Project: Travel Package Booking System

1. Introduction

This document outlines the **Low-Level Design (LLD)** for a **Travel Package Booking System**, which enables users to explore, book, and manage travel packages, including flights, hotels, and activities. Travel agencies can list customizable packages with pricing, itinerary details, and availability.

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, Travel Agent, Customer**).
- Secure **user authentication and profile management**.

2.2 Travel Package Management

- Travel agents can **create and manage tour packages**.
- Packages include **flights, hotels, sightseeing, and pricing details**.

2.3 Booking & Payment Processing

- Customers can **search, book, and pay for travel packages**.
- Integration with **payment gateways** for transactions.

2.4 Reviews & Ratings

- Customers can **rate travel packages and provide feedback**.
- Travel agents can **respond to reviews**.

2.5 Travel Insurance & Assistance

- Customers can **opt for travel insurance during booking**.
- 24/7 **travel assistance and emergency support**.

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 manages package listings, 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, Travel Agent, Customer.

4.1.2 Data Flow

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

4.1.3 Entities

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

4.2 Travel Package Management Module

4.2.1 Features

- Travel agents can **create and manage travel packages**.
- Define **package details, itinerary, and pricing**.

4.2.2 Data Flow

1. Travel agents **add travel package details**.
2. Customers **search for available packages**.
3. Availability updates when **bookings are confirmed**.

4.2.3 Entities

- **TravelPackage** (PackageID, Title, Description, Duration, Price, IncludedServices)

4.3 Booking & Payment Processing Module

4.3.1 Features

- Customers can **search and book travel packages**.
- Secure **payment processing and booking confirmation**.

4.3.2 Data Flow

1. Customers **select a package and book it**.
2. Payments are **processed securely**.
3. Booking details are **sent to customers and travel agents**.

4.3.3 Entities

- **Booking** (BookingID, UserID, PackageID, StartDate, EndDate, Status, PaymentID)
- **Payment** (PaymentID, UserID, BookingID, Amount, Status, PaymentMethod)

4.4 Reviews & Ratings Module

4.4.1 Features

- Customers can **rate and review travel packages**.
- Travel agents can **respond to feedback**.

4.4.2 Data Flow

1. Customers **submit reviews after their trip**.
2. The system **moderates and publishes reviews**.
3. Travel agents can **respond to feedback**.

4.4.3 Entities

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

4.5 Travel Insurance & Assistance Module

4.5.1 Features

- Customers can **purchase travel insurance during booking**.
- Access to **24/7 travel assistance for emergencies**.

4.5.2 Data Flow

1. Users **choose insurance options at checkout**.
2. Insurance details are **linked to the booking**.
3. Customers can **contact emergency support if needed**.

4.5.3 Entities

- **Insurance** (InsuranceID, UserID, BookingID, CoverageDetails, Provider, Status)
- **AssistanceRequest** (RequestID, UserID, IssueDescription, Status, ResolutionTime)

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)
- **TravelPackage** (PackageID, Title, Description, Duration, Price, IncludedServices)
- **Booking** (BookingID, UserID, PackageID, StartDate, EndDate, Status, PaymentID)
- **Payment** (PaymentID, UserID, BookingID, Amount, Status, PaymentMethod)
- **Review** (ReviewID, UserID, PackageID, Rating, Comment, Timestamp)
- **Insurance** (InsuranceID, UserID, BookingID, CoverageDetails, Provider, Status)
- **AssistanceRequest** (RequestID, UserID, IssueDescription, Status, ResolutionTime)

7. User Interface Design

7.1 Wireframes

- **Customer Dashboard**: Browse and book travel packages.
- **Travel Agent Dashboard**: List and manage travel packages.

- **Admin Panel**: Manage users, agents, and bookings.
- **Insurance & Assistance Page**: Purchase insurance and request help.

8. Non-Functional Requirements

8.1 Performance

- **Efficient package search and booking handling**.

8.2 Scalability

- **Supports global travel agencies and diverse packages**.

8.3 Security

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

8.4 Usability

- **User-friendly UI with seamless navigation**.

9. Assumptions and Constraints

9.1 Assumptions

- Customers can **cancel bookings up to 7 days before departure**.
- Travel agents must **verify their identity before listing packages**.

9.2 Constraints

- The system must **support multi-currency payments**.
- Travel agencies must **comply with local tourism regulations**.