**Project Structure Documentation**

1. Ecommerce.API (Presentation Layer)

The entry point of the application. Handles HTTP requests and returns responses.

1. Controllers/ – Web API controllers (e.g., ProductsController, OrdersController). Should only delegate work to the Application layer.
2. App\_Start/ – Configuration files (e.g., UnityConfig.cs for DI, route settings).
3. DTOs/ – Request/response models that shape how data is exposed over the API. Prevents exposing domain entities directly.
4. Filters/ – Custom filters (e.g., authentication, exception handling, logging).
5. Modules/ – Optional feature modules (e.g., a HealthCheck module).
6. Utils/ – API-specific helpers (e.g., response formatters, JWT utilities).
7. Ecommerce.Application (Application Layer)

Contains business logic and coordinates between API, Domain, and Infrastructure.

1. Interfaces/ – Contracts for services (e.g., IProductService).
2. Services/ – Implementations of business logic (e.g., ProductService). Should use repositories from Infrastructure.
3. DTOs/ – Data Transfer Objects used inside the Application layer (not API DTOs).
4. Validators/ – Input validators (FluentValidation or custom validation rules).
5. Helpers/ – Business-related helper methods (e.g., price calculators, discounts).
6. Mappings/ – AutoMapper profiles or custom mapping logic between entities and DTOs.
7. Ecommerce.Domain (Core Layer)

The heart of the system. Contains pure business rules with no external dependencies.

1. Entities/ – Core business entities (e.g., Product, User, Order).
2. Enums/ – Enumerations (e.g., OrderStatus, UserRole).
3. ValueObjects/ – Immutable objects representing concepts (e.g., Money, Address).
4. Exceptions/ – Domain-specific exceptions (e.g., InsufficientStockException).
5. Interfaces/ – Repository contracts or domain services (e.g., IProductRepository).

The Domain layer should not depend on any other project.

1. Ecommerce.Infrastructure (Infrastructure Layer)

Implements persistence and integrations (databases, external APIs, payment providers).

1. Persistence/ – Database context (e.g., EcommerceDbContext) and EF configurations.
2. Repositories/ – Implementations of repository interfaces from Domain (e.g., ProductRepository).
3. Migrations/ – Database migration files (if using Entity Framework).
4. ExternalServices/ – Integrations with external systems (e.g., Stripe, PayPal, email services).
5. Helpers/ – Infrastructure-specific utilities (e.g., logging, caching).
6. Configurations/ – Configuration helpers (connection strings, options classes).
7. Ecommerce.Tests (Testing Layer)

Contains unit tests and integration tests.

1. API/ – Tests for controllers and endpoints (can use integration testing).
2. Application/ – Tests for services and use cases.
3. Domain/ – Tests for business rules, entities, and value objects.
4. Infrastructure/ – Tests for repositories and DB operations (integration).