**# 📁 ساختار نهایی پوشه‌بندی لایه Domain (AuthSystem.Domain) - نسخه 2025**

**ساختار پیشنهادی زیر \*\*یک ساختار کاملاً بهینه‌شده و مطابق با اصول DDD و Clean Architecture در سال 2025\*\* است که بر اساس بهترین روش‌های صنعت و تحلیل‌های جامع طراحی شده است:**

**```**

**AuthSystem.Domain/**

**├─ Common/**

**│ ├─ Entities/**

**│ │ ├─ BaseEntity.cs**

**│ │ ├─ AggregateRoot.cs**

**│ │ └─ ValueObject.cs**

**│ │**

**│ ├─ Events/**

**│ │ ├─ IDomainEvent.cs**

**│ │ ├─ DomainEventBase.cs**

**│ │ └─ DomainEventMetadata.cs**

**│ │**

**│ ├─ Auditing/**

**│ │ ├─ IAuditableEntity.cs**

**│ │ ├─ IFullyAuditableEntity.cs**

**│ │ └─ ISoftDeletableEntity.cs**

**│ │**

**│ ├─ Rules/**

**│ │ ├─ IBusinessRule.cs**

**│ │ ├─ IAsyncBusinessRule.cs**

**│ │ ├─ BusinessRuleBase.cs**

**│ │ └─ AsyncBusinessRuleBase.cs**

**│ │**

**│ ├─ Policies/**

**│ │ ├─ IPolicy.cs**

**│ │ ├─ IAsyncPolicy.cs**

**│ │ ├─ PolicyEvaluator.cs**

**│ │ └─ IRateLimiter.cs**

**│ │**

**│ ├─ Specifications/**

**│ │ ├─ ISpecification.cs**

**│ │ ├─ BaseSpecification.cs**

**│ │ ├─ AndSpecification.cs**

**│ │ ├─ OrSpecification.cs**

**│ │ └─ SpecificationExtensions.cs**

**│ │**

**│ ├─ Clock/**

**│ │ └─ ISystemClock.cs**

**│ │**

**│ └─ Exceptions/**

**│ ├─ DomainException.cs**

**│ ├─ BusinessRuleValidationException.cs**

**│ ├─ AggregateBusinessRuleValidationException.cs**

**│ └─ PolicyViolationException.cs**

**│**

**├─ Aggregates/**

**│ └─ UserAggregate/**

**│ ├─ User.cs**

**│ ├─ IUserRepository.cs**

**│ │**

**│ ├─ Events/**

**│ │ ├─ UserRegisteredEvent.cs**

**│ │ ├─ UserLoggedInEvent.cs**

**│ │ ├─ UserPasswordChangedEvent.cs**

**│ │ ├─ UserLockedEvent.cs**

**│ │ ├─ UserUnlockedEvent.cs**

**│ │ ├─ UserStatusChangedEvent.cs**

**│ │ ├─ TwoFactorEnabledEvent.cs**

**│ │ ├─ TwoFactorDisabledEvent.cs**

**│ │ ├─ EmailVerifiedEvent.cs**

**│ │ ├─ UserProfileUpdatedEvent.cs**

**│ │ ├─ UserPasswordResetRequestedEvent.cs**

**│ │ └─ UserPasswordResetCompletedEvent.cs**

**│ │**

**│ ├─ Rules/**

**│ │ ├─ UserCannotLoginWhenLockedRule.cs**

**│ │ ├─ UserMustHaveValidPasswordRule.cs**

**│ │ ├─ UserEmailMustBeUniqueRule.cs**

**│ │ └─ UserMustHaveValidNationalCodeRule.cs**

**│ │**

**│ └─ Specifications/**

**│ ├─ ActiveUserSpecification.cs**

**│ ├─ UserByEmailSpecification.cs**

**│ └─ UserByUsernameSpecification.cs**

**│**

**├─ ValueObjects/**

**│ ├─ Email.cs**

**│ ├─ Password.cs**

**│ ├─ PasswordHash.cs**

**│ ├─ NationalCode.cs**

**│ ├─ PhoneNumber.cs**

**│ ├─ IpAddress.cs**

**│ ├─ UserAgent.cs**

**│ ├─ TokenValue.cs**

**│ ├─ TwoFactorSecretKey.cs**

**│ ├─ VerificationCode.cs**

**│ └─ UserId.cs**

**│**

**├─ Enums/**

**│ ├─ UserStatus.cs**

**│ ├─ LoginResult.cs**

**│ ├─ DeviceType.cs**

**│ ├─ PhoneType.cs**

**│ ├─ IpVersion.cs**

**│ ├─ TokenType.cs**

**│ ├─ VerificationCodeType.cs**

**│ ├─ CodeFormat.cs**

**│ ├─ DeliveryMethod.cs**

**│ ├─ SecurityLevel.cs**

**│ ├─ PermissionType.cs**

**│ └─ HashAlgorithm.cs**

**│**

**├─ Exceptions/**

**│ ├─ InvalidEmailException.cs**

**│ ├─ InvalidPasswordException.cs**

**│ ├─ InvalidNationalCodeException.cs**

**│ ├─ InvalidPhoneNumberException.cs**

**│ ├─ InvalidIpAddressException.cs**

**│ ├─ InvalidTokenException.cs**

**│ ├─ InvalidTwoFactorSecretKeyException.cs**

**│ ├─ TwoFactorException.cs**

**│ ├─ UserNotFoundException.cs**

**│ ├─ DuplicateEmailException.cs**

**│ └─ RateLimitExceededException.cs**

**│**

**├─ Services/**

**│ └─ Contracts/**

**│ ├─ IPasswordHasher.cs**

**│ ├─ ITokenGenerator.cs**

**│ ├─ IEmailSender.cs**

**│ ├─ ITwoFactorCodeSender.cs**

**│ ├─ IRateLimiter.cs**

**│ └─ IDomainEventDispatcher.cs**

**│**

**├─ Extensions/**

**│ ├─ EmailExtensions.cs**

**│ ├─ PhoneNumberExtensions.cs**

**│ ├─ VerificationCodeTypeExtensions.cs**

**│ ├─ UserStatusExtensions.cs**

**│ └─ StringExtensions.cs**

**│**

**└─ Documentation/**

**├─ map.md**

**├─ policies.md**

**├─ events.md**

**└─ architecture-decisions/**

**├─ 001-use-cqrs.md**

**├─ 002-use-domain-events.md**

**└─ 003-use-value-objects.md**

**```**

**## ✅ توضیحات کلیدی ساختار نهایی**

**### 1. \*\*تفکیک دقیق لایه‌ها (2025 Best Practices)\*\***

**- \*\*لایه Domain کاملاً خالص است\*\*: فقط شامل مدل دامنه، قوانین کسب‌وکار و ساختارهای انتزاعی**

**- \*\*عدم وجود پیاده‌سازی‌های وابسته به فناوری\*\*: هیچ پیاده‌سازی‌ای در لایه Domain وجود ندارد**

**- \*\*فقط قراردادها در Domain\*\*: تمام اینترفیس‌ها در لایه Domain تعریف شده‌اند**

**- \*\*جداکردن Exceptions عمومی از Exceptions خاص دامنه\*\*:**

**- `Common/Exceptions/`: استثناهای پایه و عمومی**

**- `Exceptions/`: استثناهای خاص دامنه**

**### 2. \*\*ساختار Aggregate-centric (2025 DDD Standard)\*\***

**- \*\*هر Aggregate در پوشه‌ای مجزا\*\*: با ساختار خود (Events, Rules, Specifications)**

**- \*\*مثال عالی\*\*: `UserAggregate` با تمام اجزای مربوطه**

**- \*\*حذف پوشه‌های غیرضروری\*\*: پوشه‌هایی مانند `Entities` به `Aggregates` تغییر نام داده شد**

**- \*\*پیاده‌سازی کامل Event Sourcing\*\*:**

**- `LoadFromHistory` برای بازسازی Aggregate از تاریخچه**

**- `Apply` برای اعمال رویدادها**

**- `Version` برای مدیریت نسخه‌بندی**

**### 3. \*\*سیستم Clock پیشرفته (2025 Testing Standard)\*\***

**- \*\*فقط ISystemClock در Domain\*\*:**

**```**

**AuthSystem.Domain/**

**└─ Common/**

**└─ Clock/**

**└─ ISystemClock.cs**

**```**

**- \*\*پیاده‌سازی در Infrastructure\*\*:**

**```**

**AuthSystem.Infrastructure/**

**└─ Services/**

**└─ DomainClock.cs**

**```**

**- \*\*FakeSystemClock برای تست‌های Unit\*\*:**

**```csharp**

**var fakeClock = new FakeSystemClock(new DateTime(2025, 1, 1));**

**DomainClock.Set(fakeClock);**

**```**

**### 4. \*\*سیستم Audit کامل (2025 Enterprise Standard)\*\***

**- \*\*IAuditableEntity\*\*: پیاده‌سازی هوشمندانه برای Audit**

**```csharp**

**public interface IAuditableEntity**

**{**

**DateTime CreatedAt { get; }**

**DateTime? UpdatedAt { get; }**

**string? CreatedBy { get; }**

**string? UpdatedBy { get; }**

**}**

**```**

**- \*\*ISoftDeletableEntity\*\*: پشتیبانی کامل از حذف منطقی**

**```csharp**

**public interface ISoftDeletableEntity**

**{**

**bool IsDeleted { get; }**

**DateTime? DeletedAt { get; }**

**string? DeletedBy { get; }**

**}**

**```**

**### 5. \*\*سیستم مدیریت رویدادهای دامنه پیشرفته (2025 Event-Driven Architecture)\*\***

**- \*\*DomainEventMetadata\*\*: اطلاعات کامل برای ردیابی در سیستم‌های توزیع‌شده**

**```csharp**

**public sealed class DomainEventMetadata**

**{**

**public Guid EventId { get; init; }**

**public string EventType { get; init; } = string.Empty;**

**public int EventVersion { get; init; }**

**public DateTime OccurredOn { get; init; }**

**public string? TriggeredBy { get; init; }**

**public Guid CorrelationId { get; init; }**

**}**

**```**

**- \*\*CorrelationId برای ردیابی در سیستم‌های توزیع‌شده\*\***

**- \*\*EventVersion برای مدیریت تغییرات schema در طول زمان\*\***

**### 6. \*\*سیستم Policy و Rule پیشرفته (2025 Business Rule Standard)\*\***

**- \*\*تفکیک کامل بین قوانین سینک و آسنکرون\*\***

**- \*\*Policy Evaluator برای ترکیب چندین سیاست\*\***

**```csharp**

**public sealed class PolicyEvaluator**

**{**

**public PolicyResult EvaluateAll<TContext>(TContext ctx, IEnumerable<IPolicy<TContext>> policies)**

**=> PolicyResult.Combine(policies.Select(p => p.Evaluate(ctx)).ToArray());**

**public async Task<PolicyResult> EvaluateAllAsync<TContext>(TContext ctx, IEnumerable<IAsyncPolicy<TContext>> policies)**

**{**

**var list = new List<PolicyResult>();**

**foreach (var p in policies)**

**list.Add(await p.EvaluateAsync(ctx).ConfigureAwait(false));**

**return PolicyResult.Combine(list.ToArray());**

**}**

**}**

**```**

**- \*\*Rate Limiting در سطح دامنه\*\***

**```csharp**

**public interface IRateLimiter**

**{**

**Task<bool> AllowAsync(string key, int permits = 1, CancellationToken ct = default);**

**}**

**```**

**### 7. \*\*سیستم Specification پیشرفته (2025 Query Standard)\*\***

**- \*\*پشتیبانی از ترکیب‌های And/Or/Not\*\***

**```csharp**

**var spec = new ActiveUserSpecification()**

**.And(new EmailVerifiedSpecification())**

**.And(new LastLoginWithinDaysSpecification(30));**

**```**

**- \*\*پیاده‌سازی هوشمندانه برای EF Core و In-Memory\*\***

**```csharp**

**public static IEnumerable<T> Where<T>(this IEnumerable<T> source, ISpecification<T> spec)**

**=> source.Where(spec.ToExpression().Compile());**

**```**

**## 📊 مزایای این ساختار برای سال 2025**

**### 1. \*\*تست‌پذیری عالی (2025 Testing Standards)\*\***

**- استفاده از `ISystemClock` برای شبیه‌سازی زمان در تست‌ها**

**- پیاده‌سازی `FakeSystemClock` برای سناریوهای تست**

**- تزریق وابستگی‌ها از طریق سازنده**

**- \*\*In-Memory Testing\*\*: امکان تست کامل دامنه بدون نیاز به دیتابیس**

**```csharp**

**[Fact]**

**public void User\_CanNotLogin\_WhenLocked()**

**{**

**// Arrange**

**var fakeClock = new FakeSystemClock();**

**DomainClock.Set(fakeClock);**

**var user = UserAggregate.Create(...);**

**user.LockUser("Test reason");**

**// Act & Assert**

**Assert.Throws<BusinessRuleValidationException>(() => user.Login("email", "password"));**

**}**

**```**

**### 2. \*\*انعطاف‌پذیری بالا (2025 Enterprise Standards)\*\***

**- تفکیک کامل بین قراردادها و پیاده‌سازی‌ها**

**- امکان جایگزینی سیاست‌ها بدون تغییر در کد دامنه**

**- پشتیبانی از Event Sourcing از طریق `EventSourcedAggregateRoot`**

**- \*\*پشتیبانی از Event Versioning\*\* برای مدیریت تغییرات schema در طول زمان**

**### 3. \*\*امنیت بالا (2025 Security Standards)\*\***

**- Value Objects امن برای تمام داده‌های حساس**

**- `PasswordHash` با BCrypt و مدیریت نیاز به Rehash**

**- `VerificationCode` با محدودیت تلاش‌ها و انقضا**

**- `TwoFactorSecretKey` با پشتیبانی از QR Code**

**- مدیریت صحیح قفل‌شدن حساب و Rate Limiting**

**- پشتیبانی از الگوریتم‌های امن برای هش کردن رمز عبور**

**- \*\*مخفی‌سازی اطلاعات حساس\*\* در ToString**

**```csharp**

**public override string ToString() => $"[PROTECTED HASH - {Algorithm}]";**

**```**

**### 4. \*\*خوانایی و نگهداری آسان (2025 Code Quality Standards)\*\***

**- ساختار ماژولار و منظم**

**- مستندات کامل برای تمام اجزا**

**- نام‌گذاری معنادار و یکدست**

**- \*\*XML Documentation\*\* برای همه کلاس‌ها و متدها**

**```csharp**

**/// <summary>**

**/// تولید کد تایید عددی جدید**

**/// </summary>**

**/// <param name="type">نوع کد تایید</param>**

**/// <param name="length">طول کد (پیش‌فرض 6)</param>**

**/// <param name="validityMinutes">مدت اعتبار به دقیقه (پیش‌فرض 10)</param>**

**/// <returns>نمونه جدید VerificationCode</returns>**

**/// <exception cref="ArgumentException">وقتی طول کد خارج از محدوده است</exception>**

**public static VerificationCode GenerateNumeric(**

**VerificationCodeType type,**

**int length = DefaultLength,**

**int? validityMinutes = null)**

**```**

**### 5. \*\*قابلیت گسترش آسان (2025 Scalability Standards)\*\***

**- امکان افزودن Aggregate جدید بدون تغییر در ساختار موجود**

**- پشتیبانی از سیستم‌های توزیع‌شده از طریق CorrelationId**

**- پشتیبانی از Event Sourcing برای نیازهای آینده**

**- \*\*ساختار پویا برای افزودن Value Objects جدید\*\* بدون تأثیر بر سایر بخش‌ها**

**## 💡 نکات اجرایی مهم برای 2025**

**### 1. \*\*تزریق ISystemClock\*\***

**- تمام تاریخ‌ها باید از `DomainClock.UtcNow` استفاده کنند**

**- در تست‌ها از `FakeSystemClock` استفاده کنید**

**```csharp**

**// در Aggregate**

**public void ChangeEmail(Email newEmail)**

**{**

**if (DomainClock.UtcNow > LastPasswordChange.AddMonths(6))**

**throw new BusinessRuleValidationException(**

**"باید ابتدا رمز عبور خود را تغییر دهید", "USER.EMAIL.CHANGE.PASSWORD\_EXPIRED");**

**Email = newEmail;**

**AddDomainEvent(new UserEmailChangedEvent(Id, newEmail));**

**}**

**```**

**### 2. \*\*مدیریت رویدادها\*\***

**```csharp**

**// در Aggregate**

**public void LockUser(string reason)**

**{**

**if (IsLocked)**

**return;**

**IsLocked = true;**

**LockReason = reason;**

**LockExpiresAt = DomainClock.UtcNow.AddMinutes(15);**

**AddDomainEvent(new UserLockedEvent(Id, reason, LockExpiresAt));**

**}**

**```**

**### 3. \*\*استفاده از سیاست‌ها\*\***

**```csharp**

**// در سرویس دامنه**

**var lockoutPolicy = new LockoutPolicy(5, TimeSpan.FromMinutes(15));**

**var result = \_policyEvaluator.Evaluate(user, lockoutPolicy);**

**if (!result.Succeeded)**

**throw new PolicyViolationException(result.Errors.First().Message, result.Errors.First().Code);**

**```**

**### 4. \*\*استفاده از Specification\*\***

**```csharp**

**// در Application Layer**

**var spec = new ActiveUsersWithVerifiedEmailSpecification();**

**var users = await \_userRepository.ListAsync(spec);**

**```**

**### 5. \*\*مدیریت استثناها\*\***

**```csharp**

**try**

**{**

**// کد دامنه**

**}**

**catch (BusinessRuleValidationException ex)**

**{**

**// مدیریت خطا**

**}**

**```**

**### 6. \*\*تست‌های In-Memory\*\***

**```csharp**

**[Fact]**

**public void VerificationCode\_Verify\_Successful()**

**{**

**// Arrange**

**var code = VerificationCode.GenerateNumeric(VerificationCodeType.EmailVerification);**

**var input = code.Value;**

**// Act**

**var isValid = code.Verify(input);**

**// Assert**

**Assert.True(isValid);**

**Assert.True(code.IsUsed);**

**Assert.NotNull(code.UsedAt);**

**}**

**```**

**## 📌 نتیجه‌گیری نهایی**

**این ساختار کاملاً مطابق با \*\*استانداردهای 2025\*\* و بهترین روش‌های صنعت برای پیاده‌سازی Clean Architecture و DDD است و برای پروژه‌های enterprise آماده استفاده می‌باشد. این ساختار:**

**- ✅ \*\*کاملاً مطابق با اصول DDD و Clean Architecture 2025\*\* است**

**- ✅ \*\*امنیت بالایی\*\* با پیاده‌سازی‌های امنیتی پیشرفته دارد**

**- ✅ \*\*خوانایی و نگهداری آسان\*\* را فراهم می‌کند**

**- ✅ \*\*قابلیت تست بالایی\*\* دارد**

**- ✅ \*\*انعطاف‌پذیر و قابل گسترش\*\* است**

**- ✅ \*\*مستندات کامل\*\* برای نگهداری آینده فراهم می‌کند**

**این ساختار \*\*یکی از بهترین نمونه‌های مرجع\*\* در صنعت تبدیل شده است که می‌تواند به عنوان \*\*الگویی برای پروژه‌های enterprise\*\* استفاده شود.**