Project: Smart Hotel Booking System

1. Introduction

This document outlines the **Low-Level Design (LLD)** for a **Smart Hotel Booking System**, which allows users to search, book, and manage hotel reservations. The platform supports hotel owners in listing their properties, defining room availability, and handling payments.

This design supports both **Java (Spring Boot)** and **.NET (ASP.NET Core)** frameworks for backend development.

2. Module Overview

2.1 User & Role Management

- Role-based access control (**Admin, Hotel Manager, Guest**).
- Secure **user authentication and profile management**.

2.2 Hotel & Room Management

- Hotel managers can **list hotels and manage rooms**.
- Set **room types, pricing, amenities, and availability**.

2.3 Booking & Payment Processing

- Users can **search, book, and pay for rooms** securely.
- Integration with **payment gateways** for transactions.

2.4 Reviews & Ratings

- Guests can **rate hotels and leave reviews**.
- Hotels can **respond to feedback**.

2.5 Loyalty & Rewards Program

- Guests earn **reward points for bookings**.
- **Redeem points for discounts on future stays**.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Angular or React
- **Backend**: REST API-based architecture
- **Database**: Relational Database (MySQL/SQL Server)

3.2 Component Interaction

- **Frontend communicates with the backend via REST APIs**.
- **Backend manages hotel data, booking operations, and payments**.

4. Module-Wise Design

4.1 User & Role Management Module

4.1.1 Features

- **User authentication using JWT tokens**.
- **Role-based permissions**: Admin, Hotel Manager, Guest.

4.1.2 Data Flow

1. Users **register and log in**.
2. Role-based **permissions are assigned**.
3. **Admins manage hotel manager accounts**.

4.1.3 Entities

- **User** (UserID, Name, Email, Password, Role, ContactNumber)

4.2 Hotel & Room Management Module

4.2.1 Features

- Hotel managers can **list and manage hotels**.
- Define **room types, pricing, and amenities**.

4.2.2 Data Flow

1. Hotel managers **add hotel details and room inventory**.
2. Users **search for available rooms**.
3. Availability updates when **bookings are confirmed**.

4.2.3 Entities

- **Hotel** (HotelID, Name, Location, ManagerID, Amenities, Rating)
- **Room** (RoomID, HotelID, Type, Price, Availability, Features)

4.3 Booking & Payment Processing Module

4.3.1 Features

- Users can **search and book rooms**.
- Secure **payment processing and booking confirmation**.

4.3.2 Data Flow

1. Users **select a hotel and book a room**.
2. Payments are **processed securely**.
3. Booking details are **sent to users and hotel managers**.

4.3.3 Entities

- **Booking** (BookingID, UserID, RoomID, CheckInDate, CheckOutDate, Status, PaymentID)
- **Payment** (PaymentID, UserID, BookingID, Amount, Status, PaymentMethod)

4.4 Reviews & Ratings Module

4.4.1 Features

- Guests can **rate hotels and write reviews**.
- Hotels can **respond to reviews**.

4.4.2 Data Flow

1. Users **submit reviews after their stay**.
2. The system **moderates and publishes reviews**.
3. Hotel managers can **respond to feedback**.

4.4.3 Entities

- **Review** (ReviewID, UserID, HotelID, Rating, Comment, Timestamp)

4.5 Loyalty & Rewards Program Module

4.5.1 Features

- Guests **earn points for bookings**.

- Points can be **redeemed for discounts**.

4.5.2 Data Flow

1. When a user **books a hotel**, reward points are **added to their account**.
2. Users can **view their accumulated points**.
3. At checkout, users can **redeem points for discounts**.

4.5.3 Entities

- **LoyaltyAccount** (LoyaltyID, UserID, PointsBalance, LastUpdated)
- **Redemption** (RedemptionID, UserID, BookingID, PointsUsed, DiscountAmount)

5. Deployment Strategy

5.1 Local Deployment

- **Frontend Deployment**: Angular/React dev server.
- **Backend Deployment**: Spring Boot/ASP.NET Core locally.
- **Database**: MySQL/PostgreSQL/SQL Server.

6. Database Design

6.1 Tables and Relationships

- **User** (UserID, Name, Email, Password, Role, ContactNumber)
- **Hotel** (HotelID, Name, Location, ManagerID, Amenities, Rating)
- **Room** (RoomID, HotelID, Type, Price, Availability, Features)
- **Booking** (BookingID, UserID, RoomID, CheckInDate, CheckOutDate, Status, PaymentID)
- **Payment** (PaymentID, UserID, BookingID, Amount, Status, PaymentMethod)
- **Review** (ReviewID, UserID, HotelID, Rating, Comment, Timestamp)
- **LoyaltyAccount** (LoyaltyID, UserID, PointsBalance, LastUpdated)
- **Redemption** (RedemptionID, UserID, BookingID, PointsUsed, DiscountAmount)

7. User Interface Design

7.1 Wireframes

- **Guest Dashboard**: Search and book rooms.
- **Hotel Manager Dashboard**: List and manage hotel properties.
- **Admin Panel**: Manage hotels, users, and reviews.
- **Loyalty Program Page**: View and redeem points.

8. Non-Functional Requirements

8.1 Performance

- **Handles high-traffic room searches efficiently**.

8.2 Scalability

- **Supports multiple hotel chains and properties**.

8.3 Security

- **Secure JWT-based authentication**.
- **Encrypted payment transactions**.

8.4 Usability

- **Mobile-friendly interface for seamless booking experience**.

9. Assumptions and Constraints

9.1 Assumptions

- Users can **cancel bookings up to 24 hours before check-in**.
- Hotels must **verify their identity before listing**.

9.2 Constraints

- The system must **support different currencies for payments**.
- Hotels must **comply with legal policies for cancellations and refunds**.