CSC 2000 Comprehensive Course Project

2024

Due: 24th October 2024- midnight

Instructions

Form groups of 10 people. Each group will submit one solution by email to my email. Also, attach a list of the group members.

EXERCISE

Create a Java application that simulates a basic airline reservation system, allowing users to search for flights and make reservations.

Requirements:

- **Loops:** Use loops for iterative tasks like searching for flights and processing reservations.
- **Conditional statements:** Use conditional statements to make decisions based on user input and flight availability.
- **Arrays:** Store flight information, seat availability, and reservation details in arrays.
- **Database (MySQL):** Use MySQL to store flight data, reservation information, and user details.

Steps:

1. **Create a database:**
    - Set up a MySQL database and create tables for flights, reservations, and users.
2. **Define Java classes:**
    - **Flight:** Represents a flight with attributes like flight number, origin, destination, departure time, arrival time, and available seats.
    - **Reservation:** Represents a reservation with attributes like reservation ID, flight, passenger name, and seat number.
    - **User:** Represents a user with attributes like user ID, name, and contact information.
3. **Implement methods:**
    - Create methods for searching flights, reserving seats, and viewing reservations.

Skeleton Code:

```
import java.sql.*;
import java.util.*;

public class AirlineReservationSystem {
    // ... (database connection, flight data, reservation data, user data)

    public static void main(String[] args) {
        // User interface loop
        while (true) {
            // Display menu options
            System.out.println("1. Search for flights");
            System.out.println("2. Make a reservation");
            System.out.println("3. View reservation details");
            System.out.println("4. Exit");

            // Get user input
            int choice = Integer.parseInt(scanner.nextLine());

            switch (choice) {
                case 1:
                    // Search for flights
                    break;
                case 2:
                    // Make a reservation
                    break;
                case 3:
                    // View reservation details
                    break;
                case 4:
                    System.out.println("Exiting...");
                    System.exit(0);
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }

    // ... (methods for searching flights, reserving seats, viewing
reservations)
}
```

Additional Considerations:

- **Error handling:** Implement error handling to catch exceptions and provide informative messages to the user.
- **Scalability:** Consider using a connection pool to manage database connections efficiently.
- **Security:** Implement security measures to protect user data and prevent unauthorized access.
- **Testing:** Thoroughly test the application to ensure it functions correctly and handles different scenarios.