ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

----📖----

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: KĨ THUẬT MÁY TÍNH

ĐỀ TÀI:

XÂY DỰNG HỆ THỐNG QUẢN LÍ THI TRỰC TUYẾN

Người hướng dẫn: **ThS. TRẦN HỒ THỦY TIÊN**

Sinh viên thực hiện: **LÊ NHẬT MINH**

Số thẻ sinh viên: **102210171**

Lớp**: 21TCLC_DT2**

Đà Nẵng, 06/2026

TÓM TẮT

Tên đề tài: Xây dựng hệ thống quản lí thi trực tuyến

Sinh viên thực hiện: Lê Nhật Minh

Số thẻ SV: 102210171 Lớp: 21TCLC_DT2

Hiện nay, hình thức thi trực tuyến đang trở nên phổ biến trong giáo dục. Tuy nhiên, bên cạnh những lợi ích về mặt linh hoạt thời gian và không gian, thi trực tuyến vẫn tồn tại nhiều bất cập. Một trong những hạn chế lớn là thiếu sự đa dạng trong cấu trúc đề thi, các hình thức trả lời còn đơn điệu và đặc biệt là chưa có giải pháp hiệu quả để kiểm soát hành vi gian lận trong quá trình làm bài. Việc này ảnh hưởng đến tính công bằng trong đánh giá kết quả học tập và gây áp lực lớn cho giảng viên trong khâu tổ chức, giám sát và chấm điểm.

1. Đặt vấn đề Mặc dù thi trực tuyến mang lại sự linh hoạt về không gian và thời gian, các giải pháp hiện hành vẫn tồn tại nhiều hạn chế. Việc phụ thuộc vào các công cụ rời rạc khiến khâu quản lý ngân hàng câu hỏi, sinh đề theo ma trận và tổ chức ca thi trở nên phức tạp. Bên cạnh đó, sự đơn điệu trong hình thức câu hỏi và sự thiếu hụt các cơ chế chống gian lận hiệu quả không chỉ làm suy giảm tính công bằng mà còn gia tăng áp lực giám sát cho giảng viên.

2. Giải pháp đề xuất Để giải quyết những thách thức trên, đề tài "Xây dựng hệ thống quản lý thi trực tuyến" được triển khai. Hệ thống cung cấp một nền tảng toàn diện với 3 phân quyền (Quản trị viên, Giảng viên, Sinh viên), hỗ trợ linh hoạt 5 loại hình câu hỏi và cho phép sinh mã đề tự động có trọng số (weighted). Đặc biệt, hệ thống tích hợp các cơ chế Anti-cheat mạnh mẽ như: cảnh báo chuyển tab, chặn thao tác sao chép – dán và theo dõi chi tiết thời gian làm bài, nhằm đảm bảo tính minh bạch tối đa.

3. Phương pháp & Công nghệ áp dụng Hệ thống được thiết kế theo các tiêu chuẩn kiến trúc phần mềm hiện đại: Clean Architecture, Domain-Driven Design (DDD) và CQRS/MediatR. Backend: C#, ASP.NET Core 8, IdentityServer. Frontend: Giao diện trực quan với React 19 (HTML, CSS, JavaScript). Cơ sở hạ tầng: SQL Server, Redis, SignalR (Real-time), Hangfire (Xử lý tác vụ ngầm).

4. Kết quả đạt được Hệ thống đã hoàn thiện 100% chức năng theo yêu cầu với hiệu năng thực tế ấn tượng như triển khai thành công 106 API endpoint, 65 Command và 45 Query, thời gian phản hồi API trung bình đạt 6.0 ms, tốc độ sinh 3 mã đề weighted trung bình chỉ mất 122.8 ms, dữ liệu thử nghiệm vận hành trơn tru với 62 câu hỏi, 12 đề và 10 ca thi.

Kính mong nhận được những nhận xét, góp ý quý báu từ quý thầy cô hội đồng để nhóm có thể tiếp tục hoàn thiện đề tài, hướng tới việc ứng dụng thực tiễn trong công tác đào tạo tại trường.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Lê Nhật Minh Số thẻ sinh viên: 102210171

Lớp: 21TCLC_DT2 Khoa: Công nghệ Thông tin Ngành: Kĩ thuật máy tính

1. Tên đề tài đồ án: Xây dựng hệ thống quản lí thi trực tuyến
2. Đề tài thuộc diện: ☐ *Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. Các số liệu và dữ liệu ban đầu: không có
4. *Nội dung các phần thuyết minh và tính toán:*

   Nội dung các phần thuyết minh bao gồm:

   - **Mở đầu:** Giới thiệu tổng quan vấn đề, mục đích và phạm vi của đề tài, hướng tiếp cận và bố cục của đồ án.
   - **Chương 1:** Cở sở lý thuyết: Trình bày các lý thuyết, các công cụ và công nghệ đã sử dụng
   - **Chương 2**: Phân tích và thiết kế: Chương này trình bày các biểu đồ, hướng tiếp cận và cách giải quyết các vấn đề.
   - **Chương 3:** Triển khai và đánh giá: Mô tả cách cài đặt, vận hành, kiểm thử hệ thống và kết quả đạt được.
   - **Kết luận:** Trình bày các kết quả đạt được, chỉ ra các hạn chế và đề xuất hướng phát triển.
5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ): không có bản vẽ đồ thị*
6. Họ tên người hướng dẫn: ThS. Trần Hồ Thủy Tiên
7. *Ngày giao nhiệm vụ đồ án: ……. /……./2026*
8. *Ngày hoàn thành đồ án: ...../...../2026*

Đà Nẵng, ngày……tháng……năm 2026

Trưởng khoa CNTT **Người hướng dẫn**

ThS. Trần Hồ Thủy Tiên

LỜI NÓI ĐẦU

Lời đầu tiên em xin cám ơn thầy ThS.Trần Hồ Thủy Tiên trong thời gian qua đã giúp đỡ em rất nhiều từ lúc định hướng đề tài cho đến giai đoạn hoàn thiện trong quá trình làm đồ án tốt nghiệp.

Hơn nữa, em cũng xin cảm ơn toàn bộ giảng viên của Khoa Công Nghệ Thông Tin, Trường Đại học Bách Khoa-Đại học Đà Nẵng, trong suốt những năm em theo học tại trường, đã nhiệt tình giảng dạy và xây dựng nền tảng kỹ thuật vững chắc giúp em có thể thực hiện được đồ án này.

Để có thể tiến tới đến ngày hôm nay, em rất biết ơn gia đình và người thân đã cho em cơ hội, nuôi dưỡng, khích lệ, động viên cũng như tạo điều kiện tốt nhất để em được vững bước tiếp tục học tập. Cảm ơn những người bạn, anh chị em đã luôn đồng hành và khích lệ giúp đỡ em trong xuất quá trình học tập.

Tuy nhiên trong học tập, cũng như trong quá trình làm đồ án tốt nghiệp không thể tránh khỏi những thiếu sót, em rất mong được sự góp ý quý báu của thầy cô giáo cũng như các bạn để em có cái nhìn trực quan và kết quả được hoàn thiện hơn cũng như có thêm nhiều kinh nghiệm cho công việc sau này.

Một lần nữa em xin chân thành cảm ơn!

Đà Nẵng, ngày … tháng … năm 2026

Sinh viên thực hiện

Lê Nhật Minh

CAM ĐOAN

Tôi xin cam đoan:

1. Báo cáo đồ án tốt nghiệp tên đề tài: Xây dựng hệ thống quản lí thi trực tuyến là công trình nghiên cứu của chính cá nhân tôi dưới sự hướng dẫn trực tiếp của giảng viên ThS. Trần Hồ Thủy Tiên.
2. Tôi đã tự đọc nghiên cứu, dịch tài liệu và tổng hợp các kiến thức đã làm nên báo cáo này và đảm bảo không sao chép ở bất cứ đâu.
3. Những lý thuyết trong luận văn đều được sử dụng tài liệu như tôi đã thao khảo ở phần tài liệu tham khảo đã có trong báo cáo.

   Nếu có vi phạm, tôi xin chịu hoàn toàn trách nhiệm.

Đà Nẵng, ngày……tháng……năm 2026

Sinh viên thực hiện

Lê Nhật Minh

MỤC LỤC

TÓM TẮT i
NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP ii
LỜI NÓI ĐẦU iii
CAM ĐOAN iv
MỤC LỤC v
DANH SÁCH CÁC BẢNG x
DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT xi
MỞ ĐẦU 1
CHƯƠNG I: CƠ SỞ LÝ THUYẾT VÀ CÔNG CỤ SỬ DỤNG 3
1.1 Tổng quan về website 3
1.2 Giới thiệu Domain-Driven Design: 3
1.2.1. Lý do tại sao nên sử dụng Domain-Driven Design: 3
1.2.2. Nguyên tắc thiết kế trong Domain-Driven Design (DDD) 4
1.2.3. Các mẫu kiến trúc hỗ trợ tốt DDD: 6
1.2.4. Quy trình triển khai DDD 7
1.2.5. Những sai lầm phổ biến khi áp dụng DDD 7
1.2.6. Khi nào nên dùng Domain-Driven Design 7
1.3. Giới thiệu Command Query Responsibility Segregation (CQRS) 8
1.3.1. Giới thiệu 8
1.3.2. Mục tiêu và lợi ích của CQRS 8
1.3.3. Kiến trúc tổng thể của CQRS 9
1.3.4. Kỹ thuật triển khai CQRS hiệu quả 10
1.3.5. Khi nào nên sử dụng CQRS 11
1.3.6. Các lưu ý khi áp dụng CQRS 11
1.3.6. Áp dụng CQRS trong hệ thống 12
1.4 . Kiến trúc tổng thể của hệ thống 13
CHƯƠNG II: PHÂN TÍCH - THIẾT KẾ HỆ THỐNG 15
2. Phân tích yêu cầu 15
2.1. Tổng quan 15
2.2. Yêu cầu các chức năng 15
2.3. Biểu đồ ca sử dụng 17
2.3.2. Sơ đồ ca sử dụng 18
2.4. Thiết kế cơ sở dữ liệu 40

2.4.1. Mô hình thiết kế cơ sở dữ liệu....40
2.4.2. Chi tiết các bảng dữ liệu....41
CHƯƠNG III: TRIỂN KHAI THỰC TẾ....52
3. Môi trường, kĩ thuật và công cụ phát triển dự án....52
3.1. Môi trường cài đặt....52
3.2. Phân tích bảo mật và xác thực....54
3.2.1. Kiến trúc xác thực (Authentication)....54
3.2.2. Các biện pháp bảo mật đã triển khai....54
3.3. Cơ chế chống gian lận (Anti-cheat)....55
3.3.1. Giám sát phía client....55
3.3.2. Xử lý phía server và tự động nộp bài....55
3.3.3. Đánh giá hiệu quả và hướng hoàn thiện....56
3.4. Đánh giá UX/UI và khả năng sử dụng....56
3.4.1. Kiến trúc giao diện và nguyên tắc thiết kế....56
3.4.2. Quy trình thao tác theo vai trò....57
3.4.3. Đánh giá khả năng sử dụng (usability)....57
3.4.4. So sánh với Moodle và Google Form....58
3.5. Triển khai thực nghiệm....58
3.5.1. Kiến trúc runtime môi trường local....58
3.5.2. Quy trình khởi chạy hệ thống....60
3.5.3. Giới hạn phạm vi và hướng phát triển....61
3.6. Kết quả thực tế — minh họa giao diện....61
3.6.1. Giao diện đối với người dùng là quản trị viên....62
3.6.2 Giao diện đối với người dùng là giảng viên....65
3.6.3. Giao diện đối với người dùng là sinh viên....74
3.6.4Giao diện đối với người dùng là thí sinh tự do....75
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN....76
4.1. Kết luận....76
4.1.1. Kết quả đạt được....76
4.1.2. Kiểm thử và đánh giá thực tế....76
4.1.3. Bài học kinh nghiệm (Lessons learned)....76
4.1.4. Hạn chế....77
4.2. Hướng phát triển....77
4.2.1. Tích hợp AI....77
4.2.2. Mở rộng ứng dụng di động....77

TÀI LIỆU THAM KHẢO 78
PHỤ LỤC 79
Phụ lục A. Liên kết mã nguồn trên GitHub 79
Phụ lục B. Hướng dẫn truy cập và chạy mã nguồn 79

DANH SÁCH CÁC HÌNH VẼ

Hình 1: Mô hình CQRS 9
Hình 2: Kiến trúc tổng thể hệ thống 13
Hình 3: Sơ đồ usecase của quản trị viên 18
Hình 4: Sơ đồ usecase của giảng viên 19
Hình 5: Sơ đồ usecase của sinh viên 20
Hình 6: Sơ đồ usecase của thí sinh tự do 21
Hình 7: Sơ đồ tuần tự các chức năng đăng nhập 21
Hình 8: Sơ đồ tuần tự của chức năng xem học phần 22
Hình 9: Sơ đồ tuần tự chức năng tạo học phần 23
Hình 10: Sơ đồ tuần tự chức năng thêm chương 24
Hình 11: Sơ đồ tuần tự chức năng thêm câu hỏi 25
Hình 12: Sơ đồ tuần tự chức năng xem chi tiết câu hỏi 26
Hình 13: Sơ đồ tuần tự chức năng tạo mẫu đề 27
Hình 14: Sơ đồ tuần tự chức năng xem chi tiết mẫu đề 28
Hình 15: Sơ đồ tuần tự chức năng tạo mạch chủ đề 29
Hình 16: Sơ đồ tuần tự chức năng tạo đề thi 30
Hình 17: Sơ đồ tuần tự chức năng tạo phiên bản đề 31
Hình 18: Sơ đồ tuần tự chức năng xem phiên bản đề 32
Hình 19: Sơ đồ tuần tự chức năng xem chi tiết đề 33
Hình 20: Sơ đồ tuần tự chức năng tạo phiên thi 34
Hình 21: Sơ đồ tuần tự chức năng xem chi tiết phiên thi 35
Hình 22: Sơ đồ hoạt động chức năng đăng nhập 38
Hình 23: Sơ đồ tuần tự chức năng tạo học phần 39
Hình 24: Mô hình thiết kế cơ sở dữ liệu 41
Hình 25: Sơ đồ ERD 48
Hình 26: Kiến trúc runtime môi trường triển khai local 59
Hình 27: Màn hình đăng nhập quản trị viên 62
Hình 28: Màn hình danh sách giảng viên 62
Hình 29: Màn hình chi tiết giảng viên 63
Hình 30: Màn hình khóa tài khoản 63
Hình 31: Màn hình mở tài khoản 64
Hình 32: Màn hình đặt lại mật khẩu 64
Hình 33: Màn hình danh sách học phần 65
Hình 34: Màn hình chi tiết học phần 65
Hình 35: Màn hình thêm chương 66
Hình 36: Màn hình thêm câu hỏi 66
Hình 37: Màn hình chi tiết câu hỏi 67
Hình 38: Màn hình khung đề thi 67
Hình 39: Màn hình chi tiết khung đề thi 68
Hình 40: Màn hình tạo khung đề thi 68
Hình 41: Màn hình tạo đề thi 69
Hình 42: Màn hình danh sách đề thi 69
Hình 43: Màn hình tạo đề thi 70
Hình 44: Màn hình chi tiết đề thi 71
Hình 45: Màn hình danh sách phiên thi 71

Hình 46: Màn hình tạo phiên thi mới 71
Hình 47: Màn hình chi tiết phiên thi 72
Hình 48: Màn hình danh sách thí sinh 72
Hình 49: Màn hình giám sát thí sinh dự thi 73
Hình 50: Màn hình kết quả thi 73
Hình 51: Màn hình chấm điểm câu hỏi tự luận 74
Hình 52: Màn hình tham gia thi 74
Hình 53: Màn hình cảnh báo rời khỏi màn hình 75
Hình 54: Màn hình cảnh báo 75

DANH SÁCH CÁC BẢNG

Bảng 1: Mapping các thành phần DDD trong hệ thống 8
Bảng 2: Mapping Command và Query trong hệ thống 12
Bảng 3:Bảng tác nhân 17
Bảng 4: Bảng mô tả Use Case theo quản trị viên 36
Bảng 5: Bảng mô tả Use Case theo giảng viên 37
Bảng 6: Bảng mô tả Use Case theo sinh viên 37
Bảng 7: Bảng mô tả Use Case theo thí sinh tự do 37
Bảng 8: Bảng Role 41
Bảng 9: Bảng User 41
Bảng 10: Bảng UserRole 41
Bảng 11: Bảng học phần 42
Bảng 12: Bảng Chương 42
Bảng 13: Bảng câu hỏi 42
Bảng 14: Bảng lựa chọn ghép nối 43
Bảng 15: Bảng lựa chọn câu hỏi chọn nhiều 43
Bảng 16: Bảng lựa chọn câu hỏi chọn một 44
Bảng 17: Bảng khung đề thi 44
Bảng 18: Bảng phần của khung đề thi 44
Bảng 19: Bảng cấu hình độ khó 44
Bảng 20: Bảng cấu hình độ khó 44
Bảng 21: Bảng đề thi 45
Bảng 22: Bảng phiên bản đề thi 45
Bảng 23: Bảng câu hỏi của phiên bản đề thi 45
Bảng 24: Bảng phiên thi 46
Bảng 25: Bảng hành động 46
Bảng 26: Bảng mã thí sinh 47
Bảng 27: Bảng đáp án 47
Bảng 28: Bảng lớp học phần 48
Bảng 29: Quan hệ một-nhiều (1-N) 49
Bảng 30: Hệ thống dùng EF Core với OwnsMany/OwnsOne 50
Bảng 31: Ngoài khóa chính, EF Core và ASP.NET Identity tạo index trên khóa ngoại và cột tra cứu thư 51
Bảng 32: Tóm tắt mức độ bảo vệ các hạng mục chính 56
Bảng 33: So sánh nhanh ba hướng triển khai 58
Bảng 34: Các thành phần runtime 60
Bảng 35: Tóm tắt kết quả đạt được 76

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Từ viết tắt	Diễn giải
API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
DDD	Domain-Driven Design
CQRS	Command Query Responsibility Segregation
DB	Database
HTTP	Hypertext Transfer Protocol

MỞ ĐẦU

1. Tổng quan đề tài

Trong bối cảnh công nghệ thông tin ngày càng phát triển mạnh mẽ, việc ứng dụng các hệ thống trực tuyến vào lĩnh vực giáo dục đang trở thành xu hướng tất yếu. Đặc biệt, thi cử trực tuyến đang dần được nhiều trường đại học, cao đẳng và cơ sở đào tạo áp dụng nhằm tiết kiệm thời gian, tối ưu nguồn lực và tạo điều kiện thuận lợi cho cả giảng viên lẫn sinh viên. Tuy nhiên, bên cạnh những lợi ích thiết thực, hình thức thi trực tuyến hiện nay vẫn tồn tại nhiều bất cập và thách thức cần được giải quyết.

Một trong những vấn đề nổi bật là tính bảo mật và công bằng trong thi cử. Hệ thống thi trực tuyến hiện nay chưa thực sự đảm bảo được việc giám sát nghiêm túc như thi trực tiếp, dẫn đến nguy cơ gian lận cao như: mở tài liệu, sao chép nội dung, nhờ người thi hộ, chuyển tab trình duyệt,... gây ảnh hưởng đến tính khách quan và độ tin cậy của kết quả đánh giá. Bên cạnh đó, nhiều hệ thống thi vẫn còn thiếu linh hoạt trong việc tạo đề, thường chỉ hỗ trợ một số dạng câu hỏi cơ bản, chưa đáp ứng được nhu cầu đánh giá đa chiều theo năng lực người học.

Xuất phát từ thực tiễn đó, đề tài **"Xây dựng hệ thống quản lí thi trực tuyến"** được lựa chọn và phát triển nhằm mang lại một giải pháp toàn diện, vừa hỗ trợ giảng viên tạo đề một cách linh hoạt với nhiều cấp độ và hình thức câu hỏi khác nhau, vừa tích hợp các cơ chế kỹ thuật giúp hạn chế tối đa hành vi gian lận trong thi cử.

2. Mục đích của đề tài

Mục tiêu của đề tài "Xây dựng hệ thống thi trực tuyến thông minh với tính năng đa dạng câu hỏi và biện pháp chống gian lận hỗ trợ cho giảng viên Trường Đại học Bách Khoa" là xây dựng một hệ thống thi trực tuyến hoàn chỉnh, hỗ trợ giảng viên dễ dàng tạo đề thi với nhiều cấp độ và hình thức câu hỏi khác nhau như trắc nghiệm, tự luận, kéo thả, ghép nối,... đồng thời cung cấp giao diện thân thiện, dễ sử dụng cho cả giảng viên và sinh viên. Hệ thống hướng đến việc tích hợp các biện pháp kỹ thuật nhằm hạn chế hành vi gian lận trong thi cử như cảnh báo khi rời khỏi màn hình, chặn thao tác sao chép – dán, theo dõi thao tác chuột và ghi nhận hành vi bất thường. Ngoài ra, hệ thống còn hỗ trợ chấm điểm tự động với các dạng câu hỏi phù hợp, thống kê kết quả thi, giúp

giảm tải công việc cho giảng viên, đồng thời góp phần nâng cao tính công bằng, minh bạch trong đánh giá kết quả học tập của sinh viên.

3. Tính năng

Đối với admin:

- Tạo / sửa / xóa tài khoản giảng viên và sinh viên
- Phân quyền (Admin, Giảng viên, Sinh viên)
- Khóa / mở khóa tài khoản
- Thống kê số lượng kỳ thi, sinh viên, điểm trung bình
- Thêm / sửa / xóa: Môn học, Chương học, Ngân hàng câu hỏi

Đối với giảng viên :

- Tạo và quản lý câu hỏi
- Quản lý ngân hàng câu hỏi cá nhân hoặc theo chương/môn học
- Import / Export câu hỏi từ file CSV
- Tạo kỳ thi, cấu hình thời gian và danh sách sinh viên tham gia
- Giao đề thi khi sinh viên bắt đầu làm bài
- Theo dõi quá trình làm bài của sinh viên
- Chấm điểm thủ công các câu hỏi tự luận
- Xem và duyệt kết quả thi

Đối với sinh viên :

- Xem danh sách kỳ thi được phân công
- Tham gia kỳ thi đúng thời gian đã lên lịch
- Tự động lưu tiến trình làm bài
- Nộp bài thủ công hoặc tự động khi hết giờ
- Xem kết quả bài thi sau khi chấm điểm
- Nhận phản hồi từ giảng viên (nếu có)

CHƯƠNG I: CƠ SỞ LÝ THUYẾT VÀ CÔNG CỤ SỬ DỤNG

1.1 Tổng quan về website

"**Website quản lí thi trực**" là một nền tảng web hỗ trợ tổ chức và quản lý các kỳ thi theo hình thức trắc nghiệm và tự luận. Hệ thống cho phép quản trị viên quản lý người dùng, giảng viên tạo ngân hang câu hỏi và kỳ thi, còn sinh viên có thể tham gia làm bài thi trực tuyến.

Website được xây dựng nhằm mục tiêu tự động hóa quy trình thi cử, giảm tải công việc thủ công và tăng tính minh bạch, chính xác trong chấm điểm. Ngoài ra, hệ thống hỗ trợ cả chấm điểm tự động (cho câu hỏi trắc nghiệm) và chấm điểm thủ công (cho câu hỏi tự luận), giúp đáp ứng nhiều nhu cầu kiểm tra khác nhau trong môi trường giáo dục hiện đại

1.2 Giới thiệu Domain-Driven Design:

Domain-Driven Design (DDD) là một phương pháp thiết kế phần mềm được Eric Evans giới thiệu trong cuốn sách "Domain-Driven Design: Tackling Complexity in the Heart of Software" (2003).Mục tiêu chính của DDD là:

- Tập trung vào logic nghiệp vụ, không chỉ là kỹ thuật.
- Tách biệt rõ ràng domain logic với hạ tầng (infrastructure).
- Tăng khả năng mở rộng, dễ bảo trì khi hệ thống lớn lên.
- Mô hình hóa domain phức tạp một cách sát thực tế.

1.2.1. Lý do tại sao nên sử dụng Domain-Driven Design:

- Khi phát triển một hệ thống phần mềm, đặc biệt là các hệ thống có quy mô lớn hoặc phục vụ nghiệp vụ phức tạp, việc duy trì chất lượng và khả năng mở rộng của phần mềm là một thách thức lớn. Domain-Driven Design (DDD) ra đời nhằm giải quyết các vấn đề phổ biến sau:
  - Nghiệp vụ ngày càng phức tạp theo thời gian

- Khi hệ thống mở rộng, các quy tắc và quy trình nghiệp vụ (business rules) cũng phát triển theo, trở nên khó hiểu và khó quản lý nếu không có cách tổ chức hợp lý. DDD giúp mô hình hóa nghiệp vụ một cách chính xác, rõ ràng, phản ánh đúng thực tế hoạt động kinh doanh.

- Logic nghiệp vụ bị phân tán khắp nơi
  - Trong nhiều dự án truyền thống, logic nghiệp vụ bị trải đều ở nhiều lớp khác nhau như controller, service, repository…, gây khó khăn khi bảo trì hoặc mở rộng. DDD đưa ra mô hình rõ ràng (Entity, Value Object, Domain Service...) giúp tập trung và bảo vệ domain logic.
- Ngôn ngữ không thống nhất giữa lập trình viên và chuyên gia nghiệp vụ
  - Lập trình viên thường dùng thuật ngữ kỹ thuật, trong khi chuyên gia nghiệp vụ lại dùng từ ngữ chuyên ngành. Điều này dễ gây hiểu nhầm, sai lệch yêu cầu. DDD giải quyết bằng cách sử dụng ngôn ngữ chung (Ubiquitous Language) — một từ vựng thống nhất được tất cả các bên cùng sử dụng trong giao tiếp và trong mã nguồn.
- Khó bảo trì và mở rộng hệ thống
  - Khi nghiệp vụ thay đổi (ví dụ: thêm quy tắc tính phí, điều kiện đặt hàng…), nếu domain không được tổ chức rõ ràng, việc cập nhật rất dễ gây lỗi, ảnh hưởng đến các phần khác của hệ thống. DDD giúp cô lập domain logic, chia hệ thống thành các Bounded Context độc lập, từ đó giảm rủi ro và tăng khả năng mở rộng.

1.2.2. Nguyên tắc thiết kế trong Domain-Driven Design (DDD)

- DDD không chỉ là cách mô hình hóa nghiệp vụ và còn là cách thiết kế phần mềm định hướng theo nghiệp vụ, nhấn mạnh sự phân tách rõ ràng giữa domain logic và các phần ký thuật. Các nguyên tắc thiết kế chính trong DDD bao gồm:
  - Separation of Concerns (Phân tách mối quan tâm)
    - DDD khuyến khích tách phần domain logic (nghiệp vụ cốt lõi) ra khỏi hạ tầng kỹ thuật như database, UI, API,...
    - Điều này giúp logic nghiệp vụ không bị trộn lẫn vào các phần khác như controller, data access, hay view.

- Đối với những trường hợp không thể đặt logic nghiệp vụ ở domain, thì sẽ bỏ tập trung vào domain service.
- Điều này sẽ giúp dễ dàng kiểm thử domain một cách độc lập

o Encapsulation (Tính đóng gói)

- Aggregate giúp đóng gói toàn bộ các thực thể (entity) và giá trị (value object) có liên quan.
- Tất cả các thay đổi đến aggregate phải thông qua aggregate root để đảm bảo tính toàn vẹn mà không phá vỡ các quy tắc nghiệp vụ
- Tính đóng gói giúp kiểm soát trạng thái của hệ thống tốt hơn, tránh lỗi do sửa trực tiếp dữ liệu không đúng cách

o Immutability (Bất biến)

- Value Object phải được thiết kế bất biến, có nghĩa là sau khi tạo ra thì không thay đổi được
- Điều này đảm bảo dữ liệu ổn định trong vòng đời một nghiệp vụ và tránh lỗi do thay đổi trạng thái ngoài ý muốn

o Consistency Boundaries (Giới hạn nhất quán)

- Aggregate đóng vai trò như một boundary đảm bảo tính nhất quán dữ liệu bên trong nó
- Khi có nhiều aggregate, DDD cho phép eventual consistency giữa các aggregate, thay vì bắt buộc consistency toàn cục
- Điều này phù hợp với mô hình phân tán hoặc microservices, nơi khó duy trì tính nhất quán tức thì trên toàn hệ thống.

o Ubitoquitous Language (Ngôn ngữ chung)

- Nguyên tắc xuyên suốt của DDD: tất cả mọi người (developer, tester, business expert) đều dùng một ngôn ngữ thống nhất để trao đổi và viết mã
- Ngôn ngữ này phản ánh đúng ngữ cảnh nghiệp vụ, giúp giảm hiểu nhầm và tăng độ chính xác khi triển khai hệ thống

o Domain-Centric Design (Lấy domain làm trung tâm)

- Thiết kế phần mềm xoay quanh các khái niệm nghiệp vụ thực tế, không xoay quanh công nghệ
- Mọi quyết định thiết kế, đặt tên, kiến trúc lớp đều phục vụ việc phản ánh đúng hành vi nghiệp vụ
- Điều đó giúp hệ thống dễ mở rộng, dễ hiểu và thay đổi theo nghiệp vụ mà không phá vỡ kiến trúc tổng thể

o Single Source Of Truth (Một nguồn dữ liệu đúng duy nhất)

- Mỗi entity nên có một định danh duy nhất và tồn tại trong một Aggregate duy nhất. Điều này giúp tránh trùng lặp dữ liệu và đồng bộ không chính xác
- Repository nên chỉ thao tác với Aggregate Root.

o Keep Domain Pure (Giữ cho domain "thuần")

- Domain layer không nên phụ thuộc vào framework, thư viện ngoài hay công nghệ lưu trữ.
- Domain chỉ nên phụ thuộc vào chính domain (các entity, value object, domain service, event, ...)
- Giúp domain dễ test, dễ di chuyển sang môi trường khác mà không bị "trói" bởi công nghệ

1.2.3. Các mẫu kiến trúc hỗ trợ tốt DDD:

- Hexagonal Architecture (Ports and Adapters):
  - Giúp tách biệt domain logic với thế giới bên ngoài (UI, DB, API, message queue,...)
  - Domain chỉ giao tiếp thông qua Ports (interface), còn adapter đảm nhiệm logic cụ thể.
- CQRS (Command Query Responsibility Segregation)
  - Tách biệt Command (ghi dữ liệu, thay đổi trạng thái) và Query (đọc dữ liệu)
  - Rất phù hợp khi dùng với DDD vì giúp domain tập trung vào xử lý nghiệp vụ (command), còn phần đọc có thể tối ưu riêng
- Event Sourcing
  - Lưu trữ các thay đổi dưới dạng chuỗi event thay vì trạng thái hiện tại

- Rất phù hợp với các hệ thống đòi hỏi khả năng khôi phục, rollback nghiệp vụ

1.2.4. Quy trình triển khai DDD

- Khám phá domain cùng chuyên gia nghiệp vụ
- Xác định bounded context
- Thiết kế mô hình domain: xác định entity, value object, aggregate, service, repository
- Tọa ngôn ngữ chung và áp dụng xuyên suốt từ phân tích đến viết mã
- Triển khai kiến trúc phù hợp: hexagonal, CQRS, microservices,...

1.2.5. Những sai lầm phổ biến khi áp dụng DDD

- Áp dụng DDD cho hệ thống đơn giản: DDD phù hợp cho hệ thống phức tạp, không nên dùng nếu nghiệp vụ đơn giản
- Chỉ dùng DDD như một mẫu thiết kế mà không hiểu triết lý của nó
- Thiếu sự hợp tác với chuyên gia nghiệp vụ, dẫn đến model không phản ảnh đúng domain thực tế
- Tập trung quá nhiều vào kỹ thuật mà bỏ quên tư duy theo nghiệp vụ

1.2.6. Khi nào nên dùng Domain-Driven Design

- Nghiệp vụ phức tạp và có thể thay đổi theo thời gian
- Cần khả năng mở rộng hệ thống và dễ bảo trì
- Cần tối ưu khả năng cộng tác giữa nhà phát triển và chuyên gia

1.2.7. Áp dụng Domain-Driven Design vào hệ thống

Bảng Mapping các thành phần DDD vào trong hệ thống

Khái niệm DDD	**Triển khai trong code**	**Ví dụ class/file**
Aggregate Root	Entity gốc, boundary giao dịch	Exam, ExamSession, Question, User
Entity (con)	Thành phần thuộc aggregate	ExamVersion, Participant, ParticipantAnswer
Value Object	Bất biến, validate nghiệp vụ	Email, ExamSessionCode, ScoreEvaluate
Domain Event	Sự kiện nghiệp vụ	CheckCorrectnessEvent
Application	Logic sinh đề (không có Domain Service riêng)	GenerateExamVersionsHandler

Bảng 1: Mapping các thành phần DDD trong hệ thống

1.3. Giới thiệu Command Query Responsibility Segregation (CQRS)

1.3.1. Giới thiệu

- CQRS (Command Query Responsibility Segregation) là một mẫu kiến trúc phần mềm được giới thiệu bởi Greg Young, nhằm tách biệt rõ ràng giữa hai loại thao tác trong hệ thống:
  - Command: thay đổi trạng thái hệ thống (ghi, cập nhất, xóa)
  - Query: truy vấn dữ liệu (chỉ đọc, không thay đổi hệ thống)
- CQRS đặc biệt hiệu quả khi áp dụng cho các hệ thống có nghiệp vụ phức tạp, yêu cầu hiệu năng cao, hoặc có khối lượng đọc và ghi không cân bằng

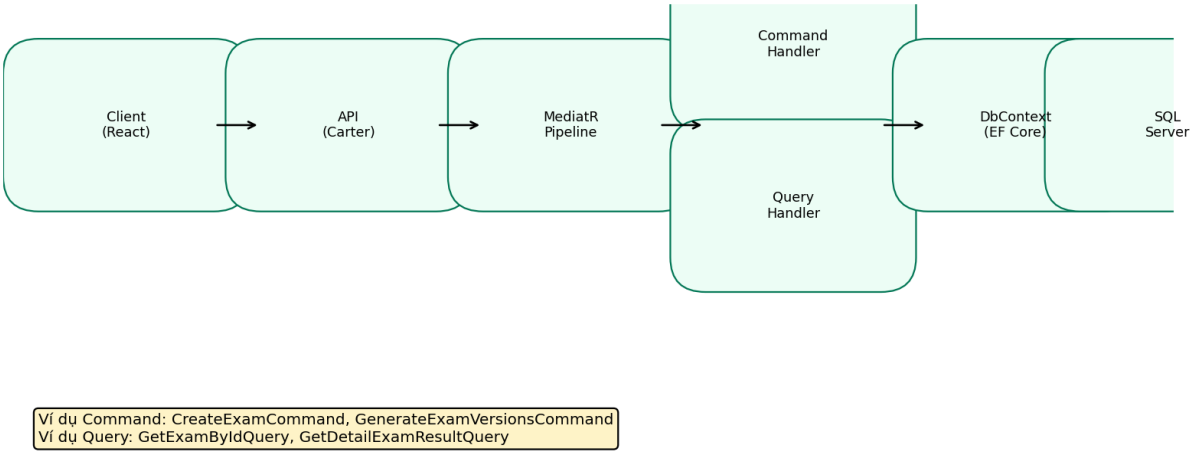
1.3.2. Mục tiêu và lợi ích của CQRS

- Tối ưu hóa hiệu năng
  - Tách biệt phần đọc và phần ghi giúp mỗi phần có thể được tối ưu độc lập (ví dụ: dùng cơ sở dữ liệu NoSQL cho phần đọc, SQL cho phần ghi).

- Giảm độ phức tạp xử lý đồng thời (concurrency), vì ghi và đọc không dùng chung mô hình.

- Dễ mở rộng (scalability)
  - Có thể scale out riêng biệt phần đọc và ghi
  - Đặc biệt phù hợp với hệ thống có tần suất đọc rất cao
- Hỗ trợ rõ ràng nguyên tắc "Single Responsibility" (SRP)
  - Command và Query được xử lý bởi những component khác nhau, giúp mã nguồn dễ hiểu, dễ test, dễ bảo trì
- Tích hợp tốt với Event Sourcing và Domain-Driven Design
  - CQRS thường được kết hợp với Event Sourcing để lưu trữ các thay đổi dưới dạng sự kiện
  - Kết hợp với DDD để quản lý nghiệp vụ phức tạp bằng cách giữ logic ghi trong domain model và logic đọc trong DTO đơn giản

1.3.3. Kiến trúc tổng thể của CQRS



Hình 1: Mô hình CQRS

Nhân yêu cầu thay đổi từ client
  - Xử lý qua Command Handler và Domain Model
  - Ghi dữ liệu vào Write Database
  - Có thể phát sinh Domain Event
- Query Side (phía đọc)

- Chỉ xử lý việc truy vấn
- Không liên quan đến logic nghiệp vụ
- Truy cập vào Read Database - thường là một dữ liệu được tối ưu cho việc đọc

1.3.4. Kỹ thuật triển khai CQRS hiệu quả

- Tách 2 cơ sở dữ liệu (Write Database và Read Database)
  - Một trong những điểm mạnh của CQRS là cho phép chia tách hoàn toàn phần ghi và phần đọc. Việc này giúp hệ thống đạt được:
  - Hiệu suất cao hơn: Đọc không bị khóa ghi, ghi không bị ảnh hưởng bởi lượng đọc lớn
  - Tối ưu hóa riêng biệt:
    - DB ghi: tập trung vào ràng buộc dữ liệu, tính toàn vẹn (SQL, transaction mạnh)
    - DB đọc: cấu trúc theo ViewModel, có thể là NoSQL, ElasticSearch, Redis,...
- Đồng bộ hóa dữ liệu giữa 2 database:
  - Để giữ cho Read Database luôn gần đúng (eventual consistency) với Write Database, ta cần sử dụng các kỹ thuật đồng bộ dữ liệu:
  - Synchronous Update (anti-pattern, ít dùng)
    - Sau khi ghi vào Write DB, ta ngay lập tức cập nhật Read DB
    - Đơn giản nhưng không đảm bảo hiệu suất và độ ổn định, dễ fail nếu Read DB lỗi
  - Event-Driven Architecture (phổ biến nhất)
    - Sau khi command được xử lý, phát sinh Domain Event, event này được:
    - Gửi vào Message Broker (như Kafka, RabbitMQ, ...)
    - Consumer lắng nghe và cập nhật Read DB
    - Ưu điểm:
    - Giảm coupling giữa write/read
    - Khả năng chịu lỗi tốt
    - Mở rộng dễ dàng (1 event có thể update nhiều nơi)

- Projection (kết hợp Event Sourcing)
  - Nếu dùng Event Sourcing, Read DB sẽ được build lại từ các sự kiện đã xảy ra
  - Luôn đảm bảo consistency theo logic nghiệp vụ
  - Phức tạp hơn, yêu cầu nhiều tài nguyên hơn

- Cách tối ưu hiệu suất phần đọc
  - Vì đọc thường chiếm 80 – 95 % lượng truy cập, nên CQRS cho phép áp dụng nhiều tối ưu:
  - Dùng cache layer: Redis / Memory Cache
    - Lưu ViewModel phổ biến trong cache (dưới dạng JSON)
    - Dễ kết hợp với CQRS vì phần đọc đã tách biệt, ít side effect
  - Dùng cở sở dữ liệu NoSQL
    - Dùng MongoDB / Elasticsearch cho truy vấn linh hoạt
    - Không bị giới hạn bởi lược đồ quan hệ phức tạp

1.3.5. Khi nào nên sử dụng CQRS

- CQRS không nên áp dụng cho mọi dự án. Tuy nhiên, nó rất hữu ích trong các tình huống sau:
  - Hệ thống có nghiệp vụ phức tạp và logic ghi phức tạp
  - Hệ thống cần mở rộng phần đọc và ghi riêng biệt
  - Ứng dụng yêu cầu roll back
  - Phần đọc cần truy vấn nhanh, linh hoạt, trong khi phần ghi cần bảo vệ logic chặt chẽ

1.3.6. Các lưu ý khi áp dụng CQRS

- Sử dụng các cơ chế như eventual consistency và message queue để đồng bộ dữ liệu giữa write và read model
- Nên áp dụng từng phần, dựa nên độ phức tạp của hệ thống

1.3.6. Áp dụng CQRS trong hệ thống

Bắt đầu: Mọi yêu cầu đều bắt đầu từ Client (React).

Gửi Yêu cầu: Client gửi yêu cầu đến API (Carter).

Pipeline: API tiếp nhận và chuyển tiếp yêu cầu đến MediatR Pipeline, đây là trung tâm điều phối yêu cầu.

Phân nhánh Xử lý: Tại đây, quy trình được chia thành hai nhánh xếp chồng lên nhau:

- Yêu cầu Command được định tuyến đến Command Handler
- Yêu cầu Query được định tuyến đến Query Giao tiếp Cơ sở dữ liệu: Cả Command Handler và Query Handler đều sử dụng DbContext (EF Core) để tương tác với cơ sở dữ liệu.

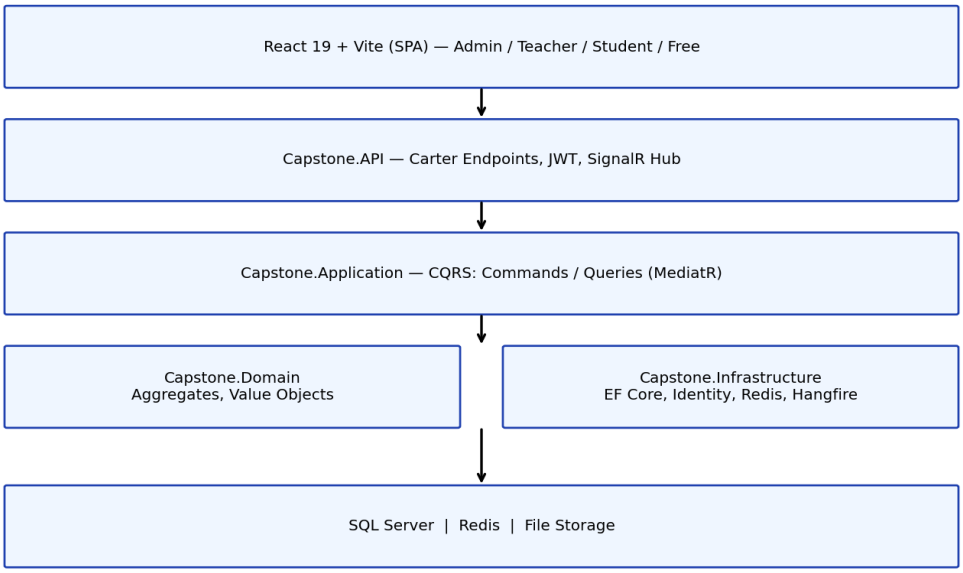
Hoàn tất: Cuối cùng, dữ liệu được ghi vào hoặc đọc ra từ SQL Server.

Bảng Mapping Command và Query trong hệ thống

Loại	**Ví dụ GV**	**Tên thực tế**	**Module**
Command	CreateExam	CreateExamCommand	ExamDomain
Command	SubmitAnswer	AnswerSingleChoiceQuestionCommand (+4 loại)	StudentDomain
Command	Sinh mã đề	GenerateExamVersionsCommand	ExamDomain
Query	GetExamDetail	GetExamByIdQuery	ExamDomain
Query	GetResult	GetDetailExamResultQuery	StudentDomain

Bảng 2: Mapping Command và Query trong hệ thống

1.4. Kiến trúc tổng thể của hệ thống



Hình 2: Kiến trúc tổng thể hệ thống

- Bước 1: (Giao diện): Người dùng tương tác trên Front-end (React 19 + Vite), gửi yêu cầu (Request) kèm mã xác thực JWT về phía máy chủ.
- Bước 2: (Tiếp nhận): Tầng Capstone.API tiếp nhận, xác thực người dùng qua JWT, điều hướng yêu cầu bằng Carter và thiết lập kết nối thời gian thực qua SignalR Hub (nếu cần).
- Bước 3:(Điều phối): Tầng Capstone.Application tiếp nhận yêu cầu từ API, áp dụng mô hình CQRS và sử dụng MediatR để phân loại chuẩn xác hành động thành Command (thay đổi dữ liệu) hoặc Query (lấy dữ liệu).
- Bước 4: (Xử lý & Liên kết Hạ tầng): Hàm xử lý (Handler) của lớp Application sẽ gọi các quy tắc nghiệp vụ cốt lõi tại Capstone.Domain (Aggregates, Value Objects). Đồng thời, nó phối hợp với Capstone.Infrastructure để dùng các công cụ như EF Core (truy vấn), Identity (tài khoản), Redis (bộ nhớ đệm), hoặc Hangfire (tác vụ ngầm).
- Bước 5: (Lưu trữ): Các lệnh từ tầng hạ tầng trực tiếp tác động xuống lớp dữ liệu vật lý gồm SQL Server, Redis, hoặc File Storage.
- Bước 6: (Phản hồi): Kết quả được truyền ngược lại từ dưới lên trên và trả về cho giao diện người dùng để hiển thị trạng thái mới nhất.

1.5. Các công nghệ sử dụng

1.5.1. Ngôn ngữ lập trình

C# (C-Sharp) là một ngôn ngữ lập trình hiện đại, hướng đối tượng do Microsoft phát triển. C# nổi bật với cú pháp rõ ràng, dễ bảo trì, hỗ trợ mạnh mẽ cho lập trình hướng đối tượng, và có khả năng mở rộng linh hoạt. Đây là một trong những ngôn ngữ chủ lực trong hệ sinh thái .NET, thường được dùng để phát triển ứng dụng web, desktop, mobile và các dịch vụ backend.

ASP.NET là một framework mạnh mẽ cũng do Microsoft phát triển, cho phép xây dựng các ứng dụng web động, dịch vụ RESTful API và các hệ thống backend hiệu suất cao. ASP.NET hỗ trợ nhiều mô hình lập trình như MVC (Model–View–Controller), Web API, Razor Pages và đặc biệt là ASP.NET Core – phiên bản đa nền tảng, nhẹ và tối ưu hơn.

JavaScript [10]: là một ngôn ngữ lập trình phổ biến được sử dụng chủ yếu trong lĩnh vực phát triển web. Được thiết kế để thêm tính năng tương tác động vào các trang web, giúp người dùng tương tác với các yếu tố trên trang, như nút bấm, hộp thoại, và các hiệu ứng động. JavaScript là một ngôn ngữ lập trình phổ biến được sử dụng chủ yếu trong lĩnh vực phát triển web.

1.5.2. Cơ sở dữ liệu

SQL Server là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) do Microsoft phát triển, được sử dụng trong dự án để lưu trữ và quản lý dữ liệu hiệu quả. Với khả năng xử lý mạnh mẽ, hỗ trợ ngôn ngữ T-SQL, tính bảo mật cao và tích hợp tốt với nền tảng .NET, SQL Server đáp ứng tốt các yêu cầu về hiệu suất, độ ổn định và khả năng mở rộng của hệ thống.

1.6. Kết luận

Trong chương cơ sở lí thuyết, những lí thuyết về thiết kế hệ thống (DDD, CQRS), công nghệ sử dụng (C#, ASP.NET, Javascript) đã được giới thiệu. Dựa trên những cơ sở lý thuyết này thì có thể tiếp tục triển khai các chức năng mà website đã đề ra.

CHƯƠNG II: PHÂN TÍCH - THIẾT KẾ HỆ THỐNG

2. Phân tích yêu cầu

2.1. Tổng quan

Hệ thống hỗ trợ thi cử và chống gian lận trực tuyến là một giải pháp phần mềm được xây dựng nhằm đáp ứng nhu cầu tổ chức các kỳ thi trực tuyến tại Trường Đại học Bách khoa – Đại học Đà Nẵng, giúp giảng viên dễ dàng quản lý, tạo đề thi, tổ chức thi và chấm điểm, đồng thời hỗ trợ sinh viên trong việc làm bài thi một cách tiện lợi và công bằng. Hệ thống phải đảm bảo tính linh hoạt trong việc thiết kế đề thi với đa dạng dạng câu hỏi và cấp độ khác nhau, đồng thời tích hợp các biện pháp kỹ thuật nhằm hạn chế các hành vi gian lận trong quá trình thi. Ngoài ra, hệ thống cần có giao diện thân thiện, dễ sử dụng, đảm bảo hiệu suất và độ ổn định khi phục vụ đồng thời nhiều người dùng.

Phạm vi của hệ thống bao gồm các chức năng chính như: quản lý tài khoản người dùng (giảng viên, sinh viên), tạo và quản lý đề thi, tổ chức các kỳ thi trực tuyến, giám sát và phát hiện gian lận, chấm điểm tự động đối với các dạng câu hỏi trắc nghiệm, lưu trữ kết quả thi và báo cáo thống kê phục vụ công tác đánh giá. Hệ thống hướng tới mục tiêu tạo ra một môi trường thi cử trực tuyến hiện đại, tin cậy và minh bạch, góp phần nâng cao chất lượng đào tạo và công tác đánh giá tại nhà trường.

2.2. Yêu cầu các chức năng

- Hệ thống cần đáp ứng các yêu cầu chức năng chính sau đây để phục vụ tốt cho giảng viên và sinh viên trong quá trình thi trực tuyến:
    - Quản lý người dùng
        - Đăng nhập, phân quyền cho giảng viên và sinh viên.
        - Quản lý thông tin cá nhân, thay đổi mật khẩu, phục hồi tài khoản.
    - Quản lý đề thi
        - Giảng viên có thể tạo mới, chỉnh sửa, xóa đề thi.

  - Hỗ trợ tạo đề thi với nhiều loại câu hỏi khác nhau: trắc nghiệm, tự luận, ghép nối, kéo thả,...
  - Định nghĩa các cấp độ câu hỏi (nhận biết, thông hiểu, vận dụng,...).
  - Tạo ngân hàng câu hỏi dùng chung cho nhiều đề thi.
- Tổ chức thi trực tuyến
  - Giảng viên tạo các đợt thi, phân bổ đề thi cho các lớp hoặc nhóm sinh viên.
  - Thiết lập thời gian làm bài, số lần làm bài, điều kiện thi (ví dụ: cho phép xem lại đề, xem đáp án...).
  - Sinh viên truy cập và làm bài thi trực tuyến qua giao diện thân thiện, bảo đảm tính bảo mật.
- Giám sát và chống gian lân
  - Theo dõi hành vi thi của sinh viên: cảnh báo khi rời khỏi màn hình, ghi nhận thao tác chuột, bàn phím.
  - Chặn các thao tác sao chép, dán trong lúc làm bài.
  - Phát hiện các hành vi bất thường hoặc gian lận và gửi cảnh báo cho giảng viên.
- Chấm điểm và phản hồi kết quả
  - Tự động chấm điểm các câu hỏi trắc nghiệm, tính điểm tổng hợp.
  - Giảng viên có thể chấm điểm thủ công cho các câu hỏi tự luận.
  - Hiển thị kết quả thi, thống kê điểm số cho giảng viên và sinh viên.
- Quản lý báo cáo và thống kê
  - Tổng hợp kết quả thi theo lớp, môn học, đợt thi.
  - Báo cáo chi tiết về tình hình làm bài, các vi phạm gian lận (nếu có).
  - Xuất dữ liệu kết quả thi phục vụ công tác lưu trữ và phân tích.

2.3. Biểu đồ ca sử dụng

2.3.1. Tác nhân

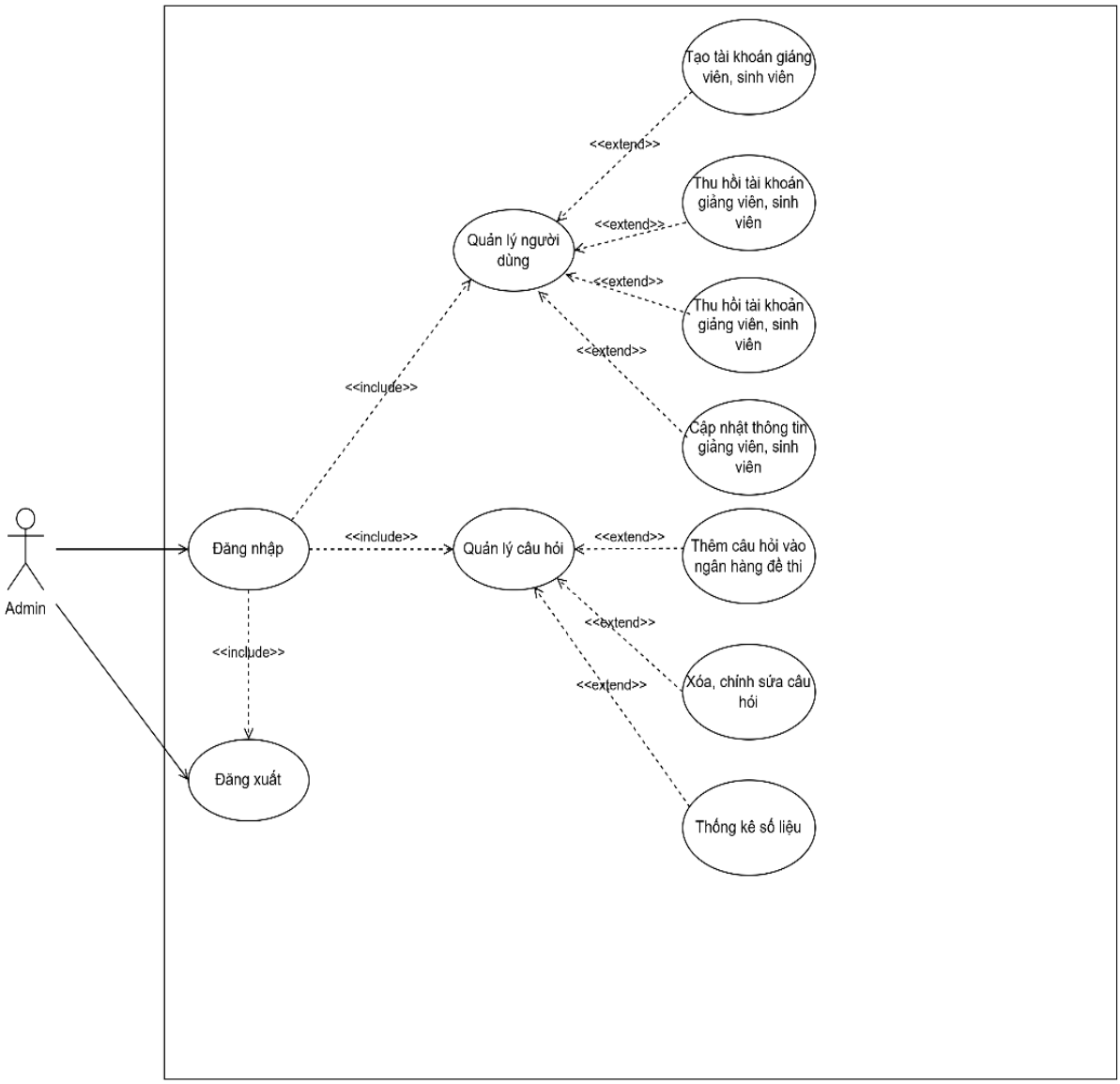
- Bảng dưới đây mô tả tất cả tác nhân tham gia vào hệ thống

Tác nhân	Mô tả
Giảng viên	- Tạo và quản lý đề thi, tổ chức các kỳ thi - Giám sát quá trình thi và xử lý các vi phạm gian lận - Chấm điểm và theo dõi kết quả thi
Sinh viên	- Tham gia làm bài thi trực tuyến - Xem kết quả thi và các phản hồi từ giảng viên
Quản trị viên	- Quản lý hệ thống chung, phân quyền người dùng, bảo trì dữ liệu.

Bảng 3:Bảng tác nhân

2.3.2. Sơ đồ ca sử dụng

- Sơ đồ ca sử dụng của quản trị viên: miêu tả các chức năng của quản trị viên có thể sử dụng trên website

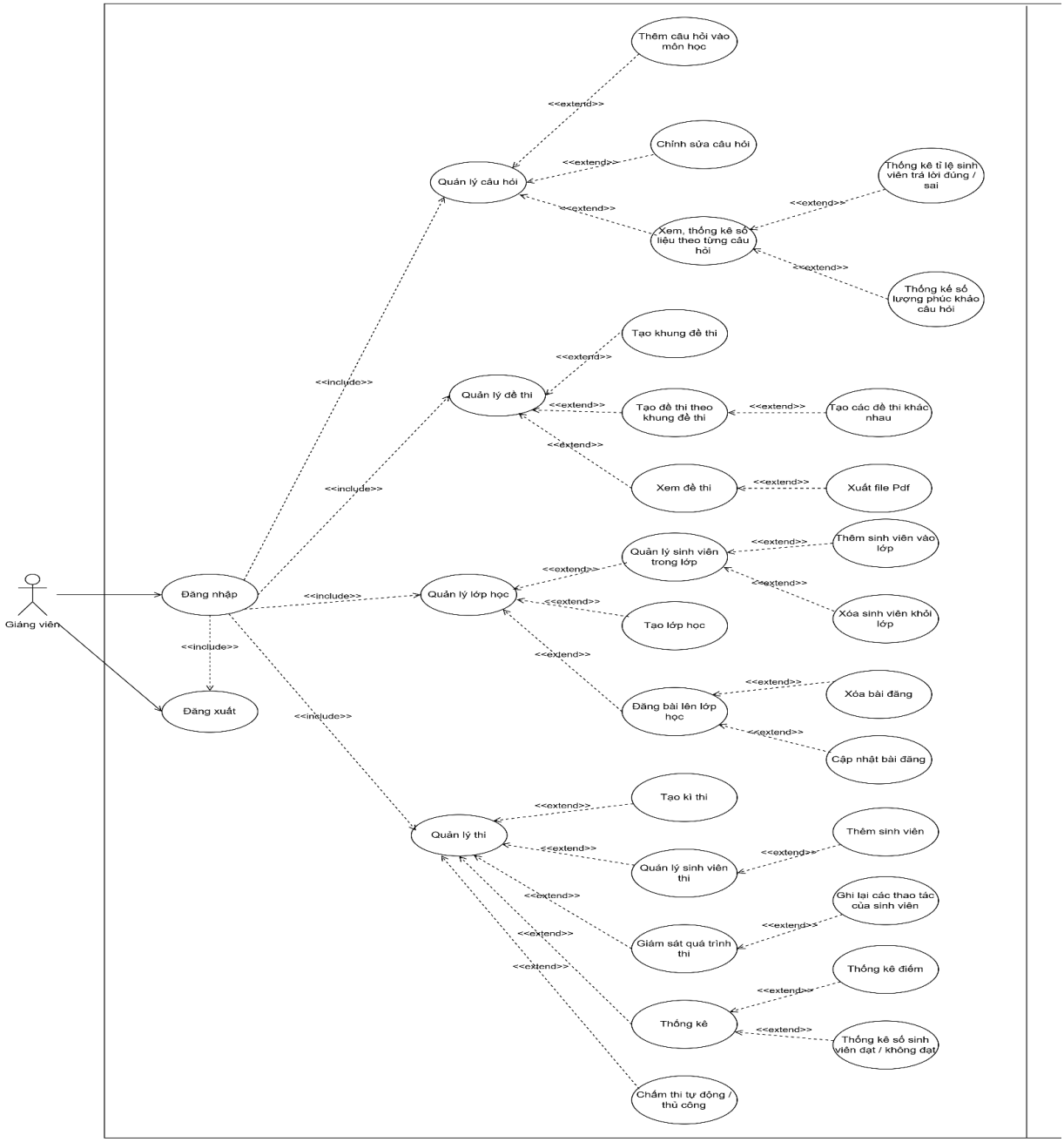


Hình 3: Sơ đồ usecase của quản trị viên

Luồng xử lý: Quản trị viên thao tác qua các use case: quản lý giáo viên/sinh viên (thêm, sửa, xem, import CSV), kích hoạt/vô hiệu tài khoản, reset mật khẩu → gọi API /admin/* kèm JWT role ADMIN.

Vai trò: React (giao diện Admin), Carter Endpoint (phân quyền), Command/Query Handler, UserManager/Identity, SQL Server.

- Sơ đồ ca sử dụng của giảng viên : miêu tả các chức năng của giảng viên có thể sử dụng trên website

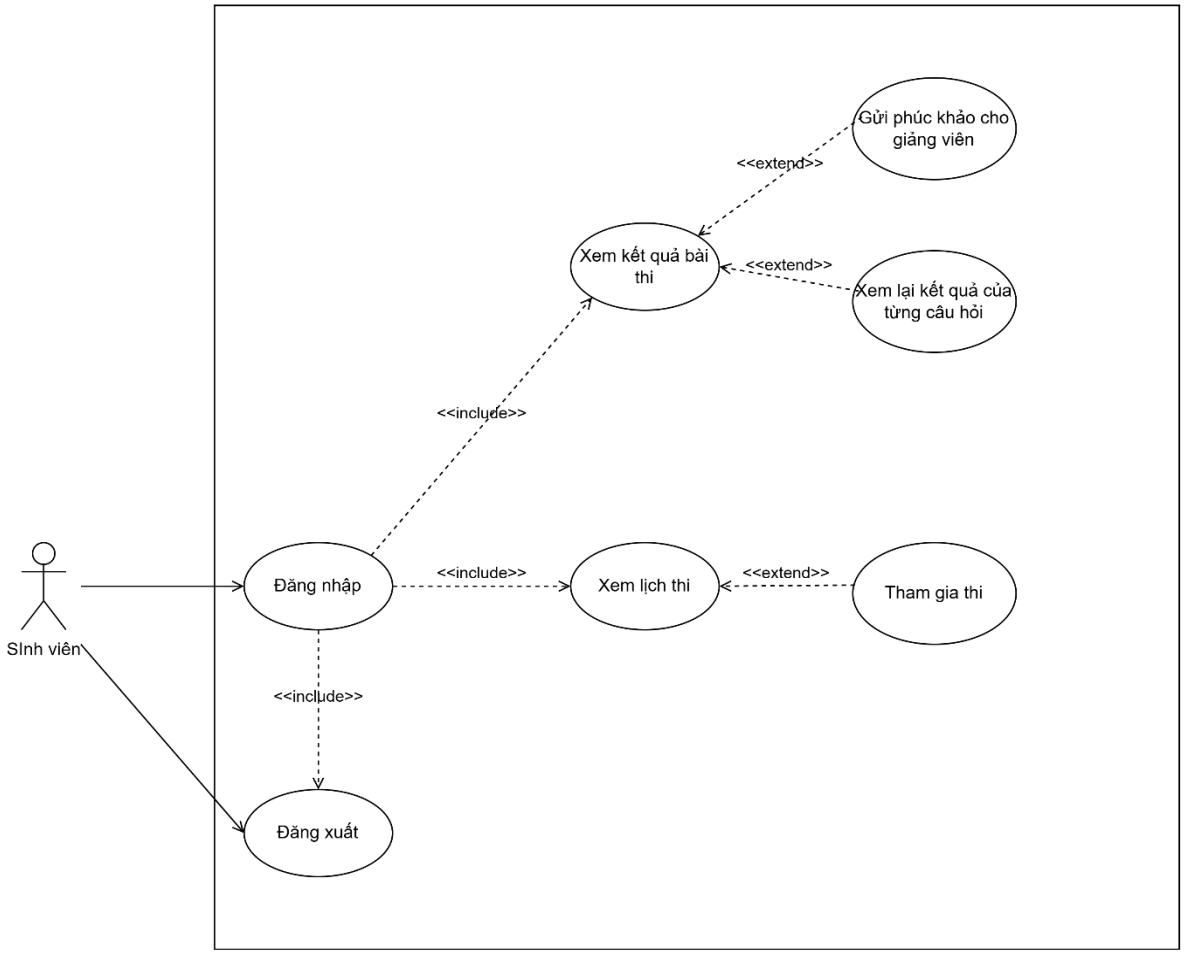


Hình 4: Sơ đồ usecase của giảng viên

Luồng xử lý: Giáo viên thực hiện chuỗi nghiệp vụ: quản lý học phần → ngân hàng câu hỏi → khung đề → sinh mã đề → tạo ca thi → giám sát/chấm điểm. Mỗi use case map tới API /teacher/* hoặc /questions/* với role TEACHER.

- Vai trò: React (Teacher UI), MediatR (CQRS), Domain aggregates (Question, Exam, ExamSession), EF Core (persist).Sơ đồ ca sử dụng của sinh viên : miêu tả các chức năng của sinh viên có thể sử dụng trên website

- Sơ đồ ca sử dụng của sinh viên : miêu tả các chức năng của sinh viên có thể sử dụng trên website

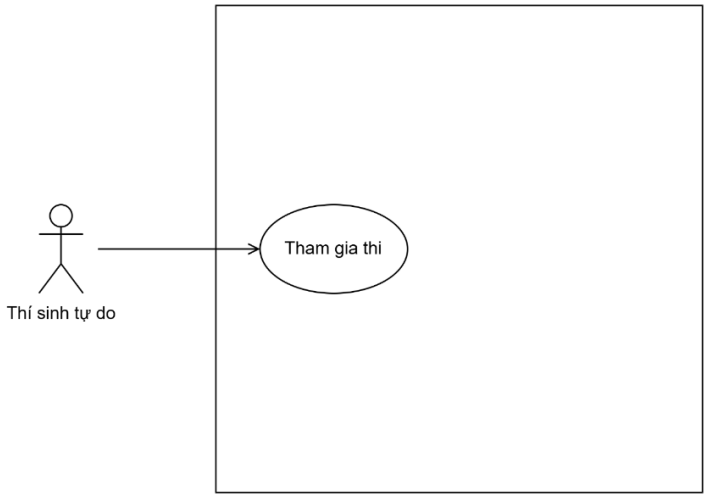


Hình 5: Sơ đồ usecase của sinh viên

Luồng xử lý: Sinh viên đăng nhập → xem lịch/ca thi → vào thi (join) → trả lời từng loại câu → nộp bài → xem kết quả/phúc khảo. Request qua /student/* với JWT role STUDENT.

Vai trò: React (Student UI), JoinExamSessionHandler, Answer*QuestionHandlers, Participant (trạng thái thi cá nhân).

- Sơ đồ ca sử dụng của thí sinh tự do: miêu tả các chức năng của sinh viên có thể sử dụng trên website

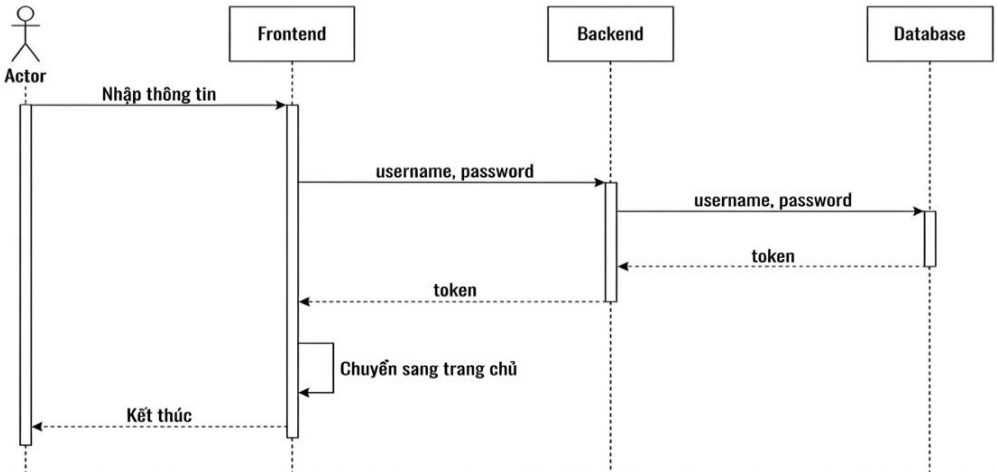


Hình 6: Sơ đồ usecase của thí sinh tự do

Luồng xử lý: Thí sinh tự do xác thực mã ca → nhập họ tên/email → vào thi → làm bài (anti-cheat client) → nộp bài qua /free/*; xác thực bằng participantId, không cần tài khoản.
Vai trò: ExamPageFree (giám sát fullscreen), FreeParticipant Handlers,

NotificationHub (báo vi phạm cho giáo viên).Sơ đồ tuần tự

- Chức năng đăng nhập

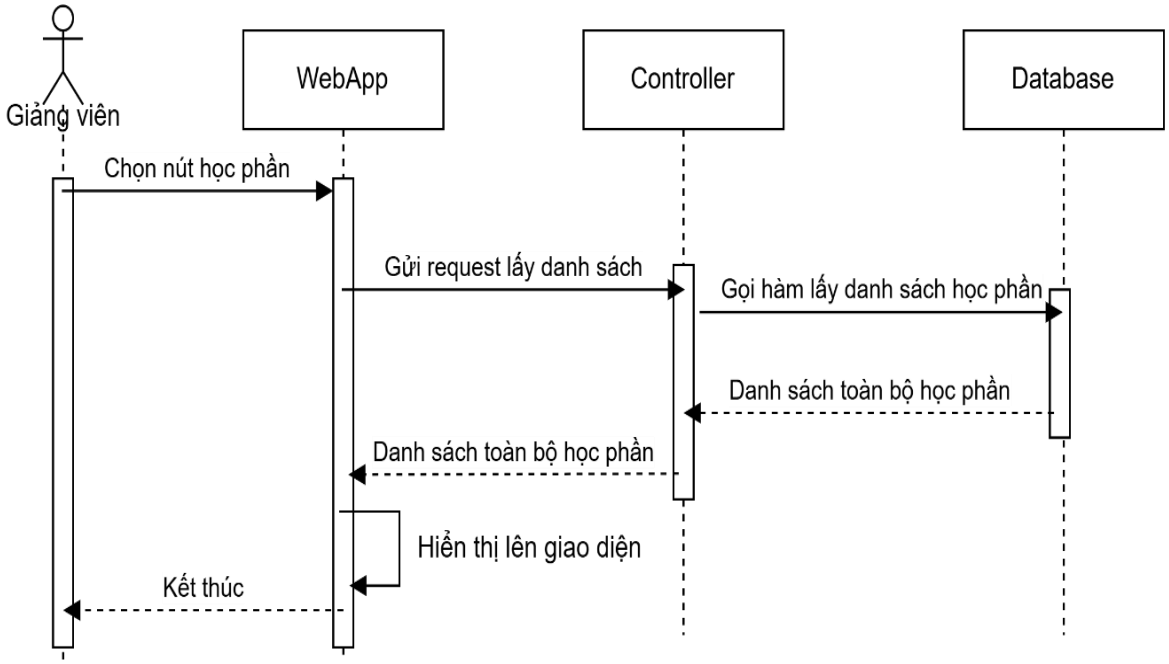


Hình 7: Sơ đồ tuần tự các chức năng đăng nhập

Luồng xử lý: Người dùng nhập tài khoản trên React → POST /identity/login → LoginHandler gọi IdentityServer (/connect/token) xác thực → trả JWT và role → lưu localStorage.
Vai trò: React (UI), Endpoint (cổng API), LoginHandler (điều phối), IdentityServer

(phát token), SQL Server (tra cứu user).

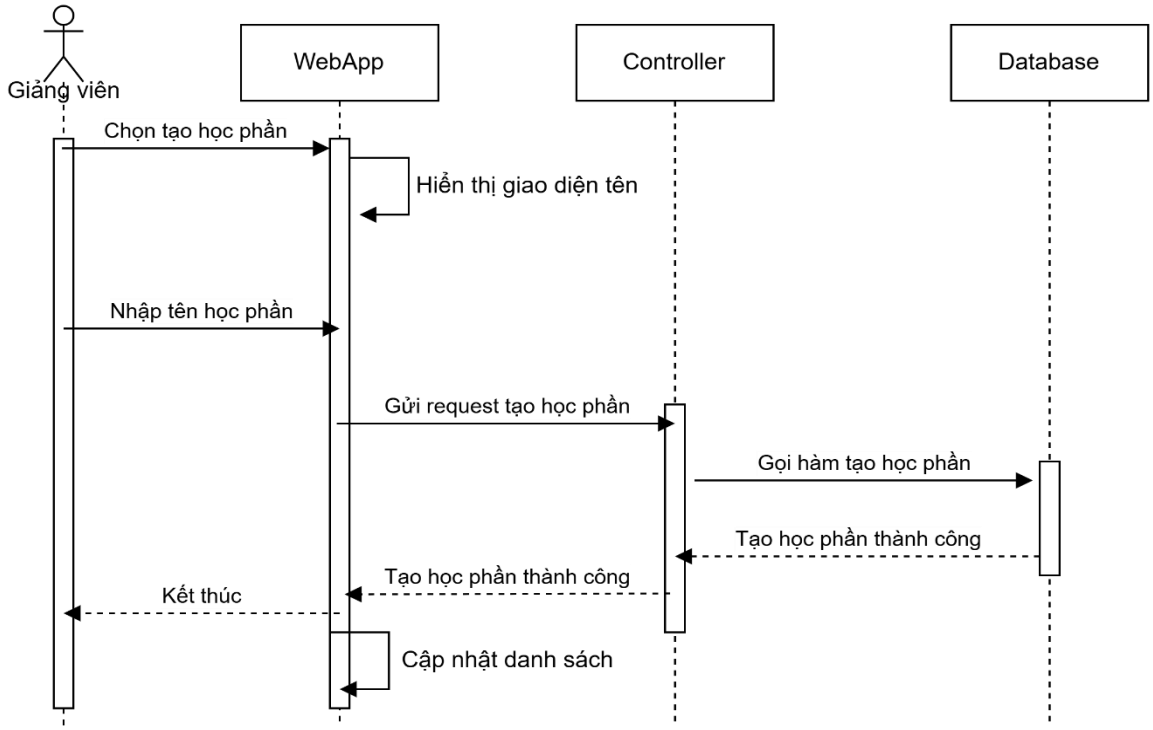
- Chức năng xem học phần



Hình 8: Sơ đồ tuần tự của chức năng xem học phần

Luồng xử lý: Giáo viên mở trang môn học → GET /teacher/subjects (JWT) → GetTeacherSubjectsHandler truy vấn TeacherSubjects theo OwnerId, phân trang → trả danh sách JSON.
Vai trò: React (hiển thị), Endpoint (auth TEACHER), Query Handler (đọc DB), EF Core (LINQ).

- Chức năng tạo học phần

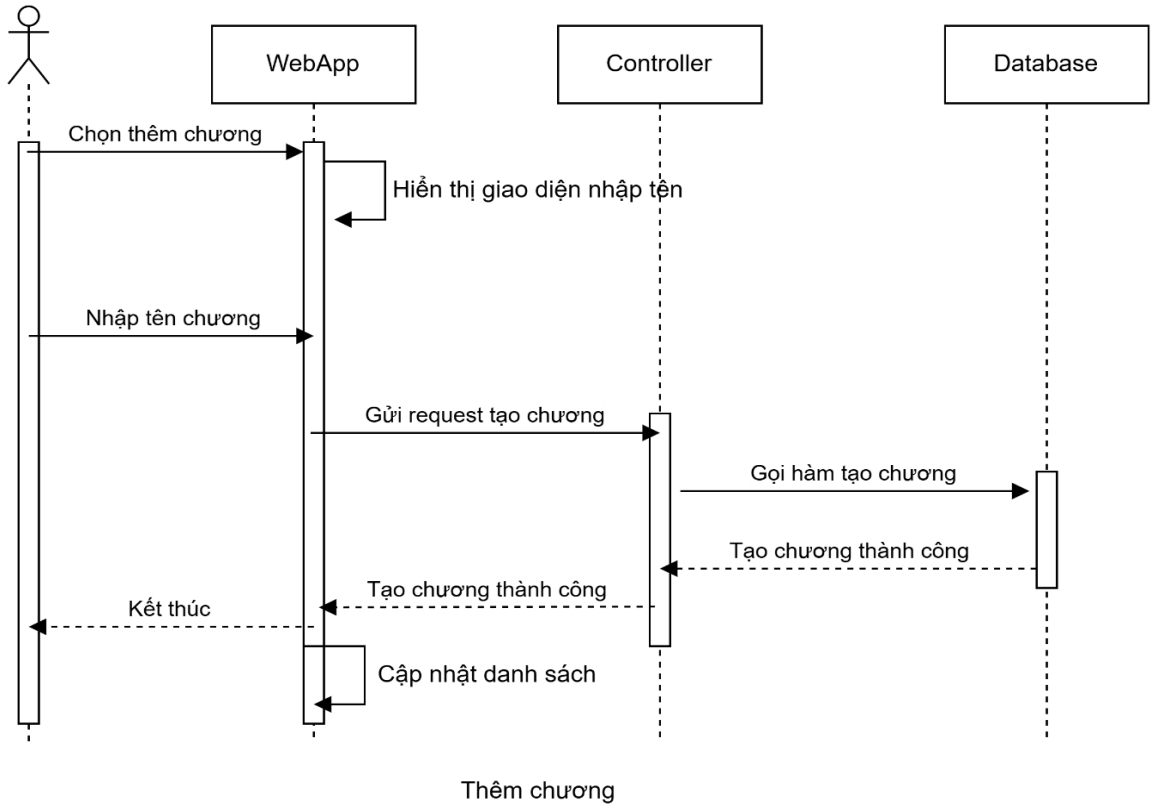


Hình 9: Sơ đồ tuần tự chức năng tạo học phần

Luồng xử lý: Giáo viên nhập tên môn → POST /teacher/subjects → AddTeacherSubjectHandler tạo TeacherSubject gắn OwnerId → lưu DB → trả subjectId.

Vai trò: React (form), Command Handler (ghi dữ liệu), TeacherSubject (aggregate), DbContext (persist).

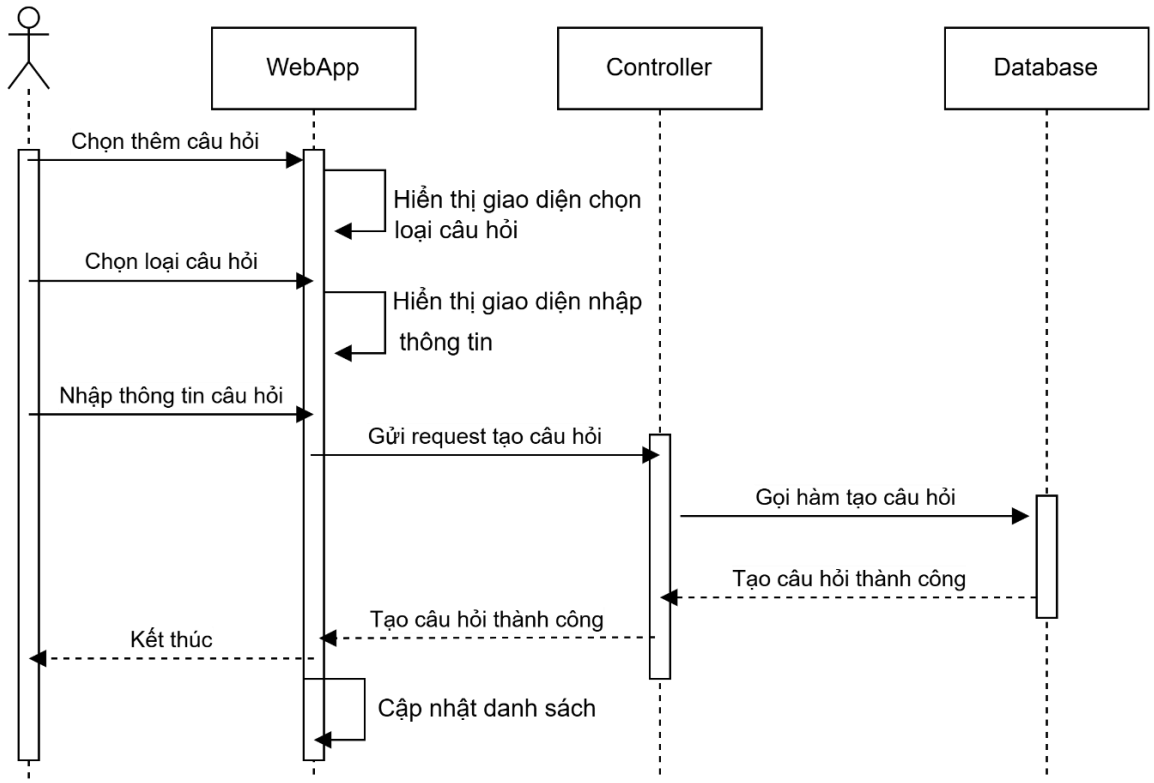
- Chức năng thêm chương



Hình 10: Sơ đồ tuần tự chức năng thêm chương

Luồng xử lý: Giáo viên nhập tiêu đề chương → POST /subjects/{id}/chapters → AddChapterHandler kiểm tra quyền, thêm Chapter vào Subject → lưu DB.
Vai trò: Subject (aggregate cha), AddChapterHandler (nghiệp vụ), Endpoint (policy ADMIN_OR_TEACHER).

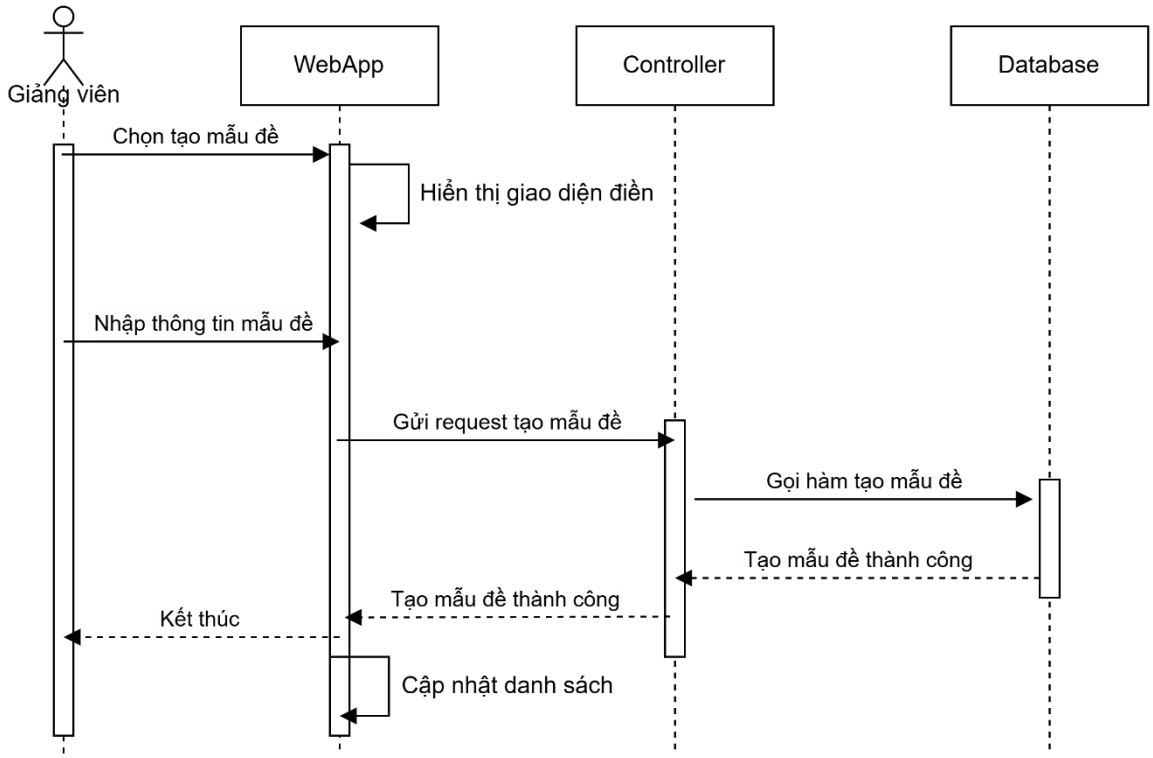
- Chức năng thêm câu hỏi



Hình 11: Sơ đồ tuần tự chức năng thêm câu hỏi

Luồng xử lý: Giáo viên chọn loại câu → POST /questions/{loại} → Add*QuestionHandler validate và tạo Question (TPH) theo chương → lưu ngân hàng câu hỏi.
Vai trò: Question hierarchy (Domain), Handler theo loại câu, EF Core (một bảng, discriminator Type).

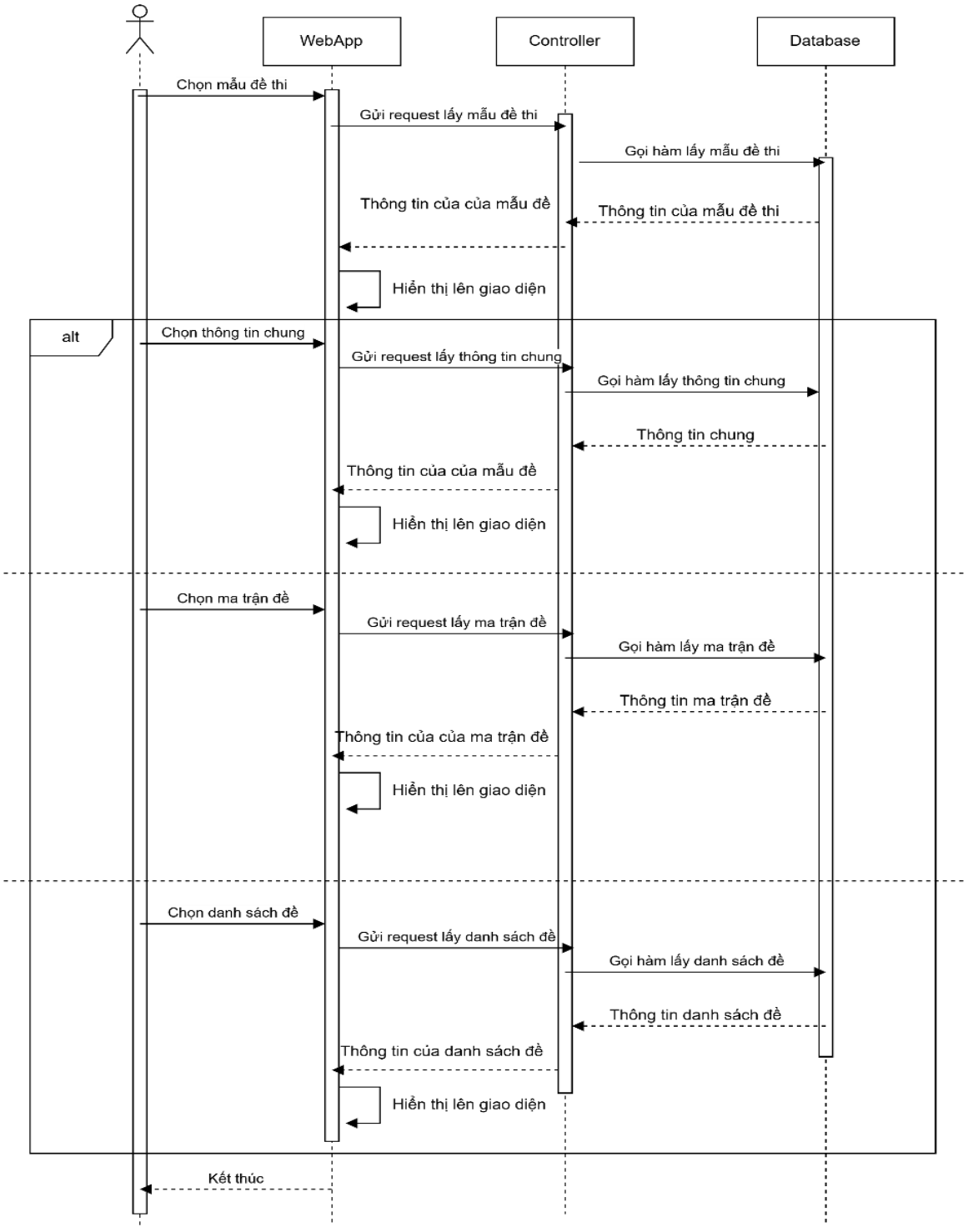
- Chức năng xem chi tiết câu hỏi



Hình 12: Sơ đồ tuần tự chức năng xem chi tiết câu hỏi

Luồng xử lý: Giáo viên chọn câu hỏi → GET /questions/{id} (JWT) → GetQuestionByIdQuery → Handler load Question aggregate theo loại (TPH) → map DTO → hiển thị nội dung và đáp án trên React.
Vai trò: React (chi tiết câu hỏi), Endpoint (auth ADMIN_OR_TEACHER), Query Handler (read-only), QuestionExtension (entity → DTO), EF Core (truy vấn ngân hàng câu).

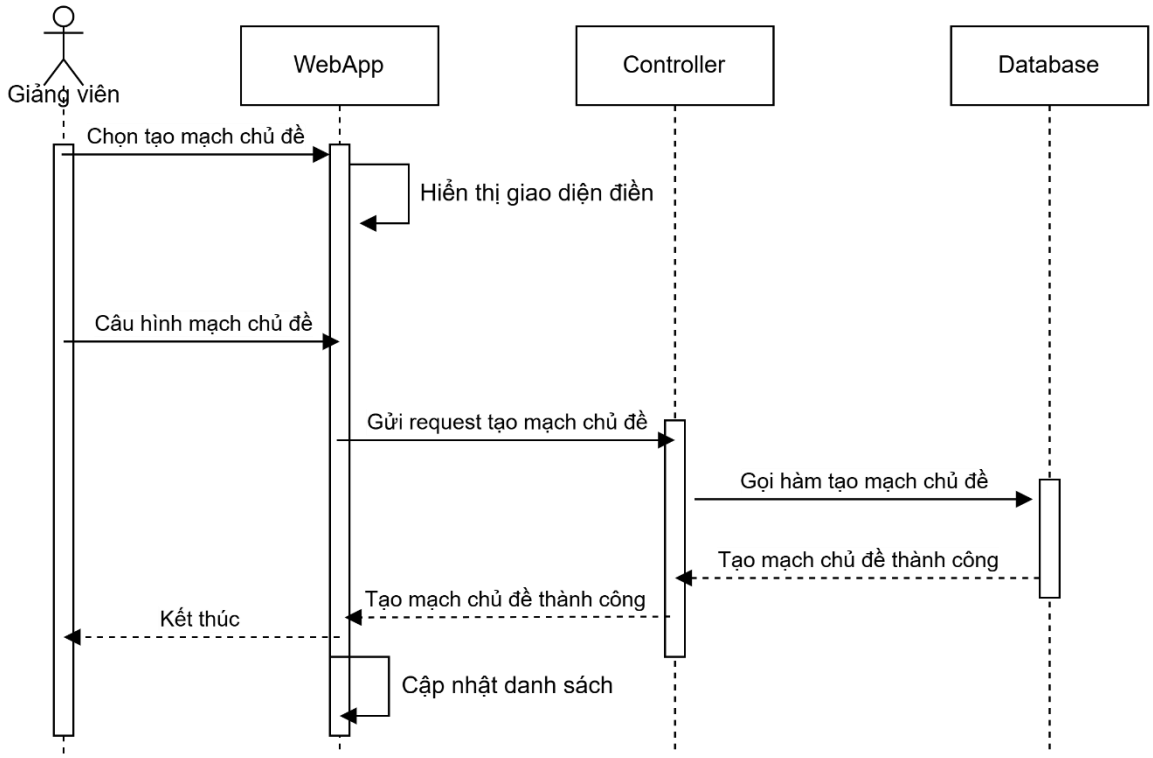
- Chức năng tạo mẫu đề



Hình 13: Sơ đồ tuần tự chức năng tạo mẫu đề

Luồng xử lý: Giáo viên chọn mẫu đề → GET /teacher/exam-template/{id} → GetExamTemplateByIdHandler load sections + cấu hình độ khó/loại câu → map DTO → hiển thị ma trận đề.
Vai trò: Query Handler (read-only), ExamTemplateSection (từng phần ma trận), React (bảng/cây cấu trúc).

- Chức năng xem chi tiết mẫu đề

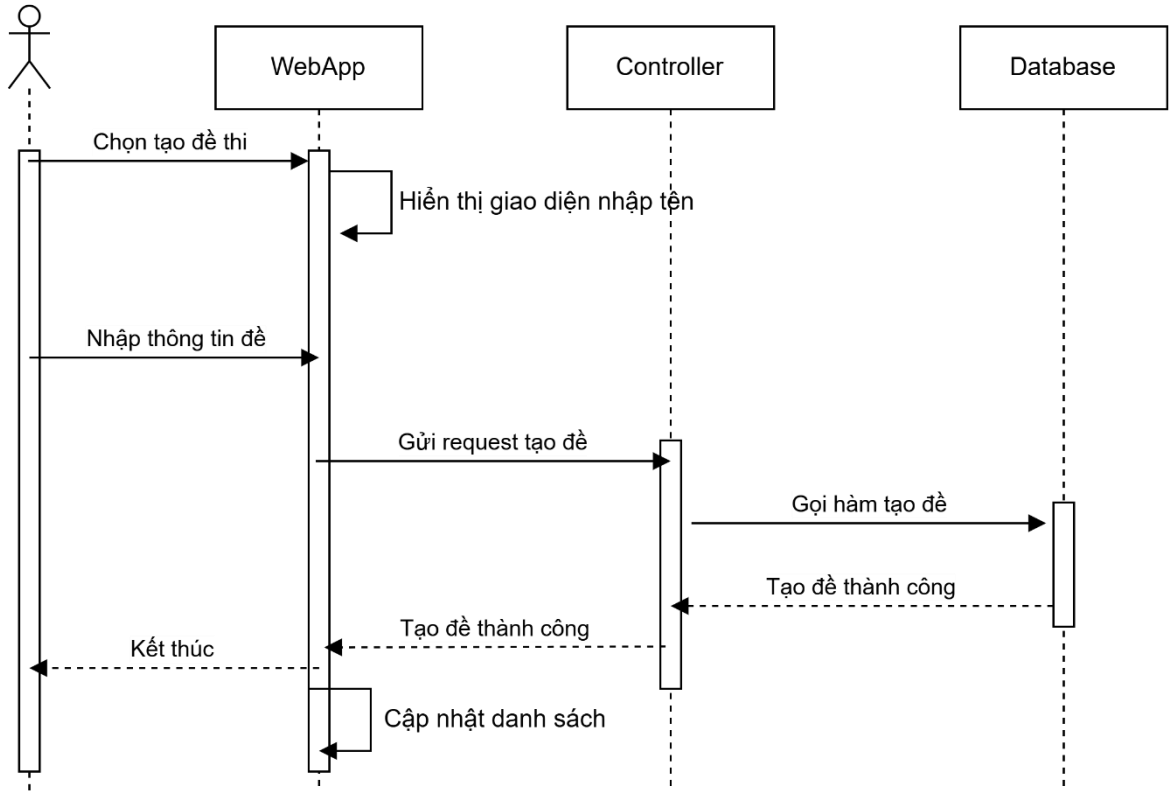


Hình 14: Sơ đồ tuần tự chức năng xem chi tiết mẫu đề

Luồng xử lý: Giáo viên cấu hình chương + độ khó + loại câu → POST /exam-template/{id}/exam-template-section → AddExamTemplateSectionHandler thêm section vào ExamTemplate → lưu DB.

- Vai trò: ExamTemplateSection (block ma trận theo chương), Handler (validate cấu hình), Chapter (liên kết ngân hàng câu).Chức năng thêm mạch chủ đề

- Chức năng tạo mạch chủ đề

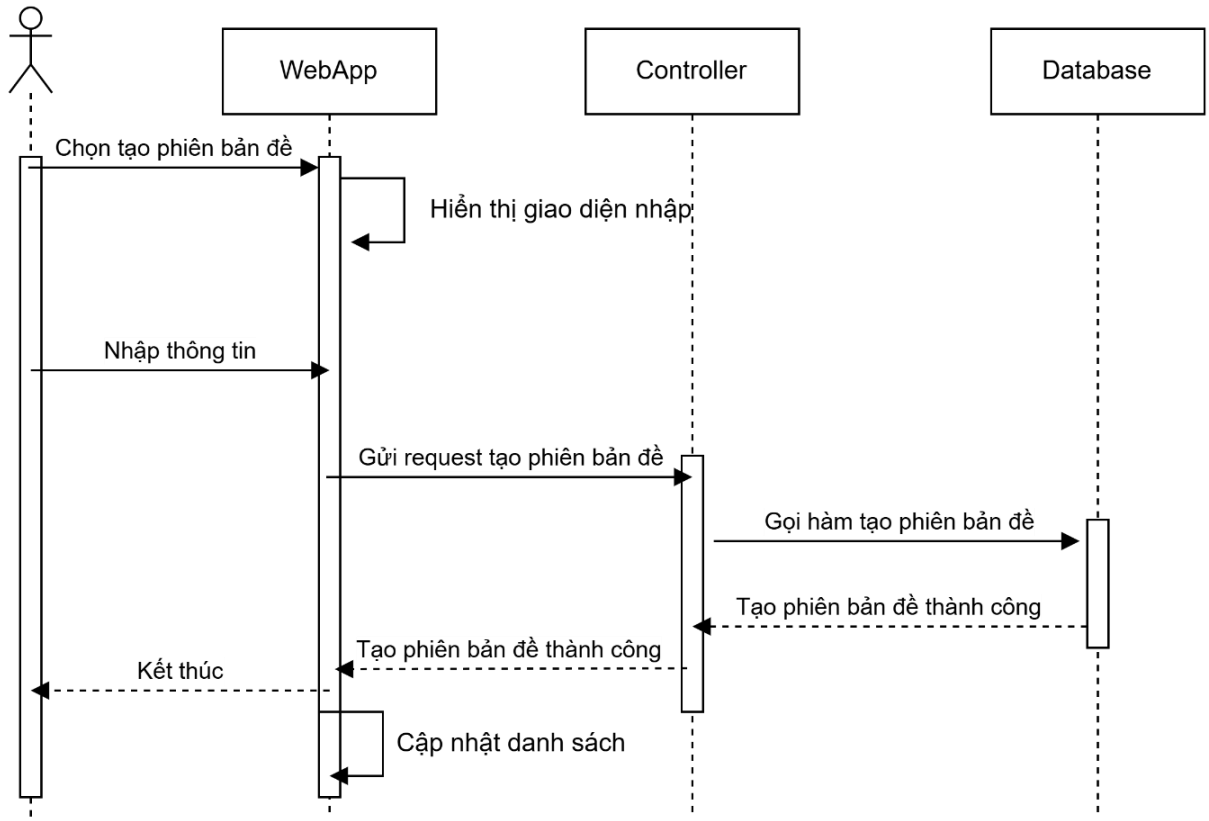


Hình 15: Sơ đồ tuần tự chức năng tạo mạch chủ đề

Luồng xử lý: Giáo viên chọn mẫu đề, nhập thời lượng → POST /teacher/exams → CreateExamHandler tạo Exam liên kết ExamTemplate → lưu DB → trả examId.

- Vai trò: Exam (container đề), ExamTemplate (cấu trúc sinh đề), Handler (khởi tạo metadata đề).Chức năng tạo đề thi

- Chức năng tạo đề thi

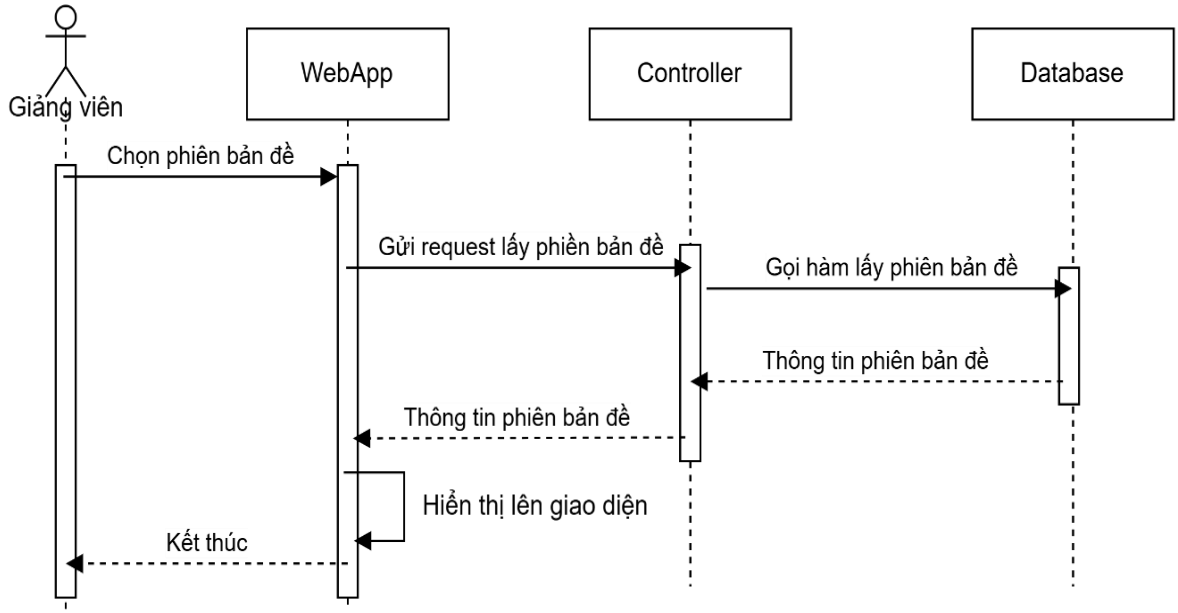


Hình 16: Sơ đồ tuần tự chức năng tạo đề thi

Luồng xử lý: Giáo viên nhập số mã đề → POST /teacher/exams/{id}/generate → GenerateExamVersionsHandler đọc ma trận, chọn câu weighted random → tạo ExamVersion + danh sách câu → lưu DB.

- Vai trò: GenerateExamVersionsHandler (thuật toán sinh đề), ExamVersion (một mã đề), Question pool (nguồn câu).Chức năng tạo phiên bản đề thi

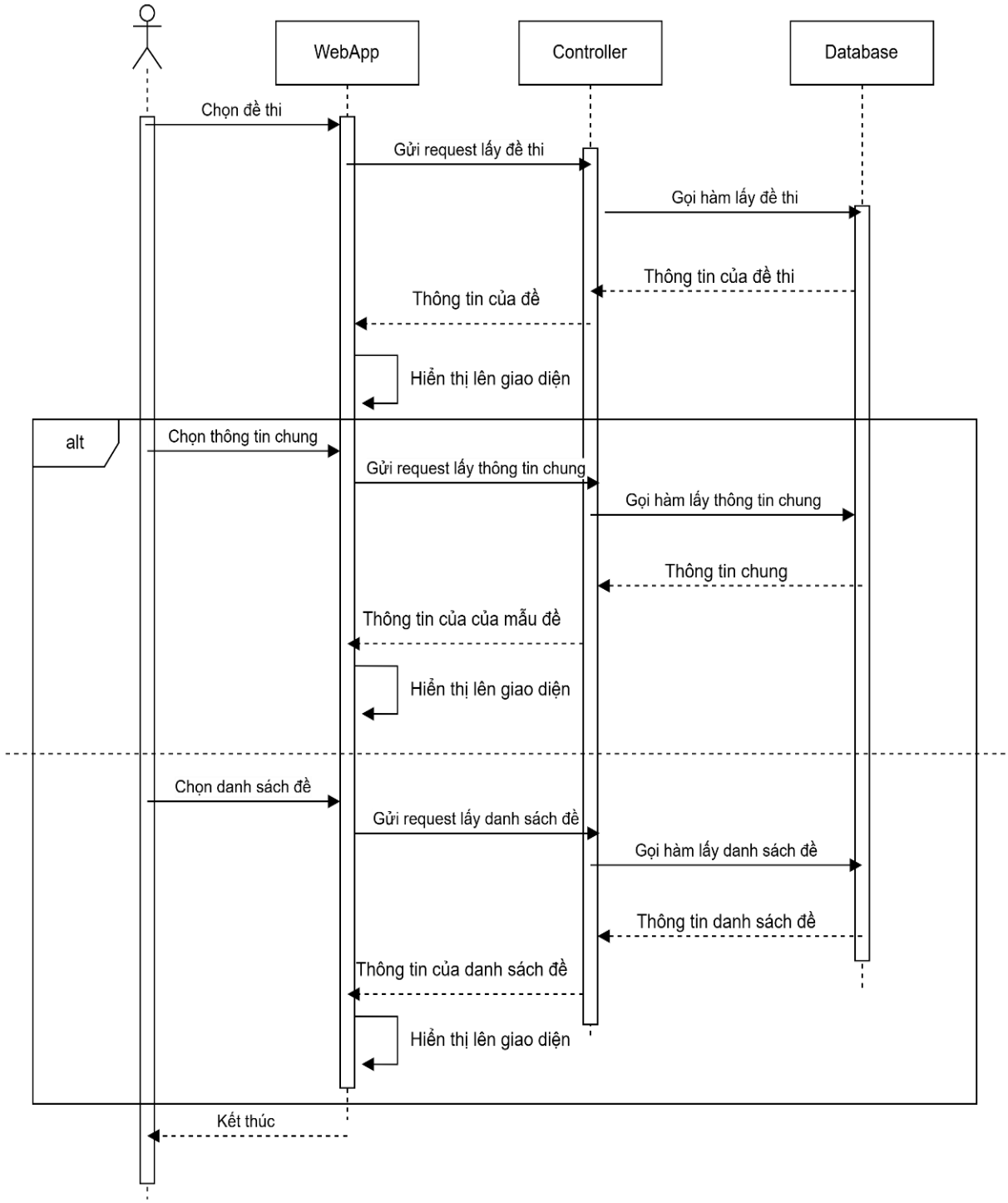
- Chức năng tạo phiên bản đề



Hình 17: Sơ đồ tuần tự chức năng tạo phiên bản đề

Luồng xử lý: Giáo viên chọn mã đề → GET /teacher/exam-versions/{id} → Handler load ExamVersion + câu hỏi (thứ tự, điểm) → DTO → preview trên UI.
Vai trò: Query Handler (join read-only), QuestionExtension (map entity → DTO), React (xem/in đề).

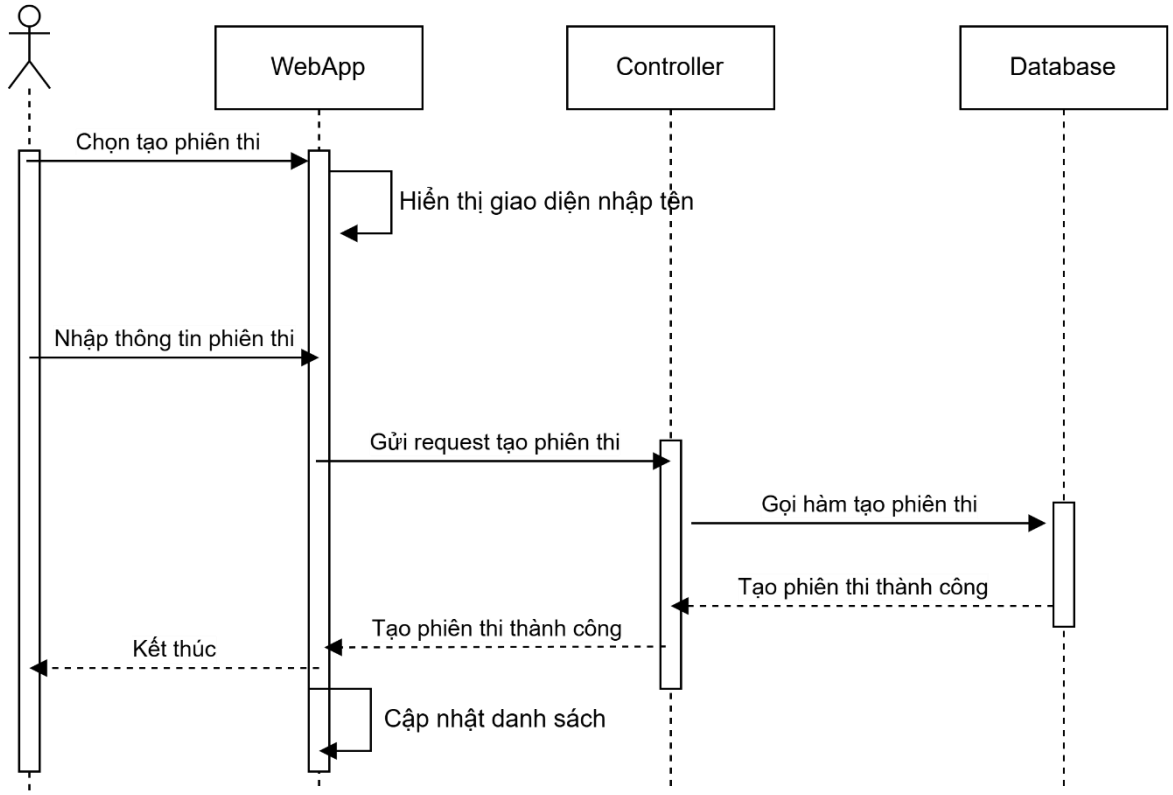
- Chức năng xem phiên bản đề



Hình 18: Sơ đồ tuần tự chức năng xem phiên bản đề

Luồng xử lý: Giáo viên chọn gói đề → GET /teacher/exams/{id} → GetExamByIdHandler load metadata Exam + template + danh sách mã đề → hiển thị trang quản lý.
Vai trò: GetExamByIdHandler (read model), Exam/ExamTemplate (quan hệ 1-N với ExamVersion)

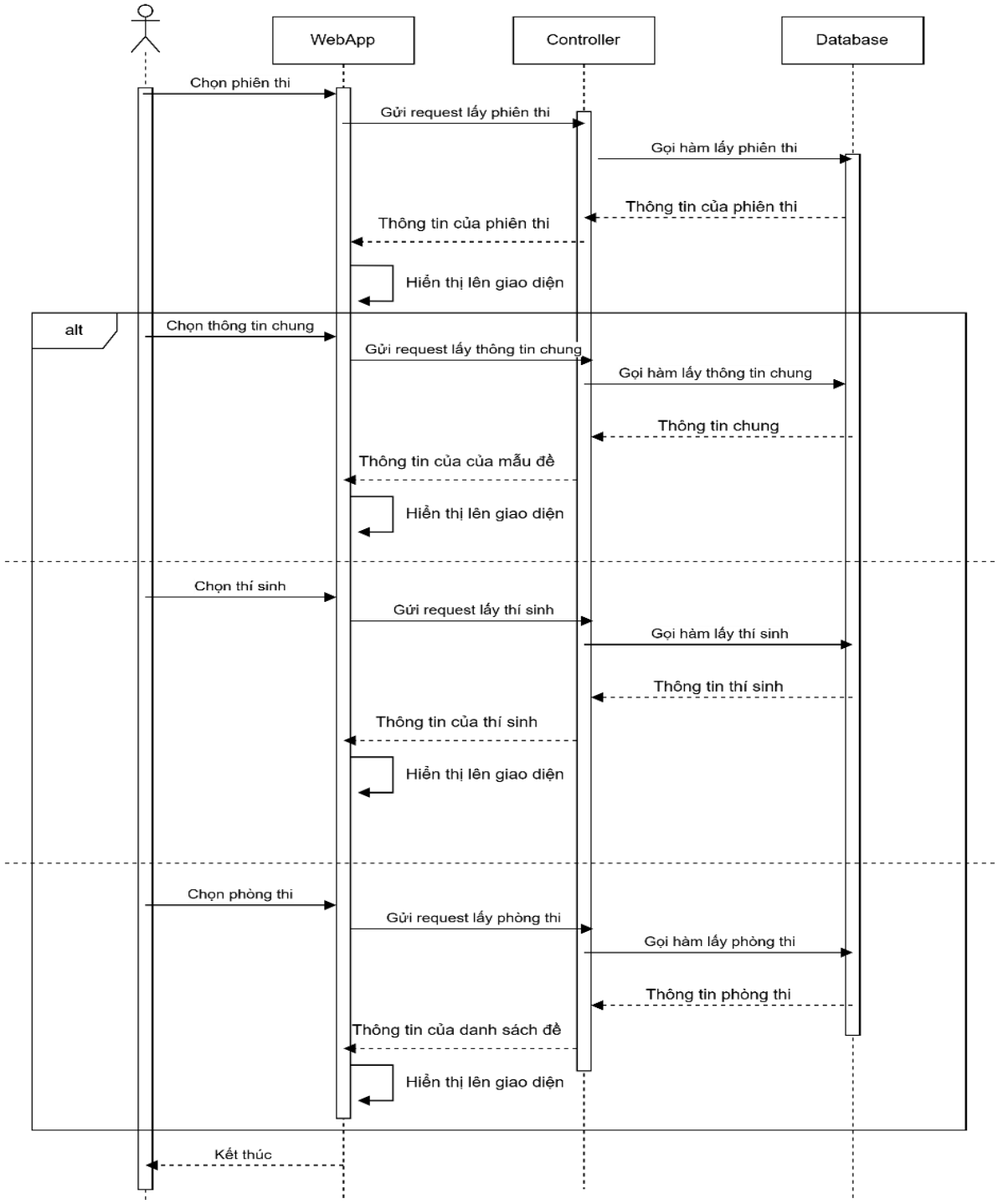
- Chức năng xem chi tiết đề



Hình 19: Sơ đồ tuần tự chức năng xem chi tiết đề

Luồng xử lý: Giáo viên cấu hình ca thi (thời gian, đề, lớp) → POST /teacher/exam-sessions → CreateExamSessionHandler tạo ExamSession, sinh Participant từ danh sách lớp → lưu DB.
Vai trò: ExamSession (ca thi), Participant (thí sinh), Class (nguồn sinh viên đăng ký).

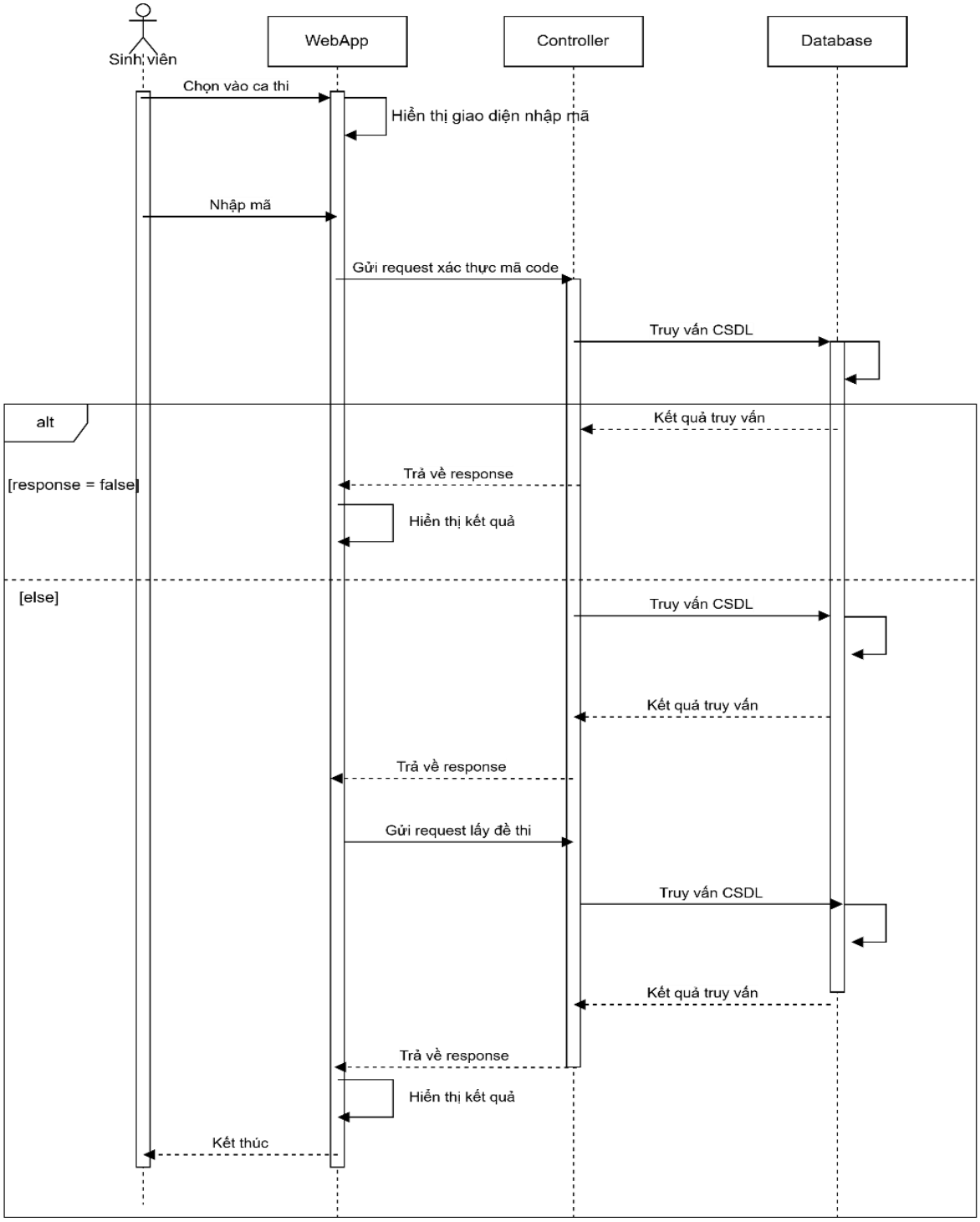
- Chức năng tạo phiên thi



Hình 20: Sơ đồ tuần tự chức năng tạo phiên thi

Luồng xử lý: Giáo viên mở ca thi → GET /teacher/exam-sessions/{id} + /participants, /doing, /actions → Handler trả trạng thái thí sinh; SignalR push vi phạm realtime.
Vai trò: Query Handlers (REST), NotificationHub (SignalR), React (tab giám sát/chấm điểm).

- Chức năng xem chi tiết phiên thi



Hình 21: Sơ đồ tuần tự chức năng xem chi tiết phiên thi

Luồng xử lý: SV/thí sinh bấm vào thi → POST .../join → Handler kiểm tra khung giờ + quyền → random ExamVersion → Participant.StartExam() → trả đề; Hangfire hẹn auto-submit.
Vai trò: JoinExamSessionHandler (SV đăng ký), JoinExamSessionFreeHandler (thi tự do), Participant (trạng thái cá nhân), Hangfire (timeout server-side).

Bảng mô tả Use Case theo quản trị viên

UC	Tên	Input	Output	Command/Query
UC-A01	Đăng nhập	username, password	JWT	LoginCommand
UC-A02	DS giảng viên	filter	list	GetTeachersQuery
UC-A03	Chi tiết GV	teacherId	detail	GetTeacherByIdQuery
UC-A04	Thêm GV	form	teacher	AddTeacherCommand
UC-A05	Import GV CSV	file	result	ImportCsvAddTeachersCommand
UC-A06	DS sinh viên	filter	list	GetStudentsQuery
UC-A07	Chi tiết SV	studentId	detail	GetStudentByIdQuery
UC-A08	Thêm SV	form	student	AddStudentCommand
UC-A09	Import SV CSV	file	result	ImportCsvAddStudentsCommand
UC-A10	Kích hoạt TK	userId	active	ActiveUserCommand
UC-A11	Vô hiệu TK	userId	inactive	DeactiveUserCommand
UC-A12	Reset MK	userId	ok	ResetPasswordCommand

Bảng 4: Bảng mô tả Use Case theo quản trị viên

Bảng mô tả Use Case theo giảng viên

UC	Tên	Input	Output	Command/Query
UC-T01	Đăng nhập	credentials	JWT	LoginCommand
UC-T10	Thêm chương	chapter	Chapter	AddChapterCommand
UC-T12	Thêm câu hỏi	question	Question	Add*QuestionCommand
UC-T15	Tạo khung đề	template	ExamTemplate	CreateExamTemplateCommand
UC-T18	Tạo đề	exam info	Exam	CreateExamCommand
UC-T19	Sinh mã đề weighted	examId, count	Versions	GenerateExamVersionsCommand

UC-T22	Tạo ca thi	config	ExamSession	CreateExamSessionCommand
UC-T24	Giám sát	sessionId	Actions	GetParticipantActionByExamSessionIdQuery
UC-T25	Kết quả	sessionId	results	GetParticipantsResultQuery

Bảng 5: Bảng mô tả Use Case theo giảng viên

Bảng mô tả Use Case theo sinh viên

UC	Tên	Input	Output	Command/Query
UC-S01	Đăng nhập	Credentials	JWT	LoginCommand
UC-S02	Lịch thi	—	sessions	GetExamSessionsQuery
UC-S03	Vào thi	sessionId	participant	JoinExamSessionCommand
UC-S04	Trả lời	Answer	saved	Answer*QuestionCommand
UC-S07	Nộp bài	participantId	submitted	SubmitExamSessionCommand
UC-S08	Xem KQ	sessionId	score	GetDetailExamResultQuery

Bảng 6: Bảng mô tả Use Case theo sinh viên

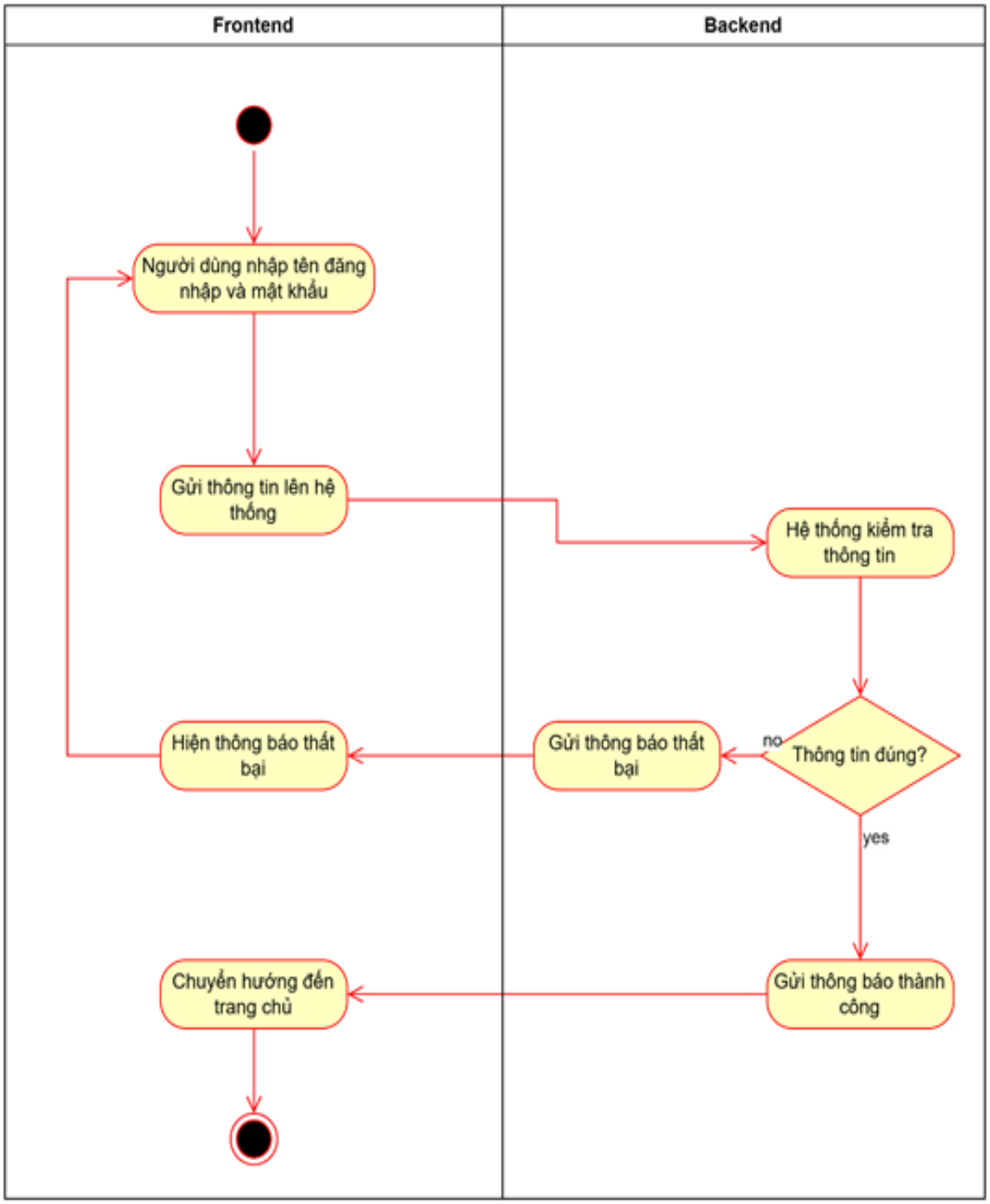
Bảng mô tả Use Case theo thí sinh tự do

UC	Tên	Input	Output	Command/Query
UC-F01	Xác thực mã	code	verified	VerifyCodeCommand
UC-F02	Tham gia	name, email	participant	JoinExamSessionFreeCommand
UC-F04	Anti-cheat	action	logged	AddActionCommand
UC-F05	Nộp bài	participantId	done	SubmitExamSessionCommand

Bảng 7: Bảng mô tả Use Case theo thí sinh tự do

2.3.3. Sơ đồ nguyên lý hoạt động

- Chức năng đăng nhập

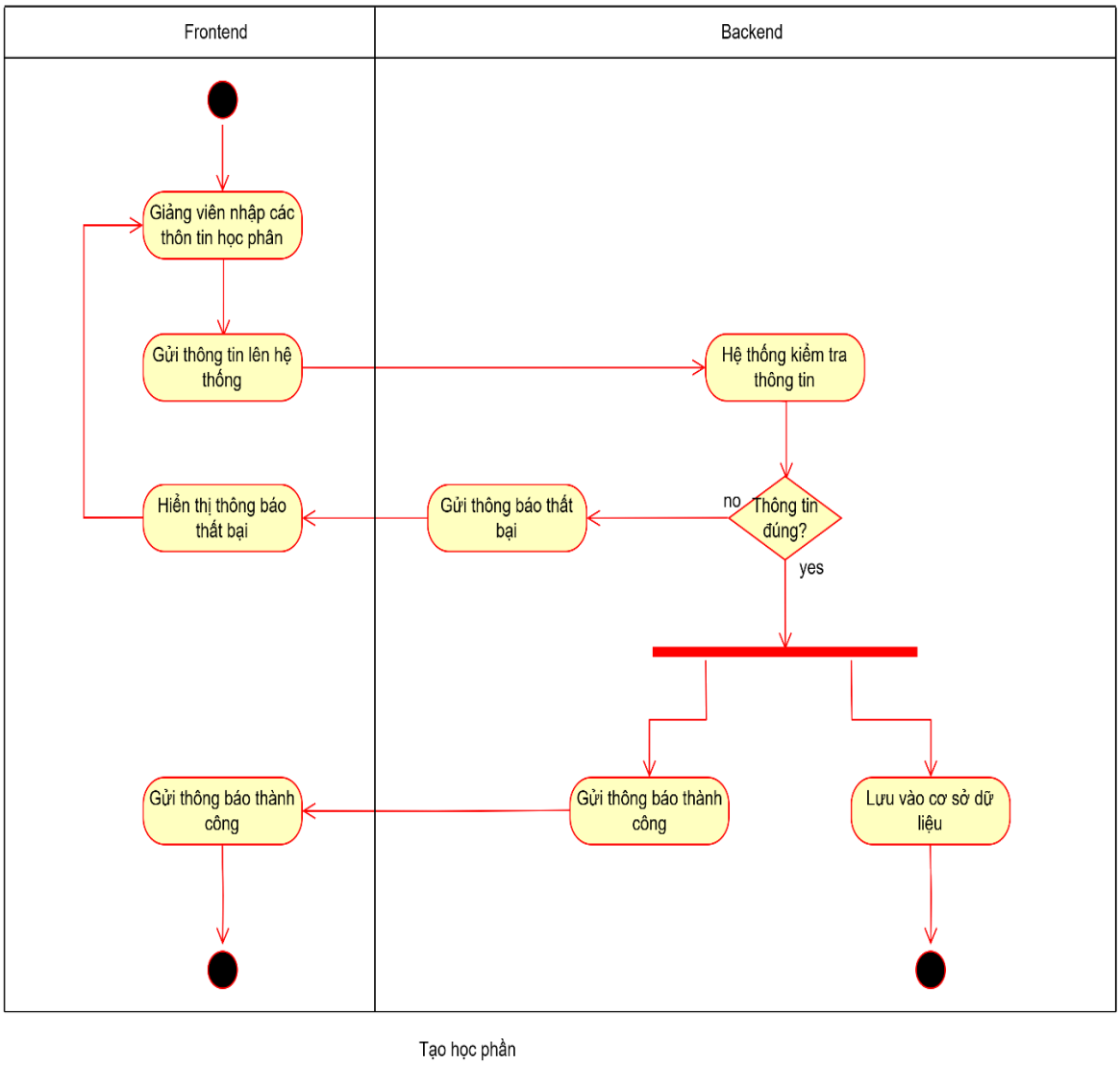


Hình 22: Sơ đồ hoạt động chức năng đăng nhập

Luồng xử lý: Nhập form → kiểm tra đủ field → xác thực tài khoản/mật khẩu → email confirmed → active → phát JWT → điều hướng theo role; nhánh lỗi hiển thị thông báo tương ứng.

Vai trò: Activity diagram làm rõ điều kiện rẽ nhánh và xử lý lỗi; bổ sung cho sequence diagram Hình 6.

- Chức năng tạo học phần

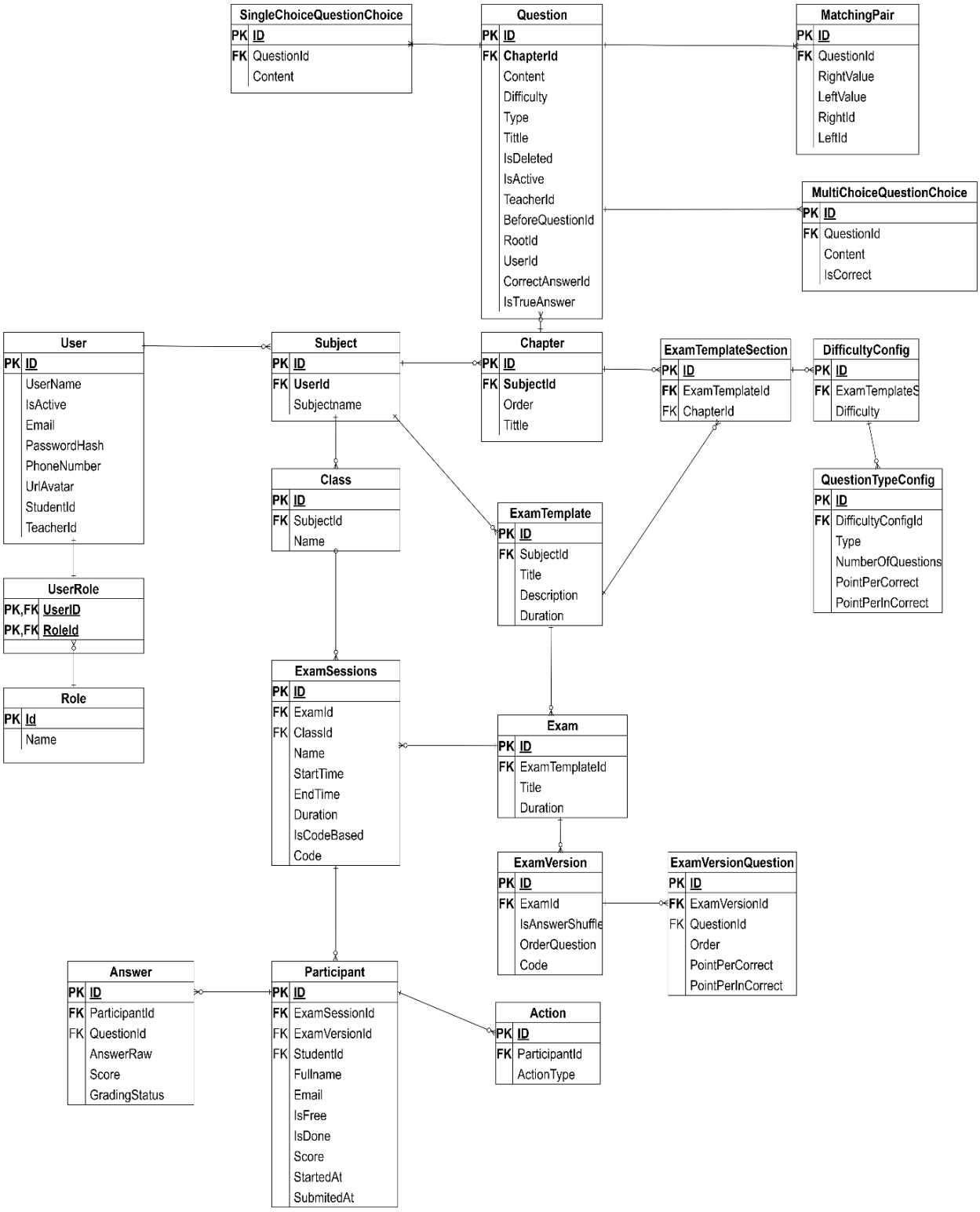


Hình 23: Sơ đồ tuần tự chức năng tạo học phần

Luồng xử lý: Giáo viên nhập tên môn → POST /teacher/subjects → AddTeacherSubjectHandler tạo TeacherSubject gắn OwnerId → lưu DB → trả subjectId.
Vai trò: React (form), Command Handler (ghi dữ liệu), TeacherSubject (aggregate), DbContext (persist).

2.4. Thiết kế cơ sở dữ liệu

2.4.1. Mô hình thiết kế cơ sở dữ liệu



Hình 24: Mô hình thiết kế cơ sở dữ liệu

2.4.2. Chi tiết các bảng dữ liệu

Bảng Role – Phân quyền cho người dùng

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã phân quyền
Name	String		Tên phân quyền

Bảng 8: Bảng Role

Bảng User – Bảng lưu thông tin người dùng

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã người dùng
UserName	String		Tên người dùng
IsActive	Boolean		Trạng thái tài khoản
Email	String	NOT NULL	Email
PasswordHash	String		Mật khẩu đã mã hóa
PhoneNumber	String		Số điện thoại
UrlAvatar	String		Nguồn ảnh đại diện
StudentId	String		Mã số sinh viên
TeacherId	String		Mã số giảng viên

Bảng 9: Bảng User

Bảng UserRole – Bảng lưu phân quyền của người dùng

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
UserId	String	Khóa chính, Khóa ngoại	Mã người dùng
RoleId	String	Khóa chính, khóa ngoại	Mã phân quyền

Bảng 10: Bảng UserRole

Bảng Subject – Bảng lưu các học phần

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã học phần
UserId	String	Khóa ngoại	Mã người dùng
SubjectName	String		Tên học phần

Bảng 11: Bảng học phần

Bảng Chapter – Bảng lưu các chương

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã chương
SubjectId	String	Khóa ngoại	Mã học phần
Title	String		Tiêu đề chương
Order	Integer		Thứ tự

Bảng 12: Bảng Chương

Bảng Question – Bảng lưu các câu hỏi

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã câu hỏi
ChapterId	String	Khóa ngoại	Mã chương
Content	String		Nội dung câu hỏi
Difficulty	Integer		Độ khó
Type	String		Loại câu hỏi
Title	String		Tiêu đề câu hỏi
BeforeQuestionId	String		Mã câu hỏi trước khi chỉnh sửa
RootId	String		Mã câu hỏi gốc
CorrectAnswerId	String		Mã đáp án đúng ()
IsTrueAnswer	Boolean		

Bảng 13: Bảng câu hỏi

Bảng MatchingPair – Bảng lưu các lựa chọn cho câu hỏi ghép nối

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã của cặp ghép nối
QuestionId	String	Khóa ngoại, INDEX	Mã câu hỏi
RightValue	String		Giá trị bên phải
LeftValue	String		Giá trị bên trái
RightId	String		Mã của giá trị bên phải
LeftId	String		Mã của giá trị bên trái

Bảng 14: Bảng lựa chọn ghép nối

Bảng MultiChoiceQuestionChoice – Bảng lưu các lựa chọn cho câu hỏi chọn nhiều

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã của lựa chọn câu hỏi chọn nhiều
QuestionId	String	Khóa ngoại, INDEX	Mã câu hỏi
Content	String		Nội dung của lựa chọn
IsCorrect	Boolean		Đáp án đúng / sai

Bảng 15: Bảng lựa chọn câu hỏi chọn nhiều

Bảng SingleChoiceQuestionChoice – Bảng lưu các lựa chọn cho câu hỏi chọn một

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã của lựa chọn câu hỏi chọn một
QuestionId	String	Khóa ngoại, INDEX	Mã câu hỏi
Content	String		Nội dung của lựa chọn

Bảng 16: Bảng lựa chọn câu hỏi chọn một

Bảng ExamTemplate – Bảng lưu khung đề thi

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã khung đề
SubjectId	String	Khóa ngoại	Mã môn học
Title	String		Tiêu đề
Description	String		Mô tả
Duration	Integer		Thời gian

Bảng 17: Bảng khung đề thi

Bảng ExamTemplateSection – Bảng lưu các phần của khung đề thi

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã phần của khung đề
ExamTemplateId	String	Khóa ngoại	Mã khung đề
ChapterId	String	Khóa ngoại	Mã chương

Bảng 18: Bảng phần của khung đề thi

Bảng DifficultyConfig – Bảng cấu hình độ khó

Bảng 19: Bảng cấu hình độ khó

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã cấu hình độ khó
ExamTemplateSectionId	String	Khóa ngoại	Mã phần của khung đề
Difficulty	Integer		Độ khó

Bảng 20: Bảng cấu hình độ khó

Bảng Exam – Bảng đề thi

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã đề thi
ExamTemplateId	String	Khóa ngoại	Mã khung đề thi
Title	String		Tiêu đề
Duration	String		Thời gian

Bảng 21: Bảng đề thi

Bảng ExamVersion – Bảng lưu phiên bản đề thi

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã phiên bản đề thi
ExamId	String	Khóa ngoại	Mã đề thi
IsAnswerShuffle			
OrderQuestion			
Code			

Bảng 22: Bảng phiên bản đề thi

Bảng ExamVersionQuestion – Bảng lưu các câu hỏi của phiên bản đề thi

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã câu hỏi phiên bản đề thi
ExamVersionId	String	Khóa ngoại	Mã phiên bản của đề thi
QuestionId	String	Khóa ngoại, INDEX	Mã câu hỏi
Order	Integer		Số thứ tự
PointPerCorrect	Float (double)		Điểm cộng trả lời đúng
PointPerInCorerct	Float (double)		Điểm trừ trả lời sai

Bảng 23: Bảng câu hỏi của phiên bản đề thi

Bảng ExamSession – Bảng lưu phiên thi

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã phiên thi
ExamId	String	Khóa ngoại	Mã đề thi
ClassId	String		Mã lớp
Name	String		Tiêu đề
StartTime	Datetime		Thời gian mở đề
EndTime	Datetime		Thời gian đóng đề
Duration	Integer		Thời gian kéo dài
IsCodeBased			
Code			

Bảng 24: Bảng phiên thi

Bảng Action – Bảng lưu các lịch sử hành động

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã lịch sử hành động
ParticipantId	String	Khóa ngoại, INDEX	Mã thí sinh
ActionType	String (nvarchar)	NOT NULL	Loại hành động

Bảng 25: Bảng hành động

Bảng Participant – Bảng lưu mã thí sinh

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã thí sinh
ExamSessionId	String	Khóa ngoại, INDEX	Mã phiên thi
ExamVersionId	String	Khóa ngoại	Loại phiên bản đề thi
StudentId	String	Khóa ngoại	Mã số sinh viên
Fullname	String (nvarchar(max))	NOT NULL	Tên đầy đủ
Email	String (nvarchar(max))	NOT NULL	Email
IsFree	Boolean		Thể hiện có phải là thí sinh tự do
IsDone	Boolean		Thể hiện đã hoàn thành bài thi
Score	Double		Kết quả
StartedAt	Datetime		Thời gian bắt đầu làm bài
SubmittedAt	Datetime		Thời gian nộp bài

Bảng 26: Bảng mã thí sinh

Bảng Answer – Bảng lưu đáp án

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã đáp án
ParticipantId	String	Khóa ngoại, INDEX	Mã thí sinh
QuestionId	String	Khóa ngoại, INDEX	Mã câu hỏi
AnswerRaw	String	NOT NULL	Đáp án gốc
Score	Float (double)		Điểm
GradingStatus	String (nvarchar)	NOT NULL	Trạng thái chấm điểm

Bảng 27: Bảng đáp án

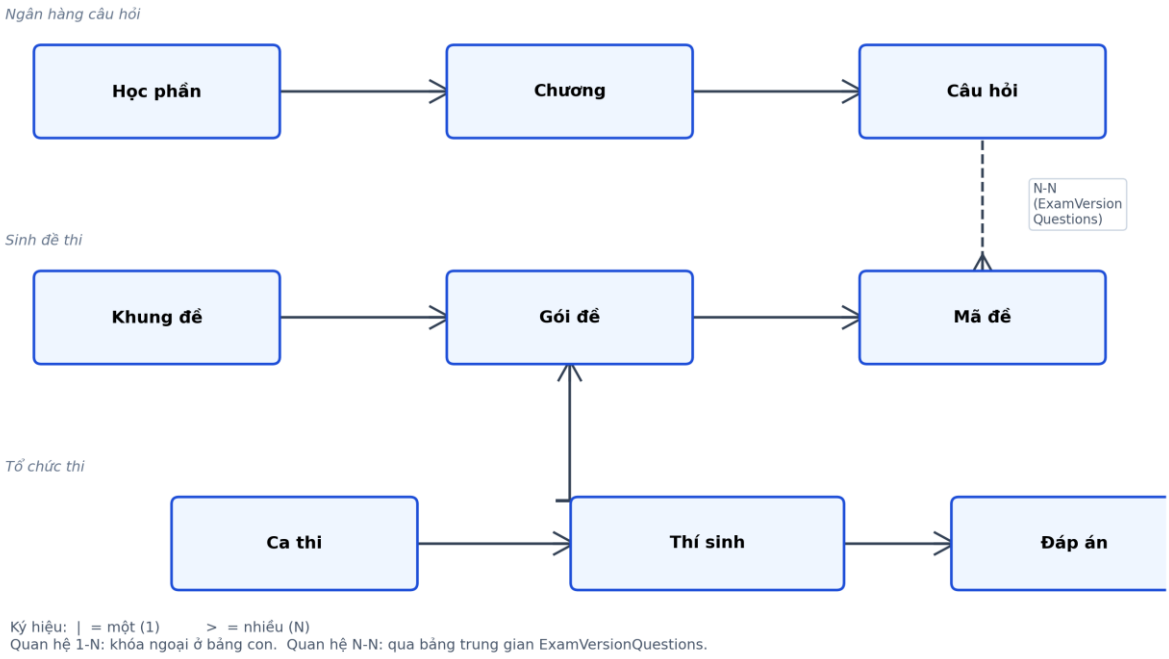
Bảng Class – Bảng lưu lớp học phần

Tên trường	Kiểu dữ liệu	Ràng buộc	Diễn giải
Id	String	Khóa chính	Mã lớp học phần
SubjectId	String	Khóa ngoại	Mã học phần
Name	String		Tiêu đề

Bảng 28: Bảng lớp học phần

2.4.3. Sơ đồ ERD

Sơ đồ ERD làm rõ cardinality giữa các thực thể nghiệp vụ. Ký hiệu crow's foot: đoạn thẳng kết thúc bằng gạch đứng (|) biểu diễn quan hệ một (one); mũi tên (>) biểu diễn quan hệ nhiều (many). Quan hệ N-N được thể hiện qua bảng trung gian (ClassStudents, ExamVersionQuestions, AspNetUserRoles).



Hình 25: Sơ đồ ERD

Quan hệ một-nhiều (1-N): một bản ghi ở bảng cha có nhiều bản ghi ở bảng con thông qua khóa ngoại đặt tại bảng con (ví dụ: một ExamSession có nhiều ExamSessionParticipants). Quan hệ nhiều-nhiều (N-N): hai thực thể liên kết qua bảng

trung gian chứa hai khóa ngoại (ví dụ: ClassStudents nối Class và Student; ExamVersionQuestions nối ExamVersion và Question).

Bảng quan hệ một-nhiều (1-N)

Quan hệ	**Cardinality**	**FK / bảng trung gian**	**Mô tả**
TeacherSubjects → Chapters	1-N	Chapters.SubjectId	Một học phần có nhiều chương
Chapters → Questions	1-N	Questions.ChapterId	Ngân hàng câu hỏi (TPH, discriminator Type)
ExamTemplates → ExamTemplateSections	1-N	ExamTemplateSections.ExamTemplateId	Ma trận đề theo chương
Exams → ExamVersions	1-N	ExamExamVersions.ExamId	Nhiều mã đề cho một gói đề
ExamVersions ↔ Questions	N-N	ExamVersionQuestions	Bảng trung gian: Order, PointPerCorrect
ExamSessions → Participants	1-N	ExamSessionParticipants.ExamSessionId	Danh sách thí sinh theo ca thi
Participants → Answers / Actions	1-N	ParticipantAnswers, ParticipantActions	Đáp án và log anti-cheat
Classes ↔ Students	N-N	ClassStudents	Sinh viên đăng ký nhiều lớp học phần
AspNetUsers ↔ Roles	N-N	AspNetUserRoles	Phân quyền ADMIN / TEACHER / STUDENT

Bảng 29: Quan hệ một-nhiều (1-N)

2.4.5. Ràng buộc toàn vẹn và quy tắc Cascade

Hệ thống dùng EF Core với OwnsMany/OwnsOne: các entity con (ParticipantAnswers, ParticipantActions, ExamVersionQuestions, …) được cấu hình CASCADE khi xóa aggregate cha. Value Object (Email, FullName, Score) được validate ở tầng Domain rồi map sang cột nvarchar/float trong SQL Server — không phải kiểu Integer.

Bảng hệ thống dùng EF Core với OwnsMany/OwnsOne

Cặp quan hệ	**On Delete**	**Ghi chú**
ExamSession → Participants → Answers/Actions	CASCADE	EF Core OwnsMany — xóa ca thi xóa dữ liệu con
Exam → ExamVersions → ExamVersionQuestions	CASCADE	Xóa đề → xóa mã đề và câu trong mã đề
Question → Choices / MatchingPairs	CASCADE	Xóa câu hỏi → xóa lựa chọn con (TPH)
AspNetUsers → UserRoles / Claims	CASCADE	Chuẩn ASP.NET Identity / IdentityServer
ExamSession → Exam	RESTRICT*	Không xóa Exam khi còn ca thi

Bảng 30: Hệ thống dùng EF Core với OwnsMany/OwnsOne

2.4.6. Index và tối ưu truy vấn

Ngoài khóa chính, EF Core và ASP.NET Identity tạo index trên khóa ngoại và cột tra cứu thường xuyên. Các truy vấn ca thi (participants, actions), sinh đề (questions theo chapter) và đăng nhập (username/email) được tối ưu nhờ index có sẵn hoặc do EF tạo tự động trên FK.

Bảng ngoài khóa chính, EF Core và ASP.NET Identity tạo index trên khóa ngoại và cột tra cứu thư

Bảng / cột	**Loại index**	**Mục đích tối ưu**
ExamSessionParticipants.ExamSessionId	INDEX (FK)	Load danh sách thí sinh theo ca thi
ExamSessionParticipants.ExamVersionId	INDEX (FK)	Tra cứu mã đề đã gán cho thí sinh
ExamVersionQuestions.ExamVersionId	INDEX (FK)	Load câu hỏi theo mã đề khi join thi
Questions.ChapterId	INDEX (FK)	Lọc ngân hàng câu khi sinh đề weighted random
AspNetUsers.NormalizedUserName	UNIQUE INDEX	Đăng nhập nhanh, không trùng username
AspNetUsers.NormalizedEmail	INDEX	Tra cứu email, xác thực tài khoản
ParticipantActions.ParticipantId	INDEX (FK)	Giám sát hành vi vi phạm theo thí sinh
PendingQuestionImages (UserId, IsAttached)	COMPOSITE INDEX	Dọn ảnh orphan theo giáo viên

Bảng 31: Ngoài khóa chính, EF Core và ASP.NET Identity tạo index trên khóa ngoại và cột tra cứu thư

Lưu ý đã sửa mô tả bảng Participant và Answer: FullName, Email kiểu String (nvarchar); Score kiểu Float (double); ActionType và GradingStatus kiểu String (enum lưu dạng chuỗi); ExamSessionId và ParticipantId có ràng buộc khóa ngoại kèm index.

2.5. Tổng kết chương

Chương này trình bày các đặc tả yêu cầu mà website có thể đáp ứng cho nhu cầu của người dùng. Những yêu cầu này chủ yếu là yêu cầu thiết kế, các sơ đồ hoạt động, vẽ luồng hoạt động của một website, hiển thị các bước và chuỗi hành động cũng như các tương tác giữa người dùng và website. Qua đó tổng quan và các luồng hoạt động của website được trình bày đầy đủ.

CHƯƠNG III: TRIỂN KHAI THỰC TẾ

3. Môi trường, kĩ thuật và công cụ phát triển dự án

3.1. Môi trường cài đặt

Để triển khai đề tài này, em đã vận dụng kiến thức và kinh nghiệm đã tích góp tại trường nhằm áp dụng những công cụ và môi trường phát triển thích hợp giúp em có thể xây dựng nên hệ thống.

- Công cụ phát triển: Visual Studio Code, Postman, Microsoft SQL Server.
- Ngôn ngữ phát triển backend:C#, ASP.NET, SQL Server.
- Ngôn ngữ phát triển frontend: JavaScript, React.js,
- Cơ sở dữ liệu: SQL Server

3.2. Xây dựng thuật toán lựa chọn câu hỏi tối ưu cho việc sinh đề thi

3.2.1. Mô tả bài toán

Trong các hệ thống thi cử trực tuyến, việc sinh đề thi tự động từ ngân hang câu hỏi là một yêu cầu quan trọng nhằm đảm bảo tính đa dạng, công bằng và tiết kiệm thời gian cho giảng viên. Tuy nhiên, quá trình sinh đề không thể thực hiện một cách hoàn toàn ngẫu nhiên, mà cần đảm bảo các tiêu chí cụ thể về nội dung, mức độ khó, độ tin cậy của câu hỏi, cũng như sự khác biệt giữa các đề thi

- Đầu vào của bài toán:
  - Chương (hoặc chủ đề kiến thức)
  - Mức độ khó
  - Loại câu hỏi
  - Phân bố mức độ khó
  - Số lượng câu hỏi cho mỗi chương
- Đầu ra của bài toán:

- Một đề thi hợp lệ thõa mãn phân bố theo khung đề, hạn chế trùng lặp với đề thi gần đây, các câu hỏi không trùng lặp nội dung hoặc mục tiêu đánh giá

3.2.2. Giải pháp đề xuất

- Mục tiêu:
  - Đáp ứng đầy đủ cấu trúc khung đề thi (chương, mức độ khó, loại câu hỏi)
  - Đảm bảo độ tin cậy và chất lượng đánh giá của đề thi
  - Tối ưu hóa tính đa dạng giữa các đề thi và giảm trùng lặp
- Ý tưởng chính
  - Không chọn hoàn toàn ngẫu nhiên, mà chọn theo trọng số phản ánh mức độ phù hợp và chất lượng của từng câu hỏi
  - Sử dụng kỹ thuật "ngẫu nhiên có trọng số" (weighted random selection) để vừa giữ tính ngẫu nhiên, vừa ưu tiên câu hỏi tốt

3.2.3. Quy trình thực thi thuật toán

- Tiền xử lý dữ liệu:
  - Đọc và phân tích khung đề thi, xác định số lượng câu hỏi cần theo từng mức độ khó và từng chương
  - Lọc ngân hang câu hỏi để loại bỏ các câu không hợp lệ (ví dụ: chưa được duyệt, quá cũ, phản hồi tiêu cực)
- Phân nhóm câu hỏi:
  - Phân loại ngân hàng câu hỏi theo chương, mức độ khó và loại câu hỏi
  - Tạo các nhóm ứng với từng tổ hợp ràng buộc trong khung đề thi
- Tính trọng số cho từng câu hỏi:
  - Tính toán trọng số dựa trên các yếu tố đã được nên ở trên
- Lựa chọn câu hỏi ngẫu nhiên có trọng số:
  - Với mỗi nhóm, chọn ngẫu nhiên số lượng câu hỏi yêu cầu bằng xác suất tỉ lệ với trọng số.
  - Đảm bảo rằng không chọn trùng các câu hỏi đã có trong đề thi gần đây, và không chọn các câu hỏi quá giống nhau trong cùng đề thi
- Kiểm tra tính hợp lệ và đa dạng của đề thi:

- Kiểm tra tổng thể đề thi có đáp ứng yêu cầu khung đề không
- Nếu không hợp lệ, quay lại bước chọn lại một số câu hỏi khác

3.2. Phân tích bảo mật và xác thực

- Hệ thống quản lý thi trực tuyến xử lý dữ liệu nhạy cảm (tài khoản người dùng, ngân hàng câu hỏi, đáp án, kết quả thi). Do đó, lớp bảo mật và xác thực được thiết kế tách biệt giữa tầng trình diễn (React SPA), tầng API (ASP.NET Core) và tầng danh tính (Duende IdentityServer). Phần này phân tích cơ chế đã triển khai trong mã nguồn, đánh giá mức độ bảo vệ thực tế và các hướng cải thiện phù hợp phạm vi đồ án.

3.2.1. Kiến trúc xác thực (Authentication)

- Luồng xử lý: Người dùng nhập tên đăng nhập và mật khẩu trên giao diện React → gọi POST /identity/login → LoginHandler xác minh tài khoản qua ASP.NET Identity (UserManager) → chuyển tiếp yêu cầu tới IdentityServer endpoint /connect/token với grant type Resource Owner Password (ROPC) → nhận access token, refresh token và thông tin role → frontend lưu token vào localStorage và gắn header Authorization: Bearer cho mọi request sau.
- Vai trò: React (Login.jsx, axiosConfig.js) điều phối phiên làm việc phía client; LoginHandler (Application) là cổng nghiệp vụ đăng nhập; Duende IdentityServer phát hành JWT; UserManager tra cứu và xác thực thông tin người dùng trong SQL Server. Refresh token được xử lý tự động qua interceptor axios khi API trả về mã 401.
- Cấu hình IdentityServer (client magic) cho phép access token sống 3600 giây, refresh token dạng sliding one-time use. Hệ thống định nghĩa ba vai trò nghiệp vụ: ADMIN, TEACHER và STUDENT; mỗi endpoint Carter gắn policy RequireAuthorization tương ứng (PolicyConstant), ví dụ endpoint giáo viên yêu cầu role TEACHER, endpoint quản trị yêu cầu ADMIN.

3.2.2. Các biện pháp bảo mật đã triển khai

- Phân quyền theo vai trò (RBAC): Toàn bộ nghiệp vụ quản lý ngân hàng câu, sinh đề, ca thi và kết quả được bảo vệ bằng JWT Bearer kèm kiểm tra role trên server — client không thể gọi API giáo viên nếu không mang token hợp lệ với role TEACHER hoặc ADMIN.
- Xác thực hai lớp cho thao tác nhạy cảm: Chức năng đặt lại mật khẩu sử dụng mã OTP gửi qua email, lưu tạm trên Redis (SendOtpService) với thời hạn, mã OTP được băm bằng PBKDF2 (OtpExtension) trước khi so khớp — hạn chế lộ mã OTP dạng plaintext trong cache.

- CORS và cookie: API cấu hình CORS cho origin frontend (localhost:5173), cho phép credentials; ApplicationCookie chuyển hướng login thành HTTP 401 thay vì redirect — phù hợp mô hình SPA.
- Tự động nộp bài server-side: Khi thí sinh vào thi, Hangfire lên lịch job AutoSubmitService sau đúng thời lượng ca thi, đảm bảo nộp bài kể cả khi client đóng trình duyệt — giảm rủi ro treo phiên thi.

3.3. Cơ chế chống gian lận (Anti-cheat)

- Chống gian lận trong thi trực tuyến được triển khai theo mô hình giám sát hành vi phía client, ghi log phía server và thông báo realtime cho giáo viên. Cơ chế tập trung vào luồng thí sinh tự do và sinh viên đi qua ExamPageFree.jsx; giao diện ExamPage.jsx dành cho sinh viên không tích hợp fullscreen hay ghi hành vi.

3.3.1. Giám sát phía client

- Luồng xử lý: Khi vào trang thi, ExamPageFree tự động yêu cầu fullscreen → lắng nghe sự kiện fullscreenchange, blur, visibilitychange và mouseleave → nếu thoát fullscreen hoặc chuyển tab thì ghi nhận vi phạm (ActionType.ExitScreen hoặc ClickOutSize) → gọi POST /free/{participantId}/actions → đồng thời invoke SignalR SendStatus tới NotificationHub → giáo viên nhận sự kiện ReceiveStatus trên ExamSessionDetail.
- Vai trò: ExamPageFree (UI + event listener) là tuyến phát hiện; AddActionHandler ghi ParticipantAction vào SQL Server; NotificationHub + Redis cache (connectionId) đẩy trạng thái realtime; giáo viên tra cứu lịch sử qua GET /teacher/exam-sessions/{id}/actions. Khi thí sinh quay lại fullscreen, hệ thống ghi ReturnScreen (4).
- Các loại hành vi được mã hóa trong enum ActionType: StartedExam (0), SubmittedExam (1), ExitScreen (2), ClickOutSize (3), ReturnScreen (4). Giao diện chặn thao tác làm bài khi requireFullscreen = true, buộc thí sinh bấm "Quay lại fullscreen" trước khi tiếp tục.

3.3.2. Xử lý phía server và tự động nộp bài

- Server không dựa hoàn toàn vào đồng hồ client: JoinExamSessionHandler và JoinExamSessionFreeHandler đều đăng ký Hangfire job SubmitExamAsync sau Duration phút. AutoSubmitService gửi SignalR AutoSubmitSuccess cho thí sinh (nếu còn kết nối) rồi gọi SubmitExamSessionCommand với cờ IsAutoSubmit = true.

- Tại tầng domain, phương thức Participant.SubmitAnswer từng có logic từ chối trả lời khi trạng thái vi phạm gần nhất là ExitScreen — hiện đoạn kiểm tra này bị comment, nghĩa là server vẫn chấp nhận đáp án dù client vừa báo vi phạm. Việc giám sát mang tính "ghi nhận và cảnh báo" hơn là "ép buộc dừng thi".

3.3.3. Đánh giá hiệu quả và hướng hoàn thiện

- Điểm mạnh: Kết hợp fullscreen, log hành vi có timestamp, giám sát realtime qua SignalR và auto-submit server-side tạo nên chuỗi giám sát khép kín hơn so với chỉ dùng timer phía trình duyệt. Giáo viên có thể đối chiếu hành vi thí sinh với timeline ca thi khi chấm điểm hoặc xử lý khiếu nại.
    - Bảng tóm tắt mức độ bảo vệ các hạng mục chính

Hạng mục	**Mức độ**	**Nhận xét**
Phân quyền Admin/Teacher API	Tốt	JWT + policy role trên Carter endpoint
Xác thực khi làm bài (/free/*)	Tốt	AllowAnonymous, chỉ dựa participantId
Giám sát hành vi (client)	Khá	Fullscreen + log ExitScreen/ReturnScreen
Giám sát realtime	Khá	SignalR + Redis connection mapping
Tự động nộp bài	Tốt	Hangfire server-side theo duration

Bảng 32: Tóm tắt mức độ bảo vệ các hạng mục chính

3.4. Đánh giá UX/UI và khả năng sử dụng

Giao diện người dùng (UI) và trải nghiệm sử dụng (UX) quyết định mức độ chấp nhận của hệ thống thi trực tuyến trong môi trường giáo dục. Phần này đánh giá thiết kế giao diện React SPA đã triển khai, mô tả quy trình thao tác theo từng vai trò, nhận xét khả năng sử dụng (usability) và so sánh với hai nền tảng phổ biến: Moodle (LMS) và Google Form (khảo sát/bài tập trực tuyến).

3.4.1. Kiến trúc giao diện và nguyên tắc thiết kế

- Hệ thống xây dựng trên React 19 (Vite), kết hợp Tailwind CSS cho layout tổng thể, Ant Design và Material UI cho một số thành phần dữ liệu (bảng, dialog, phân trang). Cấu trúc DefaultLayout gồm SideNavbar cố định bên trái (có thể thu gọn), vùng nội dung chính và TopNavBar — mô hình dashboard quen thuộc, giảm thời gian làm quen cho người dùng nội bộ trường học.
- Menu điều hướng được lọc theo role lưu trong JWT (ADMIN, TEACHER, STUDENT): quản trị viên thấy mục Giảng viên/Sinh viên; giáo viên thấy Lịch thi, Lớp học phần, Môn học, Khung đề, Đề thi; sinh viên chỉ thấy Lịch thi. Sau

đăng nhập, HomeRedirect chuyển hướng thẳng tới chức năng chính của từng role thay vì trang dashboard trung gian — rút ngắn một bước thao tác.

- Ngôn ngữ giao diện, nhãn nút và thông báo lỗi sử dụng tiếng Việt (ví dụ validate form đăng nhập, toast thao tác thành công). Trang làm bài (ExamPageFree) phân biệt loại câu bằng màu nền (single choice vàng, multi choice xanh lá, ghép đôi xanh dương…) và nút "Lưu đáp án" kèm phản hồi "Đã lưu đáp án" — hỗ trợ nhận biết trạng thái từng câu.

3.4.2. Quy trình thao tác theo vai trò

- Quản trị viên: Đăng nhập → danh sách giảng viên/sinh viên (bảng có tìm kiếm, phân trang) → xem chi tiết theo tab Hồ sơ / Tài khoản → thao tác khóa-mở tài khoản, đặt lại mật khẩu qua OTP email. Luồng tập trung quản lý tài khoản, ít bước, phù hợp nghiệp vụ IT/phòng đào tạo.
- Giảng viên: Sidebar dẫn theo chuỗi nghiệp vụ Môn học → Chương → Ngân hàng câu (AddQuestion hiển thị 5 loại câu dạng thẻ có icon) → Khung đề thi → Gói đề → Sinh mã đề → Tạo ca thi → ExamSessionDetail (tab Thông tin, Thí sinh, Vi phạm, Chấm tự luận, Kết quả, Phúc khảo). Giáo viên có thể giám sát realtime vi phạm và chốt điểm trên cùng một màn hình ca thi.
- Sinh viên: Màn Lịch thi (Material UI Table) liệt kê ca thi → hộp thoại xác nhận tham gia → chuyển sang ExamPageFree (dùng chung với thí sinh tự do) → làm bài từng câu → nộp bài → xem kết quả tại ExamResult. Thí sinh tự do: truy cập URL/mã ca (ExamPageInfo) → nhập họ tên, email không cần tài khoản → vào thi với cơ chế fullscreen và giám sát hành vi.
- Luồng xử lý tổng quát: Người dùng chọn mục sidebar → React Router điều hướng trang → gọi API qua axios (JWT/interceptor) → hiển thị bảng/form hoặc redirect bước kế tiếp. Vai trò: SideNavbar (điều hướng theo role), ProtectedRoute (chặn truy cập trái phép), các page component (nghiệp vụ), react-toastify (phản hồi tức thì).

3.4.3. Đánh giá khả năng sử dụng (usability)

- Nhận xét tổng quát: UI nhìn chung dễ dùng đối với người đã quen dashboard web và quy trình thi trực tuyến. Điểm mạnh: (1) Phân tách rõ bốn persona (admin, giáo viên, sinh viên, thí sinh tự do); (2) Chuỗi menu giáo viên bám sát quy trình nghiệp vụ từ ngân hàng câu đến ca thi; (3) Chọn loại câu hỏi trực quan bằng thẻ icon; (4) Trang ca thi gom nhiều tab giám sát/chấm điểm, giảm chuyển trang; (5) Form đăng nhập có validate client-side; (6) Làm bài có phản hồi lưu từng câu, đồng hồ đếm ngược và cảnh báo vi phạm fullscreen.
- Theo các tiêu chí usability cơ bản (learnability, efficiency, error prevention, satisfaction): hệ thống đạt mức khá cho giáo viên và sinh viên đã được hướng

dẫn; learnability của giáo viên mới ở mức trung bình do độ phức tạp nghiệp vụ; error prevention ở form nhập liệu ổn nhưng thiếu xác nhận ở thao tác quan trọng (sinh đề, chốt điểm) ở mức có thể cải thiện thêm.

3.4.4. So sánh với Moodle và Google Form

- Moodle là LMS đầy đủ với hàng chục module (khóa học, diễn đàn, nhóm, quiz plugin…). Google Form tập trung tạo form khảo sát/bài tập nhanh, không có ngân hàng câu hỏi hay sinh đề. Hệ thống đồ án chọn lọc phạm vi: thi trắc nghiệm có giám sát, ngân hàng câu có cấu trúc chương/môn, sinh nhiều mã đề weighted random — gần nhu cầu thi cụ thể hơn Moodle quiz cơ bản, chuyên sâu hơn Google Form.

Bảng so sánh nhanh ba hướng triển khai

Tiêu chí	**Hệ thống đồ án**	**Moodle**	**Google Form**
Độ dễ học (learnability)	Khá — menu theo role, tiếng Việt	Thấp–TB — nhiều module, cấu hình phức tạp	Cao — tạo form vài phút
Quy trình ra đề thi	Ngân hàng câu → khung đề → sinh mã đề → ca thi	Ngân hàng câu + quiz trong khóa học	Tạo form mới, không tái sử dụng ngân hàng
Sinh đề / trộn đề	Weighted random, nhiều mã đề	Có (plugin/random)	Không
Giám sát thi (anti-cheat)	Fullscreen, log hành vi, SignalR	Plugin proctoring (cài thêm)	Không
Phân quyền người dùng	Admin / Teacher / Student	Đa vai trò LMS	Chủ sở hữu form + respondent
Giao diện & tùy biến	Tailwind + component library	Theme Moodle, ít linh hoạt	Giao diện Google, ít tùy biến
Phù hợp bài thi có cấu trúc	Cao — TPH 5 loại câu	Cao	Thấp — chủ yếu MCQ/short answer

Bảng 33: So sánh nhanh ba hướng triển khai

3.5. Triển khai thực nghiệm

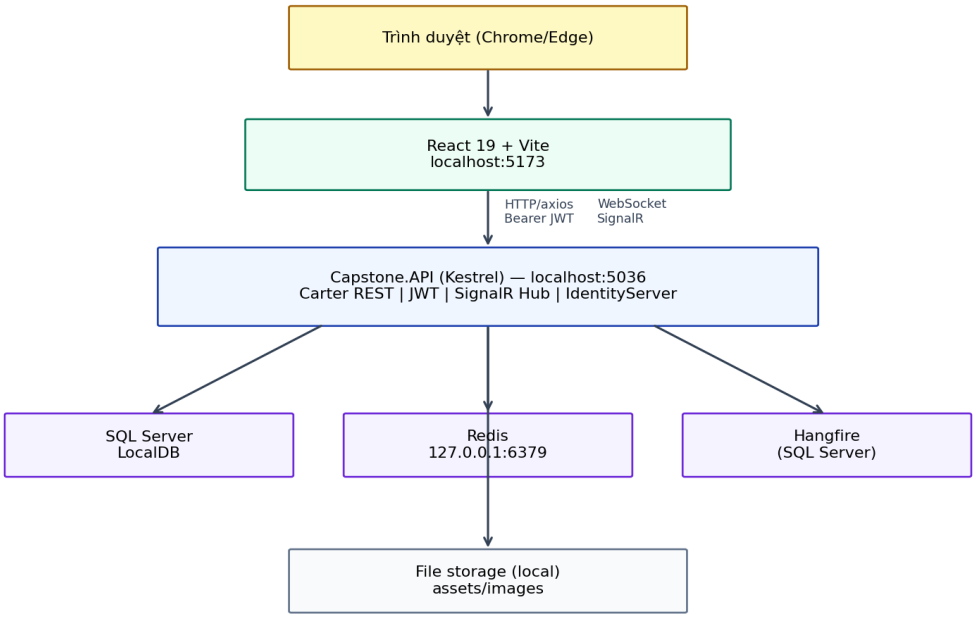
Trong phạm vi đồ án, hệ thống được triển khai và kiểm thử trên môi trường phát triển local (máy trạm Windows), chưa deploy lên server production hay cloud. Phần này mô tả kiến trúc runtime thực tế khi vận hành, quy trình khởi chạy các thành phần, đồng thời trình bày kiến trúc triển khai đề xuất khi đưa hệ thống vào sử dụng thật tại trường.

3.5.1. Kiến trúc runtime môi trường local

Khi chạy thử nghiệm, hệ thống gồm các process độc lập giao tiếp qua mạng loopback. Người dùng truy cập React SPA do Vite phục vụ tại http://localhost:5173. Frontend

gọi API ASP.NET Core (Kestrel) tại http://localhost:5036 qua axios (baseURL trong axiosConfig.js), gắn header Authorization: Bearer cho request có xác thực. SignalR kết nối tới /notification-hub trên cùng origin API để giám sát vi phạm và auto-submit realtime.

Backend tích hợp Duende IdentityServer (phát hành JWT), Carter minimal API, MediatR (CQRS), EF Core truy cập SQL Server LocalDB (connection string trong appsettings.json), Redis cho OTP và ánh xạ connectionId SignalR, Hangfire lưu job trên SQL Server (auto-submit ca thi, dọn ảnh câu hỏi orphan). Ảnh câu hỏi lưu local qua S3Service (thư mục assets/images).



Hình 26: Kiến trúc runtime môi trường triển khai local

Bảng các thành phần runtime

Thành phần	**Công nghệ**	**Địa chỉ / cấu hình**	**Vai trò runtime**
Frontend	React 19 + Vite	localhost:5173	Giao diện SPA theo role
Backend API	ASP.NET Core 8 (Kestrel)	localhost:5036	REST, JWT, SignalR, IdentityServer
Cơ sở dữ liệu	SQL Server LocalDB	(localdb)\MSSQLLocalDB — DB Capstone	Persist dữ liệu nghiệp vụ + Hangfire
Cache	Redis	127.0.0.1:6379	OTP reset password, map connection SignalR
Background job	Hangfire	Storage: SQL Server	Auto-submit ca thi, cleanup ảnh
File	S3Service (local)	Capstone.Web/.../assets/images	Lưu ảnh câu hỏi

Bảng 34: Các thành phần runtime

3.5.2. Quy trình khởi chạy hệ thống

Quy trình triển khai thực nghiệm trên máy phát triển:

- Khởi động SQL Server LocalDB và Redis (redis-server hoặc Docker Redis tùy máy).
- Chạy migration/seed — API tự gọi InitializeDatabaseAsync khi khởi động.
- dotnet run project Capstone.API (profile http/https, port 5036/7066).
- npm run dev trong Capstone.Web/capstone (Vite port 5173).
- Truy cập Swagger tại /swagger (Development) hoặc đăng nhập qua giao diện React.

Môi trường Development bật Swagger, JWT validate issuer/audience lỏng; CORS cho phép origin localhost:5173. Dữ liệu thử nghiệm: 62 câu hỏi, 12 đề, 10 ca thi (theo kết quả đo API ~13 ms trung bình trên local). Toàn bộ kiểm thử chức năng, UAT và demo đồ án thực hiện trên cấu hình này.

3.5.3. Giới hạn phạm vi và hướng phát triển

Đồ án tập trung hoàn thiện nghiệp vụ thi trực tuyến và kiểm thử trên môi trường local; chưa triển khai Docker Compose, Kubernetes hay hosting cloud thực tế — phù hợp phạm vi thời gian và nguồn lực tốt nghiệp. Việc nêu rõ runtime local và kiến trúc đề xuất thể hiện nhận thức về vận hành hệ thống, là cơ sở cho giai đoạn triển khai thật sau đồ án (container hóa, monitoring, load balancing khi số lượng thí sinh lớn).

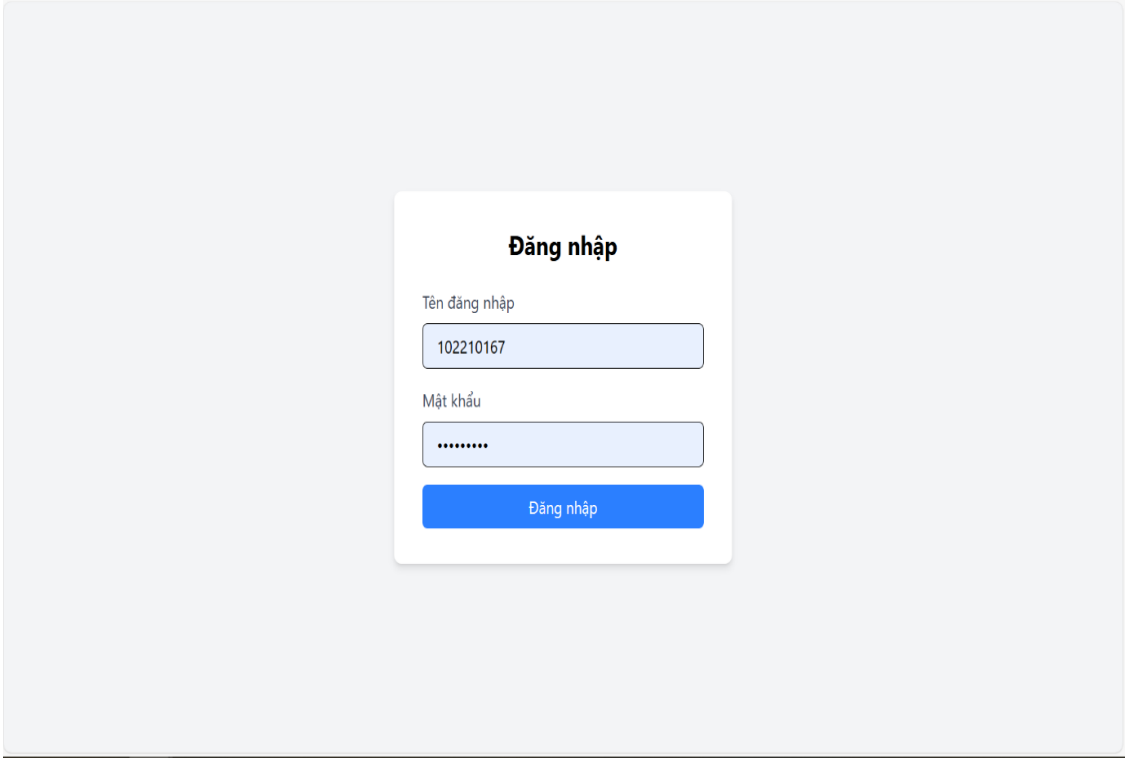
3.6. Kết quả thực tế — minh họa giao diện

Hệ thống triển khai bốn giao diện theo vai trò người dùng (minh họa tại các hình H24 trở đi):

- Giao diện đối với người dùng là quản trị viên
- Giao diện đối với người dùng là giảng viên
- Giao diện đối với người dùng là quản lý sinh viên
- Giao diện đối với người dùng là thí sinh tự do

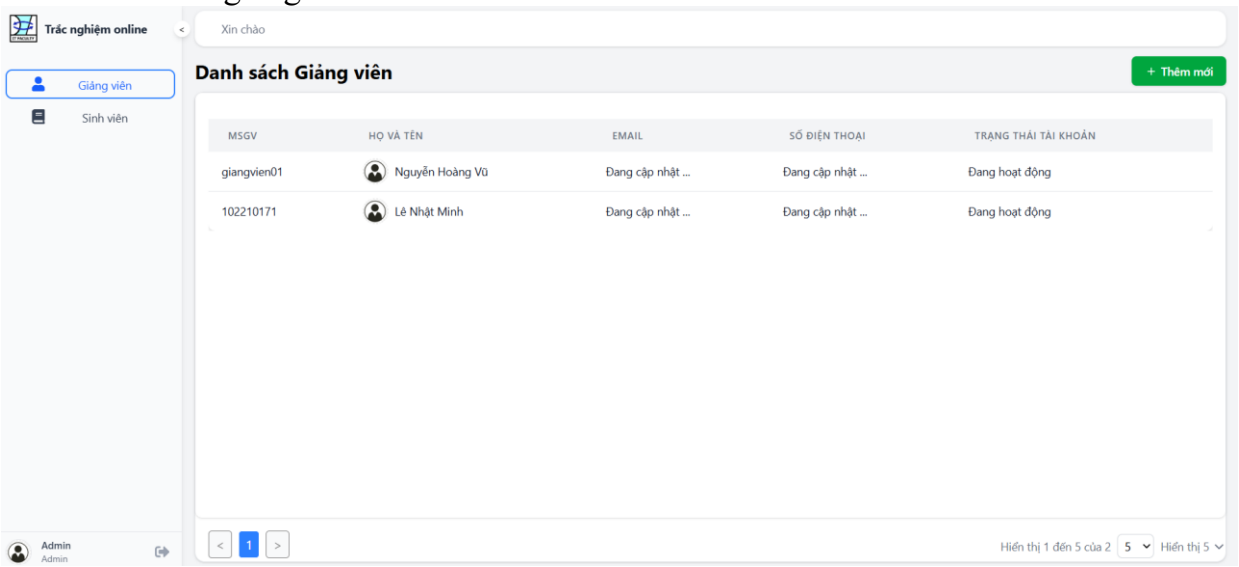
3.6.1. Giao diện đối với người dùng là quản trị viên

- Đăng nhập



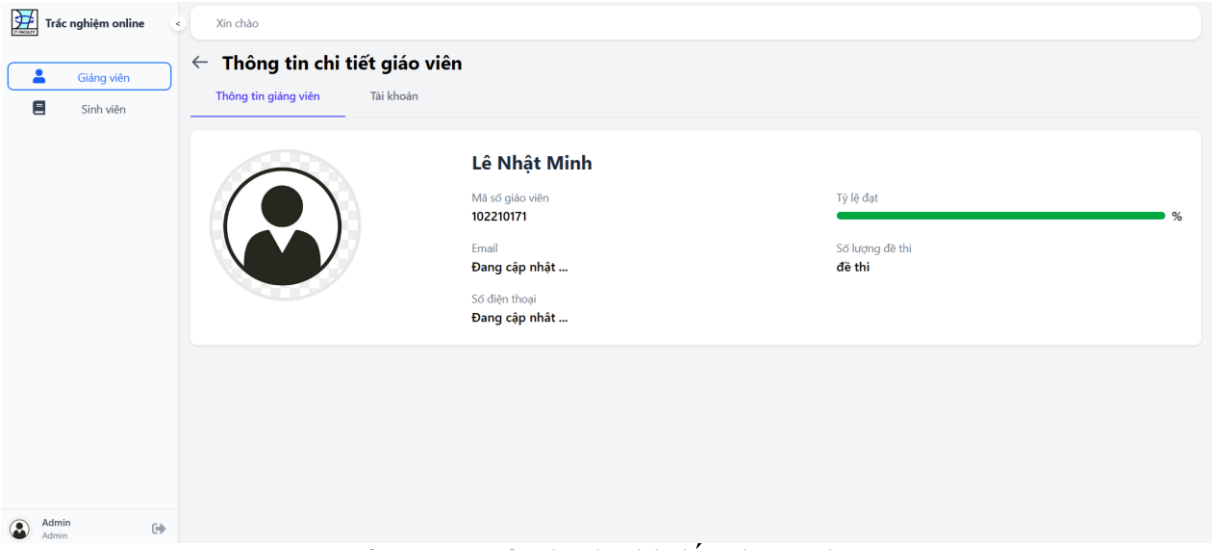
Hình 27: Màn hình đăng nhập quản trị viên

Xem danh sách giảng viên



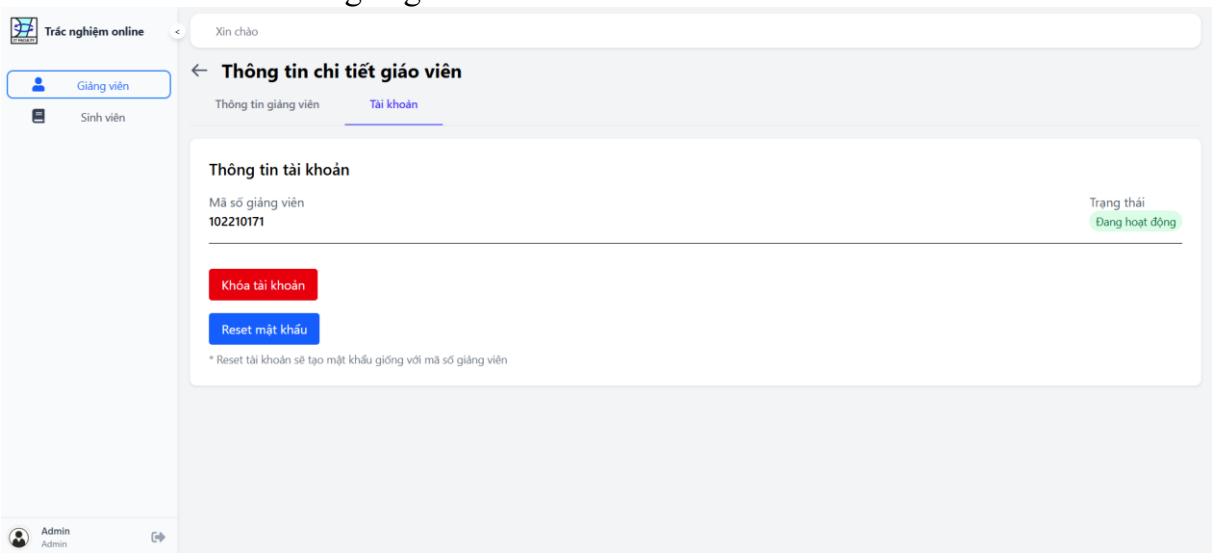
Hình 28: Màn hình danh sách giảng viên

- Xem chi tiết giảng viên



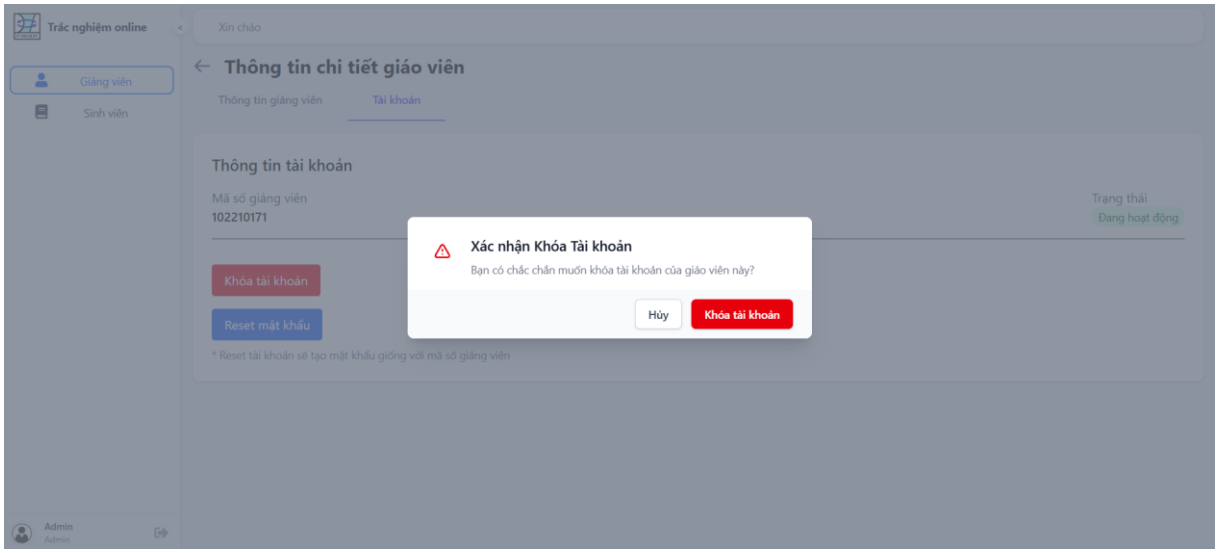
Hình 29: Màn hình chi tiết giảng viên

- Màn hình chi tiết giảng viên



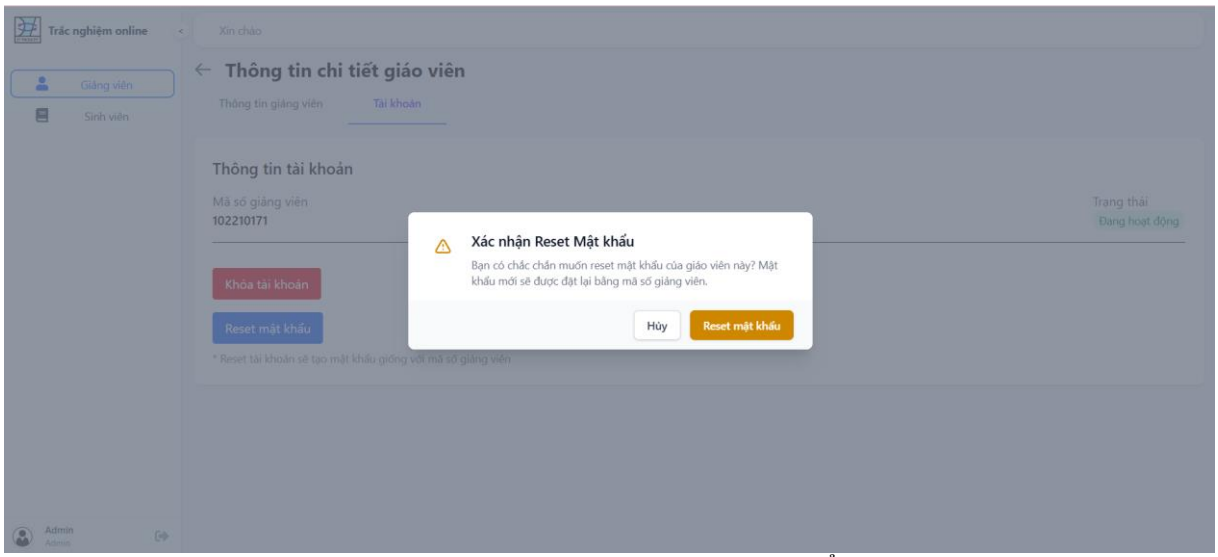
Hình 30: Màn hình khóa tài khoản

- Khóa tài khoản



Hình 31: Màn hình mở tài khoản

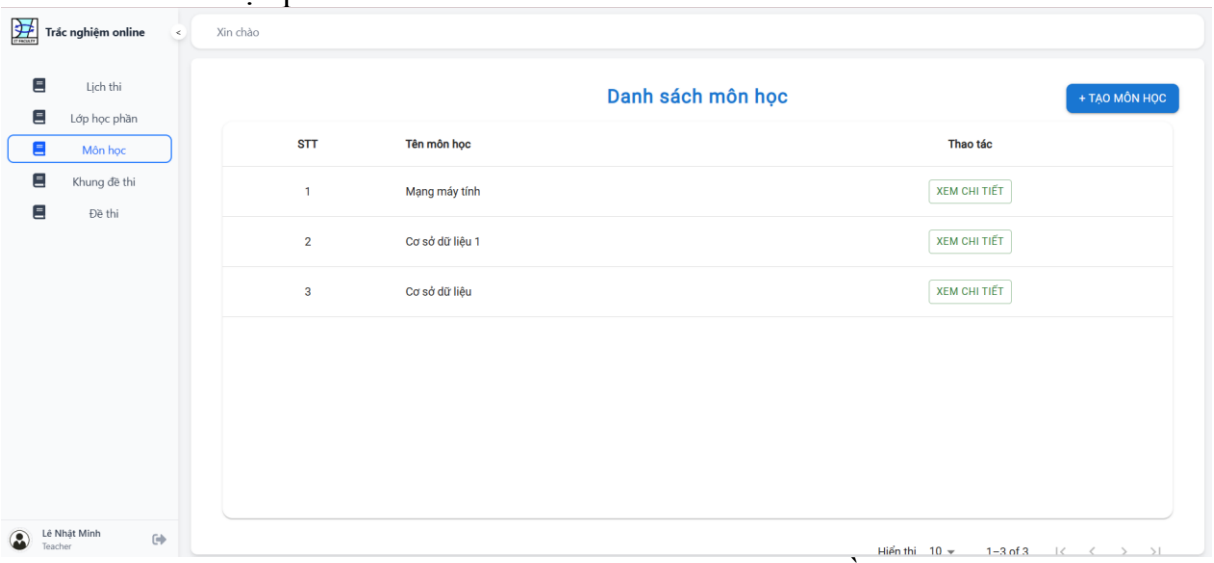
- Khóa reset mật khẩu



Hình 32: Màn hình đặt lại mật khẩu

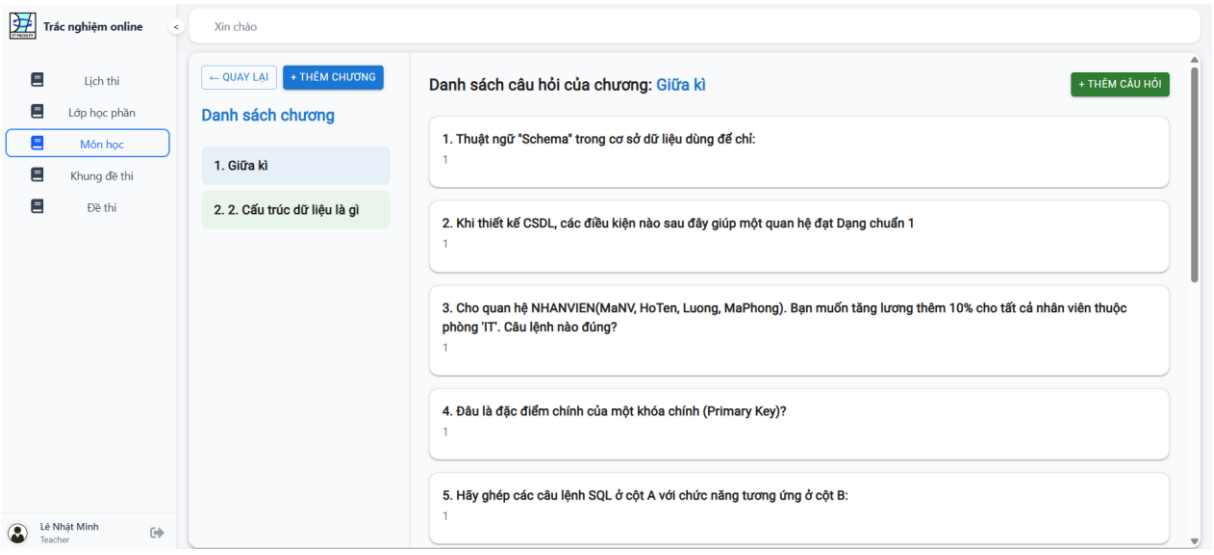
3.6.2 Giao diện đối với người dùng là giảng viên

Xem danh sách học phần



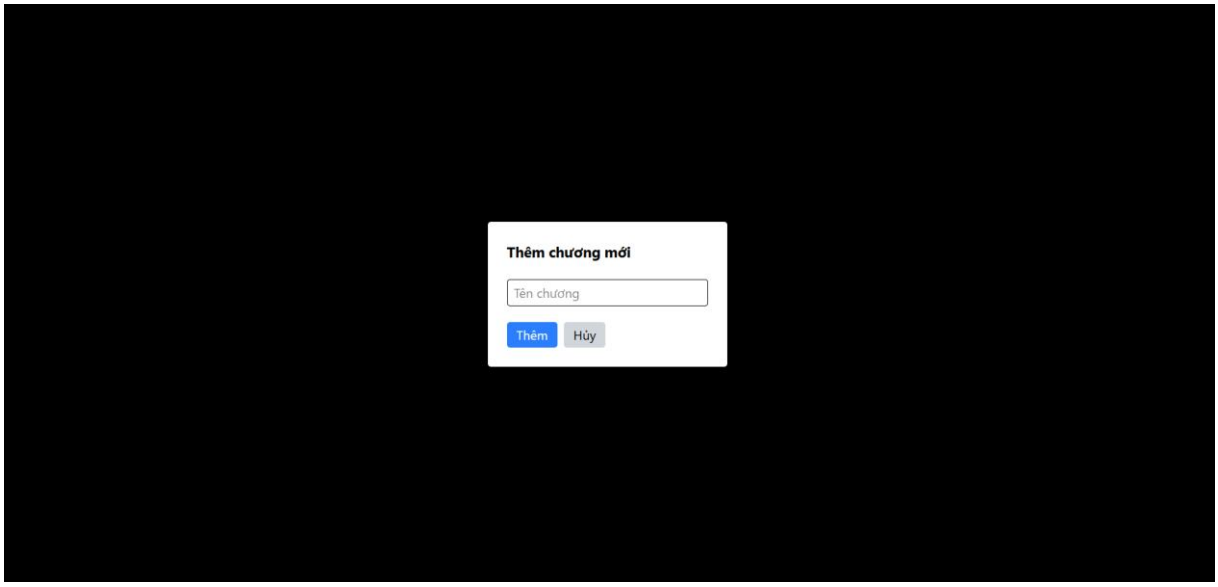
Hình 33: Màn hình danh sách học phần

- Xem chi tiết học phần



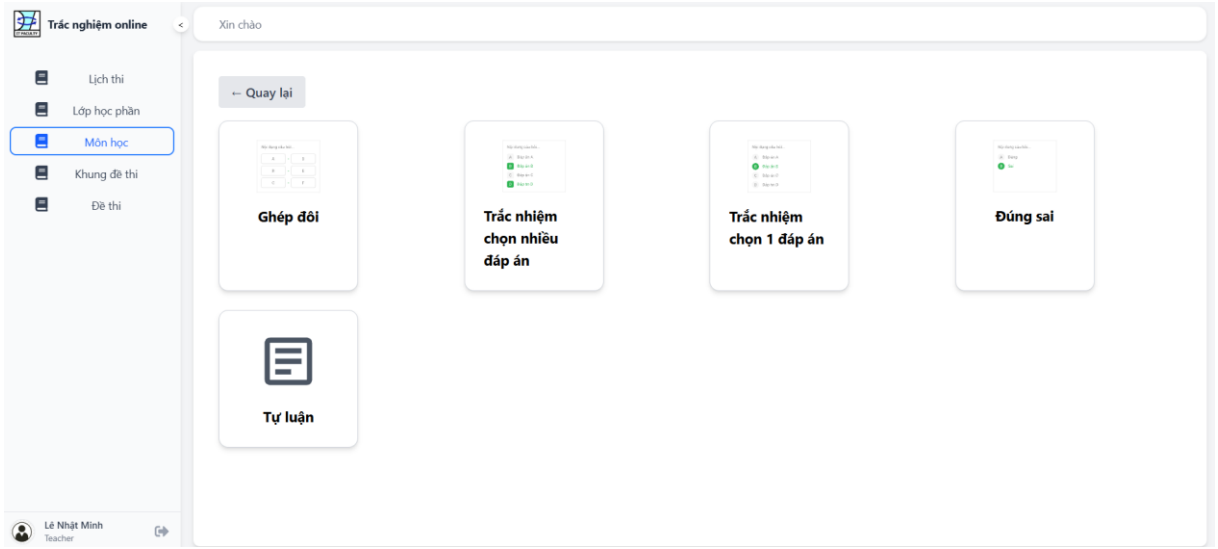
Hình 34: Màn hình chi tiết học phần

- Thêm chương



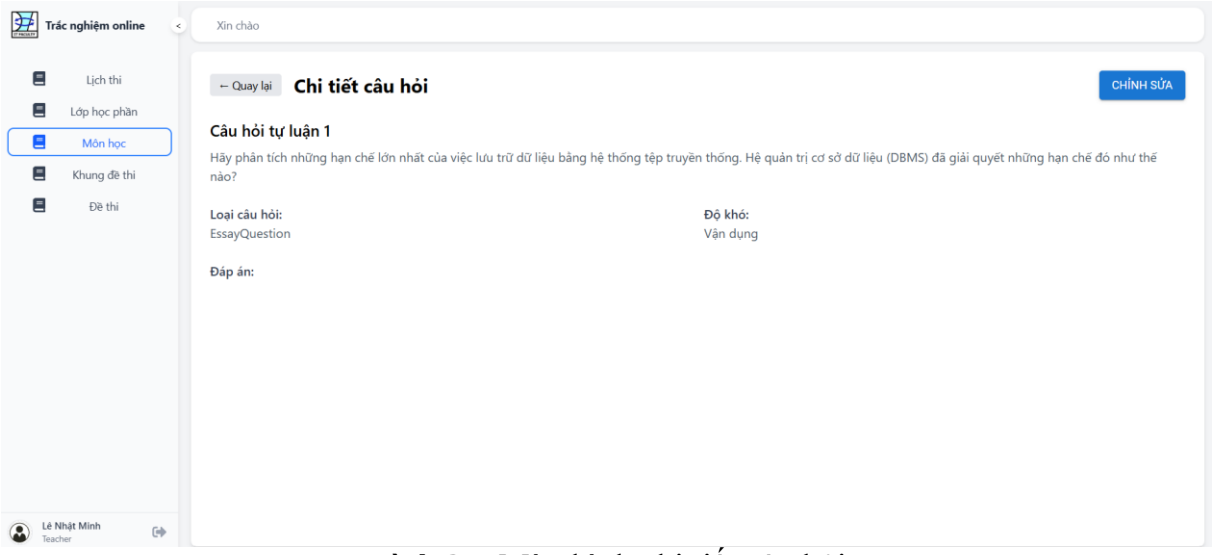
Hình 35: Màn hình thêm chương

- Thêm câu hỏi



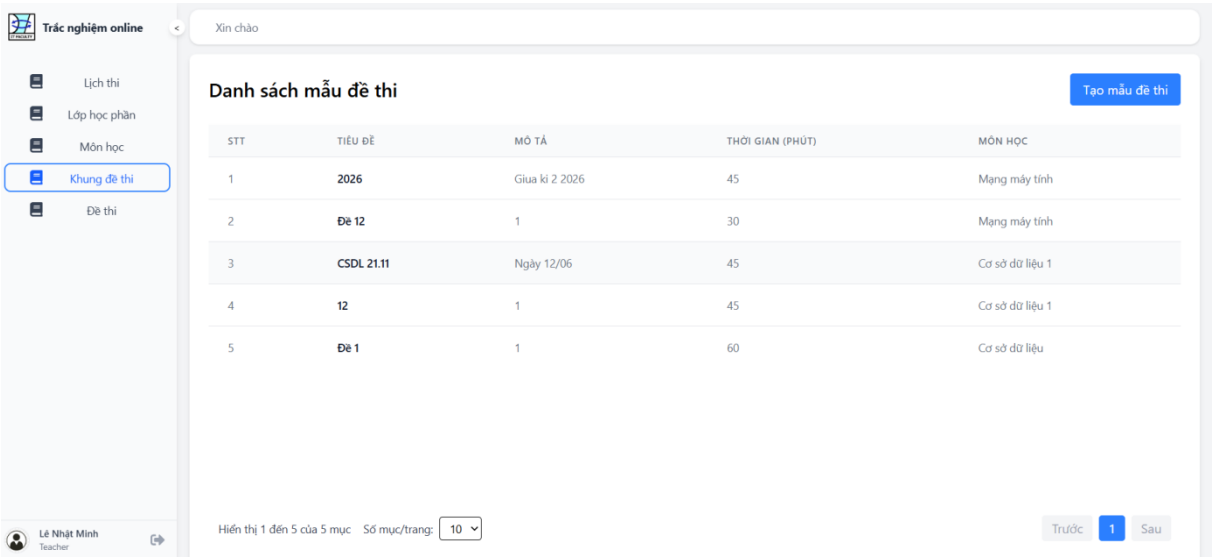
Hình 36: Màn hình thêm câu hỏi

- Chi tiết câu hỏi



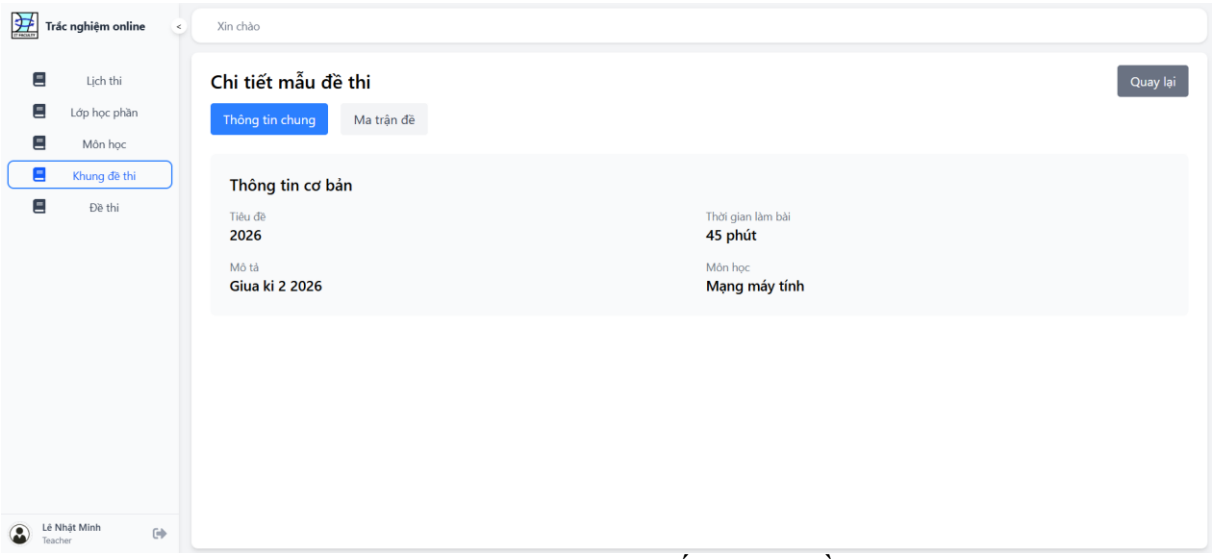
Hình 37: Màn hình chi tiết câu hỏi

- Xem khung đề thi



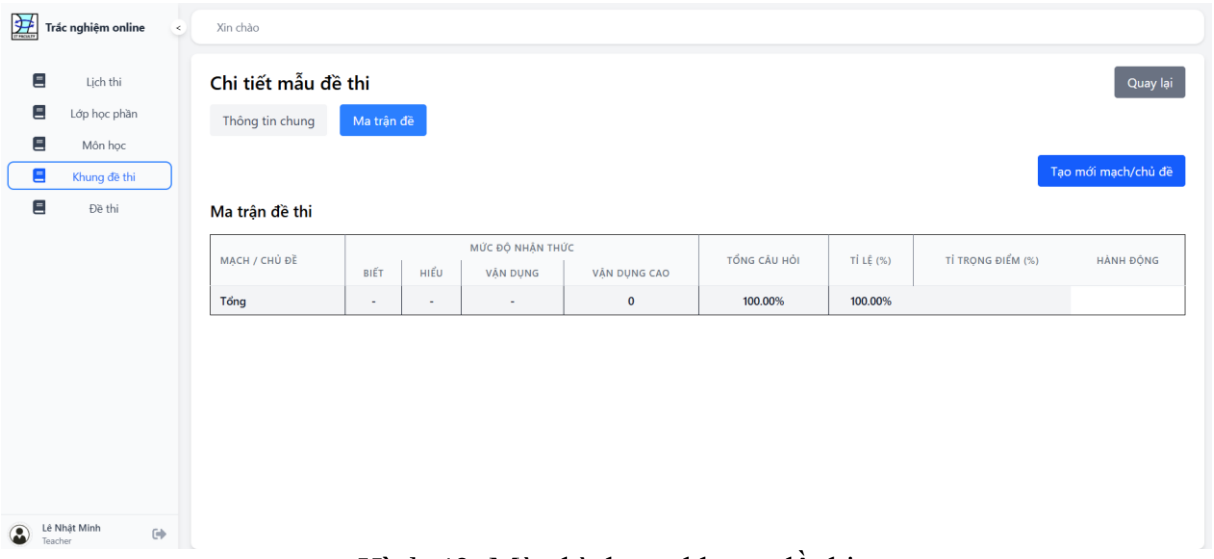
Hình 38: Màn hình khung đề thi

- Chi tiết khung đề thi



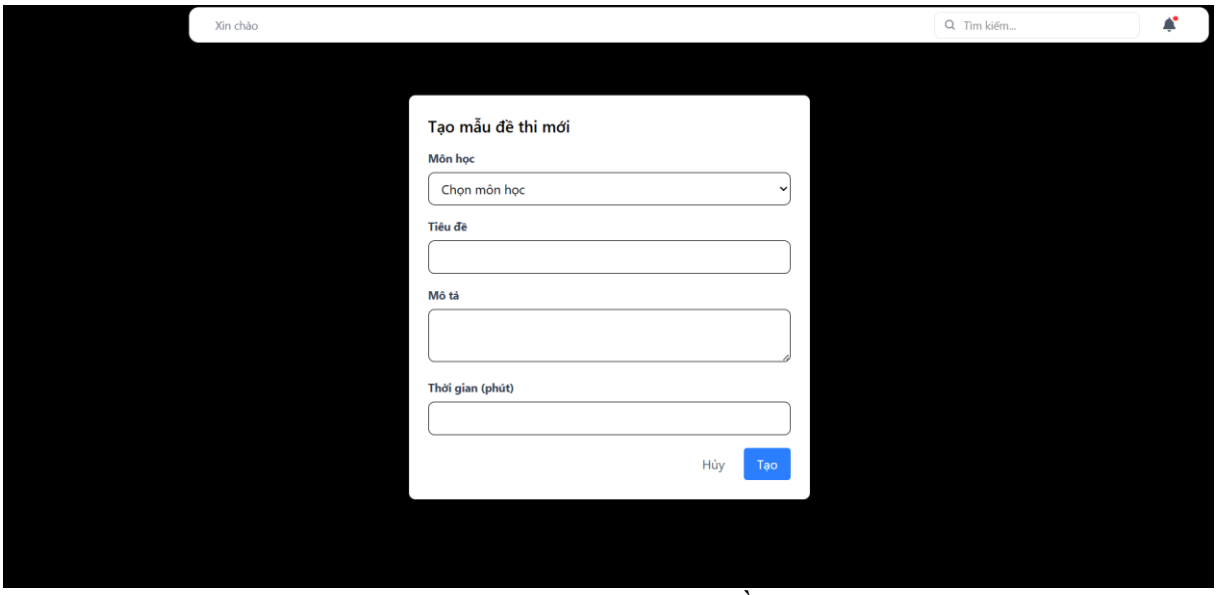
Hình 39: Màn hình chi tiết khung đề thi

- Tạo ma trận đề



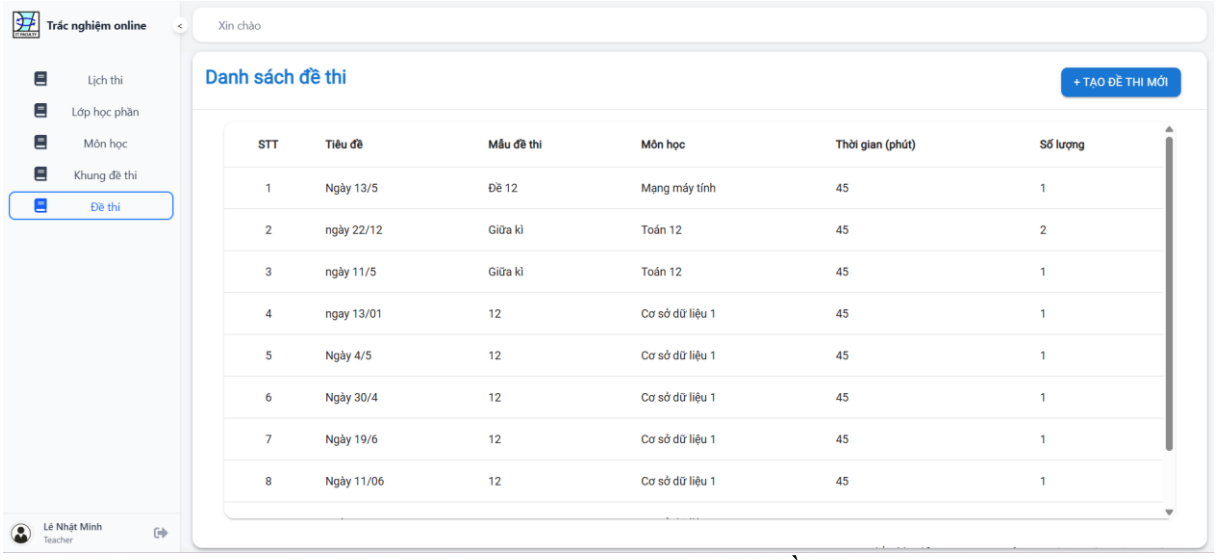
Hình 40: Màn hình tạo khung đề thi

- Tạo khung đề thi



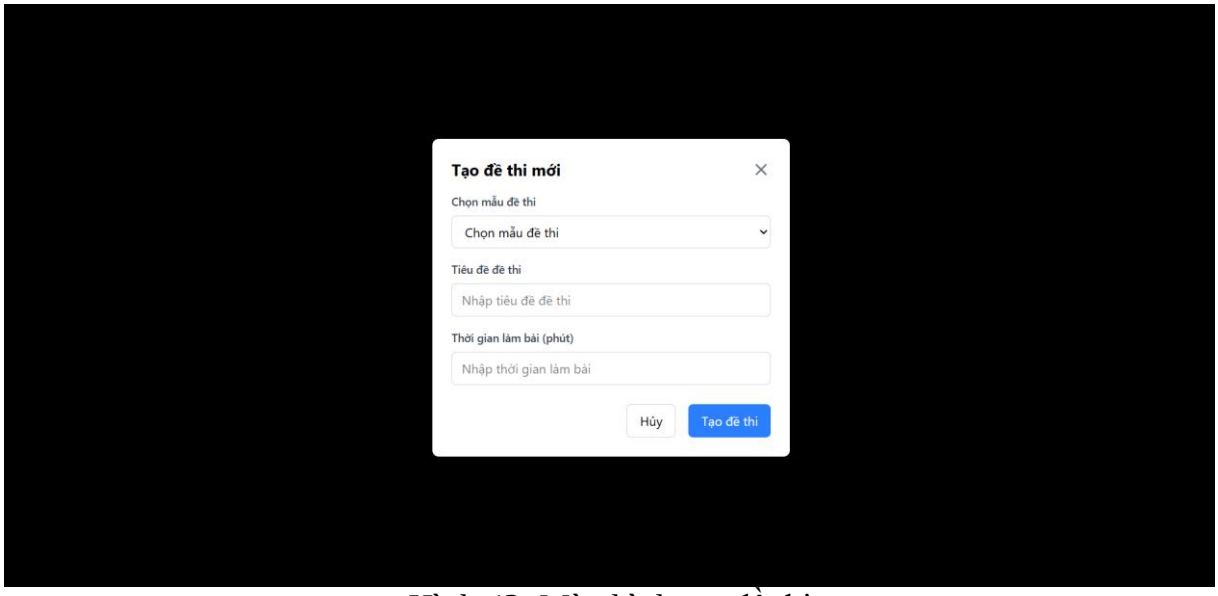
Hình 41: Màn hình tạo đề thi

- Xem danh sách đề thi



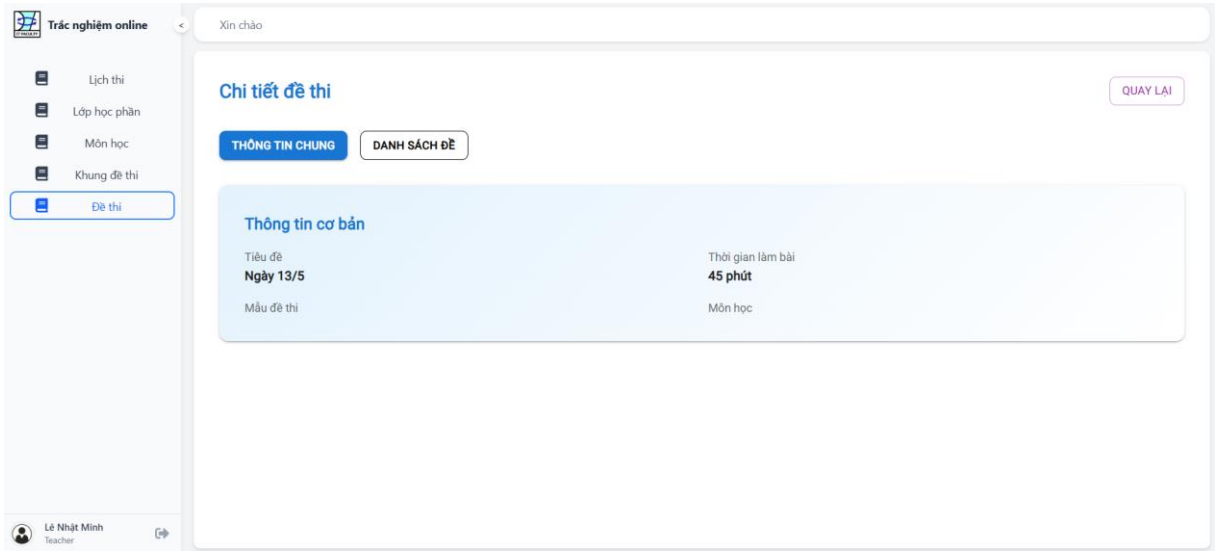
Hình 42: Màn hình danh sách đề thi

- Tạo đề thi



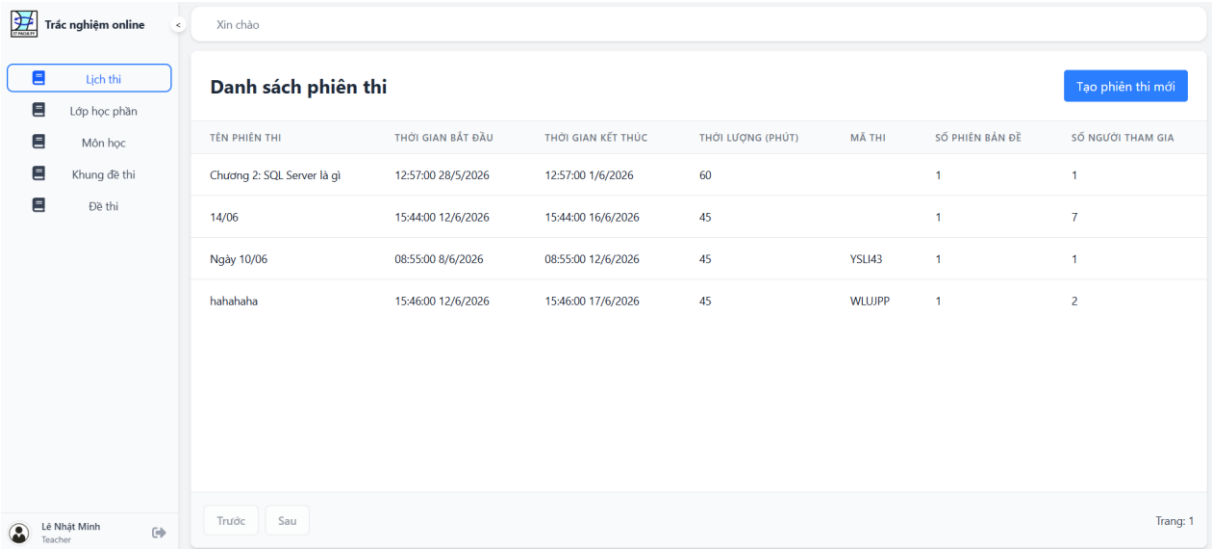
Hình 43: Màn hình tạo đề thi

- Xem chi tiết đề thi



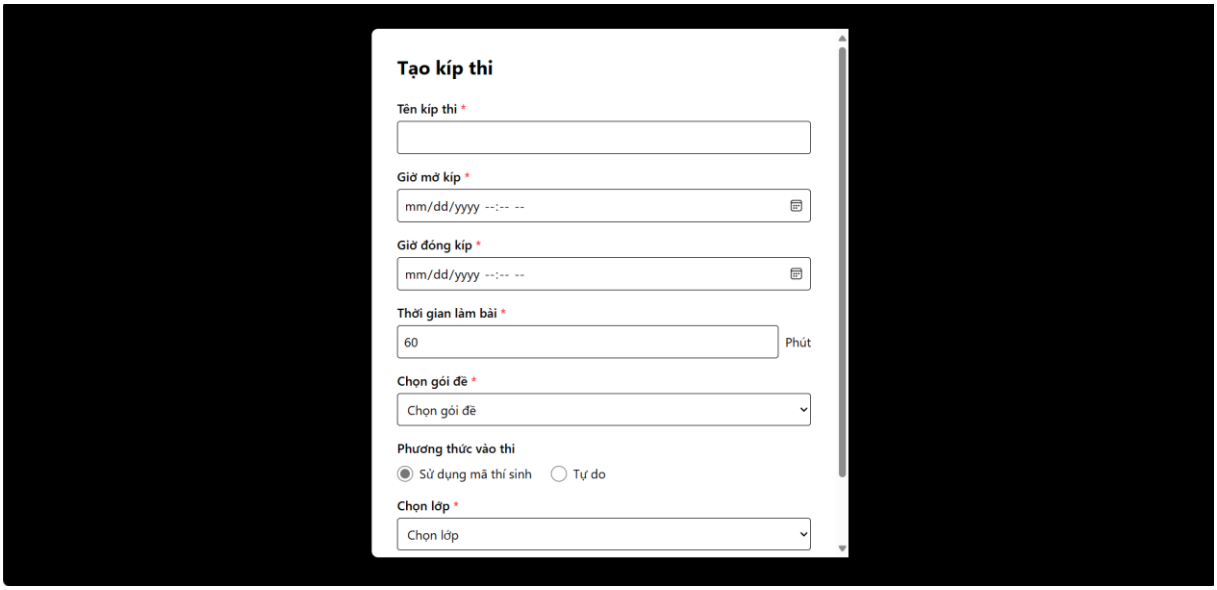
Hình 44: Màn hình chi tiết đề thi

- Xem danh sách phiên thi



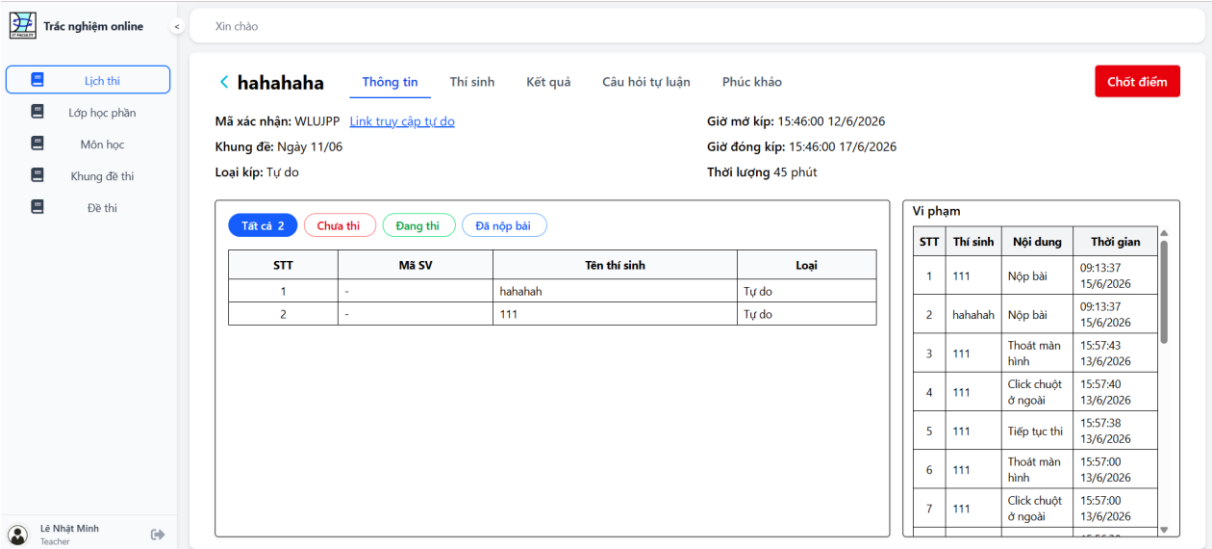
Hình 45: Màn hình danh sách phiên thi

- Tạo phiên thi mới



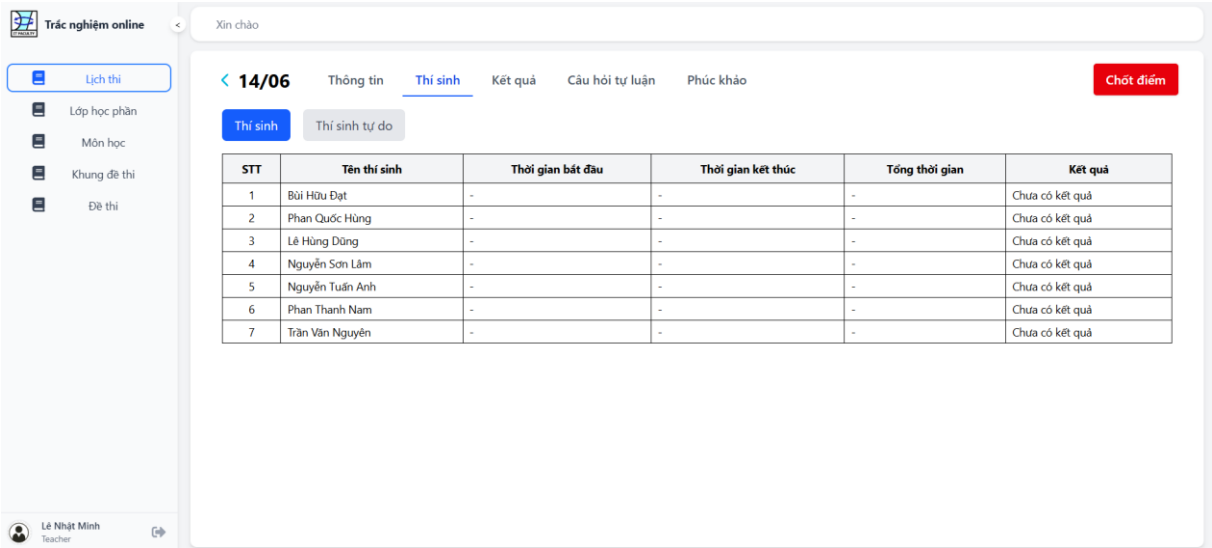
Hình 46: Màn hình tạo phiên thi mới

- Xem chi tiết phiên thi



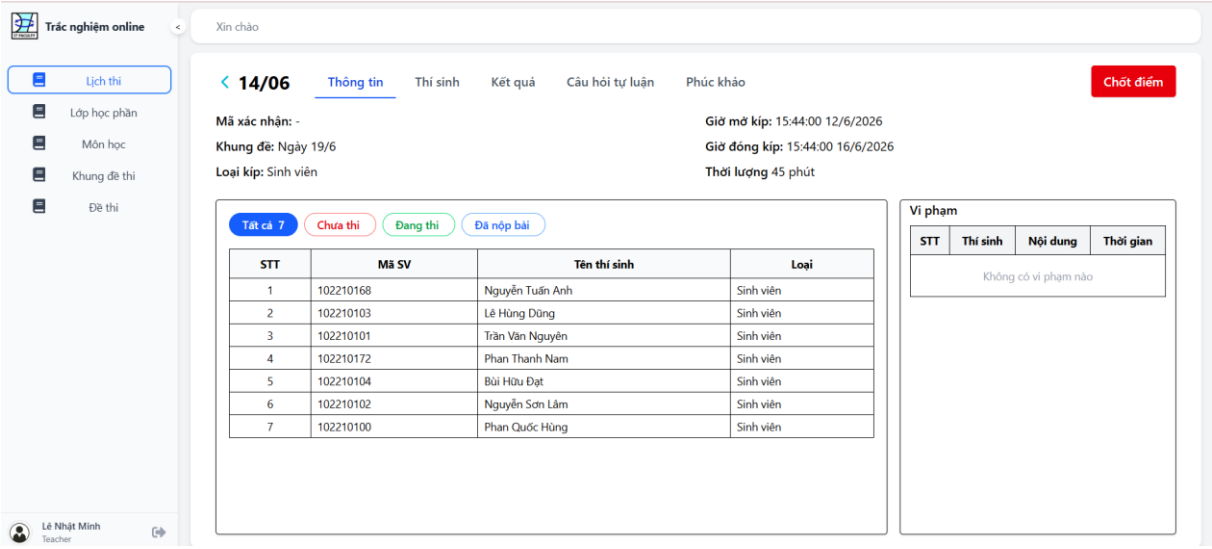
Hình 47: Màn hình chi tiết phiên thi

- Xem danh sách thí sinh



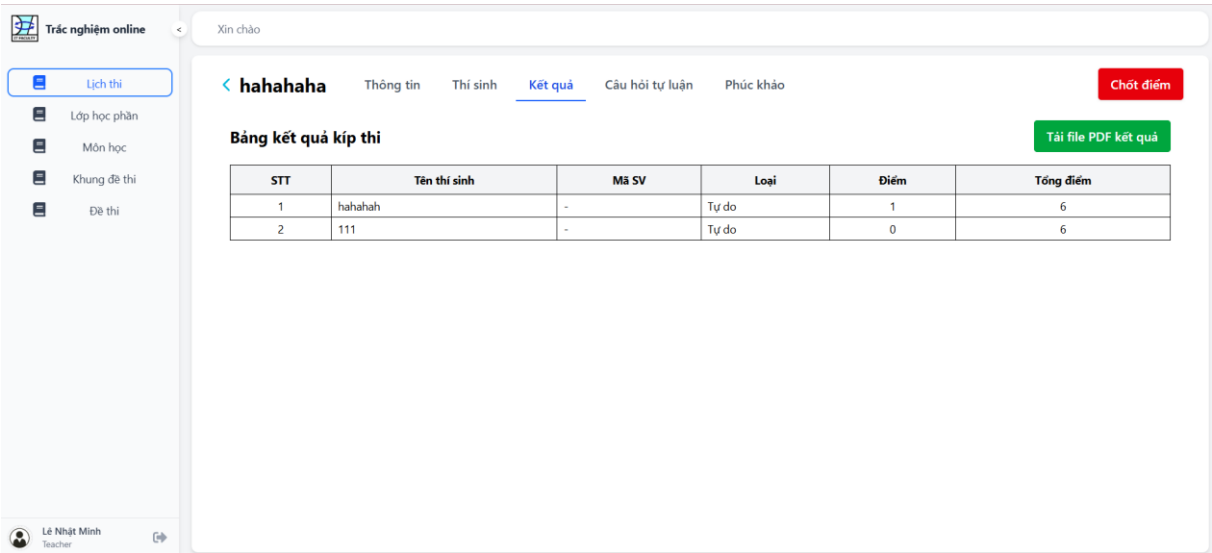
Hình 48: Màn hình danh sách thí sinh

- Giám sát thí sinh



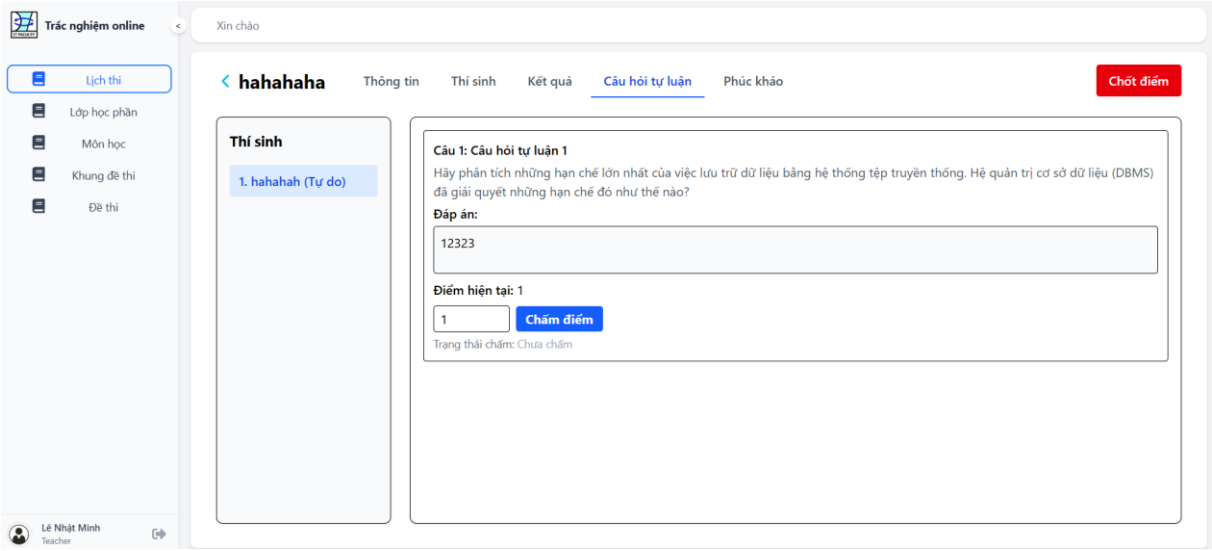
Hình 49: Màn hình giám sát thí sinh dự thi

- Kết quả thi



Hình 50: Màn hình kết quả thi

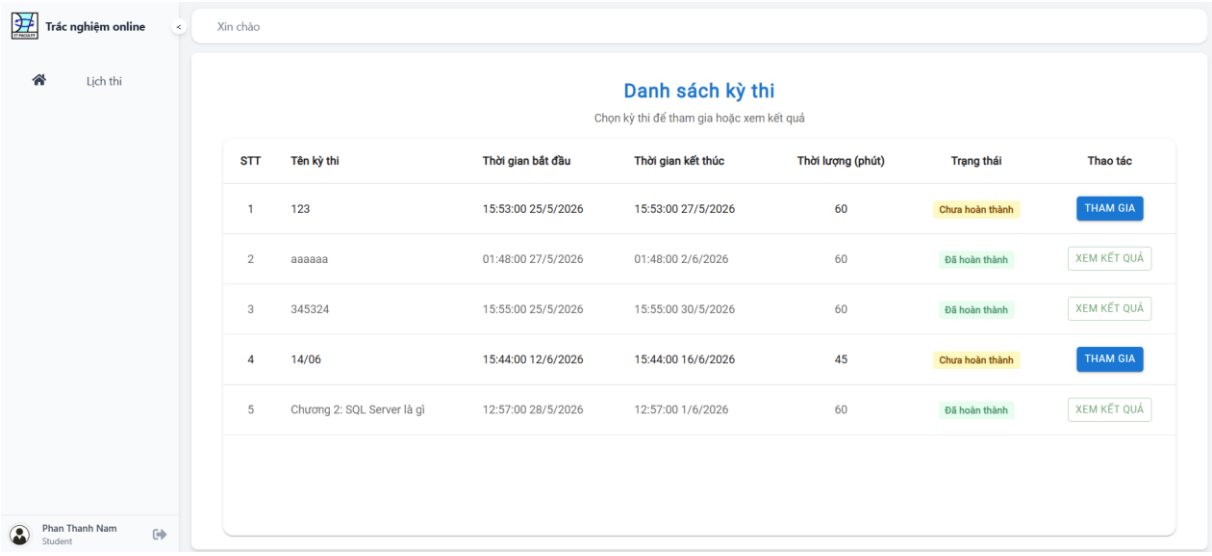
- Chấm điểm câu hỏi tự luận



Hình 51: Màn hình chấm điểm câu hỏi tự luận

3.6.3. Giao diện đối với người dùng là sinh viên

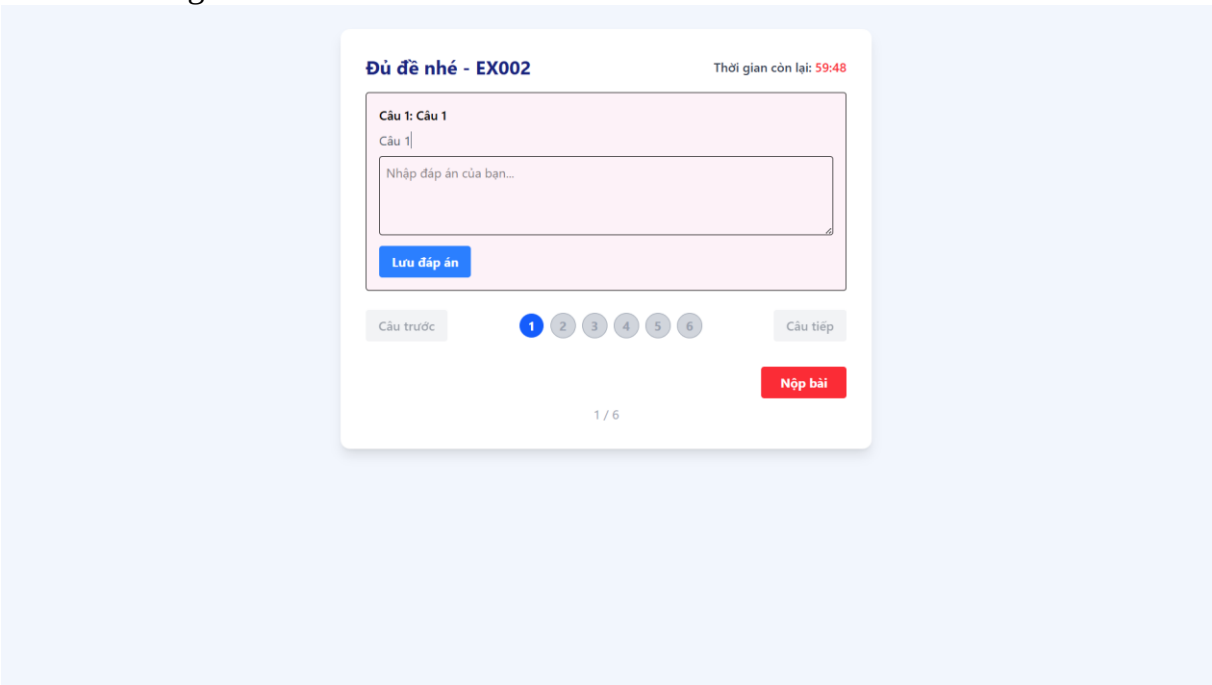
Xem lịch thi



Hình 52: Màn hình tham gia thi

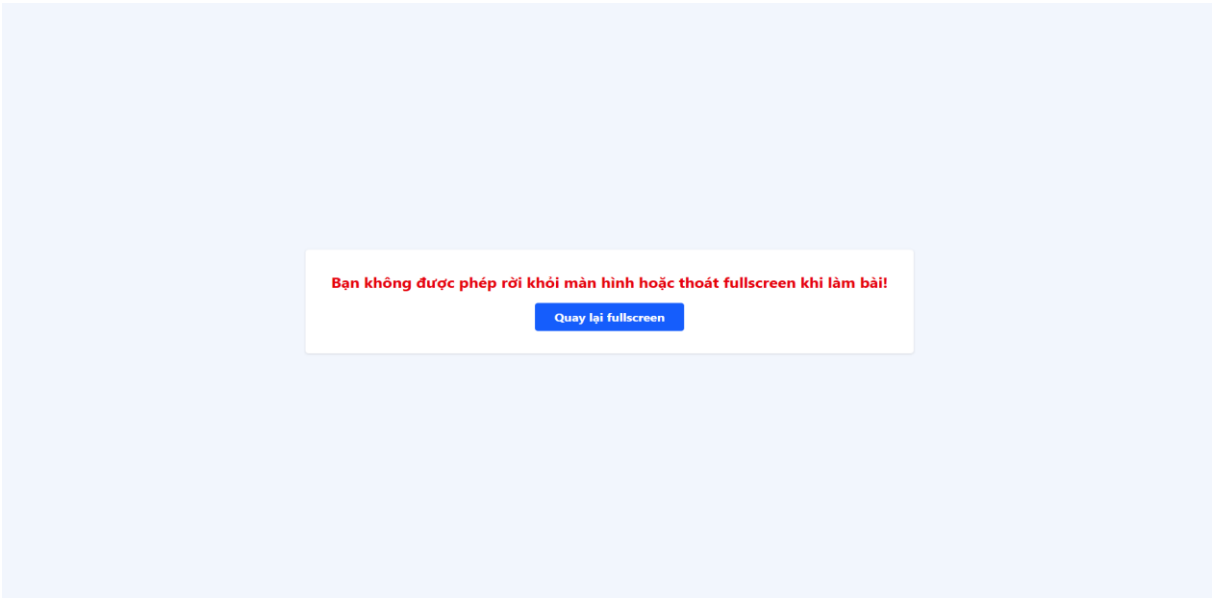
3.6.4Giao diện đối với người dùng là thí sinh tự do

- Tham gia thi



Hình 53: Màn hình cảnh báo rời khỏi màn hình

- Màn hình cảnh báo



Hình 54: Màn hình cảnh báo

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Kết luận

Qua quá trình phân tích, thiết kế và triển khai, đồ án «Xây dựng hệ thống quản lí thi trực tuyến» đã hoàn thành 100% phạm vi chức năng theo thiết kế (58/58 use case), đáp ứng nhu cầu quản lý thi trực tuyến cho bốn nhóm người dùng: quản trị viên, giảng viên, sinh viên và thí sinh tự do.

4.1.1. Kết quả đạt được

Bảng tóm tắt kết quả đạt được

Chỉ số	**Kết quả**
Use case hoàn thành	58/58 (100%): 12 UC quản trị viên, 28 UC giảng viên, 10 UC sinh viên, 8 UC thí sinh tự do
API / CQRS	106 endpoint; 65 Command; 45 Query
Loại câu hỏi	5 (một đáp án, nhiều đáp án, đúng/sai, ghép nối, tự luận)
Unit test	7/7 pass (Application.Tests)
Dữ liệu thử nghiệm	62 câu hỏi; 12 đề thi; 10 ca thi
Hiệu năng (dev local)	API TB ~6.0 ms; sinh 3 mã đề weighted TB ~123 ms
Kiểm thử người dùng (UAT)	10 người: 1 quản trị viên, 2 giảng viên, 5 sinh viên, 2 thí sinh tự do

Bảng 35: Tóm tắt kết quả đạt được

4.1.2. Kiểm thử và đánh giá thực tế

- Hệ thống được kiểm thử chức năng trên môi trường local (Swagger + React SPA), với 10 ca thi mô phỏng và 10 người dùng thử nghiệm tham gia UAT. Các luồng chính — quản lý ngân hàng câu hỏi, sinh đề weighted random, tổ chức ca thi, làm bài có giám sát anti-cheat (fullscreen, ghi log hành vi) và chấm điểm tự luận — đều vận hành ổn định trong phạm vi thử nghiệm.

4.1.3. Bài học kinh nghiệm (Lessons learned)

- CQRS/MediatR tách rõ luồng đọc–ghi, giúp API mỏng và dễ bảo trì khi số lượng use case tăng.
- DDD (Aggregate, Value Object) cho phép validate nghiệp vụ sớm và unit test domain độc lập với infrastructure.

- Weighted random sinh đề cân bằng chất lượng câu hỏi và thời gian xử lý (endpoint sinh đề ~123 ms cho 3 mã đề); random thuần nhanh hơn nhưng không đáp ứng ràng buộc phân bố đề.
- SignalR kết hợp Hangfire phù hợp cho giám sát ca thi realtime và tự động nộp bài server-side, giảm phụ thuộc đồng hồ client.
- Cần bổ sung load test đồng thời (k6/JMeter) và tối ưu responsive mobile trước khi triển khai production.

4.1.4. Hạn chế

- Một số giao diện chưa tối ưu cho thiết bị di động; module quản trị viên cần bổ sung báo cáo thống kê tổng hợp. Hệ thống mới kiểm thử trên môi trường local, chưa triển khai Docker/production và chưa có load test đồng thời.

4.2. Hướng phát triển

- Về mặt thực tiễn ứng dụng, hệ thống có thể mở rộng theo các hướng sau:

4.2.1. Tích hợp AI

- Áp dụng AI hoặc NLP để gợi ý cho giảng viên các câu hỏi có thể tạo dựa trên tài liệu mà giảng viên đưa ra.
- Chấm điểm câu hỏi tự luận bằng AI: hệ thống hỗ trợ chấm điểm sơ bộ, đề xuất điểm cho giảng viên xác nhận.

4.2.2. Mở rộng ứng dụng di động

- Xây dựng ứng dụng mobile (React Native hoặc PWA) để sinh viên thi và giảng viên giám sát thuận tiện hơn.

TÀI LIỆU THAM KHẢO

[1] V. Hoang, "Axios là gì? Cập nhật mới nhất về thư viện Axios," [Trực tuyến]. Có ı tại: *https://200lab.io/blog/axios-la-gi/.*

[2] Lê Nhật Thanh, "Trải nghiệm chuyên sâu Domain Driven Design" [Trực ến]. Có sẵn tại: *https://tech.cybozu.vn/trai-nghiem-chuyen-sau-domain-driven-ign-6eef4/*

[3] tuananhhedspibk, "CQRS Design Pattern trong kiến thúc Microservices " [Trực ến]. Có sẵn tại: *https:// viblo.asia/p/cqrs-design-pattern-trong-kien-truc-croservices-PAoJexlZ41j*

[4] thinhnotes, "Use Case Diagram và 5 sai lầm thường gặp," [Trực tuyến]. Có sẵn *https://thinhnotes.com/chuyen-nghe-ba/use-case-diagram-va-5-sai-lam-thuong-o/.*

[5] "Activity Diagram: Bản vẽ hoạt động," 25 June 2014. [Trực tuyến]. Có sẵn tại: *ps://iviettech.vn/blog/1066-activity-diagram-ban-ve-hoat-dong.html. [Đã truy ɔ 11 May 2024].*

[6] H. V. Trinh, "Phân tích thiết kế hệ thống thông tin sử dụng biểu đồ UML (Phần " 2 May 2016. [Trực tuyến]. Có sẵn tại: *https://viblo.asia/p/phan-tich-thiet-ke-thong-thong-tin-su-dung-bieu-do-uml-phan-2-0bDM6wpAG2X4. [Đã truy cập April 2024].*

PHỤ LỤC

Phụ lục A. Liên kết mã nguồn trên GitHub

Đường dẫn repository: https://github.com/banhdiduong/totnghiepfinal

Tên repository: banhdiduong/totnghiepfinal

Trạng thái: Public

Nhánh chính: main

Cấu trúc thư mục chính trong repository:

Capstone.Server — Backend ASP.NET Core (.NET 8), kiến trúc CQRS/DDD.

Capstone.Web/capstone — Frontend React (Vite).

Data / DataTest — Dữ liệu CSV mẫu phục vụ import và kiểm thử.

README.md — Hướng dẫn cài đặt và chạy hệ thống trên môi trường phát triển.

Phụ lục B. Hướng dẫn truy cập và chạy mã nguồn

Yêu cầu môi trường:

.NET 8 SDK

Node.js 18 trở lên

SQL Server LocalDB (hoặc SQL Server)

Redis (tùy chọn, theo cấu hình ConnectionStrings:Redis)

Chạy backend:

```
cd Capstone.Server/src/Capstone.API
dotnet restore
dotnet run
```

Chạy frontend:

```
cd Capstone.Web/capstone
npm install
npm run dev
```