حتما! اینجا یک ساختار پوشه‌ای کامل و استاندارد برای پروژه ASP.NET Core احراز هویت (Authorization\\_Login\\_Asp.Net) همراه با سرویس لاگینگ (Serilog) و لایه‌بندی دقیق Domain، Application، Infrastructure، API و Tests آوردم.

---

# ساختار پوشه‌بندی کامل پروژه Authorization\\_Login\\_Asp.Net

```

Authorization\_Login\_Asp.Net

│

├── Authorization\_Login\_Asp.Net.Domain # لایه دامنه (Domain Layer)

│ ├── Entities # مدل‌های اصلی دامنه (Entities)

│ │ ├── User.cs

│ │ ├── Role.cs

│ │ ├── Permission.cs

│ │ ├── RolePermission.cs

│ │ └── RefreshToken.cs

│ │

│ ├── Enums # Enumهای دامنه

│ │ ├── UserRole.cs

│ │ ├── PermissionType.cs

│ │ └── RefreshTokenStatus.cs

│ │

│ ├── ValueObjects # Value Objectها (مانند Email)

│ │ └── Email.cs

│ │

│ └── Exceptions # Exceptionهای دامنه (DomainException)

│ └── DomainException.cs

│

├── Authorization\_Login\_Asp.Net.Application # لایه کاربردی (Application Layer)

│ ├── DTOs # مدل‌های داده ورودی/خروجی (Data Transfer Objects)

│ │ ├── RegisterRequest.cs

│ │ ├── LoginRequest.cs

│ │ ├── RefreshTokenRequest.cs

│ │ └── AuthResponse.cs

│ │

│ ├── Interfaces # اینترفیس‌ها برای سرویس‌ها، ریپازیتوری‌ها و UnitOfWork

│ │ ├── IUserService.cs

│ │ ├── IJwtService.cs

│ │ ├── IRefreshTokenService.cs

│ │ ├── IRepository.cs # اینترفیس پایه ریپازیتوری

│ │ ├── IUserRepository.cs

│ │ ├── IRoleRepository.cs

│ │ ├── IPermissionRepository.cs

│ │ ├── IRolePermissionRepository.cs

│ │ ├── IRefreshTokenRepository.cs

│ │ └── IUnitOfWork.cs

│ │

│ ├── Validators # ولیدیتورهای FluentValidation

│ │ ├── RegisterRequestValidator.cs

│ │ ├── LoginRequestValidator.cs

│ │ └── RefreshTokenRequestValidator.cs

│ │

│ ├── Exceptions # استثناهای اعتبارسنجی (ValidationException)

│ │ └── ValidationException.cs

│ │

│ └── Mappings # پروفایل‌های AutoMapper

│ └── MappingProfile.cs

│

├── Authorization\_Login\_Asp.Net.Infrastructure # لایه زیرساخت (Infrastructure Layer)

│ ├── Data # EF Core DbContext و پیکربندی‌های مدل

│ │ ├── AppDbContext.cs

│ │ └── Configurations

│ │ ├── UserConfiguration.cs

│ │ ├── RoleConfiguration.cs

│ │ ├── PermissionConfiguration.cs

│ │ ├── RolePermissionConfiguration.cs

│ │ └── RefreshTokenConfiguration.cs

│ │

│ ├── Services # پیاده‌سازی سرویس‌های لایه Application

│ │ ├── UserService.cs

│ │ ├── JwtService.cs

│ │ ├── RefreshTokenService.cs

│ │ ├── PermissionService.cs

│ │ └── LoggingService.cs # سرویس اختصاصی لاگینگ (اختیاری)

│ │

│ ├── Repositories # پیاده‌سازی ریپازیتوری‌ها (در صورت جدا کردن)

│ │ ├── UserRepository.cs

│ │ ├── RoleRepository.cs

│ │ ├── PermissionRepository.cs

│ │ ├── RolePermissionRepository.cs

│ │ └── RefreshTokenRepository.cs

│ │

│ ├── Extensions # اکستنشن‌های DI و سرویس‌ها

│ │ └── ServiceCollectionExtensions.cs

│ │

│ └── Configurations # تنظیمات برنامه (مثلاً JWT)

│ └── JwtSettings.cs

│

├── Authorization\_Login\_Asp.Net.Api # لایه API (Web API)

│ ├── Controllers # کنترلرهای WebAPI

│ │ ├── AuthController.cs

│ │ └── AdminController.cs

│ │

│ ├── Middleware # میان‌افزارها (Middleware)

│ │ └── ExceptionMiddleware.cs

│ │

│ ├── Filters # فیلترها (مثلاً فیلتر اعتبارسنجی مدل)

│ │ └── ValidationFilter.cs

│ │

│ ├── Authorization # پیاده‌سازی Authorization بر پایه Claims

│ │ ├── PermissionRequirement.cs

│ │ ├── PermissionHandler.cs

│ │ └── AuthorizationPolicies.cs

│ │

│ ├── Program.cs # نقطه شروع برنامه، DI، Middleware، JWT و Serilog

│ ├── appsettings.json # تنظیمات عمومی برنامه (اتصال دیتابیس، JWT و Serilog)

│ └── appsettings.Development.json # تنظیمات محیط توسعه

│

├── Authorization\_Login\_Asp.Net.Tests # پوشه تست‌ها

│ ├── UnitTests # تست‌های واحد (Unit Tests)

│ │ ├── AuthControllerTests.cs

│ │ ├── UserServiceTests.cs

│ │ └── RefreshTokenServiceTests.cs

│ │

│ └── TestHelpers # ابزارها و کلاس‌های کمکی برای تست

│ └── FakeAppDbContext.cs

│

├── docker-compose.yml # کانفیگ Docker جهت راه‌اندازی دیتابیس SQL Server محلی

└── Authorization\_Login\_Asp.Net.sln # فایل Solution

```

---

# نکات تکمیلی در ساختار:

\* \*\*LoggingService.cs\*\* داخل Infrastructure/Services:

پیاده‌سازی کلاس لاگینگ اختصاصی (در صورت نیاز) با استفاده از `ILogger<T>` که خودش توسط Serilog کانفیگ شده است.

\* \*\*ServiceCollectionExtensions.cs\*\*:

ثبت همه سرویس‌ها، ریپازیتوری‌ها، UnitOfWork، و Logging در DI کانتینر، به طوری که در Program.cs فقط کافی باشد یک خط فراخوانی شود.

\* \*\*Program.cs\*\*:

کانفیگ کامل Serilog (خواندن تنظیمات از appsettings.json)، راه‌اندازی JWT Authentication و Authorization و افزودن Middlewareهای سفارشی مثل ExceptionMiddleware.

\* \*\*Folder Separation\*\*:

جداسازی منطق دامنه، کاربرد، زیرساخت و API باعث می‌شود کد تمیز، قابل تست و قابل نگهداری باشد.