HotelEase - Command Line Booking System

Admin Side:

1. **Add Hotels**:
    - Allow the admin to insert new hotels into the database. This can include details like hotel name, address, number of rooms, and basic amenities.

   JDBC Operations: `INSERT` operation on the `hotels` table.

2. **Modify Room Availability for a Specific Hotel**:
    - Admin can update the number of available rooms or mark certain rooms as unavailable/available for a specific hotel.

   JDBC Operations: `UPDATE` operation on the `rooms` table for the specified hotel.

3. **View Payment Records for a Specific Hotel**:
    - Admin can query and view all payment transactions for a particular hotel, including user details and payment amounts.

   JDBC Operations: `SELECT` operation on the `payments` table with a `JOIN` to link bookings and users to the respective hotel.

4. **View Booking Records for a Specific Hotel**:
    - Admin can query and view all booking transactions for a particular hotel, including user details and booking dates.

   JDBC Operations: `SELECT` operation on the `bookings` table with a `JOIN` to link users and hotel details.

5. **View Booking Records for a Specific User**:
    - Admin can query and view all booking transactions made by a specific user, including hotel details and booking dates.

   JDBC Operations: `SELECT` operation on the `bookings` table with a `JOIN` to link hotels and user details.

6. **More Coming Soon**:
    - As this will grow, you can add features like managing hotel reviews, discounts, etc., later. However, starting small with hotel management and room availability is a good first step.

User Side:

1. **Account Creation with Authentication System**:
    - Users can create an account (register) with username, password, email, and other basic details. They will also need to log in to make bookings.

JDBC Operations:

- `INSERT` for registration into the `users` table.
- `SELECT` for login and authentication (comparing input credentials).

2. **Book Rooms**:

- Users can search for available rooms based on the hotel, dates, etc., and make bookings.
- Functionality includes:
  - **Book**: User selects a room and books it for specific dates.
  - **Cancel Booking**: Users can cancel their room bookings.
  - **Reschedule Booking**: Users can change the dates of an existing booking.

JDBC Operations:

- `SELECT` to find available rooms based on input criteria.
- `INSERT` into the `bookings` table for new bookings.
- `DELETE` from the `bookings` table for cancellations.
- `UPDATE` on the `bookings` table to reschedule.

3. **View Booking History**:

- Users can view their past and upcoming bookings.

JDBC Operations: `SELECT` operation to retrieve the user's booking records from the `bookings` table.

4. **Change Profile Information**:

- Users can update their personal details, such as name, email, or password.

JDBC Operations: `UPDATE` operation on the `users` table to modify user data.

5. **View Payment Status**:

- Users can see whether their payments for room bookings are pending, completed, or failed.

JDBC Operations: `SELECT` to retrieve payment status for the user's bookings from the `payments` table.

6. **More Coming Soon**:

- This could expand into features like user reviews for hotels, loyalty points, etc., but for now, keep it simple with the essential booking and account features.

Database Structure:

Tables:

1. **users**

- `id` (PK), `username`, `password`, `email`, `phone_number`

2. **hotels**

- `id` (PK), `name`, `location`, `description`

3. **rooms**
   - `id` (PK), `hotel_id` (FK), `room_type`, `availability`, `price_per_night`
4. **bookings**
   - `id` (PK), `user_id` (FK), `hotel_id` (FK), `room_id` (FK), `start_date`, `end_date`, `booking_status`
5. **payments**
   - `id` (PK), `user_id` (FK), `booking_id` (FK), `payment_status`, `amount`, `transaction_date`

Steps to Implement:

1. **Database Setup**:
   - Create the MySQL/PostgreSQL database and tables as defined above.
2. **JDBC Configuration**:
   - Use JDBC to connect to the database.
   - Create utility methods for establishing connections, performing SQL queries, etc.
3. **Command-line Menu**:
   - Build a CLI-based menu system with options for admin and user interactions.
   - Prompt for login (for users/admin) and present different options based on their role.
4. **Admin Features**:
   - Implement functions for adding hotels, modifying room availability, and viewing payment records using SQL queries with JDBC.
5. **User Features**:
   - Implement account creation and login with password authentication.
   - Add the room booking, cancellation, and rescheduling features.
   - Include viewing booking history and payment status functionalities.

JDBC Workflow:

- **Connection to the Database**:
  - Use `DriverManager.getConnection` to establish a connection with your database.
- **CRUD Operations**:
  - Use `PreparedStatement` for all `INSERT`, `UPDATE`, and `DELETE` operations.
  - Use `ResultSet` and `PreparedStatement` for `SELECT` operations to retrieve data from the database.

Sample Flow (for Booking):

1. **User selects "Book a Room"**.
2. **JDBC**: A `SELECT` query to fetch available rooms from the `rooms` table for the specified dates.
3. **JDBC**: Once the user selects a room, an `INSERT` query adds a new booking to the `bookings` table.
4. **JDBC**: An `INSERT` query adds a record in the `payments` table to reflect the booking's payment status.

Key Considerations:

- Error handling (e.g., invalid dates, booking overlaps, etc.).
- Security (e.g., password hashing for user credentials).
- Database connection pooling (optional but helpful for scalability).
