Software Design Document (SDD) — LMS-Mini

Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| Phiên bản | Ngày | Người biên tập | Nội dung chỉnh sửa |
| 1.1 | 2025-08-25 | Sok Kim Thanh / ChatGPT | Phiên bản biên tập đầy đủ, tái cấu trúc nội dung, bổ sung Document Control, Scope, Glossary. |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Phiên bản | Ngày | Người sửa | Nội dung sửa đổi |
| 1.0 | 2025-08-20 | Sok KimThanh | Bản nháp đầu tiên. |
| 1.1 | 2025-08-25 | Sok Kim Thanh | Biên tập sạch, chuẩn hoá cấu trúc. |
| 1.2 | 2025-08-26 | Sok Kim Thanh | Bổ sung các mục Non-Functional Requirements (NFRs), Deployment, Error Handling, Testing Strategy, Backup & Disaster Recovery. Cập nhật lại Mục lục. |

Mục lục

[1 Giới thiệu 1](#_Toc207092754)

[1.1 Mục tiêu 1](#_Toc207092755)

[1.2 Phạm vi 1](#_Toc207092756)

[1.3 Độc giả mục tiêu 1](#_Toc207092757)

[1.4 Thuật ngữ (Glossary) 1](#_Toc207092758)

[2 Tổng quan hệ thống (System Context) 6](#_Toc207092759)

[2.1 Stakeholders 6](#_Toc207092760)

[2.2 Sơ đồ tổng quan hệ thống 7](#_Toc207092761)

[3 Kiến trúc & Quyết định kiến trúc 8](#_Toc207092762)

[3.1 Component Diagram – LMS Mini 8](#_Toc207092763)

[3.2 Use-Case Diagram – LMS Mini 2](#_Toc207092764)

[3.3 Mô hình kiến trúc 2](#_Toc207092765)

[3.4 Sơ đồ kiến trúc tổng thể 3](#_Toc207092766)

[3.4.1 Ví dụ: Tạo khóa học mới (Create Course) 3](#_Toc207092767)

[3.4.2 Ví dụ: Đăng ký khóa học (Enroll in Course courseId và userId) 4](#_Toc207092768)

[3.4.3 Ví dụ: Nộp bài kiểm tra (Submit Quiz) 7](#_Toc207092769)

[3.5 Vai trò, trách nhiệm và mối quan hệ 10](#_Toc207092770)

[3.6 Quy ước chung 10](#_Toc207092771)

[3.7 Interface/Port chính và nơi triển khai 11](#_Toc207092772)

[3.8 Domain Events, Outbox, Specification & Cross-Cutting Concerns 14](#_Toc207092773)

[3.8.1 Domain Events 14](#_Toc207092774)

[3.8.2 Outbox Pattern 14](#_Toc207092775)

[3.8.3 Specification Pattern 17](#_Toc207092776)

[3.8.4 Timezone / Clock 21](#_Toc207092777)

[3.8.5 Audit Log 21](#_Toc207092778)

[4 Thiết kế cơ sở dữ liệu 22](#_Toc207092779)

[4.1 Audit & Soft Delete (Chuẩn hóa Audit & Xóa mềm) 22](#_Toc207092780)

[4.1.1 Mục đích 22](#_Toc207092781)

[4.1.2 Quy tắc đặt tên & kiểu dữ liệu 23](#_Toc207092782)

[4.1.3 Phạm vi áp dụng 24](#_Toc207092783)

[4.1.4 Nguyên tắc xóa mềm (Soft Delete semantics) 24](#_Toc207092784)

[4.1.5 Ghi nhận audit tự động (EF Interceptor / middleware) 24](#_Toc207092785)

[4.1.6 Indexes & Unique constraints — tương tác với Soft Delete 26](#_Toc207092786)

[4.1.7 Concurrency (tùy chọn) 26](#_Toc207092787)

[4.1.8 Best practices & vận hành 26](#_Toc207092788)

[4.1.9 Tương thích với toàn bộ SDD 27](#_Toc207092789)

[4.2 Mô hình logic — Bảng, PK, FK, Unique, Index 27](#_Toc207092790)

[4.2.1 AspNetUsers (Identity) 27](#_Toc207092791)

[4.2.2 Courses 28](#_Toc207092792)

[4.2.3 Modules 29](#_Toc207092793)

[4.2.4 Lessons 29](#_Toc207092794)

[4.2.5 Enrollments 30](#_Toc207092795)

[4.2.6 Progresses 30](#_Toc207092796)

[4.2.7 Quizzes 30](#_Toc207092797)

[4.2.8 Questions 31](#_Toc207092798)

[4.2.9 Options 31](#_Toc207092799)

[4.2.10 QuizAttempts 32](#_Toc207092800)

[4.2.11 AttemptAnswers 32](#_Toc207092801)

[4.2.12 Notifications 33](#_Toc207092802)

[4.2.13 FileAssets 33](#_Toc207092803)

[4.2.14 AuditLogs 34](#_Toc207092804)

[4.2.15 OutboxMessages 35](#_Toc207092805)

[4.3 Indexes (tóm tắt) 35](#_Toc207092806)

[4.4 Quan hệ ERD (Tóm tắt) 36](#_Toc207092807)

[4.5 Quan hệ ERD (Diagram) 37](#_Toc207092808)

[4.6 Migration 38](#_Toc207092809)

[4.7 Chiến lược Seed Data 38](#_Toc207092810)

[5 Domain Design 39](#_Toc207092811)

[5.1 Giới thiệu 39](#_Toc207092812)

[5.2 Entities & Aggregates 39](#_Toc207092813)

[5.3 Luồng Domain Event – Outbox – Audit 39](#_Toc207092814)

[5.4 Component Diagram 39](#_Toc207092815)

[5.5 Class Diagram UML 40](#_Toc207092816)

[5.6 Business Rules 41](#_Toc207092817)

[5.6.1 Mô tả nghiệp vụ 41](#_Toc207092818)

[5.6.2 Quy tắc kiểm thử 41](#_Toc207092819)

[5.7 Domain Services 42](#_Toc207092820)

[5.8 Cross-Cutting Concerns 42](#_Toc207092821)

[6 Thiết kế API & Contract 42](#_Toc207092822)

[6.1 Contract nội bộ (Port / Adapter — Application.Abstractions ↔ Infrastructure) 42](#_Toc207092823)

[6.2 Contract bên ngoài (REST API cho client) 44](#_Toc207092824)

[6.2.1 Nguyên tắc chung (Headers, Versioning, Response Envelope, i18n, CSV, v.v.) 44](#_Toc207092825)

[6.2.2 Courses (Endpoints & contract) 45](#_Toc207092826)

[6.2.3 Modules & Lessons 46](#_Toc207092827)

[6.2.4 Enrollment 46](#_Toc207092828)

[6.2.5 Quiz 47](#_Toc207092829)

[6.2.6 Reports 47](#_Toc207092830)

[6.2.7 Error Codes (tổng hợp) 48](#_Toc207092831)

[6.3 Mapping: Endpoint ↔ Interface / Service (gợi ý triển khai) 48](#_Toc207092832)

[6.4 DTOs, OpenAPI & Phụ lục 49](#_Toc207092833)

[6.5 Ghi chú triển khai & kiểm thử 49](#_Toc207092834)

[7 Thiết kế UI/UX 49](#_Toc207092835)

[7.1 Layout tổng thể 49](#_Toc207092836)

[7.2 Khả năng truy cập (A11y) 49](#_Toc207092837)

[7.3 Độ bền kết nối — Blazor Server 50](#_Toc207092838)

[7.4 Phân tách layout Admin vs Learner 50](#_Toc207092839)

[7.5 Responsive Rules 51](#_Toc207092840)

[7.6 Component Library & Styling 51](#_Toc207092841)

[7.7 Popup Rules 51](#_Toc207092842)

[8 Luồng nghiệp vụ chính (Sequence Logic) 51](#_Toc207092843)

[8.1 Use Case: Ghi danh khóa học (EnrollCourse) 51](#_Toc207092844)

[8.3 Use Case: Nộp quiz (SubmitQuiz) 54](#_Toc207092845)

[8.4 Use Case: Làm Quiz & Lưu kết quả (QuizAttemptFlow) 56](#_Toc207092846)

[8.5 Use Case: Đánh dấu hoàn thành bài học (MarkLessonCompleted) 57](#_Toc207092847)

[8.6 Use Case: Tạo khóa học mới (CreateCourse) 58](#_Toc207092848)

[8.7 Use Case: Quản lý bài học (ManageLessons) 59](#_Toc207092849)

[8.8 Use Case: Tạo bài kiểm tra (CreateQuiz) 60](#_Toc207092850)

[8.9 Use Case: Xem tiến độ học tập (ViewProgress) 61](#_Toc207092851)

[8.10 Use Case: Học bài & Ghi nhận tiến độ (LessonProgress) 62](#_Toc207092852)

[8.11 Use Case: Gửi thông báo (SendNotification) 63](#_Toc207092853)

[8.12 Use Case: Quản lý tài nguyên học tập (FileAssets) 66](#_Toc207092854)

[8.13 Use Case: Quản lý secrets, upload & security headers (Secrets & Upload Security) 67](#_Toc207092855)

[8.14 Use Case: Observability, SLO & Alerting 68](#_Toc207092856)

[9 Quy tắc bảo mật (Security rules) 69](#_Toc207092857)

[9.1 RBAC và Policies 69](#_Toc207092858)

[9.2 Authentication và Session 69](#_Toc207092859)

[9.3 Input Validation và Output Encoding 69](#_Toc207092860)

[9.4 Protection Against Common Attacks 70](#_Toc207092861)

[9.5 Auditing và Logging 70](#_Toc207092862)

[10 Non-Functional Requirements (NFRs) 70](#_Toc207092863)

[10.1 Hiệu năng (Performance) 70](#_Toc207092864)

[10.2 Khả năng mở rộng (Scalability) 71](#_Toc207092865)

[10.3 Tính sẵn sàng & Tin cậy (Availability & Reliability) 71](#_Toc207092866)

[10.4 Bảo mật & Tuân thủ (Security & Compliance) 71](#_Toc207092867)

[10.5 Khả năng quan sát & giám sát (Observability) 71](#_Toc207092868)

[11 Deployment & Infrastructure View 71](#_Toc207092869)

[11.1 Kiến trúc triển khai 71](#_Toc207092870)

[11.2 Deployment Diagram 71](#_Toc207092871)

[11.3 CI/CD Pipeline 71](#_Toc207092872)

[12 Error Handling & Exception Flow 72](#_Toc207092873)

[12.1 Nguyên tắc chung 72](#_Toc207092874)

[12.2 Error Codes (mở rộng) 72](#_Toc207092875)

[12.3 Exception Flow (ví dụ) 72](#_Toc207092876)

[13 Testing Strategy 72](#_Toc207092877)

[13.1 Unit Test 72](#_Toc207092878)

[13.2 Integration Test 72](#_Toc207092879)

[13.3 Load Test 72](#_Toc207092880)

[13.4 Security Test 72](#_Toc207092881)

[13.5 Acceptance Test 72](#_Toc207092882)

[14 Backup & Disaster Recovery 72](#_Toc207092883)

[14.1 Backup Policy 73](#_Toc207092884)

[14.2 Recovery Objective 73](#_Toc207092885)

[14.3 Disaster Recovery Plan 73](#_Toc207092886)

[15 Coding style và vận hành 73](#_Toc207092887)

[16 Deliverables / Output (Đầu ra mong muốn) 73](#_Toc207092888)

[17 Phụ lục 75](#_Toc207092889)

[17.1 Phụ lục A: Sample SQL DDL Snippets (SQL Server) 75](#_Toc207092890)

[17.1.1 Quy ước chung 75](#_Toc207092891)

[17.1.2 Code SQL DDL Snippets 75](#_Toc207092892)

[17.2 Phụ lục B – Mã nguồn PlantUML 83](#_Toc207092893)

[17.2.1 System Context Diagram 83](#_Toc207092894)

[17.2.2 Use-case Diagram - LMS mini (Tham chiếu từ SRS) 84](#_Toc207092895)

[17.2.3 Layer Diagram (Clean Architecture) 86](#_Toc207092896)

[17.2.4 Layer Diagram (Ví dụ tạo khóa học) 87](#_Toc207092897)

[17.2.5 Layer Diagram (Ví dụ Đăng ký khóa học) 88](#_Toc207092898)

[17.2.6 Layer Diagram (Ví dụ Nộp bài kiểm tra) 89](#_Toc207092899)

[17.2.7 Query Specification Pattern - LMS-Mini Domain 90](#_Toc207092900)

[17.2.8 Use-Case diagram – Roles & Permissions 92](#_Toc207092901)

[17.2.9 Component Diagram – Security Layer 93](#_Toc207092902)

[17.2.10 Component Diagram – Domain Design 94](#_Toc207092903)

[17.2.11 Class Diagram UML – Domain Design 95](#_Toc207092904)

[17.2.12 Entity Relationship Diagram (ERD) – Database Design 99](#_Toc207092905)

[17.2.13 Sequence Diagram – Ghi danh khóa học 104](#_Toc207092906)

[17.2.14 Sequence Diagram – Học viên nộp Quiz 104](#_Toc207092907)

[17.2.15 Sequence Diagram – Đánh dấu hoàn thành một bài học 105](#_Toc207092908)

[17.2.16 Sequence Diagram – Tạo khóa học mới 105](#_Toc207092909)

[17.2.17 Sequence Diagram – Quản lý bài học trong khóa học 106](#_Toc207092910)

[17.2.18 Sequence Diagram – Tạo bài kiểm tra 106](#_Toc207092911)

[17.2.19 Sequence Diagram – Xem tiến độ học tập 107](#_Toc207092912)

[17.2.20 Sequence Diagram – Gửi thông báo 107](#_Toc207092913)

[17.2.21 Implementation Diagram - UML Interface 108](#_Toc207092914)

[17.3 Phụ lục C — DTOs, Sample Payloads & OpenA 110](#_Toc207092915)

[17.3.1 DTOs (C# samples) 110](#_Toc207092916)

[17.3.3 Sample JSON payloads 114](#_Toc207092917)

[17.3.4 OpenAPI (rút gọn) — YAML 115](#_Toc207092918)

# Giới thiệu

## Mục tiêu

Tài liệu này mô tả thiết kế phần mềm (Software Design Document) cho hệ thống LMS-Mini. Nội dung bao gồm kiến trúc hệ thống, mô hình miền, API contract, dữ liệu, luồng nghiệp vụ chính và hướng dẫn triển khai.

## Phạm vi

* Bao gồm: kiến trúc backend, domain model, API contract, dữ liệu, use-case chính (ghi danh, làm bài kiểm tra, theo dõi tiến độ, thông báo).
* Không bao gồm: chi tiết giao diện người dùng (UI pixel-perfect), hạ tầng phần cứng cụ thể.

## Độc giả mục tiêu

* Backend/Frontend Developers
* QA Engineers
* DevOps
* Reviewers / Architects

## Thuật ngữ (Glossary)

|  |  |  |  |
| --- | --- | --- | --- |
| Nhóm | Thuật ngữ (Term) | Định nghĩa (Definition) | Ghi chú (Notes) |
| UI/UX & Accessibility | A11y (Accessibility) | Khả năng truy cập của ứng dụng cho người khuyết tật, tuân thủ tiêu chuẩn WCAG. | Áp dụng cho UI/UX, form, navigation, contrast, screen reader. |
| Quản trị & Vai trò | Admin | Quản trị hệ thống – quản lý người dùng, khoá học, cài đặt, báo cáo. | Có toàn quyền. |
|  | Instructor | Giảng viên – tạo, quản lý khoá học, bài giảng, quiz, xem tiến độ học viên. | Có quyền trên khoá học của mình. |
|  | Learner | Người học – tham gia khoá học, làm quiz, theo dõi tiến độ. | Role trong RBAC. |
|  | Stakeholder | Bên liên quan tới dự án. | VD: Learner, Instructor, Admin, QA, DevOps. |
| Kiến trúc & Mẫu thiết kế | Aggregate / Aggregate Root | Nhóm các đối tượng miền xử lý như một đơn vị; root là điểm truy cập duy nhất. | Theo DDD. |
|  | Application Layer | Tầng ứng dụng – điều phối luồng xử lý nghiệp vụ, validation, authorization, transaction. | Không chứa logic giao diện hay hạ tầng. |
|  | Clean Architecture | Mẫu kiến trúc tách biệt Presentation, Application, Domain, Infrastructure. | Đảm bảo độc lập, dễ bảo trì/mở rộng. |
|  | CQRS | Command Query Responsibility Segregation – tách riêng thao tác ghi và đọc. | Giúp tối ưu xử lý.(xem phụ lục D) |
|  | Domain Layer | Tầng miền nghiệp vụ – chứa entity, value object, business rule, domain event. | Không phụ thuộc tầng khác. |
|  | Presentation Layer | Tầng giao diện – hiển thị dữ liệu, nhận input từ người dùng. | VD: Blazor UI, SPA. |
|  | Repository Pattern | Mẫu tách biệt logic truy cập dữ liệu khỏi logic nghiệp vụ. | Sử dụng interface + implement. |
|  | Specification Pattern | Mẫu đặc tả tiêu chí truy vấn dưới dạng object. | Kết hợp AND/OR/NOT. |
|  | UnitOfWork | Mẫu gom các thao tác dữ liệu vào một transaction duy nhất. | Commit/rollback đồng bộ. |
| Thành phần hệ thống & Hạ tầng | Background Worker | Tiến trình nền xử lý tác vụ bất đồng bộ. | Ví dụ: gửi email qua Outbox. |
|  | Blob Storage | Lưu trữ file dung lượng lớn dạng đối tượng. | VD: Azure Blob Storage. |
|  | Key Vault / Secret Store | Dịch vụ lưu trữ bảo mật các khóa, chuỗi kết nối, secrets. | VD: Azure Key Vault. |
|  | MediatR | Thư viện .NET triển khai pattern Mediator. | Điều phối Command/Query. |
|  | Message Broker | Thành phần trung gian nhận và phân phối thông điệp (Pub/Sub). | VD: RabbitMQ, Kafka. |
| Entity & Domain Model | Attempt | Một lượt làm quiz của người học. | Lưu trong entity QuizAttempt. |
|  | AttemptAnswer | Câu trả lời cho một câu hỏi trong lượt làm quiz. | Liên kết Attempt và Question. |
|  | Course | Khoá học – tập hợp module, lesson, quiz. | Entity Course trong domain. |
|  | Enrollment | Hành động ghi danh user vào khoá học. | Ràng buộc UQ tránh trùng. |
|  | FileAsset | Tài nguyên học tập dạng file. | Metadata lưu trong DB, nội dung ở Blob Storage. |
|  | Lesson | Bài học – đơn vị nội dung trong module. | Entity Lesson. |
|  | Module | Thành phần của khoá học, chứa nhiều bài học. | Thuộc Course. |
|  | Notification | Thông báo gửi tới người học hoặc giảng viên. | Lưu trong bảng Notifications. |
|  | Progress | Tiến độ học tập của Learner. | Lưu trong bảng Progresses. |
|  | QA Engineer | Kỹ sư đảm bảo chất lượng phần mềm. | Viết test, kiểm thử. |
|  | Question | Câu hỏi trong quiz. | Có nhiều Option. |
|  | Quiz | Bài kiểm tra thuộc khoá học. | Có nhiều Question. |
|  | QuizAttempt | Một lượt làm quiz của learner. | Liên kết Quiz và User. |
|  | Use Case | Mô tả một chức năng nghiệp vụ mà hệ thống thực hiện. | Có preconditions, main flow, exceptions. |
|  | Value Object | Đối tượng miền không có định danh, so sánh theo giá trị. | VD: Email, Money. |
| Bảo mật & Toàn vẹn dữ liệu | Audit Log | Nhật ký hành động trong hệ thống. | Lưu ở AuditLogs. |
|  | CSRF Token | Token chống tấn công cross-site request forgery. | Dùng với cookie auth. |
|  | ETag / If-Match | Cơ chế kiểm soát đồng bộ và tránh xung đột khi cập nhật qua HTTP. | Dựa trên RowVersion/Hash. |
|  | Idempotency / Idempotency-Key | Thao tác có thể thực hiện nhiều lần không đổi kết quả. | Áp dụng POST ghi danh, start quiz... |
|  | RBAC | Role-Based Access Control. | Roles: Admin, Instructor, Learner. |
|  | RowVersion | Cột kiểm soát xung đột cập nhật. | Optimistic concurrency. |
|  | Scope | Phạm vi áp dụng của tài liệu/module. | Trong SDD: backend, domain model, API... |
|  | Security Headers | HTTP headers tăng cường bảo mật. | VD: CSP, HSTS... |
|  | Soft Delete | Đánh dấu bản ghi đã xoá bằng cờ IsDeleted. | Hỗ trợ khôi phục, audit. |
| Dữ liệu / Cơ sở dữ liệu | DDL (Data Definition Language) | Lệnh SQL tạo/sửa cấu trúc DB. |  |
|  | Filtered Index | Index có điều kiện WHERE trong SQL Server. |  |
|  | Migration (EF Core) | Cơ chế thay đổi schema DB có versioning. |  |
|  | PK / FK / UQ / IX / DF | Primary Key, Foreign Key, Unique, Index, Default constraint. |  |
|  | DTO | Data Transfer Object – truyền dữ liệu giữa các tầng hoặc API. | Không chứa logic nghiệp vụ. |
|  | EF Core | Entity Framework Core – ORM của .NET. | Ánh xạ entity ↔ bảng DB. |
|  | Global Query Filter | Điều kiện lọc toàn cục trong ORM. | VD: IsDeleted = 0. |
| Vận hành / Phi chức năng | Backoff Retry | Chiến lược retry với thời gian chờ tăng dần. |  |
|  | CI/CD (Continuous Integration / Continuous Delivery) | Quy trình tự động build, test, deploy. |  |
|  | Observability | Khả năng quan sát trạng thái hệ thống qua log, metric, trace. |  |
|  | Rate limiting | Giới hạn tần suất gọi API. |  |
|  | SLA (Service Level Agreement) | Thoả thuận mức dịch vụ. |  |
|  | SLI (Service Level Indicator) | Chỉ số đo chất lượng dịch vụ. |  |
|  | SLO (Service Level Objective) | Mục tiêu chất lượng dịch vụ. |  |

# Tổng quan hệ thống (System Context)

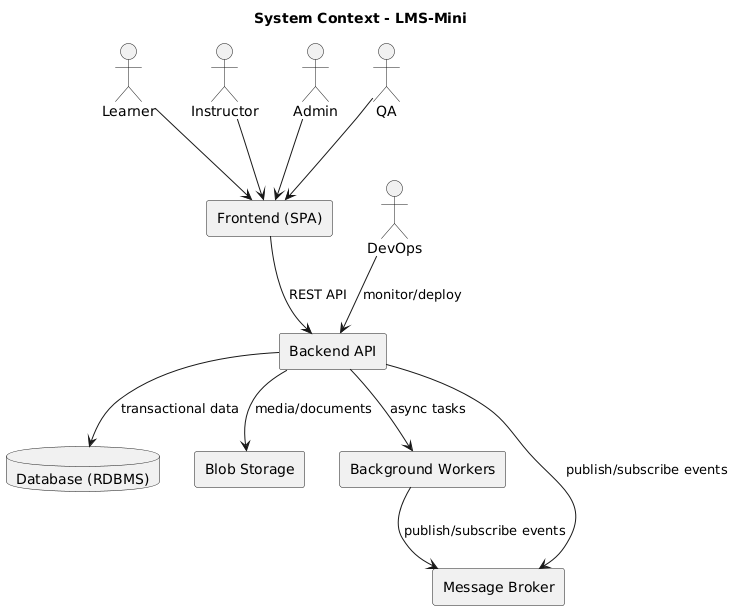
Hệ thống LMS-Mini cho phép tạo khoá học, ghi danh, làm bài kiểm tra, theo dõi tiến độ và gửi thông báo. Các thành phần chính:

* Frontend: SPA giao tiếp qua REST API.
* Backend API: xử lý use case, domain logic.
* Database (RDBMS): lưu trữ transactional data.
* Blob Storage: lưu tài liệu, media.
* Background Workers: xử lý tác vụ async.
* Message Broker: publish/subscribe các sự kiện.

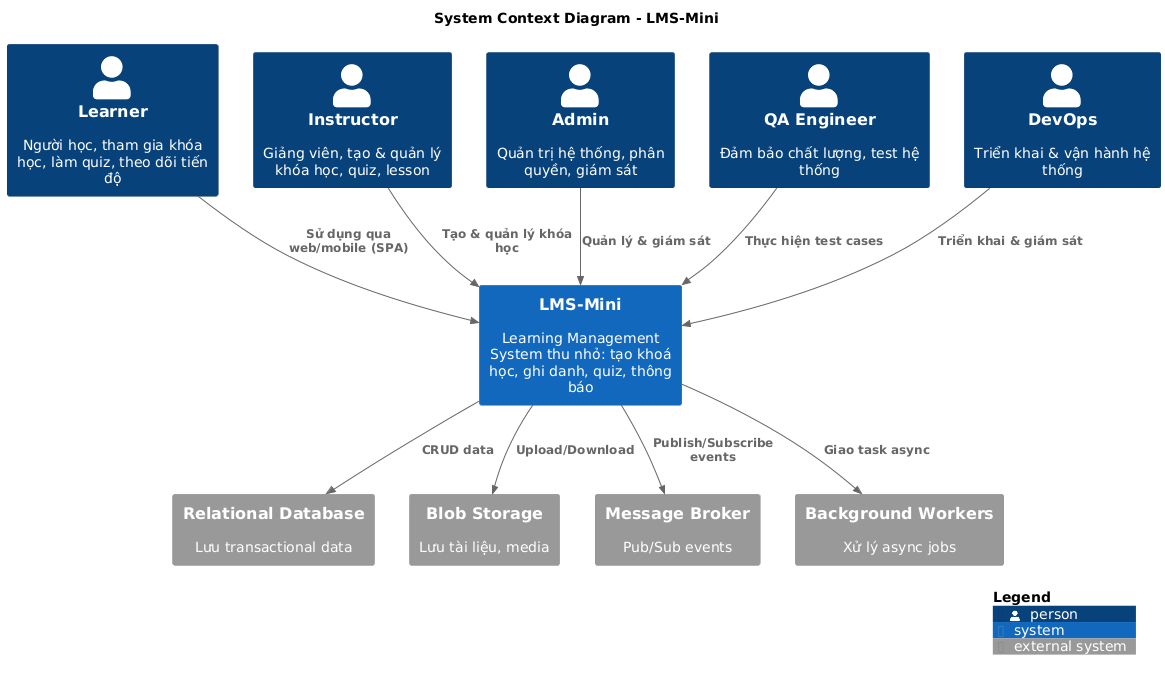
## Stakeholders

* Learner: Người dùng học tập, làm quiz, theo dõi tiến độ.
* Instructor: Tạo, quản lý khoá học, theo dõi học viên.
* Admin: Quản lý người dùng, phân quyền, giám sát hệ thống.
* QA: Đảm bảo chất lượng, viết test plan.
* DevOps: Triển khai, giám sát, vận hành hệ thống.

## Sơ đồ tổng quan hệ thống



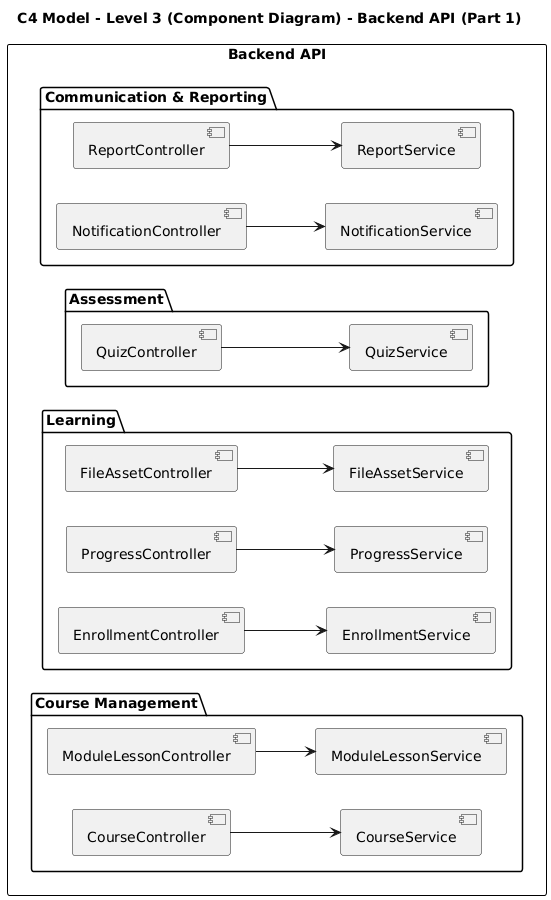
Hình 1. System Context Diagram - Stackholders



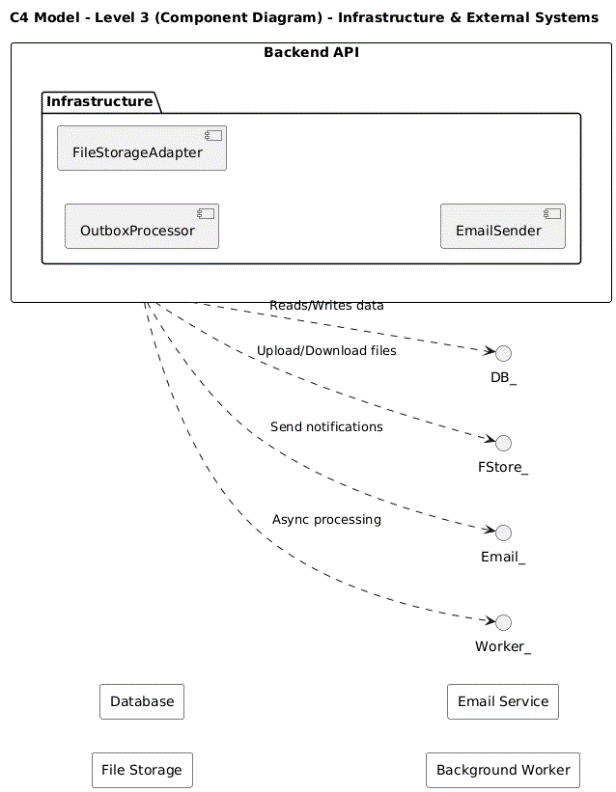
Hình 2. System Context Diagram - LMS Mini

# Kiến trúc & Quyết định kiến trúc

## Component Diagram – LMS Mini

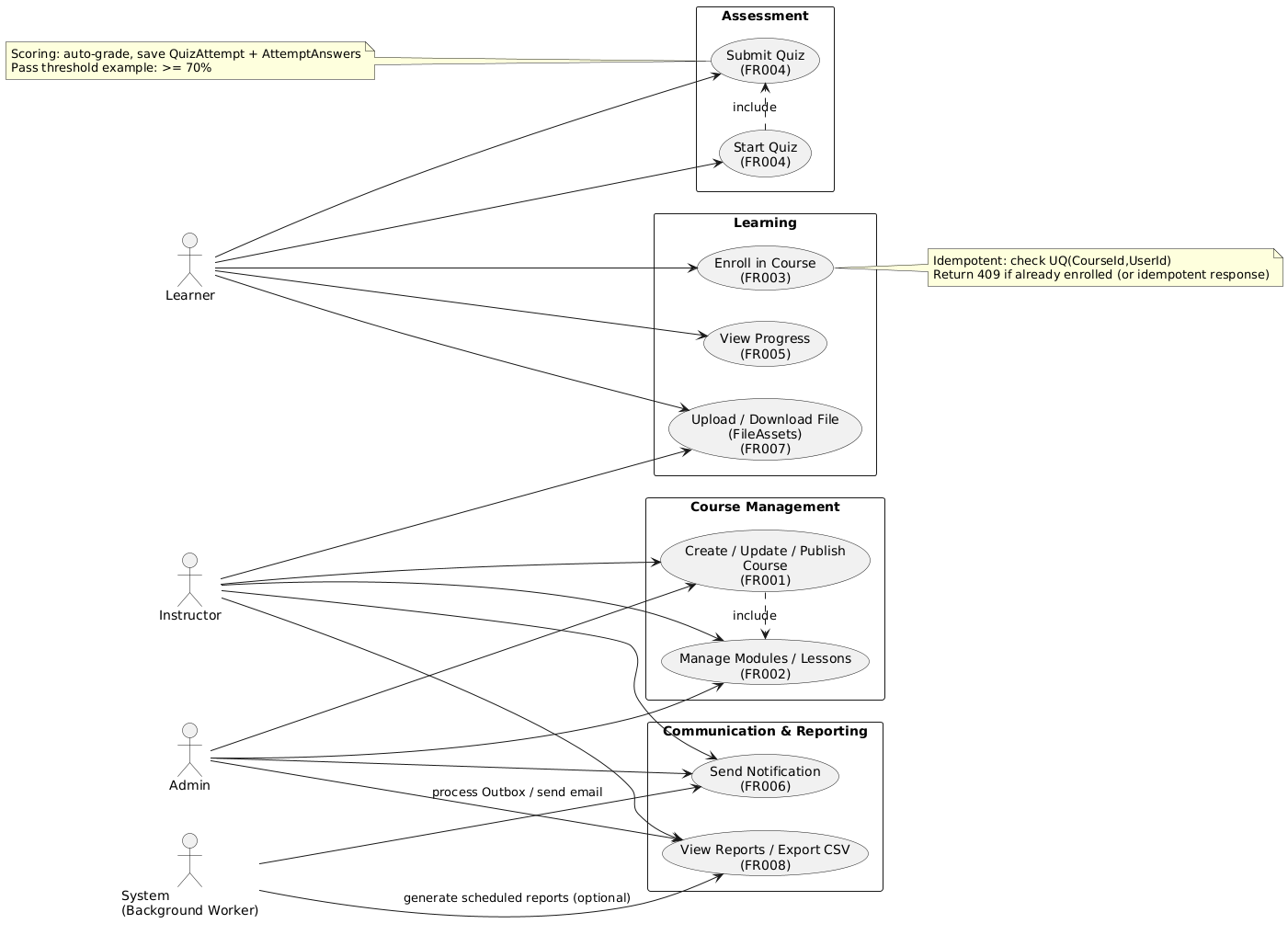


Hình 3. C4 Model - Level 3 (Component Diagram) - Backend API (Part 1)



Hình 4. C4 Model - Level 3 (Component Diagram) - Infrastructure & External Systems

## Use-Case Diagram – LMS Mini



Hình 5. Use-case Diagram - LMS Mini

## Mô hình kiến trúc

Hệ thống LMS Mini áp dụng **Clean Architecture** nhằm tách biệt rõ ràng giữa các tầng, đảm bảo tính linh hoạt, dễ bảo trì và mở rộng. Các tầng trong kiến trúc bao gồm:

**Presentation Layer:**

* Vai trò: Cung cấp giao diện người dùng (Web, Mobile).
* Nhiệm vụ: Hiển thị dữ liệu, nhận input từ người dùng, và gửi yêu cầu đến Application Layer.
* Thành phần: Razor Components, Pages, Shared Layouts, client services, authentication UI, JS interop.

**Application Layer:**

* Vai trò: Điều phối luồng xử lý nghiệp vụ.
* Nhiệm vụ: Thực thi các Use Case, xử lý logic nghiệp vụ, xác thực (validation), phân quyền (authorization), và điều hướng luồng dữ liệu giữa Presentation Layer và Domain Layer.
* Thành phần: Use Cases (Commands/Queries), DTOs, Interfaces (Ports) như ICourseRepository, IEnrollmentRepository, IQuizService.

**Domain Layer:**

* Vai trò: Lõi nghiệp vụ của hệ thống.
* Nhiệm vụ: Quản lý các thực thể (Entities), giá trị (Value Objects), quy tắc nghiệp vụ (Business Rules), và sự kiện miền (Domain Events).
* Đặc điểm: Không phụ thuộc vào bất kỳ tầng nào khác, đảm bảo tính độc lập và tái sử dụng.

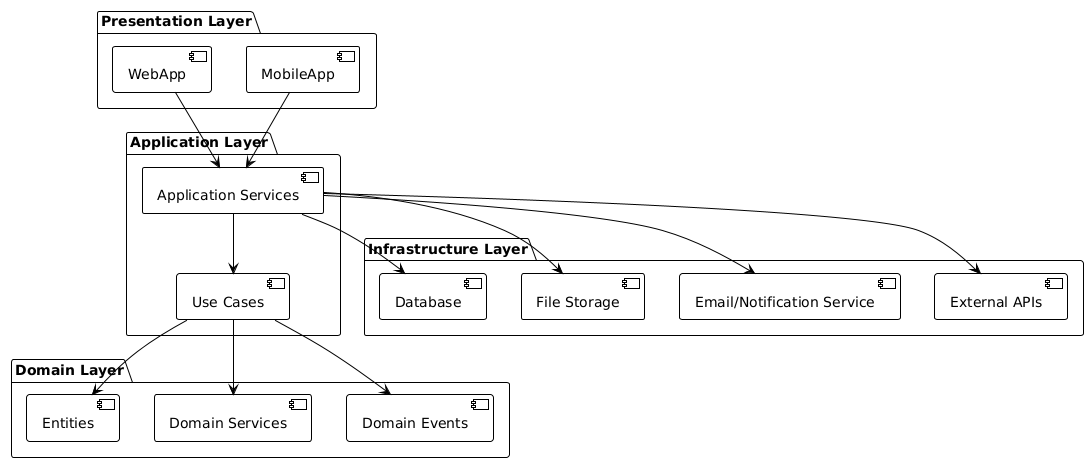
**Infrastructure Layer:**

* Vai trò: Hạ tầng kỹ thuật, hỗ trợ các tầng khác.
* Nhiệm vụ: Kết nối cơ sở dữ liệu, hệ thống file, dịch vụ thông báo, và tích hợp với các API bên ngoài.
* Thành phần: EF Core LmsDbContext, repositories, Identity Store, adapters (file/email), logging, caching.

**Điểm chính của kiến trúc:**

* Phụ thuộc một chiều: Các tầng phụ thuộc từ ngoài vào trong (Presentation → Application → Domain).
* Domain là trung tâm: Domain Layer không phụ thuộc vào bất kỳ tầng nào khác, đảm bảo tính độc lập và dễ bảo trì.

## Sơ đồ kiến trúc tổng thể



Hình 6. Sơ đồ kiến trúc tổng thể (Layer Diagram)

### Ví dụ: Tạo khóa học mới (Create Course)

* Người dùng nhập thông tin khóa học (mã, tiêu đề, mô tả) qua giao diện.
* Hệ thống kiểm tra các quy tắc nghiệp vụ (ví dụ: mã khóa học không được trùng lặp).
* Nếu hợp lệ, thông tin khóa học được lưu vào cơ sở dữ liệu.

Chi tiết từng tầng:

1. **Presentation Layer**: Người dùng nhập thông tin khóa học qua giao diện Blazor:
   * Razor Component: CreateCourse.razor
   * Nhiệm vụ: Hiển thị form và gửi yêu cầu đến Application Layer.
2. **Application Layer**: Application Layer nhận yêu cầu từ Presentation Layer:
   * Command: CreateCourseCommand
   * Handler: CreateCourseCommandHandler
   * Nhiệm vụ: Điều phối luồng nghiệp vụ và gọi Domain Layer.
3. **Domain Layer**: Domain Layer kiểm tra các quy tắc nghiệp vụ:
   * Entity: Course
   * Nhiệm vụ: Đảm bảo mã khóa học không được trống và tiêu đề hợp lệ và mã khóa học là duy nhất trong hệ thống.
4. **Infrastructure Layer:** Infrastructure Layer lưu trữ dữ liệu vào cơ sở dữ liệu:
   * Repository: CourseRepository
   * Nhiệm vụ: Thực hiện thao tác lưu trữ với EF Core.

|  |
| --- |
| **Tóm tắt luồng tương tác:**   1. Người dùng nhập thông tin khóa học và gửi yêu cầu qua giao diện. 2. Application Layer nhận yêu cầu, kiểm tra dữ liệu, và gọi Domain Layer. 3. Domain Layer kiểm tra các quy tắc nghiệp vụ và khởi tạo thực thể. 4. Infrastructure Layer lưu trữ thực thể vào cơ sở dữ liệu.     Hình 7. Sơ đồ Tạo khóa học |

### Ví dụ: Đăng ký khóa học (Enroll in Course courseId và userId)

**Mô tả Use Case**

* Người dùng chọn một khóa học và nhấn nút "Đăng ký".
* Hệ thống kiểm tra xem người dùng đã đăng ký khóa học này chưa.
* Nếu hợp lệ, thông tin đăng ký được lưu vào cơ sở dữ liệu.

**Chi tiết các tầng**

1. **Presentation Layer:**

* Thành phần: Razor Component EnrollCourse.razor.
* Nhiệm vụ: Hiển thị nút "Đăng ký" và gửi yêu cầu đến Application Layer.

|  |
| --- |
| using System.Security.Claims  @inject IMediator Mediator  <button @onclick="EnrollCourse">Enroll</button>  @code {  [Parameter] public Guid CourseId { get; set; } // từ context thực tế: route/parent component  [CascadingParameter] private Task<AuthenticationState> AuthenticationStateTask { get; set; } = default!; // user đăng nhập  private async Task EnrollCourse()  {  var authState = await AuthenticationStateTask;  var user = authState.User;  var userId = Guid.Parse(user.FindFirstValue(ClaimTypes.NameIdentifier)!); // từ context thực tế  var command = new EnrollCourseCommand(courseId: CourseId, userId: userId);  await Mediator.Send(command);  }  } |

1. **Application Layer:**

* Thành phần: EnrollCourseCommand, EnrollCourseCommandHandler.
* Nhiệm vụ: Kiểm tra tính hợp lệ và gọi Domain Layer để xử lý nghiệp vụ.

|  |
| --- |
| // Command để ghi danh người dùng vào khóa học  public record EnrollCourseCommand(Guid CourseId, Guid UserId) : IRequest;  public class EnrollCourseCommandHandler : IRequestHandler<EnrollCourseCommand>  {  private readonly IEnrollmentRepository \_enrollmentRepository;  public EnrollCourseCommandHandler(IEnrollmentRepository enrollmentRepository)  {  \_enrollmentRepository = enrollmentRepository;  }  public async Task<Unit> Handle(EnrollCourseCommand request, CancellationToken cancellationToken)  {  if (await \_enrollmentRepository.IsUserEnrolledAsync(request.UserId, request.CourseId))  throw new InvalidOperationException("User already enrolled.");  await \_enrollmentRepository.EnrollUserAsync(request.UserId, request.CourseId);  return Unit.Value;  }  } |

1. **Domain Layer:**

* Thành phần: Enrollment Entity.
* Nhiệm vụ: Đảm bảo người dùng không thể đăng ký trùng khóa học.

1. **Infrastructure Layer:**

* Thành phần: EnrollmentRepository.
* Nhiệm vụ: Lưu thông tin đăng ký vào cơ sở dữ liệu.

|  |
| --- |
| **Tóm tắt luồng tương tác**   1. Người dùng nhấn nút "Đăng ký" trên giao diện. 2. Application Layer kiểm tra xem người dùng đã đăng ký khóa học chưa. 3. Domain Layer đảm bảo các quy tắc nghiệp vụ. 4. Infrastructure Layer lưu thông tin đăng ký vào cơ sở dữ liệu.     Hình 8 Sơ đồ Đăng ký khóa học |

### Ví dụ: Nộp bài kiểm tra (Submit Quiz)

**Mô tả Use Case**

* Người dùng hoàn thành bài kiểm tra và nhấn nút "Nộp bài".
* Hệ thống kiểm tra tính hợp lệ của bài làm, tính điểm, và lưu kết quả vào cơ sở dữ liệu.
* Nếu bài kiểm tra đạt yêu cầu, tiến độ khóa học của người dùng được cập nhật.

**Chi tiết các tầng**

1. **Presentation Layer**

* Thành phần: Razor Component SubmitQuiz.razor.
* Nhiệm vụ: Hiển thị giao diện bài kiểm tra và gửi yêu cầu nộp bài đến Application Layer.

Ví dụ:

|  |
| --- |
| @inject IMediator Mediator  <button @onclick="SubmitQuiz">Submit Quiz</button>  @code {  private async Task SubmitQuiz()  {  var command = new SubmitQuizCommand(quizId: Guid.NewGuid(), userId: Guid.NewGuid(), answers: new List<AnswerDto>());  await Mediator.Send(command);  }  } |

1. **Application Layer**

* Thành phần: SubmitQuizCommand, SubmitQuizCommandHandler.
* Nhiệm vụ: Điều phối luồng nghiệp vụ, kiểm tra tính hợp lệ của bài làm, và gọi Domain Layer để tính điểm.

Ví dụ:

|  |
| --- |
| public record SubmitQuizCommand(Guid QuizId, Guid UserId, List<AnswerDto> Answers) : IRequest<SubmitQuizResultDto>;  public class SubmitQuizCommandHandler : IRequestHandler<SubmitQuizCommand, SubmitQuizResultDto>  {  private readonly IQuizRepository \_quizRepository;  private readonly IProgressRepository \_progressRepository;  public SubmitQuizCommandHandler(IQuizRepository quizRepository, IProgressRepository progressRepository)  {  \_quizRepository = quizRepository;  \_progressRepository = progressRepository;  }  public async Task<SubmitQuizResultDto> Handle(SubmitQuizCommand request, CancellationToken cancellationToken)  {  var quiz = await \_quizRepository.GetQuizWithAnswersAsync(request.QuizId);  var result = quiz.CalculateScore(request.Answers);  await \_quizRepository.SaveQuizAttemptAsync(request.UserId, request.QuizId, result.Score);  await \_progressRepository.UpdateProgressAsync(request.UserId, quiz.CourseId);  return new SubmitQuizResultDto(result.Score, result.Passed);  }  } |

1. **Domain Layer**

* Thành phần: Quiz Entity.
* Nhiệm vụ: Tính điểm bài kiểm tra và kiểm tra các quy tắc nghiệp vụ.

Ví dụ:

|  |
| --- |
| public class Quiz  {  public Guid Id { get; private set; }  public List<Question> Questions { get; private set; }  public QuizResult CalculateScore(List<AnswerDto> answers)  {  int correctAnswers = Questions.Count(q => answers.Any(a => a.QuestionId == q.Id && a.OptionId == q.CorrectOptionId));  double score = (double)correctAnswers / Questions.Count \* 100;  return new QuizResult(score, score >= 70); // Đạt nếu điểm >= 70%  }  } |

1. **Infrastructure Layer**

* Thành phần: QuizRepository, ProgressRepository.
* Nhiệm vụ: Lưu kết quả bài kiểm tra và cập nhật tiến độ khóa học.

|  |
| --- |
| **Tóm tắt luồng tương tác**   1. Người dùng nhấn nút "Nộp bài" trên giao diện. 2. Application Layer nhận yêu cầu, tính điểm bài kiểm tra, và lưu kết quả. 3. Domain Layer kiểm tra các quy tắc nghiệp vụ và tính toán điểm số. 4. Infrastructure Layer lưu kết quả và cập nhật tiến độ khóa học.     Hình 9. Class Diagram UML - Nộp Bài kiểm tra (Submit Quiz) |

## Vai trò, trách nhiệm và mối quan hệ

|  |  |  |  |
| --- | --- | --- | --- |
| Tầng/Thành phần | Vai trò | Trách nhiệm | Quan hệ |
| Presentation Layer | Giao diện người dùng | Hiển thị dữ liệu, nhận input | Gọi Application Services để thực thi các Use Case |
| Application Layer | Điều phối luồng xử lý | Thực thi Use Case, gọi Domain | Phụ thuộc vào Interface/Port được định nghĩa trong Domain Layer |
| Domain Layer | Lõi nghiệp vụ | Quản lý Entity, Business Rule, Event | Chỉ bị gọi bởi Application Layer, không phụ thuộc vào tầng khác |
| Infrastructure Layer | Hạ tầng kỹ thuật | Lưu trữ DB, File, Notification, tích hợp API | Phụ thuộc vào Interface/Port của Application Layer |

* Domain Layer
  + Chứa: Entities (Course, Module, Lesson, Quiz, Question, Option, Enrollment, Progress, QuizAttempt, AttemptAnswer, Notification, FileAsset), Value Objects, Domain Exceptions, Domain Services thuần nghiệp vụ.
  + Nhiệm vụ: Đảm bảo invariants, áp dụng quy tắc, thực hiện tính toán (ví dụ: tính điểm quiz, tiến độ).
  + Đặc điểm: Không phụ thuộc framework.
* Application Layer
  + Chứa: Use Cases (Commands/Queries), DTOs, Interfaces (Ports) như: ICourseRepository, IEnrollmentRepository, IQuizService, IFileStorage, IEmailSender, IDateTimeProvider, IUnitOfWork...
  + Kỹ thuật: sử dụng MediatR để điều phối.
  + Nhiệm vụ: Điều hướng luồng nghiệp vụ, validation (FluentValidation), phân quyền (policies), xác định transaction boundaries, mapping Domain ↔ DTO.
* Infrastructure Layer
  + Thành phần: EF Core LmsDbContext, repositories (EF), Identity Store (ASP.NET Core Identity), adapters (file/email), logging, caching.
  + Nhiệm vụ: Cấu hình persistence, triển khai interfaces của Application Layer, migrations, seed data.
* **Presentation Layer** (Blazor Server UI)
  + Thành phần: Razor Components, Pages, Shared Layouts, client services, authentication UI, JS interop.
  + Nhiệm vụ: Presentation, validation phía client, gọi Application Layer qua DI/MediatR trong server-side Blazor.

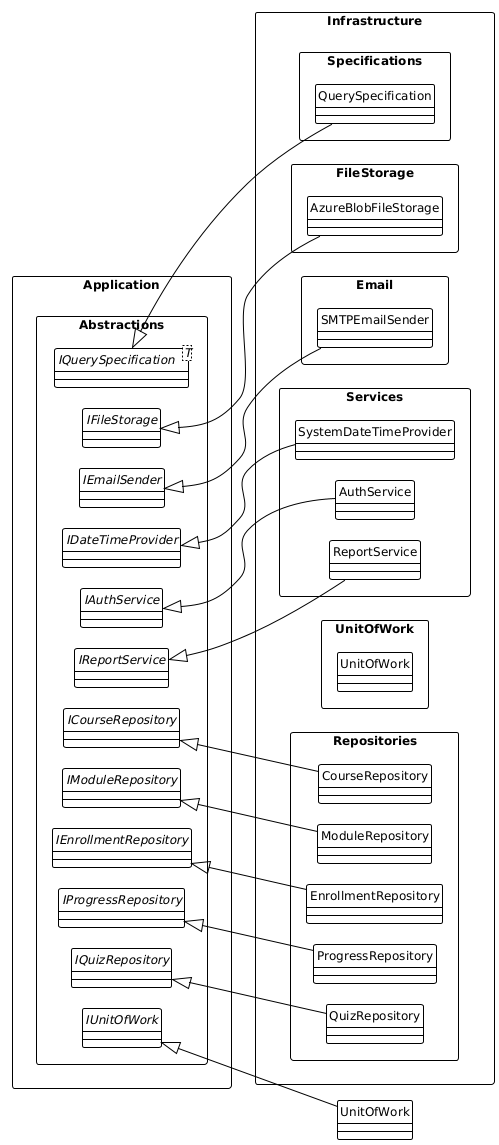
## Quy ước chung

* Audit: mọi entity kế thừa BaseAuditable với các trường: CreatedAt, CreatedBy, ModifiedAt, ModifiedBy.
* Soft Delete: dùng cờ IsDeleted = true thay vì xóa cứng, đảm bảo an toàn dữ liệu và truy vết lịch sử.
* Chuẩn hóa đặt tên:
  + Khóa chính mặc định là Id.
  + Khóa ngoại đặt theo dạng <EntityName>Id.
  + Tên bảng và entity viết theo PascalCase.

## Interface/Port chính và nơi triển khai

|  |  |  |  |
| --- | --- | --- | --- |
| Interface (Port) | Miêu tả | Định nghĩa tại | Triển khai tại |
| ICourseRepository | CRUD và truy vấn khóa học, hỗ trợ phân trang. | Application.Abstractions | Infrastructure.Repositories.CourseRepository |
| IModuleRepository | CRUD module và bài học. | Application.Abstractions | Infrastructure.Repositories.ModuleRepository |
| IEnrollmentRepository | Thao tác ghi danh học viên vào khóa học. | Application.Abstractions | Infrastructure.Repositories.EnrollmentRepository |
| IProgressRepository | Cập nhật và truy vấn tiến độ học tập. | Application.Abstractions | Infrastructure.Repositories.ProgressRepository |
| IQuizRepository | CRUD bài kiểm tra, câu hỏi, tùy chọn, và lượt làm bài. | Application.Abstractions | Infrastructure.Repositories.QuizRepository |
| IUnitOfWork | Định nghĩa ranh giới transaction. | Application.Abstractions | Infrastructure.UnitOfWork |
| IFileStorage | Upload, lấy URL có chữ ký, và xóa file. | Application.Abstractions | Infrastructure.FileStorage.AzureBlobFileStorage |
| IEmailSender | Gửi email thông báo. | Application.Abstractions | Infrastructure.Email.SMTPEmailSender |
| IDateTimeProvider | Lấy thời gian hệ thống (UTC). | Application.Abstractions | Infrastructure.Services.SystemDateTimeProvider |
| IAuthService | Kiểm tra vai trò và chính sách phân quyền. | Application.Abstractions | Infrastructure.Services.AuthService |
| IReportService | Truy vấn tổng hợp cho báo cáo. | Application.Abstractions | Infrastructure.Services.ReportService |
| IQuerySpecification<T> | Định nghĩa tiêu chí truy vấn dữ liệu bao gồm **Filters** (điều kiện lọc), **Sorting** (sắp xếp), **Includes** (nạp liên kết), và **Paging** (phân trang). Áp dụng để tách rời logic truy vấn khỏi repository và hỗ trợ tái sử dụng điều kiện. | Application.Abstractions | Infrastructure.Specifications |

**Hình ảnh mô tả:**



Hình 10. UML Interface–Implementation Diagram (PlantUML)

Note: Nên đặt interfaces trong Lms.Application.Abstractions (hoặc Lms.Application.Interfaces) để Domain Layer hoàn toàn độc lập.

## Domain Events, Outbox, Specification & Cross-Cutting Concerns

### Domain Events

* Mục tiêu: Giảm sự phụ thuộc trực tiếp giữa nghiệp vụ chính và các tác vụ bổ trợ như thống kê, thông báo, email, tích hợp ngoài; giúp hệ thống dễ bảo trì và mở rộng.
* Định nghĩa: Domain Event phản ánh thay đổi quan trọng trong domain. Chỉ được raise từ Aggregate Root sau khi validate nghiệp vụ và trước khi commit Unit of Work. Ví dụ: EnrollmentCreated khi người dùng ghi danh thành công.
* Quy tắc đặt tên & versioning:
* Tên: Động từ + quá khứ (PascalCase), ví dụ QuizSubmitted
* Payload thay đổi → tăng version trong metadata
* Payload chuẩn:

json

{

"eventId": "GUID",

"eventType": "QuizSubmitted",

"version": 1,

"occurredOn": "2025-08-20T05:30:00Z",

"correlationId": "GUID",

"tenantId": "GUID",

"causedByUserId": "GUID",

"data": { ... }

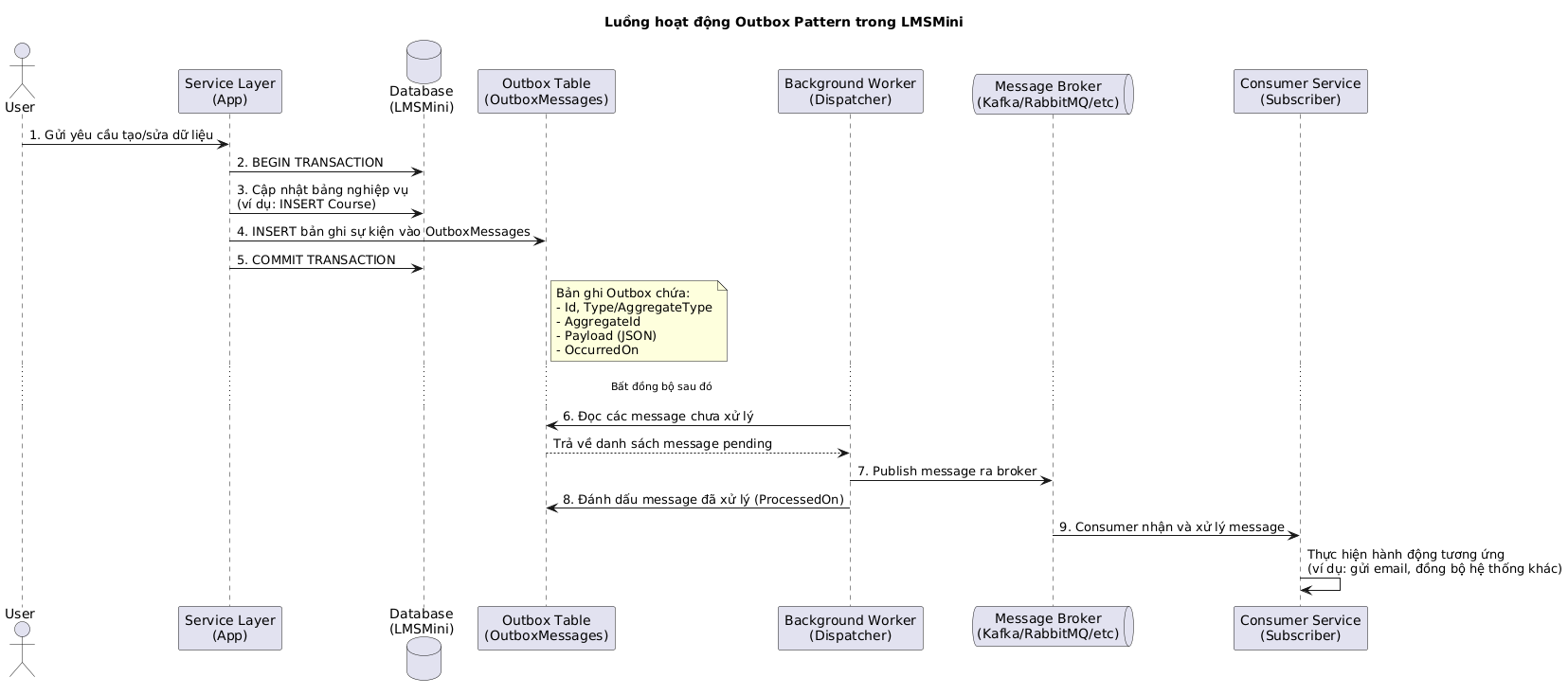
}

* Lifecycle: Raised → Persisted → Published → Consumed → Archived
* Idempotency & xử lý lỗi: Handler phải idempotent, lỗi được log và retry theo chính sách
* Event Catalog: Có phụ lục liệt kê tất cả event với phiên bản mới nhất, mô tả, trường bắt buộc
* Replay: Có thể lưu tạm để replay khi cần debug hoặc rebuild state

### Outbox Pattern

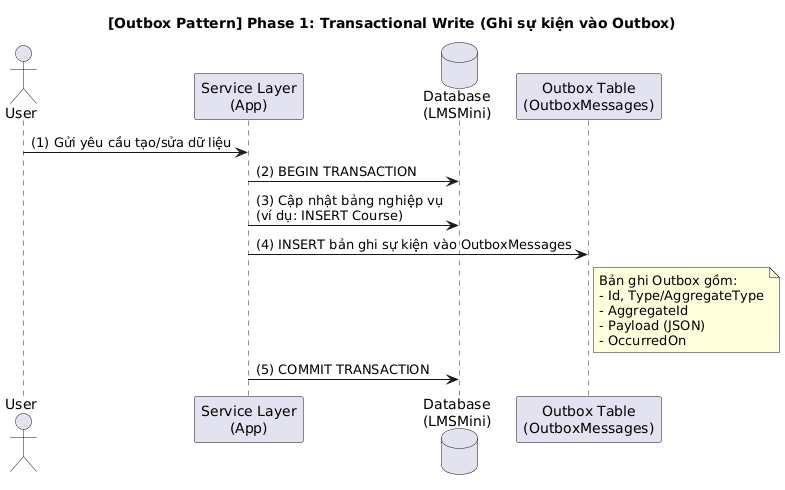
* **Mục tiêu:** Đảm bảo tính nhất quán giữa cơ sở dữ liệu giao dịch và message broker
* **Nguyên tắc**: Khi commit transaction nghiệp vụ, đồng thời ghi event vào bảng OutboxMessages
* **Cấu trúc bảng OutboxMessages:**
  + Id (GUID, PK)
  + OccurredOn (UTC)
  + Type (string)
  + Payload (JSON)
  + ProcessedOn (UTC, nullable)
  + Error (text, nullable)
  + RetryCount (int)
* **Vận hành:**
  + Persist event vào Outbox trong cùng transaction
  + Background processor đọc bản ghi chưa xử lý
  + Publish tới handler hoặc broker
  + Retry khi lỗi, log error, tăng RetryCount; set ProcessedOn khi thành công
* **Retention Policy**: Xoá hoặc archive Outbox đã xử lý sau X ngày
* **Bảo mật**: Mask hoặc encrypt PII trong payload
* **Monitoring**: Cảnh báo khi RetryCount vượt ngưỡng hoặc backlog tăng bất thường

#### Sơ đồ tổng quát Outbox Pattern

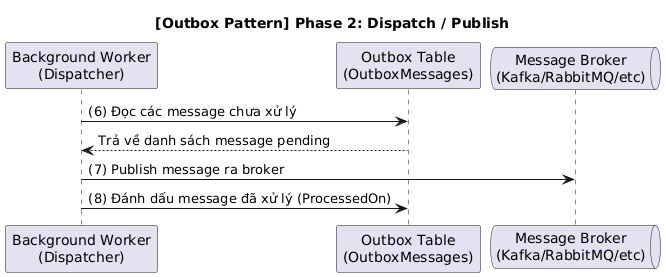


Hình 11. Luồng hoạt động Outbox Pattern trong LMSMini

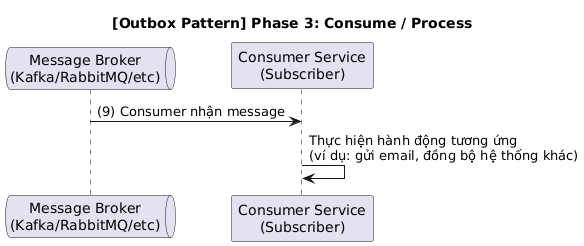
#### Sơ đồ chi tiết Outbox Pattern (theo giai đoạn)



Hình 12. [Outbox Pattern] Giai đoạn 1: Ghi sự kiện vào OutboxMessages



Hình 13. [Outbox Pattern] Giai đoạn 2: Dispatch / Publish



Hình 14. [Outbox Pattern] Giai đoạn 3: Consume/ Process

### ****Specification Pattern****

#### ****Mục tiêu****

Sử dụng **Specification Pattern** nhằm:

* Giảm số lượng phương thức đặc thù trong Repository (tránh “phình to” repository với hàng loạt hàm FindByX, GetByY…).
* Tái sử dụng và kết hợp được các truy vấn phức tạp.
* Chuẩn hóa cách định nghĩa điều kiện lọc, sắp xếp, include quan hệ, và phân trang trong hệ thống.

#### ****Triển khai trong LMS‑Mini****

1. **Định nghĩa interface** IQuerySpecification<T>

* Nằm tại namespace Lms.Application.Abstractions.
* Chứa các thành phần mô tả truy vấn:
  + Criteria: điều kiện lọc (Expression<Func<T, bool>>)
  + OrderBy/OrderByDescending: sắp xếp
  + Includes: danh sách navigation properties cần load
  + Skip / Take: phân trang

1. **Repository áp dụng Specification**

* Các repository (ví dụ: ICourseRepository, IEnrollmentRepository) nhận vào một IQuerySpecification<T> để dựng truy vấn (compose filter/sort/include/paging).

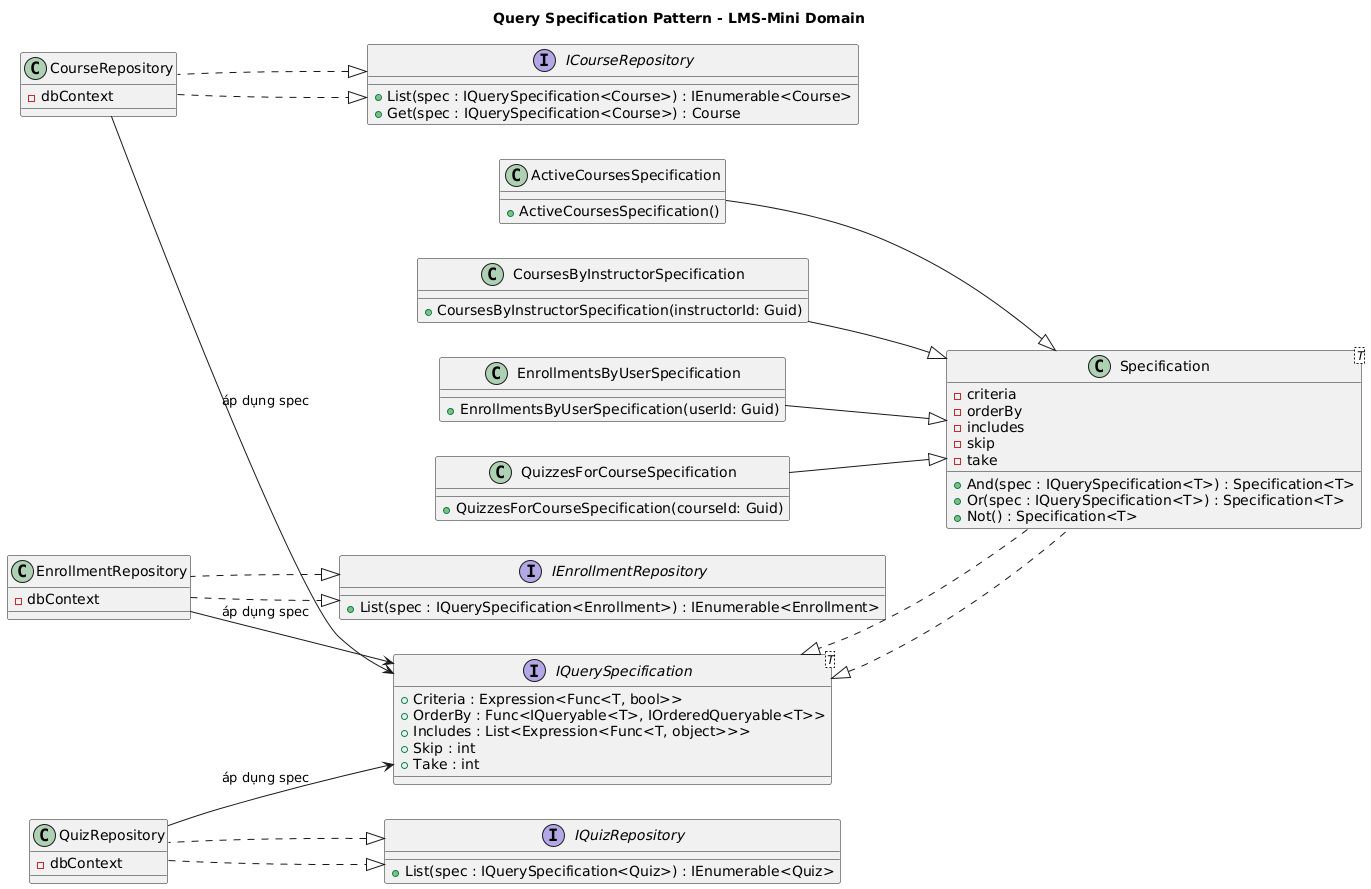
1. **Kết hợp nhiều Specification**
   * Cung cấp khả năng **AND**, **OR**, **NOT** giữa các specification để tạo điều kiện lọc phức hợp mà vẫn giữ tính rõ ràng và có thể kiểm thử.
2. **Phạm vi sử dụng trong LMS‑Mini**
   * Truy vấn khóa học đang hoạt động (ActiveCoursesSpecification)
   * Lấy danh sách khóa học theo giảng viên (CoursesByInstructorSpecification)
   * Lấy danh sách ghi danh theo người học (EnrollmentsByUserSpecification)
   * Lọc quiz thuộc một khóa học (QuizzesForCourseSpecification)

#### ****Lợi ích đạt được****

* **Tách biệt**: Logic lọc, sắp xếp, paging được tách khỏi tầng hạ tầng (Infrastructure).
* **Tái sử dụng**: Một specification có thể được dùng lại ở nhiều nơi trong hệ thống.
* **Dễ kiểm thử**: Specification thuần C#, có thể unit test độc lập không cần database thật.
* **Hiệu năng**: Có thể cache kết quả cho những specification nặng.

#### ****Sơ đồ tổng quát – Specification Pattern****

**Hình 15** minh họa mô hình tổng quát của Specification Pattern:

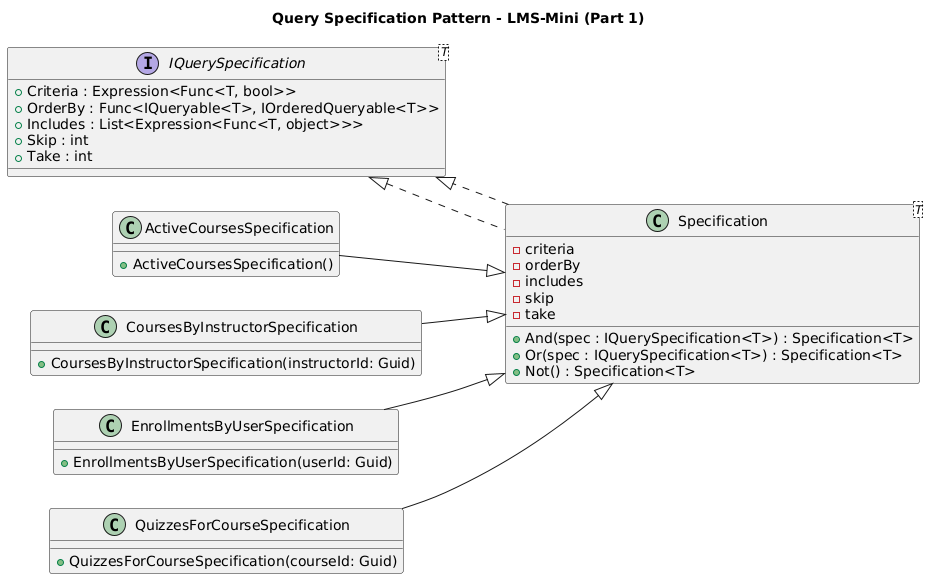


Hình 15. Minh họa mô hình tổng quát của Specification Pattern

* IQuerySpecification<T>: interface định nghĩa cấu trúc đặc tả truy vấn.
* Specification<T>: lớp cơ sở cài đặt sẵn các cơ chế kết hợp (AND/OR/NOT).
* **Các lớp Specification cụ thể**: triển khai các điều kiện nghiệp vụ cụ thể.
* **Repository**: nhận một specification và áp dụng vào truy vấn (thường với EF Core hoặc ORM khác).

**Ý nghĩa sơ đồ:** Cho thấy luồng từ việc định nghĩa điều kiện nghiệp vụ ở dạng object (Specification) → Repository áp dụng để sinh truy vấn → trả kết quả, mà không cần viết hàm truy vấn riêng cho từng tình huống.

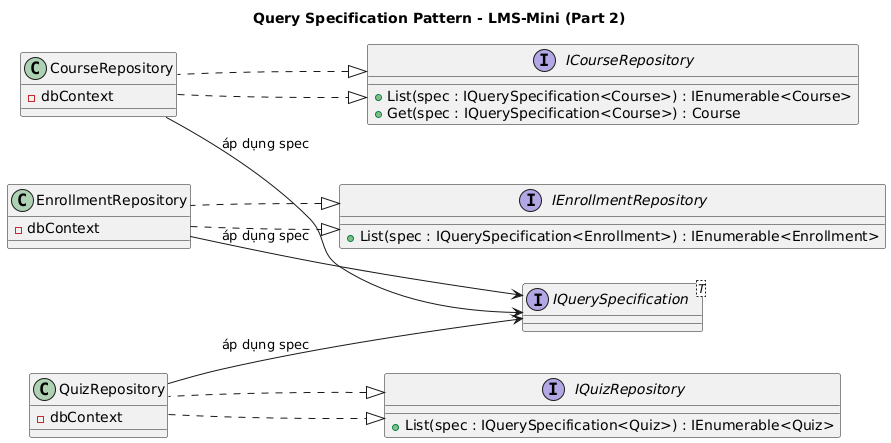
#### ****Sơ đồ triển khai tách hai phần – Query Specification Pattern****

Để trình bày rõ hơn, Hình 15 được **tách thành hai sơ đồ con**: 

Hình 16. Part 1: Minh họa mối quan hệ ActiveCoursesSpecification

**Hình 16 – Part 1:** Minh họa mối quan hệ giữa:

* Interface IQuerySpecification<T>
* Lớp cơ sở Specification<T>
* Các Specification nghiệp vụ cụ thể của LMS‑Mini (ActiveCoursesSpecification, CoursesByInstructorSpecification, …) → Giúp thấy cách pattern được chuyên biệt hóa cho domain này.



Hình 17. Part2: Minh họa mối quan hệ CoursesByInstructorSpecification

**Hình 17 – Part 2:** Minh họa mối quan hệ giữa:

* Các interface repository (ICourseRepository, IEnrollmentRepository, …)
* Các lớp triển khai repository trong Infrastructure.
* Kết nối từ repository đến IQuerySpecification<T> → Cho thấy Specification Pattern **được tiêu thụ ở đâu** trong kiến trúc, và **luồng áp dụng** từ Application Layer xuống Infrastructure.

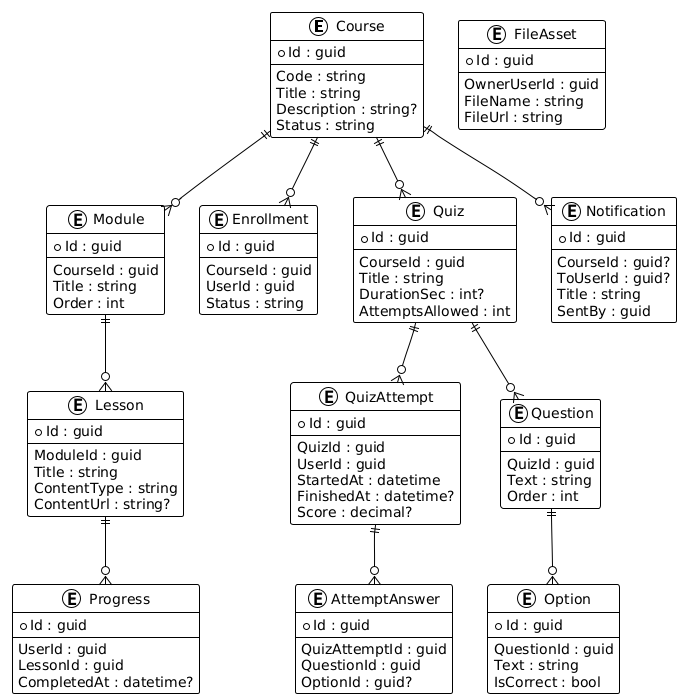
### Timezone / Clock

* Lưu tất cả thời gian ở UTC (IDateTimeProvider.UtcNow)
* Hiển thị timezone do UI quyết định dựa trên cấu hình người dùng
* API dùng định dạng ISO 8601 UTC
* Cho phép mock Clock khi test
* Nếu triển khai quốc tế, quy định rõ cách xử lý DST

### Audit Log

* Mục tiêu: Truy vết mọi hành động, hỗ trợ bảo mật, tuân thủ, phân tích sự cố
* AuditEntry gồm: id, action, entityType, entityId, performedByUserId, performedOnUtc, details, correlationId, tenantId
* Nguồn kích hoạt: Lắng nghe Domain Events hoặc interceptor ở application layer
* Lưu trữ: Bảng riêng hoặc hệ thống log tập trung
* Bảo mật: Phân quyền truy cập audit log

# Thiết kế cơ sở dữ liệu



Hình 18. Entity Diagram - LMS

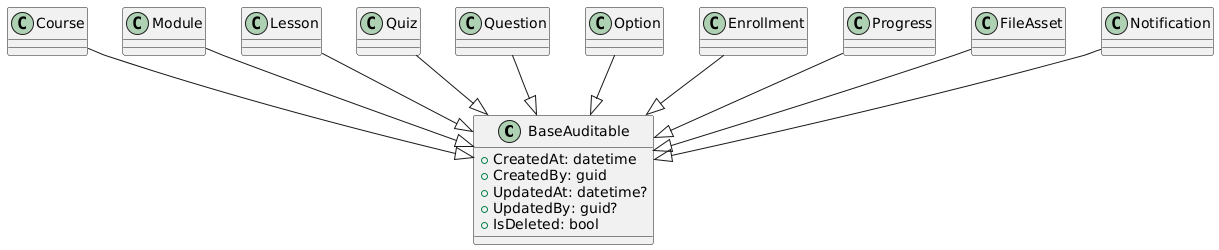
## Audit & Soft Delete (Chuẩn hóa Audit & Xóa mềm)

### Mục đích

Đảm bảo các bảng nghiệp vụ quan trọng trong hệ thống LMS có khả năng truy vết (audit) và xóa mềm (soft delete) để phục vụ: điều tra sự cố, bảo trì, tuân thủ pháp lý và cho phép phục hồi/tái tạo bản ghi khi cần.

Bảng 1 tổng hợp (rõ entity nào áp dụng audit)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Entity | CreatedAt | CreatedBy | UpdatedAt | UpdatedBy | IsDeleted |
| Course | ✅ | ✅ | ✅ | ✅ | ✅ |
| Module | ✅ | ✅ | ✅ | ✅ | ✅ |
| Lesson | ✅ | ✅ | ✅ | ✅ | ✅ |
| Quiz | ✅ | ✅ | ✅ | ✅ | ✅ |
| Question | ✅ | ✅ | ✅ | ✅ | ✅ |
| Option | ✅ | ✅ | ✅ | ✅ | ✅ |
| Enrollment | ✅ | ✅ | ✅ | ✅ | ✅ |
| Progresses | ✅ | ✅ | ✅ | ✅ | ✅ |
| FileAsset | ✅ | ✅ | ✅ | ✅ | ✅ |
| Notification | ✅ | ✅ | ✅ | ✅ | ✅ |



Hình 19. BaseAuditable & Soft Delete

### Quy tắc đặt tên & kiểu dữ liệu

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Bắt buộc | Ý nghĩa |
| CreatedAt | DATETIME2 (UTC) | ✔ | Thời điểm bản ghi được tạo |
| CreatedBy | UNIQUEIDENTIFIER | ✔ | ID người tạo (hoặc SYSTEM nếu tự động) |
| UpdatedAt | DATETIME2 (UTC) | ✔ | Thời điểm cập nhật gần nhất (null nếu chưa) |
| UpdatedBy | UNIQUEIDENTIFIER | ✔ | ID người cập nhật (null nếu chưa) |
| DeletedAt | DATETIME2 (UTC) | Tuỳ chọn | Thời điểm xóa mềm |
| DeletedBy | UNIQUEIDENTIFIER | Tuỳ chọn | ID người xóa |
| IsDeleted | BIT | ✔ | Cờ trạng thái xóa mềm (1 = đã xóa, 0 = còn hiệu lực) |

* Quy ước đặt tên: hậu tố At cho thời gian, hậu tố By cho tác nhân; boolean tiền tố Is.
* Timezone: lưu tất cả thời gian ở UTC (ví dụ SYSUTCDATETIME() trên SQL Server).
* Không lấy giá trị thời gian/tác nhân từ client — thời gian/tác nhân do hệ thống (IDateTimeProvider / CurrentUserService) cung cấp.

### Phạm vi áp dụng

* Bắt buộc áp dụng: Courses, Modules, Lessons, Quizzes, Questions, Options, FileAssets, Notifications, Enrollments.
* Áp dụng khi cần audit: Progresses (tùy yêu cầu pháp lý/kinh doanh).
* Không áp dụng (mặc định): bảng ghi nhận giao dịch tức thời như QuizAttempts, AttemptAnswers — chỉ áp dụng nếu có yêu cầu tuân thủ đặc thù.

Ghi chú: danh sách entity áp dụng phải được đồng bộ với ERD, DDL và mapping EF Core trong phụ lục.

### Nguyên tắc xóa mềm (Soft Delete semantics)

* Xóa mềm = set IsDeleted = 1 và gán DeletedAt, DeletedBy.
* Không xóa vật lý khỏi DB trừ khi có quy trình sao lưu / dữ liệu hết giá trị theo chính sách dữ liệu.
* ORM: áp dụng Global Query Filter mặc định để ẩn entity đã xóa (e => !e.IsDeleted).
* Với unique constraint/index có khả năng xung đột, sử dụng filtered index hoặc unique constraint có điều kiện WHERE IsDeleted = 0 để cho phép tái tạo bản ghi (ví dụ: tái tạo Course.Code hoặc ghi danh lại (re-enroll) user sau xóa mềm).

Ví dụ EF Core Global Query Filter (mẫu):

modelBuilder.Entity<Course>().HasQueryFilter(e => !e.IsDeleted);

modelBuilder.Entity<Module>().HasQueryFilter(e => !e.IsDeleted);

modelBuilder.Entity<Lesson>().HasQueryFilter(e => !e.IsDeleted);

modelBuilder.Entity<Quiz>().HasQueryFilter(e => !e.IsDeleted);

modelBuilder.Entity<Question>().HasQueryFilter(e => !e.IsDeleted);

modelBuilder.Entity<Option>().HasQueryFilter(e => !e.IsDeleted);

modelBuilder.Entity<FileAsset>().HasQueryFilter(e => !e.IsDeleted);

modelBuilder.Entity<Notification>().HasQueryFilter(e => !e.IsDeleted);

modelBuilder.Entity<Enrollment>().HasQueryFilter(e => !e.IsDeleted);

### Ghi nhận audit tự động (EF Interceptor / middleware)

* Insert: CreatedAt = UtcNow, CreatedBy = CurrentUserId.
* Update: UpdatedAt = UtcNow, UpdatedBy = CurrentUserId.
* Thực thi bằng EF Interceptor / DbContext SaveChanges override hoặc middleware; không chấp nhận thời gian/tác nhân từ client request.

**Mẫu interface & interceptor (tham khảo):**

public interface IAuditable

{

DateTime CreatedAt { get; set; }

Guid? CreatedBy { get; set; }

DateTime? UpdatedAt { get; set; }

Guid? UpdatedBy { get; set; }

}

public override int SaveChanges()

{

var now = \_clock.UtcNow;

var userId = \_currentUser.Id;

foreach (var e in ChangeTracker.Entries<IAuditable>())

{

if (e.State == EntityState.Added)

{

e.Entity.CreatedAt = now;

e.Entity.CreatedBy = userId;

}

if (e.State == EntityState.Modified)

{

e.Entity.UpdatedAt = now;

e.Entity.UpdatedBy = userId;

}

}

return base.SaveChanges();

}

**Base type mẫu:**

public abstract class BaseAuditable {

public Guid CreatedBy { get; set; }

public DateTime CreatedAt { get; set; }

public Guid? UpdatedBy { get; set; }

public DateTime? UpdatedAt { get; set; }

public bool IsDeleted { get; set; }

}

public class Course : BaseAuditable {

public Guid Id { get; set; }

public string Title { get; set; }

}

### Indexes & Unique constraints — tương tác với Soft Delete

* Khi cần unique constraint (ví dụ UQ\_Courses\_Code, UQ\_Enrollments\_CourseId\_UserId), cân nhắc tạo filtered unique index: WHERE IsDeleted = 0 để tránh xung đột khi bản ghi cũ bị soft-deleted.
* Với chỉ mục dùng cho báo cáo (composite index), đảm bảo kế hoạch index phản ánh truy vấn thực tế và filter IsDeleted nếu cần.
* Trong DDL sample (Phụ lục A), thêm chú thích/DDL cho các filtered index khi áp dụng trên SQL Server.

### Concurrency (tùy chọn)

* Dùng Optimistic Concurrency (ROWVERSION) cho các bảng thay đổi thường xuyên (Courses, Lessons, Questions, Options, Modules, Quizzes).
* Đặt RowVersion làm concurrency token trong EF Core và xử lý xung đột trong tầng Application (trả 409 ERR\_CONFLICT theo hợp đồng API khi cần).

**Mẫu cấu hình RowVersion:**

public byte[] RowVersion { get; set; }

builder.Property(e => e.RowVersion)

.IsRowVersion()

.IsConcurrencyToken();

### Best practices & vận hành

* Lưu UTC để tránh sai lệch múi giờ; UI chịu trách nhiệm chuyển đổi theo cấu hình user.
* Không để CreatedBy/UpdatedBy rỗng cho các bảng nghiệp vụ chính.
* Gom trường audit vào BaseAuditable để tái sử dụng.
* Viết unit/integration test cho audit interceptor, soft-delete behavior, và các query phụ thuộc IsDeleted.
* Khi thay đổi schema audit/soft-delete, cập nhật migration, test plan và tài liệu phụ lục (DDL + EF mapping).
* Document rõ các entity không áp dụng soft-delete (ví dụ QuizAttempts, AttemptAnswers) để tránh sai sót khi implement.

### Tương thích với toàn bộ SDD

* Phần DDL / EF mapping trong Phụ lục cần phản ánh chính xác các trường audit và filtered index (nếu dùng).
* Tất cả repository/queries trong Application/Infrastructure phải tuân theo global filter hoặc explicit filter IsDeleted = 0 khi cần trả cả bản ghi đã xóa.
* Cập nhật API contract (ví dụ: DELETE semantics) để nêu rõ: 204 cho soft-delete; endpoint có thể hỗ trợ force=true để hard-delete (chỉ admin & theo chính sách).

## Mô hình logic — Bảng, PK, FK, Unique, Index

### AspNetUsers (Identity)

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| UserName | NVARCHAR(256) | NOT NULL |  |
| NormalizedUserName | NVARCHAR(256) | NOT NULL, UQ | UQ\_AspNetUsers\_NormalizedUserName |
| Email | NVARCHAR(256) | NULL |  |
| NormalizedEmail | NVARCHAR(256) | NULL |  |
| EmailConfirmed | BIT | NOT NULL DEFAULT(0) |  |
| PasswordHash | NVARCHAR(MAX) | NULL |  |
| SecurityStamp | NVARCHAR(MAX) | NULL |  |
| ConcurrencyStamp | NVARCHAR(MAX) | NULL |  |
| PhoneNumber | NVARCHAR(50) | NULL |  |
| PhoneNumberConfirmed | BIT | NOT NULL DEFAULT(0) |  |
| TwoFactorEnabled | BIT | NOT NULL DEFAULT(0) |  |
| LockoutEnd | DATETIMEOFFSET | NULL |  |
| LockoutEnabled | BIT | NOT NULL DEFAULT(0) |  |
| AccessFailedCount | INT | NOT NULL DEFAULT(0) |  |

**Khóa & Indexes:**

* PK: PK\_AspNetUsers (Id)
* UQ: UQ\_AspNetUsers\_NormalizedUserName (NormalizedUserName)
* IX: IX\_AspNetUsers\_NormalizedEmail (NormalizedEmail), IX\_AspNetUsers\_UserName (UserName)

### Courses

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| Code | NVARCHAR(50) | NOT NULL, UQ | UQ\_Courses\_Code |
| Title | NVARCHAR(200) | NOT NULL |  |
| Description | NVARCHAR(MAX) | NULL |  |
| Status | NVARCHAR(20) | NOT NULL DEFAULT 'Draft' | DF\_Courses\_Status |
| CreatedBy | UNIQUEIDENTIFIER | NOT NULL, FK → AspNetUsers.Id |  |
| CreatedAt | DATETIME2 | NOT NULL DEFAULT SYSUTCDATETIME() | DF\_Courses\_CreatedAt |
| IsDeleted | BIT | NOT NULL DEFAULT 0 | DF\_Courses\_IsDeleted |

**Khóa & Indexes:**

* PK: PK\_Courses (Id)
* FK: FK\_Courses\_AspNetUsers (CreatedBy → AspNetUsers.Id)
* IX: IX\_Courses\_Status (Status), IX\_Courses\_Title (Title)

### Modules

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| CourseId | UNIQUEIDENTIFIER | NOT NULL, FK → Courses.Id |  |
| Title | NVARCHAR(200) | NOT NULL |  |
| Order | INT | NOT NULL |  |
| IsDeleted | BIT | NOT NULL DEFAULT 0 | DF\_Modules\_IsDeleted |

**Khóa & Indexes:**

* PK: PK\_Modules (Id)
* FK: FK\_Modules\_Courses (CourseId → Courses.Id)
* IX: IX\_Modules\_Course\_Order (CourseId, Order)

### Lessons

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| ModuleId | UNIQUEIDENTIFIER | NOT NULL, FK → Modules.Id |  |
| Title | NVARCHAR(200) | NOT NULL |  |
| ContentType | NVARCHAR(50) | NOT NULL |  |
| ContentUrl | NVARCHAR(1000) | NULL |  |
| DurationSec | INT | NULL |  |
| Order | INT | NOT NULL |  |
| IsDeleted | BIT | NOT NULL DEFAULT 0 | DF\_Lessons\_IsDeleted |

**Khóa & Indexes:**

* PK: PK\_Lessons (Id)
* FK: FK\_Lessons\_Modules (ModuleId → Modules.Id)
* IX: IX\_Lessons\_Module\_Order (ModuleId, Order)

### Enrollments

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| CourseId | UNIQUEIDENTIFIER | NOT NULL, FK → Courses.Id |  |
| UserId | UNIQUEIDENTIFIER | NOT NULL, FK → AspNetUsers.Id |  |
| StartAt | DATETIME2 | NOT NULL |  |
| EndAt | DATETIME2 | NULL |  |
| Status | NVARCHAR(20) | NOT NULL |  |

**Khóa & Indexes:**

* PK: PK\_Enrollments (Id)
* UQ: UQ\_Enrollments\_CourseId\_UserId (CourseId, UserId)
* IX: IX\_Enrollments\_User (UserId), IX\_Enrollments\_Course\_Status (CourseId, Status)

### Progresses

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| UserId | UNIQUEIDENTIFIER | NOT NULL, FK → AspNetUsers.Id |  |
| LessonId | UNIQUEIDENTIFIER | NOT NULL, FK → Lessons.Id |  |
| CompletedAt | DATETIME2 | NULL |  |
| Percent | TINYINT | NOT NULL |  |

**Khóa & Indexes:**

* PK: PK\_Progresses (Id)
* UQ: UQ\_Progresses\_UserId\_LessonId (UserId, LessonId)
* IX: IX\_Progresses\_User (UserId)

### Quizzes

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| CourseId | UNIQUEIDENTIFIER | NOT NULL, FK → Courses.Id |  |
| Title | NVARCHAR(200) | NOT NULL |  |
| DurationSec | INT | NULL |  |
| AttemptsAllowed | INT | NULL DEFAULT 1 | DF\_Quizzes\_AttemptsAllowed |
| ShuffleAnswers | BIT | NOT NULL |  |

**Khóa & Indexes:**

* PK: PK\_Quizzes (Id)
* FK: FK\_Quizzes\_Courses (CourseId → Courses.Id)
* IX: IX\_Quizzes\_Course (CourseId)

### Questions

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| QuizId | UNIQUEIDENTIFIER | NOT NULL, FK → Quizzes.Id |  |
| Text | NVARCHAR(MAX) | NOT NULL |  |
| Order | INT | NOT NULL |  |
| IsDeleted | BIT | NOT NULL DEFAULT 0 | DF\_Questions\_IsDeleted |

**Khóa & Indexes:**

* PK: PK\_Questions (Id)
* FK: FK\_Questions\_Quizzes (QuizId → Quizzes.Id)
* IX: IX\_Questions\_Quiz\_Order (QuizId, Order)

### Options

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| QuestionId | UNIQUEIDENTIFIER | NOT NULL, FK → Questions.Id |  |
| Text | NVARCHAR(MAX) | NOT NULL |  |
| IsCorrect | BIT | NOT NULL |  |

**Khóa & Indexes:**

* PK: PK\_Options (Id)
* FK: FK\_Options\_Questions (QuestionId → Questions.Id)
* IX: IX\_Options\_Question (QuestionId)

### QuizAttempts

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| QuizId | UNIQUEIDENTIFIER | NOT NULL, FK → Quizzes.Id |  |
| UserId | UNIQUEIDENTIFIER | NOT NULL, FK → AspNetUsers.Id |  |
| StartedAt | DATETIME2 | NOT NULL |  |
| SubmittedAt | DATETIME2 | NULL |  |
| Score | DECIMAL(5,2) | NULL DEFAULT 0 | DF\_QuizAttempts\_Score |

**Khóa & Indexes:**

* PK: PK\_QuizAttempts (Id)
* FK: FK\_QuizAttempts\_Quizzes (QuizId → Quizzes.Id), FK\_QuizAttempts\_AspNetUsers (UserId → AspNetUsers.Id)
* IX: IX\_QuizAttempts\_Quiz\_User (QuizId, UserId)

### AttemptAnswers

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| AttemptId | UNIQUEIDENTIFIER | NOT NULL, FK → QuizAttempts.Id |  |
| QuestionId | UNIQUEIDENTIFIER | NOT NULL, FK → Questions.Id |  |
| OptionId | UNIQUEIDENTIFIER | NOT NULL, FK → Options.Id |  |

**Khóa & Indexes:**

* PK: PK\_AttemptAnswers (Id)
* UQ: UQ\_AttemptAnswers\_AttemptId\_QuestionId (AttemptId, QuestionId)
* FK: FK\_AttemptAnswers\_QuizAttempts (AttemptId → QuizAttempts.Id), FK\_AttemptAnswers\_Questions (QuestionId → Questions.Id), FK\_AttemptAnswers\_Options (OptionId → Options.Id)

### Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| CourseId | UNIQUEIDENTIFIER | NULL, FK → Courses.Id |  |
| ToUserId | UNIQUEIDENTIFIER | NULL, FK → AspNetUsers.Id |  |
| Title | NVARCHAR(200) | NOT NULL |  |
| Body | NVARCHAR(MAX) | NULL |  |
| CreatedAt | DATETIME2 | NOT NULL DEFAULT SYSUTCDATETIME() | DF\_Notifications\_CreatedAt |
| SentBy | UNIQUEIDENTIFIER | NOT NULL, FK → AspNetUsers.Id |  |

**Khóa & Indexes:**

* PK: PK\_Notifications (Id)
* FK: FK\_Notifications\_Courses (CourseId → Courses.Id), FK\_Notifications\_AspNetUsers\_ToUser (ToUserId → AspNetUsers.Id), FK\_Notifications\_AspNetUsers\_SentBy (SentBy → AspNetUsers.Id)
* IX: IX\_Notifications\_ToUserId (ToUserId)

### FileAssets

|  |  |  |  |
| --- | --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc | Ghi chú |
| Id | UNIQUEIDENTIFIER | NOT NULL, PK |  |
| OwnerUserId | UNIQUEIDENTIFIER | NOT NULL, FK → AspNetUsers.Id |  |
| FileName | NVARCHAR(255) | NOT NULL |  |
| MimeType | NVARCHAR(100) | NOT NULL |  |
| Size | BIGINT | NOT NULL |  |
| StoragePath | NVARCHAR(1000) | NOT NULL |  |
| CreatedAt | DATETIME2 | NOT NULL DEFAULT SYSUTCDATETIME() | DF\_FileAssets\_CreatedAt |

**Khóa & Indexes:**

* PK: PK\_FileAssets (Id)
* FK: FK\_FileAssets\_AspNetUsers (OwnerUserId → AspNetUsers.Id)
* IX: IX\_FileAssets\_OwnerUserId (OwnerUserId)

### AuditLogs

**Mục đích:** Lưu tập trung toàn bộ nhật ký thao tác trong hệ thống, phục vụ bảo mật, tuân thủ và truy vết sự cố.

|  |  |  |
| --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc / Mô tả |
| Id | BIGINT (PK) | Khóa chính, tự tăng |
| EntityName | NVARCHAR(100) | Tên bảng/thực thể tác động |
| EntityId | BIGINT | Khóa hoặc định danh bản ghi |
| Action | NVARCHAR(50) | CREATE / UPDATE / DELETE / LOGIN… |
| OldValues | NVARCHAR(MAX) | Dữ liệu trước thay đổi (JSON) |
| NewValues | NVARCHAR(MAX) | Dữ liệu sau thay đổi (JSON) |
| PerformedBy | NVARCHAR(100) | UserName hoặc UserId |
| PerformedAt | DATETIMEOFFSET | Thời gian thao tác |
| IpAddress | NVARCHAR(45) | IPv4/IPv6 |
| UserAgent | NVARCHAR(500) | Thông tin trình duyệt / client |

**Bảo mật:** Chỉ nhóm quản trị hoặc auditor có quyền đọc. Dữ liệu có thể được mã hoá ở mức cột nếu chứa thông tin nhạy cảm.

### OutboxMessages

**Mục đích:** Triển khai Outbox Pattern, đảm bảo nhất quán giữa DB và message broker.

|  |  |  |
| --- | --- | --- |
| Cột | Kiểu dữ liệu | Ràng buộc / Mô tả |
| Id | BIGINT (PK) | Khóa chính, tự tăng |
| AggregateType | NVARCHAR(100) | Loại aggregate (vd: Order, Payment…) |
| AggregateId | BIGINT | ID bản ghi liên quan |
| Payload | NVARCHAR(MAX) | Nội dung sự kiện (JSON) |
| OccurredAt | DATETIMEOFFSET | Thời điểm phát sinh sự kiện |
| ProcessedAt | DATETIMEOFFSET | Thời điểm gửi thành công (nullable) |
| Status | NVARCHAR(50) | Pending / Processed / Failed |
| RetryCount | INT | Số lần retry gửi message |
| ErrorMessage | NVARCHAR(1000) | Lỗi gần nhất (nullable) |

**Xử lý nền:** Service/worker sẽ đọc các bản ghi Status = 'Pending', gửi sang broker, đánh dấu ProcessedAt + cập nhật Status.

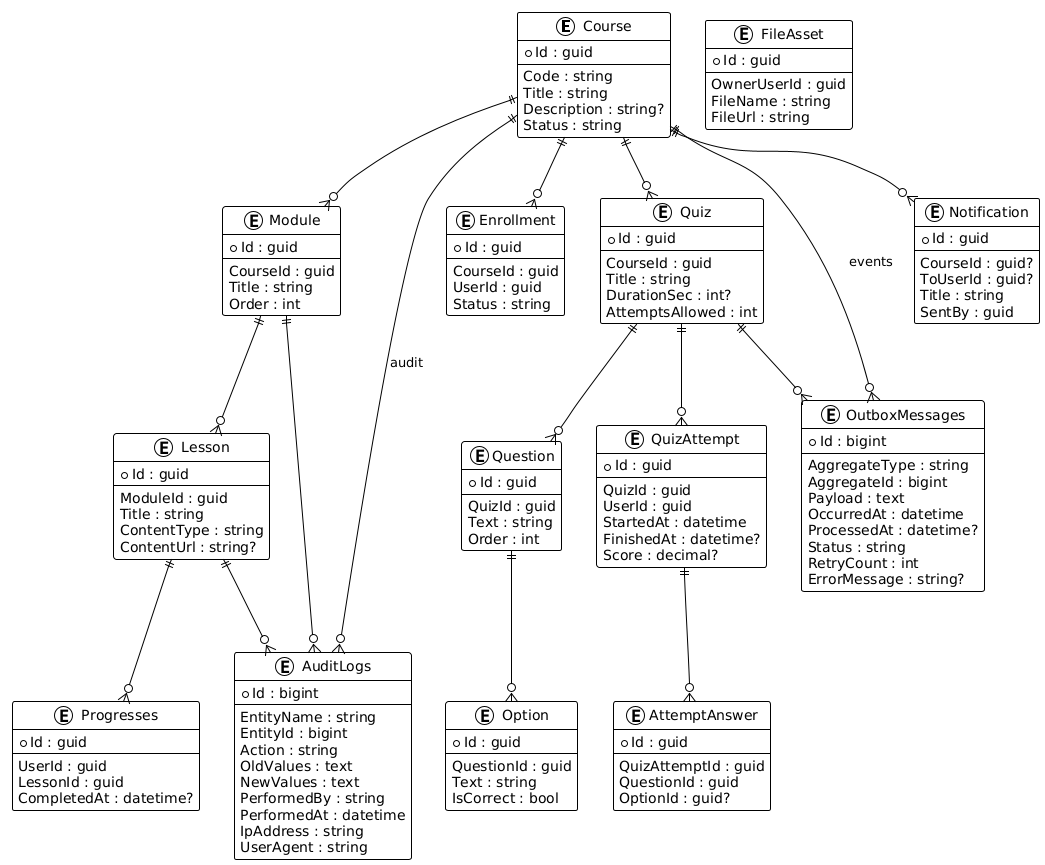
## Indexes (tóm tắt)

* AspNetUsers
  + IX\_AspNetUsers\_NormalizedEmail (NormalizedEmail)
  + IX\_AspNetUsers\_UserName (UserName)
  + UQ\_AspNetUsers\_NormalizedUserName (NormalizedUserName)
* Courses
  + IX\_Courses\_Status (Status)
  + IX\_Courses\_Title (Title)
  + UQ\_Courses\_Code (Code)
* Modules
  + IX\_Modules\_Course\_Order (CourseId, [Order])
* Lessons
  + IX\_Lessons\_Module\_Order (ModuleId, [Order])
* Enrollments
  + IX\_Enrollments\_User (UserId)
  + IX\_Enrollments\_Course\_Status (CourseId, Status)
  + UQ\_Enrollments\_CourseId\_UserId (CourseId, UserId)
* Progresses
  + IX\_Progresses\_User (UserId)
  + UQ\_Progresses\_UserId\_LessonId (UserId, LessonId)
* Quizzes
  + IX\_Quizzes\_Course (CourseId)
* Questions
  + IX\_Questions\_Quiz\_Order (QuizId, [Order])
* Options
  + IX\_Options\_Question (QuestionId)
* QuizAttempts
  + IX\_QuizAttempts\_Quiz\_User (QuizId, UserId)
* AttemptAnswers
  + UQ\_AttemptAnswers\_AttemptId\_QuestionId (AttemptId, QuestionId)
* Notifications
  + IX\_Notifications\_ToUserId (ToUserId)
* FileAssets
  + IX\_FileAssets\_OwnerUserId (OwnerUserId)

## Quan hệ ERD (Tóm tắt)

* Courses ↔ Modules
  + Quan hệ: 1–n
  + Ràng buộc: Modules.CourseId → Courses.Id (FK\_Modules\_Courses)
* Modules ↔ Lessons
  + Quan hệ: 1–n
  + Ràng buộc: Lessons.ModuleId → Modules.Id (FK\_Lessons\_Modules)
* Courses ↔ Enrollments
  + Quan hệ: 1–n
  + Ràng buộc: Enrollments.CourseId → Courses.Id (FK\_Enrollments\_Courses)
* AspNetUsers ↔ Enrollments
  + Quan hệ: 1–n
  + Ràng buộc: Enrollments.UserId → AspNetUsers.Id (FK\_Enrollments\_AspNetUsers)
  + Ghi chú: Quan hệ n–n Users ↔ Courses thông qua Enrollments, với ràng buộc duy nhất UQ\_Enrollments\_CourseId\_UserId trên (CourseId, UserId)
* AspNetUsers ↔ Progresses
  + Quan hệ: 1–n
  + Ràng buộc: Progresses.UserId → AspNetUsers.Id (FK\_Progresses\_AspNetUsers)
* Lessons ↔ Progresses
  + Quan hệ: 1–n
  + Ràng buộc: Progresses.LessonId → Lessons.Id (FK\_Progresses\_Lessons)
  + Ràng buộc duy nhất: UQ\_Progresses\_UserId\_LessonId trên (UserId, LessonId)
* Courses ↔ Quizzes
  + Quan hệ: 1–n
  + Ràng buộc: Quizzes.CourseId → Courses.Id (FK\_Quizzes\_Courses)
* Quizzes ↔ Questions
  + Quan hệ: 1–n
  + Ràng buộc: Questions.QuizId → Quizzes.Id (FK\_Questions\_Quizzes)
* Questions ↔ Options
  + Quan hệ: 1–n
  + Ràng buộc: Options.QuestionId → Questions.Id (FK\_Options\_Questions)
* Quizzes ↔ QuizAttempts
  + Quan hệ: 1–n
  + Ràng buộc: QuizAttempts.QuizId → Quizzes.Id (FK\_QuizAttempts\_Quizzes)
* AspNetUsers ↔ QuizAttempts
  + Quan hệ: 1–n
  + Ràng buộc: QuizAttempts.UserId → AspNetUsers.Id (FK\_QuizAttempts\_AspNetUsers)
* QuizAttempts ↔ AttemptAnswers
  + Quan hệ: 1–n
  + Ràng buộc:
    - AttemptAnswers.AttemptId → QuizAttempts.Id (FK\_AttemptAnswers\_QuizAttempts)
    - Liên kết tới Questions: FK\_AttemptAnswers\_Questions
    - Liên kết tới Options: FK\_AttemptAnswers\_Options
  + Ràng buộc duy nhất: UQ\_AttemptAnswers\_AttemptId\_QuestionId trên (AttemptId, QuestionId)
* Notifications (tùy chọn) ↔ Courses
  + Quan hệ: n–1 (tùy chọn)
  + Ràng buộc: Notifications.CourseId → Courses.Id (FK\_Notifications\_Courses)
* Notifications (tùy chọn) ↔ AspNetUsers
  + Trường ToUserId: Notifications.ToUserId → AspNetUsers.Id (FK\_Notifications\_AspNetUsers\_ToUser)
  + Trường SentBy (bắt buộc): Notifications.SentBy → AspNetUsers.Id (FK\_Notifications\_AspNetUsers\_SentBy)
* FileAssets ↔ AspNetUsers (Owner)
  + Quan hệ: 1–n
  + Ràng buộc: FileAssets.OwnerUserId → AspNetUsers.Id (FK\_FileAssets\_AspNetUsers)

## Quan hệ ERD (Diagram)



Hình 20. ERD Chi tiết với Audit và Soft Delete

## Migration

* Công cụ: EF Core Migrations.
* Quy trình:
  1. Phát triển: dotnet ef migrations add <Name>
  2. Kiểm thử: apply migration trên staging.
  3. Triển khai: dotnet ef database update trong CI/CD pipeline.
* Mỗi migration được versioning và review code trước khi merge.

## Chiến lược Seed Data

1. Roles: tạo Admin, Instructor, Learner.
2. Admin user: tạo user admin, gán role Admin.
3. Khóa học mẫu: 1 course (CreatedBy = admin), 1 module, 2 lessons.
4. Quiz mẫu: 1 quiz thuộc course, N questions + options.
5. Learners: tạo vài user role Learner, enroll vào course.

Ràng buộc:

* Tuân thủ UQ/PK/FK (UQ\_Courses\_Code; UQ\_Enrollments\_CourseId\_UserId; UQ\_Progresses\_UserId\_LessonId).
* Thứ tự chèn theo phụ thuộc.
* Thời gian dùng SYSUTCDATETIME().

# Domain Design

## Giới thiệu

Miền nghiệp vụ LMS bao gồm các thực thể cốt lõi phản ánh toàn bộ chu trình học tập: tổ chức nội dung, quản lý học viên, theo dõi tiến trình, đánh giá qua quiz và gửi thông báo. Thiết kế này hiện thực các nguyên tắc tại mục 1.5.

## Entities & Aggregates

Entities chính:

* Course: Khoá học, chứa nhiều Module, Lesson, Quiz
* Module: Thành phần của Course, nhóm nhiều Lesson
* Lesson: Bài học, có nội dung và tài nguyên
* Quiz: Bài kiểm tra thuộc Course, gồm nhiều Question
* Question: Câu hỏi của Quiz, có nhiều Option
* Option: Lựa chọn trả lời, có đánh dấu đúng/sai
* Enrollment: Ghi danh User vào Course
* Progresses: Tiến trình User trong Lesson
* QuizAttempt: Lượt làm Quiz của User
* AttemptAnswer: Câu trả lời trong một QuizAttempt
* Notification: Thông báo tới User hoặc theo Course
* FileAsset: Tài nguyên file (tài liệu, video)

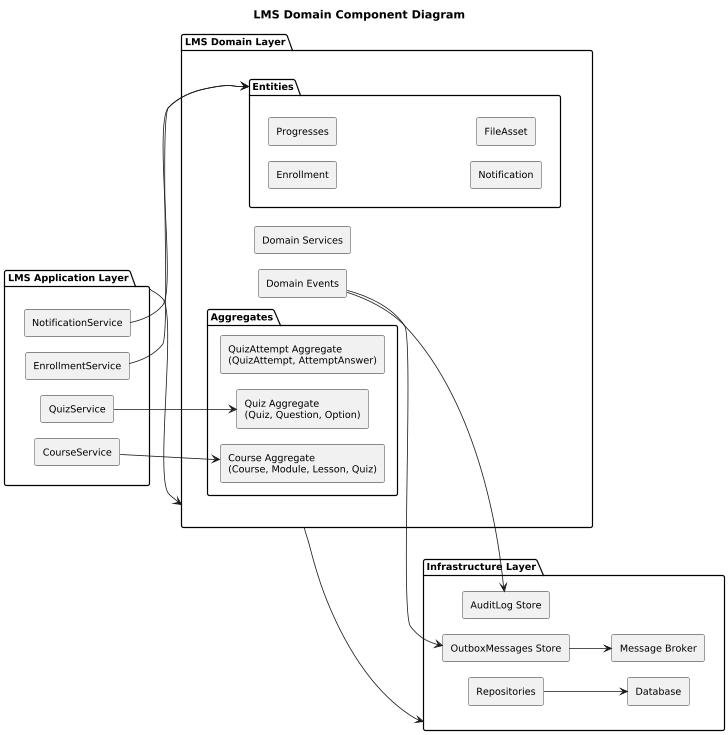
Aggregates:

* Course Aggregate: Root là Course, quản lý Module, Lesson, Quiz
* Quiz Aggregate: Root là Quiz, quản lý Question, Option
* QuizAttempt Aggregate: Root là QuizAttempt, quản lý AttemptAnswer

## Luồng Domain Event – Outbox – Audit

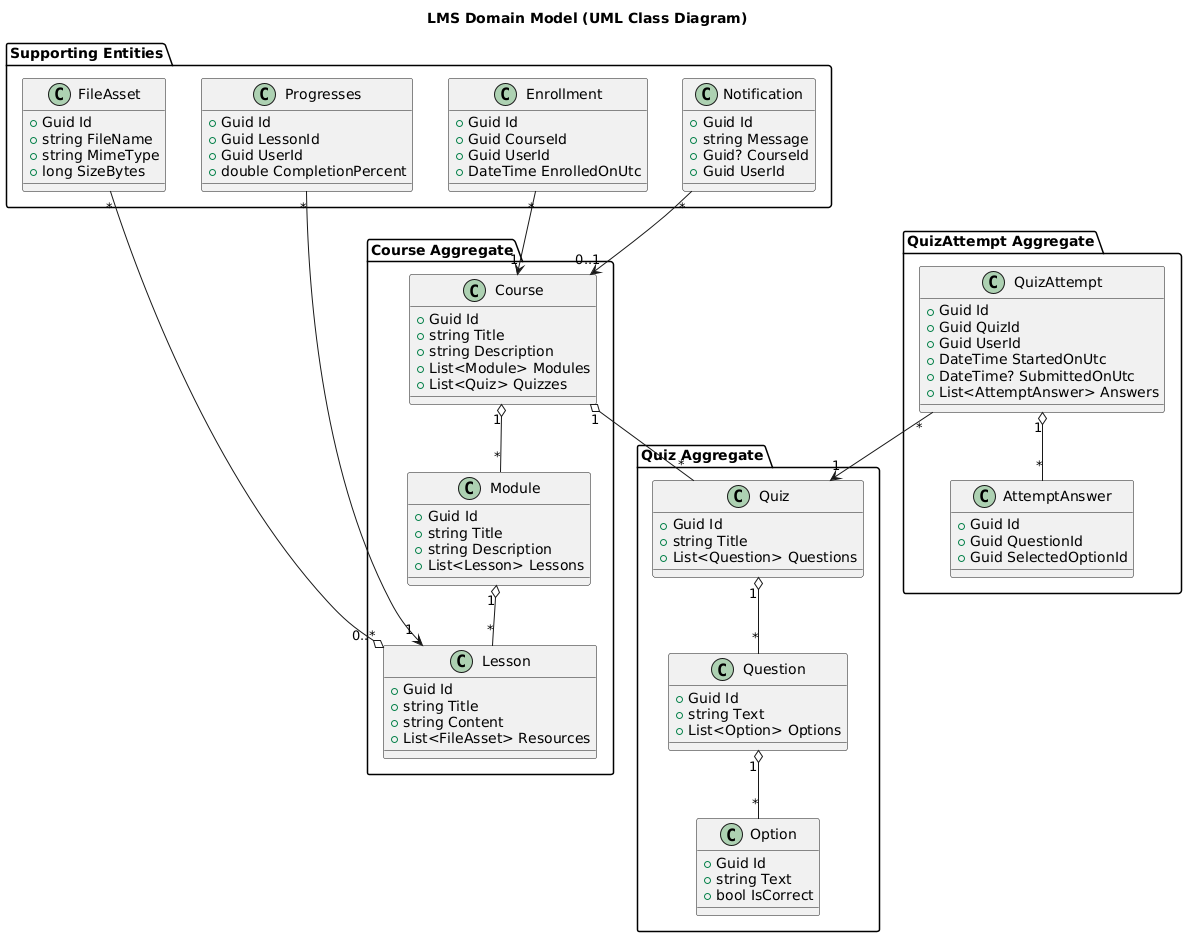
1. Aggregate thực hiện nghiệp vụ → raise Domain Event
2. Persist event vào Outbox và (nếu cần) tạo AuditEntry trong cùng transaction
3. Background processor publish event từ Outbox
4. Handler/subscriber thực hiện hành động bổ sung (gửi thông báo, tích hợp ngoài…)

## Component Diagram



Hình 21. Component Diagram Domain Design

## Class Diagram UML



Hình 22. Class Diagram Domain Design

## Business Rules

### Mô tả nghiệp vụ

* Một Course có thể có nhiều Module, mỗi Module nhiều Lesson
* Một Course có thể có nhiều Quiz, mỗi Quiz nhiều Question, mỗi Question nhiều Option
* Một User ghi danh vào Course và có Progress theo Lesson
* QuizAttempt thuộc Quiz và User; chỉ submit khi quiz mở
* Không xoá Lesson nếu có Progress liên quan
* Notification có thể gắn với Course hoặc gửi thẳng tới User

### Quy tắc kiểm thử

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RuleID | Mô tả | TestCaseID | Input | Expected Output |
| BR001 | User phải xác thực mới được truy cập | TC001 | Token hợp lệ | 200 OK |
| BR002 | Quota file không vượt quá 100MB | TC002 | Upload 120MB | Error: Quota exceeded |
| BR003 | Thông báo chỉ gửi cho user tồn tại | TC003 | UserID invalid | Error: User Not Found |
| BR004 | Mọi thao tác trong hệ thống phải được ghi nhận vào audit log | TC004 | Tạo Course mới | Log entry created |

## Domain Services

* EnrollmentService: xử lý ghi danh kèm thanh toán, gửi email
* ProgressCalculationService: tính tiến trình từ nhiều nguồn dữ liệu
* NotificationService: tạo thông báo từ domain event và đẩy qua hạ tầng gửi

## Cross-Cutting Concerns

* Audit Logging: ghi nhận mọi thay đổi quan trọng
* Security & Permissions: kiểm tra quyền trước khi thay đổi entity
* Outbox Reliability: đảm bảo at-least-once delivery
* **Specification Pattern** là một mẫu thiết kế giúp tách rời logic lọc, sắp xếp, và phân trang ra khỏi repository, cho phép tái sử dụng và kết hợp điều kiện một cách linh hoạt. Trong LMS‑Mini, pattern này được hiện thực qua interface IQuerySpecification<T>, áp dụng cho các tình huống như:
  + Lọc các khoá học đang hoạt động (ActiveCoursesSpecification)
  + Lọc khoá học theo giảng viên (CoursesByInstructorSpecification)
  + Lọc câu hỏi quiz theo khoá học (QuizByCourseSpecification)

# Thiết kế API & Contract

## Contract nội bộ (Port / Adapter — Application.Abstractions ↔ Infrastructure)

Mục này liệt kê các interface (port) ở tầng Application / Domain và nơi triển khai ở Infrastructure. Dùng để backend engineers triển khai theo Clean Architecture.

|  |  |  |  |
| --- | --- | --- | --- |
| Interface (Port) | Miêu tả | Định nghĩa tại | Triển khai tại |
| ICourseRepository | CRUD và truy vấn khóa học, hỗ trợ phân trang | Application.Abstractions | Infrastructure.Repositories.CourseRepository |
| IModuleRepository | CRUD module và bài học | Application.Abstractions | Infrastructure.Repositories.ModuleRepository |
| IEnrollmentRepository | Thao tác ghi danh học viên vào khóa học | Application.Abstractions | Infrastructure.Repositories.EnrollmentRepository |
| IProgressRepository | Cập nhật và truy vấn tiến độ học tập | Application.Abstractions | Infrastructure.Repositories.ProgressRepository |
| IQuizRepository | CRUD bài kiểm tra, câu hỏi, tùy chọn, và lượt làm bài | Application.Abstractions | Infrastructure.Repositories.QuizRepository |
| IUnitOfWork | Định nghĩa ranh giới transaction | Application.Abstractions | Infrastructure.UnitOfWork |
| IFileStorage | Upload, lấy URL có chữ ký, và xóa file | Application.Abstractions | Infrastructure.FileStorage.AzureBlobFileStorage |
| IEmailSender | Gửi email thông báo | Application.Abstractions | Infrastructure.Email.SMTPEmailSender |
| IDateTimeProvider | Lấy thời gian hệ thống (UTC) | Application.Abstractions | Infrastructure.Services.SystemDateTimeProvider |
| IAuthService | Kiểm tra vai trò và chính sách phân quyền | Application.Abstractions | Infrastructure.Services.AuthService |
| IReportService | Truy vấn tổng hợp cho báo cáo | Application.Abstractions | Infrastructure.Services.ReportService |
| IQuerySpecification<T> | Gom điều kiện lọc, sắp xếp, phân trang thành Specification object | Application.Abstractions | Infrastructure.Specifications |

Ghi chú (implementation):

* Audit/Soft Delete/RowVersion/Outbox xử lý ở tầng Repository/Infrastructure; Application Service chỉ gọi interface.
* Mapping Endpoint → Service → Interface được nêu ở phần 4.3.

## Contract bên ngoài (REST API cho client)

### Nguyên tắc chung (Headers, Versioning, Response Envelope, i18n, CSV, v.v.)

* Base URL: /api/v1
* Auth: JWT Bearer Token (AspNet Identity).
* Format dữ liệu: JSON.
* Response Envelope (thống nhất):
  + Thành công:

|  |
| --- |
| {  "success": true,  "data": { },  "error": null,  "traceId": "..."  } |

* + Lỗi:

|  |
| --- |
| {  "success": false,  "data": null,  "error": { "code": "ERR\_CODE", "message": "Human message", "details": null },  "traceId": "..."  } |

* Idempotency:
  + Header: Idempotency-Key dùng cho POST có side-effect (ví dụ: Enroll, Start Quiz, Create Course).
  + Lưu key tối thiểu 24h; cache theo (user + route + payload hash); trả 409 hoặc idempotent response khi duplicate.
* Concurrency (ETag):
  + PUT / PATCH yêu cầu header If-Match = ETag (map từ RowVersion).
  + Nếu mismatch → trả 412 Precondition Failed.
  + ETag mẫu: ETag: "\"rowversion-base64\"".
* Pagination:
  + Query params: page, size (size tối đa 100); sort=field:asc|desc (chỉ cho field allowlist).
  + Response headers: X-Total-Count, Link (rel="next"/"prev").
* Rate limiting:
  + Trả 429 Too Many Requests khi vượt. Kèm headers: Retry-After, RateLimit-Limit, RateLimit-Remaining, RateLimit-Reset.
  + Áp dụng chặt cho /reports/, /files/, login.
* i18n:
  + Đọc Accept-Language; error.code cố định, error.message dịch theo locale.
* CSV export safety:
  + Escape ký tự đầu dòng = + - @ bằng ' hoặc prefix tab.
  + Thêm header X-Content-Type-Options: nosniff.
  + Gợi ý thêm BOM UTF-8 nếu cần hiển thị Unicode trên Excel.
* Sample Response Headers:

|  |
| --- |
| ETag: "\"rowversion-base64\""  X-Total-Count: 240  Link: </api/v1/courses?page=3&size=20>; rel="next"  Retry-After: 60 |

### Courses (Endpoints & contract)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Path | Request DTO | Response DTO | Status |
| GET | /api/v1/courses?search=&status=&page=&size= | — | PagedResult<CourseListDto> | 200 |
| GET | /api/v1/courses/{id} | — | CourseDto | 200 / 404 |
| POST | /api/v1/courses | CreateCourseRequest | CreatedResponse { id } | 201 / 400 |
| PUT | /api/v1/courses/{id} | UpdateCourseRequest | — | 204 / 400 / 403 |
| POST | /api/v1/courses/{id}/publish | — | CourseDto | 200 / 404 |
| DELETE | /api/v1/courses/{id} | — | — | 204 / 404 |

* DELETE semantics: Soft Delete (IsDeleted = 1).
* Quy ước đặc biệt:
* POST /courses: Idempotency-Key tùy chọn; 412 nếu ETag mismatch; 409 nếu CodeExists.
* DTOs (ví dụ C#):

public record CreateCourseRequest(string Code, string Title, string? Description);

public record UpdateCourseRequest(string Title, string? Description, string Status);

public record CourseDto(Guid Id, string Code, string Title, string? Description, string Status, DateTime CreatedAt);

public record CourseListDto(Guid Id, string Code, string Title, string Status);

### Modules & Lessons

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Path | Request DTO | Response |
| POST | /api/v1/courses/{courseId}/modules | CreateModuleRequest | 201 |
| GET | /api/v1/courses/{courseId}/modules | — | List<ModuleDto> |
| POST | /api/v1/modules/{moduleId}/lessons | CreateLessonRequest (incl. ContentType, ContentUrl) | 201 |
| PUT | /api/v1/lessons/{id} | UpdateLessonRequest | 204 |
| POST | /api/v1/lessons/{id}/complete | — | 200 (updated progress) |

* DELETE semantics: Soft Delete cho Modules & Lessons.

### Enrollment

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Path | Request DTO | Response |
| POST | /api/v1/courses/{id}/enroll | EnrollRequest { userId } or CSV | 200 |
| GET | /api/v1/courses/{id}/enrollments | filters | List<EnrollmentDto> |
| DELETE | /api/v1/enrollments/{id} | — | 204 |

* DELETE semantics: Soft Delete (IsDeleted = 1).
* Khuyến nghị DB: Dùng unique index (CourseId, UserId) có WHERE IsDeleted = 0 để cho phép ghi danh lại (re-enroll).
* Quy ước: POST /courses/{id}/enroll Idempotency-Key bắt buộc; idempotent theo (CourseId, UserId).

Mẫu lỗi (ví dụ)

* 404 Not Found (ERR\_NOT\_FOUND) — Envelope lỗi như phần 4.2.1.
* 409 Conflict (ERR\_CONFLICT) — idempotency duplicate / business conflict.

### Quiz

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Path | Request DTO | Response DTO |
| POST | /api/v1/courses/{courseId}/quizzes | CreateQuizRequest | 201 |
| GET | /api/v1/quizzes/{id} | — | QuizDto |
| POST | /api/v1/quizzes/{id}/start | — | StartQuizResponse |
| POST | /api/v1/quizzes/{id}/submit | SubmitQuizRequest | SubmitQuizResponse |
| GET | /api/v1/quizzes/{id}/attempts?userId= | — | List<QuizAttemptDto> |

* SubmitQuizRequest

{

"attemptId": "guid",

"answers": [

{ "questionId": "guid", "optionId": "guid" }

]

}

* SubmitQuizResponse

{ "attemptId": "guid", "score": 85.5, "correctCount": 17, "total": 20 }

**Quy ước đặc biệt**

* POST /quizzes/{id}/start: Idempotency-Key bắt buộc; chỉ 1 attempt mở duy nhất mỗi (user, quiz).
* POST /quizzes/{id}/submit: validate answers, tính điểm, cập nhật QuizAttempts.
* PUT với If-Match bắt buộc cho các cập nhật rủi ro concurrency.

### Reports

|  |  |  |
| --- | --- | --- |
| Method | Path | Response |
| GET | /api/v1/reports/course/{id}/progress?from=&to= | CourseProgressReportDto |
| GET | /api/v1/reports/course/{id}/scores | List<StudentScoreDto> |

* CSV export: hỗ trợ Accept: text/csv hoặc ?format=csv, áp dụng CSV safety (escape, nosniff, BOM).

### Error Codes (tổng hợp)

|  |  |  |
| --- | --- | --- |
| Code | HTTP Status | Ý nghĩa |
| ERR\_NOT\_FOUND | 404 | Không tìm thấy tài nguyên |
| ERR\_FORBIDDEN | 403 | Không được phép |
| ERR\_VALIDATION | 400 | Dữ liệu không hợp lệ |
| ERR\_CONFLICT | 409 | Xung đột nghiệp vụ |
| ERR\_INTERNAL | 500 | Lỗi hệ thống nội bộ |
| ERR\_PRECONDITION | 412 | ETag mismatch / precondition failed |

* Mỗi lỗi phải kèm traceId. message tuỳ theo Accept-Language.

## Mapping: Endpoint ↔ Interface / Service (gợi ý triển khai)

|  |  |  |
| --- | --- | --- |
| Endpoint (ví dụ) | Application Service | Interface(s) (Port) |
| POST /api/v1/courses/{id}/enroll | EnrollmentService.Enroll() | IEnrollmentRepository, IUnitOfWork, IAuthService |
| POST /api/v1/lessons/{id}/complete | LessonService.CompleteLesson() | IProgressRepository, IUnitOfWork |
| POST /api/v1/quizzes/{id}/start | QuizService.StartQuiz() | IQuizRepository, IUnitOfWork |
| POST /api/v1/quizzes/{id}/submit | QuizService.SubmitQuiz() | IQuizRepository, IUnitOfWork |
| POST /api/v1/notifications | NotificationService.Create() | INotificationRepository, IEmailSender, IUnitOfWork |
| POST /api/v1/files | FileService.Upload() | IFileStorage, IFileRepository, IUnitOfWork |

Ghi chú: mapping này giúp developer biết endpoint sẽ gọi service nào và service sẽ tương tác với port nào (Application → Domain Abstraction → Infrastructure).

## DTOs, OpenAPI & Phụ lục

* DTO chi tiết (request/response): đưa vào Phụ lục C – DTOs & Sample Payloads để tránh làm mục 4 quá dài.
* OpenAPI (rút gọn): nếu cần, tạo file openapi.yaml/openapi.json và đính kèm / tham chiếu trong phụ lục.
* Sequence Logic (flow step-by-step) nằm ở Mục 6 — không lặp lại ở Mục 4. Mục 4 chỉ nêu contract (giao tiếp).

## Ghi chú triển khai & kiểm thử

* Các endpoint có yêu cầu Idempotency-Key / If-Match phải có kiểm thử tương ứng (unit + integration).
* Các lỗi business (409) và concurrency (412) phải mô tả test case rõ ràng trong test plan.
* API phải tuân theo envelope + traceId để dễ trace khi deploy.

# Thiết kế UI/UX

## Layout tổng thể

* TopBar: Logo, tên tổ chức, thanh tìm kiếm toàn cục, biểu tượng thông báo, menu người dùng (avatar dropdown).
* SideNav (thu gọn được): Dashboard, Courses (List / Create), Learners, Quizzes, Reports, Settings (Admin only).
* ContentArea: Page Header (Title + Breadcrumb), khu vực actions (Create, Filter), nội dung chính (cards, tables, forms).
* Footer (tùy chọn): Thông tin bản quyền, phiên bản hệ thống.

Thuật ngữ: Learners = AspNetUsers với role Learner, đồng bộ với mô hình Identity.

## Khả năng truy cập (A11y)

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí | Pass | Fail | Ghi chú |
| Focus trap cho modal, aria-labelledby / aria-describedby, trả focus về trigger | ☐ |  | Modal CourseEditor đã thêm role="dialog", aria-modal, trap bằng focus-guard; đóng modal trả focus về nút Open. |
| Toast/Alert có `aria-live=polite |  | ☐ | Toast lỗi đang thiếu aria-live; bổ sung aria-live="assertive" cho lỗi và polite cho thông báo thường. Chủ trì: FE, Hạn: 29/08. |
| Keyboard navigation đầy đủ (tables sort/filter, quiz chọn 1–9, Enter Next/Submit) |  | ☐ | DataTable chưa có shortcut sort bằng phím; Quiz thiếu handler phím số. Bổ sung onkeydown; viết test bUnit. |
| Visible focus ring trên mọi control tương tác | ☐ |  | Đã bỏ outline: none; áp dụng :focus-visible với màu tương phản. |
| Contrast tối thiểu 4.5 : 1 | ☐ |  | Palette primary/secondary đã đo ≥ 4.5:1; cần recheck cho badge “Muted”. |
| Error summary ở đầu form, liên kết tới field lỗi |  | ☐ | Thêm vùng <div role="alert" aria-labelledby="formErrors"> với anchor đến input có aria-invalid="true". |
| AuthorizeView điều khiển hiển thị theo vai trò | ☐ |  | Đã ẩn nút Admin với Learner; confirm server-side policy đồng nhất. |
| Reconnect banner & disable hành động nhạy cảm |  | ☐ | Banner có, nhưng chưa disable các nút POST. Thêm state isConnected để disabled. Chủ trì: FE, Hạn: 30/08. |
| Alt text cho hình ảnh | ☐ |  | Đã áp dụng |
| |  | | --- | |  |   Contrast ≥ 4.5:1 | ☐ |  | Verified |
| |  | | --- | |  |   Keyboard navigation đầy đủ |  | ☐ | |  | | --- | |  |   Cần bổ sung phím tắt |
| |  | | --- | |  |   Screen reader compatibility | ☐ |  | Test bằng NVDA |
| |  | | --- | |  |   Form validation có thông báo rõ ràng |  | ☐ | Thiếu mesassage Tiếng Việt |

## Độ bền kết nối — Blazor Server

* **Reconnect UI:**
  + Banner hiển thị trạng thái “đang reconnect” / “mất kết nối” và vô hiệu các hành động nhạy cảm.
  + Retry: Exponential backoff kèm feedback cho người dùng.
* **Autosave Quiz:**
  + Lưu nháp câu trả lời mỗi 5–10 giây hoặc khi người dùng ngừng tương tác.
  + Ưu tiên lưu server; fallback sang LocalStorage qua JS Interop.
  + Chống mất đồng bộ: Gắn version / ETag cho bản nháp, merge khi reconnect.
* **Throttling:**
  + Debounce nhập liệu 300–500 ms.
  + Batch cập nhật để giảm số roundtrip.

## Phân tách layout Admin vs Learner

* Admin / Instructor: Menus quản trị (Users, Courses, Reports, Settings). Dashboard hiển thị KPI hệ thống.
* Learner: Menus My Courses, Progress, Profile. Dashboard hiển thị progress bar, upcoming quizzes.
* Component Switching: Sử dụng AuthorizeView để thay đổi giao diện theo role.

## Responsive Rules

* Breakpoints: Sử dụng mặc định Bootstrap 5.
* Desktop: SideNav mở rộng.
* Tablet: SideNav collapsed mặc định, có nút toggle.
* Mobile: SideNav dạng off‑canvas; bổ sung bottom‑nav với icon cho Dashboard, Courses, Me.

## Component Library & Styling

* Tech stack: Bootstrap 5 + custom SCSS, Font Awesome 6, SweetAlert2 (JS Interop).
* Reusable components:
  + LmsDataTable<T>: Server‑side paging, sort, filter, selection (pageSize, pageIndex).
  + LmsForm<T>: EditForm + FluentValidation + Save/Cancel.
  + ProgressBar: Hiển thị %.
  + QuizRunner: Câu hỏi, phương án, timer.
  + KpiCard: KPI số + delta.
  + Alert/Toast: Thông báo toàn cục.
  + ConfirmDialog: Modal xác nhận.
  + FileUploader: Kéo‑thả, hiển thị tiến trình.
* A11y: Bổ sung ARIA attributes, hỗ trợ keyboard navigation, quản lý focus cho modals.

## Popup Rules

* Modal Service: Service trung tâm (singleton), aria-modal="true".
* Confirmations: Dùng SweetAlert2 với template chuẩn (Title, Body, Primary/Secondary).
* Forms in Modal: Không autosave; có nút Save; cảnh báo khi đóng nếu form dirty.
* Large Content: Nội dung > 10 MB mở ở trang / viewer riêng, tránh nhúng vào modal.

# Luồng nghiệp vụ chính (Sequence Logic)

## Use Case: Ghi danh khóa học (EnrollCourse)

**Mục tiêu**: Cho phép người học ghi danh vào một khoá học. Hệ thống tạo bản ghi Enrollment và ngăn ghi danh trùng.

**Preconditions**:

* User đã đăng nhập.
* Course tồn tại và chưa bị xoá.
* Sử dụng UUID; áp dụng Audit/Soft Delete.

**Main Flow:**

1. UI gọi Enroll(courseId).
2. Application gọi IAuthService để kiểm tra quyền.
3. Kiểm tra ICourseRepository (Course chưa xoá).
4. Kiểm tra UQ\_Enrollments\_CourseId\_UserId.
5. Mở UnitOfWork.
6. IEnrollmentRepository.Add() tạo bản ghi Enrollment.
7. Commit transaction.
8. Ghi Domain Event EnrollmentCreated vào Outbox.
9. Trả ApiResponse về UI.

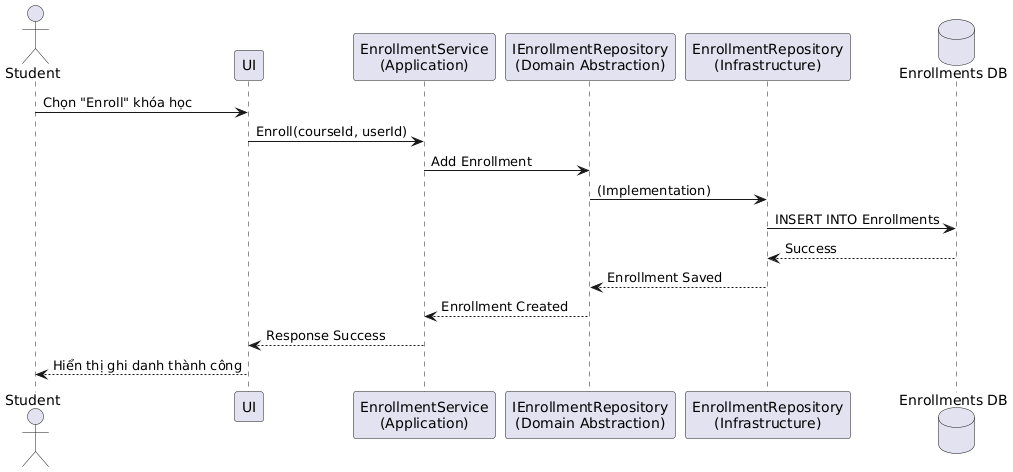
**Alternative/Exception:**

* Course không tồn tại → 404.
* User đã ghi danh → 409 hoặc idempotent.
* Không đủ quyền → 403.

**Data touched**: Courses (read), Enrollments (write).

**API**: POST /api/v1/courses/{id}/enroll.

**Diagram**



Hình 23. Sequence Diagram Ghi danh khóa học

**Mô tả**: Sinh viên chọn một khóa học để ghi danh. Hệ thống tạo bản ghi Enrollment (Audit, StartAt, Status) và đảm bảo không cho phép ghi danh trùng (UQ\_Enrollments\_CourseId\_UserId).

**Preconditions**

* User đã đăng nhập.
* Course tồn tại và không bị IsDeleted.
* Các định danh theo chuẩn SDD (UUID); Audit/Soft Delete áp dụng.

**Main flow**

1. UI gọi hành động Enroll(courseId).
2. Application (EnrollmentService/Handler) gọi IAuthService để kiểm tra quyền (nếu chính sách yêu cầu).
3. Service kiểm tra ICourseRepository để đảm bảo Course còn hoạt động (IsDeleted = 0).
4. Service áp dụng Specification "chưa ghi danh" (kiểm tra UQ\_Enrollments\_CourseId\_UserId).
5. Mở UnitOfWork.
6. Gọi IEnrollmentRepository.Add() để tạo bản ghi Enrollment (gán StartAt, Status, CreatedAt, CreatedBy).
7. Commit UnitOfWork.
8. (Tuỳ cấu hình) Ghi Domain Event EnrollmentCreated vào Outbox để xử lý sau (ví dụ: gửi thông báo, thống kê).
9. Trả kết quả về UI (ApiResponse envelope, kèm traceId).

**Alternative / Exception**

* Course không tồn tại hoặc đã xoá → trả 404 ERR\_NOT\_FOUND.
* Người dùng đã ghi danh trước đó → 409 ERR\_CONFLICT (hoặc trả message nghiệp vụ idempotent).
* Không đủ quyền → 403 ERR\_FORBIDDEN. Data touched: Courses (read), Enrollments (write).

**Diagram**: Hình 23

## Use Case: Nộp quiz (SubmitQuiz)

**Mục tiêu**: Người học bắt đầu quiz và nộp kết quả. Bao gồm snapshot, scoring, idempotency.

**Preconditions**:

* User đã đăng nhập và ghi danh.
* Quiz đã publish, còn hiệu lực.

**Main Flow (Start):**

1. UI gửi StartQuizCommand(quizId, userId).
2. Application xác thực enrollment, số lần attempt.
3. Nếu có Attempt InProgress → trả lại (idempotent).
4. Sinh snapshot questionSetVersion.
5. Tạo bản ghi QuizAttempt (InProgress).
6. Commit transaction.
7. Trả AttemptId, expireAt, duration.

**Main Flow (Submit):**

1. UI gửi SubmitQuiz(attemptId, answers).
2. Application xác thực Attempt.
3. Tính điểm từ snapshot.
4. Lưu AttemptAnswers và cập nhật QuizAttempt.
5. Cập nhật Progress.
6. Ghi Domain Event QuizSubmitted.
7. Commit transaction.
8. Trả kết quả (score, passed).

**Alternative/Exception**:

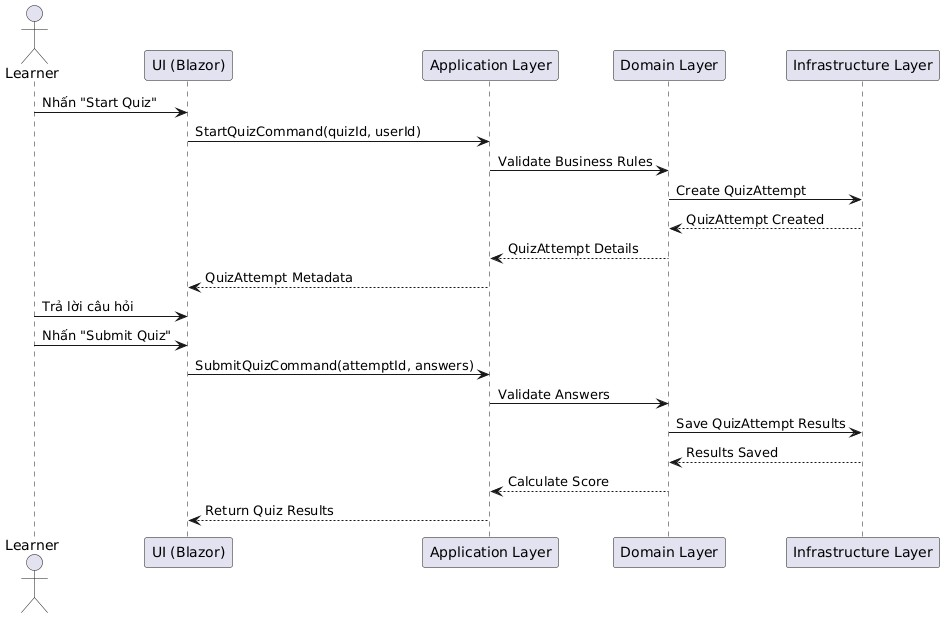
* Attempt không hợp lệ → 403/409.
* Quiz không tồn tại → 404.

**Data touched**: Quizzes, QuizAttempts, AttemptAnswers, Progresses.

**API**:

* POST /api/v1/quizzes/{id}/start
* POST /api/v1/attempts/{id}/submit

**Diagram**:



Hình 24. Sequence Diagram Học viên nộp quiz

Phần này tóm tắt luồng Start → Submit, nhấn mạnh các điểm đặc thù: snapshot/versioning, idempotency, validation, scoring và update tiến độ.

**Learner (UI)**

1. Người học nhấn "Start" → gửi StartQuizCommand(quizId, userId). Application Layer (Start)
2. Xác thực: kiểm tra user đã ghi danh vào course chứa quiz chưa; kiểm tra giới hạn lượt làm (AttemptsAllowed).
3. Kiểm tra điều kiện mở: quiz đã publish, trong khoảng thời gian cho phép, đáp ứng prerequisite.
4. Nếu tồn tại QuizAttempt hợp lệ ở trạng thái InProgress, trả lại thông tin đó (idempotency).
5. Ngẫu nhiên hoá câu hỏi theo seed cố định và tạo snapshot của quiz (questionSetVersion) để đảm bảo tính bất biến khi chấm.
6. Gọi IQuizRepository.CreateQuizAttempt() tạo QuizAttempt (Status = InProgress, StartedAt, Audit, snapshot metadata).
7. Commit transaction; trả về AttemptId, expireAt, duration, questionSetVersion, autoGradable. Learner (UI) — làm bài • Người học trả lời; có thể auto-save (những thao tác lưu tạm là async, non-blocking). Application Layer (Submit)
8. Xác thực: kiểm tra attempt thuộc user; trạng thái là InProgress; chưa hết hạn (hoặc xử lý chính sách quá hạn).
9. Tải đáp án đúng từ snapshot và tính điểm bằng logic Domain (Quiz.CalculateScore).
10. Ghi AttemptAnswers (bulk insert) và cập nhật QuizAttempt (Score, CorrectCount, SubmittedAt, Status).
11. Cập nhật Progress nếu quiz ảnh hưởng tới tiến độ course.
12. Phát Domain Event QuizSubmitted (ghi Outbox nếu cần).
13. Commit transaction.
14. Trả kết quả: score, correctCount, totalQuestions, passed.

**Key checks**

* Authorization (EnrolledOnly policy).
* Giới hạn số lần làm bài và timeout.
* Transaction atomicity giữa AttemptAnswers và QuizAttempt (đảm bảo consistency).
* Idempotency: sử dụng AttemptId hoặc Idempotency-Key để tránh duplicate submissions.

**Alternative / Exception**

* Attempt không thuộc user hoặc đã nộp → 403/409.
* Quiz không tồn tại → 404.

**Data touched**: Quizzes (read snapshot), QuizAttempts, AttemptAnswers, Progresses.

## Use Case: Làm Quiz & Lưu kết quả (QuizAttemptFlow)

Mô tả (tóm tắt): Khi user bắt đầu quiz, tạo QuizAttempt snapshot; khi submit, validate, lưu AttemptAnswers, tính điểm và cập nhật QuizAttempt.

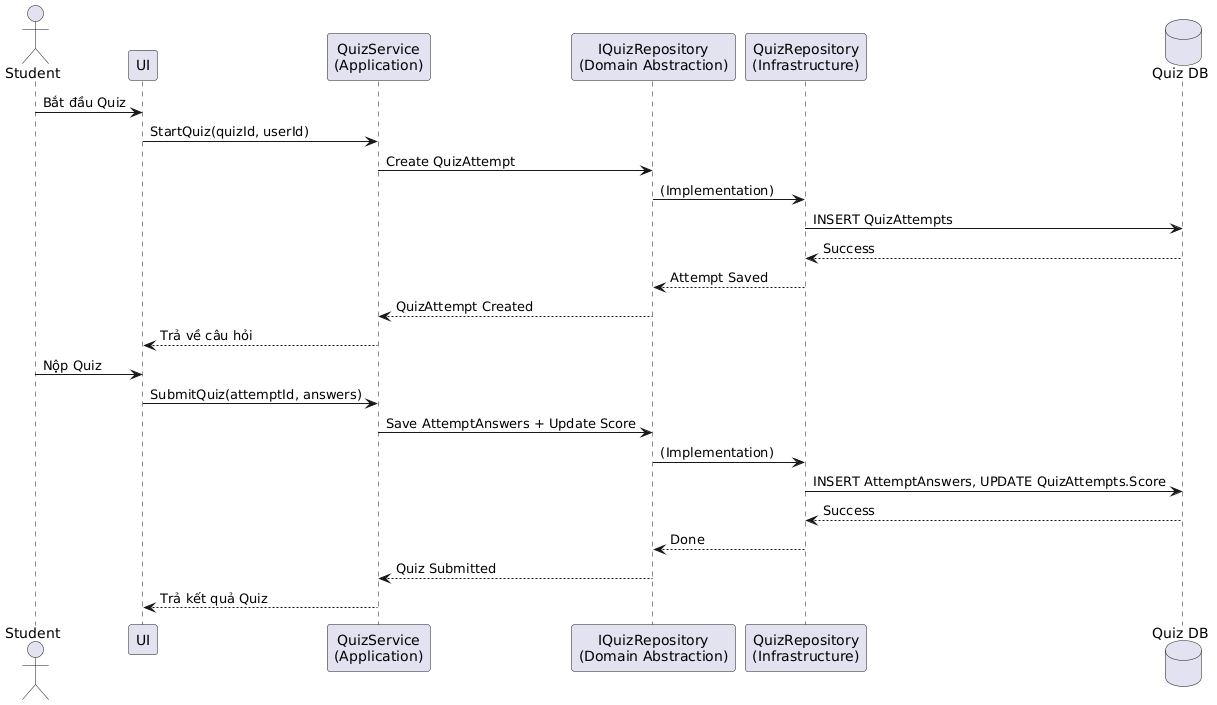
**Preconditions**: User đăng nhập; có quyền làm quiz; quiz thuộc course hợp lệ. Main flow (Start)

1. UI gọi StartQuiz(quizId, userId).
2. QuizService gọi IAuthService kiểm tra quyền.
3. Dùng IQuizRepository tải Questions + Options (áp dụng shuffle nếu cấu hình).
4. Tạo QuizAttempt (Audit, StartedAt, Score=0, snapshot metadata).
5. Commit UnitOfWork và trả câu hỏi cho UI. Main flow (Submit)
6. UI gọi SubmitQuiz(attemptId, answers).
7. Service validate: answers thuộc questions/options của snapshot; attempt thuộc user.
8. Ghi AttemptAnswers hàng loạt; tính Score.
9. Cập nhật QuizAttempts.Score, SubmittedAt, Status.
10. (Tùy) Ghi Domain Event QuizSubmitted vào Outbox.
11. Commit UnitOfWork.
12. Trả kết quả.

**Alternative / Exception**

* Attempt không thuộc user hoặc đã nộp → 403/409.
* Quiz không tồn tại → 404. Data touched: Quizzes, Questions, Options, QuizAttempts, AttemptAnswers.

**Diagram**



Hình 25. Sequence Diagram (bổ sung Quiz Attempt Flow).

## Use Case: Đánh dấu hoàn thành bài học (MarkLessonCompleted)

**Mục tiêu**: Người học đánh dấu lesson hoàn thành.

**Preconditions**: User đã enroll, lesson tồn tại.

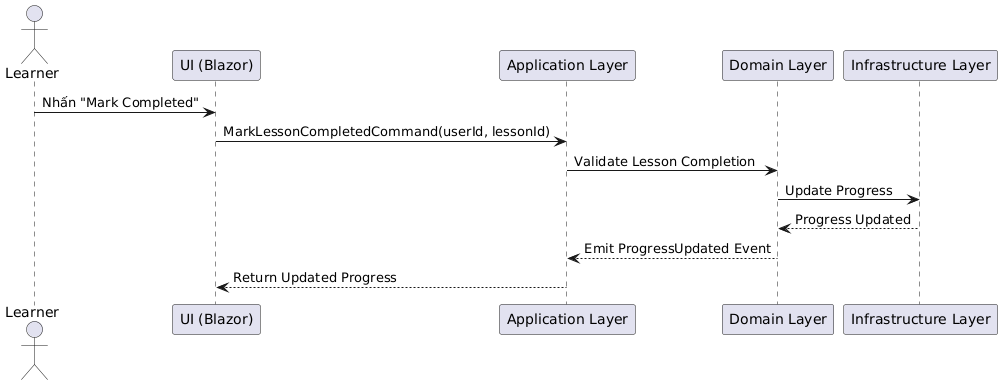
**Main Flow:**

1. UI gửi MarkLessonCompleted(userId, lessonId).
2. Application xác thực.
3. Update hoặc create Progress (CompletedAt, Percent=100).
4. Refresh tiến độ course.
5. Ghi ProgressUpdated event.
6. Commit transaction.
7. Trả updatedPercent.

**Alternative/Exception**: Đảm bảo idempotency (nếu đã completed).

**Data touched**: Progresses, Courses.

**Diagram**:



Hình 26. Sequence Diagram Đánh dấu hoàn thành bài học

**Mô tả**: Người học đánh dấu một lesson là hoàn thành; hệ thống lưu/update Progress và tính lại tiến độ khóa học.

**Main flow**

1. UI gửi MarkLessonCompletedCommand(userId, lessonId).
2. Application xác thực (user đã enroll; lesson tồn tại).
3. Cập nhật hoặc tạo Progress với CompletedAt = UtcNow, Percent = 100 (idempotent).
4. Tính toán hoặc refresh tiến độ course (CoursePercent) qua repository hoặc báo cáo.
5. Phát ProgressUpdated event nếu cần; commit.
6. Trả updatedPercent cho UI.

**Notes**: đảm bảo idempotency (nếu đã hoàn thành, không tạo bản ghi mới).

## Use Case: Tạo khóa học mới (CreateCourse)

**Mục tiêu**: Admin/Instructor tạo course.

Preconditions: User có quyền tạo course.

**Main Flow:**

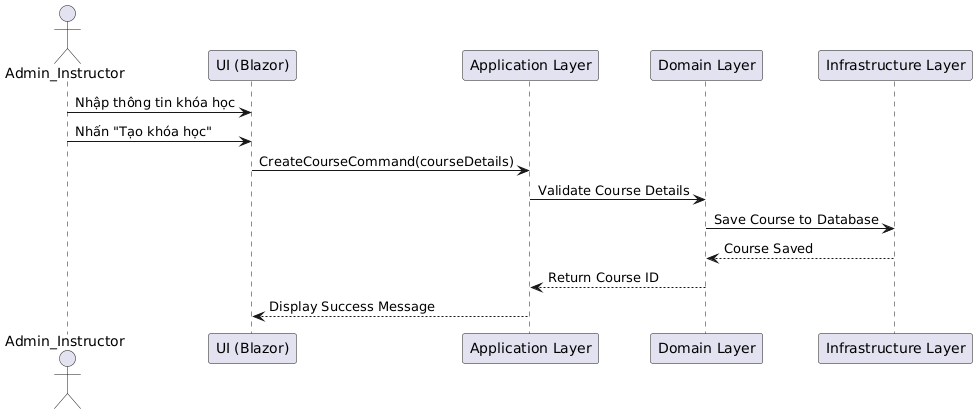
1. UI gửi CreateCourseCommand.
2. Application validate dữ liệu.
3. Domain kiểm tra business rules.
4. Repository thêm Course.
5. Commit transaction.
6. Trả CreatedResponse { id }.

**Alternative/Exception**: Dữ liệu không hợp lệ → 400.

**Data touched**: Courses.

**API**: POST /api/v1/courses.

**Diagram**:



Hình 27. Sequence Diagram Tạo khóa học mới

**Mô tả**: Admin/Instructor tạo course; hệ thống validate thông tin và lưu course (Audit).

**Sequence logic (tóm tắt):**

* UI -> Application (CreateCourseCommand) -> Domain validations -> Infrastructure (CourseRepository.Add) -> Commit -> Trả response (CreatedResponse { id }).

## Use Case: Quản lý bài học (ManageLessons)

**Mục tiêu (Goal):** Cho phép giảng viên (Instructor) thêm, sửa, xóa (soft-delete) bài học (Lesson) cho một khoá học (Course); đảm bảo audit và rule consistency.

**Preconditions:**

Người thao tác đã đăng nhập và có quyền quản lý khoá học (CourseOwner / Instructor).

Khoá học liên quan tồn tại và không ở trạng thái archived/locked.

**Main Flow:**

1. UI gọi endpoint tương ứng:
2. Tạo: POST /api/v1/courses/{courseId}/lessons
3. Cập nhật: PUT /api/v1/lessons/{lessonId}
4. Xóa: DELETE /api/v1/lessons/{lessonId} (soft-delete)
5. API controller chuyển request thành command (Create/Update/DeleteLessonCommand) kèm traceId.
6. Application layer (LessonService/Handler) authorize qua IAuthService.
7. Service validate payload (title, content, duration, attachments) theo business rules.
8. Domain kiểm tra ràng buộc nghiệp vụ (ví dụ số lesson tối đa nếu có policy).
9. Nếu hợp lệ, gọi ILessonRepository.Add/Update/Delete. Với delete: set IsDeleted = true + ghi audit fields.
10. Commit UnitOfWork; (tùy cấu hình) ghi Domain Event (LessonCreated/LessonUpdated/LessonDeleted) vào Outbox.
11. Trả ApiResponse (Lesson DTO hoặc 204 No Content) kèm traceId.

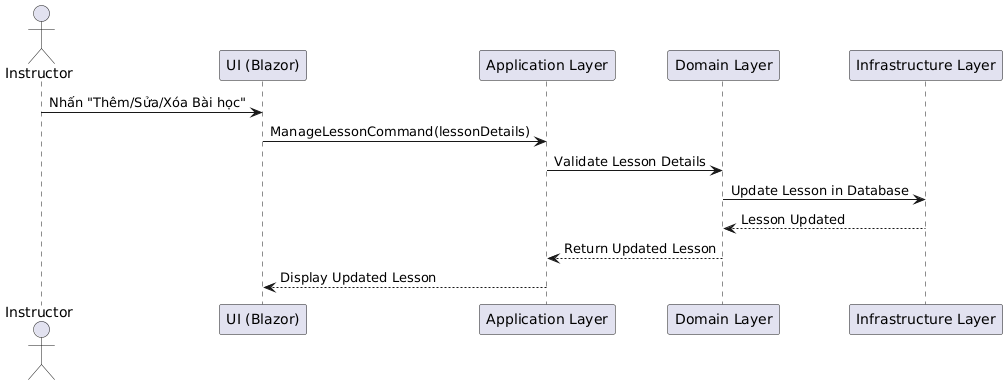
**Alternative / Exception:**

* Lesson không tồn tại → 404 ERR\_NOT\_FOUND.
* Dữ liệu không hợp lệ → 400 ERR\_VALIDATION (list errors).
* Không có quyền → 403 ERR\_FORBIDDEN.

**Data touched:** Lessons, Courses, Outbox (nếu dùng), audit tables.

**API (ví dụ):** POST /api/v1/courses/{courseId}/lessons, PUT /api/v1/lessons/{lessonId}, DELETE /api/v1/lessons/{lessonId}.

**Diagram tham chiếu:**



Hình 28. Sequence Diagram Quản lý bài học trong khóa học

**Mô tả & Sequence**: Giảng viên thêm/sửa/xoá lesson. Pattern: UI → Application (validation, auth) → Domain → Infra (repositories). DELETE semantics: soft-delete.

## Use Case: Tạo bài kiểm tra (CreateQuiz)

**Mục tiêu (Goal):** Cho phép Instructor tạo Quiz (kèm Questions/Options), cấu hình AttemptsAllowed, DurationSec, ShuffleAnswers.

**Preconditions:**

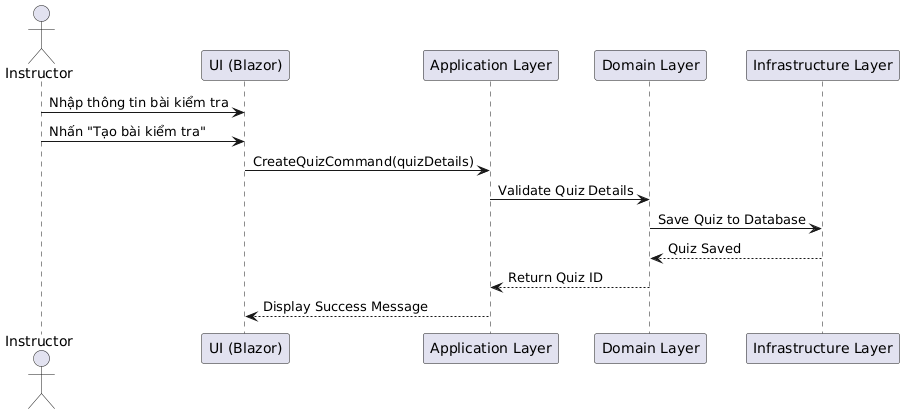
* User là Instructor có quyền trên Course.
* Payload gồm title, questions, settings ở định dạng hợp lệ.

**Main Flow:**

1. UI gửi POST /api/v1/quizzes với CreateQuizCommand.
2. Application validate schema + authorize (IAuthService).
3. Domain kiểm tra business rules (min questions, question types, total score).
4. IQuizRepository.Add(quizAggregate) persist Quiz + Questions/Options (kèm audit fields).
5. Commit UnitOfWork; (tùy) ghi Domain Event QuizCreated vào Outbox.
6. Trả 201 Created với { data: { id: guid } }.

**Alternative / Exception:**

* Payload sai → 400 ERR\_VALIDATION.
* Không có quyền → 403 ERR\_FORBIDDEN.

**Data touched:** Quizzes, Questions, Options, Outbox.  
**API:** POST /api/v1/quizzes.  
**Diagram tham chiếu:**

Hình 29. Sequence Diagram Tạo bài kiểm tra trong khóa học

**Mô tả & Sequence**: Giảng viên tạo quiz; tương tự Create Course: UI → Application → Domain → Infra. Lưu ý: set AttemptsAllowed, DurationSec, ShuffleAnswers.

## Use Case: Xem tiến độ học tập (ViewProgress)

**Mục tiêu (Goal):** Cho phép Learner hoặc Instructor/Admin xem báo cáo tiến độ (CourseProgressReport) cho course hoặc user.

**Preconditions:**

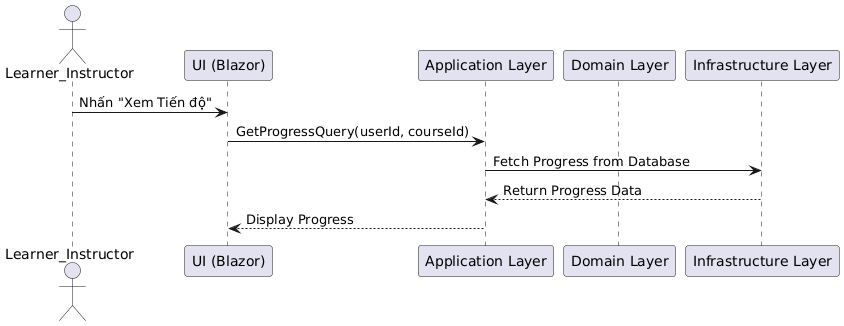
Learner phải đã enroll để xem tiến độ cá nhân; Instructor/Admin có quyền xem học viên.

**Main Flow:**

1. UI gọi GET /api/v1/courses/{id}/progress (hoặc GET /api/v1/users/{userId}/progress).
2. Application authorize request.
3. Service/Domain tổng hợp dữ liệu từ Progresses, QuizAttempts, Enrollments. (Có thể dùng cache/aggregation cho performance.)
4. Infra query dữ liệu tối ưu (indexes) và trả CourseProgressReportDto gồm percentCompleted, lessonStatuses, quizSummaries.
5. Trả ApiResponse kèm traceId.

**Alternative / Exception:**

Không có dữ liệu → trả báo cáo rỗng hoặc 204 theo policy.

**Data touched:** Progresses, QuizAttempts, Enrollments, Courses.  
**API:** GET /api/v1/courses/{id}/progress.  
**Diagram tham chiếu:** 

Hình 30. Sequence Diagram Xem tiến độ học tập

**Mô tả & Sequence:** UI request → Application → Domain computes/aggregates → Infra queries → trả về CourseProgressReportDto hoặc tương tự.

## Use Case: Học bài & Ghi nhận tiến độ (LessonProgress)

**Mục tiêu (Goal):** Khi Learner mở lesson, tạo/cập nhật LessonProgress (draft/started); khi hoàn thành, cập nhật CompletedAt và percent.

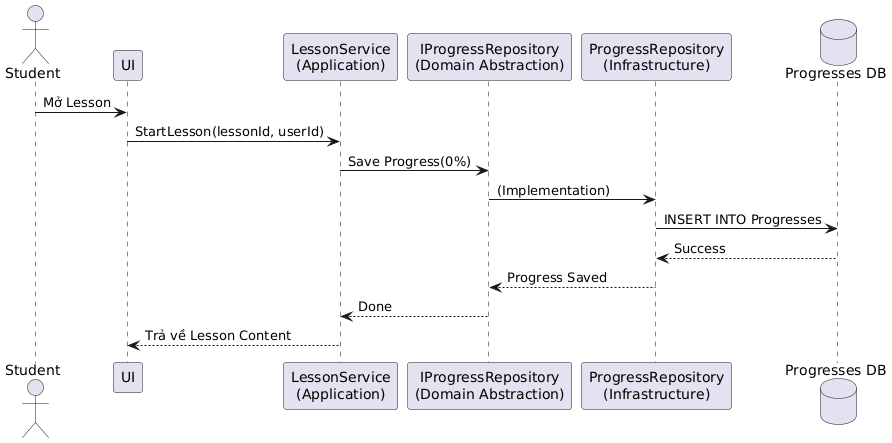
**Preconditions:** Learner đã enroll; lesson tồn tại.

**Main Flow:**

1. Khi mở lesson, UI gọi POST /api/v1/lessons/{id}/progress/start.
2. Application authorize và tạo LessonProgress nếu chưa tồn tại (Status = Started, StartedAt = UtcNow).
3. Khi Learner hoàn thành, UI gọi POST /api/v1/lessons/{id}/progress/complete.
4. Service cập nhật CompletedAt, Percent = 100, commit.
5. Trigger recompute course percent nếu cần; ghi ProgressUpdated event.
6. Trả updated progress DTO.

**Alternative / Exception:**

* Nếu đã completed trước đó → trả 200 OK (idempotent).

**Data touched:** LessonProgress, Enrollments, Courses.  
**Diagram tham chiếu:**

Hình 31. Sequence Diagram Học bài và lưu tiến độ học tập

**Mô tả**: Khi mở lesson, hệ thống có thể auto-create/update Progress (nháp/started); khi người học hoàn thành, update percent/CompletedAt.

## Use Case: Gửi thông báo (SendNotification)

**Mục tiêu (Goal):** Cho phép Instructor/Admin gửi Notification tới một hoặc nhiều Learner; hỗ trợ gửi đồng bộ hoặc ghi Outbox để xử lý bất đồng bộ.

**Preconditions:**

* Sender logged in và có quyền (CourseOwner hoặc Admin).
* Nếu toUserId chỉ định: recipient tồn tại.

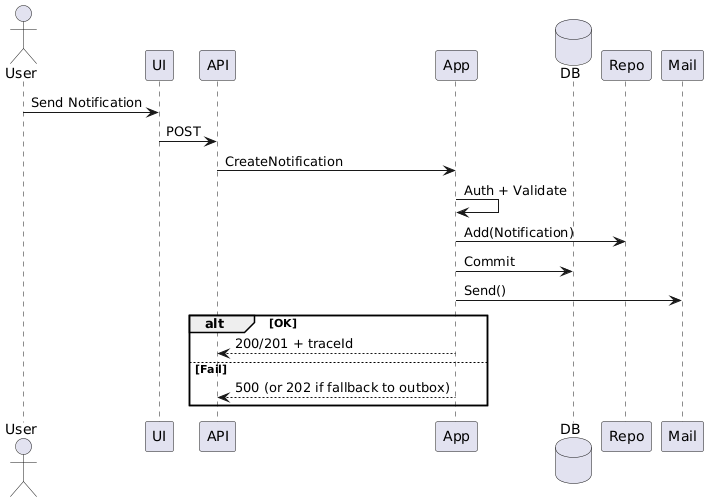
**Main Flow:**

1. UI gửi POST /api/v1/notifications với body { toUserId?, courseId?, title, body, attachments? } (có thể kèm Idempotency-Key).
2. API Controller truyền CreateNotificationCommand đến NotificationService.
3. Application validate, authorize qua IAuthService và kiểm tra quota/rate-limits.
4. Domain tạo Notification entity; gọi INotificationRepository.Add(notification) trong UnitOfWork.
5. Nếu cấu hình Outbox: persist một Outbox message chứa payload delivery trong cùng transaction.
6. Commit UnitOfWork.
7. Post-commit: background worker đọc Outbox và thực hiện delivery (email/push); nếu synchronous send được cấu hình, gọi IEmailSender.Send(...) (cẩn trọng latency/3rd-party errors).
8. Trả ApiResponse: 200/201 (synchronous) hoặc 202 Accepted (asynchronous) kèm traceId.

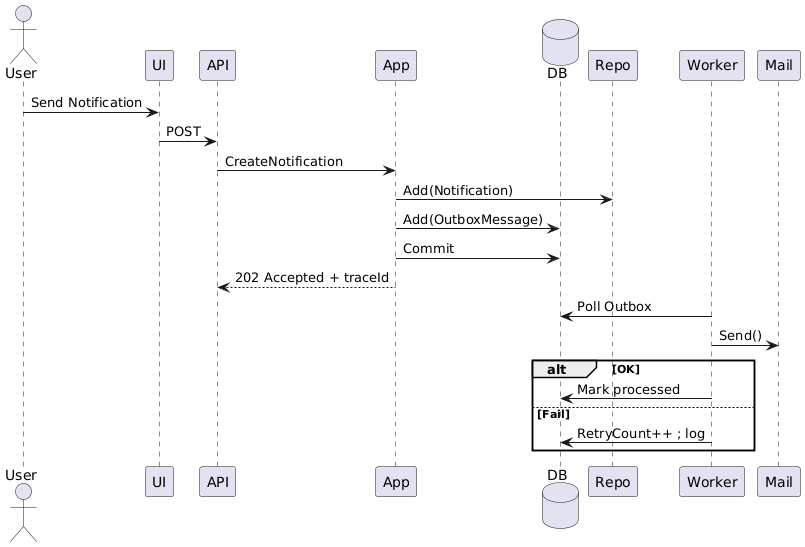
**Alternative / Exception:**

* No permission → 403.
* Recipient not found → 400/404.
* Quota exceeded → 429.
* Synchronous delivery failure → theo policy: 500 hoặc fallback persist Outbox + trả 202.

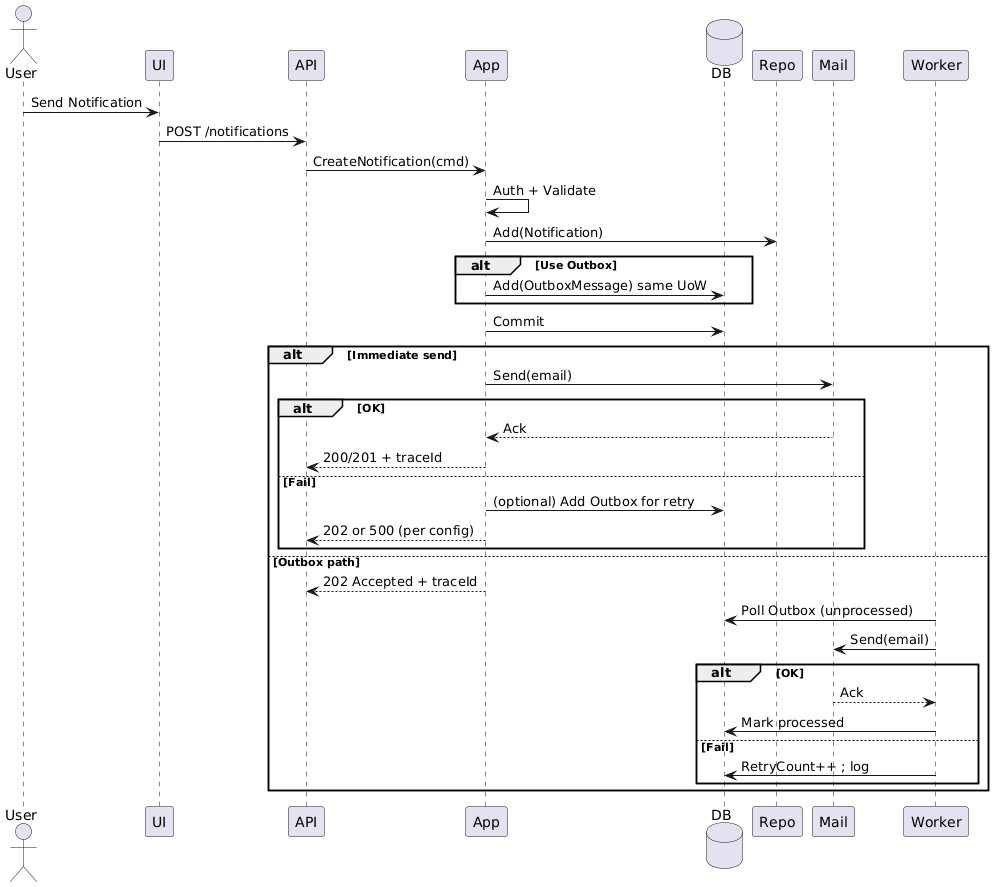
**Data touched:** Notifications, AspNetUsers (read), Outbox, Email subsystem.  
**API:** POST /api/v1/notifications.  
**Diagram tham chiếu:**

****

Hình 32. Immediate Send Notify E-Mail

****

Hình 33. Outbox Send Notify E-Mail



Hình 34. Full Option Send Notification E-Mail (Outbox + Immediately)

## Use Case: Quản lý tài nguyên học tập (FileAssets)

**Mục tiêu (Goal):** Cho phép Instructor upload/cập nhật/quản lý tài nguyên học tập; lưu file trên Blob Storage, metadata trên DB FileAssets.

**Preconditions:**

* Instructor logged in & authorized.
* File hợp lệ (MIME type, size, pass virus-scan).

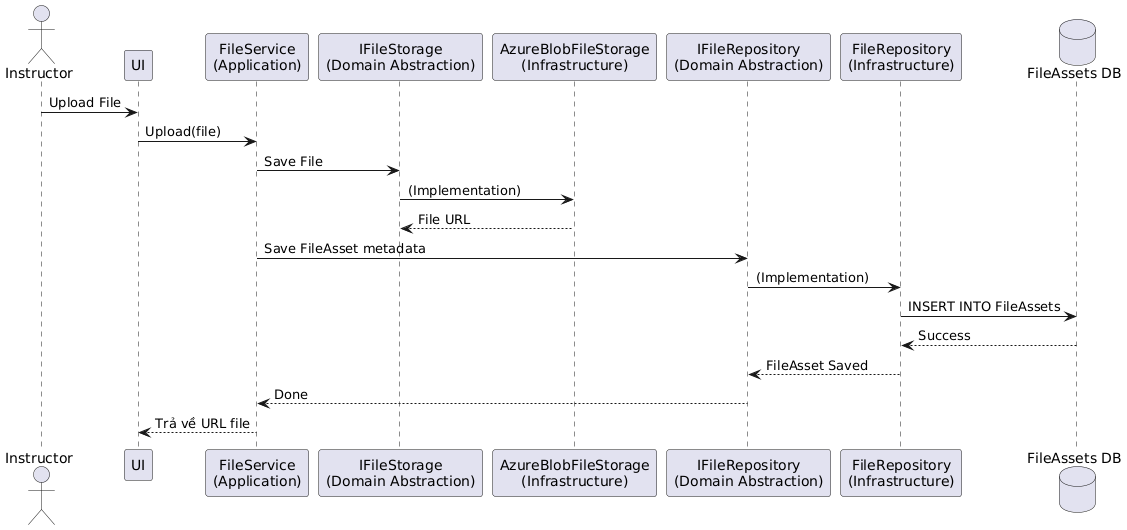
**Main Flow:**

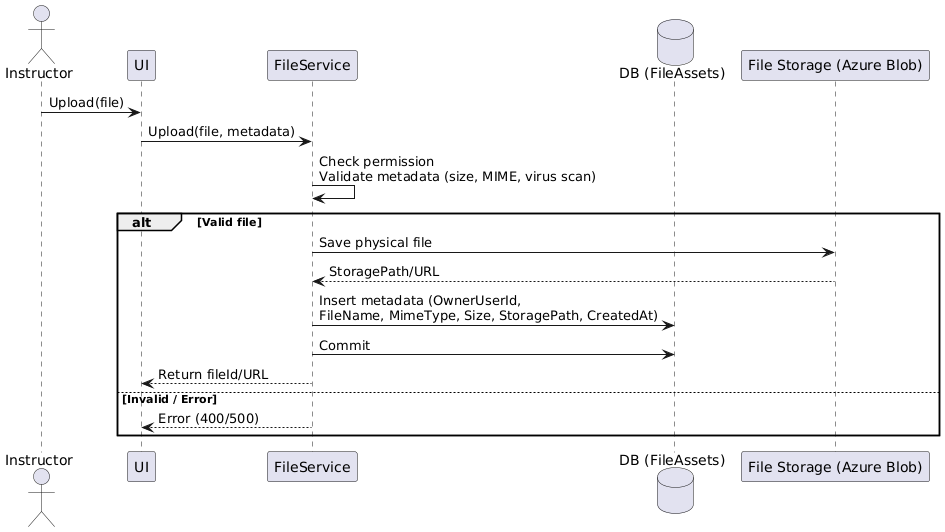
1. UI gửi POST /api/v1/files (multipart) kèm metadata và optional Idempotency-Key.
2. API forward tới FileService. FileService validate size/MIME/quota và run virus-scan nếu có.
3. Lưu file vật lý vào Blob Storage (private container); nhận storagePath/signedUrl.
4. Persist metadata vào FileAssets (OwnerUserId, FileName, MimeType, Size, StoragePath, Hash(SHA256), CreatedAt).
5. Commit UnitOfWork.
6. Trả 201 Created với fileId + access URL (signed URL).

**Alternative / Exception:**

* File too large / invalid MIME → 400.
* Storage failure → rollback và 500.
* Quota exceeded → 403/429.

**Data touched:** FileAssets, Blob Storage, Outbox (nếu cần).  
**API:** POST /api/v1/files.  
**Diagram tham chiếu:**



Hình 35. Sequence Diagram Quản lý tài nguyên học tập

Hình 36 Quản lý tài nguyên học tập (Rút gọn)

## Use Case: Quản lý secrets, upload & security headers (Secrets & Upload Security)

**Mục tiêu (Goal):** Đảm bảo secrets được quản lý an toàn; upload file an toàn; web app có security headers chuẩn.

**Preconditions:** Production environment; Key Vault / secret store đã cấu hình.

**Main Flow:**

* **Secrets:** Sử dụng Azure Key Vault (hoặc tương đương) + Managed Identity; không lưu secrets trong appsettings.production; rotation policy (ví dụ 90 ngày); audit access.
* **Upload security:** Server-side MIME sniffing; size limits; virus-scan (ClamAV/Defender); compute SHA256; lưu file vào private Blob Storage; cung cấp signed URL ngắn hạn.
* **Security headers:** CSP, HSTS, X-Frame-Options (DENY), X-Content-Type-Options (nosniff), Referrer-Policy (no-referrer).

**Alternative / Exception:**

* Secret expired → trigger rotate procedure.
* File fails virus-scan → reject upload (400).

**Data touched:** KeyVault, Blob Storage, FileAssets metadata.  
**Diagram tham chiếu:** N/A

## Use Case: Observability, SLO & Alerting

**Mục tiêu (Goal):** Định nghĩa & giám sát SLOs; thu thập metrics; cấu hình logging/tracing và alert để đáp ứng vận hành.

**Preconditions:** Monitoring stack (Prometheus/Grafana, ELK/EFK, Jaeger/OTel) đã tích hợp.

**Main Flow:**

1. Định nghĩa SLOs (ví dụ availability 99.9%, p95 latency target).
2. Thu thập metrics: RPS per endpoint, error rate per status, p95/p99 latencies, DB slow queries, outbox backlog.
3. Structured logging: JSON logs gồm traceId/correlationId, anonymized userId, route, duration, outcome.
4. Distributed tracing: propagate traceId qua gateway → backend → workers.
5. Alerting: thresholds + runbooks + escalation paths.
6. Retention policy: logs/metrics retention (ví dụ ≥ 90 ngày cho logs quan trọng).

**Alternative / Exception:**

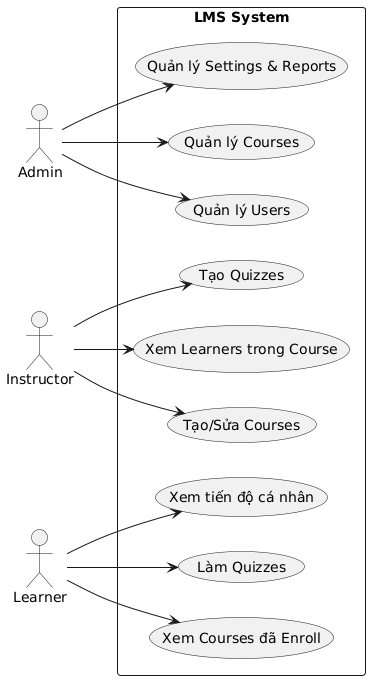
* Alert breach → trigger runbook & escalation.

**Data touched:** Logs store (ELK/EFK), Metrics DB (Prometheus), Traces (Jaeger/OTel).  
**Diagram tham chiếu:** N/A

# Quy tắc bảo mật (Security rules)

## RBAC và Policies

* Roles
  + Admin: Quản lý users, courses, settings, reports
  + Instructor: Tạo/sửa courses do mình tạo, xem learners của courses đó, tạo quizzes
  + Learner: Xem courses đã ghi danh, làm quizzes, xem tiến độ cá nhân
* Authorization Policies
  + CourseOwnerOrAdmin: Cho phép nếu user.IsInRole(Admin) hoặc user.IsInRole(Instructor) && course.CreatedBy == userId
  + EnrolledOnly: Chỉ cho phép truy cập lesson/quiz nếu tồn tại Enrollment với (CourseId, UserId) tương ứng
* Áp dụng
  + Handlers kiểm tra policy theo courseId/moduleId/lessonId
  + Với CourseOwnerOrAdmin: query course để đối chiếu CreatedBy
  + Với EnrolledOnly: kiểm tra ràng buộc UQ Enrollments (CourseId, UserId)



Hình 37. Roles security

## Authentication và Session

* Blazor Server: Cookie Authentication (ASP.NET Identity); CookieOptions: SameSite=Strict, Secure
* API-first (tương lai): Có thể dùng JWT khi chuyển sang Blazor WASM

## Input Validation và Output Encoding

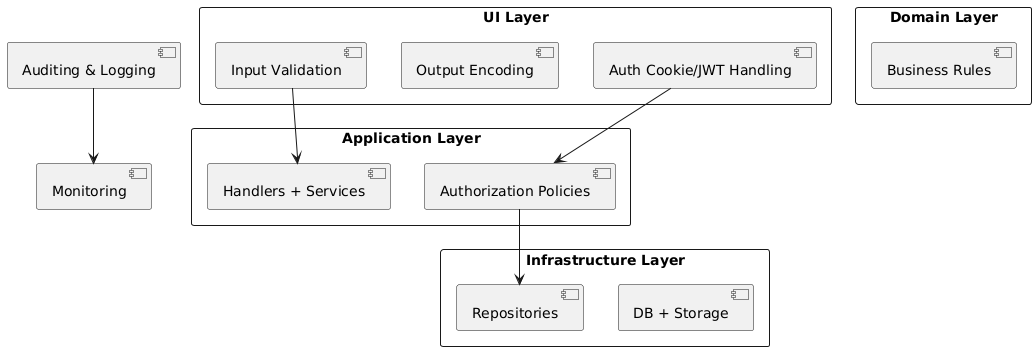
* Validation: Sử dụng FluentValidation trên Commands/DTOs
* Encoding: Blazor auto-encode text; với rich HTML thì sanitize server-side (ví dụ Ganss.XSS)

## Protection Against Common Attacks

* SQL Injection: EF Core parameterized queries / LINQ
* XSS: Sanitize rich HTML; áp dụng CSP
* CSRF: Anti-forgery tokens cho endpoints dùng cookie
* Rate Limiting: Giới hạn login, ví dụ 5 attempts/15 phút
* Security Headers:
  + X-Frame-Options: DENY
  + Referrer-Policy: no-referrer
  + Strict-Transport-Security: max-age=31536000; includeSubDomains
  + Content-Security-Policy hợp lý
* File Upload: Validate loại và kích thước file; lưu tại private blob storage; trả signed URLs ngắn hạn

## Auditing và Logging

* Audit: Ghi log các hành động quan trọng (admin create/delete course, enrollment changes, role changes); lưu vào AuditLogs table hoặc Application Insights
* Correlation: Ghi traceId để liên kết log và sự cố



Hình 38. Component Diagram - Security Layer

# ****Non-Functional Requirements (NFRs)****

Hệ thống LMS-Mini ngoài việc đáp ứng chức năng còn cần thoả mãn các yêu cầu phi chức năng sau:

## Hiệu năng (Performance)

* Hỗ trợ tối thiểu **1,000 concurrent users** trên môi trường production.
* Thời gian phản hồi trung bình cho API < **500ms** với payload nhỏ (<50KB).
* Các thao tác ghi (enroll, submit quiz, update progress) phải được xử lý trong vòng < **1 giây** ở điều kiện tải bình thường.
* Hệ thống cần hỗ trợ truy vấn báo cáo trên tập dữ liệu **>1 triệu bản ghi** mà vẫn đảm bảo thời gian phản hồi < **5s**.

## Khả năng mở rộng (Scalability)

* Hệ thống thiết kế theo hướng **stateless API** để có thể scale-out theo chiều ngang (horizontal scaling) bằng cách bổ sung thêm instance.
* Message broker và DB phải có khả năng scale độc lập.
* Blob Storage hỗ trợ lưu trữ **tối thiểu 1TB** nội dung học tập.

## Tính sẵn sàng & Tin cậy (Availability & Reliability)

* Mức độ khả dụng (Availability) tối thiểu **99.9%/tháng**.
* Tất cả giao dịch quan trọng (Enroll, Submit Quiz, Progress update) phải đảm bảo **ACID** và cơ chế retry (Outbox + Idempotency).
* Worker xử lý Outbox phải có chính sách retry với backoff.

## Bảo mật & Tuân thủ (Security & Compliance)

* Mọi dữ liệu cá nhân (PII) phải được mã hoá **at-rest** và **in-transit** (TLS 1.2+).
* Session/Token phải tuân thủ chuẩn **OWASP ASVS**.
* Tuân thủ các quy định **GDPR/FERPA** liên quan đến dữ liệu học viên (nếu áp dụng quốc tế).

## Khả năng quan sát & giám sát (Observability)

* Toàn bộ API cần sinh log có **traceId** để dễ dàng correlation.
* Metrics: request/second, error rate, DB query time phải được xuất ra Prometheus/Grafana.
* Alerting: cảnh báo khi error rate >5% trong 5 phút, queue backlog >1000 message, DB CPU >80%.

# ****Deployment & Infrastructure View****

## Kiến trúc triển khai

* **Web Frontend (Blazor Server):** chạy trong Azure App Service / Container.
* **Backend API:** ASP.NET Core Web API, chạy trong cluster (Kubernetes/Docker Swarm).
* **Database:** SQL Server (Azure SQL Database), cấu hình HA (geo-replication).
* **Blob Storage:** Azure Blob Storage (hot + archive tier).
* **Message Broker:** RabbitMQ hoặc Azure Service Bus.
* **Background Workers:** chạy trong container riêng, tiêu thụ Outbox/Events.

## Deployment Diagram

Mô tả sơ đồ (UML Deployment):

* **Clients (Browser)** → HTTPS → **Load Balancer (API Gateway)**
* → **App Service / Kubernetes Pods (Backend API + Blazor Server)**
* → **SQL Database Cluster**
* → **Blob Storage**
* → **Message Broker (RabbitMQ / Service Bus)**
* → **Background Worker Pods**

## CI/CD Pipeline

* Build & Test: GitHub Actions / Azure DevOps Pipelines.
* Deploy staging: tự động sau merge.
* Deploy production: yêu cầu manual approval.

# ****Error Handling & Exception Flow****

## Nguyên tắc chung

* Tất cả API trả về trong **Response Envelope chuẩn**.
* Exception được map sang error code (ERR\_xxx).
* Ghi nhận audit log cho mọi exception nghiêm trọng.

## Error Codes (mở rộng)

* **ERR\_ENROLL\_DUPLICATE**: User đã ghi danh trước đó.
* **ERR\_QUIZ\_EXPIRED**: Quiz đã hết hạn, không thể nộp.
* **ERR\_FILE\_TOO\_LARGE**: Upload vượt quá quota.
* **ERR\_CONCURRENCY\_CONFLICT**: RowVersion mismatch.
* **ERR\_UNAUTHORIZED\_ACTION**: User không đủ quyền.

## Exception Flow (ví dụ)

* **Enroll Course:** nếu user đã ghi danh → trả 409 Conflict kèm error code.
* **Submit Quiz:** nếu quiz hết hạn → trả 400 BadRequest kèm error code.
* **Update Lesson:** nếu RowVersion mismatch → trả 412 Precondition Failed.

# ****Testing Strategy****

## Unit Test

* Bao phủ ≥ **80% domain services** (Course, Enrollment, Quiz).
* Test business rule: không cho enroll trùng, không cho submit quiz quá hạn.

## Integration Test

* Test API contract bằng **REST-assured / HttpClientFactory**.
* Kết hợp DB in-memory hoặc SQL container.

## Load Test

* Công cụ: **k6 / JMeter**.
* Kịch bản: 1,000 concurrent enrolls, 500 concurrent quiz submissions.
* Đảm bảo latency P95 < 1s.

## Security Test

* Kiểm thử SQL Injection, XSS, CSRF, IDOR.
* Dùng **OWASP ZAP / BurpSuite**.

## Acceptance Test

* Test theo từng Use Case trong mục 8 (Enroll, Submit Quiz, Progress).

# ****Backup & Disaster Recovery****

## Backup Policy

* Database backup **hằng ngày** (full) + **transaction log mỗi 15 phút**.
* Blob Storage bật **Soft Delete** và **Versioning**.

## Recovery Objective

* **RPO (Recovery Point Objective):** tối đa 15 phút dữ liệu mất mát.
* **RTO (Recovery Time Objective):** hệ thống khôi phục < 2 giờ.

## Disaster Recovery Plan

* Nếu SQL chính down → failover sang geo-replica.
* Nếu message broker mất → Outbox tiếp tục retry sau khi hệ thống hồi phục.
* Định kỳ chạy drill (test khôi phục) 6 tháng/lần.

# Coding style và vận hành

* **Namespaces:** Lms.Domain, Lms.Application, Lms.Infrastructure, Lms.Web.
* **Async:** Tất cả IO methods kết thúc Async.
* **DTOs:** CreateXRequest, XDto, XResponse.
* **Handlers:** CreateCourseCommandHandler implements IRequestHandler<CreateCourseCommand, Guid>.
* **Migrations:** Định danh yyyyMMddHHmm\_initial\_courses, v.v.
* **Tests:** Unit cho Domain; Integration cho repository (test DB); bUnit cho Blazor components.
* **CI/CD:** Static analysis, unit tests, security scanning; deploy staging trước prod.

# Deliverables / Output (Đầu ra mong muốn)

1. **Architecture & Design**

* **Architecture diagrams**: Logical + Physical (PlantUML/Draw.io source).
* **Sequence diagrams**: các luồng nghiệp vụ chính (Enrollment, Quiz, Notification, FileAssets, Lesson Progress, ) kèm PlantUML.
* **Component diagram**: mô tả quan hệ giữa UI, Application Layer, Domain Layer, Infrastructure Layer (nếu đã thống nhất pattern).

1. **Data Model**

* **ERD**: sơ đồ quan hệ thực thể.
* **DDL**: script SQL định nghĩa bảng (Courses, Enrollments, Quizzes, Attempts, Lessons, Progress, Notifications, FileAssets, AuditLogs, ).
* **Seed strategy**: pseudo-code DbInitializer để sinh dữ liệu mẫu.

1. **API Specification**

* **API endpoints**: bảng liệt kê endpoint, method, request/response.
* **DTOs**: định nghĩa chuẩn đặt tên (CreateXRequest, XDto, XResponse) + ví dụ minh họa.

1. **UI/UX**

* **UI wireframes**: mô tả text + danh sách components; tuỳ chọn prototype HTML/CSS.
* **User flows**: gắn với sequence diagrams (Enroll, Quiz, Progress, ).

1. **Security & Operations**

* **Security rules & policies**: RBAC (Admin, Instructor, Learner), authorization policies (CourseOwnerOrAdmin, EnrolledOnly), input validation, chống tấn công (SQLi, XSS, CSRF), security headers.
* **Operational notes**: Coding style (namespaces, async, DTOs, handlers, migrations), CI/CD pipeline, testing strategy (unit, integration, bUnit).
* **Observability & SLOs**: metrics, alerting, rate limiting, logging/auditing.

1. **Checklists & Governance**

* **Checklist PR & pre-acceptance**: tiêu chí review và chấp nhận.
* **Coding style guide**: chuẩn áp dụng xuyên suốt (namespace, naming, async, test, migration).

|  |  |
| --- | --- |
| Deliverable | Nội dung / Hình minh họa |
| Architecture diagrams | Component Diagram |
| Sequence diagrams | Enrollment, Quiz, Notification (PlantUML) |
| Security rules & policies | Component Diagram: Security Layer |
| ERD + DDL | ERD sơ đồ DB + SQL script |
| API endpoints + DTOs | Bảng API + ví dụ JSON |
| UI wireframes | Wireframe màn hình chính (trích xuất từ prototype) |
| Seed strategy | Pseudo-code DbInitializer |
| Checklist PR & Acceptance | Checklist table |

# Phụ lục

## Phụ lục A: Sample SQL DDL Snippets (SQL Server)

### Quy ước chung

* Prefix ràng buộc: PK\_, FK\_, UQ\_, DF\_, IX\_.
* Thứ tự khai báo: CREATE TABLE (chỉ cột) → PK → FK → UQ → DF → INDEX.
* Mọi thời gian lưu UTC (SYSUTCDATETIME()); áp dụng Soft Delete & audit theo yêu cầu nghiệp vụ.

### Code SQL DDL Snippets

/\* ===== AspNetUsers ===== \*/

CREATE TABLE dbo.AspNetUsers (

Id UNIQUEIDENTIFIER NOT NULL,

UserName NVARCHAR(256) NOT NULL,

NormalizedUserName NVARCHAR(256) NOT NULL,

Email NVARCHAR(256) NULL,

NormalizedEmail NVARCHAR(256) NULL,

EmailConfirmed BIT NOT NULL CONSTRAINT DF\_AspNetUsers\_EmailConfirmed DEFAULT (0),

PasswordHash NVARCHAR(MAX) NULL,

SecurityStamp NVARCHAR(MAX) NULL,

ConcurrencyStamp NVARCHAR(MAX) NULL,

PhoneNumber NVARCHAR(50) NULL,

PhoneNumberConfirmed BIT NOT NULL CONSTRAINT DF\_AspNetUsers\_PhoneNumberConfirmed DEFAULT (0),

TwoFactorEnabled BIT NOT NULL CONSTRAINT DF\_AspNetUsers\_TwoFactorEnabled DEFAULT (0),

LockoutEnd DATETIMEOFFSET NULL,

LockoutEnabled BIT NOT NULL CONSTRAINT DF\_AspNetUsers\_LockoutEnabled DEFAULT (0),

AccessFailedCount INT NOT NULL CONSTRAINT DF\_AspNetUsers\_AccessFailedCount DEFAULT (0)

);

ALTER TABLE dbo.AspNetUsers ADD CONSTRAINT PK\_AspNetUsers PRIMARY KEY (Id);

ALTER TABLE dbo.AspNetUsers ADD CONSTRAINT UQ\_AspNetUsers\_NormalizedUserName UNIQUE (NormalizedUserName);

CREATE INDEX IX\_AspNetUsers\_NormalizedEmail ON dbo.AspNetUsers(NormalizedEmail);

CREATE INDEX IX\_AspNetUsers\_UserName ON dbo.AspNetUsers(UserName);

/\* ===== Courses ===== \*/

CREATE TABLE dbo.Courses (

Id UNIQUEIDENTIFIER NOT NULL,

Code NVARCHAR(50) NOT NULL,

Title NVARCHAR(200) NOT NULL,

Description NVARCHAR(MAX) NULL,

Status NVARCHAR(20) NOT NULL CONSTRAINT DF\_Courses\_Status DEFAULT ('Draft'),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Courses\_CreatedAt DEFAULT (SYSUTCDATETIME()),

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Courses\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.Courses ADD CONSTRAINT PK\_Courses PRIMARY KEY (Id);

ALTER TABLE dbo.Courses ADD CONSTRAINT FK\_Courses\_AspNetUsers FOREIGN KEY (CreatedBy) REFERENCES dbo.AspNetUsers(Id);

-- Chuyển UQ sang filtered UNIQUE INDEX để hỗ trợ soft delete, vẫn giữ tên UQ\_Courses\_Code

CREATE UNIQUE INDEX UQ\_Courses\_Code ON dbo.Courses(Code) WHERE IsDeleted = 0;

CREATE INDEX IX\_Courses\_Status ON dbo.Courses(Status);

CREATE INDEX IX\_Courses\_Title ON dbo.Courses(Title);

/\* ===== Modules ===== \*/

CREATE TABLE dbo.Modules (

Id UNIQUEIDENTIFIER NOT NULL,

CourseId UNIQUEIDENTIFIER NOT NULL,

Title NVARCHAR(200) NOT NULL,

[Order] INT NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Modules\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Modules\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.Modules ADD CONSTRAINT PK\_Modules PRIMARY KEY (Id);

ALTER TABLE dbo.Modules ADD CONSTRAINT FK\_Modules\_Courses FOREIGN KEY (CourseId) REFERENCES dbo.Courses(Id);

CREATE INDEX IX\_Modules\_Course\_Order ON dbo.Modules(CourseId, [Order]);

/\* ===== Lessons ===== \*/

CREATE TABLE dbo.Lessons (

Id UNIQUEIDENTIFIER NOT NULL,

ModuleId UNIQUEIDENTIFIER NOT NULL,

Title NVARCHAR(200) NOT NULL,

ContentType NVARCHAR(50) NOT NULL,

ContentUrl NVARCHAR(1000) NULL,

DurationSec INT NULL,

[Order] INT NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Lessons\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Lessons\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.Lessons ADD CONSTRAINT PK\_Lessons PRIMARY KEY (Id);

ALTER TABLE dbo.Lessons ADD CONSTRAINT FK\_Lessons\_Modules FOREIGN KEY (ModuleId) REFERENCES dbo.Modules(Id);

CREATE INDEX IX\_Lessons\_Module\_Order ON dbo.Lessons(ModuleId, [Order]);

/\* ===== Quizzes ===== \*/

CREATE TABLE dbo.Quizzes (

Id UNIQUEIDENTIFIER NOT NULL,

CourseId UNIQUEIDENTIFIER NOT NULL,

Title NVARCHAR(200) NOT NULL,

DurationSec INT NULL,

AttemptsAllowed INT NULL CONSTRAINT DF\_Quizzes\_AttemptsAllowed DEFAULT (1),

ShuffleAnswers BIT NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Quizzes\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Quizzes\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.Quizzes ADD CONSTRAINT PK\_Quizzes PRIMARY KEY (Id);

ALTER TABLE dbo.Quizzes ADD CONSTRAINT FK\_Quizzes\_Courses FOREIGN KEY (CourseId) REFERENCES dbo.Courses(Id);

CREATE INDEX IX\_Quizzes\_Course ON dbo.Quizzes(CourseId);

/\* ===== Questions ===== \*/

CREATE TABLE dbo.Questions (

Id UNIQUEIDENTIFIER NOT NULL,

QuizId UNIQUEIDENTIFIER NOT NULL,

[Text] NVARCHAR(MAX) NOT NULL,

[Order] INT NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Questions\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Questions\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.Questions ADD CONSTRAINT PK\_Questions PRIMARY KEY (Id);

ALTER TABLE dbo.Questions ADD CONSTRAINT FK\_Questions\_Quizzes FOREIGN KEY (QuizId) REFERENCES dbo.Quizzes(Id);

CREATE INDEX IX\_Questions\_Quiz\_Order ON dbo.Questions(QuizId, [Order]);

/\* ===== Options ===== \*/

CREATE TABLE dbo.Options (

Id UNIQUEIDENTIFIER NOT NULL,

QuestionId UNIQUEIDENTIFIER NOT NULL,

[Text] NVARCHAR(MAX) NOT NULL,

IsCorrect BIT NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Options\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Options\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.Options ADD CONSTRAINT PK\_Options PRIMARY KEY (Id);

ALTER TABLE dbo.Options ADD CONSTRAINT FK\_Options\_Questions FOREIGN KEY (QuestionId) REFERENCES dbo.Questions(Id);

CREATE INDEX IX\_Options\_Question ON dbo.Options(QuestionId);

/\* ===== QuizAttempts (không audit/soft delete) ===== \*/

CREATE TABLE dbo.QuizAttempts (

Id UNIQUEIDENTIFIER NOT NULL,

QuizId UNIQUEIDENTIFIER NOT NULL,

UserId UNIQUEIDENTIFIER NOT NULL,

StartedAt DATETIME2 NOT NULL,

SubmittedAt DATETIME2 NULL,

Score DECIMAL(5,2) NULL CONSTRAINT DF\_QuizAttempts\_Score DEFAULT (0)

);

ALTER TABLE dbo.QuizAttempts ADD CONSTRAINT PK\_QuizAttempts PRIMARY KEY (Id);

ALTER TABLE dbo.QuizAttempts ADD CONSTRAINT FK\_QuizAttempts\_Quizzes FOREIGN KEY (QuizId) REFERENCES dbo.Quizzes(Id);

ALTER TABLE dbo.QuizAttempts ADD CONSTRAINT FK\_QuizAttempts\_AspNetUsers FOREIGN KEY (UserId) REFERENCES dbo.AspNetUsers(Id);

CREATE INDEX IX\_QuizAttempts\_Quiz\_User ON dbo.QuizAttempts(QuizId, UserId);

/\* ===== AttemptAnswers (không audit/soft delete) ===== \*/

CREATE TABLE dbo.AttemptAnswers (

Id UNIQUEIDENTIFIER NOT NULL,

AttemptId UNIQUEIDENTIFIER NOT NULL,

QuestionId UNIQUEIDENTIFIER NOT NULL,

OptionId UNIQUEIDENTIFIER NOT NULL

);

ALTER TABLE dbo.AttemptAnswers ADD CONSTRAINT PK\_AttemptAnswers PRIMARY KEY (Id);

ALTER TABLE dbo.AttemptAnswers ADD CONSTRAINT FK\_AttemptAnswers\_QuizAttempts FOREIGN KEY (AttemptId) REFERENCES dbo.QuizAttempts(Id);

ALTER TABLE dbo.AttemptAnswers ADD CONSTRAINT FK\_AttemptAnswers\_Questions FOREIGN KEY (QuestionId) REFERENCES dbo.Questions(Id);

ALTER TABLE dbo.AttemptAnswers ADD CONSTRAINT FK\_AttemptAnswers\_Options FOREIGN KEY (OptionId) REFERENCES dbo.Options(Id);

ALTER TABLE dbo.AttemptAnswers ADD CONSTRAINT UQ\_AttemptAnswers\_AttemptId\_QuestionId UNIQUE (AttemptId, QuestionId);

/\* ===== Enrollments ===== \*/

CREATE TABLE dbo.Enrollments (

Id UNIQUEIDENTIFIER NOT NULL,

CourseId UNIQUEIDENTIFIER NOT NULL,

UserId UNIQUEIDENTIFIER NOT NULL,

StartAt DATETIME2 NOT NULL,

EndAt DATETIME2 NULL,

Status NVARCHAR(20) NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Enrollments\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Enrollments\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.Enrollments ADD CONSTRAINT PK\_Enrollments PRIMARY KEY (Id);

ALTER TABLE dbo.Enrollments ADD CONSTRAINT FK\_Enrollments\_Courses FOREIGN KEY (CourseId) REFERENCES dbo.Courses(Id);

ALTER TABLE dbo.Enrollments ADD CONSTRAINT FK\_Enrollments\_AspNetUsers FOREIGN KEY (UserId) REFERENCES dbo.AspNetUsers(Id);

-- Filtered unique index (giữ tên gốc)

CREATE UNIQUE INDEX UQ\_Enrollments\_CourseId\_UserId ON dbo.Enrollments(CourseId, UserId) WHERE IsDeleted = 0;

CREATE INDEX IX\_Enrollments\_User ON dbo.Enrollments(UserId);

CREATE INDEX IX\_Enrollments\_Course\_Status ON dbo.Enrollments(CourseId, Status);

/\* ===== Progresses ===== \*/

CREATE TABLE dbo.Progresses (

Id UNIQUEIDENTIFIER NOT NULL,

UserId UNIQUEIDENTIFIER NOT NULL,

LessonId UNIQUEIDENTIFIER NOT NULL,

CompletedAt DATETIME2 NULL,

[Percent] TINYINT NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Progresses\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Progresses\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.Progresses ADD CONSTRAINT PK\_Progresses PRIMARY KEY (Id);

ALTER TABLE dbo.Progresses ADD CONSTRAINT FK\_Progresses\_AspNetUsers FOREIGN KEY (UserId) REFERENCES dbo.AspNetUsers(Id);

ALTER TABLE dbo.Progresses ADD CONSTRAINT FK\_Progresses\_Lessons FOREIGN KEY (LessonId) REFERENCES dbo.Lessons(Id);

ALTER TABLE dbo.Progresses ADD CONSTRAINT UQ\_Progresses\_UserId\_LessonId UNIQUE (UserId, LessonId);

CREATE INDEX IX\_Progresses\_User ON dbo.Progresses(UserId);

/\* ===== Notifications ===== \*/

CREATE TABLE dbo.Notifications (

Id UNIQUEIDENTIFIER NOT NULL,

CourseId UNIQUEIDENTIFIER NULL,

ToUserId UNIQUEIDENTIFIER NULL,

Title NVARCHAR(200) NOT NULL,

Body NVARCHAR(MAX) NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_Notifications\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_Notifications\_IsDeleted DEFAULT (0),

SentBy UNIQUEIDENTIFIER NOT NULL

);

ALTER TABLE dbo.Notifications ADD CONSTRAINT PK\_Notifications PRIMARY KEY (Id);

ALTER TABLE dbo.Notifications ADD CONSTRAINT FK\_Notifications\_Courses FOREIGN KEY (CourseId) REFERENCES dbo.Courses(Id);

ALTER TABLE dbo.Notifications ADD CONSTRAINT FK\_Notifications\_AspNetUsers\_ToUser FOREIGN KEY (ToUserId) REFERENCES dbo.AspNetUsers(Id);

ALTER TABLE dbo.Notifications ADD CONSTRAINT FK\_Notifications\_AspNetUsers\_SentBy FOREIGN KEY (SentBy) REFERENCES dbo.AspNetUsers(Id);

CREATE INDEX IX\_Notifications\_ToUserId ON dbo.Notifications(ToUserId);

/\* ===== FileAssets ===== \*/

CREATE TABLE dbo.FileAssets (

Id UNIQUEIDENTIFIER NOT NULL,

OwnerUserId UNIQUEIDENTIFIER NOT NULL,

FileName NVARCHAR(255) NOT NULL,

MimeType NVARCHAR(100) NOT NULL,

Size BIGINT NOT NULL,

StoragePath NVARCHAR(1000) NOT NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_FileAssets\_CreatedAt DEFAULT (SYSUTCDATETIME()),

CreatedBy UNIQUEIDENTIFIER NOT NULL,

UpdatedAt DATETIME2 NULL,

UpdatedBy UNIQUEIDENTIFIER NULL,

IsDeleted BIT NOT NULL CONSTRAINT DF\_FileAssets\_IsDeleted DEFAULT (0)

);

ALTER TABLE dbo.FileAssets ADD CONSTRAINT PK\_FileAssets PRIMARY KEY (Id);

ALTER TABLE dbo.FileAssets ADD CONSTRAINT FK\_FileAssets\_AspNetUsers FOREIGN KEY (OwnerUserId) REFERENCES dbo.AspNetUsers(Id);

CREATE INDEX IX\_FileAssets\_OwnerUserId ON dbo.FileAssets(OwnerUserId);

/\* ===== OutboxMessages (theo SDD – Outbox Pattern) ===== \*/

CREATE TABLE dbo.OutboxMessages (

Id UNIQUEIDENTIFIER NOT NULL,

OccurredOn DATETIME2 NOT NULL CONSTRAINT DF\_OutboxMessages\_OccurredOn DEFAULT (SYSUTCDATETIME()),

[Type] NVARCHAR(200) NOT NULL,

Payload NVARCHAR(MAX) NOT NULL,

ProcessedOn DATETIME2 NULL,

Error NVARCHAR(MAX) NULL,

RetryCount INT NOT NULL CONSTRAINT DF\_OutboxMessages\_RetryCount DEFAULT (0)

);

ALTER TABLE dbo.OutboxMessages ADD CONSTRAINT PK\_OutboxMessages PRIMARY KEY (Id);

CREATE INDEX IX\_OutboxMessages\_ProcessedOn ON dbo.OutboxMessages(ProcessedOn);

CREATE TABLE dbo.AuditLogs (

Id UNIQUEIDENTIFIER NOT NULL,

UserId UNIQUEIDENTIFIER NULL,

Action NVARCHAR(200) NOT NULL,

Entity NVARCHAR(200) NOT NULL,

EntityId UNIQUEIDENTIFIER NULL,

Data NVARCHAR(MAX) NULL,

CreatedAt DATETIME2 NOT NULL CONSTRAINT DF\_AuditLogs\_CreatedAt DEFAULT (SYSUTCDATETIME())

);

ALTER TABLE dbo.AuditLogs ADD CONSTRAINT PK\_AuditLogs PRIMARY KEY (Id);

ALTER TABLE dbo.AuditLogs ADD CONSTRAINT FK\_AuditLogs\_AspNetUsers FOREIGN KEY (UserId) REFERENCES dbo.AspNetUsers(Id);

CREATE INDEX IX\_AuditLogs\_UserId ON dbo.AuditLogs(UserId);

CREATE INDEX IX\_AuditLogs\_Entity\_EntityId ON dbo.AuditLogs(Entity, EntityId);

## Phụ lục B – Mã nguồn PlantUML

### System Context Diagram

@startuml

!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4\_Context.puml

title System Context Diagram - LMS-Mini

Person(learner, "Learner", "Người học, tham gia khóa học, làm quiz, theo dõi tiến độ")

Person(instructor, "Instructor", "Giảng viên, tạo & quản lý khóa học, quiz, lesson")

Person(admin, "Admin", "Quản trị hệ thống, phân quyền, giám sát")

Person(qa, "QA Engineer", "Đảm bảo chất lượng, test hệ thống")

Person(devops, "DevOps", "Triển khai & vận hành hệ thống")

System(lms, "LMS-Mini", "Learning Management System thu nhỏ: tạo khoá học, ghi danh, quiz, thông báo")

System\_Ext(db, "Relational Database", "Lưu transactional data")

System\_Ext(storage, "Blob Storage", "Lưu tài liệu, media")

System\_Ext(broker, "Message Broker", "Pub/Sub events")

System\_Ext(worker, "Background Workers", "Xử lý async jobs")

Rel(learner, lms, "Sử dụng qua web/mobile (SPA)")

Rel(instructor, lms, "Tạo & quản lý khóa học")

Rel(admin, lms, "Quản lý & giám sát")

Rel(qa, lms, "Thực hiện test cases")

Rel(devops, lms, "Triển khai & giám sát")

Rel(lms, db, "CRUD data")

Rel(lms, storage, "Upload/Download")

Rel(lms, broker, "Publish/Subscribe events")

Rel(lms, worker, "Giao task async")

SHOW\_LEGEND()

@enduml

### Use-case Diagram - LMS mini (Tham chiếu từ SRS)

|  |
| --- |
| @startuml  ' Use Case diagram for LMS mini  left to right direction  skinparam packageStyle rectangle  actor "Admin" as Admin  actor "Instructor" as Instructor  actor "Learner" as Learner  actor "System\n(Background Worker)" as System  package "Course Management" {  usecase "Create / Update / Publish\nCourse\n(FR001)" as UC\_CreateCourse  usecase "Manage Modules / Lessons\n(FR002)" as UC\_ManageModules  }  package "Learning" {  usecase "Enroll in Course\n(FR003)" as UC\_Enroll  usecase "View Progress\n(FR005)" as UC\_ViewProgress  usecase "Upload / Download File\n(FileAssets)\n(FR007)" as UC\_FileAssets  }  package "Assessment" {  usecase "Start Quiz\n(FR004)" as UC\_StartQuiz  usecase "Submit Quiz\n(FR004)" as UC\_SubmitQuiz  }  package "Communication & Reporting" {  usecase "Send Notification\n(FR006)" as UC\_Notify  usecase "View Reports / Export CSV\n(FR008)" as UC\_Reports  }  ' Actors -> Use cases  Admin --> UC\_CreateCourse  Admin --> UC\_ManageModules  Admin --> UC\_Notify  Admin --> UC\_Reports  Instructor --> UC\_CreateCourse  Instructor --> UC\_ManageModules  Instructor --> UC\_FileAssets  Instructor --> UC\_Notify  Instructor --> UC\_Reports  Learner --> UC\_Enroll  Learner --> UC\_StartQuiz  Learner --> UC\_SubmitQuiz  Learner --> UC\_ViewProgress  Learner --> UC\_FileAssets  ' System (background) interactions  System --> UC\_Notify : "process Outbox / send email"  System --> UC\_Reports : "generate scheduled reports (optional)"  ' Include / extend relationships (logical)  UC\_StartQuiz .> UC\_SubmitQuiz : include  UC\_CreateCourse .> UC\_ManageModules : include  note right of UC\_Enroll  Idempotent: check UQ(CourseId,UserId)  Return 409 if already enrolled (or idempotent response)  end note  note left of UC\_SubmitQuiz  Scoring: auto-grade, save QuizAttempt + AttemptAnswers  Pass threshold example: >= 70%  end note  @enduml |

### Layer Diagram (Clean Architecture)

|  |
| --- |
| @startuml  !theme plain  hide stereotype  package "Presentation Layer" {  [WebApp]  [MobileApp]  }  package "Application Layer" {  [Application Services]  [Use Cases]  }  package "Domain Layer" {  [Entities]  [Domain Services]  [Domain Events]  }  package "Infrastructure Layer" {  [Database]  [File Storage]  [Email/Notification Service]  [External APIs]  }  [WebApp] --> [Application Services]  [MobileApp] --> [Application Services]  [Application Services] --> [Use Cases]  [Use Cases] --> [Entities]  [Use Cases] --> [Domain Services]  [Use Cases] --> [Domain Events]  [Application Services] --> [Database]  [Application Services] --> [File Storage]  [Application Services] --> [Email/Notification Service]  [Application Services] --> [External APIs]  @enduml |

### Layer Diagram (Ví dụ tạo khóa học)

|  |
| --- |
| @startuml  !theme plain  hide stereotype  actor User as U  package "Presentation Layer" {  component "CreateCourse.razor" as Razor  }  package "Application Layer" {  component "CreateCourseCommand" as Command  component "CreateCourseCommandHandler" as Handler  }  package "Domain Layer" {  component "Course Entity" as Entity  }  package "Infrastructure Layer" {  component "CourseRepository" as Repository  component "LmsDbContext" as DbContext  }  U -> Razor : Submit course details  Razor -> Command : CreateCourseCommand  Command -> Handler : Handle(command)  Handler -> Entity : Validate business rules  Handler -> Repository : Save course  Repository -> DbContext : Persist data  @enduml |

### Layer Diagram (Ví dụ Đăng ký khóa học)

|  |
| --- |
| @startuml  !theme plain  hide stereotype  actor User as U  package "Presentation Layer" {  component "EnrollCourse.razor" as Razor  }  package "Application Layer" {  component "EnrollCourseCommand" as Command  component "EnrollCourseCommandHandler" as Handler  }  package "Domain Layer" {  component "Enrollment Entity" as Entity  }  package "Infrastructure Layer" {  component "EnrollmentRepository" as Repository  }  U -> Razor : Click Enroll  Razor -> Command : EnrollCourseCommand  Command -> Handler : Handle(command)  Handler -> Repository : Check enrollment  Handler -> Repository : Save enrollment  @enduml |

### Layer Diagram (Ví dụ Nộp bài kiểm tra)

|  |
| --- |
| @startuml  !theme plain  hide stereotype  actor User as U  package "Presentation Layer" {  component "SubmitQuiz.razor" as Razor  }  package "Application Layer" {  component "SubmitQuizCommand" as Command  component "SubmitQuizCommandHandler" as Handler  }  package "Domain Layer" {  component "Quiz Entity" as Entity  }  package "Infrastructure Layer" {  component "QuizRepository" as Repository  component "ProgressRepository" as ProgressRepo  }  U -> Razor : Submit quiz  Razor -> Command : SubmitQuizCommand  Command -> Handler : Handle(command)  Handler -> Entity : Calculate score  Handler -> Repository : Save quiz attempt  Handler -> ProgressRepo : Update progress  @enduml |

### Query Specification Pattern - LMS-Mini Domain

@startuml

left to right direction

title Query Specification Pattern - LMS-Mini Domain

' ==== Interfaces ====

interface "IQuerySpecification<T>" as IQuerySpecificationT {

+Criteria : Expression<Func<T, bool>>

+OrderBy : Func<IQueryable<T>, IOrderedQueryable<T>>

+Includes : List<Expression<Func<T, object>>>

+Skip : int

+Take : int

}

' ==== Base Specification ====

class "Specification<T>" as SpecificationT implements IQuerySpecificationT {

-criteria

-orderBy

-includes

-skip

-take

+And(spec : IQuerySpecification<T>) : Specification<T>

+Or(spec : IQuerySpecification<T>) : Specification<T>

+Not() : Specification<T>

}

' ==== Domain-specific Specifications ====

class ActiveCoursesSpecification {

+ActiveCoursesSpecification()

}

class CoursesByInstructorSpecification {

+CoursesByInstructorSpecification(instructorId: Guid)

}

class EnrollmentsByUserSpecification {

+EnrollmentsByUserSpecification(userId: Guid)

}

class QuizzesForCourseSpecification {

+QuizzesForCourseSpecification(courseId: Guid)

}

' ==== Repository interface & implementation ====

interface "ICourseRepository" {

+List(spec : IQuerySpecification<Course>) : IEnumerable<Course>

+Get(spec : IQuerySpecification<Course>) : Course

}

interface "IEnrollmentRepository" {

+List(spec : IQuerySpecification<Enrollment>) : IEnumerable<Enrollment>

}

interface "IQuizRepository" {

+List(spec : IQuerySpecification<Quiz>) : IEnumerable<Quiz>

}

class "CourseRepository" implements ICourseRepository {

-dbContext

}

class "EnrollmentRepository" implements IEnrollmentRepository {

-dbContext

}

class "QuizRepository" implements IQuizRepository {

-dbContext

}

' ==== Relationships ====

ActiveCoursesSpecification --|> SpecificationT

CoursesByInstructorSpecification --|> SpecificationT

EnrollmentsByUserSpecification --|> SpecificationT

QuizzesForCourseSpecification --|> SpecificationT

SpecificationT ..|> IQuerySpecificationT

CourseRepository ..|> ICourseRepository

EnrollmentRepository ..|> IEnrollmentRepository

QuizRepository ..|> IQuizRepository

CourseRepository --> IQuerySpecificationT : áp dụng spec

EnrollmentRepository --> IQuerySpecificationT : áp dụng spec

QuizRepository --> IQuerySpecificationT : áp dụng spec

@enduml

### Use-Case diagram – Roles & Permissions

|  |
| --- |
| @startuml  left to right direction  actor Admin  actor Instructor  actor Learner  rectangle System {  usecase "Manage Users\nCourses\nSettings\nReports" as UC\_Admin  usecase "Create/Edit Own Courses" as UC\_InstructorCourse  usecase "View Learners in Own Courses" as UC\_InstructorLearners  usecase "Create Quizzes" as UC\_InstructorQuiz  usecase "View Enrolled Courses" as UC\_LearnerCourses  usecase "Take Quizzes" as UC\_LearnerQuiz  usecase "View Own Progress" as UC\_LearnerProgress  }  Admin --> UC\_Admin  Instructor --> UC\_InstructorCourse  Instructor --> UC\_InstructorLearners  Instructor --> UC\_InstructorQuiz  Learner --> UC\_LearnerCourses  Learner --> UC\_LearnerQuiz  Learner --> UC\_LearnerProgress  @enduml |

### Component Diagram – Security Layer

|  |
| --- |
| @startuml  rectangle "UI Layer" {  [Input Validation]  [Output Encoding]  [Auth Cookie/JWT Handling]  }  rectangle "Application Layer" {  [Authorization Policies]  [Handlers + Services]  }  rectangle "Domain Layer" {  [Business Rules]  }  rectangle "Infrastructure Layer" {  [Repositories]  [DB + Storage]  }  [Auditing & Logging] --> [Monitoring]  [Input Validation] --> [Handlers + Services]  [Auth Cookie/JWT Handling] --> [Authorization Policies]  [Authorization Policies] --> [Repositories]  @enduml |

### Component Diagram – Domain Design

|  |
| --- |
| @startuml  scale 0.8  skinparam defaultFontSize 12  skinparam nodesep 20  skinparam ranksep 20  left to right direction  title LMS Domain Component Diagram  package "LMS Application Layer" as App {  rectangle CourseService  rectangle QuizService  rectangle EnrollmentService  rectangle NotificationService  }  package "LMS Domain Layer" as Domain {  package "Aggregates" as Aggregates {  rectangle "Course Aggregate\n(Course, Module, Lesson, Quiz)" as CourseAgg  rectangle "Quiz Aggregate\n(Quiz, Question, Option)" as QuizAgg  rectangle "QuizAttempt Aggregate\n(QuizAttempt, AttemptAnswer)" as AttemptAgg  }  package "Entities" as Entities {  rectangle Enrollment  rectangle Progresses  rectangle Notification  rectangle FileAsset  }  rectangle "Domain Services" as DomainServices  rectangle "Domain Events" as DomainEvents  }  package "Infrastructure Layer" as Infra {  rectangle Repositories  rectangle "OutboxMessages Store"  rectangle "AuditLog Store"  rectangle "Message Broker"  rectangle Database  }  ' Relationships  App --> Domain  Domain --> Infra  CourseService --> CourseAgg  QuizService --> QuizAgg  EnrollmentService --> Entities  NotificationService --> Entities  DomainEvents --> "OutboxMessages Store"  "OutboxMessages Store" --> "Message Broker"  DomainEvents --> "AuditLog Store"  Repositories --> Database  @enduml |

### Class Diagram UML – Domain Design

|  |
| --- |
| @startuml  hide circle  hide stereotype  !theme plain  class Course {  +Id: guid  +Code: string  +Title: string  +Description: string?  +Status: string  }  class Module {  +Id: guid  +CourseId: guid  +Title: string  +Order: int  }  class Lesson {  +Id: guid  +ModuleId: guid  +Title: string  +ContentType: string  +ContentUrl: string?  }  class Enrollment {  +Id: guid  +CourseId: guid  +UserId: guid  +Status: string  }  class Progresses {  +Id: guid  +UserId: guid  +LessonId: guid  +CompletedAt: datetime?  }  class Quiz {  +Id: guid  +CourseId: guid  +Title: string  +DurationSec: int?  +AttemptsAllowed: int  }  class QuizAttempt {  +Id: guid  +QuizId: guid  +UserId: guid  +StartedAt: datetime  +FinishedAt: datetime?  +Score: decimal?  }  class AttemptAnswer {  +Id: guid  +QuizAttemptId: guid  +QuestionId: guid  +OptionId: guid?  }  class Question {  +Id: guid  +QuizId: guid  +Text: string  +Order: int  }  class Option {  +Id: guid  +QuestionId: guid  +Text: string  +IsCorrect: bool  }  class FileAsset {  +Id: guid  +OwnerUserId: guid  +FileName: string  +FileUrl: string  }  class Notification {  +Id: guid  +CourseId: guid?  +ToUserId: guid?  +Title: string  +SentBy: guid  }  ' ====== NEW: AuditLogs ======  class AuditLogs {  +Id: bigint  +EntityName: string  +EntityId: bigint  +Action: string  +OldValues: text  +NewValues: text  +PerformedBy: string  +PerformedAt: datetime  +IpAddress: string  +UserAgent: string  }  ' ====== NEW: OutboxMessages ======  class OutboxMessages {  +Id: bigint  +AggregateType: string  +AggregateId: bigint  +Payload: text  +OccurredAt: datetime  +ProcessedAt: datetime?  +Status: string  +RetryCount: int  +ErrorMessage: string?  }  ' ====== RELATIONSHIPS ======  Course "1" -- "many" Module  Module "1" -- "many" Lesson  Course "1" -- "many" Enrollment  Lesson "1" -- "many" Progresses  Course "1" -- "many" Quiz  Quiz "1" -- "many" QuizAttempt  QuizAttempt "1" -- "many" AttemptAnswer  Quiz "1" -- "many" Question  Question "1" -- "many" Option  Course "1" -- "many" Notification  ' Gợi ý: liên kết AuditLogs và OutboxMessages  Course "1" -- "many" AuditLogs : audit  Course "1" -- "many" OutboxMessages : events  @enduml |

### Entity Relationship Diagram (ERD) – Database Design

|  |
| --- |
| @startuml  !theme plain  entity Course {  \* Id : guid  --  Code : string  Title : string  Description : string?  Status : string  }  entity Module {  \* Id : guid  --  CourseId : guid  Title : string  Order : int  }  entity Lesson {  \* Id : guid  --  ModuleId : guid  Title : string  ContentType : string  ContentUrl : string?  }  entity Enrollment {  \* Id : guid  --  CourseId : guid  UserId : guid  Status : string  }  entity Progresses {  \* Id : guid  --  UserId : guid  LessonId : guid  CompletedAt : datetime?  }  entity Quiz {  \* Id : guid  --  CourseId : guid  Title : string  DurationSec : int?  AttemptsAllowed : int  }  entity QuizAttempt {  \* Id : guid  --  QuizId : guid  UserId : guid  StartedAt : datetime  FinishedAt : datetime?  Score : decimal?  }  entity AttemptAnswer {  \* Id : guid  --  QuizAttemptId : guid  QuestionId : guid  OptionId : guid?  }  entity Question {  \* Id : guid  --  QuizId : guid  Text : string  Order : int  }  entity Option {  \* Id : guid  --  QuestionId : guid  Text : string  IsCorrect : bool  }  entity FileAsset {  \* Id : guid  --  OwnerUserId : guid  FileName : string  FileUrl : string  }  entity Notification {  \* Id : guid  --  CourseId : guid?  ToUserId : guid?  Title : string  SentBy : guid  }  ' ====== NEW ENTITY: AuditLogs ======  entity AuditLogs {  \* Id : bigint  --  EntityName : string  EntityId : bigint  Action : string  OldValues : text  NewValues : text  PerformedBy : string  PerformedAt : datetime  IpAddress : string  UserAgent : string  }  ' ====== NEW ENTITY: OutboxMessages ======  entity OutboxMessages {  \* Id : bigint  --  AggregateType : string  AggregateId : bigint  Payload : text  OccurredAt : datetime  ProcessedAt : datetime?  Status : string  RetryCount : int  ErrorMessage : string?  }  ' ====== RELATIONSHIPS ======  Course ||--o{ Module  Module ||--o{ Lesson  Course ||--o{ Enrollment  Lesson ||--o{ Progresses  Course ||--o{ Quiz  Quiz ||--o{ QuizAttempt  QuizAttempt ||--o{ AttemptAnswer  Quiz ||--o{ Question  Question ||--o{ Option  Course ||--o{ Notification  ' Gợi ý liên kết AuditLogs: mỗi thực thể nghiệp vụ đều có thể ghi log  Course ||--o{ AuditLogs : "audit"  Module ||--o{ AuditLogs  Lesson ||--o{ AuditLogs  ' Gợi ý liên kết OutboxMessages: phát sự kiện từ các aggregate quan trọng  Course ||--o{ OutboxMessages : "events"  Quiz ||--o{ OutboxMessages  @enduml |

### Sequence Diagram – Ghi danh khóa học

|  |
| --- |
| @startuml  actor Student  participant "UI" as UI  participant "EnrollmentService\n(Application)" as Service  participant "IEnrollmentRepository\n(Domain Abstraction)" as IRepo  participant "EnrollmentRepository\n(Infrastructure)" as Repo  database "Enrollments DB" as DB  Student -> UI : Chọn "Enroll" khóa học  UI -> Service : Enroll(courseId, userId)  Service -> IRepo : Add Enrollment  IRepo -> Repo : (Implementation)  Repo -> DB : INSERT INTO Enrollments  DB --> Repo : Success  Repo --> IRepo : Enrollment Saved  IRepo --> Service : Enrollment Created  Service --> UI : Response Success  UI --> Student : Hiển thị ghi danh thành công  @enduml |

### Sequence Diagram – Học viên nộp Quiz

|  |
| --- |
| @startuml  actor Learner as U  participant "UI (Blazor)" as UI  participant "Application Layer" as App  participant "Domain Layer" as Domain  participant "Infrastructure Layer" as Infra  U -> UI: Nhấn "Start Quiz"  UI -> App: StartQuizCommand(quizId, userId)  App -> Domain: Validate Business Rules  Domain -> Infra: Create QuizAttempt  Infra --> Domain: QuizAttempt Created  Domain --> App: QuizAttempt Details  App --> UI: QuizAttempt Metadata  U -> UI: Trả lời câu hỏi  U -> UI: Nhấn "Submit Quiz"  UI -> App: SubmitQuizCommand(attemptId, answers)  App -> Domain: Validate Answers  Domain -> Infra: Save QuizAttempt Results  Infra --> Domain: Results Saved  Domain --> App: Calculate Score  App --> UI: Return Quiz Results  @enduml |

### Sequence Diagram – Đánh dấu hoàn thành một bài học

|  |
| --- |
| @startuml  actor Learner as U  participant "UI (Blazor)" as UI  participant "Application Layer" as App  participant "Domain Layer" as Domain  participant "Infrastructure Layer" as Infra  U -> UI: Nhấn "Mark Completed"  UI -> App: MarkLessonCompletedCommand(userId, lessonId)  App -> Domain: Validate Lesson Completion  Domain -> Infra: Update Progress  Infra --> Domain: Progress Updated  Domain --> App: Emit ProgressUpdated Event  App --> UI: Return Updated Progress  @enduml |

### Sequence Diagram – Tạo khóa học mới

|  |
| --- |
| @startuml  actor Admin/Instructor as U  participant "UI (Blazor)" as UI  participant "Application Layer" as App  participant "Domain Layer" as Domain  participant "Infrastructure Layer" as Infra  U -> UI: Nhập thông tin khóa học  U -> UI: Nhấn "Tạo khóa học"  UI -> App: CreateCourseCommand(courseDetails)  App -> Domain: Validate Course Details  Domain -> Infra: Save Course to Database  Infra --> Domain: Course Saved  Domain --> App: Return Course ID  App --> UI: Display Success Message  @enduml |

### Sequence Diagram – Quản lý bài học trong khóa học

|  |
| --- |
| @startuml  actor Instructor as U  participant "UI (Blazor)" as UI  participant "Application Layer" as App  participant "Domain Layer" as Domain  participant "Infrastructure Layer" as Infra  U -> UI: Nhấn "Thêm/Sửa/Xóa Bài học"  UI -> App: ManageLessonCommand(lessonDetails)  App -> Domain: Validate Lesson Details  Domain -> Infra: Update Lesson in Database  Infra --> Domain: Lesson Updated  Domain --> App: Return Updated Lesson  App --> UI: Display Updated Lesson  @enduml |

### Sequence Diagram – Tạo bài kiểm tra

|  |
| --- |
| @startuml  actor Instructor as U  participant "UI (Blazor)" as UI  participant "Application Layer" as App  participant "Domain Layer" as Domain  participant "Infrastructure Layer" as Infra  U -> UI: Nhập thông tin bài kiểm tra  U -> UI: Nhấn "Tạo bài kiểm tra"  UI -> App: CreateQuizCommand(quizDetails)  App -> Domain: Validate Quiz Details  Domain -> Infra: Save Quiz to Database  Infra --> Domain: Quiz Saved  Domain --> App: Return Quiz ID  App --> UI: Display Success Message  @enduml |

### Sequence Diagram – Xem tiến độ học tập

|  |
| --- |
| @startuml  actor Learner/Instructor as U  participant "UI (Blazor)" as UI  participant "Application Layer" as App  participant "Domain Layer" as Domain  participant "Infrastructure Layer" as Infra  U -> UI: Nhấn "Xem Tiến độ"  UI -> App: GetProgressQuery(userId, courseId)  App -> Infra: Fetch Progress from Database  Infra --> App: Return Progress Data  App --> UI: Display Progress  @enduml |

### Sequence Diagram – Gửi thông báo

|  |
| --- |
| @startuml  actor Admin/Instructor as U  participant "UI (Blazor)" as UI  participant "Application Layer" as App  participant "Domain Layer" as Domain  participant "Infrastructure Layer" as Infra  U -> UI: Nhập nội dung thông báo  U -> UI: Nhấn "Gửi Thông báo"  UI -> App: SendNotificationCommand(notificationDetails)  App -> Domain: Validate Notification  Domain -> Infra: Save Notification to Database  Infra --> Domain: Notification Saved  Domain -> Infra: Send Notification via Email  Infra --> Domain: Email Sent  Domain --> App: Return Success  App --> UI: Display Success Message  @enduml |

### Implementation Diagram - UML Interface

|  |
| --- |
| @startuml  !theme plain  hide circle  skinparam packageStyle rectangle  skinparam defaultFontName Arial  skinparam defaultFontSize 12  skinparam nodesep 20  skinparam ranksep 20  left to right direction  package "Application.Abstractions" {  interface ICourseRepository  interface IModuleRepository  interface IEnrollmentRepository  interface IProgressRepository  interface IQuizRepository  interface IUnitOfWork  interface IFileStorage  interface IEmailSender  interface IDateTimeProvider  interface IAuthService  interface IReportService  interface "IQuerySpecification<T>" as IQuerySpecification  }  package "Infrastructure.Repositories" {  class CourseRepository  class ModuleRepository  class EnrollmentRepository  class ProgressRepository  class QuizRepository  }  package "Infrastructure.UnitOfWork" {  class UnitOfWork  }  package "Infrastructure.FileStorage" {  class AzureBlobFileStorage  }  package "Infrastructure.Email" {  class SMTPEmailSender  }  package "Infrastructure.Services" {  class SystemDateTimeProvider  class AuthService  class ReportService  }  package "Infrastructure.Specifications" {  class QuerySpecification  }  ' Quan hệ Implementation  CourseRepository -up-|> ICourseRepository  ModuleRepository -up-|> IModuleRepository  EnrollmentRepository -up-|> IEnrollmentRepository  ProgressRepository -up-|> IProgressRepository  QuizRepository -up-|> IQuizRepository  UnitOfWork -up-|> IUnitOfWork  AzureBlobFileStorage -up-|> IFileStorage  SMTPEmailSender -up-|> IEmailSender  SystemDateTimeProvider -up-|> IDateTimeProvider  AuthService -up-|> IAuthService  ReportService -up-|> IReportService  QuerySpecification -up-|> IQuerySpecification  @enduml |

## Phụ lục C — DTOs, Sample Payloads & OpenA

### DTOs (C# samples)

|  |  |  |  |
| --- | --- | --- | --- |
| Nhóm | Tên DTO | Thuộc tính | Mô tả |
| Envelope | ApiResponse<T> | Success: bool – kết quả chung<br>Data: T? – payload dữ liệu<br>Error: ApiError? – thông tin lỗi<br>TraceId: string – mã truy vết | Mẫu bao ngoài cho mọi response API |
|  | ApiError | Code: string – mã lỗi<br>Message: string – mô tả lỗi<br>Details: object? – chi tiết bổ sung | Thông tin lỗi chuẩn |
| Common | CreatedResponse | Id: Guid | Response khi tạo mới resource |
|  | PagedResult<T> | Items: IEnumerable<T><br>TotalCount: int<br>Page: int<br>Size: int | Kết quả phân trang |
| Courses | CreateCourseRequest | Code: string<br>Title: string<br>Description: string? | Yêu cầu tạo khoá học |
|  | UpdateCourseRequest | Title: string<br>Description: string?<br>Status: string | Yêu cầu cập nhật khoá học |
|  | CourseDto | Id: Guid<br>Code: string<br>Title: string<br>Description: string?<br>Status: string<br>CreatedAt: DateTime | DTO chi tiết khoá học |
|  | CourseListDto | Id: Guid<br>Code: string<br>Title: string<br>Status: string | DTO rút gọn khoá học |
| Modules / Lessons | CreateModuleRequest | Title: string<br>Order: int | Tạo module mới |
|  | ModuleDto | Id: Guid<br>CourseId: Guid<br>Title: string<br>Order: int | DTO module |
|  | CreateLessonRequest | Title: string<br>ContentType: string<br>ContentUrl: string?<br>Order: int<br>DurationSec: int? | Tạo lesson |
|  | LessonDto | Id: Guid<br>ModuleId: Guid<br>Title: string<br>ContentType: string<br>ContentUrl: string?<br>DurationSec: int?<br>Order: int | DTO lesson |
| Enrollment | EnrollRequest | UserId: Guid? | Ghi danh (null = current user) |
|  | EnrollmentDto | Id: Guid<br>CourseId: Guid<br>UserId: Guid<br>StartAt: DateTime<br>EndAt: DateTime?<br>Status: string | DTO ghi danh |
| Progress | ProgressDto | Id: Guid<br>UserId: Guid<br>LessonId: Guid<br>Percent: byte<br>CompletedAt: DateTime? | DTO tiến độ |
| Quiz | CreateQuizRequest | Title: string<br>DurationSec: int?<br>AttemptsAllowed: int?<br>ShuffleAnswers: bool | Tạo quiz |
|  | QuizDto | Id: Guid<br>CourseId: Guid<br>Title: string<br>DurationSec: int?<br>AttemptsAllowed: int<br>ShuffleAnswers: bool | DTO quiz |
|  | StartQuizResponse | AttemptId: Guid<br>Questions: IEnumerable<QuestionDto> | Response bắt đầu quiz |
|  | QuestionDto | Id: Guid<br>Text: string<br>Order: int<br>Options: IEnumerable<OptionDto> | Câu hỏi quiz |
|  | OptionDto | Id: Guid<br>Text: string<br>IsCorrect: bool | Lựa chọn; IsCorrect có thể ẩn ở response |
|  | SubmitQuizRequest | AttemptId: Guid<br>Answers: IEnumerable<AnswerDto> | Nộp quiz |
|  | AnswerDto | QuestionId: Guid<br>OptionId: Guid | Câu trả lời |
|  | SubmitQuizResponse | AttemptId: Guid<br>Score: decimal<br>CorrectCount: int<br>Total: int | Kết quả nộp quiz |
|  | QuizAttemptDto | Id: Guid<br>QuizId: Guid<br>UserId: Guid<br>StartedAt: DateTime<br>SubmittedAt: DateTime?<br>Score: decimal? | DTO lượt làm quiz |
| Notification | CreateNotificationRequest | CourseId: Guid?<br>ToUserId: Guid?<br>Title: string<br>Body: string? | Gửi thông báo |
|  | NotificationDto | Id: Guid<br>CourseId: Guid?<br>ToUserId: Guid?<br>Title: string<br>Body: string?<br>CreatedAt: DateTime<br>SentBy: Guid | DTO thông báo |
| File | FileUploadResponse | Id: Guid<br>Url: string<br>FileName: string<br>Size: long<br>MimeType: string | Kết quả upload file |

### Sample JSON payloads

1. Create course (request)

{

"code": "CS101",

"title": "Intro to Programming",

"description": "Basic programming course"

}

Create course (successful response)

{

"success": true,

"data": { "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6" },

"error": null,

"traceId": "abcdef-12345"

}

1. Enroll (request)

{

"userId": "9f1c3f45-1111-2222-3333-444455556666"

}

Submit quiz (request)

{

"attemptId": "11111111-2222-3333-4444-555555555555",

"answers": [

{ "questionId": "aa111111-2222-3333-4444-666666666666", "optionId": "bb111111-2222-3333-4444-777777777777" },

{ "questionId": "aa222222-2222-3333-4444-666666666666", "optionId": "bb222222-2222-3333-4444-777777777777" }

]

}

1. Error response (validation)

{

"success": false,

"data": null,

"error": {

"code": "ERR\_VALIDATION",

"message": "Dữ liệu không hợp lệ",

"details": {

"Title": ["Title is required", "Length must be <= 200"],

"Code": ["Code already exists"]

}

},

"traceId": "trace-1234"

}

### OpenAPI (rút gọn) — YAML

Ghi chú: đây là OpenAPI rút gọn chỉ chứa các path cốt lõi để làm phụ lục; bạn có thể lưu thành openapi.yaml và mở bằng Swagger / Redoc. Mình giữ ngắn để dễ chèn, bạn có thể mở rộng theo nhu cầu.

openapi: 3.0.3

info:

title: LMS Mini API (rút gọn)

version: "1.0.0"

servers:

- url: /api/v1

components:

schemas:

CreateCourseRequest:

type: object

properties:

code: { type: string }

title: { type: string }

description: { type: string, nullable: true }

required: [code, title]

CourseDto:

type: object

properties:

id: { type: string, format: uuid }

code: { type: string }

title: { type: string }

description: { type: string, nullable: true }

status: { type: string }

createdAt: { type: string, format: date-time }

EnrollRequest:

type: object

properties:

userId: { type: string, format: uuid, nullable: true }

paths:

/courses:

get:

summary: Get courses (paged)

parameters:

- name: page; in: query; schema: {type: integer, default:1}

- name: size; in: query; schema: {type: integer, default:20}

responses:

'200':

description: paged list

content:

application/json:

schema:

$ref: '#/components/schemas/CourseDto'

post:

summary: Create course

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/CreateCourseRequest'

responses:

'201':

description: created

/courses/{id}/enroll:

post:

summary: Enroll to course

parameters:

- name: id; in: path; required: true; schema: {type: string, format: uuid}

requestBody:

content:

application/json:

schema:

$ref: '#/components/schemas/EnrollRequest'

responses:

'200': { description: OK }

/quizzes/{id}/start:

post:

summary: Start quiz

parameters:

- name: id; in: path; required: true; schema: {type: string, format: uuid}

responses:

'200': { description: StartQuizResponse (see appendix) }

/quizzes/{id}/submit:

post:

summary: Submit quiz

parameters:

- name: id; in: path; required: true; schema: {type: string, format: uuid}

requestBody:

content:

application/json:

schema:

type: object

properties:

attemptId: { type: string, format: uuid }

answers:

type: array

items:

type: object

properties:

questionId: { type: string, format: uuid }

optionId: { type: string, format: uuid }

responses:

'200': { description: SubmitQuizResponse }

* Tất cả DTO được chuẩn hóa thành OpenAPI schema (components.schemas).
* Mapping 1–1 với domain entity (User, Course, Notification, FileAsset, Report).
* Ví dụ:

Components:

schemas:

UserDto:

type: object

properties:

id: { type: string }

username: { type: string }

email: { type: string, format: email }

## ****Phụ lục D – Thư viện & Framework sử dụng****

Dự án sử dụng các thư viện bên ngoài nhằm hỗ trợ hiện thực kiến trúc phần mềm, đảm bảo hiệu năng, khả năng mở rộng và tính dễ bảo trì. Các thư viện này được tích hợp trực tiếp trong file Program.cs và các thành phần liên quan như Controller, Service, Handler.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Thư viện | Mục đích sử dụng | Namespace | Vị trí sử dụng | Liên hệ với Use Case / SDD | Ghi chú kỹ thuật |
| **Serilog** | Ghi log hệ thống, phục vụ giám sát và debug | Serilog | Program.cs, các handler | Theo dõi hoạt động của các Use Case như EnrollCourseCommand, CreateUserCommand | Có thể mở rộng sang file, Seq hoặc ElasticSearch |
| **MediatR** | Hiện thực mô hình CQRS, tách biệt logic xử lý | MediatR | Program.cs, các handler | Dùng để xử lý các Use Case như CreateCourseCommand, GetCourseQuery | Tăng khả năng kiểm thử và mở rộng |
| **AutoMapper** | Ánh xạ giữa DTO và Entity | AutoMapper | Program.cs, các service | Dùng trong quá trình chuyển đổi dữ liệu giữa tầng API và tầng Domain | Sử dụng profile để ánh xạ tự động, giảm mã lặp |
| **FluentValidation** | Kiểm tra dữ liệu đầu vào | FluentValidation, FluentValidation.AspNetCore | Program.cs, các lớp validator | Kiểm tra dữ liệu cho các request như CreateCourseRequest, RegisterUserRequest | Tích hợp với ASP.NET Core để kiểm tra tự động |
| **Swagger (Swashbuckle)** | Tài liệu hóa API và giao diện thử nghiệm | Microsoft.OpenApi.Models | Program.cs, Startup.cs | Hỗ trợ kiểm thử các Use Case qua giao diện Swagger, tích hợp bảo mật JWT | Cần cấu hình cẩn thận để tránh lộ thông tin trong môi trường production |
| **ASP.NET Core** | Framework chính của ứng dụng web | Microsoft.AspNetCore.\* | Toàn bộ dự án | Hiện thực các controller, middleware và DI container | Tích hợp tốt với các thư viện khác, dễ mở rộng |

### Ghi chú bổ sung

* **Phiên bản thư viện**: Các thư viện được quản lý qua NuGet, phiên bản cụ thể được ghi trong file \*.csproj.
* **Kiểm thử thư viện**: Các thư viện được kiểm thử thông qua các use case cụ thể và tích hợp trong môi trường staging.
* **Rủi ro & Giải pháp**:
  + Swagger cần được cấu hình để ẩn endpoint nhạy cảm khi deploy production.
  + Serilog cần giới hạn mức log để tránh ghi quá nhiều trong môi trường thật.