Project Report on

Bus Tourism Management

For the evaluation of innovative work for MTE component

3rd semester, 2020-21

DS SE-201

Submitted to: **Prof. Indu Singh**

Department of Computer Science

Kshitiz Goel (2K19/SE/064)

Mayank Mittal (2K19/SE/071)

B. Tech Software Engineering

Delhi Technological University

Bawana Rd, Shahbad Daulatpur Village, Rohini, Delhi, 110042

INDEX

ACKNOWLEDGEMENT

We would like to express our gratitude to all those who gave us the possibility to complete this project. We want to thank the Department of Computer Science in the college for giving us permission to develop a project in this instance. The development of this project was only possible by the support, encouragement, cooperation of our teachers and we have got full support from our teachers .

We would also like to express heartfelt thanks to the people who are directly and indirectly part of this project. We have to thank our subject teacher **Prof. Indu Singh** for his help by providing the essential suggestions, help and the solution for the problem during project development.

ABSTRACT

Introduction

Millions of people travel on buses daily in India. Buses are the most important means of transportation for the common man as these provide close to home connectivity. In this project, our main aim is to provide the user with all the information about buses available to facilitate their transport.

Objectives

This program gives all the necessary information which will provide a pleasant, safe, and sound journey to their destination. The user at each step will be asked a question according to which the details will be provided to her. From the total time to reach the destination to the total seats vacant, all the information will be fetched from the database to the user within a few clicks.

System Requirements & Softwares Used

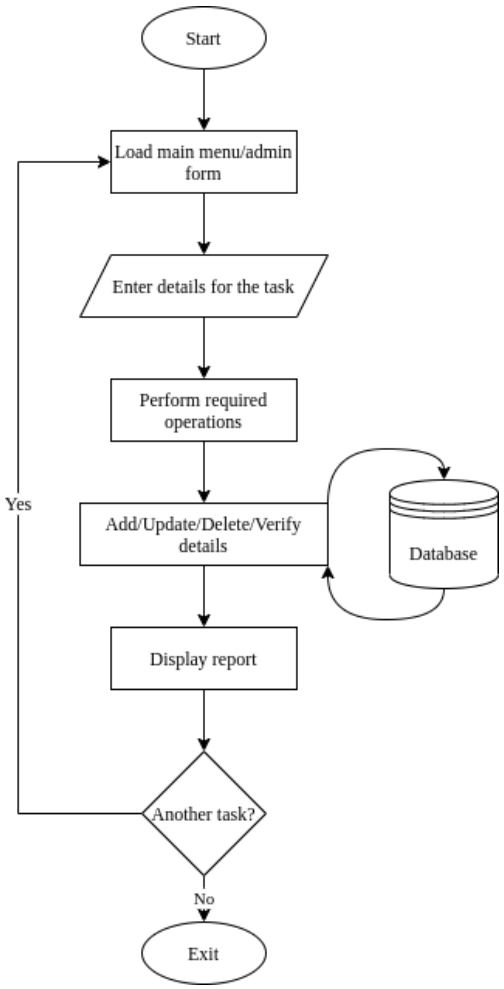
- A 32-bit or a 64-bit architecture machine
- Any operating system like Linux/Windows/Mac
- Preferably g++/gcc v10.2 installed
- Preferably OpenJDK Java SE-11 installed
- Any IDE with modern compiler support like VS Code and Eclipse

METHODOLOGY

The program is implemented as a menu driven program that loops till the user decides to break out. Files are loaded when the program starts and written into when the program ends.

There are two log in modes:

- **Admin** mode (password protected)
- **Customer** mode



The **admin** should be able to perform the following tasks:

- Book a ticket
- Cancel a ticket
- Reschedule a ticket
- Print details of bus
- Print details of trip
- Print all trips to a particular location
- Print all logs (all trips to all locations)
- Print a ticket
- Change admin password
- Edit a bus
- Reset a bus/trip
- Add a new destination
- Remove a destination
- Reset all logs (all trips to all locations)
- Edit a trip
- Log out

The **customer** is allowed to perform all operations of admin except:

- Change admin password
- Edit a bus
- Reset a bus/trip
- Add a new destination
- Remove a destination
- Reset all logs (all trips to all locations)
- Edit a trip

This ensures safe handling of data and security.

IMPLEMENTATION

The program is written with an extensive use of data structures and algorithms to ensure a fast and scalable application. Some prominent data structures used are:

- HashMap/unordered_map
- ArrayList/vector
- HashSet/unordered_set

All the code has been maintained as a public repository on Github.

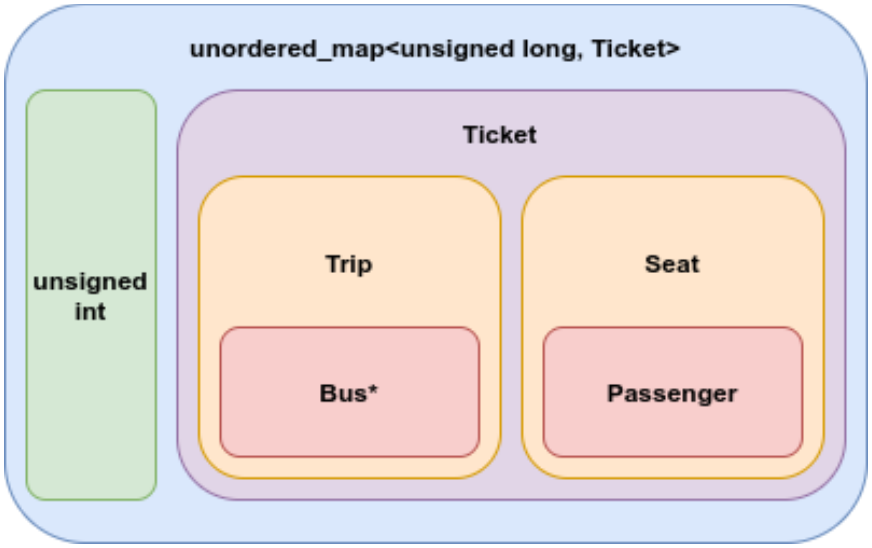


Fig. unordered_map<unsigned long, Ticket> mapTicket

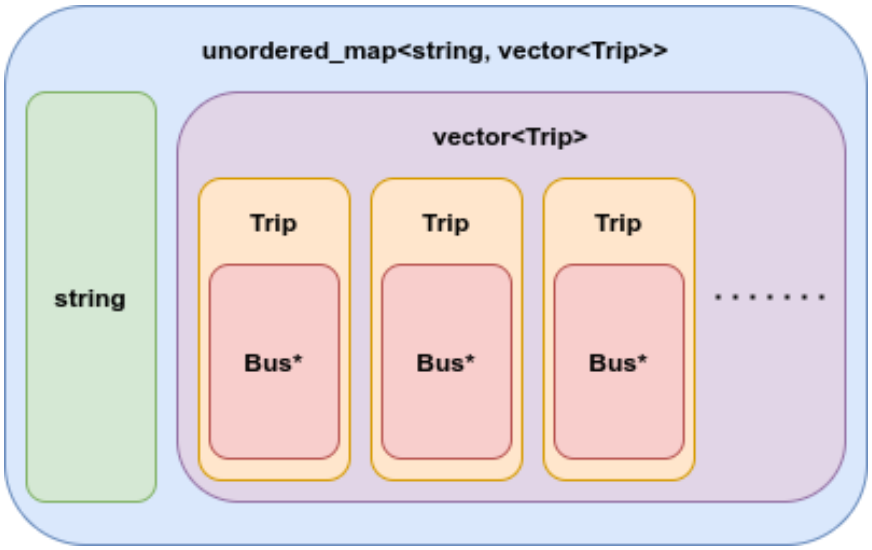


Fig. unordered_map<unsigned long, Ticket> mapTrips

How Do The Operations Work?

- **BOOKING**
    a. Accept destination
    b. Print all trips to the destination
        i. Find all corresponding trips in the map of trips
        ii. Display each trip
            1. Display trip specific information
            2. Display vacant bus seats
                a. Traverse through the seats set
                b. Display each seats' coordinates
    c. Accept bus tier
    d. Find corresponding trip in the map of trips
    e. Book a seat in the trip
        i. Locate the bus
        ii. Accept relevant details
        iii. Create passenger

iv. Add seat to seats set

f. Generate ticket
    i. Send all details as arguments to constructor
    ii. Match and assign details
g. Assign ticket number using rand()
h. Print ticket
    i. Display ticket specific details
    ii. Display seat details
        1. Display seat coordinates
        2. Display passenger details

- **CANCELLING**
    a. Accept ticket number
    b. Check if ticket present in the map of tickets
    c. Ask for confirmation
    d. If no, return
    e. If yes, get ticket from the map of tickets
    f. Locate trip and the bus
    g. Erase seat from set of seats
    h. Erase ticket from the map of tickets

- **RESCHEDULING**
    a. Cancel previous ticket
        i. Accept ticket number
        ii. Check if ticket present in the map of tickets
        iii. Ask for confirmation
        iv. If no, return
        v. If yes, get ticket from the map of tickets
        vi. Locate trip and the bus
        vii. Erase seat from set of seats
        viii. Erase ticket from the map of tickets

b. Book new ticket
    i. Accept destination
    ii. Print all trips to the destination
        1. Find all corresponding trips in the map of trips
        2. Display each trip
            a. Display trip specific information
            b. Display vacant bus seats
                i. Traverse through the seats set
                ii. Display each seats' coordinates
    iii. Accept bus tier
    iv. Find corresponding trip in the map of trips
    v. Book a seat in the trip
        1. Locate the bus
        2. Accept relevant details
        3. Create passenger
        4. Add seat to seats set
    vi. Generate ticket
        1. Send all details as arguments to constructor
        2. Match and assign details
    vii. Assign ticket number using rand()
    viii. Print ticket
        1. Display ticket specific details
        2. Display seat details
            a. Display seat coordinates
            b. Display passenger details

- **PRINT BUS INFORMATION**
    a. Accept bus tier
    b. Locate bus of entered tier using conditionals
    c. Print details of the bus

- **PRINT TRIP**
    a. Accept destination
    b. Find all corresponding trips from the map of trips
    c. Accept bus tier
    d. Find trip from all retrieved trips using bus tier
    e. Display the trip
        i. Display trip specific information
        ii. Display vacant bus seats
            1. Traverse through the seats set
            2. Display each seats' coordinates

- **PRINT ALL TRIPS TO A LOCATION**
    a. Accept destination
    b. Find all corresponding trips in the map of trips
    c. Display each trip
        i. Display trip specific information
        ii. Display vacant bus seats
            1. Traverse through the seats set
            2. Display each seats' coordinate

- **PRINT ALL LOGS**
    a. Traverse through the map of trips
    b. For each destination print all trips
        i. Find all corresponding trips in the map of trips
        ii. Display each trip
            1. Display trip specific information
            2. Display vacant bus seats
                a. Traverse through the seats set
                b. Display each seats' coordinate

- **PRINT TICKET**
    a. Accept ticket number
    b. Check if ticket present in the map of tickets
    c. Get ticket from the map of tickets
    d. Display ticket specific details
    e. Display seat details
        i. Display seat coordinates
        ii. Display passenger details

- **EDIT BUS**
    a. Accept bus tier
    b. Locate bus with entered tier
    c. Accept new details
    d. Overwrite new details over previous ones

- **RESET BUS/TRIP**
    a. Accept destination
    b. Find all corresponding trips from the map of trips
    c. Accept bus tier
    d. Find trip from all retrieved trips using bus tier
    e. Locate the bus
    f. Clear set of seats

- **ADD DESTINATION**
    a. Accept destination
    b. Create destination trip list
        i. Accept all details for each tier/trip
        ii. Add all details to a list
    c. Add destination and destination list to map of trips

- **REMOVE DESTINATION**
    a. Accept destination
    b. Check if destination present in the map of trips
    c. Erase entry corresponding to the destination in map of trips

- **EDIT TRIP**
    a. Accept destination
    b. Find all corresponding trips from the map of trips
    c. Accept bus tier
    d. Find trip from all retrieved trips using bus tier
    e. Accept new details
    f. Overwrite new details over previous ones

- **CHANGE PASSWORD**
    a. Open the password file and clear all contents
    b. Accept new password
    c. Write new password to file
    d. Close the file

Structure Of The Code

The program contains the following classes and members: private, public, global

- **Passenger**
  - string name
  - int age
  - string gender
  - string phone
  - string email
  - void acceptDetails()
  - void displayPassenger()

- **Seat**
  - Passenger passenger
  - int row
  - int column
  - int count
  - void displayCoordinates()
  - void displaySeat()

- **Bus**
  - int tier
  - unordered_set<Seat, SeatHash> seats
  - virtual void displayVacantSeats()
  - virtual void displayVacantSeats()
  - virtual void cancelSeat(Seat)
  - virtual void clearBus()

- **Tier1, Tier2, Tier3 (Inherited from Bus)**
  - static const int rows
  - static const int columns
  - static bool ac
  - static bool food
  - static bool sleeper
  - static bool toilet
  - static bool tv
  - void printInfo()
  - void changeBus()
  - void clearBus()
  - void displayVacantSeats()
  - Seat bookSeat()
  - void cancelSeat(Seat)

- **Trip**
  - Bus *bus
  - string startDay
  - string startTime
  - string endDay
  - string endTime
  - int fare
  - Seat bookTrip()
  - void cancelTrip(Seat)
  - void displayTrip()
  - int discount()
  - void editTrip()
  - void clearTrip()

- **Ticket**
  - static const string source
  - string destination
  - Trip trip;
  - Seat seat;
  - int fare;
  - void setFare()
  - void displayTicket()
  - void cancelTicket()

- **Log**
  - unordered_map<unsigned long, Ticket> mapTicket
  - unordered_map<string, vector<Trip>> mapTrips
  - vector<Trip> mumbaiList()
  - vector<Trip> jaipurList()
  - vector<Trip> chennaiList()
  - vector<Trip> kolkataList()
  - vector<Trip> destinationList()
  - Bus *getTierBus(int)
  - void displayTrips(string)
  - void booking()
  - void cancelling()
  - void rescheduling()
  - void printBus()
  - void printTrip()
  - void printTrips()
  - void printLogs()
  - void printTicket()
  - void editBus()
  - void resetTrip()
  - void addDestination()

    - void removeDestination()
    - void editTrip()

- **Global**
    - Log storage
    - void readLogs()
    - void writeLogs()
    - void defaultPassword()
    - void changePassword()
    - string getPassword()
    - void customerMode()
    - void adminMode()
    - int main()

CONCLUSION

Through the program, the user will be able to conveniently plan their trip and get all the information required by their side. Through extensive use of data structures and algorithms, we build a very efficient, easy to use, and scalable application to book bus tickets.

REFERENCES

[1] Yatra.com

https://www.yatra.com/bus-tickets/

[2] Intercity.com

https://www.intrcity.com/

[3] RailYatri.com

https://www.railyatri.in/bus-booking

OUTPUTS

```
Log in as:-
1. Admin
2. Customer
3. Exit
Enter your choice: 1

Enter admin password: abcd
Wrong password entered, entry denied!

Log in as:-
1. Admin
2. Customer
3. Exit
Enter your choice: 1

Enter admin password: hello

1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 1

Enter destination: Mumbai
```

```
Available choices:-
Bus tier: 1
Start day: Tuesday
Start time: 1200
End day: Thursday
End time: 1200
Seats range, Rows: 5, Columns: 2
Non-vacant seats:-
Row: 1, Column: 1
Row: 1, Column: 1
Orignal fare: 5000

Bus tier: 2
Start day: Saturday
Start time: 1200
End day: Monday
End time: 1200
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 3000

Bus tier: 3
Start day: Thursday
Start time: 1800
End day: Saturday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 1700


Enter your bus tier: 1
Enter seat co-ordinates:-
Row: 2
Column: 1
Enter your details:-
Name: Mayank
Age: 19
Gender: Male
```

```
Phone No: 8860
E-mail: sample@gmail.com
Seat booked!

You have been selected for a discount round wherein you will have to
perform a simple task and on the basis of that you will be awarded a
discount on fare
Enter 2 numbers from 1 to 10:-
First number: 2
Second number: 3
Congratulations! You have got a flat 7% reduction in fares

Ticket Number: 188435804
Source: Delhi
Destination: Mumbai
Bus tier: 1
Start day: Tuesday
Start time: 1200
End day: Thursday
End time: 1200
Seats range, Rows: 5, Columns: 2
Non-vacant seats:-
Row: 1, Column: 1
Row: 2, Column: 1
Row: 1, Column: 1
Orignal fare: 5000
Discounted fare: 4650
Name: Mayank
Age: 19
Gender: Male
Phone No: 8860
E-mail: sample@gmail.com
Row: 2, Column: 1


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
```

```
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 3

Your previous ticket will be cancelled and new ticket will be booked
Please enter the ticket number: 188435804
Are you sure, You want to re-schedule the trip? Press Y or N: Y
Seat cancelled!

Enter destination: Kolkata
Available choices:-
Bus tier: 1
Start day: Friday
Start time: 1200
End day: Sunday
End time: 1200
Seats range, Rows: 5, Columns: 2
Non-vacant seats:-
All vacant!
Orignal fare: 5500

Bus tier: 2
Start day: Wednesday
Start time: 1800
End day: Friday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 4000

Bus tier: 3
Start day: Sunday
```

```
Start time: 1800
End day: Tuesday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 2500


Enter your bus tier: 2
Enter seat co-ordinates:-
Row: 3
Column: 2
Enter your details:-
Name: Kshitiz
Age: 19
Gender: Male
Phone No: 9811
E-mail: coding@gmail.com
Seat booked!

You have been selected for a discount round wherein you will have to
perform a simple task and on the basis of that you will be awarded a
discount on fare
Enter 2 numbers from 1 to 10:-
First number: 1
Second number: 1
Congratulations! You have got a flat 7% reduction in fares

Ticket Number: 298561843
Source: Delhi
Destination: Kolkata
Bus tier: 2
Start day: Wednesday
Start time: 1800
End day: Friday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
Row: 3, Column: 2
Orignal fare: 4000
```

```
Discounted fare: 3720
Name: Kshitiz
Age: 19
Gender: Male
Phone No: 9811
E-mail: coding@gmail.com
Row: 3, Column: 2



1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 4

Enter bus tier: 1
Number of rows: 5
Number of columns: 2
AC: false
Food: false
Sleeper: false
Toilet: false
TV: false


1. Booking
2. Cancelling
3. Re-schedule/edit details
```

```
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 5

Enter destination: Chennai
Enter bus tier: 3
Bus tier: 3
Start day: Saturday
Start time: 1800
End day: Tuesday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 3500


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
```

```
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 6

Enter destination: Jaipur
Bus tier: 1
Start day: Wednesday
Start time: 1200
End day: Thursday
End time: 1200
Seats range, Rows: 5, Columns: 2
Non-vacant seats:-
All vacant!
Orignal fare: 3500

Bus tier: 2
Start day: Sunday
Start time: 1200
End day: Monday
End time: 1200
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 2500

Bus tier: 3
Start day: Friday
Start time: 1800
End day: Saturday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 1000


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
```

```
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 7

Destination: Chennai
Bus tier: 1
Start day: Thursday
Start time: 1200
End day: Sunday
End time: 1200
Seats range, Rows: 5, Columns: 2
Non-vacant seats:-
All vacant!
Orignal fare: 7500

Bus tier: 2
Start day: Tuesday
Start time: 1800
End day: Friday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 5000

Bus tier: 3
Start day: Saturday
Start time: 1800
End day: Tuesday
End time: 1800
Seats range, Rows: 10, Columns: 4
```

```
Non-vacant seats:-
All vacant!
Orignal fare: 3500

Destination: Jaipur
Bus tier: 1
Start day: Wednesday
Start time: 1200
End day: Thursday
End time: 1200
Seats range, Rows: 5, Columns: 2
Non-vacant seats:-
All vacant!
Orignal fare: 3500

Bus tier: 2
Start day: Sunday
Start time: 1200
End day: Monday
End time: 1200
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 2500

Bus tier: 3
Start day: Friday
Start time: 1800
End day: Saturday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 1000

Destination: Kolkata
Bus tier: 1
Start day: Friday
Start time: 1200
End day: Sunday
End time: 1200
```

```
Seats range, Rows: 5, Columns: 2
Non-vacant seats:-
All vacant!
Orignal fare: 5500

Bus tier: 2
Start day: Wednesday
Start time: 1800
End day: Friday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
Row: 3, Column: 2
Orignal fare: 4000

Bus tier: 3
Start day: Sunday
Start time: 1800
End day: Tuesday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 2500

Destination: Mumbai
Bus tier: 1
Start day: Tuesday
Start time: 1200
End day: Thursday
End time: 1200
Seats range, Rows: 5, Columns: 2
Non-vacant seats:-
Row: 1, Column: 1
Row: 1, Column: 1
Orignal fare: 5000

Bus tier: 2
Start day: Saturday
Start time: 1200
End day: Monday
```

```
End time: 1200
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 3000

Bus tier: 3
Start day: Thursday
Start time: 1800
End day: Saturday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
All vacant!
Orignal fare: 1700


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 8

Please enter the ticket number: 298561843
Source: Delhi
Destination: Kolkata
Bus tier: 2
Start day: Wednesday
Start time: 1800
```

```
End day: Friday
End time: 1800
Seats range, Rows: 10, Columns: 4
Non-vacant seats:-
Row: 3, Column: 2
Orignal fare: 4000
Discounted fare: 3720
Name: Kshitiz
Age: 19
Gender: Male
Phone No: 9811
E-mail: coding@gmail.com
Row: 3, Column: 2


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 9

Enter new password: India
Password changed!


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
```

```
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 10

Enter tier: 3
Enter new details:-
AC: false
Food: false
Sleeper: false
Toilet: true
TV: false
Bus updated!


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 11
```

```
Enter destination: Mumbai
Enter bus tier: 1
Bus resetted!


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 12

Enter destination: Jammu
Enter details for tier: 1
Enter start day: Monday
Enter start time: 1200
Enter end day: Wednesday
Enter end time: 1200
Enter fare: 1500
Trip added!

Enter details for tier: 2
Enter start day: Tuesday
Enter start time: 1600
Enter end day: Thursday
Enter end time: 1600
Enter fare: 1500
Trip added!
```

```
Enter details for tier: 3
Enter start day: Wednesday
Enter start time: 1800
Enter end day: Friday
Enter end time: 1800
Enter fare: 1500
Trip added!

Destination added!


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 13

Enter destination: Jammu
Destination removed!


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
```

```
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 15

Enter destination: Mumbai
Enter bus tier: 2
Enter start day: Wednesday
Enter start time: 1500
Enter end day: Thursday
Enter end time: 1500
Enter fare: 2000
Trip edited!


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 2

Please enter the ticket number: 298561843
Are you sure, You want to cancel the trip? Press Y or N: Y
Seat cancelled!
```

```
1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 14


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Change password
10. Edit bus information
11. Reset bus/trip
12. Add new destination
13. Remove destination
14. Reset logs
15. Edit trip
16. Log out
Enter your choice: 16

Logged out
```

```
Log in as:-
1. Admin
2. Customer
3. Exit
Enter your choice: 2


1. Booking
2. Cancelling
3. Re-schedule/edit details
4. Print bus information
5. Print a trip
6. Print all trips to a location
7. Print all logs
8. Print ticket
9. Log out
Enter your choice: 9

Logged out

Log in as:-
1. Admin
2. Customer
3. Exit
Enter your choice: 3

Exited
```

Code

```
#include <bits/stdc++.h>
using namespace std;

int seatCount = 0;

class Passenger
{
private:
   string name;
   int age;
   string gender;
   string phone;
   string email;

public:
   Passenger()
   {
   }

   void acceptDetails()
   {
       cout << "Enter your details:-" << endl;
       cout << "Name: ";
       getline(cin, name);
       cout << "Age: ";
       cin >> age;
       cin.ignore();
       cout << "Gender: ";
       getline(cin, gender);
       cout << "Phone No: ";
       getline(cin, phone);
       cout << "E-mail: ";
       getline(cin, email);
   }

   void displayPassenger()
   {
       cout << "Name: " << name << endl;
       cout << "Age: " << age << endl;
       cout << "Gender: " << gender << endl;
       cout << "Phone No: " << phone << endl;
```

```
        cout << "E-mail: " << email << endl;
    }
};

class Seat
{
private:
    Passenger passenger;
    int row;
    int column;

public:
    int count;

    Seat()
    {
    }

    Seat(Passenger passenger, int row, int column)
    {
        seatCount++;

        count = seatCount;
        this->passenger = passenger;
        this->row = row;
        this->column = column;
    }

    void displayCoordinates()
    {
        cout << "Row: " << row << ", Column: " << column << endl;
    }

    void displaySeat()
    {
        passenger.displayPassenger();
        cout << "Row: " << row << ", Column: " << column << endl;
    }
    bool operator==(const Seat &s) const
    {
        return this->count == s.count;
    }
};
```

```
class SeatHash
{
public:
   size_t operator()(const Seat &s) const
   {
       return s.count;
   }
};

class Bus
{
public:
   int tier;
   unordered_set<Seat, SeatHash> seats;

   virtual void displayVacantSeats() = 0;
   virtual Seat bookSeat() = 0;
   virtual void cancelSeat(Seat seat) = 0;
   virtual void clearBus() = 0;
};

class Tier1 : public Bus
{
private:
   static const int rows = 5;
   static const int columns = 2;

   static bool ac;
   static bool food;
   static bool sleeper;
   static bool toilet;
   static bool tv;

public:
   Tier1()
   {
       tier = 1;
       ac = true;
       food = true;
       sleeper = true;
       toilet = true;
       tv = true;
   }
```

```
    static void printInfo()
    {
        cout << "Number of rows: " << rows << endl;
        cout << "Number of columns: " << columns << endl;
        cout << "AC: " << ac << endl;
        cout << "Food: " << food << endl;
        cout << "Sleeper: " << sleeper << endl;
        cout << "Toilet: " << toilet << endl;
        cout << "TV: " << tv << endl;
    }

    static void changeBus()
    {
        cout << "Enter new details:-" << endl;
        cout << "AC: ";
        cin >> ac;
        cin.ignore();
        cout << "Food: ";
        cin >> food;
        cin.ignore();
        cout << "Sleeper: ";
        cin >> sleeper;
        cin.ignore();
        cout << "Toilet: ";
        cin >> toilet;
        cin.ignore();
        cout << "TV: ";
        cin >> tv;
        cin.ignore();
        cout << "Bus updated!" << endl;
    }

    void clearBus()
    {
        seats.clear();
        cout << "Bus reseted!" << endl;
    }

    void displayVacantSeats()
    {
        cout << "Seats range, Rows: " << rows << ", Columns: " << columns <<
endl;
        cout << "Non-vacant seats:-" << endl;
        if (seats.empty())
```

```
        {
            cout << "All vacant!" << endl;
        }
        else
        {
            for (Seat seat : seats)
            {
                seat.displayCoordinates();
            }
        }
    }

    Seat bookSeat()
    {
        cout << "Enter seat coordinates:-" << endl;
        cout << "Row: ";
        int row;
        cin >> row;
        cin.ignore();
        cout << "Column: ";
        int column;
        cin >> column;
        cin.ignore();
        Passenger passenger;
        passenger.acceptDetails();
        Seat seat(passenger, row, column);
        seats.emplace(seat);
        cout << "Seat booked!" << endl;
        return seat;
    }

    void cancelSeat(Seat seat)
    {
        seats.erase(seat);
        cout << "Seat cancelled!" << endl;
    }
};

bool Tier1::ac;
bool Tier1::food;
bool Tier1::sleeper;
bool Tier1::toilet;
bool Tier1::tv;
```

```
class Tier2 : public Bus
{
private:
   static const int rows = 10;
   static const int columns = 4;

   static bool ac;
   static bool food;
   static bool sleeper;
   static bool toilet;
   static bool tv;

public:
   Tier2()
   {
       tier = 2;
       ac = true;
       food = true;
       sleeper = false;
       toilet = false;
       tv = false;
   }

   static void printInfo()
   {
       cout << "Number of rows: " << rows << endl;
       cout << "Number of columns: " << columns << endl;
       cout << "AC: " << ac << endl;
       cout << "Food: " << food << endl;
       cout << "Sleeper: " << sleeper << endl;
       cout << "Toilet: " << toilet << endl;
       cout << "TV: " << tv << endl;
   }

   static void changeBus()
   {
       cout << "Enter new details:-" << endl;
       cout << "AC: ";
       cin >> ac;
       cin.ignore();
       cout << "Food: ";
       cin >> food;
       cin.ignore();
       cout << "Sleeper: ";
```

```
        cin >> sleeper;
        cin.ignore();
        cout << "Toilet: ";
        cin >> toilet;
        cin.ignore();
        cout << "TV: ";
        cin >> tv;
        cin.ignore();
        cout << "Bus updated!" << endl;
    }

    void clearBus()
    {
        seats.clear();
        cout << "Bus reseted!" << endl;
    }

    void displayVacantSeats()
    {
        cout << "Seats range, Rows: " << rows << ", Columns: " << columns <<
endl;
        cout << "Non-vacant seats:-" << endl;
        if (seats.empty())
        {
            cout << "All vacant!" << endl;
        }
        else
        {
            for (Seat seat : seats)
            {
                seat.displayCoordinates();
            }
        }
    }

    Seat bookSeat()
    {
        cout << "Enter seat coordinates:-" << endl;
        cout << "Row: ";
        int row;
        cin >> row;
        cin.ignore();
        cout << "Column: ";
        int column;
```

```
        cin >> column;
        cin.ignore();
        Passenger passenger;
        passenger.acceptDetails();
        Seat seat(passenger, row, column);
        seats.emplace(seat);
        cout << "Seat booked!" << endl;
        return seat;
    }

    void cancelSeat(Seat seat)
    {
        seats.erase(seat);
        cout << "Seat cancelled!" << endl;
    }
};

bool Tier2::ac;
bool Tier2::food;
bool Tier2::sleeper;
bool Tier2::toilet;
bool Tier2::tv;

class Tier3 : public Bus
{
private:
    static const int rows = 10;
    static const int columns = 4;

    static bool ac;
    static bool food;
    static bool sleeper;
    static bool toilet;
    static bool tv;

public:
    Tier3()
    {
        tier = 3;
        ac = false;
        food = false;
        sleeper = false;
        toilet = false;
        tv = false;
```

```
    }

    static void printInfo()
    {
        cout << "Number of rows: " << rows << endl;
        cout << "Number of columns: " << columns << endl;
        cout << "AC: " << ac << endl;
        cout << "Food: " << food << endl;
        cout << "Sleeper: " << sleeper << endl;
        cout << "Toilet: " << toilet << endl;
        cout << "TV: " << tv << endl;
    }

    static void changeBus()
    {
        cout << "Enter new details:-" << endl;
        cout << "AC: ";
        cin >> ac;
        cin.ignore();
        cout << "Food: ";
        cin >> food;
        cin.ignore();
        cout << "Sleeper: ";
        cin >> sleeper;
        cin.ignore();
        cout << "Toilet: ";
        cin >> toilet;
        cin.ignore();
        cout << "TV: ";
        cin >> tv;
        cin.ignore();
        cout << "Bus updated!" << endl;
    }

    void clearBus()
    {
        seats.clear();
        cout << "Bus reseted!" << endl;
    }

    void displayVacantSeats()
    {
        cout << "Seats range, Rows: " << rows << ", Columns: " << columns <<
endl;
```

```
        cout << "Non-vacant seats:-" << endl;
        if (seats.empty())
        {
            cout << "All vacant!" << endl;
        }
        else
        {
            for (Seat seat : seats)
            {
                seat.displayCoordinates();
            }
        }
    }

    Seat bookSeat()
    {
        cout << "Enter seat coordinates:-" << endl;
        cout << "Row: ";
        int row;
        cin >> row;
        cin.ignore();
        cout << "Column: ";
        int column;
        cin >> column;
        cin.ignore();
        Passenger passenger;
        passenger.acceptDetails();
        Seat seat(passenger, row, column);
        seats.emplace(seat);
        cout << "Seat booked!" << endl;
        return seat;
    }

    void cancelSeat(Seat seat)
    {
        seats.erase(seat);
        cout << "Seat cancelled!" << endl;
    }
};

bool Tier3::ac;
bool Tier3::food;
bool Tier3::sleeper;
bool Tier3::toilet;
```

```
bool Tier3::tv;

class Trip
{
private:
   Bus *bus;
   string startDay;
   string startTime;
   string endDay;
   string endTime;
   int fare;

public:
   Trip()
   {
   }

   Trip(Bus *bus, string startDay, string startTime, string endDay, string
endTime, int fare)
   {
       this->bus = bus;
       this->startDay = startDay;
       this->startTime = startTime;
       this->endDay = endDay;
       this->endTime = endTime;
       this->fare = fare;
   }

   Seat bookTrip()
   {
       return bus->bookSeat();
   }

   void cancelTrip(Seat seat)
   {
       bus->cancelSeat(seat);
   }

   void displayTrip()
   {
       cout << "Bus tier: " << bus->tier << endl;
       cout << "Start day: " << startDay << endl;
       cout << "Start time: " << startTime << endl;
       cout << "End day: " << endDay << endl;
```

```
        cout << "End time: " << endTime << endl;
        bus->displayVacantSeats();
        cout << "Orignal fare: " << fare << endl;
    }

    int discount()
    {
        cout << "You have been selected for a discount round wherein you will
have to perform a simple task and on the basis of that you will be awarded a
discount on fare" << endl;
        cout << "Enter 2 numbers from 1 to 10:-" << endl;
        cout << "First number: ";
        int a;
        cin >> a;
        cin.ignore();
        cout << "Second number: ";
        int b;
        cin >> b;
        cin.ignore();
        int discount;
        int prevFare = fare;
        int c = a | b;
        if (c > 13)
        {
            cout << "Congratulations! You have got a flat 10% reduction in
fares" << endl;
            discount = 10;
            prevFare = fare * (100 - discount) / 100;
        }
        else if (c<13 | c> 9)
        {
            cout << "Congratulations! You have got a flat 7% reduction in fares"
<< endl;
            discount = 7;
            prevFare = fare * (100 - discount) / 100;
        }
        else if (c<9 | c> 5)
        {
            cout << "Congratulations! You have got a flat 5% reduction in fares"
<< endl;
            discount = 5;
            prevFare = fare * (100 - discount) / 100;
        }
        else if (c<5 & c> 0)
```

```
        {
            cout << "You have got no reduction in fares! Better luck next time!"
<< endl;
        }
        return prevFare;
    }

    void editTrip()
    {
        cout << "Enter start day: ";
        getline(cin, startDay);
        cout << "Enter start time: ";
        getline(cin, startTime);
        cout << "Enter end day: ";
        getline(cin, endDay);
        cout << "Enter end time: ";
        getline(cin, endTime);
        cout << "Enter fare: ";
        cin >> fare;
        cin.ignore();
        cout << "Trip edited!" << endl;
    }

    void clearTrip()
    {
        bus->clearBus();
    }

    void remove()
    {
        delete bus;
    }
};

class Ticket
{
private:
    static const string source;

    string destination;
    Trip trip;
    Seat seat;
    int fare;
```

```
    void setFare()
    {
        fare = trip.discount();
    }

public:
    Ticket(string destination, Trip trip, Seat seat)
    {
        this->destination = destination;
        this->trip = trip;
        this->seat = seat;
        setFare();
    }

    void displayTicket()
    {
        cout << "Source: " << source << endl;
        cout << "Destination: " << destination << endl;
        trip.displayTrip();
        cout << "Discounted fare: " << fare << endl;
        seat.displaySeat();
    }

    void cancelTicket()
    {
        trip.cancelTrip(seat);
    }
};

string const Ticket::source = "Delhi";

class Log
{
private:
    unordered_map<unsigned long, Ticket> mapTicket;
    unordered_map<string, vector<Trip>> mapTrips;

    vector<Trip> mumbaiList()
    {
        vector<Trip> trips;
        trips.push_back(Trip(new Tier1(), "Tuesday", "1200", "Thursday", "1200",
5000));
        trips.push_back(Trip(new Tier2(), "Saturday", "1200", "Monday", "1200",
3000));
```

```
        trips.push_back(Trip(new Tier3(), "Thursday", "1800", "Saturday",
"1800", 1700));
        return trips;
    }

    vector<Trip> jaipurList()
    {
        vector<Trip> trips;
        trips.push_back(Trip(new Tier1(), "Wednesday", "1200", "Thursday",
"1200", 3500));
        trips.push_back(Trip(new Tier2(), "Sunday", "1200", "Monday", "1200",
2500));
        trips.push_back(Trip(new Tier3(), "Friday", "1800", "Saturday", "1800",
1000));
        return trips;
    }

    vector<Trip> chennaiList()
    {
        vector<Trip> trips;
        trips.push_back(Trip(new Tier1(), "Thursday", "1200", "Sunday", "1200",
7500));
        trips.push_back(Trip(new Tier2(), "Tuesday", "1800", "Friday", "1800",
5000));
        trips.push_back(Trip(new Tier3(), "Saturday", "1800", "Tuesday", "1800",
3500));
        return trips;
    }

    vector<Trip> kolkataList()
    {
        vector<Trip> trips;
        trips.push_back(Trip(new Tier1(), "Friday", "1200", "Sunday", "1200",
5500));
        trips.push_back(Trip(new Tier2(), "Wednesday", "1800", "Friday", "1800",
4000));
        trips.push_back(Trip(new Tier3(), "Sunday", "1800", "Tuesday", "1800",
2500));
        return trips;
    }

    vector<Trip> destinationList()
    {
        vector<Trip> trips;
```

```
        for (int i = 1; i <= 3; i++)
        {
            cout << "Enter details for tier: " << i << endl;
            cout << "Enter start day: ";
            string startDay;
            getline(cin, startDay);
            cout << "Enter start time: ";
            string startTime;
            getline(cin, startTime);
            cout << "Enter end day: ";
            string endDay;
            getline(cin, endDay);
            cout << "Enter end time: ";
            string endTime;
            getline(cin, endTime);
            cout << "Enter fare: ";
            int fare;
            cin >> fare;
            cin.ignore();
            trips.push_back(Trip(getTierBus(i), startDay, startTime, endDay,
endTime, fare));
            cout << "Trip added!" << endl;
            cout << endl;
        }
        return trips;
    }

    Bus *getTierBus(int tier)
    {
        Bus *bus;
        if (tier == 1)
        {
            bus = new Tier1();
        }
        else if (tier == 2)
        {
            bus = new Tier2();
        }
        bus = new Tier3();
        return bus;
    }

    void displayTrips(string location)
    {
```

```
        vector<Trip> trips = mapTrips.at(location);
        for (Trip trip : trips)
        {
            trip.displayTrip();
            cout << endl;
        }
    }

public:
    Log()
    {
        mapTrips.emplace("Mumbai", mumbaiList());
        mapTrips.emplace("Jaipur", jaipurList());
        mapTrips.emplace("Chennai", chennaiList());
        mapTrips.emplace("Kolkata", kolkataList());
    }

    void booking()
    {
        cout << "Enter destination: ";
        string location;
        getline(cin, location);
        cout << "Available choices:-" << endl;
        displayTrips(location);
        cout << endl;
        cout << "Enter your bus tier: ";
        int tier;
        cin >> tier;
        cin.ignore();
        Trip trip = mapTrips.at(location).at(tier - 1);
        Seat seat = trip.bookTrip();
        cout << endl;
        Ticket ticket(location, trip, seat);
        cout << endl;
        srand(time(NULL));
        unsigned long ticketNumber = rand();
        mapTicket.emplace(ticketNumber, ticket);
        cout << "Ticket Number: " << ticketNumber << endl;
        ticket.displayTicket();
        cout << endl;
    }

    void cancelling()
    {
```

```
        cout << "Please enter the ticket number: ";
        unsigned long ticketNumber;
        cout << "Please enter the ticket number: ";
        cin >> ticketNumber;
        cin.ignore();
        if (mapTicket.find(ticketNumber) != mapTicket.end())
        {
            cout << "Are you sure, You want to cancel the trip? Press Y or N: ";
            char choice;
            cin >> choice;
            cin.ignore();
            if (choice == 'Y')
            {
                Ticket ticket = mapTicket.at(ticketNumber);
                ticket.cancelTicket();
                cout << endl;
                mapTicket.erase(ticketNumber);
            }
            else if (choice == 'N')
            {
                cout << "Ticket not cancelled!" << endl;
            }
        }
        else
        {
            cout << "The ticket number does not exist in our database!" << endl;
        }
        cout << endl;
    }

    void rescheduling()
    {
        cout << "Your previous ticket will be cancelled and new ticket will be
booked" << endl;
        unsigned long ticketNumber;
        cout << "Please enter the ticket number: ";
        cin >> ticketNumber;
        cin.ignore();
        if (mapTicket.find(ticketNumber) != mapTicket.end())
        {
            cout << "Are you sure, You want to re-schedule the trip? Press Y or
N: ";
            char choice;
            cin >> choice;
```

```
            cin.ignore();
            if (choice == 'Y')
            {
                Ticket ticket = mapTicket.at(ticketNumber);
                ticket.cancelTicket();
                cout << endl;
                mapTicket.erase(ticketNumber);
                booking();
            }
            else if (choice == 'N')
            {
                cout << "Ticket not re-scheduled!" << endl;
            }
        }
        else
        {
            cout << "The ticket number does not exist in our database!" << endl;
        }
        cout << endl;
    }

    void printBus()
    {
        cout << "Enter bus tier: ";
        int tier;
        cin >> tier;
        cin.ignore();
        if (tier == 1)
        {
            Tier1::printInfo();
        }
        else if (tier == 2)
        {
            Tier2::printInfo();
        }
        else if (tier == 3)
        {
            Tier3::printInfo();
        }
        else
        {
            cout << "Tier does not exist!" << endl;
        }
        cout << endl;
```

```
}

void printTrip()
{
    cout << "Enter destination: ";
    string location;
    getline(cin, location);
    cout << "Enter bus tier: ";
    int tier;
    cin >> tier;
    cin.ignore();
    vector<Trip> trips = mapTrips.at(location);
    Trip trip = trips.at(tier - 1);
    trip.displayTrip();
    cout << endl;
}

void printTrips()
{
    cout << "Enter destination: ";
    string location;
    getline(cin, location);
    displayTrips(location);
}

void printLogs()
{
    for (auto &pair : mapTrips)
    {
        string location = pair.first;
        cout << "Destination: " << location << endl;
        displayTrips(location);
    }
}

void printTicket()
{
    cout << "Please enter the ticket number: ";
    unsigned long ticketNumber;
    cout << "Please enter the ticket number: ";
    cin >> ticketNumber;
    cin.ignore();
    if (mapTicket.find(ticketNumber) != mapTicket.end())
    {
```

```
            Ticket ticket = mapTicket.at(ticketNumber);
            ticket.displayTicket();
        }
        else
        {
            cout << "The ticket number does not exist in our database!" << endl;
        }
        cout << endl;
    }

    void editBus()
    {
        cout << "Enter tier: ";
        int tier;
        cin >> tier;
        cin.ignore();
        if (tier == 1)
        {
            Tier1::changeBus();
        }
        else if (tier == 2)
        {
            Tier2::changeBus();
        }
        else if (tier == 3)
        {
            Tier3::changeBus();
        }
        else
        {
            cout << "Tier does not exist!" << endl;
        }
        cout << endl;
    }

    void resetTrip()
    {
        cout << "Enter destination: ";
        string location;
        getline(cin, location);
        cout << "Enter bus tier: ";
        int tier;
        cin >> tier;
        cin.ignore();
```

```
    vector<Trip> trips = mapTrips.at(location);
    Trip trip = trips.at(tier - 1);
    trip.clearTrip();
    cout << endl;
}

void addDestination()
{
    cout << "Enter destination: ";
    string destination;
    getline(cin, destination);
    mapTrips.emplace(destination, destinationList());
    cout << "Destination added!" << endl;
    cout << endl;
}

void removeDestination()
{
    cout << "Enter destination: ";
    string location;
    getline(cin, location);
    if (mapTrips.find(location) != mapTrips.end())
    {
        mapTrips.erase(location);
        cout << "Destination removed!" << endl;
    }
    else
    {
        cout << "Destination not present!" << endl;
    }
    cout << endl;
}

void editTrip()
{
    cout << "Enter destination: ";
    string location;
    getline(cin, location);
    cout << "Enter bus tier: ";
    int tier;
    cin >> tier;
    cin.ignore();
    vector<Trip> trips = mapTrips.at(location);
    Trip trip = trips.at(tier - 1);
```

```
        trip.editTrip();
        cout << endl;
    }

    ~Log()
    {
        for (auto &pair : mapTrips)
        {
            vector<Trip> trips = pair.second;
            for (Trip trip : trips)
            {
                trip.remove();
            }
        }
    }
};

Log storage;

static void readLogs()
{
    ifstream file;
    file.open("logs.dat");
    if (!file.is_open())
    {
        return;
    }
    file.seekg(ios::end);
    long size = file.tellg();
    file.seekg(ios::beg);
    file.read((char *)&storage, size);
    file.close();
}

void writeLogs()
{
    ofstream file;
    file.open("logs.dat", ios::trunc);
    file.write((char *)&storage, sizeof(storage));
    cout << sizeof(storage);
    file.close();
}

void defaultPassword()
```

```
{
    fstream file;
    file.open("password.dat", ios::in);
    if (file.is_open())
    {
        file.close();
        return;
    }
    file.open("password.dat", ios::out);
    string password = "abcd";
    file << password;
    file.close();
}

void changePassword()
{
    ofstream file;
    file.open("password.dat", ios::trunc);
    cout << "New password: ";
    string password;
    getline(cin, password);
    file << password;
    file.close();
    cout << "Password changed!" << endl;
    cout << endl;
}

string getPassword()
{
    ifstream file;
    file.open("password.dat");
    string password;
    getline(file, password);
    file.close();
    return password;
}

void customerMode()
{
    int choice;
    do
    {
        cout << endl;
        cout << "1. Booking" << endl;
```

```
        cout << "2. Cancelling" << endl;
        cout << "3. Re-schedule/edit details" << endl;
        cout << "4. Print bus information" << endl;
        cout << "5. Print a trip" << endl;
        cout << "6. Print all trips to a location" << endl;
        cout << "7. Print all logs" << endl;
        cout << "8. Print ticket" << endl;
        cout << "9. Log out" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore();
        cout << endl;
        switch (choice)
        {
        case 1:
            ::storage.booking();
            break;
        case 2:
            ::storage.cancelling();
            break;
        case 3:
            ::storage.rescheduling();
            break;
        case 4:
            ::storage.printBus();
            break;
        case 5:
            ::storage.printTrip();
            break;
        case 6:
            ::storage.printTrips();
            break;
        case 7:
            ::storage.printLogs();
            break;
        case 8:
            ::storage.printTicket();
            break;
        case 9:
            cout << "Logged out" << endl;
            break;
        default:
            cout << "Choice doesn't match options, try again" << endl;
            break;
```

```
        }
    } while (choice != 9);
}

void adminMode()
{
    defaultPassword();
    cout << "Enter admin password: ";
    string password;
    getline(cin, password);
    string key = getPassword();
    if (key.compare(password) != 0)
    {
        cout << "Wrong password entered, entry denied!" << endl;
        return;
    }
    int choice;
    do
    {
        cout << endl;
        cout << "1. Booking" << endl;
        cout << "2. Cancelling" << endl;
        cout << "3. Re-schedule/edit details" << endl;
        cout << "4. Print bus information" << endl;
        cout << "5. Print a trip" << endl;
        cout << "6. Print all trips to a location" << endl;
        cout << "7. Print all logs" << endl;
        cout << "8. Print ticket" << endl;
        cout << "9. Change password" << endl;
        cout << "10. Edit bus information" << endl;
        cout << "11. Reset bus/trip" << endl;
        cout << "12. Add new destination" << endl;
        cout << "13. Remove destination" << endl;
        cout << "14. Reset logs" << endl;
        cout << "15. Edit trip" << endl;
        cout << "16. Log out" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore();
        cout << endl;
        switch (choice)
        {
        case 1:
            ::storage.booking();
```

```
            break;
        case 2:
            ::storage.cancelling();
            break;
        case 3:
            ::storage.rescheduling();
            break;
        case 4:
            ::storage.printBus();
            break;
        case 5:
            ::storage.printTrip();
            break;
        case 6:
            ::storage.printTrips();
            break;
        case 7:
            ::storage.printLogs();
            break;
        case 8:
            ::storage.printTicket();
            break;
        case 9:
            changePassword();
            break;
        case 10:
            ::storage.editBus();
            break;
        case 11:
            ::storage.resetTrip();
            break;
        case 12:
            ::storage.addDestination();
            break;
        case 13:
            ::storage = Log();
            break;
        case 14:
            ::storage.removeDestination();
            break;
        case 15:
            ::storage.editTrip();
            break;
        case 16:
```

```
            cout << "Logged out" << endl;
            break;
        default:
            cout << "Choice doesn't match options, try again" << endl;
            break;
        }
    } while (choice != 16);
}

int main()
{
    readLogs();
    int choice;
    do
    {
        cout << endl;
        cout << "Log in as:-" << endl;
        cout << "1. Admin" << endl;
        cout << "2. Customer" << endl;
        cout << "3. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore();
        cout << endl;
        switch (choice)
        {
        case 1:
            adminMode();
            break;
        case 2:
            customerMode();
            break;
        case 3:
            writeLogs();
            cout << "Exited" << endl;
            break;
        default:
            cout << "Choice doesn't match options, try again" << endl;
            break;
        }
    } while (choice != 3);
}
```