## ****Giai đoạn 1: Xây dựng nền tảng cốt lõi****

**Mục tiêu**: Tạo các service cơ bản (User, Course, Media) và API Gateway để hệ thống hoạt động với chức năng tối thiểu.

### ****1. User Service (Spring Boot)****

#### ****Use Case****

Dựa trên chức năng ưu tiên (đăng ký, đăng nhập, lấy thông tin người dùng, xác thực JWT), các **Use Case** cần thiết là:

* **Đăng ký tài khoản** (Actor: Guest)
  + Mô tả: Người dùng mới (Guest) tạo tài khoản để trở thành Student hoặc Instructor.
  + Thao tác: Nhập thông tin (tên, email, mật khẩu, vai trò), hệ thống lưu vào database.
* **Đăng nhập** (Actor: Student, Instructor, Admin)
  + Mô tả: Người dùng xác thực danh tính để truy cập hệ thống.
  + Thao tác: Nhập email/mật khẩu, nhận JWT token.
* **Quản lý thông tin cá nhân** (Actor: Student, Instructor, Admin)
  + Mô tả: Người dùng xem và cập nhật thông tin cá nhân (tên, email, mật khẩu).
  + Thao tác: Xem thông tin, cập nhật thông tin.
* **Quản lý người dùng** (Actor: Admin)
  + Mô tả: Admin thêm, xem, sửa, xóa người dùng và phân quyền (gán vai trò Student, Instructor, Admin).
  + Thao tác: Thêm, xem danh sách/chi tiết, cập nhật, xóa, gán vai trò.

#### ****Sơ đồ cần vẽ****

* **Use Case Diagram**:
  + Mô tả các Use Case trên với Actor: Guest, Student, Instructor, Admin.
  + Mối quan hệ **<<include>></include>** để chỉ ra các thao tác như "Tạo người dùng", "Xác thực người dùng", "Cập nhật thông tin cá nhân".
* **Class Diagram**:
  + Lớp chính: User (thuộc tính: id, name, email, password, role).
  + Lớp phụ: Token (thuộc tính: user\_id, token, expires\_at).
  + Mối quan hệ: User 1-n Token (một người dùng có nhiều token JWT).
* **Sequence Diagram**:
  + Luồng **Đăng ký tài khoản**: Guest -> Frontend -> API Gateway -> User Service -> Database.
  + Luồng **Đăng nhập**: Actor -> Frontend -> API Gateway -> User Service -> Database -> JWT.
* **ERD (Entity-Relationship Diagram)**:
  + Bảng: Users (id, name, email, password, role).
  + Bảng: Tokens (id, user\_id, token, expires\_at, created\_at).
  + Mối quan hệ: Users 1-n Tokens (foreign key: user\_id).
* **Architecture Diagram**:
  + Mô tả User Service tích hợp với API Gateway và PostgreSQL.

#### ****API tối ưu****

Để tránh dư thừa, chỉ thiết kế các API cần thiết dựa trên Use Case:

* POST /api/users/register (Đăng ký tài khoản)
  + Input: { name, email, password, role }
  + Output: { userId, message }
* POST /api/users/login (Đăng nhập)
  + Input: { email, password }
  + Output: { token, userId, role }
* GET /api/users/:id (Xem thông tin cá nhân)
  + Yêu cầu JWT, chỉ cho phép xem thông tin của chính người dùng hoặc Admin.
  + Output: { id, name, email, role }
* PUT /api/users/:id (Cập nhật thông tin cá nhân)
  + Yêu cầu JWT, chỉ cho phép chính người dùng hoặc Admin.
  + Input: { name, email, password }
  + Output: { message }
* POST /api/users (Thêm người dùng, chỉ Admin)
  + Yêu cầu JWT (vai trò Admin).
  + Input: { name, email, password, role }
  + Output: { userId, message }
* GET /api/users (Xem danh sách người dùng, chỉ Admin)
  + Yêu cầu JWT (vai trò Admin).
  + Output: [{ id, name, email, role }]
* DELETE /api/users/:id (Xóa người dùng, chỉ Admin)
  + Yêu cầu JWT (vai trò Admin).
  + Output: { message }
* PUT /api/users/:id/role (Phân quyền, chỉ Admin)
  + Yêu cầu JWT (vai trò Admin).
  + Input: { role }
  + Output: { message }

**Tổng số API**: 8 API (đủ để hỗ trợ tất cả Use Case mà không dư thừa).

* **Tránh dư thừa**: Không cần API riêng cho mỗi thao tác nhỏ (ví dụ: không cần API riêng để cập nhật từng trường như email, password). Thay vào đó, PUT /api/users/:id cho phép cập nhật toàn bộ thông tin cần thiết.

#### ****Thời gian ước tính****

* Thiết kế và phát triển: **1-2 tuần** (như bạn đề xuất).
* Công cụ: Spring Boot, PostgreSQL, Spring Security (JWT).

### ****2. Course Service (FastAPI)****

#### ****Use Case****

Dựa trên chức năng ưu tiên (tạo, cập nhật, xóa, tìm kiếm khóa học; quản lý danh mục, chương, bài học):

* **Quản lý khóa học** (Actor: Instructor)
  + Mô tả: Instructor tạo, xem, cập nhật, xóa khóa học.
  + Thao tác: Tạo khóa học, xem danh sách/chi tiết, cập nhật, xóa.
* **Quản lý danh mục, chương, bài học** (Actor: Instructor)
  + Mô tả: Instructor tạo, quản lý cấu trúc phân cấp (Category > Course > Section > Lesson).
  + Thao tác: Tạo, xem, cập nhật, xóa danh mục, chương, bài học.
* **Xem khóa học** (Actor: Student)
  + Mô tả: Student xem danh sách khóa học, chi tiết khóa học, danh mục, chương, bài học. -45 **Kiểm duyệt khóa học** (Actor: Admin)
  + Mô tả: Admin xem, cập nhật trạng thái (duyệt/từ chối), xóa khóa học.
  + Thao tác: Xem danh sách/chi tiết, cập nhật trạng thái, xóa.

#### ****Sơ đồ cần vẽ****

* **Use Case Diagram**:
  + Mô tả các Use Case trên với Actor: Student, Instructor, Admin.
  + Mối quan hệ **<<include>></include>** cho các thao tác như "Tạo khóa học", "Tạo danh mục", "Xem chi tiết khóa học".
* **Class Diagram**:
  + Lớp chính: Category (id, name), Course (id, title, description, instructor\_id, category\_id, status), Section (id, course\_id, title), Lesson (id, section\_id, title, content).
  + Mối quan hệ: Category 1-n Course, Course 1-n Section, Section 1-n Lesson.
* **Sequence Diagram**:
  + Luồng **Tạo khóa học**: Instructor -> Frontend -> API Gateway -> Course Service -> Database.
  + Luồng **Xem khóa học**: Student -> Frontend -> API Gateway -> Course Service -> Database.
* **ERD**:
  + Bảng: Categories (id, name), Courses (id, title, description, instructor\_id, category\_id, status), Sections (id, course\_id, title), Lessons (id, section\_id, title, content).
  + Mối quan hệ: Categories 1-n Courses, Courses 1-n Sections, Sections 1-n Lessons.
* **Architecture Diagram**:
  + Mô tả Course Service tích hợp với API Gateway, User Service, và PostgreSQL.

#### ****API tối ưu****

* **Category**:
  + POST /api/categories (Tạo danh mục, chỉ Instructor/Admin)
  + GET /api/categories (Xem danh sách danh mục)
  + GET /api/categories/:id (Xem chi tiết danh mục)
  + PUT /api/categories/:id (Cập nhật danh mục, chỉ Instructor/Admin)
  + DELETE /api/categories/:id (Xóa danh mục, chỉ Instructor/Admin)
* **Course**:
  + POST /api/courses (Tạo khóa học, chỉ Instructor)
  + GET /api/courses (Xem danh sách khóa học)
  + GET /api/courses/:id (Xem chi tiết khóa học)
  + PUT /api/courses/:id (Cập nhật khóa học, chỉ Instructor)
  + DELETE /api/courses/:id (Xóa khóa học, chỉ Instructor/Admin)
  + PUT /api/courses/:id/status (Cập nhật trạng thái khóa học, chỉ Admin)
* **Section**:
  + POST /api/sections (Tạo chương, chỉ Instructor)
  + GET /api/sections?course\_id={id} (Xem danh sách chương của khóa học)
  + PUT /api/sections/:id (Cập nhật chương, chỉ Instructor)
  + DELETE /api/sections/:id (Xóa chương, chỉ Instructor)
* **Lesson**:
  + POST /api/lessons (Tạo bài học, chỉ Instructor)
  + GET /api/lessons?section\_id={id} (Xem danh sách bài học của chương)
  + PUT /api/lessons/:id (Cập nhật bài học, chỉ Instructor)
  + DELETE /api/lessons/:id (Xóa bài học, chỉ Instructor)

**Tổng số API**: 19 API (đủ để hỗ trợ quản lý cấu trúc phân cấp Category > Course > Section > Lesson).

* **Tránh dư thừa**: Không cần API riêng cho từng trường của khóa học (ví dụ: chỉ cần PUT /api/courses/:id để cập nhật toàn bộ thông tin). Sử dụng query parameter (như course\_id, section\_id) để lọc dữ liệu thay vì tạo API riêng.

#### ****Thời gian ước tính****

* Thiết kế và phát triển: **2-3 tuần** (như bạn đề xuất).
* Công cụ: FastAPI, PostgreSQL.

### ****3. Media Service (Spring Boot)****

#### ****Use Case****

Dựa trên chức năng ưu tiên (upload/download/streaming video/tài liệu, lưu metadata):

* **Quản lý media** (Actor: Instructor)
  + Mô tả: Instructor tải lên, xem, cập nhật, xóa video hoặc tài liệu (PDF, Word, PowerPoint).
  + Thao tác: Upload, download/streaming, cập nhật metadata, xóa.
* **Xem media** (Actor: Student)
  + Mô tả: Student xem hoặc tải tài liệu/video của bài học.
* **Kiểm duyệt media** (Actor: Admin)
  + Mô tả: Admin xem, xóa media nếu không phù hợp.

#### ****Sơ đồ cần vẽ****

* **Use Case Diagram**:
  + Mô tả các Use Case trên với Actor: Student, Instructor, Admin.
  + Mối quan hệ **<<include>></include>** cho các thao tác như "Upload media", "Xem media".
* **Class Diagram**:
  + Lớp chính: Media (id, lesson\_id, url, type, metadata).
  + Mối quan hệ: Lesson 1-n Media.
* **Sequence Diagram**:
  + Luồng **Upload media**: Instructor -> Frontend -> API Gateway -> Media Service -> MinIO -> Database.
  + Luồng **Streaming media**: Student -> Frontend -> API Gateway -> Media Service -> MinIO.
* **ERD**:
  + Bảng: Media (id, lesson\_id, url, type, metadata, uploaded\_at).
  + Mối quan hệ: Lessons 1-n Media (foreign key: lesson\_id).
* **Architecture Diagram**:
  + Mô tả Media Service tích hợp với MinIO, API Gateway, Course Service, và PostgreSQL.

#### ****API tối ưu****

* POST /api/media (Upload media, chỉ Instructor)
  + Input: Multipart file (video/PDF/Word/PPT), metadata { lesson\_id, type }
  + Output: { mediaId, url, message }
* GET /api/media/:id (Xem/tải/streaming media)
  + Yêu cầu JWT, chỉ Student (đã đăng ký khóa học) hoặc Instructor/Admin.
  + Output: File stream hoặc { url }
* PUT /api/media/:id (Cập nhật metadata, chỉ Instructor)
  + Input: { lesson\_id, type, metadata }
  + Output: { message }
* DELETE /api/media/:id (Xóa media, chỉ Instructor/Admin)
  + Output: { message }
* GET /api/media?lesson\_id={id} (Lấy danh sách media của bài học)
  + Output: [{ id, url, type, metadata }]

**Tổng số API**: 5 API (đủ để hỗ trợ upload, download, streaming, quản lý metadata).

* **Tránh dư thừa**: Không cần API riêng cho từng loại file (video, PDF, v.v.). Sử dụng trường type trong metadata để phân biệt.

#### ****Thời gian ước tính****

* Thiết kế và phát triển: **1-2 tuần** (như bạn đề xuất).
* Công cụ: Spring Boot, PostgreSQL, MinIO.

### ****4. API Gateway (Spring Cloud Gateway)****

#### ****Use Case****

* **Định tuyến yêu cầu** (Actor: Frontend, External System)
  + Mô tả: Định tuyến yêu cầu từ client đến các service (User, Course, Media).
* **Xác thực yêu cầu** (Actor: Frontend, External System)
  + Mô tả: Kiểm tra JWT token trước khi chuyển tiếp yêu cầu.

#### ****Sơ đồ cần vẽ****

* **Use Case Diagram**:
  + Actor: Frontend, External System.
  + Use Case: Định tuyến yêu cầu, Xác thực yêu cầu.
* **Sequence Diagram**:
  + Luồng **Xác thực và định tuyến**: Frontend -> API Gateway -> User Service (kiểm tra JWT) -> Target Service (User/Course/Media).
* **Architecture Diagram**:
  + Mô tả API Gateway tích hợp với User Service, Course Service, Media Service, và các client (Frontend, External System).

#### ****API tối ưu****

API Gateway không định nghĩa API riêng mà hoạt động như một proxy:

* **Định tuyến**:
  + /api/users/\* -> User Service.
  + /api/courses/\*, /api/categories/\*, /api/sections/\*, /api/lessons/\* -> Course Service.
  + /api/media/\* -> Media Service.
* **Xác thực**:
  + Kiểm tra JWT token trong header Authorization cho tất cả yêu cầu (trừ /api/users/register và /api/users/login).
* **Tích hợp**:
  + Sử dụng Spring Cloud Gateway để định tuyến và bộ lọc (filter) để xác thực JWT.

**Tổng số API**: Không thêm API mới, chỉ cấu hình định tuyến và xác thực.

#### ****Thời gian ước tính****

* Thiết kế và phát triển: **1 tuần** (như bạn đề xuất).
* Công cụ: Spring Cloud Gateway, tích hợp với Spring Security.

## ****Tổng số API trong Giai đoạn 1****

* **User Service**: 8 API.
* **Course Service**: 19 API.
* **Media Service**: 5 API.
* **API Gateway**: 0 API (chỉ định tuyến).
* **Tổng cộng**: 32 API.

**Tránh dư thừa**:

* Sử dụng **query parameter** (như course\_id, lesson\_id) để lọc dữ liệu thay vì tạo API riêng.
* Gộp các thao tác cập nhật vào một API (ví dụ: PUT /api/users/:id thay vì API riêng cho từng trường).
* Tái sử dụng API giữa các Actor (ví dụ: GET /api/courses cho cả Student, Instructor, Admin).

## ****Các giai đoạn tiếp theo (Tóm tắt)****

Để tránh lặp lại quá chi tiết, tôi sẽ cung cấp hướng dẫn tổng quát cho các giai đoạn còn lại, tập trung vào cách xác định Use Case, sơ đồ, và API tối ưu.

### ****Giai đoạn 2: Theo dõi tiến độ và giao diện người dùng****

#### ****Progress Service (FastAPI)****

* **Use Case**:
  + Theo dõi tiến độ học tập (Actor: Student): Lưu phần trăm hoàn thành khóa học/bài học, vị trí video.
  + Quản lý tiến độ (Actor: Admin): Xem tiến độ của người dùng.
* **Sơ đồ**:
  + Use Case Diagram: Các Use Case "Theo dõi tiến độ học tập", "Quản lý tiến độ".
  + Class Diagram: UserCourseProgress (user\_id, course\_id, completion\_percentage), LessonProgress (user\_id, lesson\_id, video\_position).
  + Sequence Diagram: Luồng lưu/cập nhật tiến độ.
  + ERD: Bảng UserCourseProgress, LessonProgress.
* **API tối ưu**:
  + POST /api/progress (Lưu tiến độ khóa học/bài học)
  + GET /api/progress?user\_id={id}&course\_id={id} (Xem tiến độ khóa học)
  + PUT /api/progress/lesson (Cập nhật vị trí video)
  + **Tổng số API**: ~3-4 API.

#### ****Frontend Service (React TypeScript + Vite)****

* **Use Case**:
  + Xem danh sách/chi tiết khóa học (Actor: Student).
  + Phát video, hiển thị tiến độ (Actor: Student).
  + Đăng ký/đăng nhập (Actor: Guest, Student, Instructor).
* **Sơ đồ**:
  + Wireframe/Mockup: Thiết kế giao diện cho danh sách khóa học, trang chi tiết, phát video.
  + Sequence Diagram: Luồng tương tác Frontend -> API Gateway -> Backend.
* **API sử dụng**:
  + Tái sử dụng API từ User Service (/api/users/register, /api/users/login).
  + Tái sử dụng API từ Course Service (/api/courses, /api/courses/:id).
  + Tái sử dụng API từ Media Service (/api/media/:id).
  + Tái sử dụng API từ Progress Service (/api/progress/\*).
* **Thời gian ước tính**: 3-4 tuần.

### ****Giai đoạn 3: Tích hợp thanh toán và tìm kiếm****

#### ****Payment Service (Spring Boot)****

* **Use Case**:
  + Thanh toán khóa học (Actor: Student): Mua khóa học, kiểm tra trạng thái.
  + Quản lý thanh toán (Actor: Admin): Xem, xử lý hoàn tiền.
* **Sơ đồ**:
  + Use Case Diagram: "Thanh toán khóa học", "Quản lý thanh toán".
  + Class Diagram: Payment (id, user\_id, course\_id, amount, status).
  + Sequence Diagram: Luồng thanh toán qua Stripe/PayPal.
  + ERD: Bảng Payments.
* **API tối ưu**:
  + POST /api/payments (Tạo giao dịch)
  + GET /api/payments/:id (Kiểm tra trạng thái)
  + POST /api/payments/refund (Hoàn tiền, chỉ Admin)
  + **Tổng số API**: ~3 API.

#### ****Search Service (Spring Boot)****

* **Use Case**:
  + Tìm kiếm khóa học (Actor: Student, Instructor): Tìm theo từ khóa.
  + Đồng bộ dữ liệu tìm kiếm (Actor: System): Cập nhật index từ Course Service.
* **Sơ đồ**:
  + Use Case Diagram: "Tìm kiếm khóa học", "Đồng bộ dữ liệu tìm kiếm".
  + Sequence Diagram: Luồng tìm kiếm và đồng bộ.
  + Architecture Diagram: Search Service tích hợp với Elasticsearch.
* **API tối ưu**:
  + GET /api/search?q={keyword} (Tìm kiếm khóa học)
  + POST /api/search/sync (Đồng bộ dữ liệu, chỉ System)
  + **Tổng số API**: ~2 API.

### ****Giai đoạn 4: Tích hợp AI và đánh giá****

#### ****AI Service (FastAPI)****

* **Use Case**:
  + Gợi ý khóa học/bài học (Actor: Student): Dựa trên chủ đề, lịch sử học.
  + Tạo câu hỏi/tóm tắt (Actor: Instructor): Tạo câu hỏi từ nội dung bài học.
* **Sơ đồ**:
  + Use Case Diagram: "Gợi ý khóa học", "Tạo câu hỏi/tóm tắt".
  + Sequence Diagram: Luồng gợi ý (RAG), trích xuất nội dung (Whisper, PyPDF2).
  + Class Diagram: LessonContent (id, lesson\_id, content), UserQuery (id, user\_id, query).
* **API tối ưu**:
  + GET /api/recommendations?user\_id={id} (Gợi ý khóa học)
  + POST /api/questions (Tạo câu hỏi từ nội dung)
  + POST /api/summaries (Tạo tóm tắt)
  + **Tổng số API**: ~3 API.

#### ****Review Service (FastAPI)****

* **Use Case**:
  + Gửi đánh giá (Actor: Student): Đánh giá, bình luận khóa học.
  + Quản lý đánh giá (Actor: Admin): Xem, xóa đánh giá.
* **Sơ đồ**:
  + Use Case Diagram: "Gửi đánh giá", "Quản lý đánh giá".
  + Class Diagram: Review (id, user\_id, course\_id, rating, comment).
  + ERD: Bảng Reviews.
* **API tối ưu**:
  + POST /api/reviews (Gửi đánh giá)
  + GET /api/reviews?course\_id={id} (Xem đánh giá)
  + DELETE /api/reviews/:id (Xóa đánh giá, chỉ Admin)
  + **Tổng số API**: ~3 API.

### ****Giai đoạn 5: Tính năng nâng cao****

#### ****Quiz Service (FastAPI)****

* **Use Case**:
  + Làm bài kiểm tra (Actor: Student): Làm quiz, xem kết quả.
  + Quản lý quiz (Actor: Instructor): Tạo, sửa, xóa quiz.
* **Sơ đồ**:
  + Use Case Diagram: "Làm bài kiểm tra", "Quản lý quiz".
  + Class Diagram: Quiz (id, course\_id, questions), UserQuizResult (user\_id, quiz\_id, score).
  + ERD: Bảng Quizzes, UserQuizResults.
* **API tối ưu**:
  + POST /api/quizzes (Tạo quiz)
  + GET /api/quizzes?course\_id={id} (Xem danh sách quiz)
  + POST /api/quizzes/:id/submit (Nộp bài kiểm tra)
  + GET /api/quizzes/:id/result (Xem kết quả)
  + **Tổng số API**: ~4 API.

#### ****Gamification Service (Spring Boot)****

* **Use Case**:
  + Nhận huy hiệu/điểm (Actor: Student): Khi hoàn thành bài học/quiz.
  + Quản lý gamification (Actor: Admin): Cập nhật quy tắc huy hiệu.
* **Sơ đồ**:
  + Use Case Diagram: "Nhận huy hiệu/điểm", "Quản lý gamification".
  + Class Diagram: UserAchievement (user\_id, achievement\_id, earned\_at), Leaderboard (user\_id, score).
  + ERD: Bảng UserAchievements, Leaderboards.
* **API tối ưu**:
  + POST /api/achievements (Gán huy hiệu)
  + GET /api/achievements?user\_id={id} (Xem huy hiệu)
  + GET /api/leaderboard (Xem bảng xếp hạng)
  + **Tổng số API**: ~3 API.

#### ****Notification Service (FastAPI)****

* **Use Case**:
  + Nhận thông báo (Actor: Student, Instructor): Email/push khi hoàn thành khóa học.
  + Quản lý thông báo (Actor: Admin): Gửi thông báo thủ công.
* **Sơ đồ**:
  + Use Case Diagram: "Nhận thông báo", "Quản lý thông báo".
  + Sequence Diagram: Luồng gửi thông báo qua Kafka.
  + ERD: Bảng Notifications.
* **API tối ưu**:
  + POST /api/notifications (Gửi thông báo)
  + GET /api/notifications?user\_id={id} (Xem thông báo)
  + **Tổng số API**: ~2 API.

### ****Giai đoạn 6: Tính năng chuyên biệt****

#### ****Recommendation Service (FastAPI)****

* **Use Case**:
  + Gợi ý cá nhân hóa (Actor: Student): Khóa học/bài học dựa trên sở thích.
* **Sơ đồ**:
  + Use Case Diagram: "Gợi ý cá nhân hóa".
  + Class Diagram: UserPreference (user\_id, preferences).
* **API tối ưu**:
  + GET /api/recommendations/personalized?user\_id={id} (Gợi ý cá nhân hóa)
  + POST /api/preferences (Lưu sở thích)
  + **Tổng số API**: ~2 API.

#### ****Live Class Service (Spring Boot)****

* **Use Case**:
  + Tham gia lớp học trực tiếp (Actor: Student, Instructor): Tạo lịch, tham gia, lưu bản ghi.
  + Quản lý lớp học (Actor: Admin): Xem, xóa lịch.
* **Sơ đồ**:
  + Use Case Diagram: "Tham gia lớp học trực tiếp", "Quản lý lớp học".
  + ERD: Bảng LiveSessions, UserSessionAttendance.
* **API tối ưu**:
  + POST /api/live-sessions (Tạo lịch)
  + GET /api/live-sessions?course\_id={id} (Xem lịch)
  + POST /api/live-sessions/:id/join (Tham gia)
  + **Tổng số API**: ~3 API.

#### ****Content Moderation Service (FastAPI)****

* **Use Case**:
  + Kiểm duyệt nội dung (Actor: Admin): Kiểm tra khóa học, media, review.
  + Báo cáo vi phạm (Actor: Student, Instructor).
* **Sơ đồ**:
  + Use Case Diagram: "Kiểm duyệt nội dung", "Báo cáo vi phạm".
  + ERD: Bảng ContentReports.
* **API tối ưu**:
  + POST /api/reports (Báo cáo vi phạm)
  + GET /api/reports (Xem báo cáo, chỉ Admin)
  + PUT /api/reports/:id (Xử lý báo cáo, chỉ Admin)
  + **Tổng số API**: ~3 API.

### ****Giai đoạn 7: Tối ưu hóa và khuyến mãi****

#### ****Analytics Service (Spring Boot)****

* **Use Case**:
  + Xem báo cáo học tập (Actor: Admin, Instructor): Thống kê thời gian học, tỷ lệ hoàn thành.
* **Sơ đồ**:
  + Use Case Diagram: "Xem báo cáo học tập".
  + ERD: Bảng UserAnalytics.
* **API tối ưu**:
  + GET /api/analytics?course\_id={id} (Thống kê khóa học)
  + GET /api/analytics/users?user\_id={id} (Thống kê người dùng)
  + **Tổng số API**: ~2 API.

#### ****Coupon Service (Spring Boot)****

* **Use Case**:
  + Áp dụng coupon (Actor: Student): Sử dụng mã giảm giá khi thanh toán.
  + Quản lý coupon (Actor: Admin): Tạo, xóa coupon.
* **Sơ đồ**:
  + Use Case Diagram: "Áp dụng coupon", "Quản lý coupon".
  + ERD: Bảng Coupons, UserCoupons.
* **API tối ưu**:
  + POST /api/coupons (Tạo coupon, chỉ Admin)
  + POST /api/coupons/apply (Áp dụng coupon)
  + GET /api/coupons (Xem danh sách coupon, chỉ Admin)
  + **Tổng số API**: ~3 API.

## ****Tổng số API và Use Case (Toàn bộ dự án)****

* **Tổng số API**: ~70-80 API (Giai đoạn 1: 32, Giai đoạn 2: ~7-8, Giai đoạn 3: ~5-6, Giai đoạn 4: ~6, Giai đoạn 5: ~9-10, Giai đoạn 6: ~8-9, Giai đoạn 7: ~5).
* **Tổng số Use Case**: ~20-25 Use Case, bao gồm:
  + Giai đoạn 1: ~8 Use Case (Đăng ký, Đăng nhập, Quản lý thông tin cá nhân, Quản lý người dùng, Quản lý khóa học, Quản lý danh mục/chương/bài học, Quản lý media, Kiểm duyệt media).
  + Giai đoạn 2: ~3 Use Case (Theo dõi tiến độ, Xem khóa học qua Frontend, Quản lý tiến độ).
  + Giai đoạn 3: ~3 Use Case (Thanh toán, Tìm kiếm, Quản lý thanh toán).
  + Giai đoạn 4: ~3 Use Case (Gợi ý khóa học, Tạo câu hỏi/tóm tắt, Gửi/Quản lý đánh giá).
  + Giai đoạn 5: ~3 Use Case (Làm bài kiểm tra, Nhận huy hiệu, Nhận thông báo).
  + Giai đoạn 6: ~3 Use Case (Gợi ý cá nhân hóa, Tham gia lớp học trực tiếp, Kiểm duyệt nội dung).
  + Giai đoạn 7: ~2 Use Case (Xem báo cáo học tập, Áp dụng coupon).

## ****Tránh viết thừa API****

* **Gộp các thao tác tương tự**:
  + Ví dụ: Sử dụng PUT /api/users/:id để cập nhật tất cả thông tin thay vì API riêng cho từng trường (email, password).
* **Sử dụng query parameter**:
  + Ví dụ: GET /api/sections?course\_id={id} thay vì API riêng cho mỗi khóa học.
* **Tái sử dụng API**:
  + Ví dụ: GET /api/courses được dùng bởi Student (xem khóa học), Instructor (quản lý khóa học), và Admin (kiểm duyệt).
* **Kiểm tra quyền truy cập**:
  + Sử dụng JWT và vai trò (role) để giới hạn quyền (ví dụ: chỉ Admin gọi DELETE /api/users/:id).
* **Tài liệu hóa API**:
  + Sử dụng **Swagger/OpenAPI** để liệt kê và kiểm tra các API, tránh trùng lặp.