Project: Online Travel & Hospitality Booking System

1. Introduction

This document outlines the **Low-Level Design (LLD)** for an **Online Travel & Hospitality Booking System**, which allows users to book hotels, flights, and holiday packages while providing hotel managers and travel agents with a platform to manage their services.

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, Traveler, Hotel Manager, Travel Agent**).
- Secure **user authentication and profile management**.

2.2 Hotel & Flight Booking

- Search and **book hotels and flights**.
- View **availability, pricing, and customer reviews**.

2.3 Package & Itinerary Management

- Travel agents can **create and sell vacation packages**.
- Customizable itineraries for travelers.

2.4 Payment & Billing

- Secure **online payments, invoices, and refunds**.
- Integration with **payment gateways (Stripe, PayPal, Razorpay)**.

2.5 Review & Customer Support

- Travelers can **leave reviews for hotels and services**.
- Integrated **support system for complaints and queries**.

3. Architecture Overview

3.1 Architectural Style

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

3.2 Component Interaction

- **Frontend communicates with the backend via REST APIs**.
- **Backend handles authentication, booking logic, and data persistence**.

4. Module-Wise Design

4.1 User & Role Management Module

4.1.1 Features

- **User authentication** using JWT tokens.
- Role-based access control: **Admin, Traveler, Hotel Manager, Travel Agent**.

4.1.2 Data Flow

1. Users **register and log in** to the system.
2. Role-based **permissions are assigned**.
3. **Admin manages user accounts** and permissions.

4.1.3 Entities

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

4.2 Hotel & Flight Booking Module

4.2.1 Features

- Users can **search for hotels and flights** based on location and dates.
- **Booking confirmation and e-ticket generation**.

4.2.2 Data Flow

1. Users **search for available hotels or flights**.
2. The system **fetches availability and pricing** from the database.
3. Travelers **confirm bookings and proceed with payment**.
4. The booking is **recorded and a confirmation email is sent**.

4.2.3 Entities

- **Hotel** (HotelID, Name, Location, RoomsAvailable, Rating, PricePerNight)
- **Flight** (FlightID, Airline, Departure, Arrival, Price, Availability)
- **Booking** (BookingID, UserID, Type[Hotel/Flight], Status, PaymentID)

4.3 Package & Itinerary Management Module

4.3.1 Features

- Travel agents can **create custom travel packages**.
- Travelers can **modify itineraries based on preferences**.

4.3.2 Data Flow

1. **Travel agents define packages** (hotels, flights, activities).
2. Travelers **browse and book packages**.
3. The system **generates a complete itinerary**.

4.3.3 Entities

- **Package** (PackageID, Name, IncludedHotels, IncludedFlights, Activities, Price)
- **Itinerary** (ItineraryID, UserID, PackageID, CustomizationDetails)

4.4 Payment & Billing Module

4.4.1 Features

- Secure **online payments and refunds**.
- **Multi-currency support** and tax calculation.

4.4.2 Data Flow

1. Travelers **proceed to checkout** after selecting a booking/package.
2. Payments are **processed via third-party payment gateways**.
3. Invoices and payment confirmation emails **are generated**.

4.4.3 Entities

- **Payment** (PaymentID, UserID, BookingID, Amount, Status, PaymentMethod)
- **Invoice** (InvoiceID, BookingID, UserID, TotalAmount, Timestamp)

4.5 Review & Customer Support Module

4.5.1 Features

- Users can **leave ratings and reviews** for hotels and services.
- In-app **customer support and complaint resolution**.

4.5.2 Data Flow

1. Travelers **submit reviews for booked services**.
2. The system **moderates and publishes reviews**.
3. Users can **raise support tickets**, and admins assign them to support agents.

4.5.3 Entities

- **Review** (ReviewID, UserID, HotelID, Rating, Comment, Timestamp)
- **SupportTicket** (TicketID, UserID, Issue, Status, AssignedAgent)

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, RoomsAvailable, Rating, PricePerNight)
- **Flight** (FlightID, Airline, Departure, Arrival, Price, Availability)
- **Booking** (BookingID, UserID, Type[Hotel/Flight], Status, PaymentID)
- **Package** (PackageID, Name, IncludedHotels, IncludedFlights, Activities, Price)
- **Itinerary** (ItineraryID, UserID, PackageID, CustomizationDetails)
- **Payment** (PaymentID, UserID, BookingID, Amount, Status, PaymentMethod)
- **Invoice** (InvoiceID, BookingID, UserID, TotalAmount, Timestamp)

- **Review (ReviewID, UserID, HotelID, Rating, Comment, Timestamp)**
- **SupportTicket (TicketID, UserID, Issue, Status, AssignedAgent)**

7. User Interface Design

7.1 Wireframes

- **Traveler Dashboard**: Search and book hotels/flights/packages.
- **Hotel Manager Dashboard**: Manage hotel rooms, pricing, availability.
- **Travel Agent Dashboard**: Create and sell travel packages.
- **Admin Panel**: Manage users, bookings, and customer queries.

8. Non-Functional Requirements

8.1 Performance

- **Concurrent support for 10,000+ users** without performance degradation.

8.2 Security

- **OAuth2 and JWT authentication** for user security.
- **PCI-DSS compliance** for payment transactions.

8.3 Usability

- **Mobile-friendly UI** with real-time booking updates.

9. Assumptions and Constraints

9.1 Assumptions

- Hotels must **update availability every 24 hours**.
- Users can **cancel bookings only within cancellation policy limits**.

9.2 Constraints

- The system must handle **booking transactions in real time**.
- Data retention for **user history must be maintained for 5 years**.