LMS 👍

## **Project LMS: System Architecture & Requirements Specification**

**Version: 1.1**

### **1. نظرة عامة والمبادئ المعمارية (Overview & Architectural Principles)**

هذا المستند يقدم التصميم الكامل لنظام إدارة التعلم (LMS). الهدف هو بناء نظام قوي، قابل للصيانة، وقابل للتطوير. لتحقيق ذلك، يرتكز النظام على المبادئ الأساسية التالية:

* **Clean Architecture**: لضمان الفصل التام بين الاهتمامات (Separation of Concerns). طبقة الـ Domain (قلب العمل) لا تعتمد على أي تفاصيل تقنية مثل قواعد البيانات أو واجهات برمجة التطبيقات.
* **CQRS with MediatR**: لفصل عمليات الكتابة (Commands) عن عمليات القراءة (Queries). هذا يزيد من وضوح الكود، ويحسن الأداء، ويسهل صيانة كل عملية على حدة.
* **Vertical Slicing**: لتنظيم الكود حسب الميزة (Feature) بدلاً من الطبقة. كل ميزة تحتوي على كل ما تحتاجه لتعمل (الـCommand، الـHandler، الـValidator، إلخ)، مما يزيد من تماسك الكود (Cohesion).

### **2. مواصفات طبقة النطاق (Domain Layer Specification)**

هذه الطبقة تحتوي على كل كيانات العمل وقواعده الأساسية.

* **Course Entity 📚**: يمثل الدورة التعليمية.
  + **Properties**: Id, Title, Description, Price, AcademicYear, State, CoverImageUrl, TeacherId.
  + **Rules**: لا يمكن نشره بدون دروس، السعر لا يمكن أن يكون سالبًا.
* **Lesson Entity 📖**: يمثل الدرس الواحد داخل الدورة.
  + **Properties**: Id, Title, Objectives, VideoUrl, AttachmentUrl, Order, CourseId.
  + **Rules**: يجب أن يحتوي على فيديو أو ملف مرفق على الأقل.
* **Exam Entity 📝**: يمثل الامتحان.
  + **Properties**: Id, Title, DurationInMinutes, PassingScore, State, CourseId.
  + **Rules**: لا يمكن نشره بدون أسئلة، المدة يجب أن تكون أكبر من صفر.
* **Question & Option Entities ❓**: يمثل السؤال وخياراته.
  + **Question Properties**: Id, Content, Score, QuestionType, ExamId.
  + **Option Properties**: Id, Text, IsCorrect.
  + **Rules**: يجب أن يحتوي على خيارين على الأقل، وإجابة صحيحة واحدة فقط لنوع SingleChoice.
* **Student Entity 👤**: يمثل حساب الطالب.
  + **Properties**: Id, FullName, PhoneNumber (فريد), ParentPhoneNumber, AcademicYear, PasswordHash.
  + **Rules**: PhoneNumber يجب أن يكون فريدًا.
* **Enrollment Entity 🎓**: يمثل اشتراك الطالب في دورة.
  + **Properties**: Id, StudentId, CourseId, EnrollmentDate, Progress.
  + **Rules**: لا يمكن للطالب الاشتراك في نفس الدورة مرتين.
  + **Relationships**: يحتوي على قائمة بنتائج امتحانات الطالب في هذه الدورة.

### **3. مواصفات طبقة التطبيق (Application Layer Specification)**

تحتوي هذه الطبقة على منطق حالات الاستخدام (Use Cases)، ممثلة في Commands و Queries.

#### **حالات استخدام المدير/المدرس (Admin/Teacher Use Cases)**

* **Commands (عمليات الكتابة):**
  + CreateCourseCommand: لإنشاء دورة جديدة بالبيانات الأساسية.
  + UpdateCourseInfoCommand: لتحديث تفاصيل دورة موجودة.
  + AddLessonToCourseCommand: لإضافة درس جديد إلى دورة.
  + PublishCourseCommand: لتغيير حالة الدورة من Draft إلى Published.
  + CreateExamCommand: لإنشاء امتحان جديد وربطه بدورة.
  + AddQuestionToExamCommand: لإضافة سؤال وخياراته إلى امتحان.
  + PublishExamCommand: لنشر امتحان وجعله متاحًا للطلاب.
* **Queries (عمليات القراءة):**
  + GetCourseByIdQuery: لجلب تفاصيل دورة واحدة بكل دروسها وامتحاناتها.
  + GetAllCoursesQuery: لجلب كل الدورات التي أنشأها المدرس.
  + GetEnrollmentsForCourseQuery: لعرض قائمة الطلاب المشتركين في دورة معينة.
  + GetStudentProgressQuery: لعرض تفاصيل تقدم طالب معين في دورة معينة ودرجاته.

#### **حالات استخدام الطالب (Student Use Cases)**

* **Commands (عمليات الكتابة):**
  + RegisterStudentCommand: لتسجيل حساب طالب جديد.
  + EnrollInCourseCommand: لاشتراك الطالب في دورة.
  + CompleteLessonCommand: ليقوم الطالب بتحديد درس على أنه مكتمل (لتحديث الـ Progress).
  + SubmitExamCommand: لتقديم إجابات الطالب في امتحان وحساب الدرجة.
* **Queries (عمليات القراءة):**
  + GetMyCoursesQuery: لجلب قائمة بالدورات التي اشترك بها الطالب.
  + GetCourseContentQuery: لعرض محتوى دورة مشترَك بها (الدروس والفيديوهات والامتحانات).
  + GetMyGradesQuery: لعرض سجل درجات الطالب في جميع الامتحانات.

### **4. الخدمات الأساسية (Core Services & Cross-Cutting Concerns)**

هذه هي الخدمات التقنية التي ستدعم كل عمليات النظام لضمان الجودة والأمان.

* **Validation (التحقق من صحة البيانات):**
  + **الاستراتيجية**: سيتم استخدام مكتبة FluentValidation لتعريف قواعد التحقق لكل Command.
  + **التنفيذ**: سيتم دمجها في MediatR Pipeline Behavior لتشغيل الـ validator تلقائيًا قبل وصول أي command إلى الـ handler الخاص به. أي خطأ في التحقق سيؤدي إلى إرجاع استجابة HTTP 400 Bad Request بشكل تلقائي مع تفاصيل الأخطاء.
* **Error Handling (معالجة الأخطاء):**
  + **الاستراتيجية**: بناء Global Exception Handling Middleware.
  + **التنفيذ**: هذا الـ middleware سيقوم بالتقاط أي أخطاء (exceptions) تحدث في النظام ومنع تسرب التفاصيل التقنية للمستخدم. سيقوم بتحويل الأخطاء إلى استجابات JSON قياسية بصيغة ProblemDetails (مثل 404 Not Found أو 500 Internal Server Error).
* **Logging (تسجيل الأحداث):**
  + **الاستراتيجية**: استخدام Structured Logging عبر مكتبة Serilog.
  + **التنفيذ**: سيتم إعداد MediatR Pipeline Behavior آخر ليقوم بتسجيل كل طلب (request) يدخل للنظام، مع تسجيل مدة تنفيذه ونجاحه من فشله. سيتم إثراء كل سجل بـ CorrelationId لتسهيل تتبع مسار الطلب عبر النظام.