Bus Booking System

By Sidhant Choudhary,2019B5A70720G

https://github.com/sidhant0720/Bus-Booking-System-Moveinsync

Case Study Requirements -

Design a Bus Booking System
In this system, there are two distinct user roles:

A. Admin:
The admin possesses the authority to:
Manage Bus Details:

- Add, update, and delete information related to buses.
- Bus Details include:
- Bus Name
- Total Number of Seats
- Current Occupancy
- Available Davs of Operation for each bus.
- Bus route Creation

B. User:
The user is endowed with the following capabilities:

Browse Available Buses:

- Retrieve information on the number of buses available between the source and destination.
- Display the distance and Estimated Time of Arrival (ETA) for each bus.

Check Seat Availability:

- View the number of seats available on a specific bus.

Seat Booking:

- Reserve a seat on a selected bus.
- Users are presented with buses located near them first.

Cancel Seat Booking:

- Cancel a previously booked seat on the bus.

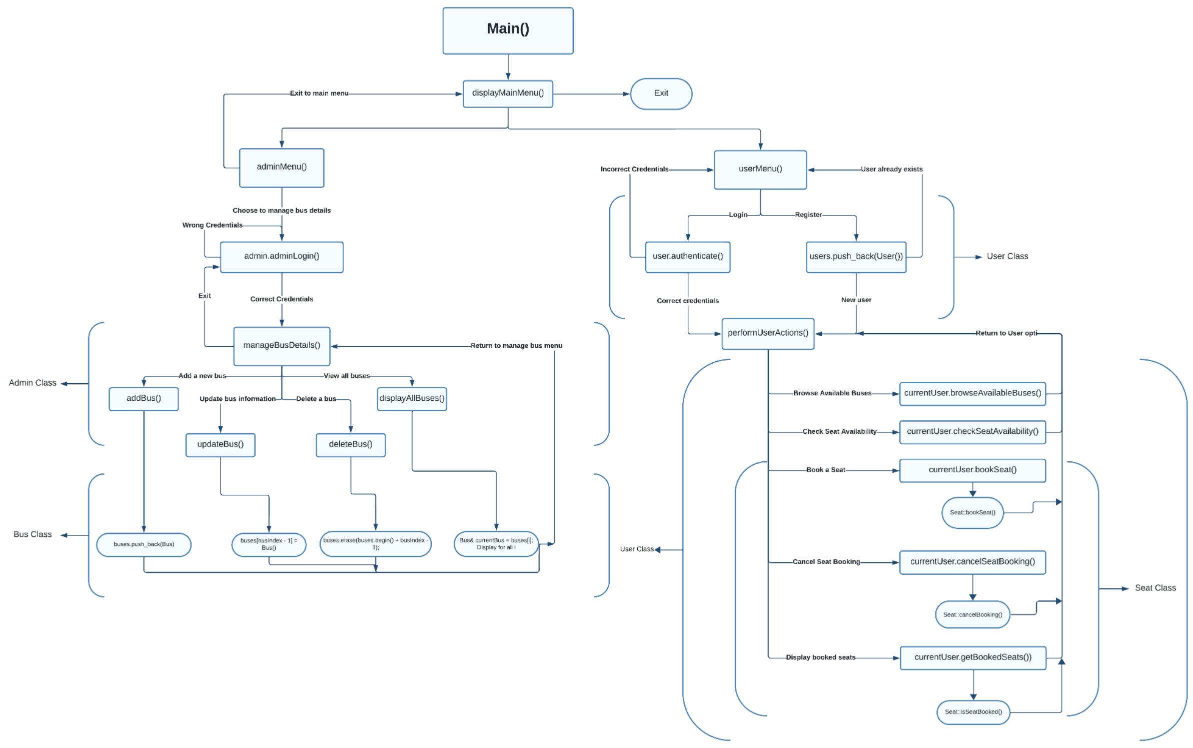
Constraints:

- After booking, each user is assigned a distinctive seat number.
- Implement measures to ensure that two users cannot concurrently book the same seat.
- The color codes will provide users with a quick and intuitive way to assess the availability of bus seats. Here's how you can define the constraint:
  Green (60% Full or Less):
  If the occupancy percentage is 60% or less, display seats in green.

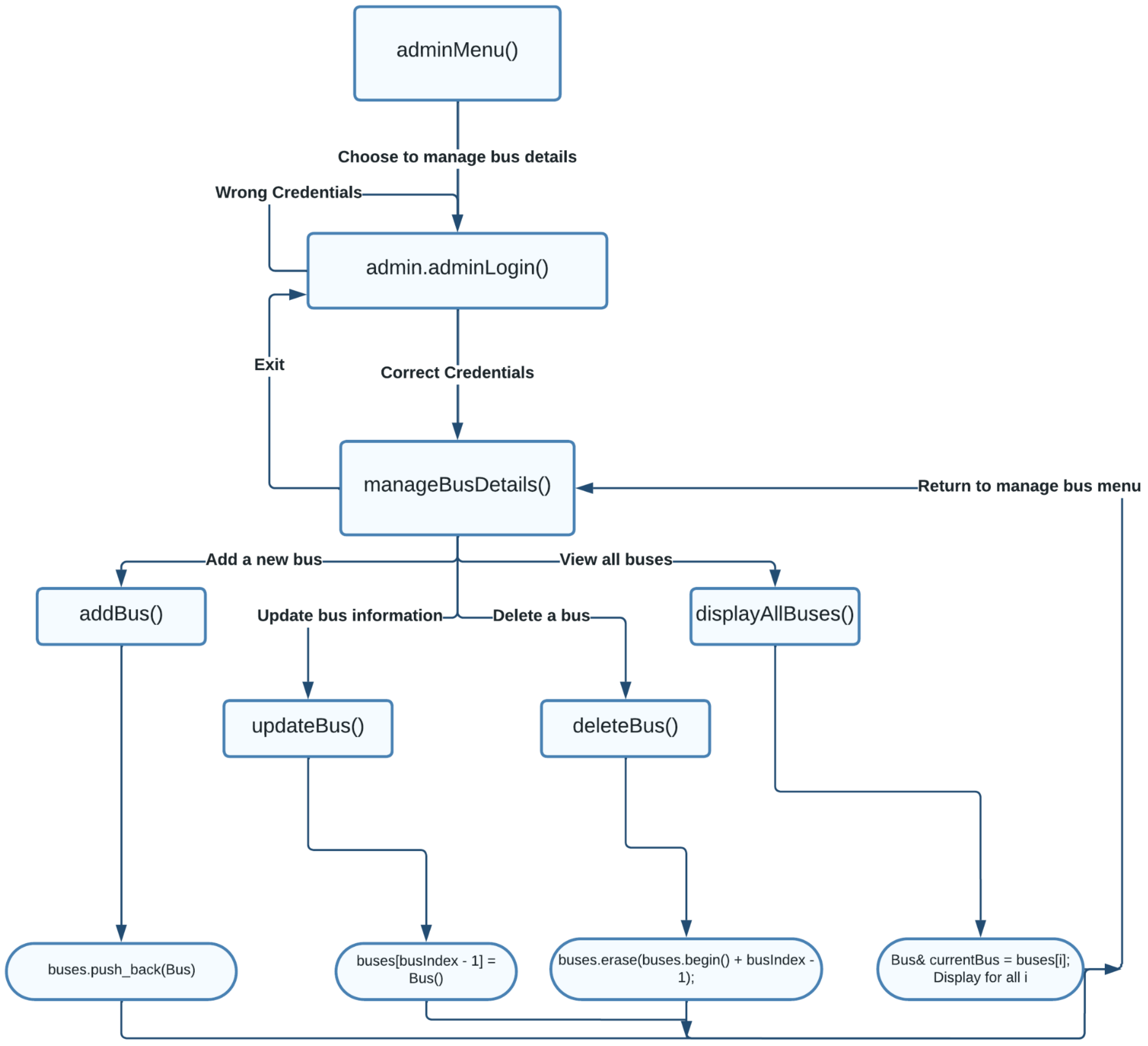
Indicates that there is ample availability, and users can comfortably book their seats.
Yellow (60% - 90% Full):
If the occupancy percentage is between 60% and 90%, display seats in yellow. Serves as a cautionary indicator. informing users that seats are filling up but there is still moderate availability.
Red (90% - 100% Full):
If the occupancy percentage is between 90% and 100%, display seats in red. Indicates limited availability. and users should act quickly to secure their seats.

Interface and User Experience

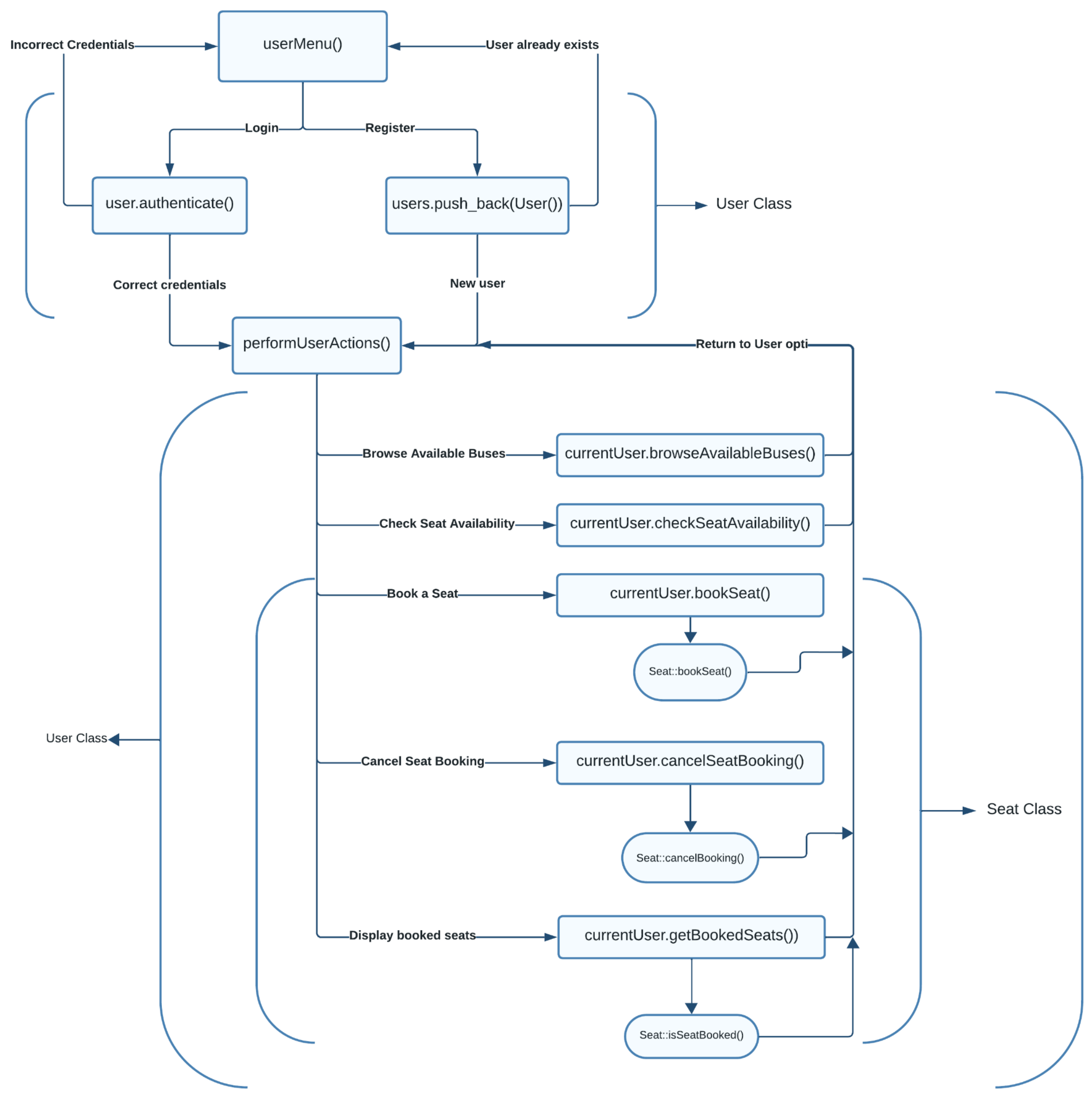
Flow of the program -



Zoomed in view of admin menu -

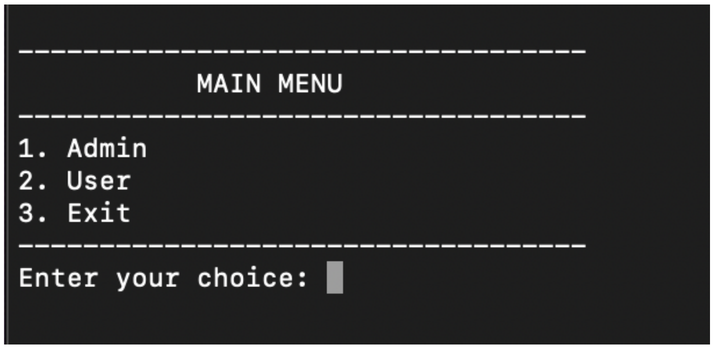


Zoomed in view of user menu -



Main menu -

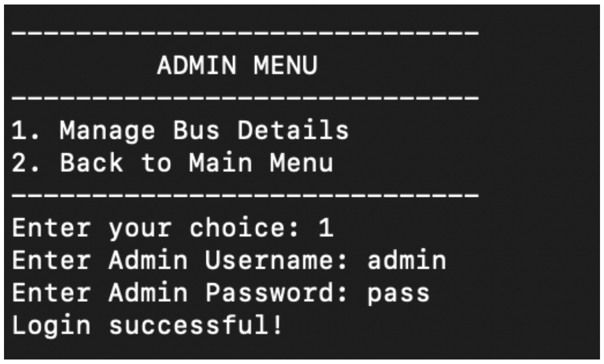
The program starts by displaying the main menu. Users can choose between Admin, User, or Exit.



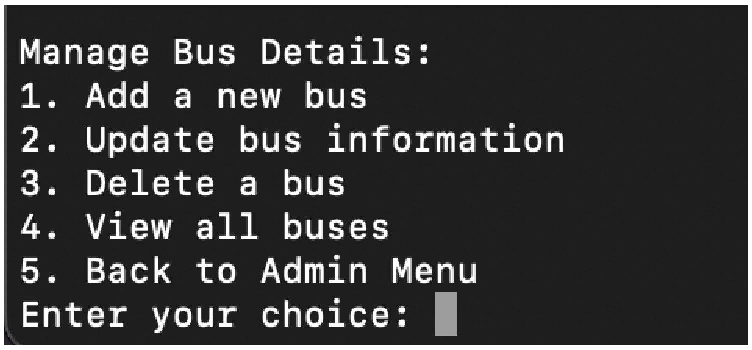
```
-----------------------------------
           MAIN MENU
-----------------------------------
1. Admin
2. User
3. Exit
-----------------------------------
Enter your choice: 
```

Admin Menu -

Choosing the admin option takes you to the login page for admin , after successful login you can access the admin options panel.

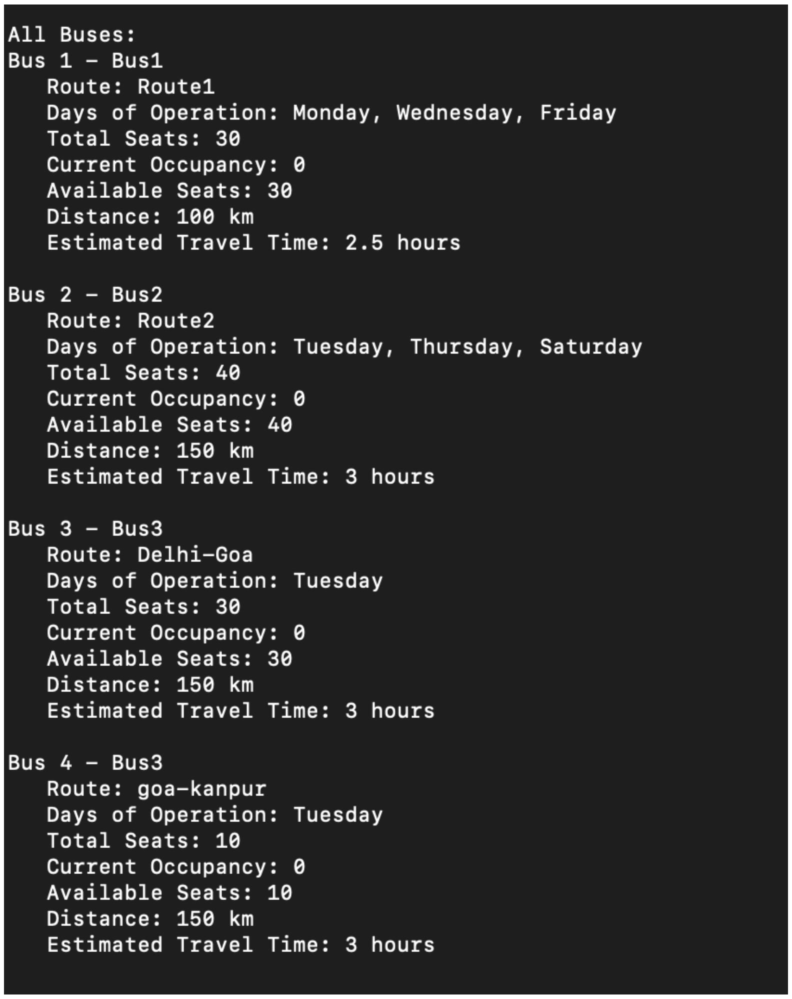


```
-----------------------------
         ADMIN MENU
-----------------------------
1. Manage Bus Details
2. Back to Main Menu
-----------------------------
Enter your choice: 1
Enter Admin Username: admin
Enter Admin Password: pass
Login successful!
```



```
Manage Bus Details:
1. Add a new bus
2. Update bus information
3. Delete a bus
4. View all buses
5. Back to Admin Menu
Enter your choice: 
```

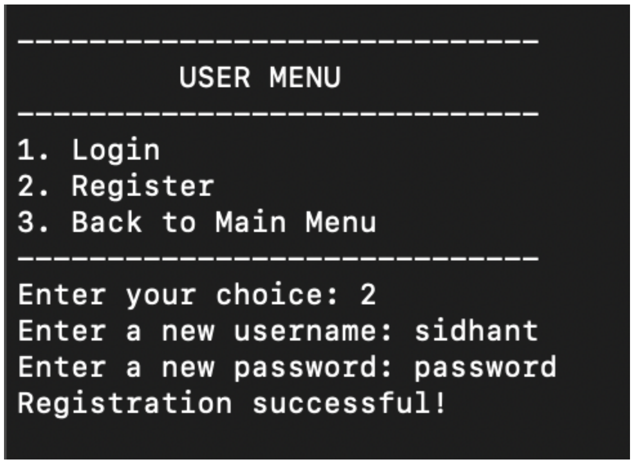
From the admin panel, you have the option to add new buses and routes, update existing buses, delete a bus, or view all the current buses.



```
All Buses:
Bus 1 - Bus1
   Route: Route1
   Days of Operation: Monday, Wednesday, Friday
   Total Seats: 30
   Current Occupancy: 0
   Available Seats: 30
   Distance: 100 km
   Estimated Travel Time: 2.5 hours

Bus 2 - Bus2
   Route: Route2
   Days of Operation: Tuesday, Thursday, Saturday
   Total Seats: 40
   Current Occupancy: 0
   Available Seats: 40
   Distance: 150 km
   Estimated Travel Time: 3 hours

Bus 3 - Bus3
   Route: Delhi-Goa
   Days of Operation: Tuesday
   Total Seats: 30
   Current Occupancy: 0
   Available Seats: 30
   Distance: 150 km
   Estimated Travel Time: 3 hours

Bus 4 - Bus3
   Route: goa-kanpur
   Days of Operation: Tuesday
   Total Seats: 10
   Current Occupancy: 0
   Available Seats: 10
   Distance: 150 km
   Estimated Travel Time: 3 hours
```

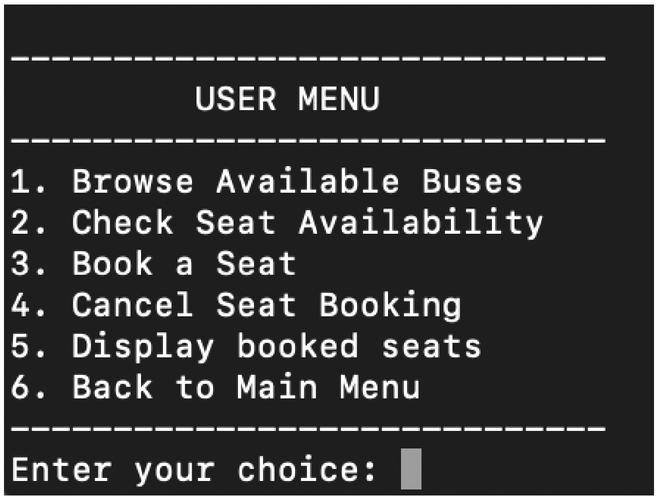
User Menu -

Choosing the user menu, takes you to the user login page, where you can register if you are a first time user, or login directly if you are already registered.



```
------------------------------
          USER MENU
------------------------------
1. Login
2. Register
3. Back to Main Menu
------------------------------
Enter your choice: 2
Enter a new username: sidhant
Enter a new password: password
Registration successful!
```

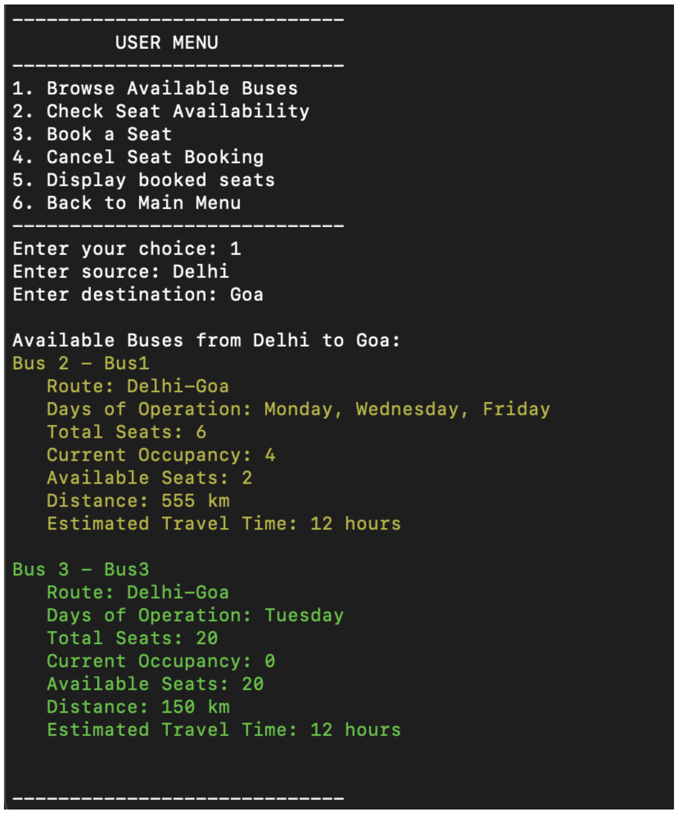
After successful login , user is taken to the user menu where he has the option to Browse all the available buses between a destination and source. Check the seat availability , book a seat, cancel a seat booking, fetch all the tickets booked by them.



```
------------------------------
          USER MENU
------------------------------
1. Browse Available Buses
2. Check Seat Availability
3. Book a Seat
4. Cancel Seat Booking
5. Display booked seats
6. Back to Main Menu
------------------------------
Enter your choice: 
```

1 - Browse Available buses -

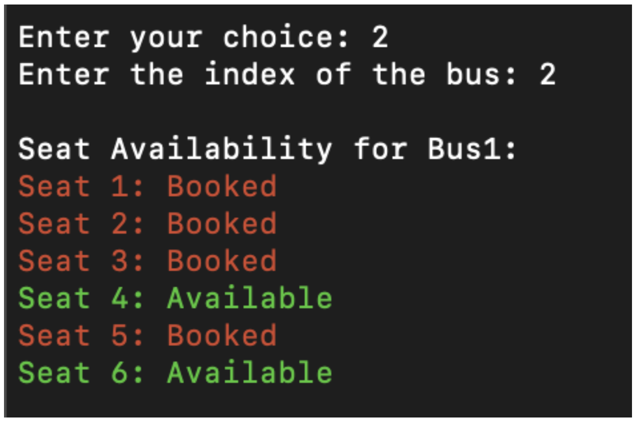
The user can input the source and destination, and they will be presented with all the available buses for that route. The buses are color-coded based on the percentage of seats available. The color codes are green for less than 60% booked, yellow for 60-90% booked, and red if the bus is booked more than 90%.



```
-----------------------------
         USER MENU
-----------------------------
1. Browse Available Buses
2. Check Seat Availability
3. Book a Seat
4. Cancel Seat Booking
5. Display booked seats
6. Back to Main Menu
-----------------------------
Enter your choice: 1
Enter source: Delhi
Enter destination: Goa

Available Buses from Delhi to Goa:
Bus 2 - Bus1
   Route: Delhi-Goa
   Days of Operation: Monday, Wednesday, Friday
   Total Seats: 6
   Current Occupancy: 4
   Available Seats: 2
   Distance: 555 km
   Estimated Travel Time: 12 hours

Bus 3 - Bus3
   Route: Delhi-Goa
   Days of Operation: Tuesday
   Total Seats: 20
   Current Occupancy: 0
   Available Seats: 20
   Distance: 150 km
   Estimated Travel Time: 12 hours


-----------------------------
```

2 - Check seat availability

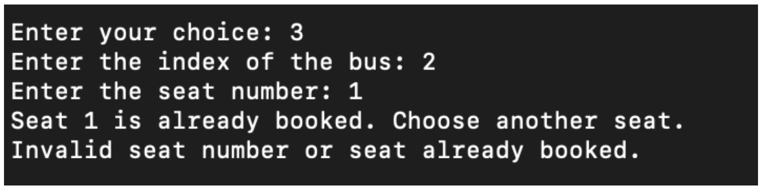
User can check the seat status on any of the busses, the seats are also color coded for ease of Understanding.



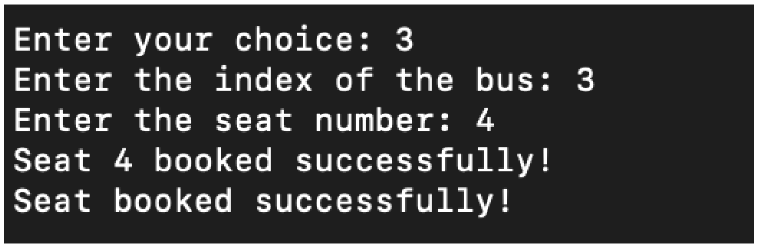
```
Enter your choice: 2
Enter the index of the bus: 2

Seat Availability for Bus1:
Seat 1: Booked
Seat 2: Booked
Seat 3: Booked
Seat 4: Available
Seat 5: Booked
Seat 6: Available
```

3 - Book a seat

The user has the ability to reserve any available seat on any bus but cannot book a seat that has already been reserved.



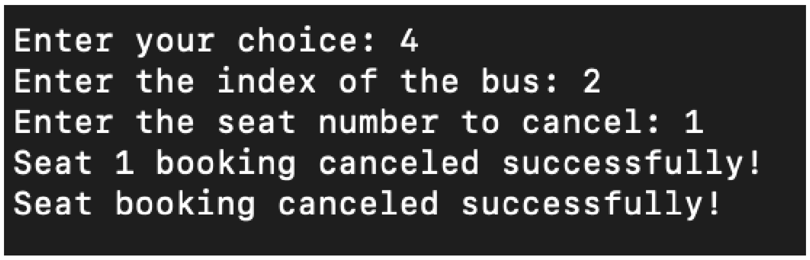
```
Enter your choice: 3
Enter the index of the bus: 2
Enter the seat number: 1
Seat 1 is already booked. Choose another seat.
Invalid seat number or seat already booked.
```



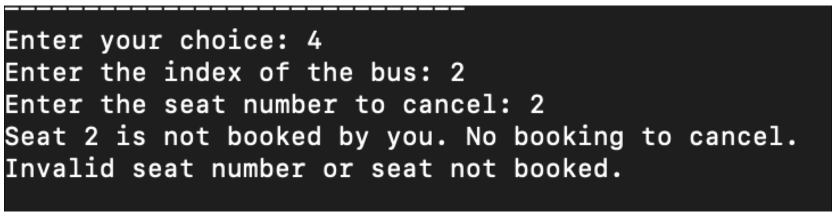
```
Enter your choice: 3
Enter the index of the bus: 3
Enter the seat number: 4
Seat 4 booked successfully!
Seat booked successfully!
```

4 - Cancel Seat booking

Users have the option to cancel any ticket they have booked, but they are unable to cancel a ticket that has been booked by someone else.



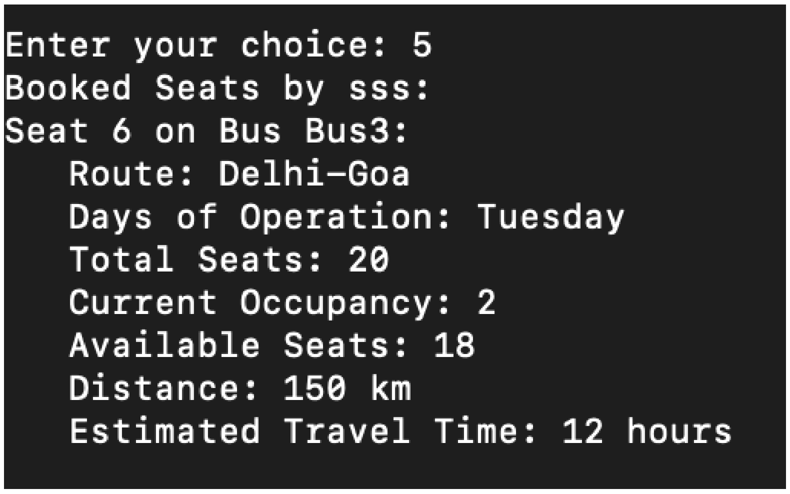
```
Enter your choice: 4
Enter the index of the bus: 2
Enter the seat number to cancel: 1
Seat 1 booking canceled successfully!
Seat booking canceled successfully!
```



```
-----------------------------
Enter your choice: 4
Enter the index of the bus: 2
Enter the seat number to cancel: 2
Seat 2 is not booked by you. No booking to cancel.
Invalid seat number or seat not booked.
```

5 - Display booked seats

Users can get the list of all the seats booked by them.



```
Enter your choice: 5
Booked Seats by sss:
Seat 6 on Bus Bus3:
   Route: Delhi-Goa
   Days of Operation: Tuesday
   Total Seats: 20
   Current Occupancy: 2
   Available Seats: 18
   Distance: 150 km
   Estimated Travel Time: 12 hours
```

Features -

List of features in this bus reservation system project :

Admin Features:

- Admin login with username and password.
- Ability to add a new bus with details such as name, total seats, route, days of operation, distance, and estimated travel time.
- Update information for an existing bus.
- Delete a bus.
- View details of all buses.

Bus Features:

- Bus class to represent each bus with associated details.
- Seat class to represent individual seats in a bus.
- Ability to book a seat, cancel a seat booking, and check seat availability.
- Distance and estimated travel time associated with each bus.

User Features:
- User registration with a new username and password.
- User login with username and password.
- Browse available buses based on source and destination.
- Check seat availability on a specific bus.
- Book a seat on a bus.
- Cancel a seat booking.
- Display booked seats by the current user.

Console Interaction:
- Simple command-line interface for user interaction.
- Main menu with options to access Admin, User, or Exit functionalities.
- Input validation for user choices and data entry.
- Visual feedback using ANSI escape codes for different seat availability statuses.

Data Management:

- Use of vectors to store information about buses and users.
- Storage and retrieval of bus and user details.

Authentication:
Basic admin authentication with a hardcoded username and password.
User authentication during login.

User Feedback:
Informative messages for successful operations and error handling.
Display of bus details, seat availability, and booked seats.

Dynamic Route Matching:
The system matches buses based on the specified source and destination routes.

Color-Coded Output:
Use of ANSI escape codes for color-coded output in the console based on seat availability and occupancy.

Looping Menus:
Ability to navigate through various menus with looping options until the user chooses to exit.

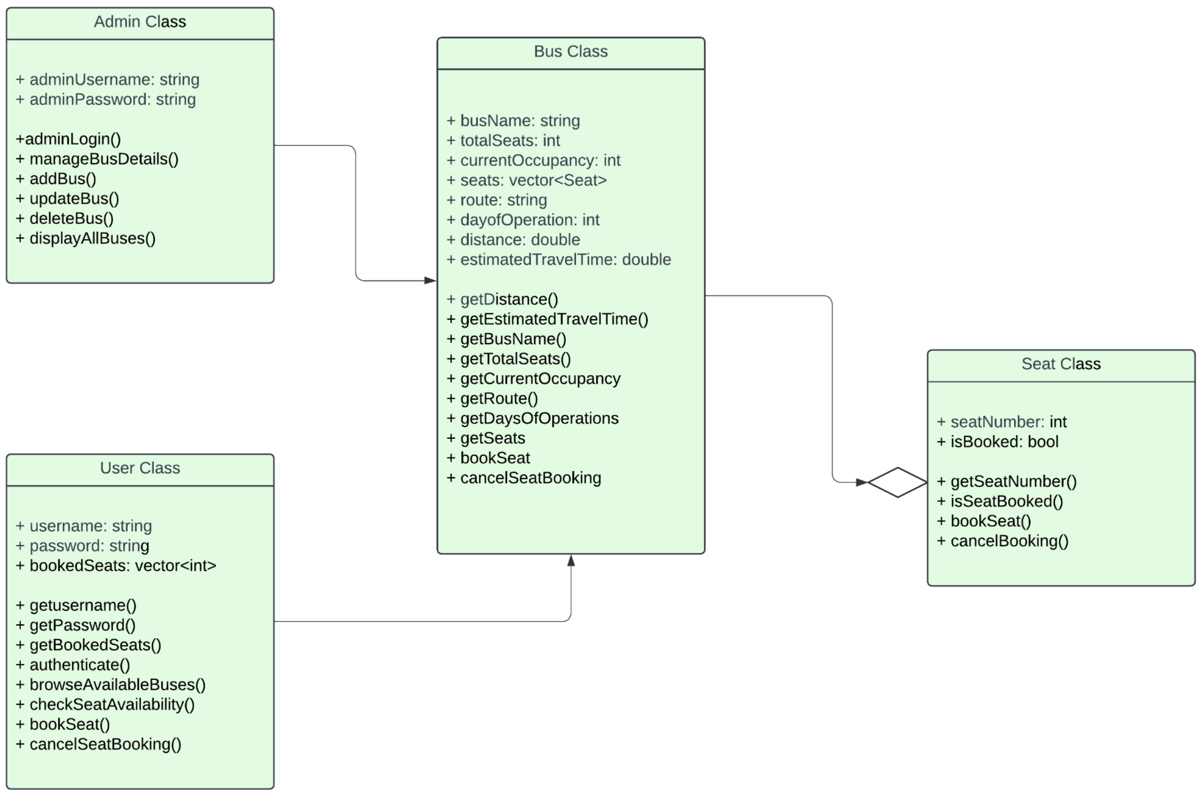
Dynamic Bus Management:

Dynamically manage buses by adding, updating, and deleting them during runtime.

Security Considerations:
Basic security measures, such as admin authentication, are implemented.

System Design

Classes:



Seat Class -

Represents a seat in a bus. Tracks seat number and booking status.
Here is the declaration for the class without the implementation details.

```
#ifndef SEAT_H
#define SEAT_H

class Seat {
private:
   int seatNumber;
   bool isBooked;

public:
   Seat(int number);

   int getSeatNumber() const;
   bool isSeatBooked() const;
   void bookSeat();
   void cancelBooking();
};

#endif // SEAT_H
```

Bus Class -

Represents a bus with details such as name, total seats, route, days of operation, distance, and estimated travel time.
Manages a collection of seats.
Provides methods to book and cancel seats.

```
#ifndef BUS_H
#define BUS_H

#include <iostream>
#include <vector>
#include <string>
#include "seat.h"

class Bus {
private:
   std::string busName;
   int totalSeats;
   int currentOccupancy;
   std::vector<Seat> seats;
   std::string route;
   std::string daysOfOperation;
   double distance;  // Added distance
   double estimatedTravelTime;  // Added estimated travel time

public:
   Bus(const std::string& name, int seats, const std::string& route, const std::string&
days,
       double distance, double estimatedTime);
   double getDistance() const;
   double getEstimatedTravelTime() const;
   const std::string& getBusName() const;
   int getTotalSeats() const;
   int getCurrentOccupancy() const;
   const std::string& getRoute() const;
   const std::string& getDaysOfOperation() const;
   const std::vector<Seat>& getSeats() const;

   bool bookSeat(int seatNumber);
   bool cancelSeatBooking(int seatNumber);
};

#endif // BUS_H
```

Admin Class -

Manages admin authentication and bus management operations. Allows adding, updating, deleting, and viewing buses.

```
#ifndef ADMIN_H
#define ADMIN_H

#include <iostream>

#include <vector>
#include "bus.h"

class Admin {
private:
   const std::string adminUsername = "admin";
   const std::string adminPassword = "pass";

public:
   void adminLogin(std::vector<Bus>& buses);
   void manageBusDetails(std::vector<Bus>& buses);

private:
   void addBus(std::vector<Bus>& buses);
   void updateBus(std::vector<Bus>& buses);
   void deleteBus(std::vector<Bus>& buses);
   void displayAllBuses(const std::vector<Bus>& buses);
};

#endif // ADMIN_H
```

User Class -

Represents a user with a username, password, and a list of booked seats.
Provides methods for browsing available buses, checking seat availability, booking, and canceling seats.

```
#ifndef USER_H
#define USER_H

#include <iostream>
#include <vector>
#include <string>
#include "bus.h"

class User {
private:
   std::string username;
   std::string password;
   std::vector<int> bookedSeats;

public:
   User();
   User(const std::string& uname, const std::string& pwd);

   const std::string& getUsername() const;
   const std::string& getPassword() const;
   const std::vector<int>& getBookedSeats() const;

   bool authenticate(const std::string& enteredUsername, const std::string& enteredPassword)
const;

   void browseAvailableBuses(const std::vector<Bus>& buses, const std::string& source, const
std::string& destination);
   void checkSeatAvailability(const Bus& bus);
   bool bookSeat(Bus& bus, int seatNumber);
   bool cancelSeatBooking(Bus& bus, int seatNumber);
};

#endif // USER_H
```

Functions:

displayMainMenu:
Displays the main menu options for choosing between Admin, User, or Exit.

adminMenu:
Handles the admin menu with options to manage bus details or go back to the main menu.

performUserActions:
Handles user actions like browsing buses, checking seat availability, booking, canceling, and displaying booked seats.

userMenu:
Manages user login, registration, and user-specific actions.

main:
Acts as the entry point of the program.
Initializes a vector of buses and provides options to access admin or user functionalities.

System Design Diagram -

Cost Estimation - Time and Space:

Time Complexity -

Admin Module:
- Login: O(1) - Constant time for username and password comparison.
- Manage Bus Details: Depends on the number of buses and operations. In general, O(n) for iterating through the buses.

User Module:
- Login: O(n) - Linear time for iterating through users to find a match.
- Register: O(n) - Linear time for checking username availability.
- Browse Available Buses: O(n) - Linear time for iterating through buses to find matching routes.
- Check Seat Availability: O(m) - Linear time for iterating through seats in a bus.
- Book Seat: O(m) - Linear time for iterating through seats to book.
- Cancel Seat Booking: O(m) - Linear time for iterating through booked seats to cancel.
- Display Booked Seats: O(k) - Linear time for iterating through booked seats.

Bus Reservation and Management:
- Booking and canceling seats involve iterating through seats in a bus, resulting in O(m) time complexity.

Main Functionality:
- Main Program: O(1) - Constant time for user input and menu navigation.

Overall time complexity depends on the user's interactions and the number of buses and seats involved.

Space Complexity -

Admin Module:
Storage of buses: O(n) - Linear space for storing bus details.

User Module:
Storage of users: O(n) - Linear space for storing user details.
Storage of booked seats: O(k) - Linear space for storing booked seat details.

Bus Reservation and Management:
Each bus object includes a vector of seats, resulting in O(m) space complexity per bus.

Main Functionality: Overall space complexity depends on the number of buses, users, and booked seats.

Authentication:

Object-Oriented Programming Language (OOPS) -

1. Encapsulation:

Implementation:
Classes such as Seat, Bus, Admin, and User encapsulate related data members and member functions.
Access specifiers (public and private) are used to control the visibility of class members.

Benefits:
Encapsulation helps in bundling data and methods that operate on the data, preventing direct access to internal details from outside the class.
It promotes information hiding and protects the integrity of the internal state.

2. Abstraction:

Implementation:
Classes abstract the underlying complexities by exposing only necessary details to the outside world.
Methods like bookSeat, cancelSeatBooking, and authenticate provide a high-level interface.

Benefits:
Abstraction simplifies the user's interaction with the system by presenting a clear, high-level interface.
It allows users (whether administrators or customers) to interact with the system without needing to understand its internal implementation.

4. Polymorphism:

Implementation:
The project exhibits compile-time polymorphism (function overloading) in various places, such as multiple bookSeat and cancelSeatBooking functions with different parameters.
Benefits:
Polymorphism allows methods to operate on objects of different classes in a uniform way.
It promotes flexibility and extensibility in the code.

5. Modularity:

Implementation:
The project is divided into modules like Admin, User, Bus, and Seat, each handling specific responsibilities.
Methods within these classes serve specific purposes, promoting modularity.

Benefits:
Modularity enhances maintainability by isolating functionalities into manageable units.
It allows for easier updates, bug fixes, and extensions to specific parts of the system without affecting the entire codebase.

Future Developments -

In the future this project can be extended to include features like -

Concurrency and Performance:
Current State:
The project may not be designed for concurrent access by multiple users.

Improvement:
Introduce concurrency controls and thread safety mechanisms to handle multiple users accessing the system simultaneously.
Prevent race conditions and ensure data integrity.

Enhanced Security Measures:
Current State:
Admin authentication is based on hardcoded credentials.

Improvement:
Implement a more secure authentication mechanism, such as using hashed passwords and salting for user credentials.
Consider role-based access control for better security.

Logging and Monitoring:
Current State:
The project lacks logging and monitoring functionalities.

Improvement:
Implement a logging system to capture important events and errors for later analysis.
Introduce monitoring features for system health and performance.

Periodic Backups:
Current State:
The project lacks a systematic approach for data backup, risking data loss during unexpected events or crashes.

Improvement:
Implement a periodic backup mechanism to save crucial data (such as bus details, user information, and booking records) to a separate file at regular intervals.