**TRAVEL AGENCY MANAGEMENT SYSTEM**

**Submitted To:**

Mam Anum Umera

**Members:**

1. Hassan Mehmood (f2023266686)
2. Anas Azam (f2023266645)
3. Haisham Faizan (f2023266619)
4. Bilal Khan Pirzada (f2023266614)

**Section:**

**V-15**

University of Management & Technology, Lhr

**INTRODUCTION:**

A travel agency management system for user:

* Sign-in & Login Feature
* Customer can book and cancel his trips and check his trips as well.
* Employee to confirm or cancel the bookings of customers.
* Admin to handle customers as well as employees.

**Design Pattern:**

Modal View Controller (**MVC**).

Singelton Pattern

**Concept:**

Object Oriented Programming Style (**OOPS**).

**Language:**

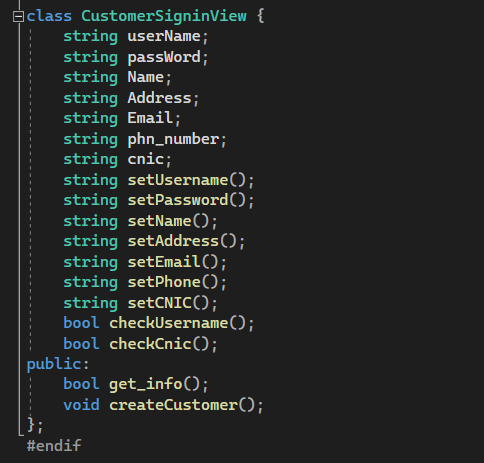
C++

**PROJECT SCOPE:**

This project covers the OOPS concept:

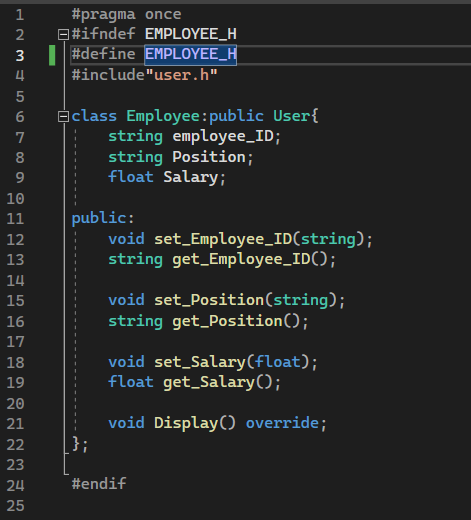
1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism
5. **Encapsulation:**

Relevant data fields and methods are encapsulated in their respective classes so that they cannot be accessed outside the member class. For example, we don’t want the fields like **username**, **password** etc. to be accessed outside the class. So, they are made private. Moreover, certain attributes and methods are also made private so they are not modified by external classes directly otherwise the program will be more vulnerable to bugs and errors.



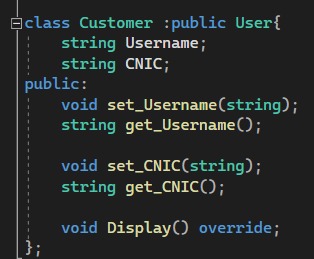
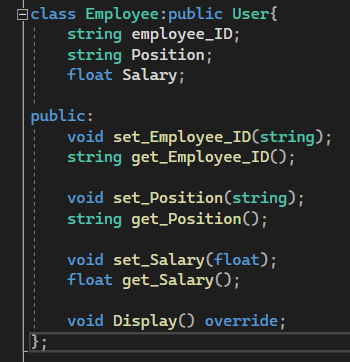
**2)Abstraction:**

Abstraction is performed to make code less complex and reuseable. It also hides the implementation from the user making it easy to use for anyone.

****

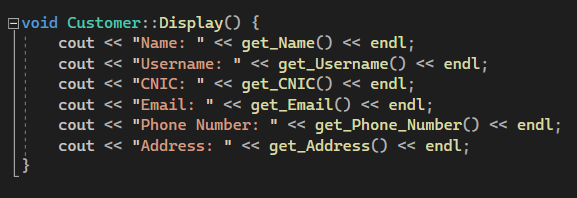
**3) Inheritance:**

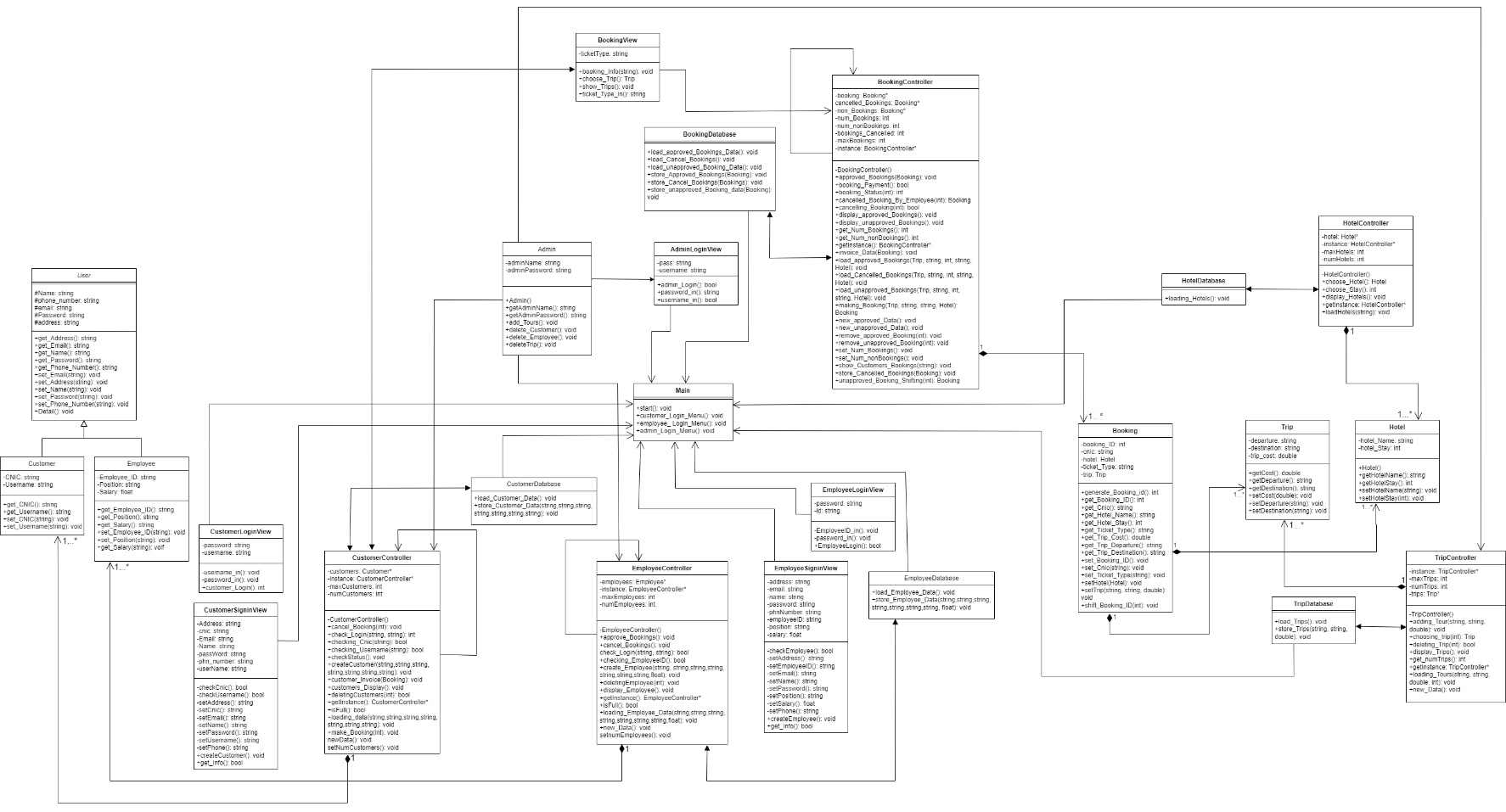
**Employee** and **Customer** are both specialized types of **User**, inheriting common attributes and methods from User while adding their own specific attributes and behaviours tailored to their respective roles within the system. This inheritance hierarchy allows for code reuse and organization based on the "is-a" relationship between different types of users in the system.

****

**4) Polymorphism:**

In the User class which contains the main method for the Display, polymorphic behaviour can be observed. This method is then used in Customer and Employee classes to display their attributes.

****

**CLASS DIAGRAM:**

**PROJECT FUNCTIONALITY:**

**Customer:**

**Customer Modal:**

Contains all the data members to store the useful information of the customer. It is a modal of customer which is used by CustomerController to store customer’s data.

**Customer Controller:**

The Customer model class defines attributes like username, password, name, address, email, phone number, and CNIC. It provides methods to set and get these attributes, ensuring structured data management. Integrated with CustomerController, it supports operations such as customer creation, data validation, and interaction management within the system, maintaining consistency and facilitating customer-related functionalities like bookings and invoices.

**Customer Database:**

CustomerDatabase handles the storage and retrieval of customer data in a CSV format, ensuring that the application can persist and retrieve customer information efficiently across sessions.

**Customer Login View:**

CustomerLoginView handles the presentation layer of customer login functionality, ensuring users can securely authenticate themselves by entering their username and password. It plays a crucial role in user interaction and authentication management within the application.

**Customer Sign-In View:**

CustomerSigninView plays a crucial role in facilitating the registration process for new customers. It ensures that customer inputs are gathered securely, validated against existing data, and passed to CustomerController for persistent storage. This integration ensures smooth customer registration while maintaining data integrity and system stability.

**Employee:**

**Employee Modal:**

Contains all the data members to store the useful information of the employee. It is a modal of employee which is used by EmployeeController to store employee’s data.

**Employee Controller:**

EmployeeController serves as a critical component in managing employee-related functionalities within the system, ranging from employee creation and authentication to handling booking approvals and cancellations. It ensures effective management and integration of employee data across various system operations.

**Employee Database:**

EmployeeDatabase serves as an interface between the application and the persistent storage (CSV file) for employee information. It handles the tasks of writing new employee data and loading existing employee data into the application's memory through interactions with EmployeeController. This separation of concerns helps in maintaining data integrity and facilitates efficient data management within the application.

**Employee Log-In View:**

EmployeeLoginView serves as the interface for employees to authenticate themselves. It handles user inputs for employee ID and password, interacts with EmployeeController to verify credentials, and manages the login retry mechanism. This separation of concerns helps in maintaining clear responsibilities between user interface interactions and business logic related to employee authentication.

**Employee Sign-In View:**

EmployeeSigninView is pivotal in ensuring that new employees can register into the system securely and efficiently, handling both user interaction and data validation before passing off the details to the controller for actual data handling and storage.

**Hotel:**

**Hotel Modal:**

Contains all the data members to store the useful information of the hotel. It is a modal of hotel which is used by HotelController to store hotel’s data.

**Hotel Controller:**

The HotelController class provides functionalities to manage and interact with a list of hotels, allowing users to choose a hotel and specify their stay duration. It encapsulates hotel-related operations and ensures robust handling of user input.

**Hotel Database:**

The HotelDatabase class serves as a bridge between the data stored in the CSV file (hoteldata.csv) and the HotelController that manages hotel-related operations within the application. It facilitates the loading of hotel data into the system during initialization or whenever required.

**Trip:**

**Trip Modal:**

Contains all the data members to store the useful information of the trip. It is a modal of trip which is used by TripController to store trip’s data.

**Trip Controller:**

The TripController facilitates CRUD (Create, Read, Update, Delete) operations for trips, interacts with the TripDatabase for data storage and retrieval, and provides methods to manage and display trip information within the travel management system.

**Trip Database:**

TripDatabase plays a crucial role in the travel management system by handling the persistent storage and retrieval of trip data, ensuring that trip information is accurately stored and easily accessible for various operations within the application.

**Bookings:**

**Booking Modal:**

Contains all the data members to store the useful information of the booking. It is a modal of Booking which is used by BookingController to store booking’s data.

**Booking Controller:**

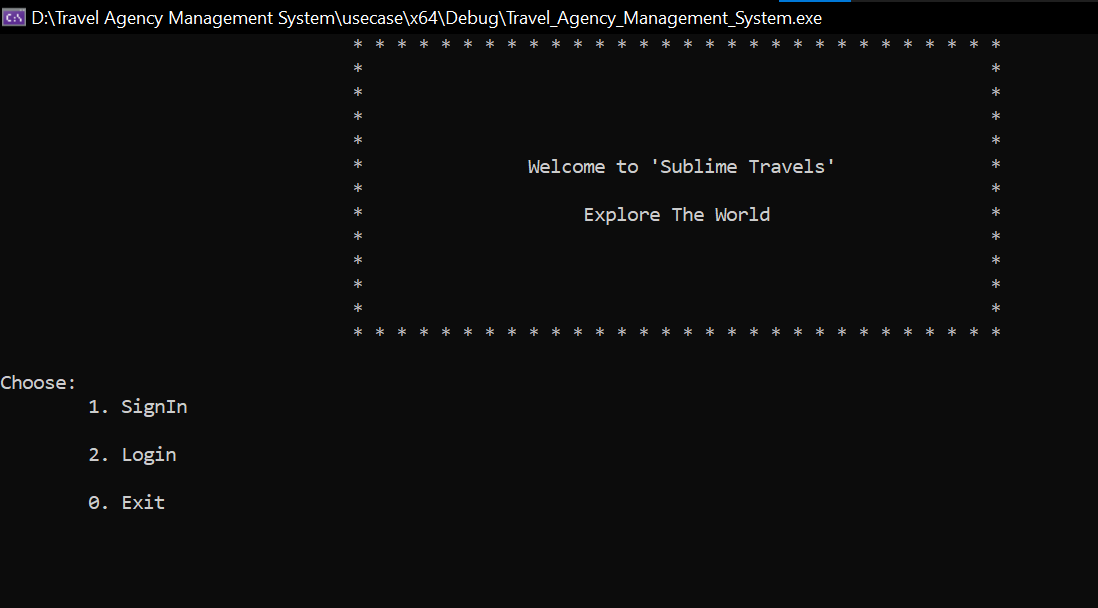
The BookingController within an application is to manage the entire lifecycle of bookings, ensuring efficient handling from creation to storage and retrieval.

**Booking Database:**

The BookingDatabase class handles the persistence of booking data, providing mechanisms to store and retrieve booking information from files.

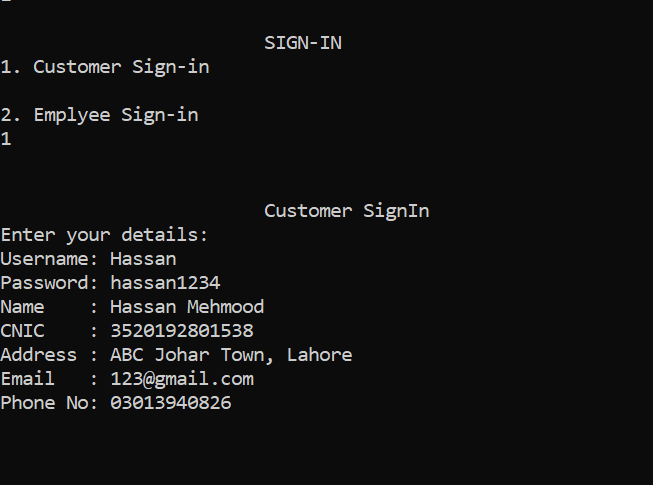
**OUTPUTS:-**

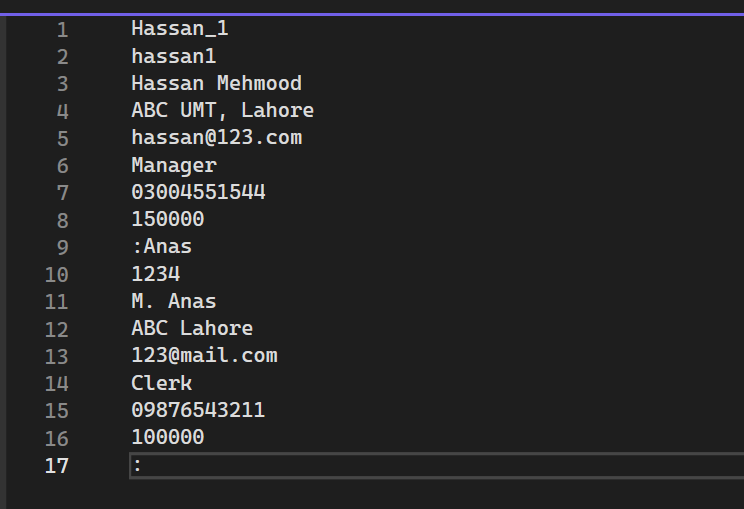
1. **Main Menu:**



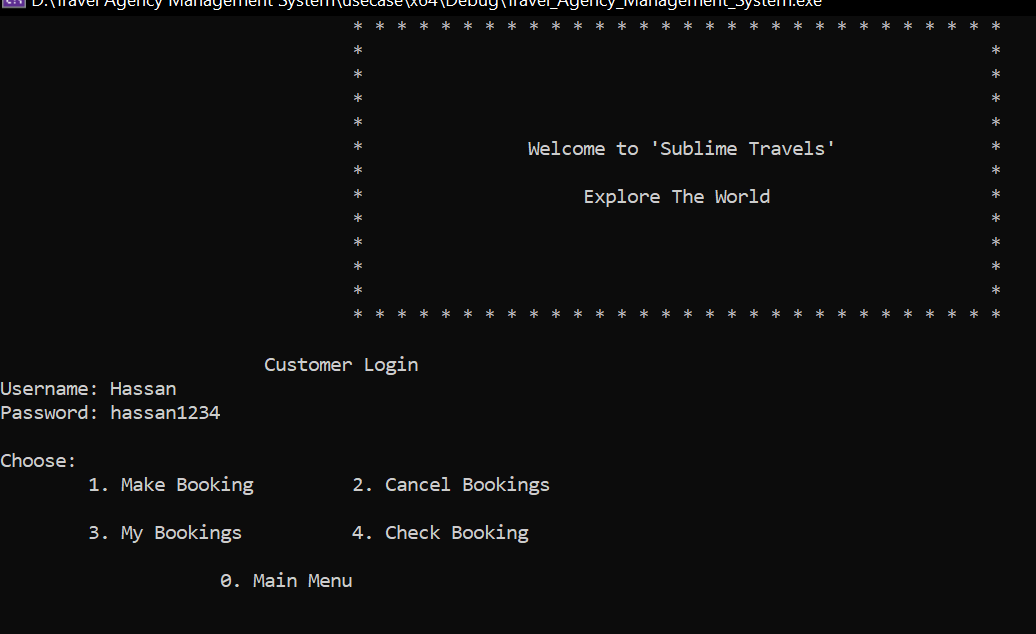
1. **Customer:**

* **Customer Sign-in:**

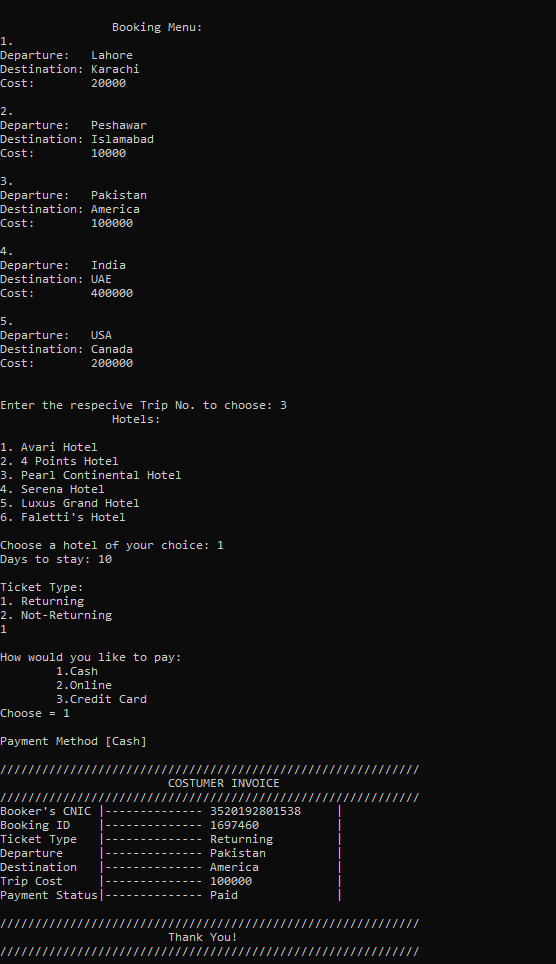
****

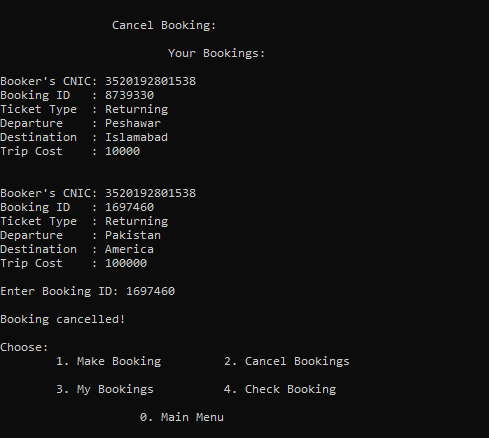
****

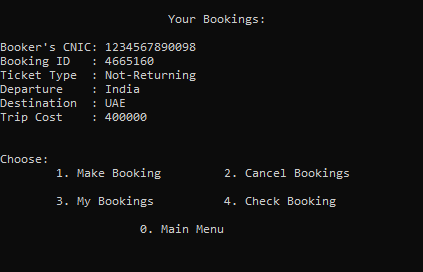
* **Customer Login:**

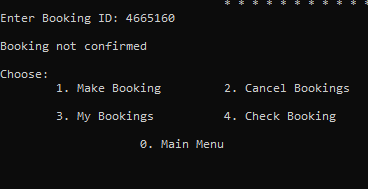
****

* **Customer Booking Options:**



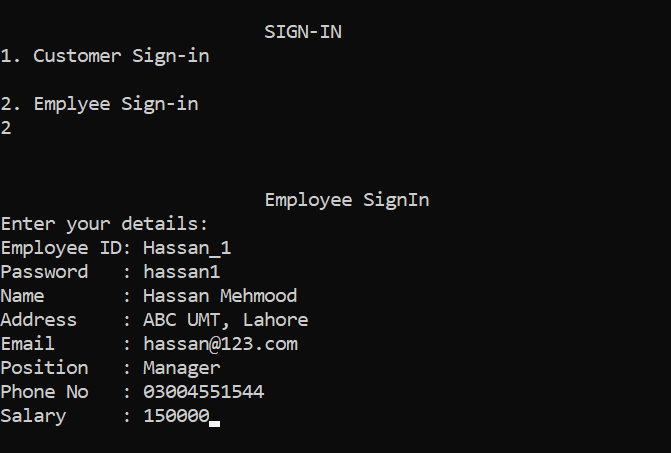


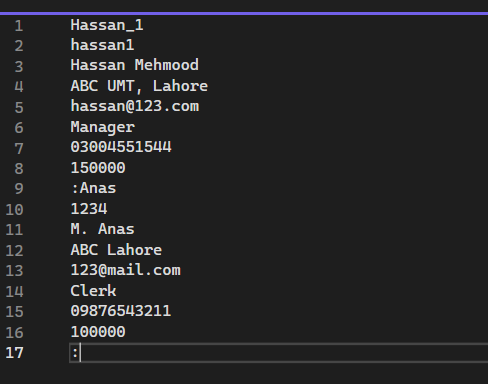




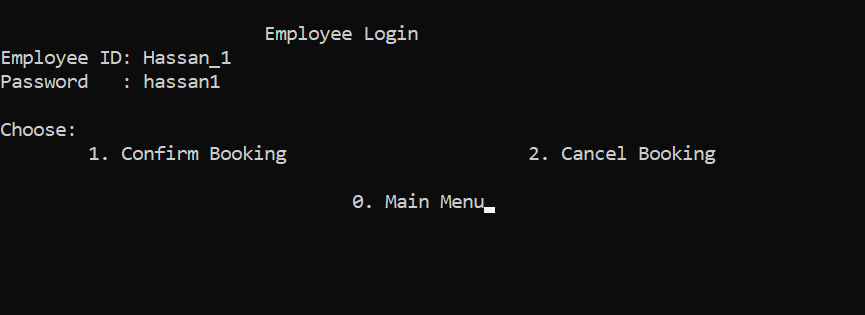
1. **Employee:**

* **Employee Sign-in:**

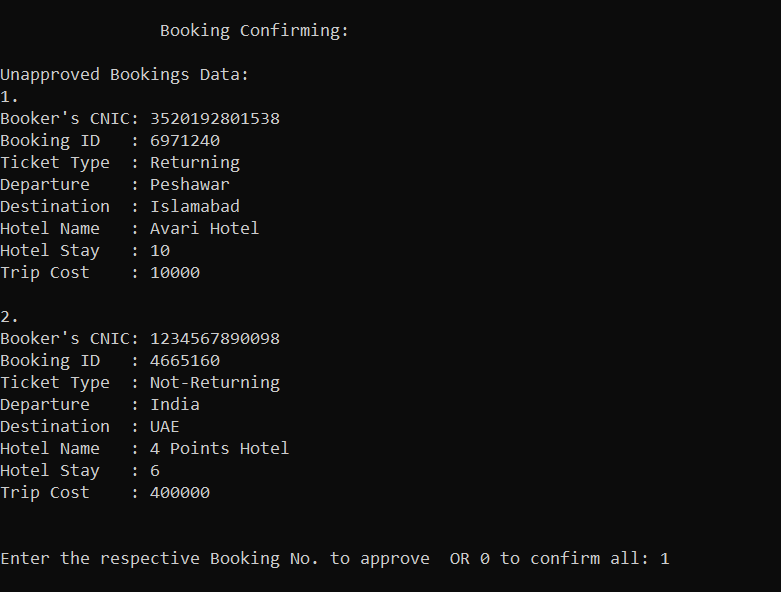
****

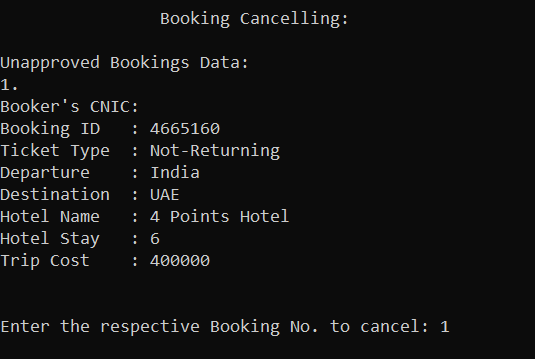
****

* **Employee Login:**

****

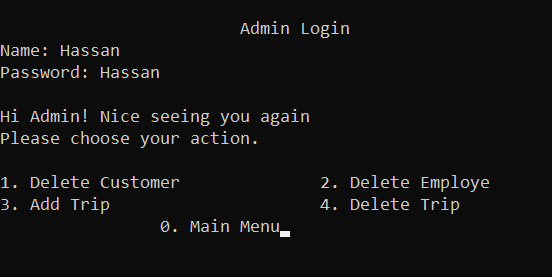
* **Employee Booking Options:**

****

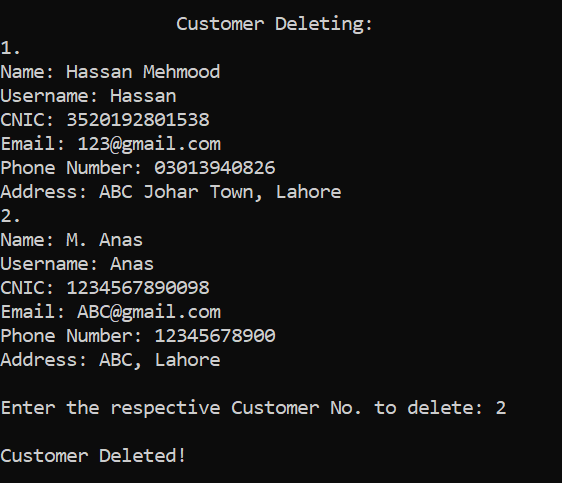
****

1. **Admin:**

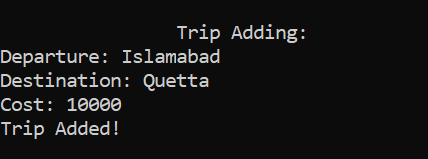
* **Admin Login:**

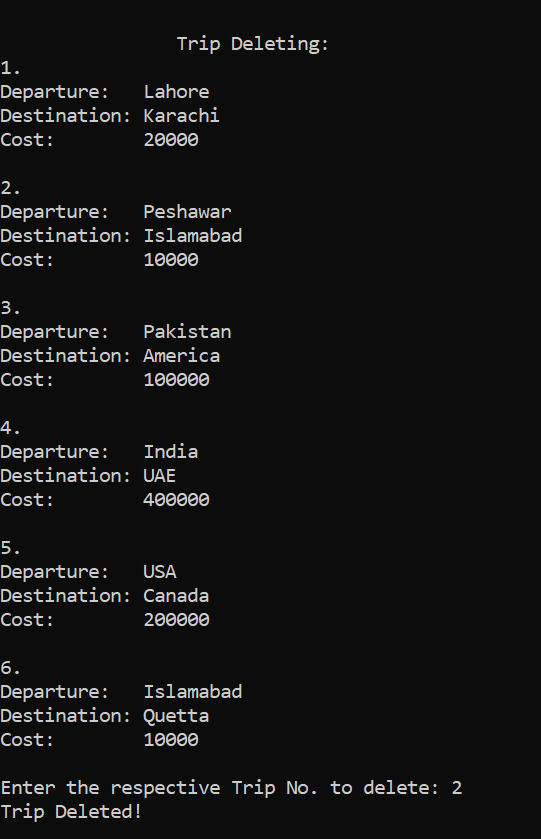
****

* **Admin Options:**

****

****

****

****