Project: Real Estate Property Listing System

1. Introduction

This document outlines the **Low-Level Design (LLD)** for a **Real Estate Property Listing System** that allows users to list, search, and purchase or rent properties. It supports different user roles such as **buyers, sellers, and agents** and provides an intuitive platform for property transactions.

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

2. Module Overview

2.1 User Management

- Authentication and role-based access control (**Buyer, Seller, Agent, Admin**).
- User profile creation and management.

2.2 Property Listing & Management

- Sellers and agents can list properties with images, details, and pricing.
- Buyers can browse and filter property listings.

2.3 Search & Recommendation System

- Advanced property search based on location, price, type, and amenities.
- AI-driven property recommendations based on user preferences.

2.4 Booking & Inquiry Management

- Buyers can schedule property visits.
- Sellers and agents receive inquiries and manage appointments.

2.5 Payment & Transaction Processing

- Secure payment processing for property bookings or purchases.
- Integration with third-party payment gateways.

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

- The frontend communicates with the backend via REST APIs.
- The backend handles user authentication, business logic, and data storage.
- Users access the system through a responsive web interface.

4. Module-Wise Design

4.1 User Management Module

4.1.1 Features

- **User authentication** (Registration, Login, JWT-based Authentication).
- **Role management** (Buyer, Seller, Agent, Admin).
- **Profile management** (Update personal and business details).

4.1.2 Data Flow

1. Users register through the frontend.
2. The backend validates user details and assigns roles.
3. User data is stored in the database.
4. Upon login, a JWT token is generated for authentication.

4.1.3 Entities

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

4.2 Property Listing & Management Module

4.2.1 Features

- Sellers and agents can **add, update, and remove** property listings.
- Listings include images, price, location, size, and amenities.
- Buyers can view and save property listings.

4.2.2 Data Flow

1. A seller or agent submits a property listing.
2. The backend stores the property details.
3. Buyers browse properties using various filters.
4. Property details are retrieved and displayed.

4.2.3 Entities

- **Property** (PropertyID, SellerID, Title, Location, Price, Type, Size, Bedrooms, Bathrooms, Status)
- **Images** (ImageID, PropertyID, URL)

4.3 Search & Recommendation System Module

4.3.1 Features

- Buyers can search properties by **location, price, type, and amenities**.
- AI-based recommendation system suggests properties based on user interactions.

4.3.2 Data Flow

1. Buyers input search criteria via the frontend.
2. The backend fetches matching properties from the database.
3. AI-based recommendations are generated based on previous searches.
4. The system returns the search results.

4.3.3 Entities

- **SearchHistory** (SearchID, UserID, Filters, Timestamp)
- **Recommendations** (RecID, UserID, PropertyID, Score)

4.4 Booking & Inquiry Management Module

4.4.1 Features

- Buyers can **schedule property visits** with sellers or agents.
- Sellers and agents receive and manage property inquiries.

4.4.2 Data Flow

1. Buyers request a **property visit** via the frontend.
2. The backend validates the request and notifies the seller/agent.

3. Sellers/agents accept or decline the request.
4. If confirmed, a **meeting is scheduled**.

4.4.3 Entities

- **Booking** (BookingID, BuyerID, PropertyID, SellerID, Date, Status)
- **Inquiry** (InquiryID, UserID, PropertyID, Message, Status)

4.5 Payment & Transaction Processing Module

4.5.1 Features

- Buyers can make **secure payments** for bookings or purchases.
- Integration with **third-party payment gateways**.

4.5.2 Data Flow

1. A buyer initiates a **payment transaction**.
2. The backend processes the payment via a **payment gateway**.
3. Upon successful payment, the **transaction is recorded**.
4. The seller receives a confirmation notification.

4.5.3 Entities

- **Transaction** (TransactionID, BuyerID, PropertyID, Amount, PaymentStatus, PaymentDate)

5. Deployment Strategy

5.1 Local Deployment

- **Frontend Deployment**: Runs on a local development server (ng serve for Angular, local React server).
- **Backend Deployment**: Spring Boot/ASP.NET Core application running on a local server.
- **Database**: MySQL/PostgreSQL/SQL Server set up for development.

6. Database Design

6.1 Tables and Relationships

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

- **Property** (PropertyID, SellerID, Title, Location, Price, Type, Size, Bedrooms, Bathrooms, Status)
- **Images** (ImageID, PropertyID, URL)
- **SearchHistory** (SearchID, UserID, Filters, Timestamp)
- **Recommendations** (RecID, UserID, PropertyID, Score)
- **Booking** (BookingID, BuyerID, PropertyID, SellerID, Date, Status)
- **Inquiry** (InquiryID, UserID, PropertyID, Message, Status)
- **Transaction** (TransactionID, BuyerID, PropertyID, Amount, PaymentStatus, PaymentDate)

7. User Interface Design

7.1 Wireframes

- **Buyer Dashboard**: Browse properties, search, book visits.
- **Seller Dashboard**: Manage property listings, view inquiries.
- **Agent Dashboard**: Assist buyers, schedule viewings.
- **Admin Dashboard**: Manage users, properties, and transactions.
- **Property Page**: Displays images, price, location, and inquiry options.

8. Non-Functional Requirements

8.1 Performance

- The system should handle **10,000 concurrent users**.

8.2 Security

- **JWT-based authentication** and **role-based access control**.
- Secure **payment processing** using encryption.

8.3 Usability

- **Mobile-responsive UI** with real-time property availability updates.

9. Assumptions and Constraints

9.1 Assumptions

- Users must create an account to list or book a property.
- Property visits can be scheduled **up to 30 days in advance**.

9.2 Constraints

- Payments are processed via **third-party gateways**.
- **No physical contract handling** (only online transactions).