations in a route string by computing the difference in their positions.

3. updateWalletAmount

- Function Prototype: void updateWalletAmount(const char *username, int newWalletAmount);
- Inputs: const char *username, int newWalletAmount - The username and new wallet amount to be updated.
- Outputs: None.
- Description: Updates the wallet amount for the specified user in "login.csv", modifying the file with the new amount.

4. calculateTotalCost

- Function Prototype: int calculateTotalCost(int stationsCrossed, int coachType);
- Inputs: int stationsCrossed, int coachType - Number of stations crossed and type of coach.
- Outputs: int - The total cost of the journey.
- Description: Computes the total journey cost based on the number of stations

5. readTrainData

- Function Prototype: void readTrainData(struct Availability *trains, const char *filename);
- Inputs: struct Availability *trains, const char *filename - Array to populate and the filename to read from.
- Outputs: None.
- Description: Reads train data from "trains.csv" into an array of struct Availability, processing each line to fill in the train details.

6. showAvailableTrains

- Function Prototype: void showAvailableTrains(struct Availability *trains, int num_trains, char *from, char *to, char *date);
- Inputs: struct Availability *trains, int num_trains, char *from, char *to, char *date - Array of trains, count, and journey details.
- Outputs: None.
- Description: Displays available trains for a specified journey date and route, printing details if criteria match.

7. calculateDaysDifference

- Function Prototype: int calculateDaysDifference(struct tm date1, struct tm date2);
- Inputs: struct tm date1, struct tm date2 - The two dates to compare.
- Outputs: int - The difference in days between the two dates.
- Description: Computes the number of days between two struct tm dates to determine how many days are left before a journey.

8. displayChart

- Function Prototype: void displayChart(int coach, int seats[TOTAL_COACHES][SEATS_S], int genderSeats[TOTAL_COACHES][SEATS_S], int seatCount, int rows, int columnsLeft, int columnsRight, struct tm journeyDate);
- Inputs: int coach, int seats[TOTAL_COACHES][SEATS_S], int genderSeats[TOTAL_COACHES][SEATS_S], int seatCount, int rows, int columnsLeft, int columnsRight, struct tm journeyDate - Seating details and journey date.
- Outputs: None.

- Description: Prints a seating chart for a specified coach, showing seat availability and gender information with booking restrictions based on the journey date.

9. countAvailableSeats

- Function Prototype: int countAvailableSeats(int coach, int seats[TOTAL_COACHES][SEATS_S], int seatCount, struct tm journeyDate);
- Inputs: int coach, int seats[TOTAL_COACHES][SEATS_S], int seatCount, struct tm journeyDate - Coach and seat details with journey date.
- Outputs: int - Number of available seats.
- Description: Counts available seats in a specified coach, considering booking restrictions for seats close to the journey date.

10. displayAvailableSeats

- Function Prototype: void displayAvailableSeats(int seats[TOTAL_COACHES][SEATS_S], struct tm journeyDate);
- Inputs: int seats[TOTAL_COACHES][SEATS_S], struct tm journeyDate - Seat details and journey date.
- Outputs: None.
- Description: Displays the number of available seats in each coach of a train, providing an overview of seat availability across different coach types.

11. canBookSeat

- Function Prototype: int canBookSeat(int coach, int seatIndex, int gender, int genderSeats[TOTAL_COACHES][SEATS_S], int seatCount, struct tm journeyDate, char fam[]);

- Inputs: int coach, int seatIndex, int gender, int genderSeats[TOTAL_COACHES][SEATS_S], int seatCount, struct tm journeyDate, char fam[] - Seat booking details and restrictions.
- Outputs: int - Returns 1 if the seat can be booked, otherwise 0.
- Description: Checks if a seat can be booked based on the passenger's gender, adjacent seat gender, and booking restrictions near the journey date.

12. update_train_details

- Function Prototype: void update_train_details(int trainno, int seats[TOTAL_COACHES][SEATS_S], int genderSeats[TOTAL_COACHES][SEATS_S]);
- Inputs: int trainno, int seats[TOTAL_COACHES][SEATS_S], int genderSeats[TOTAL_COACHES][SEATS_S] - Train number and updated seat details.
- Outputs: None.
- Description: Updates the seat and gender seat details for a specific train in the binary file "seatsf.bin", modifying the file with the new details.

13. generatePNR

- Function Prototype: void generatePNR(char *pnr, int length);
- Inputs: char *pnr, int length - PNR string and the desired length.
- Outputs: None.
- Description: Generates a random PNR (Passenger Name Record) of the specified length using alphanumeric characters.

14. writePlanningDataToFile

- Function Prototype: void writePlanningDataToFile(const char *filename, struct Planning *plan);

- Inputs: const char *filename, struct Planning *plan - Filename and booking details.
- Outputs: None.
- Description: Writes booking details stored in a struct Planning to a file, logging the journey details and passenger information.

15. pnr_enquiry

pnr_enquiry(const char* filename, const char* pnr_input):

- Inputs:

- `const char* filename`: The name of the file containing booking data.
- `const char* pnr_input`: The PNR to be searched.

- Outputs:

- None (void function).

- Working:

- This function searches for a booking by PNR in the specified file, parses the booking and passenger details, and displays the information if a match is found. It uses `sscanf` to parse the data and calls `display_booking_details` to print the details.

16. display_booking_details

- Function Prototype: display_booking_details(struct Booking booking, struct Passenger passengers[], int passenger_count):
- Inputs:
  - `struct Booking booking`: The booking details, including travel information.
  - `struct Passenger passengers[]`: An array of passenger details.
  - `int passenger_count`: The number of passengers.
- Outputs:
  - None (void function).
- Working:

- This function prints the booking details and passenger information, including name, age, gender, and seat number, for easy review.

Limitations:

1. **Lack of additional Quotas:**

   Additional quotas such as premium tatkal, senior citizen, disability or duty pass are not available in this system.

2. **Food and Beverage delivery:**

   Food and beverages cannot be ordered along the stops of the train in this system.

3. **Goods booking:**

   Transport of goods through the railway cannot be booked using the system.

4. **User Interface:**

   C language may not offer as rich and interactive user interfaces as modern web or mobile applications, potentially leading to a less intuitive user experience.

5. **Security:**

   Implementing robust security features, such as encryption or Robot/Human verification may be more challenging in C compared to languages with built-in security libraries and frameworks.

6. **Space Complexity:**

   Since each ticket is stored as a separate file, space complexity increases with increase in number of bookings.

Technical Limitations

1. **Scalability**: C programming may not be the best choice for building highly scalable systems. As the size and complexity of the reservation system grows, it may become challenging to maintain and extend the codebase.

2. **Platform Dependence**: C programs are often platform-dependent, meaning they may not run seamlessly on different operating systems without modifications. This could limit the system's compatibility with various platforms and environments.
3. **Limited Libraries**: While C has standard libraries for common tasks, it may lack comprehensive libraries for complex functionalities such as network communication, database interaction, and web services integration. This could result in more manual implementation and potential limitations in functionality.
4. **Maintenance**: C codebases can be more challenging to maintain, and update compared to higher-level languages with more advanced features and abstractions. As the system evolves and requirements change, maintaining and extending the codebase could become increasingly complex and time-consuming.

Observations from Societal, Legal, Environmental, and Ethical Perspectives for the C Railway Project

Societal Perspective

- Accessibility: Design the reservation system to be accessible to all users, including those with disabilities, ensuring compliance with accessibility standards.

- User-Friendliness: Develop a clear and intuitive command-line interface that is easy to navigate, even for users who are not tech-savvy.
- Reliability and Performance: Ensure the system is robust, capable of handling high volumes of transactions without performance degradation.
- Inclusivity: Support multiple languages to cater to users from diverse backgrounds and regions.

Legal Perspective

- Data Protection Compliance: Adhere to data protection laws such as GDPR or CCPA, ensuring user data is collected, stored, and processed securely.
- Consumer Protection Laws: Implement straightforward refund and cancellation policies that comply with consumer protection laws.
- Cybersecurity: Comply with cybersecurity regulations to protect the system from data breaches and cyber-attacks.
- Accessibility Laws: Ensure the system meets legal requirements for accessibility.

Environmental Perspective

- Code Optimization: Optimize the code to run efficiently, reducing the energy consumption of servers and client devices.
- Electronic Documentation: Encourage the use of electronic tickets and receipts to minimize paper usage.
- Eco-Friendly Data Centers: Use data centers powered by renewable energy sources to minimize the carbon footprint.
- Promote Public Transport: Promote the use of public transportation through the system to reduce overall carbon emissions by encouraging train travel over personal vehicles.

Ethical Perspective

- Non-Discrimination: Ensure the system does not discriminate against users based on race, gender, socioeconomic status, or any other characteristic.
- Data Transparency: Clearly communicate how user data is collected, stored, and used.
- Data Security: Implement robust security measures to protect user data from unauthorized access.
- Accountability: Maintain accountability for the system's performance and any issues that arise.
- Fair Algorithms: Ensure algorithms for recommendations, pricing, or availability do not unfairly disadvantage any group of users.

Features of the Railway Reservation System

PNR Status Check

- Functionality: Allow passengers to check the status of their PNR (confirmed, RAC, waiting list).

Data Storage

- Functionality: Store details of trains, fares, PNRs, and reservations in a structured database.

Cancellation

- Functionality: Enable passengers to cancel tickets using their PNR number and handle requests efficiently.

Refund Rules

- Admin Control: Allow the admin to manage refund policies based on the reservation date and fare.

Admin Control

- System Maintenance: Provide the admin with the ability to manage system maintenance, including updating train and fare details.

Learning Outcomes

Technical Skills

1. Programming Proficiency: Improved C programming skills, including memory management, file handling, and debugging.
2. System Design: Gained experience in designing a complex system with a robust database schema.
3. Algorithm Development: Developed efficient algorithms for managing reservations and implemented security features.
4. User Interface Design: Created a user-friendly command-line interface.

Professional Skills

1. Communication: Enhanced technical communication skills in both writing and presentations, and effectively communicated work to senior authorities.

Team-work Skills

1. Effective Cooperation: Demonstrated effective team cooperation.

2. Regular Updates: Provided clear, regular updates and shared ideas.
3. Constructive Mediation: Mediated disagreements constructively.
4. Task Assignment: Efficiently assigned tasks based on team members' skills.
5. Coordination: Coordinated schedules and met deadlines.

References

1. https://www.irctc.co.in/nget/

2. https://www.geeksforgeeks.org

3. https://www.redbus.com

4. https://www.smtp.com

5.https://learn.microsoft.com/en-us/cpp/c-language/c-language-reference?view=msvc-170