This document outlines how the **Clean Architecture** principle is applied in the House Broker Application. The goal is to ensure **separation of concerns**, **testability**, and **maintainability** of the application through a layered architecture approach.

**1. Domain Layer (HouseBrokerApp.Domain)**

**Purpose**:  
The core of the application that contains all the business models, logic, and rules.

**Responsibilities**:

* Define entities like PropertyListing, CommissionRate, ApplicationUser
* Define enums like PropertyType
* No dependency on other layers

### 2. ****Application Layer**** (HouseBrokerApp.Application)

**Purpose**:  
Contains business use cases, service contracts, DTOs, and validation logic.

**Responsibilities**:

* Service interfaces like IPropertyListingService
* DTOs such as CreatePropertyListingRequest, PropertyListingResponse
* FluentValidation validators
* AutoMapper profiles

**Benefits**:

* Highly testable
* No dependency on infrastructure or UI

### 3. ****Infrastructure Layer**** (HouseBrokerApp.Infrastructure)

**Purpose**:  
Implements services like database access, JWT token generation, and external integrations.

**Responsibilities**:

* HouseBrokerAppDbContext using Entity Framework Core
* Implements services like JwtTokenService
* Implements repositories such as GenericRepository<T>
* Uploads images and reads configurations

**4. API Layer (HouseBrokerApp.API)**

**Purpose**:  
Handles incoming HTTP requests and routes them to application services.

**Responsibilities**:

* ASP.NET Core Controllers (e.g., PropertyController, AuthController)
* Middleware for global exception handling
* Dependency injection and authentication setup

**Security**:

* JWT Authentication
* Role-based Authorization using [Authorize(Roles = "...")]

**5. Presentation Layer (HouseBrokerApp.UI - Blazor Server)**

**Purpose**:  
The user interface layer where brokers and house seekers interact with the application.

**Responsibilities**:

* User login, registration, and dashboard
* Property creation, editing, and search
* Uses HttpClient to consume the API
* Auth state management using CustomAuthStateProvider

[WebClient]

↓

[API]

↓

[Application] ↔ [Contracts]

↓

[Domain]

↑

[Infrastructure]

// 🧱 Folder Structure Based on Clean Architecture (with Domain Separation + Frontend + Contracts Layer)

HouseBrokerApp/

│

├── src/

│ ├── HouseBrokerApp.API/ // Presentation Layer (Backend API)

│ │ ├── Controllers/ // Only Controllers and Filters/Middleware

│ │ ├── Middleware/

│ │ ├── DependencyInjection.cs

│ │ └── Program.cs / appsettings.json

│ │

│ ├── HouseBrokerApp.Application/ // Application Layer

│ │ ├── Interfaces/ // Interfaces for Services & UseCases

│ │ ├── Services/ // Business logic

│ │ └── Validators/ // FluentValidation classes

│ │

│ ├── HouseBrokerApp.Contracts/ // 🔹 New DTO Layer (Shared Data Contracts)

│ │ ├── Requests/ // DTOs for Create/Update

│ │ ├── Responses/ // DTOs for Fetching/Returning

│ │ └── Shared/ // Common DTOs (e.g., PagedResults)

│ │

│ ├── HouseBrokerApp.Domain/ // Domain Layer

│ │ ├── Entities/ // Core domain models (PropertyListing, User)

│ │ ├── ValueObjects/ // Optional domain-specific types (e.g., Money, Address)

│ │ ├── Enums/ // Role enums, status enums, etc.

│ │ └── Interfaces/ // Domain abstractions (e.g., IRepository)

│ │

│ ├── HouseBrokerApp.Infrastructure/ // Infrastructure Layer

│ │ ├── Persistence/ // EF Core DbContext, migrations

│ │ │ ├── Configurations/ // Fluent API configs

│ │ │ └── Migrations/

│ │ ├── Identity/ // Identity system setup

│ │ ├── Services/ // Email, Cache, CommissionService etc.

│ │ └── Repositories/ // Concrete implementations

│

│ ├── HouseBrokerApp.WebClient/ // Frontend Layer (Blazor or React/Angular)

│ │ ├── Pages/ // Main UI Pages

│ │ ├── Components/ // Reusable UI Components

│ │ ├── Services/ // API integration (via HttpClient or Axios)

│ │ ├── Models/ // ViewModels for the frontend

│ │ ├── App.{razor|tsx} // App startup component

│ │ └── wwwroot/ or public/ // Static files (css, images)

│

├── tests/

│ ├── HouseBrokerApp.Application.Tests/ // Unit Tests (Service logic)

│

├── README.md

└── HouseBrokerApp.sln

**Final Dependency Rules Summary**

| **Layer** | **Depends On** |
| --- | --- |
| 🧱 Domain | Nothing |
| 🧠 Application | Domain, Contracts |
| 📦 Contracts | Domain (optional) |
| 🛠 Infrastructure | Domain, Application |
| 🌐 API | Application, Contracts, Domain |
| 🎨 UI (Blazor) | Contracts, Domain (via HTTP calls) |