به نام خدا

خب، با توجه به تمام بحث‌ها و پیاده‌سازی‌های قبلی، الان \*\*ساختار کامل، نهایی، حرفه‌ای و آمادهٔ تولید\*\* برای پوشه `AuthSystem.Application` رو برات می‌نویسم.

این ساختار بر اساس بهترین شیوه‌های معماری مدرن در سال ۲۰۲۵ طراحی شده:

\*\*Vertical Slice Architecture\*\*, \*\*CQRS\*\*, \*\*MediatR\*\*, \*\*Clean Architecture\*\*, و \*\*Dependency Inversion\*\*.

---

## 📁 ساختار کامل پوشه `AuthSystem.Application`

```

AuthSystem.Application/

├── AssemblyMarker.cs

├── AuthSystem.Application.csproj

├── Common/

│ ├── Abstractions/

│ │ ├── Messaging/

│ │ │ ├── IEmailSender.cs

│ │ │ ├── ISmsSender.cs

│ │ │ └── INotificationService.cs

│ │ ├── Persistence/

│ │ │ ├── IApplicationDbContext.cs

│ │ │ └── IUnitOfWork.cs

│ │ ├── Security/

│ │ │ ├── IPasswordHasher.cs

│ │ │ ├── ITokenService.cs

│ │ │ ├── ICurrentUserService.cs

│ │ │ └── IPermissionService.cs

│ │ ├── Storage/

│ │ │ ├── IFileStorageService.cs

│ │ │ └── IImageProcessingService.cs

│ │ ├── Time/

│ │ │ └── IDateTimeProvider.cs

│ │ └── ExternalServices/

│ │ ├── IExternalAuthProvider.cs

│ │ └── IExternalApiClient.cs

│ │

│ ├── Primitives/

│ │ ├── Error.cs

│ │ ├── Errors.cs

│ │ ├── Result.cs

│ │ ├── ResultOfT.cs

│ │ ├── ITransactionalCommand.cs

│ │ └── ResultExtensions.cs

│ │

│ ├── Exceptions/

│ │ ├── ApplicationException.cs

│ │ ├── DomainException.cs

│ │ ├── InfrastructureException.cs

│ │ ├── ValidationException.cs

│ │ ├── NotFoundException.cs

│ │ ├── ConflictException.cs

│ │ ├── BadRequestException.cs

│ │ ├── UnauthorizedException.cs

│ │ ├── ForbiddenException.cs

│ │ ├── ConcurrencyException.cs

│ │ └── ExternalServiceException.cs

│ │

│ ├── Behaviors/

│ │ ├── UnhandledExceptionBehavior.cs

│ │ ├── LoggingBehavior.cs

│ │ ├── ValidationBehavior.cs

│ │ └── TransactionBehavior.cs

│ │

│ ├── Mappings/

│ │ ├── AutoMapperProfiles.cs

│ │ └── UserMappingProfile.cs

│ │

│ ├── Services/

│ │ ├── IUserManager.cs

│ │ ├── UserManager.cs

│ │ └── IEmailTemplateRenderer.cs

│ │

│ └── Extensions/

│ ├── ServiceCollectionExtensions.cs

│ └── ResultToHttpResponseExtensions.cs

│

├── Features/

│ ├── Users/

│ │ ├── RegisterUser/

│ │ │ ├── RegisterUserCommand.cs

│ │ │ ├── RegisterUserCommandHandler.cs

│ │ │ ├── RegisterUserCommandValidator.cs

│ │ │ └── RegisterUserResponse.cs

│ │ ├── GetUserById/

│ │ │ ├── GetUserByIdQuery.cs

│ │ │ ├── GetUserByIdQueryHandler.cs

│ │ │ └── GetUserByIdResponse.cs

│ │ └── UpdateProfile/

│ │ ├── UpdateProfileCommand.cs

│ │ └── UpdateProfileCommandHandler.cs

│ │

│ ├── Auth/

│ │ ├── Login/

│ │ │ ├── LoginCommand.cs

│ │ │ ├── LoginCommandHandler.cs

│ │ │ └── LoginResponse.cs

│ │ ├── RefreshToken/

│ │ │ ├── RefreshTokenCommand.cs

│ │ │ └── RefreshTokenCommandHandler.cs

│ │ └── RevokeToken/

│ │ ├── RevokeTokenCommand.cs

│ │ └── RevokeTokenCommandHandler.cs

│ │

│ ├── Roles/

│ │ ├── CreateRole/

│ │ │ ├── CreateRoleCommand.cs

│ │ │ ├── CreateRoleCommandHandler.cs

│ │ │ └── CreateRoleCommandValidator.cs

│ │ └── AssignPermissionToRole/

│ │ ├── AssignPermissionToRoleCommand.cs

│ │ └── AssignPermissionToRoleCommandHandler.cs

│ │

│ └── Permissions/

│ └── ManagePermissionCommands...

│

├── Shared/

│ ├── DTOs/

│ ├── Exceptions/

│ └── Mappers/

│

├── Security/

│ ├── AuthorizationPolicies.cs

│ └── PermissionNames.cs

│

├── Specifications/

│ └── UserSpecificationsAdapter.cs

│

├── Integration/

│ ├── OutboxMessage.cs

│ └── OutboxProcessor.cs

│

└── Internal/

└── AssemblyMarkers, Helpers

```

---

## ✅ توضیحات کلیدی

### 1. \*\*`Common` — ستون فقرات لایه Application\*\*

- \*\*`Abstractions`\*\*: تمام قراردادها (Interfaces) — وابستگی به پیاده‌سازی ندارد.

- \*\*`Primitives`\*\*: مدل‌های بنیادی `Error`, `Result`, `Result<T>`.

- \*\*`Exceptions`\*\*: استثناهای سفارشی برای مدیریت خطا در لایه بالاتر.

- \*\*`Behaviors`\*\*: رفتارهای MediatR برای Cross-Cutting Concerns.

- \*\*`Mappings`\*\*: پروفایل‌های AutoMapper.

- \*\*`Services`\*\*: سرویس‌های ترکیبی (Orchestration) مانند `UserManager`.

- \*\*`Extensions`\*\*: متدهای کمکی.

### 2. \*\*`Features` — قلب تپندهٔ سیستم\*\*

- هر فیچر در یک پوشه مستقل (Vertical Slice).

- هر فیچر شامل `Command/Query`, `Handler`, `Validator`, `Response`.

- نام‌گذاری یکدست: `RegisterUserCommand`, `RegisterUserCommandHandler`.

### 3. \*\*`Shared` — برای چیزهای واقعاً مشترک\*\*

- DTOها یا مپ‌هایی که بین چند فیچر استفاده می‌شن.

### 4. \*\*`Security` — سیاست‌ها و پرمیژن‌ها\*\*

- `PermissionNames.cs`: لیست ثابت پرمیژن‌ها.

- `AuthorizationPolicies.cs`: سیاست‌های احراز هویت.

### 5. \*\*`Specifications` — اگر از Specification Pattern استفاده می‌کنی\*\*

- `UserSpecificationsAdapter.cs`: برای استفاده آسان از Specificationها در Application.

### 6. \*\*`Integration` — اگر از Outbox/Event-Driven استفاده می‌کنی\*\*

- `OutboxMessage.cs`, `OutboxProcessor.cs`.

### 7. \*\*`Internal` — فایل‌های کم‌اهمیت\*\*

- `AssemblyMarker.cs` برای ثبت خودکار در DI.

---

## ✅ نکات مهم

| نکته | توضیح |

|------|-------|

| ✅ \*\*AssemblyMarker.cs\*\* | یک کلاس خالی برای `RegisterServicesFromAssembly` در MediatR و DI. |

| ✅ \*\*`sealed record` برای Command/Query/Response\*\* | immutable، کوتاه، و خواناتر. |

| ✅ \*\*`ITransactionalCommand`\*\* | فقط Commandهایی که نیاز به تراکنش دارن این اینترفیس رو پیاده‌سازی می‌کنن. |

| ✅ \*\*ترتیب Behaviors در DI\*\* | UnhandledException → Logging → Validation → Transaction. |

---

## ✅ گام بعدی: پیاده‌سازی `Behaviors`

حالا که ساختار رو داریم، می‌تونیم بریم سراغ:

> 🚀 \*\*`Common/Behaviors`\*\*

که مستقیماً از `Primitives`, `Exceptions`, و `Abstractions` استفاده می‌کنن.

آماده‌ای؟ بزن بریم جلو! 💪