

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
Khoa Khoa học và Kỹ thuật máy tính

CÔNG NGHỆ PHẦN MỀM (CO3001)
BÁO CÁO BÀI TẬP LỚN - BÀI 3



ĐỀ TÀI:
HỆ THỐNG HỖ TRỢ TUTOR TẠI
TRƯỜNG ĐẠI HỌC BÁCH KHOA – ĐHQG TP.HCM

Lớp L01 - Nhóm CESE - HK251
Giáo viên hướng dẫn: ThS. Lê Đình Thuận

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 11 - 2025

Mục lục

1 Giới thiệu	7
1.1 Đề tài	7
1.2 Stakeholders	8
1.3 User stories	8
2 Các yêu cầu hệ thống	10
2.1 Yêu cầu chức năng (Functional Requirements)	10
2.2 Yêu cầu phi chức năng (Non-functional requirements)	12
3 Use-case diagrams và use-case scenario	13
3.1 Sơ đồ tổng hợp chương trình Tutor	13
3.2 Quản lý truy cập và xác thực	14
3.3 Đăng ký chương trình	22
3.4 Tạo chương trình học	26
3.5 Thiết lập lịch trình	32
3.6 Đánh giá	36
3.7 Phân tích & báo cáo	40
3.8 Thông báo và nhắn tin	45
3.9 Quản lý tài liệu	49
4 Activity Diagram	51
4.1 Quản lý truy cập và xác thực	51
4.2 Đăng ký chương trình	57
4.3 Tạo chương trình học	61
4.4 Thiết lập lịch trình cho sinh viên	67
4.5 Đánh giá	72
4.6 Phân tích và báo cáo	74
4.7 Thông báo và nhắn tin	78
4.8 Quản lý tài liệu	83
5 Sequence Diagrams	86
5.1 Quản lý truy cập và xác thực	86
5.2 Đăng ký chương trình	95
5.3 Tạo chương trình học	98
5.4 Thiết lập lịch trình cho sinh viên	104
5.5 Đánh giá	110
5.6 Phân tích và báo cáo	114
5.7 Thông báo và nhắn tin	116
5.8 Quản lý tài liệu	121

6 Giao diện người dùng	125
6.1 Giao diện hệ thống của Admin	125
6.2 Giao diện người dùng của Tutor	130
6.3 Giao diện người dùng của Sinh viên	139
6.4 Liên kết kiểm thử giao diện người dùng	150
7 Class Diagram	151
7.1 Quản lý truy cập và xác thực	152
7.2 Đăng ký chương trình	155
7.3 Tạo chương trình học	157
7.4 Thiết lập lịch trình cho sinh viên	161
7.5 Đánh giá	164
7.6 Phân tích và báo cáo	165
7.7 Thông báo và nhắn tin	166
7.8 Quản lý tài liệu	169
8 Development view	171
8.1 Quản lý truy cập và xác thực	171
8.2 Đăng ký chương trình	173
8.3 Tạo chương trình học	175
8.4 Thiết lập lịch trình cho sinh viên	177
8.5 Đánh giá	180
8.6 Phân tích và báo cáo	181
8.7 Thông báo và nhắn tin	182
8.8 Quản lý tài liệu	184
9 Deployment View của hệ thống Turto	186
9.1 Thiết bị Người dùng (Client Device)	186
9.2 Kho mã nguồn GitLab và CI/CD	186
9.3 Nền tảng Render.com (Môi trường Production)	187
9.3.1 Render Static Site (Frontend)	187
9.3.2 Render Reverse Proxy	187
9.3.3 Render Web Service (Backend)	187
9.4 Các Dịch vụ Bên ngoài (External Services)	187
9.4.1 MongoDB Atlas	187
9.4.2 Dịch vụ SMTP	187
9.4.3 Google OAuth 2.0	188
9.5 Môi trường Phát triển Cục bộ (Local Development)	188
9.5.1 Frontend	188
9.5.2 Backend	188
10 Sử dụng Generative AI	189
10.1 Sử dụng AI trong phát triển use-case "Thông báo và nhắn tin"	189





Revision History

Revision	Date	Author	Description
1.0	9/2025	Trần Đăng Khoa	Use-case diagram + use-case scenario: Quản lý xác thực và đăng nhập
1.0	9/2025	Nguyễn Lê Khôi Nguyên	Use-case diagram + use-case scenario: Đăng ký chương trình học
1.0	9/2025	Nguyễn Đức Dũng	Use-case diagram + use-case scenario: Tạo chương trình học
1.0	9/2025	Nguyễn Hà Viết Thống	Use-case diagram + use-case scenario: Thiết lập lịch trình cho sinh viên
1.0	9/2025	Lương Trí Thịnh	Use-case diagram + use-case scenario: Đánh giá
1.0	9/2025	Võ Đức Cao Thượng	Use-case diagram + use-case scenario: Phân tích và báo cáo
1.0	9/2025	Huỳnh Trung Kiên	User-stories + Use-case diagram hệ thống
1.0	9/2025	Vũ Anh Tuấn	Yêu cầu phi chức năng
2.0	10/2025	Huỳnh Trung Kiên	Use-case diagram + use-case scenario + Activity diagram + Sequence diagram + User Interface: Thông báo và nhắn tin
2.0	10/2025	Vũ Anh Tuấn	Use-case diagram + use-case scenario + Activity diagram + Sequence diagram + User Interface: Quản lý tài liệu
2.0	10/2025	Trần Đăng Khoa	Activity diagram + Sequence diagram + User Interface: Quản lý xác thực và truy cập Giao diện người dùng tổng quan
2.0	10/2025	Nguyễn Lê Khôi Nguyên	Activity diagram + Sequence diagram + User Interface: Đăng ký chương trình học
2.0	10/2025	Nguyễn Đức Dũng	Cập nhật sơ đồ use-case và Use-case scenario Activity diagram + Sequence diagram + User Interface: Tạo chương trình học
2.0	10/2025	Nguyễn Hà Viết Thống	Activity diagram + Sequence diagram + User Interface: Thiết lập lịch trình cho sinh viên
2.0	10/2025	Lương Trí Thịnh	Activity diagram + Sequence diagram + User Interface: Đánh giá
2.0	10/2025	Võ Đức Cao Thượng	Activity diagram + Sequence diagram + User Interface: Phân tích và báo cáo



3.0	10/2025	Huỳnh Trung Kiên	Activity diagram description + Sequence diagram description + Class diagram + Component diagram: Thông báo và nhắn tin
2.0	10/2025	Vũ Anh Tuấn	Activity diagram description + Sequence diagram description + Class diagram + Component diagram: Quản lý tài liệu
2.0	10/2025	Trần Đăng Khoa	Activity diagram description and fix diagram + Sequence diagram description + Class diagram + Component diagram: Quản lý xác thực và truy cập
2.0	10/2025	Nguyễn Lê Khôi Nguyễn	Activity diagram description and fix diagram + Sequence diagram description + Class diagram + Component diagram: Đăng ký chương trình học
2.0	10/2025	Nguyễn Đức Dũng	Activity diagram description and fix diagram + Sequence diagram description + Class diagram + Component diagram: Tạo chương trình học
2.0	10/2025	Nguyễn Hà Việt Thống	Activity diagram description + Sequence diagram description + Class diagram + Component diagram: Thiết lập lịch trình cho sinh viên
2.0	10/2025	Lương Trí Thịnh	Activity diagram description + Sequence diagram description + Class diagram + Component diagram: Đánh giá
2.0	10/2025	Võ Đức Cao Thượng	Activity diagram description + Sequence diagram description + Class diagram + Component diagram: Phân tích và báo cáo

1 Giới thiệu

1.1 Đề tài

Tại Trường Đại học Bách Khoa – ĐHQG TP.HCM (HCMUT), chương trình Tutor/Mentor được triển khai nhằm hỗ trợ sinh viên trong quá trình học tập và phát triển kỹ năng. Các Tutor có thể là giảng viên, nghiên cứu sinh, hoặc sinh viên năm trên có thành tích học tập tốt, được phân công để hướng dẫn và đồng hành cùng một nhóm sinh viên cụ thể. Nhà trường mong muốn xây dựng một hệ thống phần mềm để quản lý và vận hành chương trình Tutor một cách hiệu quả, hiện đại và có khả năng mở rộng, đáp ứng nhu cầu thực tiễn trong môi trường giáo dục đại học.

Hệ thống cần cho phép quản lý thông tin tutor và sinh viên (hồ sơ cá nhân, lĩnh vực chuyên môn, nhu cầu hỗ trợ), hỗ trợ sinh viên đăng ký tham gia chương trình, lựa chọn hoặc được gợi ý tutor phù hợp. Tutor có thể thiết lập lịch rảnh, mở các buổi tư vấn, và quản lý các buổi gặp gỡ trực tiếp hoặc trực tuyến. Bên cạnh đó, hệ thống hỗ trợ đặt lịch, hủy/đổi lịch, gửi thông báo tự động, nhắc lịch và tổng hợp biên bản buổi gặp (nếu cần). Song song, hệ thống cung cấp công cụ phản hồi và đánh giá: sinh viên phản hồi chất lượng buổi học, tutor theo dõi và ghi nhận tiến bộ của người được kèm, trong khi khoa/bộ môn có thể khai thác dữ liệu đánh giá để nắm tình hình học tập của sinh viên ở các môn cụ thể, phòng Đào tạo sử dụng báo cáo tổng quan nhằm tối ưu phân bổ nguồn lực, và phòng Công tác Sinh viên có thể căn cứ vào kết quả tham gia để cộng điểm rèn luyện hoặc xét học bổng.

Tích hợp hạ tầng công nghệ của HCMUT. Để bảo đảm an toàn và đồng bộ, hệ thống phải tích hợp với dịch vụ xác thực tập trung HCMUT_SSO nhằm quản lý đăng nhập thống nhất cho sinh viên, giảng viên và cán bộ. Dữ liệu cá nhân cơ bản (họ tên, MSSV/ Mã cán bộ, khoa/chuyên ngành, email học vụ, trạng thái học tập/giảng dạy...) được đồng bộ từ HCMUT_DATACORE thông qua dịch vụ chia sẻ dữ liệu, bảo đảm tính chính xác, nhất quán và giảm thiểu thao tác nhập liệu thủ công. Việc phân quyền (sinh viên/tutor/điều phối viên/chủ nhiệm bộ môn/ban quản lý) được ràng buộc theo thông tin vai trò lấy từ hệ thống tập trung của trường. Đồng thời, hệ thống cũng cần kết nối với HCMUT_LIBRARY để cho phép sinh viên và tutor truy cập, chia sẻ tài liệu, sách, và giáo trình liên quan đến buổi học; từ đó tăng tính hỗ trợ học tập, bảo đảm nguồn học liệu chính thống và đồng bộ với cơ sở dữ liệu tài nguyên học tập của toàn trường.

Ngoài các tính năng cốt lõi, hệ thống cũng có thể được mở rộng với các chức năng nâng cao, triển khai tùy theo nhu cầu và nguồn lực:

- Ghép cặp tutor – sinh viên thông minh (tích hợp AI): hệ thống sử dụng kỹ thuật AI để gợi ý ghép cặp.
- Cộng đồng trực tuyến cho tutor – mentee.
- Chương trình tutor học thuật và phi học thuật.
- Hỗ trợ học tập cá nhân hóa (tích hợp AI).

1.2 Stakeholders

- Student (sinh viên): người học, chủ yếu sử dụng hệ thống để truy cập tài liệu, tham gia các buổi học và tương tác với Tutor.
- Tutor/mentor (giảng viên, nghiên cứu sinh,...): người hướng dẫn, cung cấp nội dung học tập, tổ chức các buổi học và hỗ trợ sinh viên.
- The Office of Academic Affairs (Phòng Đào tạo): một cơ quan của nhà trường chịu trách nhiệm phân bổ nguồn lực tutor cho phù hợp với nhu cầu thực tế của sinh viên.
- The Office of Student Affairs (Phòng công tác sinh viên): một cơ quan của nhà trường có chức năng đánh giá thành tích của sinh viên để cộng điểm rèn luyện, cấp học bổng.
- Faculty/Department (Khoa/Bộ môn): Các khoa của trường đại học Bách Khoa có chức năng quản lý, đánh giá thành tích học tập của sinh viên trong nội bộ Khoa.
- Administrator (Quản trị viên): có vai trò phân quyền, quản lý tài khoản người dùng.

1.3 User stories

- Là một người dùng, tôi muốn đăng nhập vào hệ thống để có thể truy cập các chức năng tương ứng với vai trò của mình.
- Là một người dùng đã quên mật khẩu, tôi muốn đặt lại mật khẩu thông qua email của mình để có thể lấy lại quyền truy cập vào tài khoản.
- Là một người dùng, tôi muốn đăng xuất khỏi hệ thống một cách an toàn để bảo vệ tài khoản của mình trên các thiết bị dùng chung.
- Là một người dùng, tôi muốn hệ thống có chức năng ghi nhớ đăng nhập.
- Là một Sinh viên, tôi muốn đăng nhập bằng tài khoản của trường Đại học Bách Khoa.
- Là một Sinh viên, tôi muốn có thanh điều hướng đến các chức năng của hệ thống.
- Là một Sinh viên, tôi muốn có chức năng tìm kiếm và sàng lọc tutor/mentor phù hợp.
- Là một Sinh viên, tôi muốn xem thông tin nghiệp vụ của các tutor/mentor.
- Là một Sinh viên, tôi muốn có chức năng đánh giá và bình luận về khóa học của các tutor/mentor.
- Là một Sinh viên, tôi muốn có chức năng trao đổi trực tiếp với tutor/mentor qua hệ thống message.
- Là một Tutor, tôi muốn hệ thống quản lý lịch trình trực quan, dễ sử dụng.
- Là một Tutor, tôi muốn hệ thống thông báo mỗi khi đến lịch trình đã được đặt ra.
- Là một Tutor, tôi muốn xem chi tiết thông tin học vụ và năng lực của học viên được tích hợp từ MyBK.
- Là một Tutor, tôi muốn có nguồn tài liệu phù hợp từ thư viện trường được tích hợp vào hệ thống.



- Là Phòng Đào tạo, tôi muốn báo cáo được tổng hợp một cách trực quan, dễ đánh giá nhất.
- Là Phòng Công tác Sinh viên, tôi muốn báo cáo về sinh viên tham gia chương trình được tổng hợp một cách cụ thể, dễ phân loại nhất.
- Là một Quản trị viên, tôi muốn gán và thay đổi vai trò của người dùng để đảm bảo họ có quyền truy cập phù hợp với nhiệm vụ của mình ngoài ra tôi có thể quản lý thông tin của người dùng.



2 Các yêu cầu hệ thống

2.1 Yêu cầu chức năng (Functional Requirements)

Quản lý truy cập và xác thực:

- Đăng ký tài khoản:** Hệ thống phải cho phép người dùng (ví dụ: Sinh viên/Tutor bên ngoài) tạo tài khoản mới bằng cách cung cấp thông tin như họ tên, email và mật khẩu.
- Đăng nhập:** Người dùng phải có thể đăng nhập vào hệ thống bằng tài khoản đã tạo hoặc thông qua dịch vụ xác thực tập trung (SSO).
- Đăng xuất:** Mọi người dùng phải có thể kết thúc phiên làm việc của mình một cách an toàn.
- Đặt lại mật khẩu:** Người dùng quên mật khẩu phải có khả năng yêu cầu đặt lại mật khẩu.
- Quản lý vai trò:** Quản trị viên phải có quyền quản lý, gán và thay đổi vai trò cho các tài khoản người dùng.
- Xác thực email:** Hệ thống phải xác thực địa chỉ email của người dùng trong các quy trình quan trọng như đăng ký và đặt lại mật khẩu.

Đăng ký lớp học:

- Hệ thống có thể đề xuất danh sách lớp học nổi bật để sinh viên có thể xem được thông tin lớp học trong một thẻ cụ thể.
- Sinh viên có thể tự tìm kiếm theo nhu cầu và hệ thống hiển thị các lớp học phù hợp.
- Sinh viên có thể xem mô tả thông tin của lớp học được chọn.
- Sinh viên có thể xem thông tin tutor của lớp học được chọn.
- Sinh viên có thể xem đánh giá về tutor của lớp học được chọn.

Tạo chương trình học:

- Thiết lập lịch rảnh:** Cho phép tutor khai báo khoảng thời gian rảnh để sinh viên tham khảo.
- Tạo lớp học:** Cho phép tutor mở lớp học mới, thiết lập điều kiện tham gia và đồng bộ với trang đăng ký lớp của sinh viên.
- Quản lý lớp & lịch dạy:** Cho phép tutor theo dõi danh sách lớp và lịch dạy của mình, đồng thời hỗ trợ các thao tác hủy/chỉnh sửa, xem thông tin chi tiết từng lớp và thông báo đến sinh viên.

Thiết lập lịch trình cho sinh viên:

Sinh viên đổi lịch, hủy lịch hoặc xem lịch trình các buổi học.

- Sinh viên có thể chỉnh sửa, hủy lịch và phải thông báo cho tutor (nếu được yêu cầu) và hệ thống sẽ ghi nhận.
- Sinh viên có thể xem lịch trình các lớp học đã đăng ký một cách trực quan.



Đánh giá:

Sinh viên đánh giá tutor, Tutor theo dõi và ghi nhận tiến bộ.

- Sinh viên có thể rating tutor theo mức độ hài lòng.
- Sinh viên có thể bình luận về chất lượng chương trình để các sinh viên khác tham khảo.
- Tutor có thể theo dõi tiến độ học tập của sinh viên.
- Tutor có thể đánh giá thái độ và thành tích học tập của sinh viên bằng điểm số.
- Tutor và sinh viên có thể sửa đánh giá trong khoảng thời gian nhất định sau lần đánh giá trước.

Phân tích và Báo cáo:

Phòng tạo báo cáo/Phòng công tác sinh viên phân tích dữ liệu và báo cáo.

- Phòng Đào tạo xem đánh giá Tutor, phân tích lại để tối ưu nguồn lực.
- Phòng Công tác sinh viên xem báo cáo về sinh viên (số chương trình tham gia, số buổi vắng không phép,...) để xem xét cộng hoặc trừ điểm thành tích của sinh viên.

Thông báo và tin nhắn:

Gửi thông báo và tin nhắn giữa các bên liên quan.

- Tutor và sinh viên có thể nhắn tin cho nhau thông qua hệ thống để trao đổi thông tin liên quan đến lớp học.
- Nhà trường có thể gửi thông báo quan trọng đến tất cả người dùng hoặc nhóm người dùng cụ thể (ví dụ: sinh viên, tutor).

Quản lý tài liệu:

Lưu trữ và chia sẻ tài liệu học tập.

- Tutor có thể tải lên tài liệu học tập (ví dụ: bài giảng, bài tập) và chia sẻ với sinh viên trong lớp học.
- Sinh viên có thể truy cập để xem và tải xuống tài liệu học tập được chia sẻ bởi tutor.



2.2 Yêu cầu phi chức năng (Non-functional requirements)

Bảo mật:

- Hỏi về vai trò của người dùng (sinh viên, tutor hay admin) trước khi nhấn nút đăng nhập.
- Ứng với sinh viên, tutor hoặc admin, hệ thống sẽ hiển thị các tác vụ phù hợp.
- Mã hóa mật khẩu người dùng bằng các loại thuật toán mã hóa hashing (Argon 2, Bcrypt, scrypt,...).
- Hiện cảnh báo khi hệ thống đánh giá mật khẩu người dùng đặt khi tạo tài khoản là yếu.
- Các giao tiếp phải được mã hóa để đảm bảo an toàn bảo mật.

Đồng bộ và nhất quán:

- Khi truy cập dữ liệu, hệ thống sẽ tải đúng dữ liệu này từ nguồn chính (HCMUT_DATACORE/ HCMUT_LIBRARY).
- Mỗi lần khi thoát ra và vào lại hoặc thay đổi tác vụ, nếu có thay đổi dữ liệu từ nguồn chính (HCMUT_DATACORE/ HCMUT_LIBRARY) thì hệ thống ngay lập tức hiển thị ngay phiên bản mới.

Hiệu năng:

- Thời gian tải trang mỗi khi thoát ra và vào tác vụ mới không vượt quá 2.5s khi kết nối ổn định.
- Thời gian phản hồi lại các thao tác như truy cập, tải xuống, tải lên, tìm kiếm, ... không vượt quá 1.5s.
- Khi nhiều người dùng đồng thời truy cập hệ thống, hệ thống vẫn phải đảm bảo không bị quá tải, tắc nghẽn để không làm giảm thời gian phản hồi của hệ thống.

Tính mở rộng:

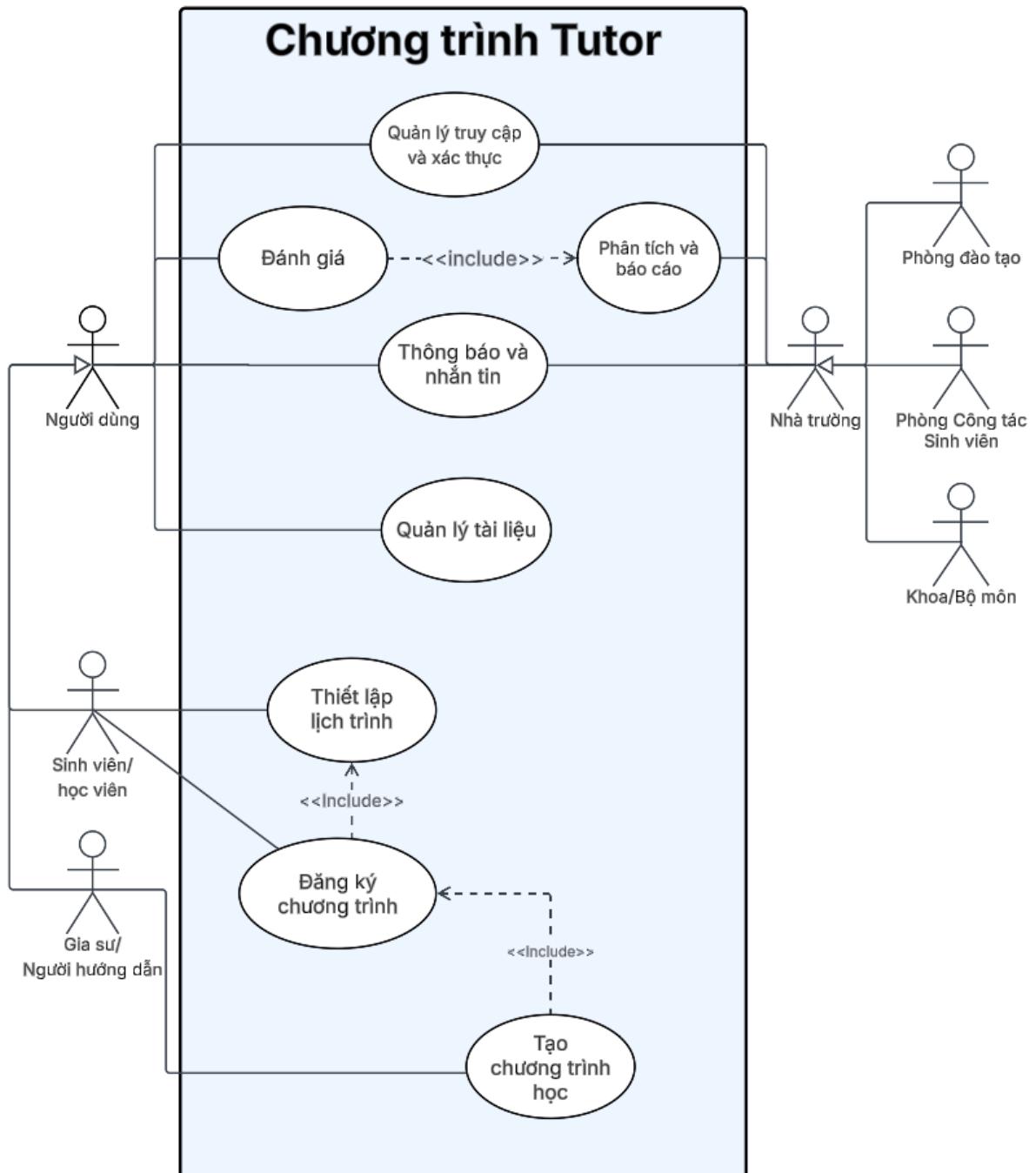
- Có thể dễ dàng thêm các tính năng mới mà không thay đổi cấu trúc cũ của hệ thống.
- Hỗ trợ tích hợp bên thứ ba vào hệ thống.
- Hỗ trợ việc dễ dàng mở rộng hạ tầng để tăng hiệu suất hệ thống mà không ảnh hưởng đến những cái đã có.

Tính có sẵn và khôi phục:

- Các tác vụ trong hệ thống đảm bảo luôn hoạt động, thậm chí phải hoạt động được trong các ngày nghỉ, ngày lễ hay ngoài giờ hành chính.
- Mỗi lần xảy ra sự cố, hệ thống phải đảm bảo dữ liệu vẫn phải còn để có thể phục hồi sau khi khắc phục sự cố.

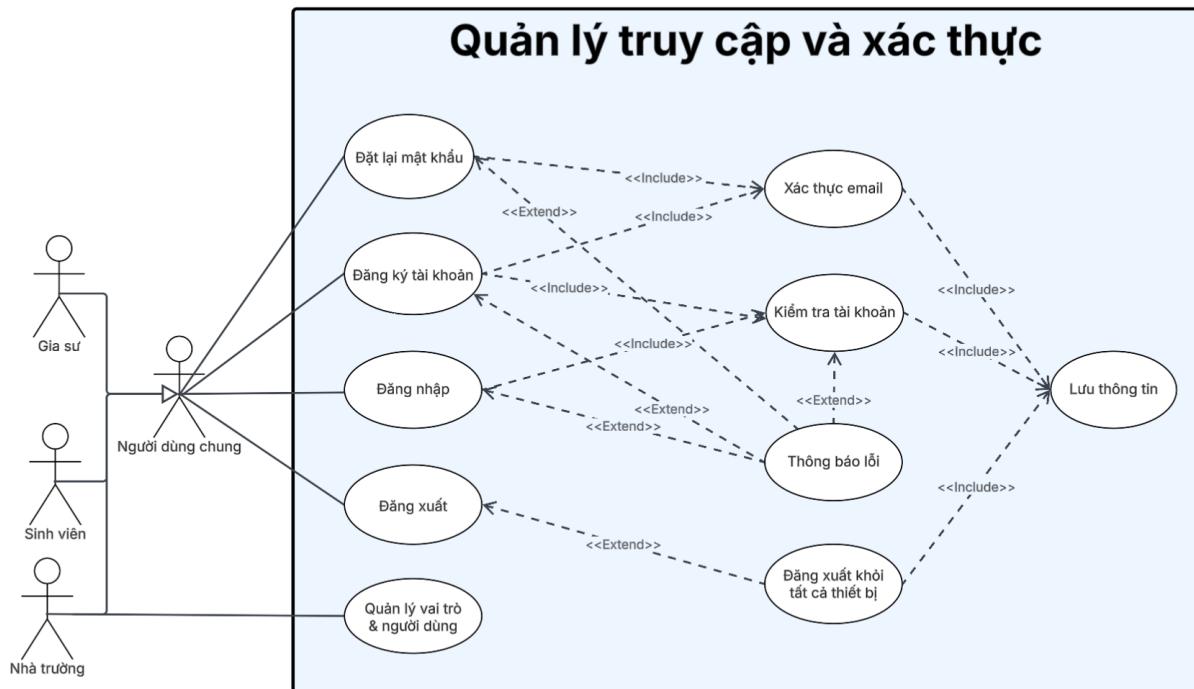
3 Use-case diagrams và use-case scenario

3.1 Sơ đồ tổng hợp chương trình Tutor



Hình 3.1: Sơ đồ use-case hệ thống tổng quát

3.2 Quản lý truy cập và xác thực



Hình 3.2: Sơ đồ use-case cho chức năng quản lý truy cập và xác thực tài khoản



Bảng 3.1: Bảng đặc tả Use-case: Đăng ký tài khoản (UC-AUTH-01)

Use-case	UC-AUTH-01: Đăng ký tài khoản
Actor	Người dùng chung (Sinh viên, Tutor, Nhà trường chưa có tài khoản)
Description	Cho phép người dùng tạo một tài khoản mới để truy cập hệ thống.
Include	UC-AUTH-06: Xác thực Email UC-AUTH-07: Kiểm tra tài khoản
Extend	Không
Precondition	Người dùng chưa có tài khoản trong hệ thống.
Postcondition	Tài khoản người dùng được tạo và chờ kích hoạt qua email.
Trigger	Người dùng ấn vào chữ "Đăng ký tài khoản"
Normal Flows	<ol style="list-style-type: none"> 1. Người dùng chọn chức năng "Đăng ký". 2. Hệ thống hiển thị form đăng ký. 3. Người dùng điền thông tin (họ tên, email, mật khẩu) và gửi đi. 4. Hệ thống gọi use-case UC-AUTH-07 để kiểm tra email đã tồn tại chưa. 5. Hệ thống kiểm tra tính hợp lệ của các thông tin khác. 6. Hệ thống gọi use-case UC-AUTH-06 để gửi email xác thực. 7. Hệ thống hiển thị thông báo yêu cầu kiểm tra email để kích hoạt.
Alternative Flows	Không
Exception Flows	4.1. Email đã tồn tại: Hệ thống hiển thị thông báo lỗi. 5.1. Dữ liệu không hợp lệ: Hệ thống hiển thị thông báo lỗi tương ứng.

Bảng 3.2: Bảng đặc tả Use-case: Đặt lại mật khẩu (UC-AUTH-02)

Use-case	UC-AUTH-02: Đặt lại mật khẩu
Actor	Người dùng chung
Description	Cho phép người dùng đã quên mật khẩu thiết lập một mật khẩu mới.
Include	UC-AUTH-06: Xác thực Email
Extend	UC-AUTH-09 Thông báo lỗi
Precondition	Người dùng có tài khoản đã được kích hoạt.
Postcondition	Mật khẩu của người dùng được cập nhật thành công.
Trigger	Người dùng bấm vào chữ "Quên mật khẩu?"
Normal Flows	<ol style="list-style-type: none"> 1. Người dùng chọn "Quên mật khẩu?". 2. Người dùng nhập địa chỉ email đã đăng ký. 3. Hệ thống gọi use-case UC-AUTH-06 gửi email chứa token đặt lại mật khẩu. 4. Người dùng mở link và nhập mật khẩu mới. 5. Hệ thống cập nhật mật khẩu và thông báo thành công.
Alternative Flows	Không
Exception Flows	<p>3.1. Check Email Failed: Nếu xác thực email UC-AUTH-06 trả về kết quả thất bại, hệ thống sẽ thực hiện use-case UC-AUTH-09.</p> <p>3.2. Email không tồn tại: Hiển thị thông báo lỗi.</p> <p>4.1. Link không hợp lệ/hết hạn: Hiển thị thông báo lỗi.</p>

Bảng 3.3: Bảng đặc tả Use-case: Đăng nhập (UC-AUTH-03)

Use-case	UC-AUTH-03: Đăng nhập
Actor	Người dùng chung
Description	Cho phép người dùng xác thực và truy cập hệ thống.
Include	UC-AUTH-07: Kiểm tra tài khoản
Extend	UC-AUTH-09: Thông báo lỗi
Precondition	Người dùng chưa đăng nhập.
Postcondition	Người dùng được chuyển hướng đến trang chính theo vai trò của mình.
Trigger	Người dùng điền thông tin đăng nhập rồi ấn "Đăng nhập"
Normal Flows	<ul style="list-style-type: none"> 1. Người dùng truy cập trang đăng nhập. 2. Người dùng nhập thông tin (email, mật khẩu). 3. Hệ thống gọi use-case UC-AUTH-07 để xác thực. 4. Nếu xác thực thành công, hệ thống tạo phiên làm việc và thực hiện Post-condition.
Alternative Flows	Không
Exception Flows	3.1. LoginFailed: Nếu UC-AUTH-07 trả về kết quả thất bại, hệ thống sẽ thực hiện use-case UC-AUTH-09 .

Bảng 3.4: Bảng đặc tả Use-case: Đăng xuất (UC-AUTH-04)

Use-case	UC-AUTH-04: Đăng xuất
Actor	Người dùng chung
Description	Cho phép người dùng kết thúc phiên làm việc hiện tại.
Include	Không
Extend	Không
Precondition	Người dùng đã đăng nhập.
Postcondition	Phiên làm việc bị hủy và người dùng được chuyển về trang đăng nhập.
Trigger	Người dùng ấn vào nút "Đăng xuất"
Normal Flows	<ul style="list-style-type: none"> 1. Người dùng chọn chức năng "Đăng xuất". 2. Hệ thống hủy session/token. 3. Hệ thống quay lại trang đăng nhập.
Alternative Flows	Không
Exception Flows	Không

Bảng 3.5: Bảng đặc tả Use-case: Quản lý vai trò và người dùng (UC-AUTH-05)

Use-case	UC-AUTH-05: Quản lý vai trò và người dùng
Actor	Nhà trường
Description	Cho phép Nhà trường xem danh sách, tìm kiếm, chỉnh sửa thông tin (bao gồm cả vai trò) và thay đổi trạng thái của các tài khoản người dùng.
Include	Không
Extend	Không
Precondition	1. Nhà trường đã đăng nhập. 2. Ít nhất một tài khoản người dùng đã tồn tại trong hệ thống.
Postcondition	Thông tin, vai trò hoặc trạng thái của tài khoản người dùng được cập nhật thành công trong hệ thống.
Trigger	Không
Normal Flows	1. Nhà trường truy cập trang "Quản lý Người dùng". 2. Hệ thống hiển thị danh sách người dùng kèm công cụ tìm kiếm/ lọc. 3. Nhà trường tìm và chọn một tài khoản để chỉnh sửa. 4. Nhà trường thực hiện các thay đổi như: a. Chỉnh sửa thông tin cá nhân (ví dụ: họ tên). b. Thay đổi vai trò (Sinh viên, Tutor, Nhà trường). c. Thay đổi trạng thái (ví dụ: Kích hoạt, Vô hiệu hóa). 5. Nhà trường lưu lại các thay đổi. 6. Hệ thống xác thực và cập nhật dữ liệu vào cơ sở dữ liệu. 7. Hệ thống hiển thị thông báo cập nhật thành công.
Alternative Flows	Không
Exception Flows	4.1. Dữ liệu đầu vào không hợp lệ: Hệ thống hiển thị thông báo lỗi tương ứng. 6.1. Lỗi cập nhật cơ sở dữ liệu: Hệ thống hiển thị thông báo lỗi chung.

Bảng 3.6: Bảng đặc tả Use-case: Xác thực email (UC-AUTH-06)

Use-case	UC-AUTH-06: Xác thực Email
Actor	Không
Description	Gửi một email chứa liên kết xác thực đến người dùng. use-case này được gọi bởi các use-case khác.
Include	UC-AUTH-08: Lưu thông tin
Extend	Không
Precondition	Hệ thống nhận được yêu cầu gửi email đến một địa chỉ email hợp lệ.
Postcondition	Email chứa token xác thực được gửi tới người dùng.
Trigger	Không
Normal Flows	<ol style="list-style-type: none"> 1. Hệ thống tạo một token xác thực duy nhất, có thời hạn. 2. Hệ thống gọi use-case UC-AUTH-08 để lưu token này vào cơ sở dữ liệu, liên kết với tài khoản người dùng. 3. Hệ thống gửi email chứa link kèm token tới địa chỉ email của người dùng.
Alternative Flows	Không
Exception Flows	Không

Bảng 3.7: Bảng đặc tả Use-case: Kiểm tra tài khoản (UC-AUTH-07)

Use-case	UC-AUTH-07: Kiểm tra tài khoản
Actor	Không
Description	Kiểm tra thông tin đăng nhập (email, mật khẩu) mà người dùng cung cấp.
Include	UC-AUTH-08: Lưu thông tin
Extend	UC-AUTH-09: Thông báo lỗi
Precondition	Hệ thống nhận được thông tin đăng nhập từ người dùng.
Postcondition	Trả về kết quả xác thực (thành công hoặc thất bại).
Trigger	Không
Normal Flows	<ol style="list-style-type: none"> 1. Hệ thống tìm tài khoản dựa trên email. 2. Nếu tìm thấy, hệ thống so sánh hash của mật khẩu người dùng nhập với hash trong cơ sở dữ liệu. 3. Nếu khớp, trả về "thành công". 4. (Tùy chọn) Hệ thống gọi use-case UC-AUTH-08 để ghi lại nhật ký đăng nhập.
Alternative Flows	Không
Exception Flows	2.1. CheckFailed: Nếu xác thực thất bại ở Normal Flows (email không tồn tại hoặc mật khẩu sai), hệ thống sẽ thực hiện use-case UC-AUTH-09 .

Bảng 3.8: Bảng đặc tả Use-case: Lưu thông tin (UC-AUTH-08)

Use-case	UC-AUTH-08: Lưu thông tin
Actor	Không
Description	Ghi hoặc cập nhật dữ liệu vào cơ sở dữ liệu (ví dụ: thông tin người dùng, token, nhât ký).
Include	Không
Extend	Không
Precondition	Hệ thống có dữ liệu hợp lệ cần lưu trữ.
Postcondition	Dữ liệu được lưu thành công vào cơ sở dữ liệu.
Trigger	Không
Normal Flows	Không
Alternative Flows	Không
Exception Flows	Không

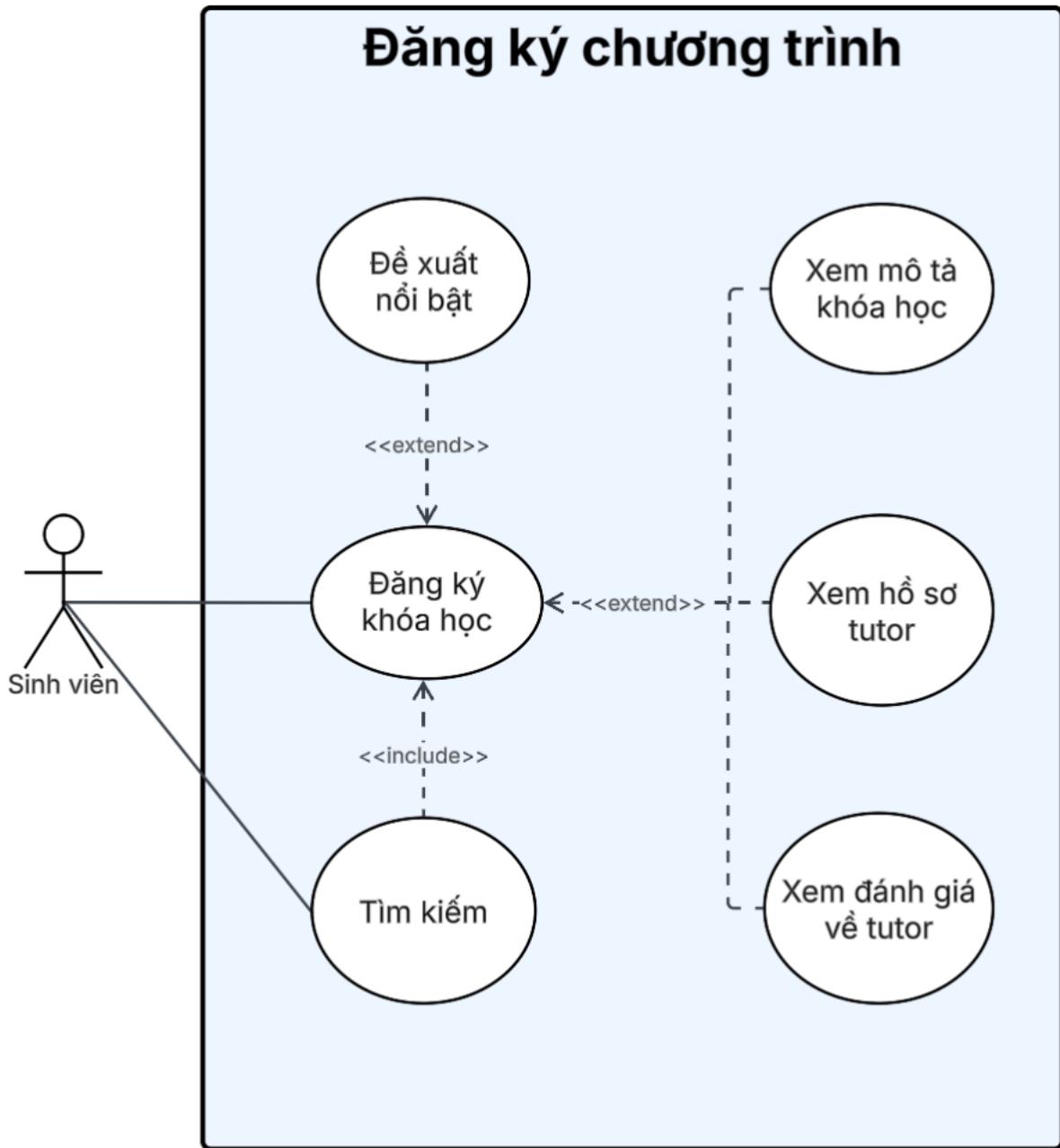
Bảng 3.9: Bảng đặc tả Use-case: Thông báo lỗi (UC-AUTH-09)

Use-case	UC-AUTH-09: Thông báo lỗi
Actor	Không
Description	Hiển thị một thông báo lỗi thân thiện cho người dùng khi có ngoại lệ xảy ra.
Include	Không
Extend	Không
Precondition	Một Exception Flows xảy ra trong một use-case khác (ví dụ: Đăng nhập thất bại).
Postcondition	Người dùng thấy được thông báo lỗi trên giao diện.
Trigger	Không
Normal Flows	Không
Alternative Flows	Không
Exception Flows	Không

Bảng 3.10: Bảng đặc tả Use-case: Đăng xuất khỏi tất cả các thiết bị (UC-AUTH-10)

Use-case	UC-AUTH-10: Đăng xuất khỏi tất cả thiết bị
Actor	Người dùng chung
Description	Cho phép người dùng kết thúc phiên làm việc của tài khoản trên tất cả các thiết bị đã đăng nhập. Use-case này là một chức năng mở rộng của use-case Đăng xuất.
Include	UC-AUTH-08: Lưu thông tin
Extend	Không
Precondition	Người dùng đã đăng nhập và đang thực hiện thao tác đăng xuất.
Postcondition	Tất cả các phiên làm việc của người dùng trên mọi thiết bị đã được hủy. Người dùng được chuyển về trang đăng nhập.
Trigger	Người dùng chọn tùy chọn "Đăng xuất khỏi tất cả thiết bị" khi đang thực hiện thao tác đăng xuất.
Normal Flows	<ol style="list-style-type: none"> Người dùng chọn tùy chọn "Đăng xuất khỏi tất cả thiết bị" sau khi chọn đăng xuất. Hệ thống hủy tất cả các session/token đã được lưu cho tài khoản này. Hệ thống gọi use-case UC-AUTH-08 để ghi lại nhật ký đăng xuất. Hệ thống chuyển hướng người dùng về trang đăng nhập và hiển thị thông báo thành công.
Alternative Flows	Không
Exception Flows	2.1 Lỗi hệ thống khi hủy session/token: Hệ thống hiển thị thông báo lỗi "Không thể đăng xuất. Vui lòng thử lại sau."

3.3 Đăng ký chương trình



Hình 3.3: Sơ đồ use-case cho chức năng Đăng ký chương trình học của sinh viên

Bảng 3.11: Bảng đặc tả Use-case: Đề xuất tutor (UC-REG-01)

Use-case	UC-REG-01: Đề xuất tutor
Actor	Sinh viên
Descriptions	Sau khi chọn "Đăng ký" ở trang chính, hệ thống hiển thị các khóa học/tutor nổi bật (rating cao, hoạt động nhiều, liên quan đến các tìm kiếm trước đó (nếu có), ...) dưới dạng card. Sinh viên có thể chọn 1 card để xem chi tiết.
Include	Không
Extend	UC-REG-03: Đăng ký khóa học
Precondition	1. Sinh viên đã đăng nhập thành công và chọn "Đăng ký" ở trang chính
Postcondition	1. Danh sách các tutor/khóa học nổi bật được hiển thị 2. Nếu sinh viên bấm chọn 1 card, thì chuyển sang UC-REG-03
Trigger	Sinh viên chọn icon "Đăng ký" ở trang chính
Normal Flows	1. Hệ thống lấy dữ liệu từ tutor nổi bật 2. Hiển thị ở trang chính 3. Sinh viên chọn 1 khóa học/tutor để đăng ký → chuyển qua UC-REG-03
Alternative Flows	Không
Exception Flows	2.1. Lỗi tải đề xuất: hiển thị thông báo lỗi và yêu cầu nhấn tải lại.

Bảng 3.12: Bảng đặc tả Use-case: Tìm kiếm (UC-REG-02)

Use-case	UC-REG-02: Tìm kiếm
Actor	Sinh viên
Descriptions	Sinh viên nhập từ khóa vào thanh tìm kiếm và nhấn Enter. Hệ thống hiển thị trang gồm các khóa học liên quan, kèm filter để tùy chỉnh.
Include	UC-REG-03: Đăng ký khóa học
Extend	Không
Precondition	1. Sinh viên đã đăng nhập thành công 2. Sinh viên chọn "Đăng ký" ở trang chính
Postcondition	Trang kết quả hiển thị danh sách các card khóa học phù hợp và lưu trạng thái filter
Trigger	Sinh viên nhấn Enter sau khi nhập thông tin trên thanh tìm kiếm
Normal Flows	1. Sinh viên nhập nhu cầu trên thanh tìm kiếm và nhấn Enter 2. Hệ thống tìm khóa học/tutor phù hợp 3. Hiển thị danh sách khóa học, kèm filter 4. Sinh viên chọn 1 khóa học/tutor để đăng ký → Chuyển sang UC-REG-03
Alternative Flows	2.1. Không tìm thấy kết quả phù hợp: hiển thị thông báo
Exception Flows	Không

Bảng 3.13: Bảng đặc tả Use-case: Đăng ký khóa học (UC-REG-03)

Use-case	UC-REG-03: Đăng ký khóa học
Actor	Sinh viên
Descriptions	Sinh viên chọn một khóa học trong danh sách (từ đề xuất hoặc kết quả tìm kiếm) để xem thông tin chi tiết và đăng ký
Include	Không
Extend	UC-REG-04: Xem mô tả khóa học UC-REG-05: Xem hồ sơ tutor UC-REG-06: Xem đánh giá về tutor
Precondition	1. Sinh viên đã đăng nhập thành công 2. Đã có danh sách khóa học từ UC-REG-01: Đề xuất tutor hoặc UC-REG-02: Tìm kiếm
Postcondition	Đăng ký khóa học thành công
Trigger	Sinh viên nhấn vào item/card của khóa học
Normal Flows	1. Sinh viên chọn item/card khóa học từ đề xuất/kết quả tìm kiếm 2. Sinh viên có thể nhấn vào khóa học để xem thông tin về: a. UC-REG-04: Xem mô tả khóa học b. UC-REG-05: Xem hồ sơ tutor c. UC-REG-06: Xem đánh giá về tutor 3. Sinh viên nhấn đăng ký khóa học
Alternative Flows	3.1. Sinh viên không đăng ký khóa học: giữ nguyên page hiện tại.
Exception Flows	3.1. Khóa học vừa bị gỡ xuống: hệ thống báo lỗi và ở lại trang hiện tại

Bảng 3.14: Bảng đặc tả Use-case: Xem mô tả khóa học (UC-REG-04)

Use-case	UC-REG-04: Xem mô tả khóa học
Actor	Sinh viên
Descriptions	Hiển thị nội dung khóa học: mục tiêu, phạm vi,...
Inlcude	Không
Extend	Không
Precondition	Sinh viên bấm vào một khóa học
Postcondition	Thông tin mô tả khóa học được hiển thị
Trigger	Tự động hiển thị trên trang chi tiết
Normal Flows	Hệ thống tải và hiển thị thông tin khóa học
Alternative Flows	Không
Exception Flows	Lỗi tải thông tin

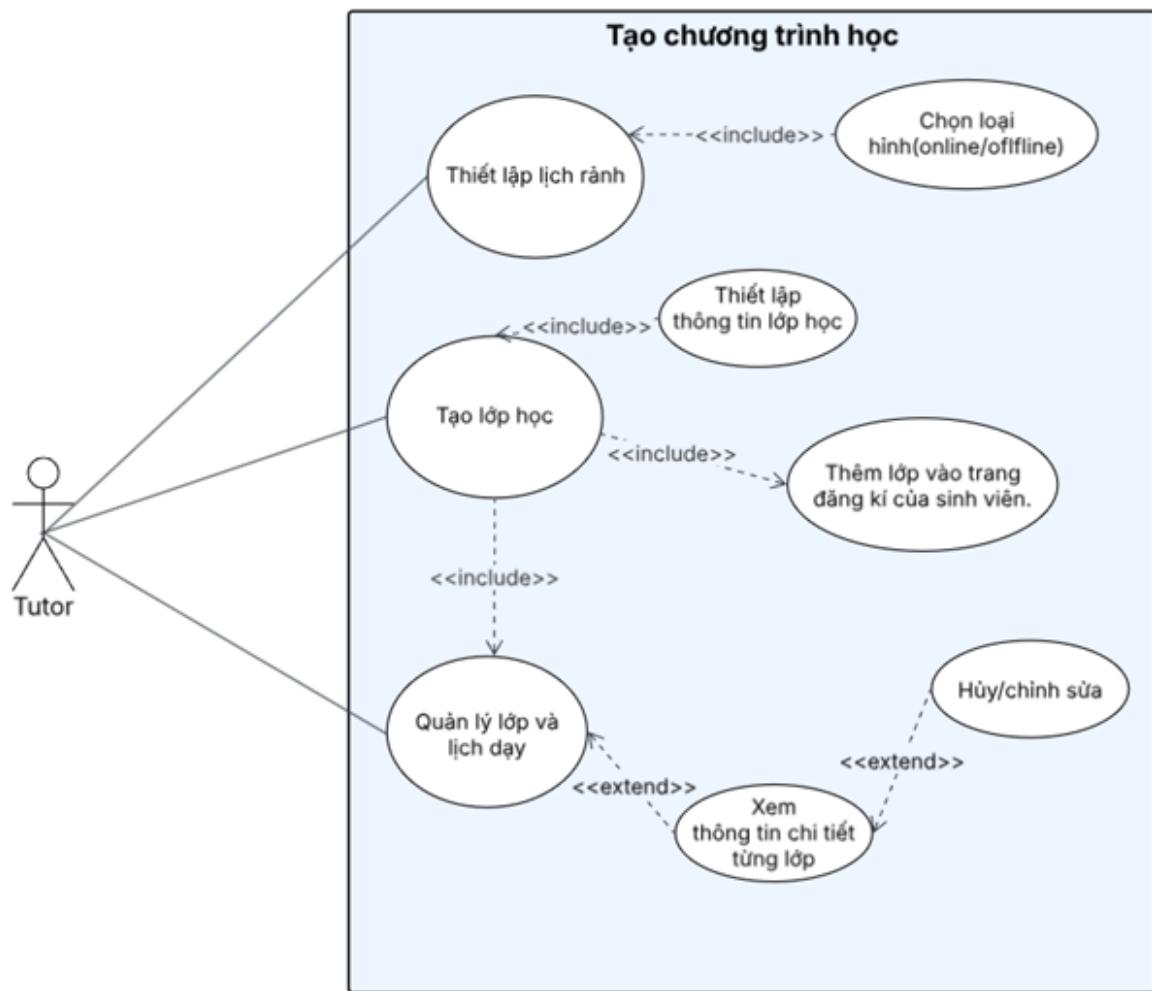
Bảng 3.15: Bảng đặc tả Use-case: Xem hồ sơ tutor (UC-REG-05)

Use-case	UC-REG-05: Xem hồ sơ tutor
Actor	Sinh viên
Descriptions	Hiển thị hồ sơ tutor: chuyên ngành, lĩnh vực, kinh nghiệm,...
Inlcude	Không
Extend	Không
Precondition	Sinh viên đang ở trang thông tin chi tiết về khóa học
Postcondition	Thông tin hồ sơ tutor được hiển thị
Trigger	Sinh viên chọn “Hồ sơ tutor”
Normal Flows	Hệ thống tải và hiển thị hồ sơ tutor
Alternative Flows	Không
Exception Flows	Lỗi tải thông tin

Bảng 3.16: Bảng đặc tả Use-case: Xem đánh giá về tutor (UC-REG-06)

Use-case	UC-REG-06: Xem đánh giá về tutor
Actor	Sinh viên
Descriptions	Hiển thị rating, nhận xét từ sinh viên khác về tutor/khoa học
Inlcude	Không
Extend	Không
Precondition	Sinh viên đang ở trang thông tin chi tiết về khóa học
Postcondition	Đánh giá về tutor được hiển thị
Trigger	Sinh viên chọn “Xem đánh giá về tutor”
Normal Flows	Hệ thống tải và hiển thị đánh giá về tutor
Alternative Flows	Không
Exception Flows	Lỗi tải thông tin

3.4 Tạo chương trình học



Hình 3.4: Sơ đồ use-case cho chức năng "Tạo chương trình học" của Tutor



Bảng 3.17: Bảng đặc tả Use-case: Thiết lập lịch rảnh (UC-CLASS-01)

Use-case	UC-CLASS-01: Thiết lập lịch rảnh
Actor	Tutor
Description	Tutor có thể thiết lập khoảng thời gian rảnh để hệ thống ghi nhận và hiển thị cho sinh viên tham khảo khi cần trao đổi với Tutor.
Include	UC-CLASS-04: Chọn loại hình (online/offline)
Extend	Không
Precondition	Tutor đã đăng nhập vào hệ thống thành công và hệ thống hoạt động ổn định.
Postcondition	Lịch rảnh của tutor được lưu vào hệ thống và hiển thị cho sinh viên tham khảo.
Trigger	Tutor chọn chức năng “Thiết lập lịch rảnh”
Normal Flows	<ol style="list-style-type: none"> 1. Tutor chọn chức năng “Thiết lập lịch rảnh”. 2. Hệ thống hiển thị giao diện chọn thời gian rảnh. 3. Tutor nhập thông tin thời gian rảnh. 4. Hệ thống gọi use-case UC-CLASS-04 để Tutor lựa chọn loại hình. 5. Tutor xác nhận lưu lịch. 6. Hệ thống cập nhật lịch rảnh và công khai cho sinh viên xem.
Alternative Flows	4.1. Tutor hủy thao tác → Hệ thống quay lại bước 2.
Exception Flows	Không

Bảng 3.18: Bảng đặc tả Use-case: Tạo lớp học (UC-CLASS-02)

Use-case	UC-CLASS-02: Tạo lớp học
Actor	Tutor
Description	Tutor có thể tạo lớp học mới để sinh viên có thể đăng kí tham gia.
Include	UC-CLASS-05: Thiết lập thông tin lớp học UC-CLASS-06: Đăng kí lớp học UC-CLASS-03: Quản lý lớp và lịch dạy
Extend	Không
Trigger	Tutor chọn chức năng “Tạo lớp học”.
Precondition	Tutor đãng nhập thành công vào hệ thống.
Postcondition	Lớp học mới được tạo thành công và được lưu vào hệ thống.
Normal Flows	1. Tutor chọn chức năng “Tạo lớp học”. 2. Hệ thống gọi use-case UC-CLASS-05 để tutor nhập thông tin lớp học. 3. Tutor nhập thông tin lớp học (tên, số lượng, thời gian, hình thức,...). 4. Tutor xác nhận tạo lớp học. 5. Hệ thống lưu thông tin lớp học, hiển thị lớp học trên trang đăng ký của sinh viên và đồng bộ với danh sách các lớp của tutor (gọi đến UC-CLASS-03).
Alternative Flows	3.1. Tutor nhập thiếu thông tin → Hệ thống thông báo lỗi, quay lại bước 2.
Exception Flows	Không

Bảng 3.19: Bảng đặc tả Use-case: Quản lý lớp và lịch dạy (UC-CLASS-03)

Use-case	UC-CLASS-03: Quản lý lớp và lịch dạy
Actor	Tutor
Description	Tutor có thể xem danh sách các lớp học đã được tạo, xem chi tiết từng lớp, thời gian diễn ra và có thể hủy/chỉnh sửa thông tin từng lớp cụ thể.
Include	Không
Extend	UC-CLASS-07: Xem thông tin lớp học UC-CLASS-08: Hủy/Chỉnh sửa
Precondition	Tutor đã đăng nhập thành công vào hệ thống.
Postcondition	Danh sách lớp học và thời gian diễn ra được hiển thị, các thao tác cập nhật (nếu có) được lưu vào hệ thống.
Trigger	Tutor chọn chức năng “Quản lý lớp & lịch dạy”.
Normal Flows	<ol style="list-style-type: none"> 1. Tutor chọn chức năng “Quản lý lớp & lịch dạy”. 2. Hệ thống hiển thị danh sách các lớp học mà tutor phụ trách. 3. Tutor có thể chọn một lớp cụ thể để xem chi tiết (gọi đến UC-CLASS-07). 4. Tutor có thể chọn thao tác Hủy/Chỉnh sửa (gọi đến UC-CLASS-08) của lớp cụ thể.
Alternative Flows	2.1. Tutor chỉ xem danh sách mà không thực hiện thao tác nào khác → Use-case kết thúc tại bước 2.
Exception Flows	Không

Bảng 3.20: Bảng đặc tả Use-case: Chọn loại hình (online/offline) (UC-CLASS-04)

Use-case	UC-CLASS-04: Chọn loại hình (online/offline)
Actor	Tutor
Description	Cho phép Tutor lựa chọn loại hình khi thiết lập lịch rảnh.
Include	Không
Extend	Không
Precondition	Đang trong quá trình thiết lập lịch rảnh hoặc đăng ký lớp học.
Postcondition	Lựa chọn hình thức được lưu lại.
Trigger	Trigger từ use-case Thiết lập lịch rảnh (UC-CLASS-01)
Normal Flows	<ol style="list-style-type: none"> 1. Hệ thống hiển thị lựa chọn loại hình. 2. Tutor chọn loại hình mong muốn. 3. Hệ thống lưu lại lựa chọn.
Alternative Flows	Không
Exception Flows	Không

Bảng 3.21: Bảng đặc tả Use-case: Thiết lập thông tin lớp học (UC-CLASS-05)

Use-case	UC-CLASS-05: Thiết lập thông tin lớp học
Actor	Tutor
Description	Cho phép Tutor nhập thông tin chi tiết lớp học (tên lớp, số lượng, thời gian, hình thức...).
Include	Không
Extend	Không
Precondition	Tutor đang trong quá trình tạo lớp.
Postcondition	Thông tin lớp học được lưu thành công.
Trigger	Trigger từ use-case Tạo lớp học (UC-CLASS-02).
Normal Flows	<ol style="list-style-type: none"> 1. Hệ thống hiển thị form nhập liệu. 2. Tutor điền các thông tin cần thiết. 3. Tutor nhấn lưu. 4. Hệ thống lưu thông tin.
Alternative Flows	Không
Exception Flows	Không

Bảng 3.22: Bảng đặc tả Use-case: Thêm lớp vào trang đăng ký của sinh viên (UC-CLASS-06)

Use-case	UC-CLASS-06: Thêm lớp vào trang đăng ký của sinh viên
Actor	Hệ thống
Description	Thêm lớp mà tutor tạo vào trang đăng ký lớp của sinh viên.
Include	Không
Extend	Không
Precondition	Tutor thiết lập đầy đủ thông tin của lớp được tạo và ấn lưu.
Postcondition	Hệ thống thêm lớp vào trang đăng ký lớp của sinh viên thành công.
Trigger	Không
Normal Flows	Hệ thống thêm lớp vào trang đăng ký của sinh viên.
Alternative Flows	Không
Exception Flows	Không

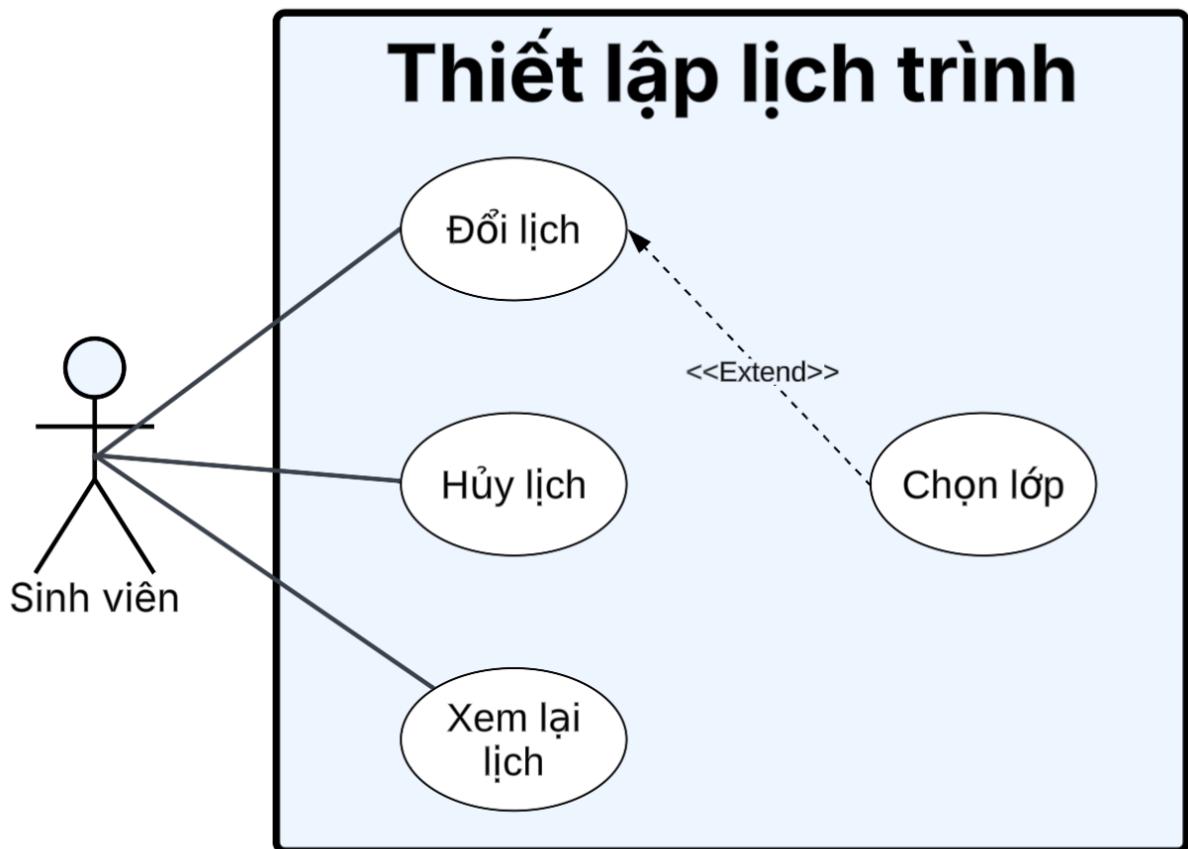
Bảng 3.23: Bảng đặc tả Use-case: Xem thông tin chi tiết từng lớp (UC-CLASS-07)

Use-case	UC-CLASS-07: Xem thông tin chi tiết từng lớp
Actor	Tutor
Description	Cho phép tutor xem các thông tin chi tiết của từng lớp mà tutor dạy.
Include	Không
Extend	UC-CLASS-08: Hủy/Chỉnh sửa
Precondition	Lớp đã được tạo thành công.
Postcondition	Thông tin lớp được hiển thị trên giao diện.
Trigger	Trigger từ use-case Quản lý lớp & lịch dạy (UC-CLASS-03).
Normal Flows	<ol style="list-style-type: none"> 1. Tutor chọn “ Xem chi tiết” lớp cụ thể. 2. Hệ thống truy xuất dữ liệu. 3. Hệ thống hiển thị chi tiết thông tin lớp (mô tả,số lượng, danh sách,..).
Alternative Flows	Không
Exception Flows	Không

Bảng 3.24: Bảng đặc tả Use-case: Hủy/Chỉnh sửa (UC-CLASS-08)

Use-case	UC-CLASS-08: Hủy/Chỉnh sửa
Actor	Tutor
Description	Cho phép tutor hủy lớp hoặc chỉnh sửa thông tin của lớp (thời gian, số lượng, yêu cầu,...).
Include	Không
Extend	Không
Precondition	Tutor đã tạo lớp thành công và có quyền chỉnh sửa.
Postcondition	Thông tin lớp được hủy hoặc cập nhật thành công.
Trigger	Trigger từ use-case UC-CLASS-07: Xem thông tin chi tiết từng lớp.
Normal Flows	<ol style="list-style-type: none"> 1. Tutor chọn chức năng hủy/chỉnh sửa sau khi ấn xem chi tiết. 2. Tutor lựa chọn Hủy lớp hoặc Chỉnh sửa. 3. Hệ thống cập nhật và lưu thông tin.
Alternative Flows	Không
Exception Flows	Không

3.5 Thiết lập lịch trình



Hình 3.5: Sơ đồ use-case cho chức năng thiết lập lịch trình của sinh viên

Bảng 3.25: Bảng đặc tả Use-case: Đổi lịch (UC-SCD-01)

Use-case	UC-SCD-01: Đổi lịch
Actor	Sinh viên
Description	Sinh viên có thể thay đổi lịch hẹn đã đăng ký trước đó.
Include	Không
Extend	UC-SCD-04: Chọn lớp
Precondition	Sinh viên đã có lịch đăng ký hợp lệ trong hệ thống. Tutor cho phép đổi lịch (chưa đến hạn khóa lịch).
Postcondition	Lịch cũ bị hủy, lịch mới được tạo.
Trigger	Sinh viên chọn chức năng "Đổi lịch".
Normal Flows	<ol style="list-style-type: none"> 1. Sinh viên chọn "Đổi lịch". 2. Hệ thống hiển thị danh sách các lịch hiện tại. 3. Sinh viên chọn lịch muốn đổi. 4. Hệ thống hiển thị các khung giờ khác của tutor. 5. Sinh viên chọn khung giờ mới. 6. Hệ thống gọi use-case UC-SCD-04 yêu cầu chọn lớp (extend). 7. Sinh viên chọn lớp mới. 8. Hệ thống cập nhật lịch và hiển thị xác nhận.
Alternative Flows	4.1. Nếu tutor không còn slot khả dụng → hệ thống thông báo “Không thể đổi lịch”.
Exception Flows	8.1. Lỗi khi cập nhật dữ liệu → trả lại về lịch cũ.

Bảng 3.26: Bảng đặc tả Use-case: Hủy lịch (UC-SCD-02)

Use-case	UC-SCD-02: Hủy lịch
Actor	Sinh viên
Description	Sinh viên có thể hủy một lịch hẹn đã đăng ký.
Include	Không
Extend	Không
Precondition	Sinh viên đã có ít nhất một lịch đăng ký trong hệ thống.
Postcondition	Lịch bị xóa khỏi thời khóa biểu của sinh viên
Trigger	Sinh viên chọn "Hủy lịch".
Normal Flows	<ol style="list-style-type: none"> 1. Sinh viên chọn "Hủy lịch". 2. Hệ thống hiển thị danh sách lịch đã đăng ký. 3. Sinh viên chọn lịch muốn hủy. 4. Hệ thống yêu cầu xác nhận hủy. 5. Sinh viên xác nhận. 6. Hệ thống xóa lịch và hiển thị thông báo thành công.
Alternative Flows	5.1. Sinh viên hủy thao tác → hệ thống quay lại màn hình trước đó.
Exception Flows	6.1. Lỗi khi xóa lịch → hiển thị thông báo “Không thể hủy lịch”.

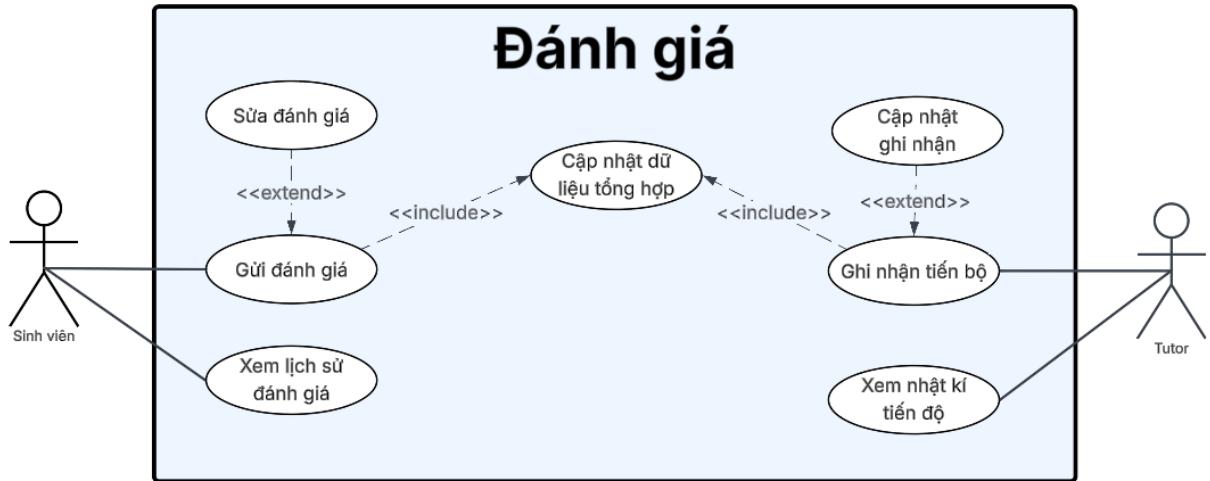
Bảng 3.27: Bảng đặc tả Use-case: Xem lại lịch (UC-SCD-03)

Use-case	UC-SCD-03: Xem lại lịch
Actor	Sinh viên
Description	Sinh viên có thể xem lại toàn bộ lịch đã đăng ký.
Include	Không
Extend	Không
Precondition	Sinh viên đã đăng nhập thành công.
Postcondition	Lịch được hiển thị đúng với thông tin trong hệ thống.
Trigger	Sinh viên chọn "Xem lại lịch".
Normal Flows	<ol style="list-style-type: none"> 1. Sinh viên chọn "Xem lại lịch". 2. Hệ thống truy xuất dữ liệu lịch từ database. 3. Hệ thống hiển thị lịch theo tuần/tháng.
Alternative Flows	2.1. Nếu không có lịch nào → hiển thị thông báo “Không có lịch”.
Exception Flows	2.1. Lỗi khi tải dữ liệu → hiển thị thông báo “Không thể tải lịch”.

Bảng 3.28: Bảng đặc tả Use-case: Chọn lớp (UC-SCD-04)

Use-case	UC-SCD-04: Chọn lớp
Actor	Sinh viên
Description	Khi đăng ký hoặc đổi lịch, sinh viên cần chọn lớp phù hợp.
Include	Không
Extend	Không
Precondition	Hệ thống đã có danh sách lớp khả dụng.
Postcondition	Lớp được gán cho lịch đã chọn.
Trigger	Sinh viên ấn "Chọn lớp"
Normal Flows	<ol style="list-style-type: none"> 1. Sinh viên chọn "Chọn lớp". 2. Hệ thống hiển thị danh sách lớp còn chõ. 3. Sinh viên chọn một lớp. 4. Hệ thống gán lớp cho lịch.
Alternative Flows	2.1. Nếu không có lớp trống → thông báo “Không còn lớp”.
Exception Flows	2.1. Lỗi khi truy xuất dữ liệu lớp → thông báo “Không thể tải danh sách lớp”.

3.6 Đánh giá



Hình 3.6: Sơ đồ use-case cho chức năng đánh giá sinh viên/tutor

Bảng 3.29: Bảng đặc tả Use-case: Gửi đánh giá (UC-EVAL-01)

Tên use-case	UC-EVAL-01: Gửi đánh giá
Actors	Sinh viên
Description	Sinh viên gửi đánh giá cho từng buổi học theo thang điểm 1-5 và nhận xét ngắn.
Precondition	1. Sinh viên đã đăng nhập. 2. Buổi học thuộc lịch của sinh viên và còn thời hạn cho phép đánh giá.
Postcondition	Đánh giá được lưu, hệ thống cập nhật dữ liệu tổng hợp theo buổi học, lớp, tutor.
Include	UC-EVAL-07: Cập nhật dữ liệu tổng hợp
Extend	Không
Trigger	Sinh viên mở trang “Đánh giá buổi học” sau khi buổi học kết thúc.
Normal Flows	1. Sinh viên mở trang đánh giá của buổi học và chọn tiêu chí. 2. Sinh viên nhập điểm và nhận xét rồi nhấn “Gửi”. 3. Hệ thống lưu đánh giá, cập nhật dữ liệu tổng hợp (gọi UC-EVAL-07).
Alternative Flows	Chọn chế độ ẩn danh nếu hệ thống bật.
Exception Flows	2.1. Thiếu trường bắt buộc (rating, comment,...) → báo lỗi. 2.2. Hết hạn gửi đánh giá → chặn gửi.

Bảng 3.30: Bảng đặc tả Use-case: Sửa đánh giá (UC-EVAL-02)

Tên use-case	UC-EVAL-02: Sửa đánh giá
Actors	Sinh viên
Description	Sinh viên chỉnh sửa đánh giá đã gửi trong thời hạn cho phép.
Precondition	Có đánh giá hiện hữu của sinh viên cho buổi học và còn thời hạn sửa.
Postcondition	Đánh giá được cập nhật và hệ thống cập nhật lại dữ liệu tổng hợp.
Include	Không
Extend	UC-EVAL-01: Gửi đánh giá
Trigger	Sinh viên chọn Sửa tại bản đánh giá của buổi học.
Normal Flows	<ol style="list-style-type: none"> 1. Mở chi tiết đánh giá, chọn "Sửa". 2. Cập nhật điểm/nhận xét, chọn "Lưu". 3. Hệ thống lưu phiên bản mới, hệ thống cập nhật dữ liệu tổng hợp (gọi use-case UC-EVAL-07).
Alternative Flows	2.1. Hủy chỉnh sửa → quay lại, không thay đổi dữ liệu.
Exception Flows	1.1. Hết thời hạn sửa đánh giá → chặn thao tác.

Bảng 3.31: Bảng đặc tả Use-case: Xem lịch sử đánh giá (UC-EVAL-03)

Tên use-case	UC-EVAL-03: Xem lịch sử đánh giá
Actors	Sinh viên
Description	Xem danh sách các đánh giá đã gửi, kèm trạng thái đã gửi, đã sửa, hết hạn.
Precondition	Sinh viên đã đăng nhập.
Postcondition	Danh sách hiển thị theo thời gian, hỗ trợ lọc và tìm kiếm.
Include	Không
Extend	Không
Trigger	Sinh viên mở mục “Lịch sử đánh giá”.
Normal Flows	<ol style="list-style-type: none"> 1. Mở Lịch sử đánh giá → tải danh sách. 2. (Tùy chọn) Lọc theo môn/tutor/khoảng ngày (nếu cần) → danh sách cập nhật. 3. Chọn bản ghi → xem chi tiết.
Alternative Flows	Không
Exception Flows	3.1. Chưa có dữ liệu, hệ thống hiển thị thông báo phù hợp.

Bảng 3.32: Bảng đặc tả Use-case: Ghi nhận tiền bộ (UC-EVAL-04)

Tên use-case	UC-EVAL-04: Ghi nhận tiền bộ
Actors	Tutor
Description	Ghi đánh giá/nhận xét đạt hoặc không, đánh dấu rủi ro nếu có, ghi chú và kế hoạch buổi học sau.
Precondition	Tutor đã đăng nhập và là tutor phụ trách buổi học.
Postcondition	Ghi nhận được lưu và hệ thống cập nhật dữ liệu tổng hợp.
Include	UC-EVAL-07: Cập nhật dữ liệu tổng hợp
Extend	Không
Trigger	Tutor mở form “Ghi nhận buổi học” sau khi buổi học kết thúc.
Normal Flows	<ol style="list-style-type: none"> Mở form ghi nhận của buổi học → nhập đánh giá/nhận xét, rủi ro, ghi chú, kế hoạch. Nhấn Lưu → hệ thống cập nhật dữ liệu tổng hợp (gọi UC-EVAL-07).
Alternative Flows	2.1. Lưu bản nháp để hoàn tất sau.
Exception Flows	1.1. Thiếu trường trọng yếu (ví dụ đánh giá/nhận xét) → báo lỗi và yêu cầu bổ sung.

Bảng 3.33: Bảng đặc tả Use-case: Cập nhật ghi nhận (UC-EVAL-05)

Tên use-case	UC-EVAL-05: Cập nhật ghi nhận
Actors	Tutor
Description	Chỉnh sửa nội dung ghi nhận của buổi học sau khi đã lưu trước đó.
Precondition	Có ghi nhận hiện hữu và còn quyền chỉnh sửa.
Postcondition	Ghi nhận được cập nhật và hệ thống cập nhật lại dữ liệu tổng hợp.
Include	Không
Extend	UC-EVAL-04: Ghi nhận tiền bộ
Trigger	Tutor chọn “Cập nhật ghi nhận” của buổi học.
Normal Flows	<ol style="list-style-type: none"> Mở ghi nhận → Cập nhật. Sửa nội dung → Lưu. Hệ thống lưu phiên bản mới → cập nhật dữ liệu tổng hợp (gọi đến use-case UC-EVAL-07).
Alternative Flows	2.1. Huỷ cập nhật ghi nhận → quay lại, không thay đổi dữ liệu.
Exception Flows	1.1. Ghi nhận đã khoá → chỉ cho xem.

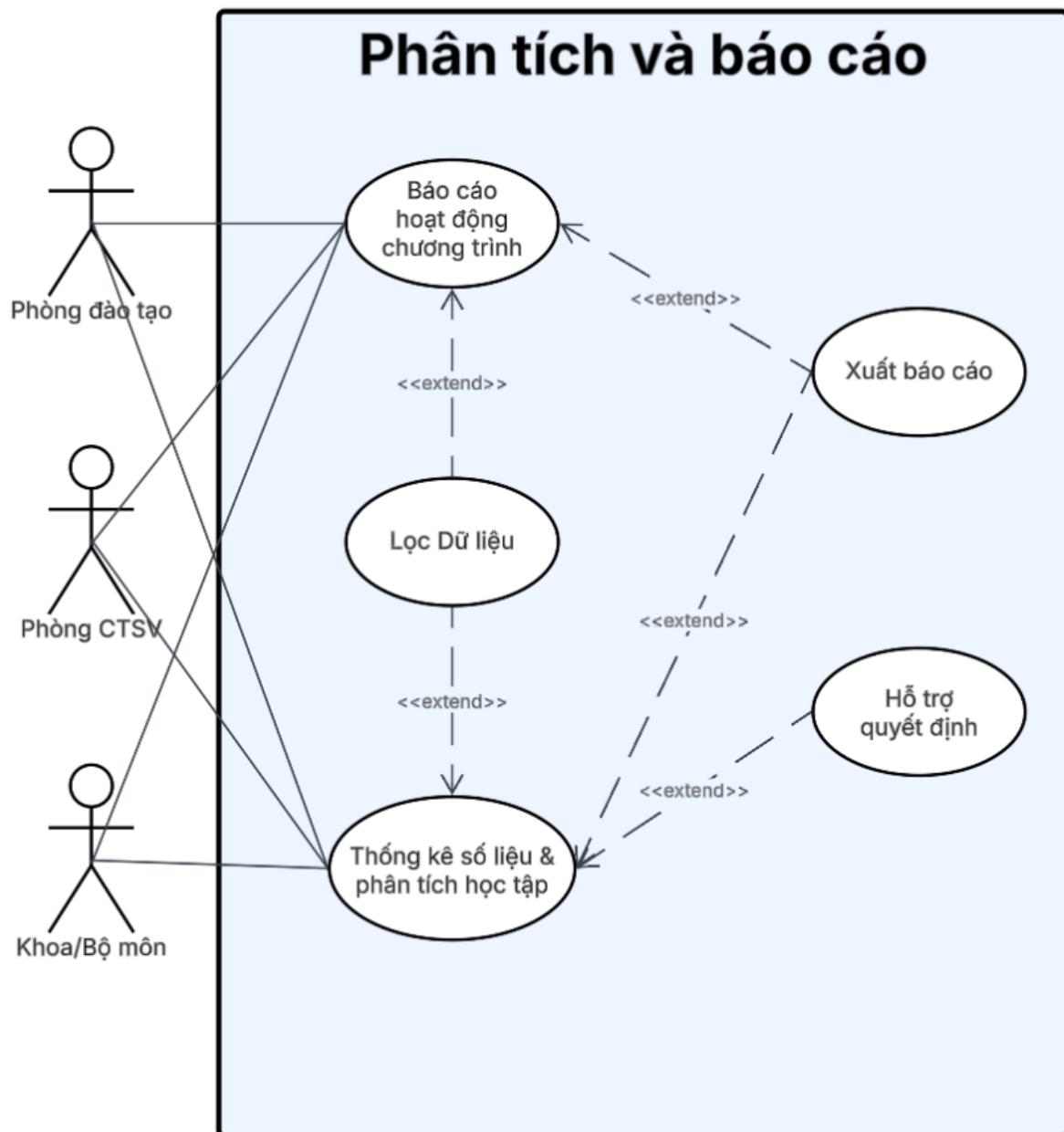
Bảng 3.34: Bảng đặc tả Use-case: Xem nhật kí tiền độ (UC-EVAL-06)

Tên use-case	UC-EVAL-06: Xem nhật kí tiền độ
Actors	Tutor
Description	Xem nhật kí ghi nhận theo buổi học
Precondition	Tutor đã đăng nhập và có quyền xem lớp/nhóm tương ứng.
Postcondition	Danh sách hiển thị có phân trang, bộ lọc.
Include	Không
Extend	Không
Trigger	Tutor mở mục “Nhật kí tiền độ”.
Normal Flows	<ol style="list-style-type: none"> 1. Mở “Nhật kí tiền độ” → tải danh sách mặc định theo thời gian gần nhất. 2. Tutor có thể chọn bộ lọc (lớp/sinh viên/...) → danh sách cập nhật. 3. Chọn một mục → xem chi tiết ghi nhận của buổi học.
Alternative Flows	Không
Exception Flows	1.1. Không có quyền xem lớp đã chọn → chặn truy cập và hiển thị thông báo. 2.1. Chưa có dữ liệu phù hợp → hiển thị thông báo.

Bảng 3.35: Bảng đặc tả Use-case: Cập nhật dữ liệu tổng hợp (UC-EVAL-07)

Tên use-case	UC-EVAL-07: Cập nhật dữ liệu tổng hợp
Actors	Không
Description	Tự động tổng hợp dữ liệu từ Gửi đánh giá, Sửa đánh giá, Ghi nhận tiền bộ, Cập nhật ghi nhận để tạo dữ liệu tổng hợp: điểm trung bình đánh giá, số lượng đánh giá, tỉ lệ hoàn thành đánh giá, số buổi học.
Precondition	Có dữ liệu đầu vào hợp lệ từ các use-case liên quan.
Postcondition	Dữ liệu tổng hợp được cập nhật và các trang tra cứu hoặc báo cáo đọc được dữ liệu mới.
Include	Không
Extend	Không
Trigger	Sự kiện lưu thành công từ các use-case UC-EVAL-01/UC-EVAL-02 hoặc UC-EVAL-04/UC-EVAL-05 .
Normal Flows	<ol style="list-style-type: none"> 1. Nhận sự kiện đã lưu thành công → truy xuất dữ liệu liên quan. 2. Tính toán và cập nhật dữ liệu tổng hợp theo cấu hình → lưu kết quả theo buổi học/lớp/tutor.
Alternative Flows	Không
Exception Flows	2.1. Lỗi tính toán hoặc lưu → ghi log và thử lại theo chính sách, không chặn thao tác người dùng.

3.7 Phân tích & báo cáo



Hình 3.7: Sơ đồ use-case cho chức năng phân tích và báo cáo

Bảng 3.36: Bảng đặc tả Use-case: Báo cáo hoạt động chương trình (UC-ANLY-01)

Tên use-case	UC-ANLY-01: Báo cáo hoạt động chương trình
Actor	Phòng đào tạo, Phòng Công tác sinh viên, Khoa/Bộ môn
Description	Actor có thể xem báo cáo tổng hợp về hoạt động chương trình (số buổi tư vấn, tỷ lệ tham gia, số sinh viên/tutor tham gia).
Include	Không
Extend	UC-ANLY-03: Lọc dữ liệu UC-ANLY-04: Xuất báo cáo
Precondition	Actor đã đăng nhập thành công và có quyền xem báo cáo.
Postcondition	Báo cáo hiển thị thành công trên giao diện.
Trigger	Actor chọn chức năng “Báo cáo hoạt động chương trình”.
Normal Flows	<ol style="list-style-type: none"> 1. Actor chọn chức năng “Báo cáo”. 2. Hệ thống gửi yêu cầu truy xuất dữ liệu tới database. 3. Database trả dữ liệu tổng hợp. 4. Hệ thống hiển thị báo cáo tổng hợp.
Alternative Flows	<p>1.1. Actor chọn “Lọc dữ liệu” và hệ thống gọi use-case UC-ANLY-03.</p> <p>4.1. Actor chọn “Xuất báo cáo” và hệ thống gọi use-case UC-ANLY-04.</p>
Exception Flows	<p>2.1. Lỗi kết nối Cơ sở dữ liệu: Hệ thống thông báo lỗi.</p> <p>3.1. Không có dữ liệu: Hệ thống hiển thị “Chưa có dữ liệu”.</p>

Bảng 3.37: Bảng đặc tả Use-case: Thống kê số liệu & phân tích học tập (UC-ANLY-02)

Tên use-case	UC-ANLY-02: Thống kê số liệu & phân tích học tập
Actors	Phòng đào tạo, Phòng Công tác sinh viên, Khoa/Bộ môn
Description	Actor có thể xem thống kê chi tiết về số liệu học tập của sinh viên theo môn học/lĩnh vực.
Include	UC-ANLY-04: Xuất báo cáo
Extend	UC-ANLY-03: Lọc dữ liệu UC-ANLY-05: Hỗ trợ quyết định
Precondition	Actor đã đăng nhập và có quyền truy cập dữ liệu thống kê.
Postcondition	Hệ thống hiển thị bảng/biểu đồ thống kê.
Trigger	Actor chọn chức năng “Thống kê số liệu & phân tích học tập”.
Normal Flows	<ol style="list-style-type: none"> 1. Actor chọn “Thống kê số liệu”. 2. Hệ thống gửi yêu cầu truy xuất dữ liệu chi tiết. 3. Database gửi dữ liệu về cho hệ thống. 4. Hệ thống hiển thị thống kê.
Alternative Flows	1.1. Actor chọn “Lọc dữ liệu” và hệ thống gọi use-case UC-ANLY-03 . 4.1. Actor chọn “Xuất báo cáo” và hệ thống gọi use-case UC-ANLY-04 . 4.2. Actor chọn “Hỗ trợ quyết định” và hệ thống gọi use-case UC-ANLY-05 .
Exception Flows	3.1. Không có dữ liệu: Hệ thống hiển thị “Chưa có dữ liệu”.

Bảng 3.38: Bảng đặc tả Use-case: Bộ lọc dữ liệu (UC-ANLY-03)

Tên use-case	UC-ANLY-03: Lọc dữ liệu
Actors	Phòng Đào tạo, Phòng Công tác sinh viên, Khoa/Bộ môn
Description	Cho phép actor chọn các tiêu chí (theo thời gian, khoa/bộ môn, loại báo cáo,...) để giới hạn và tinh chỉnh dữ liệu hiển thị trong báo cáo/thống kê.
Include	Không
Extend	Không
Precondition	Actor đã đăng nhập và đang trong quá trình xem báo cáo/thống kê.
Postcondition	Hệ thống hiển thị dữ liệu đã được lọc theo tiêu chí actor chọn.
Trigger	Actor chọn chức năng “Lọc dữ liệu”.
Normal flows	<ol style="list-style-type: none"> Actor nhấp vào nút/biểu tượng lọc trên giao diện. Hệ thống hiển thị các tiêu chí lọc. Actor chọn các tiêu chí mong muốn. Hệ thống áp dụng bộ lọc và hiển thị kết quả.
Alternative flows	Không
Exception flows	Không

Bảng 3.39: Bảng đặc tả Use-case: Xuất báo cáo (UC-ANLY-04)

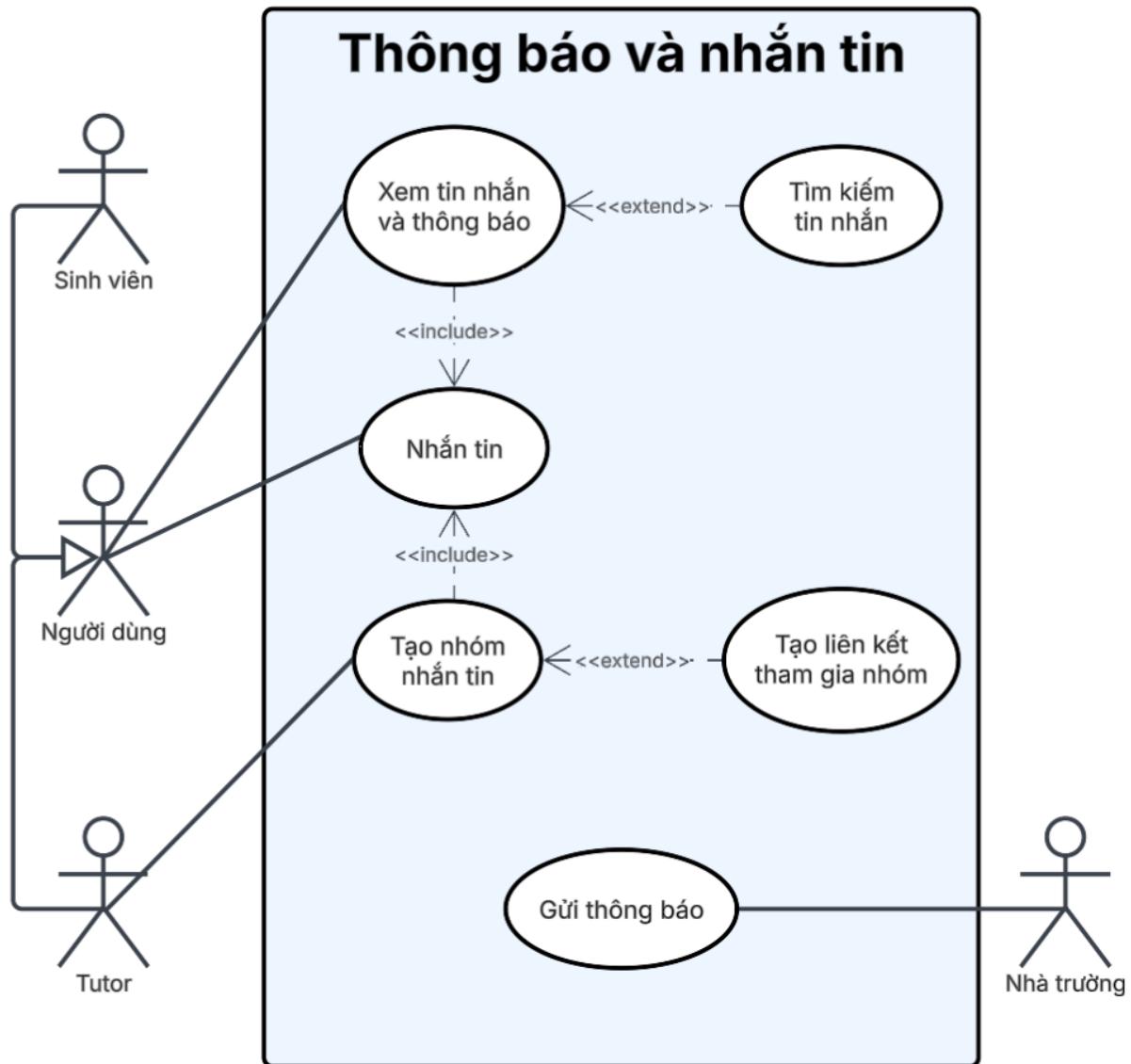
Tên use-case	UC-ANLY-04: Xuất báo cáo
Actors	Phòng Đào tạo, Phòng Công tác sinh viên, Khoa/Bộ môn
Description	Actor có thể xuất báo cáo ra các định dạng PDF, Excel,... với tùy chọn bộ lọc.
Include	Không
Extend	Không
Precondition	Actor đăng nhập và có quyền truy cập.
Postcondition	File báo cáo được tải xuống thành công.
Trigger	Actor chọn chức năng “Xuất báo cáo”.
Normal Flows	<ol style="list-style-type: none"> Actor mở mục “Báo cáo”. Actor chọn định dạng xuất. Hệ thống truy xuất dữ liệu từ database. Hệ thống tạo báo cáo và gửi tệp báo cáo cho Actor. Actor có thể tải tệp báo cáo về máy.
Alternative Flows	2.1. Nếu Actor để mặc định (không chỉ định định dạng), hệ thống xuất ra PDF.
Exception Flows	Không



Bảng 3.40: Bảng đặc tả Use-case: Hỗ trợ quyết định (UC-ANLY-05)

Tên use-case	UC-ANLY-05: Hỗ trợ quyết định
Actors	Phòng đào tạo, Phòng Công tác sinh viên
Description	Hệ thống cung cấp dữ liệu phân tích để phục vụ ra quyết định (xét điểm rèn luyện, học bỗng, phân bổ nguồn lực tutor).
Include	Không
Extend	Không
Precondition	Actor đã đăng nhập và có quyền ra quyết định.
Postcondition	Actor nhận được dữ liệu/khuyến nghị từ hệ thống.
Trigger	Actor chọn chức năng “Hỗ trợ quyết định”.
Normal Flows	<ol style="list-style-type: none"> 1. Hệ thống hiển thị mục tiêu hỗ trợ. 2. Actor chọn mục tiêu cần hỗ trợ. 3. Hệ thống truy xuất dữ liệu thống kê liên quan từ database. 4. Database trả dữ liệu. 6. Hệ thống hiển thị kết quả phân tích và quyết định.
Alternative Flows	Không
Exception Flows	Không

3.8 Thông báo và nhắn tin



Hình 3.8: Sơ đồ use-case cho chức năng Thông báo và nhắn tin

Bảng 3.41: Bảng đặc tả Use-case: Xem tin nhắn và thông báo (UC-CHAT-01)

Use-case	UC-CHAT-01: Xem tin nhắn và thông báo.
Actor	Người dùng (Sinh viên/Tutor).
Descriptions	Sinh viên có thể xem thông báo/tin nhắn từ Tutor/Nhà trường; Tutor có thể xem tin nhắn từ sinh viên và thông báo từ Nhà trường.
Include	Không
Extend	UC-CHAT-02: Tìm kiếm tin nhắn.
Precondition	Người dùng đã đăng nhập thành công.
Postcondition	Hiển thị danh sách các tin nhắn và thông báo từ người dùng/Nhà trường.
Trigger	Người dùng chọn icon "Tin nhắn" ở trang chính.
Normal Flows	<ol style="list-style-type: none"> 1. Người dùng nhấn vào biểu tượng "Tin nhắn" trên màn hình trang chủ. 2. Hệ thống hiển thị danh sách các tin nhắn. 3. Người dùng có thể nhấn biểu tượng "Tìm kiếm" và nhập tên Người dùng khác/Nhóm để tìm tin nhắn cần xem → hệ thống gọi UC-CHAT-02. 4. Người dùng có thể chọn Người dùng khác/nhóm với tin nhắn cần xem. 5. Hệ thống chuyển vào hộp thoại nhắn tin.
Alternative Flows	2.1 Không có tin nhắn hay thông báo → hệ thống hiện "Không có tin nhắn". 3.1 Người dùng thoát khỏi danh sách tin nhắn mà không cần xem tin nhắn.
Exception Flows	2.1. Lỗi tải dữ liệu tin nhắn/thông báo, hệ thống in ra lỗi phù hợp.



Bảng 3.42: Bảng đặc tả Use-case: Nhắn tin (UC-CHAT-02)

Use-case	UC-CHAT-03: Nhắn tin.
Actor	Người dùng (Sinh viên/Tutor).
Descriptions	Người dùng nhắn tin trên hộp thoại tin nhắn.
Include	UC-CHAT-01 Xem tin nhắn và thông báo. UC-CHAT-04 Tạo nhóm nhắn tin.
Extend	Không
Precondition	1. Người dùng đã đăng nhập thành công. 2. Người dùng đã chọn một hộp thoại nhắn tin từ danh sách tin nhắn xuất hiện ở UC-CHAT-01 . 3. Nếu nhắn tin trên nhóm thì nhóm phải được tạo từ Tutor bằng UC-CHAT-04 .
Postcondition	Tin nhắn được gửi thành công.
Trigger	Không
Normal Flows	1. Người dùng chọn một cuộc hội thoại cần nhắn tin trong danh sách tin nhắn xuất hiện ở UC-CHAT-01 . 2. Hệ thống hiển thị hộp thoại để nhắn tin. 3. Người dùng nhập nội dung tin nhắn. 4. Người dùng nhấn biểu tượng "Gửi" để gửi nội dung tin nhắn.
Alternative Flows	2.1 Người dùng không nhắn tin, thoát khỏi hộp thoại. 4.1 Người dùng không muốn gửi tin nhắn, xóa tất cả và thoát.
Exception Flows	2.1. Lỗi tải dữ liệu tin nhắn/thông báo, hệ thống in ra lỗi phù hợp.

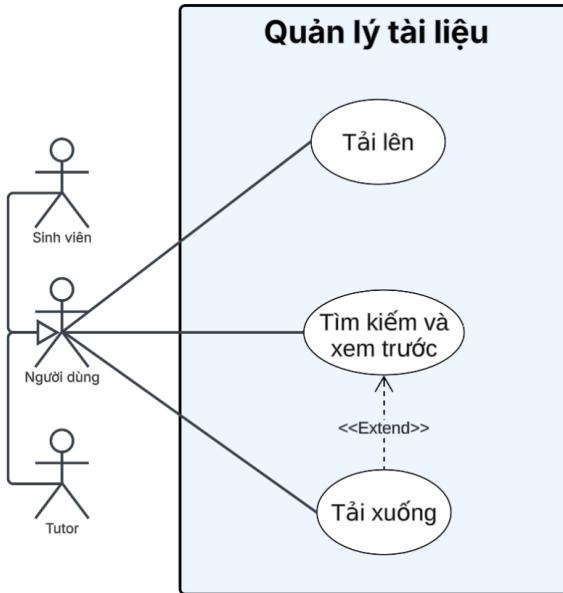
Bảng 3.43: Bảng đặc tả Use-case: Tạo nhóm nhắn tin (UC-CHAT-04)

Use-case	UC-CHAT-04: Tạo nhóm.
Actor	Tutor.
Descriptions	Tutor tạo một nhóm để nhắn tin cho lớp học của mình.
Include	Không
Extend	UC-CHAT-05: Tạo liên kết tham gia nhóm.
Precondition	1. Tutor đã đăng nhập thành công.
Postcondition	Một nhóm nhắn tin mới được tạo.
Trigger	Biểu tượng "Tạo nhóm".
Normal Flows	<ol style="list-style-type: none"> 1. Tutor nhấn vào biểu tượng "Tạo nhóm" trên hộp danh sách tin nhắn. 2. Hệ thống hiển thị hộp thoại "Tạo nhóm mới". 3. Tutor đặt tên và chọn các sinh viên để thêm vào nhóm. 4. Nếu số lượng sinh viên quá đông Tutor có thể nhấn vào biểu tượng "Chia sẻ link tham gia nhóm", hệ thống sẽ gọi UC-CHAT-05 và tạo một liên kết tham gia nhóm và Tutor có thể chia sẻ trên bất cứ nền tảng nào. 5. Tutor nhấn "Tạo nhóm" và nhóm mới sẽ được tạo.
Alternative Flows	5.1 Tutor không tạo nhóm, thoát hộp thoại → quay về danh sách tin nhắn.
Exception Flows	5.1. Lỗi tạo nhóm → Hệ thống thông báo lỗi và quay về danh sách tin nhắn.

Bảng 3.44: Bảng đặc tả Use-case: Gửi thông báo (UC-CHAT-06)

Use-case	UC-CHAT-06: Gửi thông báo.
Actor	Nhà trường.
Descriptions	Nhà trường gửi thông báo chung cho sinh viên/Tutor.
Include	Không
Extend	Không
Precondition	Nhà trường đã đăng nhập thành công vào tài khoản.
Postcondition	Thông báo được gửi đi cho sinh viên/Tutor.
Trigger	Biểu tượng "Gửi thông báo".
Normal Flows	<ol style="list-style-type: none"> 1. Nhà trường nhấn vào biểu tượng "Gửi thông báo" trên màn hình chính. 2. Hệ thống hiển thị hộp thoại gửi thông báo. 3. Nhà trường nhập nội dung cần thông báo. 4. Nhà trường nhấn biểu tượng "Gửi".
Alternative Flows	4.1 Nhà trường không gửi thông báo → quay về màn hình chính.
Exception Flows	4.1. Lỗi gửi thông báo → Hệ thống thông báo lỗi và quay về màn hình chính.

3.9 Quản lý tài liệu



Hình 3.9: Sơ đồ use-case cho chức năng Quản lý tài liệu

Bảng 3.45: Bảng đặc tả Use-case: Tải lên (UC-LIB-01)

Use-case	UC-LIB-01: Tải lên.
Actor	Sinh viên, Tutor.
Description	Tải tài liệu (pdf, pptx, docx, img, zip) lên hệ thống; chọn quyền xem.
Include	Không
Extend	Không
Precondition	File có định dạng hợp lệ; kích thước 15 MB.
Postcondition	File cùng với metadata, version của file được lưu.
Trigger	Actor nhấn nút “Tải lên tài liệu”.
Normal flow	<ol style="list-style-type: none"> Actor chọn file và nhập tiêu đề, môn, quyền. Hệ thống kiểm tra định dạng và kích thước. Nếu hợp lệ → lưu file, tạo metadata và version cho file. Trả thông báo “Tải lên thành công”.
Alternative flow	Không
Exceptions flow	<p>2.1. Lỗi lưu file (kích thước file lớn) → hiển thị lỗi “Tải lên thất bại”.</p> <p>2.2. Định dạng không hợp lệ → thông báo định dạng cho phép.</p> <p>3.1. Mất kết nối giữa chừng → Tải lên thất bại, không tạo metadata và version.</p> <p>3.2. Lỗi lưu metadata → không tải file lên, thông báo lỗi "Lỗi lưu file"</p>

Bảng 3.46: Bảng đặc tả Use-case: Tìm kiếm và xem trước (UC-LIB-02)

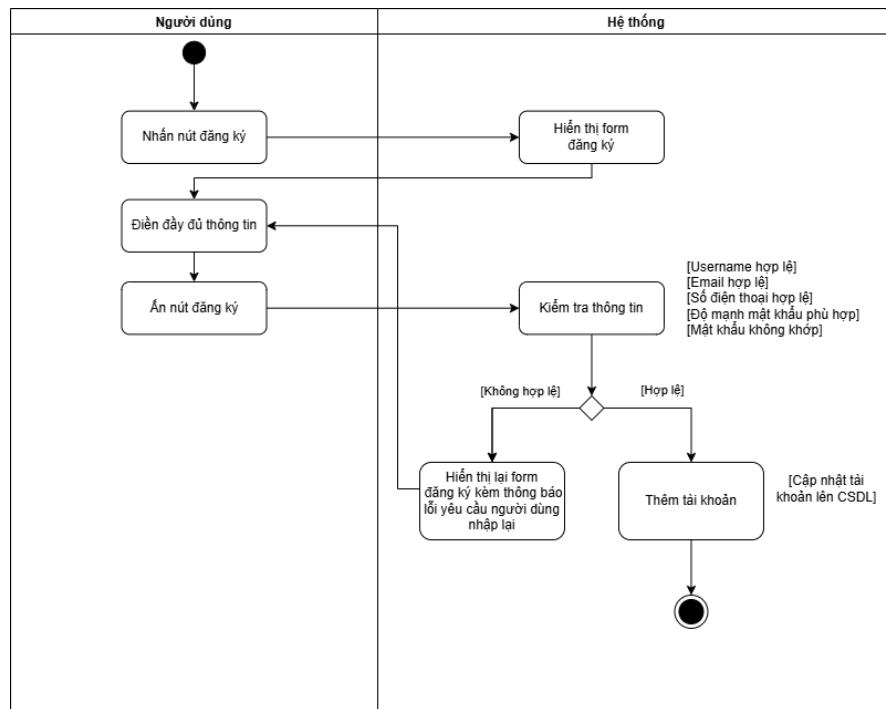
Use-case	UC-LIB-02: Tìm kiếm và xem trước.
Actor	Sinh viên, Tutor.
Description	Tìm tài liệu theo từ khóa / mã môn và xem bản preview; nếu có quyền tải xuống thì hiển thị nút tải.
Include	Không
Extend	UC-LIB-03: Tải xuống.
Precondition	Tồn tại tài liệu phù hợp trong hệ thống.
Postcondition	Hiện màn hình xem trước tài liệu.
Trigger	Nhấn vào tài liệu để xem.
Normal flow	<ol style="list-style-type: none"> Actor nhập từ khóa / chọn bộ lọc. Hệ thống trả danh sách kết quả. Actor click vào tài liệu.
Alternative flow	<p>1.1. Không tìm thấy kết quả → hiển thị “Không tìm thấy”.</p> <p>3.1. Actor có thêm quyền tải xuống → hiện nút download.</p>
Exceptions flow	<p>1.1. Lỗi truy vấn tìm kiếm (timeout) → hiển thị thông báo “Tìm kiếm tạm thời không khả dụng”.</p> <p>3.1. Lỗi preview (file hỏng) → hiển thị thông báo “Không thể xem tài liệu”.</p>

Bảng 3.47: Bảng đặc tả Use-case: Tải xuống (UC-LIB-03)

Use-case	UC-LIB-03: Tải xuống.
Actor	Sinh viên, Tutor.
Description	Tải file tài liệu về máy để xem offline; hệ thống kiểm tra quyền.
Include	Không
Extend	Không
Precondition	Tài liệu tồn tại; actor có quyền Tải xuống.
Postcondition	File được gửi cho actor.
Trigger	Actor nhấn nút “Tải xuống” trên trang chi tiết tài liệu.
Normal flow	<ol style="list-style-type: none"> Actor nhấn “Download”. Kiểm tra quyền tải xuống của người dùng. Nếu có quyền → tải file về máy actor.
Alternative flow	Không
Exceptions flow	<p>1.1. File bị thiếu/không tìm thấy → hiện “Tài liệu tạm thời không khả dụng”.</p> <p>2.1. Nếu file không cho tải, thông báo “Tài liệu không cho phép tải xuống”.</p> <p>3.1. Lỗi truyền file (timeout) → hiển thị lỗi, actor thử lại.</p>

4 Activity Diagram

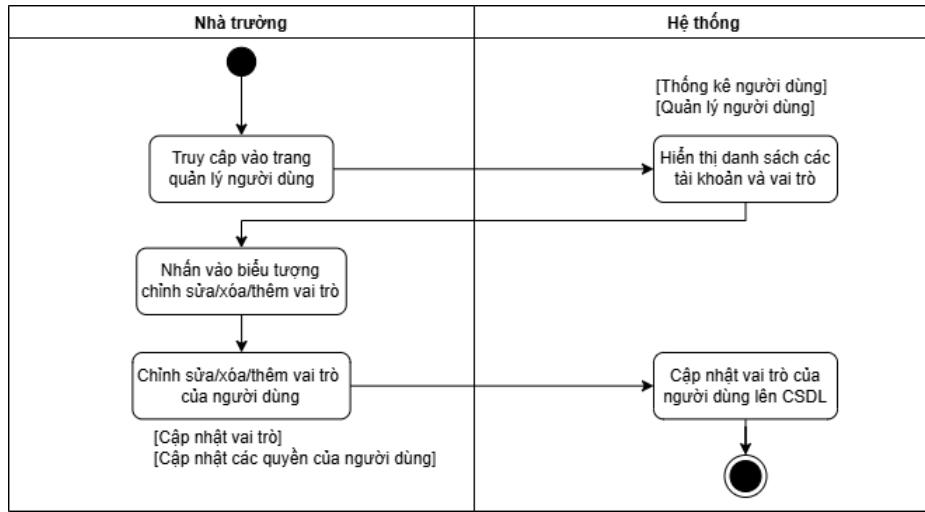
4.1 Quản lý truy cập và xác thực



Hình 4.1: Sơ đồ hoạt động cho use-case "Đăng ký"

Mô tả sơ đồ hoạt động: Đăng ký

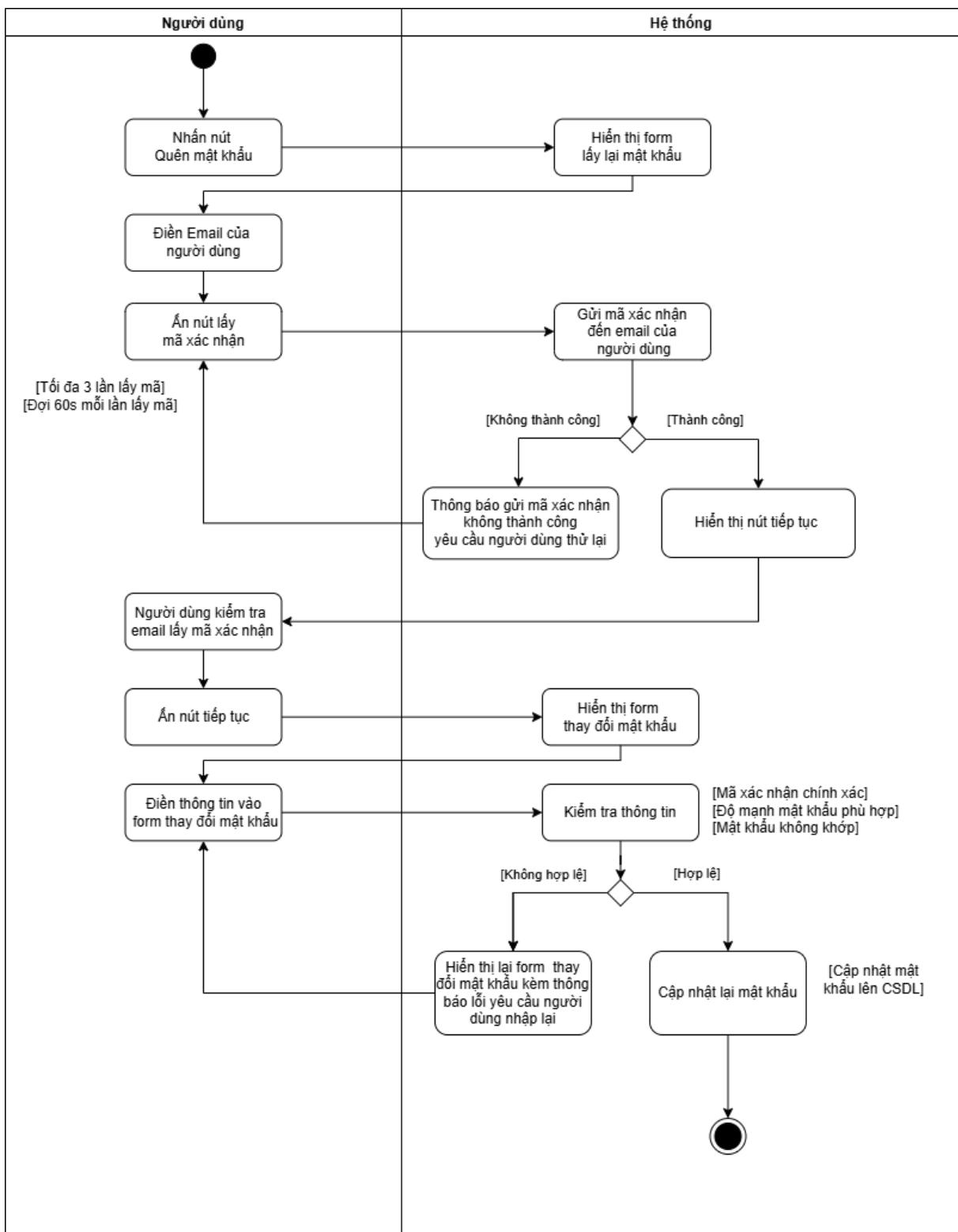
- **Người dùng** chọn nút “Đăng ký”.
- **Hệ thống** hiển thị form đăng ký.
- **Người dùng** nhập đầy đủ các thông tin yêu cầu như tên, email, số điện thoại, username và mật khẩu.
- **Người dùng** nhấn nút “Đăng ký”.
- **Hệ thống** kiểm tra thông tin:
 - Username hợp lệ.
 - Email hợp lệ.
 - Số điện thoại hợp lệ.
 - Mật khẩu đạt yêu cầu.
 - Nhập lại mật khẩu trùng khớp.
- Nếu thông tin không hợp lệ: hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.
- Nếu hợp lệ: hệ thống tạo tài khoản mới và lưu vào cơ sở dữ liệu.



Hình 4.2: Sơ đồ hoạt động cho use-case "Quản lý tài khoản"

Mô tả sơ đồ hoạt động: Quản lý tài khoản

- **Nhà trường** truy cập trang quản lý tài khoản người dùng.
- **Hệ thống** hiển thị danh sách tài khoản và vai trò tương ứng.
- **Nhà trường** chọn thao tác chỉnh sửa, xoá hoặc thêm vai trò.
- **Nhà trường** cập nhật vai trò hoặc quyền của tài khoản.
- **Hệ thống** ghi nhận thay đổi và cập nhật lên cơ sở dữ liệu.

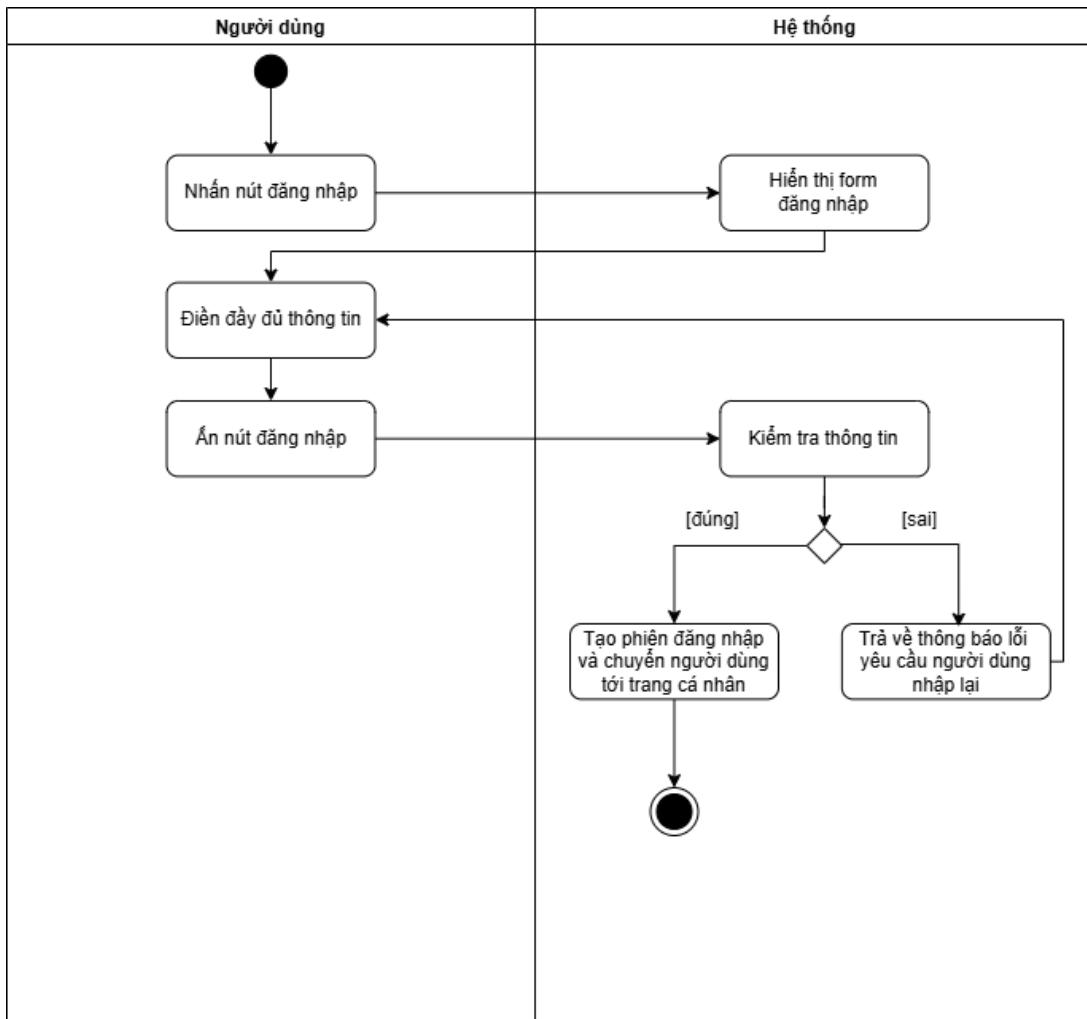


Hình 4.3: Sơ đồ hoạt động cho use-case "Đặt lại mật khẩu"



Mô tả sơ đồ hoạt động: Đặt lại mật khẩu

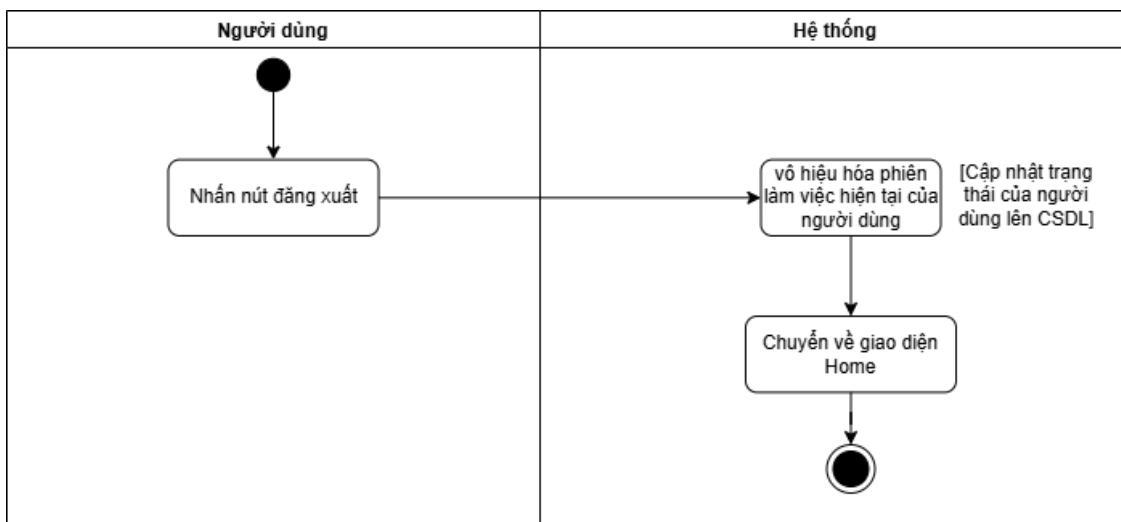
- **Người dùng** nhấn nút “Quên mật khẩu”.
- **Hệ thống** hiển thị form yêu cầu nhập email.
- **Người dùng** điền email của mình.
- **Người dùng** nhấn nút “Lấy mã xác nhận”.
- **Hệ thống** gửi mã xác nhận đến email của người dùng.
- Nếu gửi mã thất bại: hệ thống hiển thị thông báo và yêu cầu người dùng thực hiện lại.
- Nếu gửi mã thành công: hiển thị nút “Tiếp tục”.
- **Người dùng** kiểm tra email và lấy mã xác nhận.
- **Người dùng** nhấn nút “Tiếp tục”.
- **Hệ thống** hiển thị form đặt lại mật khẩu.
- **Người dùng** nhập mật khẩu mới và mã xác nhận.
- **Hệ thống** kiểm tra:
 - Mã xác nhận hợp lệ.
 - Mật khẩu mới mạnh và hợp lệ.
 - Nhập lại mật khẩu trùng khớp.
- Nếu thông tin không hợp lệ: hiển thị thông báo lỗi và yêu cầu nhập lại.
- Nếu hợp lệ: cập nhật mật khẩu mới vào cơ sở dữ liệu.



Hình 4.4: Sơ đồ hoạt động cho use-case "Đăng nhập"

Mô tả sơ đồ hoạt động: Đăng nhập

- **Người dùng** nhấn nút “Đăng nhập” trên giao diện hệ thống.
- **Hệ thống** hiển thị form đăng nhập.
- **Người dùng** nhập thông tin đăng nhập gồm email/username và mật khẩu.
- **Người dùng** nhấn nút “Đăng nhập”.
- **Hệ thống** kiểm tra thông tin:
 - Nếu thông tin không chính xác: trả về thông báo lỗi và yêu cầu nhập lại.
 - Nếu thông tin chính xác: tạo phiên đăng nhập cho người dùng.
- **Hệ thống** chuyển người dùng đến trang cá nhân.

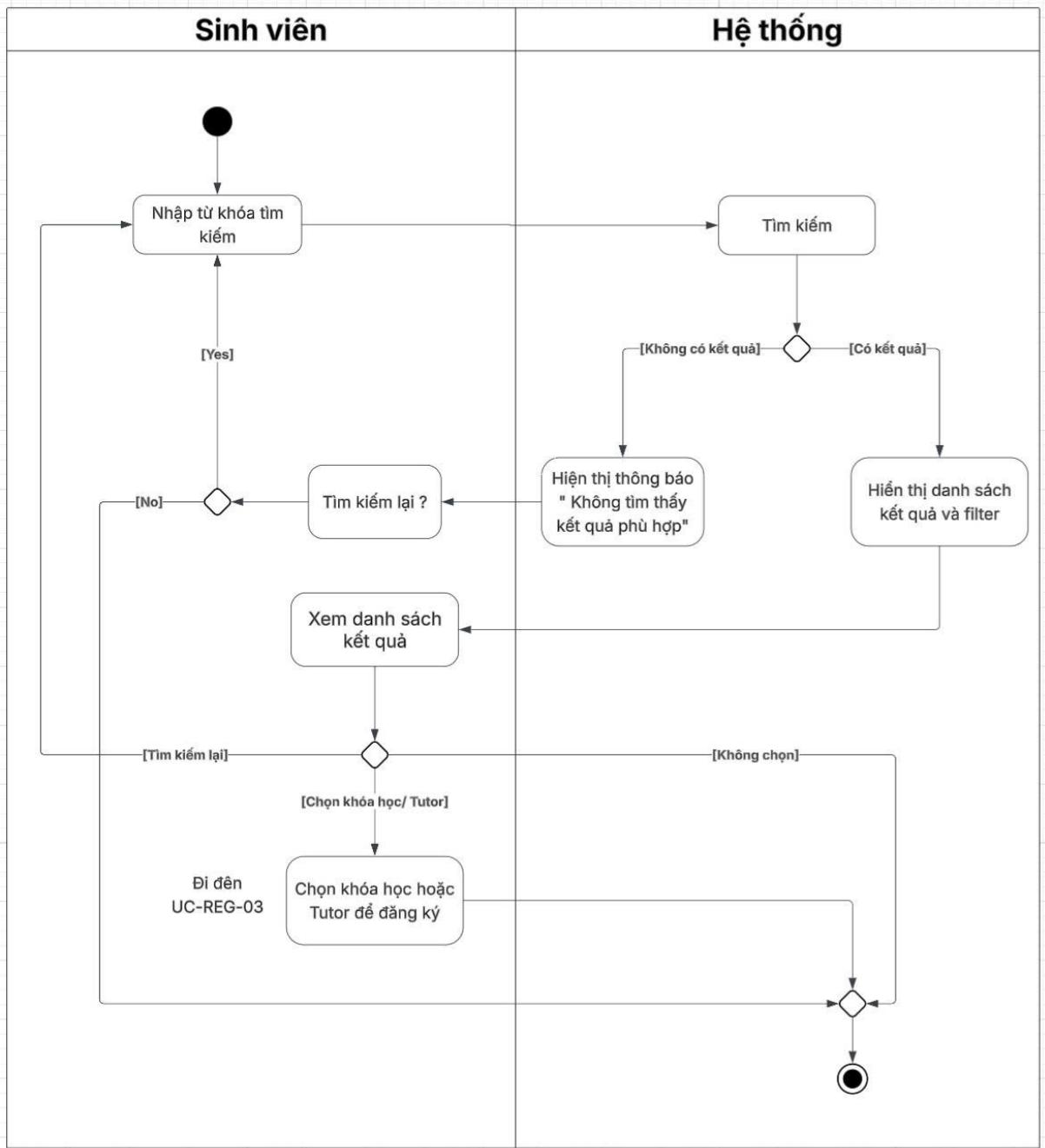


Hình 4.5: Sơ đồ hoạt động cho use-case "Đăng xuất"

Mô tả sơ đồ hoạt động: Đăng xuất

- **Người dùng** nhấn nút “Đăng xuất”.
- **Hệ thống** vô hiệu hóa phiên làm việc hiện tại.
- **Hệ thống** cập nhật trạng thái đăng xuất vào cơ sở dữ liệu.
- **Hệ thống** chuyển người dùng về giao diện Home.

4.2 Đăng ký chương trình



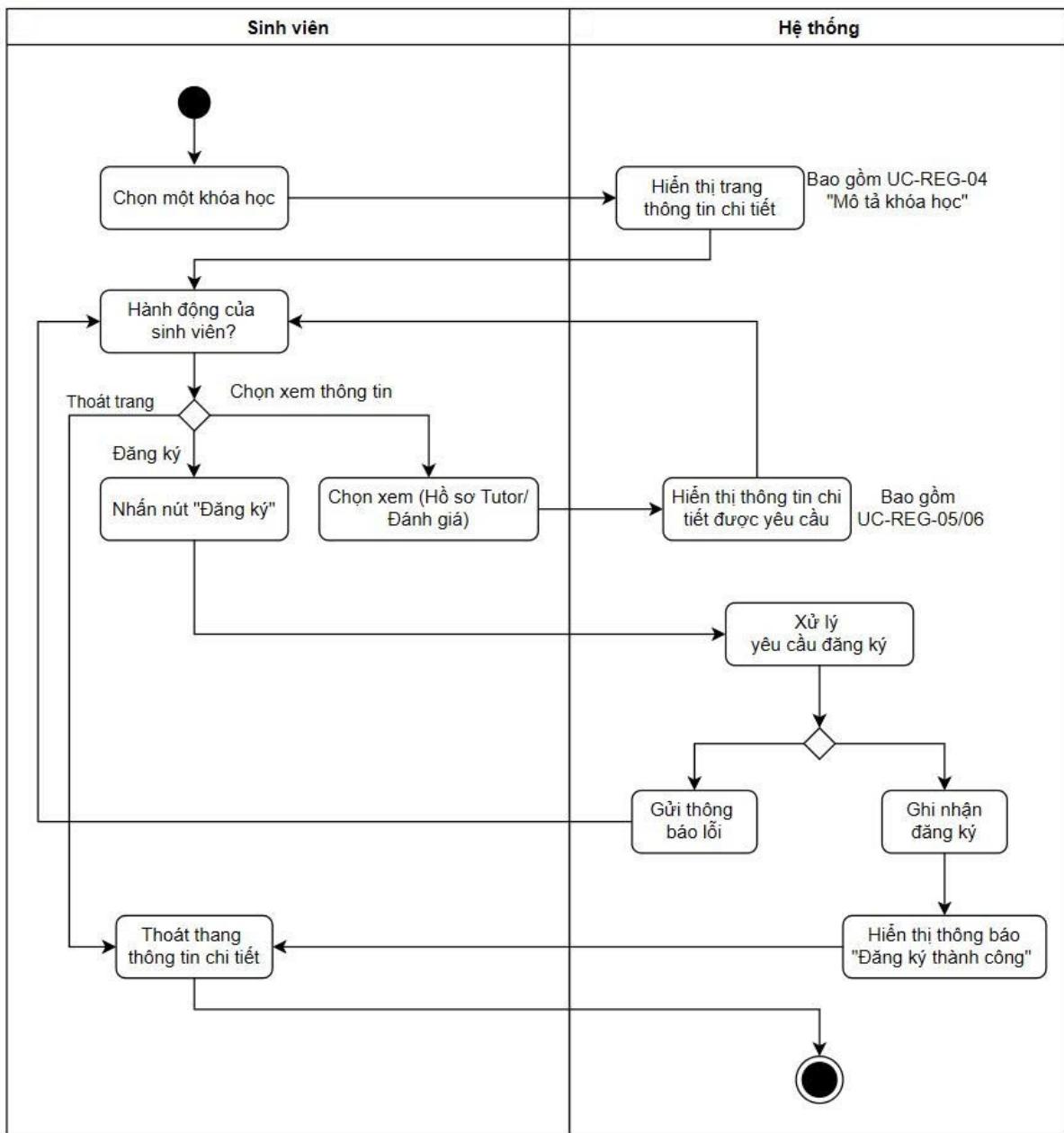
Hình 4.6: Sơ đồ hoạt động cho use-case "Tìm kiếm chương trình"



Mô tả sơ đồ hoạt động: Tìm kiếm chương trình

Sơ đồ này mô tả luồng nghiệp vụ khi sinh viên thực hiện chức năng tìm kiếm khóa học hoặc Tutor trên hệ thống.

- **Bắt đầu:** Luồng được kích hoạt khi Sinh viên bắt đầu tìm kiếm.
- **Nhập liệu:** Sinh viên thực hiện hành động "Nhập từ khóa tìm kiếm".
- **Hệ thống Xử lý:** Hệ thống tiếp nhận từ khóa và thực hiện nghiệp vụ "Tìm kiếm".
- **Rẽ nhánh (Kết quả tìm kiếm):** Sau khi tìm kiếm, Hệ thống rẽ nhánh dựa trên kết quả:
 - Nếu "Không có kết quả": Hệ thống "Hiển thị thông báo 'Không tìm thấy kết quả phù hợp'". Luồng chuyển đến bước 5.
 - Nếu "Có kết quả": Hệ thống "Hiển thị danh sách kết quả và filter". Luồng chuyển đến bước 6.
- **Tìm kiếm lại (Sau khi không có kết quả):** Sinh viên nhận thông báo và quyết định "Tìm kiếm lại?".
 - Nếu "Yes": Quay lại bước 2, "Nhập từ khóa tìm kiếm".
 - Nếu "No": Luồng kết thúc.
- **Hành động của Sinh viên (Sau khi có kết quả):** Sinh viên "Xem danh sách kết quả". Tại đây, sinh viên có 3 lựa chọn:
 - "Tìm kiếm lại": Quay về bước 2, "Nhập từ khóa tìm kiếm".
 - "Chọn khóa học/Tutor": Sinh viên chọn một mục cụ thể. Hành động này sẽ kích hoạt Use Case UC-REG-03. Luồng kết thúc.
 - "Không chọn": Sinh viên không tương tác gì thêm. Luồng kết thúc.
- **Kết thúc:** Luồng dừng lại khi sinh viên thoát hoặc chuyển sang một Use Case khác (UC-REG-03).



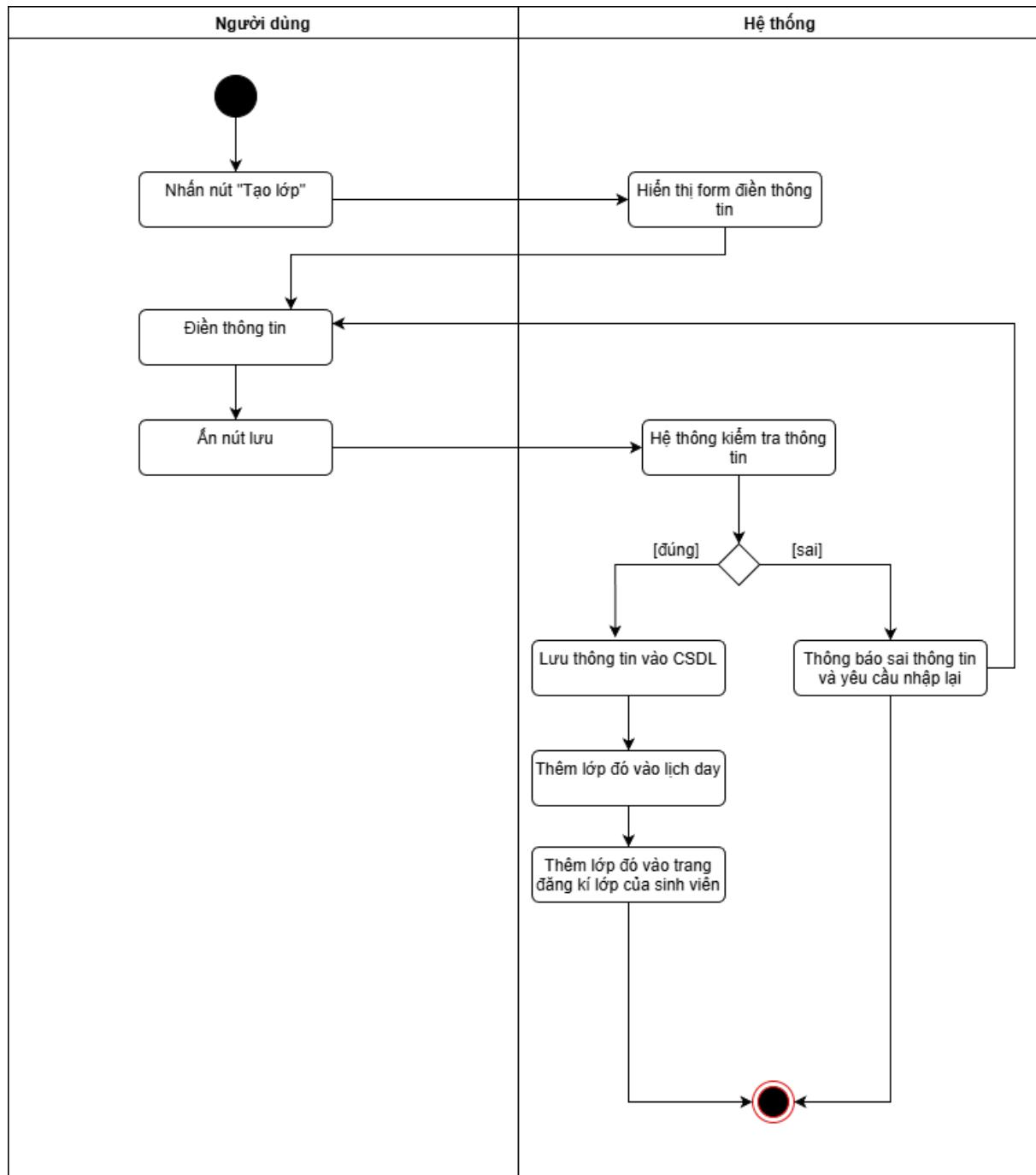
Hình 4.7: Sơ đồ hoạt động cho use-case "Đăng ký chương trình"

Mô tả sơ đồ hoạt động: Đăng ký chương trình

Sơ đồ này mô tả luồng nghiệp vụ chi tiết khi sinh viên đã chọn một khóa học cụ thể và thực hiện các hành động trên trang chi tiết, bao gồm cả việc đăng ký.

- **Bắt đầu:** Luồng được kích hoạt khi Sinh viên "Chọn một khóa học" (từ UC-REG-02).
- **Hiển thị Thông tin:** Hệ thống tiếp nhận yêu cầu và "Hiển thị trang thông tin chi tiết".
Trang này bao gồm thông tin mô tả khóa học (thuộc UC-REG-04).
- **Rẽ nhánh (Hành động của Sinh viên):** Hệ thống chờ hành động tiếp theo của Sinh viên.
Sinh viên có 3 lựa chọn chính:
 - "Thoát trang": Sinh viên thực hiện "Thoát trang thông tin chi tiết". Luồng kết thúc.
 - "Chọn xem thông tin": Sinh viên chọn xem thông tin bổ sung (ví dụ: "Chọn xem (Hồ sơ Tutor/ Đánh giá)"). Hệ thống sẽ "Hiển thị thông tin chi tiết được yêu cầu" (thuộc UC-REG-05/06). Sau khi xem xong, luồng quay lại trạng thái chờ "Hành động của sinh viên?".
 - "Đăng ký": Sinh viên "Nhấn nút 'Đăng ký'". Luồng chuyển đến bước 4.
- **Xử lý Đăng ký:** Hệ thống tiếp nhận yêu cầu và thực hiện "Xử lý yêu cầu đăng ký".
- **Rẽ nhánh (Kết quả Đăng ký):** Hệ thống kiểm tra kết quả xử lý nghiệp vụ:
 - **Nếu thất bại (Lỗi):** Hệ thống "Gửi thông báo lỗi" (ví dụ: hết chỗ, đã đăng ký). Luồng quay lại bước 3, chờ "Hành động của sinh viên?".
 - **Nếu thành công:** Hệ thống "Ghi nhận đăng ký". Luồng chuyển đến bước 6.
- **Hoàn tất:** Hệ thống "Hiển thị thông báo 'Đăng ký thành công'" cho Sinh viên.
- **Kết thúc:** Luồng kết thúc sau khi sinh viên đăng ký thành công.

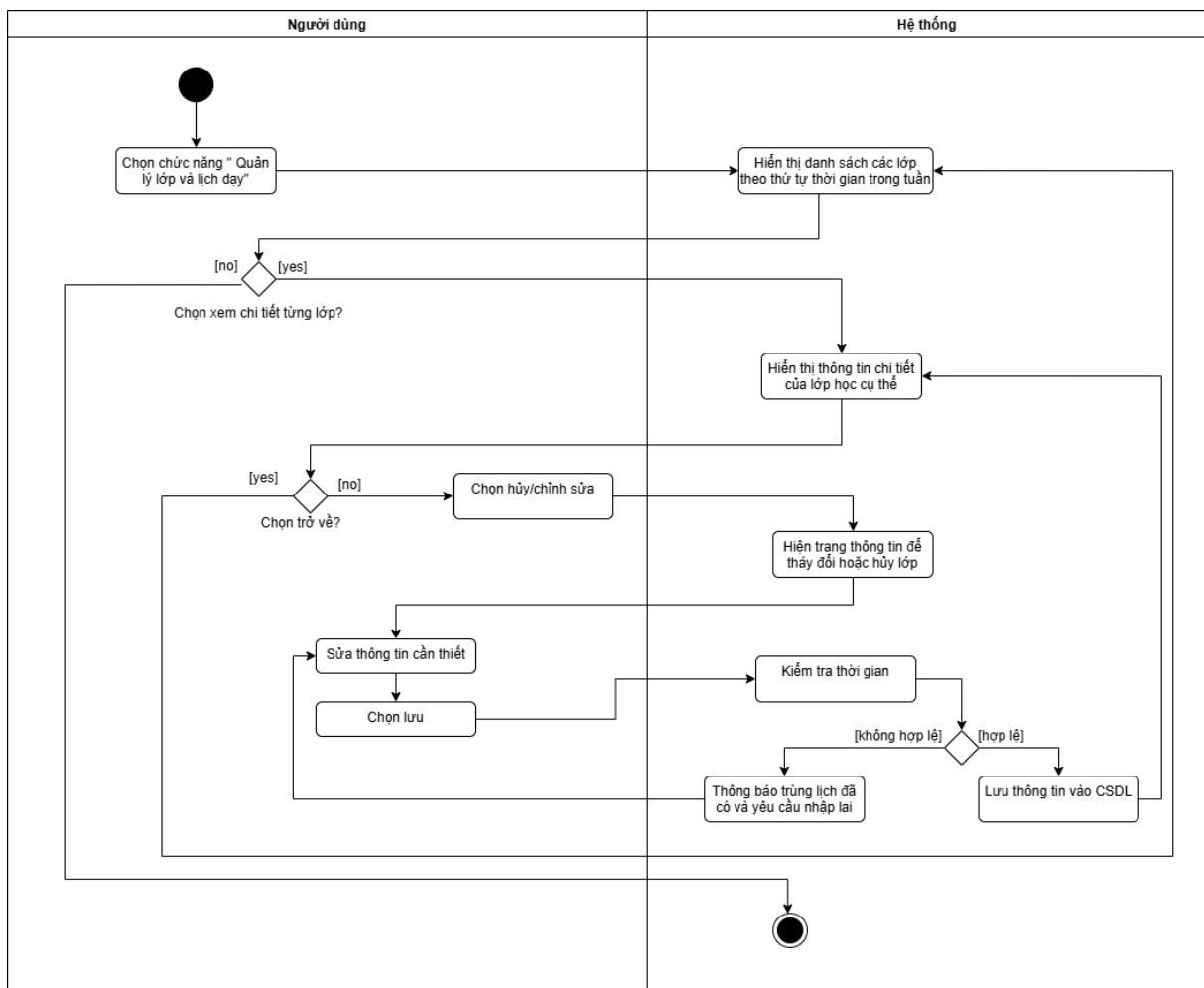
4.3 Tạo chương trình học



Hình 4.8: Sơ đồ use-case cho chức năng "Tạo lớp học"

Mô tả sơ đồ hoạt động: Tạo lớp học

- Người dùng bắt đầu quy trình và nhấn nút “*Tạo lớp*”.
- Hệ thống hiển thị form nhập thông tin lớp học.
- Người dùng nhập các thông tin cần thiết cho lớp học.
- Người dùng nhấn nút “*Lưu*”.
- Hệ thống tiếp nhận dữ liệu và tiến hành kiểm tra thông tin.
- Tại bước kiểm tra, hệ thống thực hiện hai nhánh xử lý:
 - Trường hợp thông tin hợp lệ:
 - * Hệ thống lưu thông tin lớp học vào cơ sở dữ liệu.
 - * Hệ thống thêm lớp học vào lịch dạy tương ứng.
 - * Hệ thống đồng thời thêm lớp học này vào danh sách lớp để sinh viên có thể đăng ký.
 - Trường hợp thông tin không hợp lệ:
 - * Hệ thống hiển thị thông báo lỗi và yêu cầu người dùng nhập lại thông tin.
- Quy trình kết thúc khi thông tin lớp được lưu thành công hoặc người dùng sửa lại thông tin theo yêu cầu hệ thống.

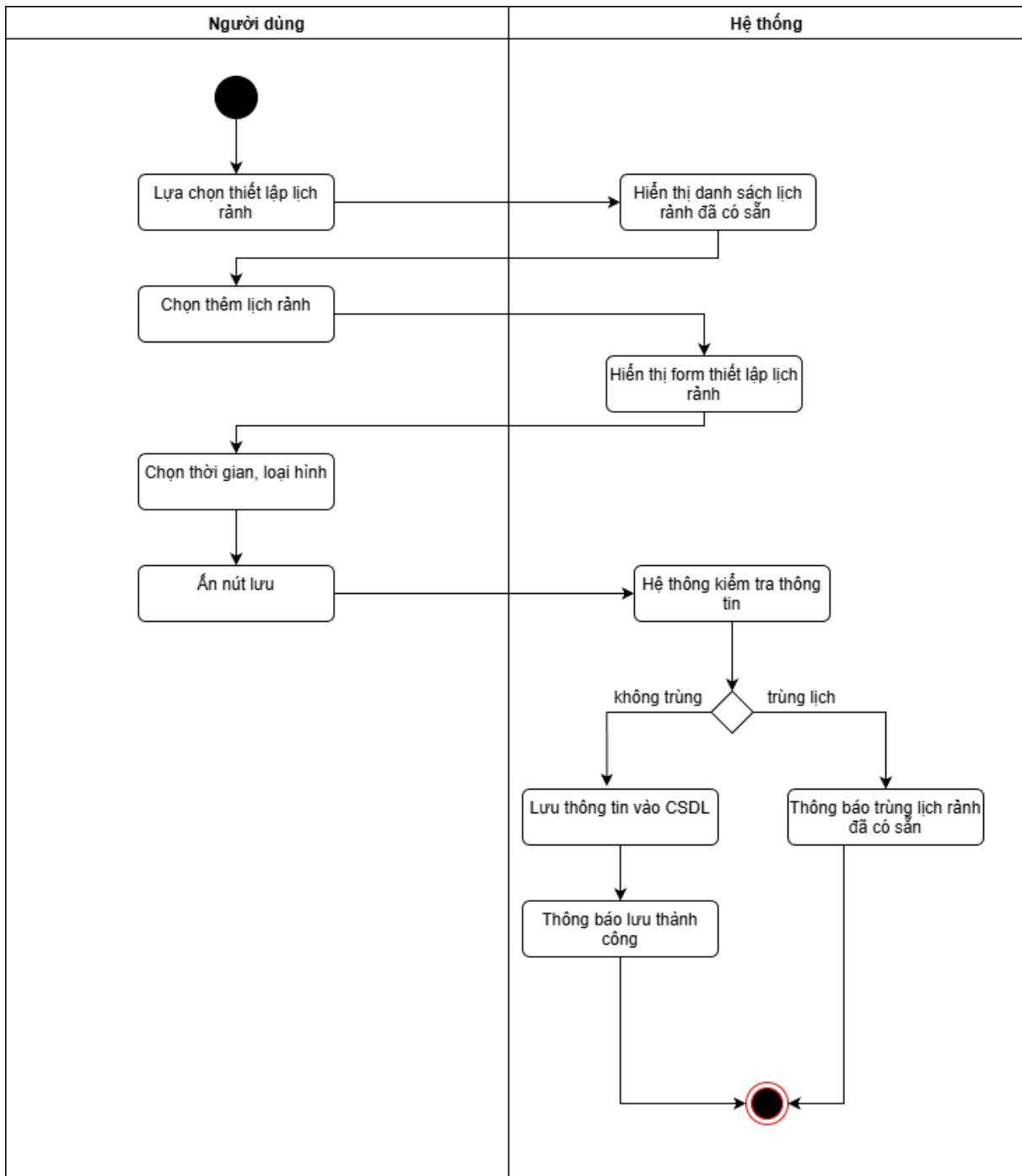


Hình 4.9: Sơ đồ use-case cho chức năng "Quản lý lớp và lịch dạy"



Mô tả sơ đồ hoạt động: Quản lý lớp và lịch dạy

- Người dùng bắt đầu bằng cách chọn chức năng “Quản lý lớp và lịch dạy”.
- Hệ thống hiển thị danh sách tất cả các lớp học mà tutor đã tạo trước đó.
- Người dùng lựa chọn xem chi tiết một lớp học cụ thể.
 - Nếu người dùng không chọn xem chi tiết, quy trình kết thúc.
 - Nếu người dùng chọn xem chi tiết, hệ thống sẽ hiển thị đầy đủ thông tin của lớp học đó.
- Tại giao diện chi tiết lớp học, người dùng có thể:
 - Chọn trả về danh sách lớp.
 - Hoặc chọn chức năng *hủy lớp* hoặc *chỉnh sửa lớp*.
- Nếu người dùng chọn chỉnh sửa, hệ thống hiển thị trạng thái lớp để cho phép thay đổi thông tin.
- Người dùng tiến hành nhập và sửa đổi các thông tin cần thiết, sau đó chọn “Lưu”.
- Hệ thống kiểm tra tính hợp lệ của thời gian:
 - Nếu thời gian không hợp lệ hoặc trùng lịch, hệ thống thông báo lỗi và yêu cầu người dùng nhập lại.
 - Nếu thời gian hợp lệ, hệ thống lưu thông tin mới vào cơ sở dữ liệu.
- Quy trình kết thúc sau khi thông tin lớp được cập nhật hoặc người dùng thoát khỏi chức năng.



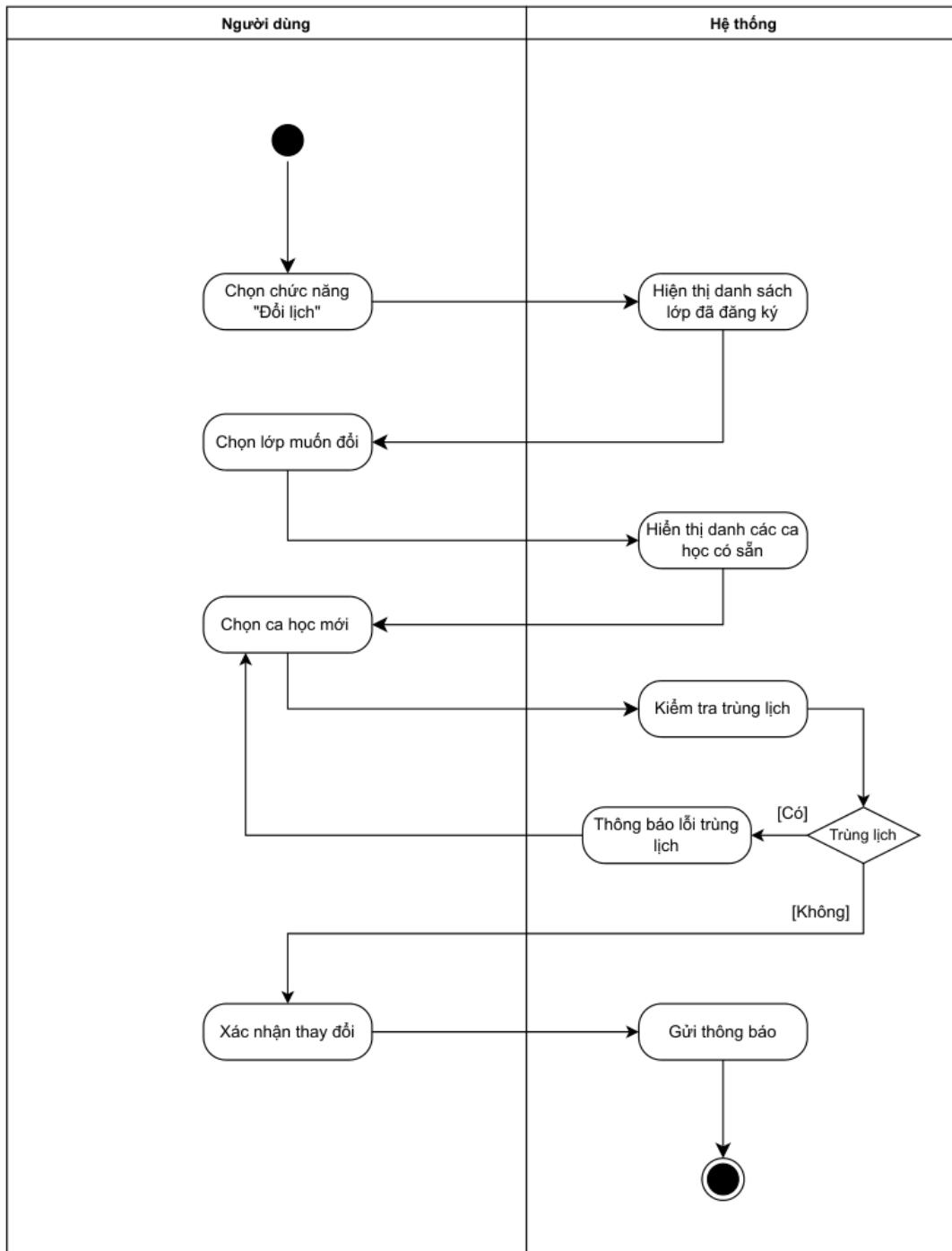
Hình 4.10: Sơ đồ use-case cho chức năng "Lịch rành"



Mô tả sơ đồ hoạt động: Thiết lập lịch rảnh

- **Người dùng** bắt đầu quy trình và chọn chức năng *thiết lập lịch rảnh*.
- **Hệ thống** hiển thị danh sách các lịch rảnh đã có sẵn.
- **Người dùng** chọn thao tác *thêm lịch rảnh*.
- **Hệ thống** hiển thị form thiết lập lịch rảnh mới.
- **Người dùng** chọn thời gian và loại hình phù hợp.
- **Người dùng** nhấn nút *Lưu*.
- **Hệ thống** thực hiện kiểm tra thông tin:
 - Nếu không trùng lịch đã tồn tại:
 - * **Hệ thống** lưu thông tin vào cơ sở dữ liệu.
 - * **Hệ thống** gửi thông báo lưu thành công.
 - Nếu trùng lịch rảnh đã có:
 - * **Hệ thống** hiển thị thông báo trùng lịch, yêu cầu **người dùng** điều chỉnh.
- Quy trình kết thúc sau khi thông tin được lưu thành công hoặc sau khi **hệ thống** hiển thị thông báo trùng lịch.

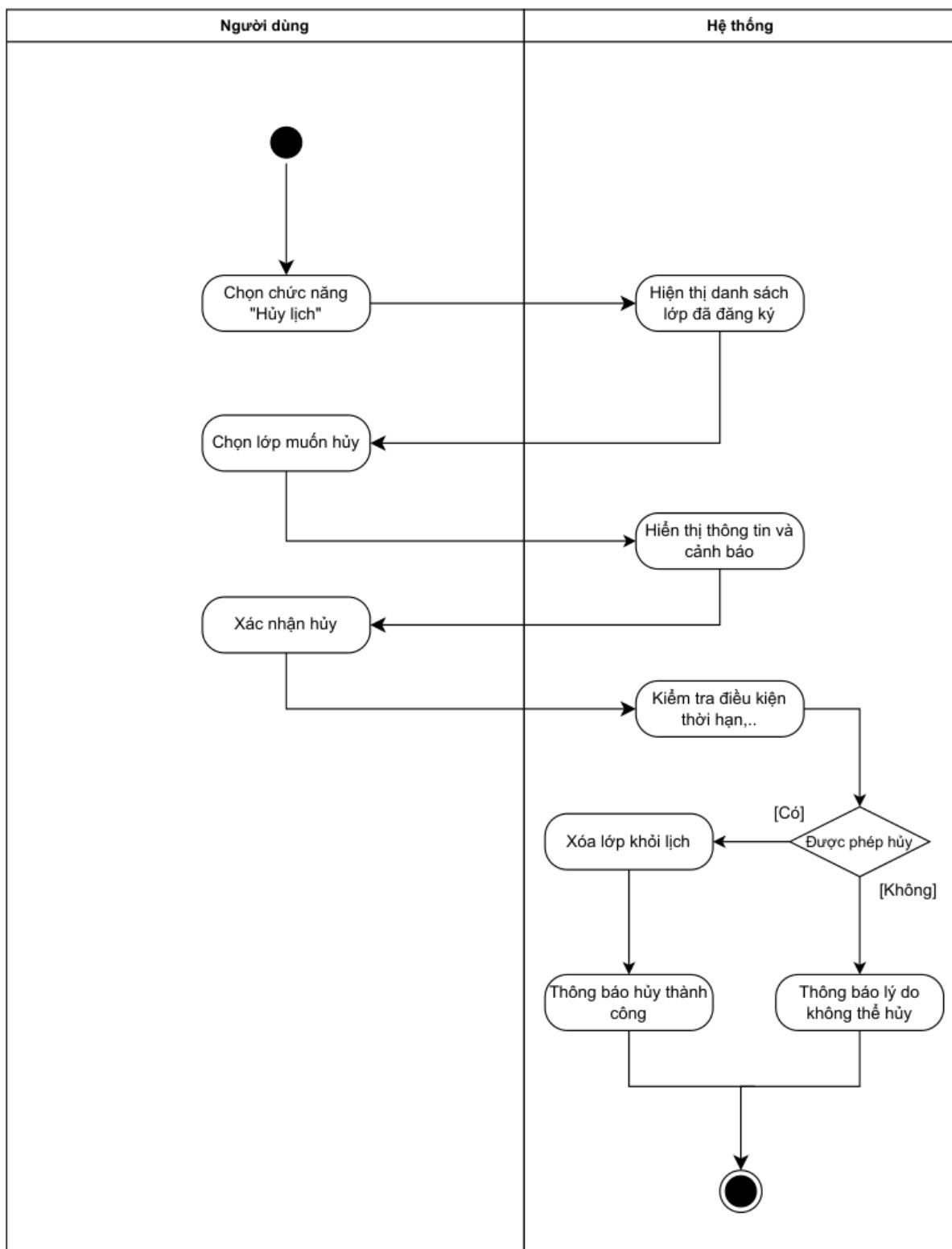
4.4 Thiết lập lịch trình cho sinh viên



Hình 4.11: Sơ đồ hoạt động cho use-case "Đổi lịch học" cho sinh viên

Mô tả sơ đồ hoạt động: Đổi lịch học

- **Người dùng** chọn chức năng “**Đổi lịch**” trên giao diện.
- **Hệ thống** hiển thị danh sách các lớp mà sinh viên đã đăng ký.
- **Người dùng** chọn lớp muốn đổi.
- **Hệ thống** hiển thị danh sách các ca học mới có sẵn cho lớp được chọn.
- **Người dùng** chọn ca học mới.
- **Hệ thống** tiến hành kiểm tra trùng lịch với các lớp khác trong thời khóa biểu hiện tại.
 - Nếu **bị trùng lịch**: **Hệ thống** hiển thị thông báo lỗi “Trùng lịch” và yêu cầu người dùng chọn lại.
 - Nếu **không trùng lịch**: **Hệ thống** cập nhật thay đổi và **gửi thông báo** xác nhận đổi lịch thành công.
- **Người dùng** xác nhận thay đổi và kết thúc use-case.

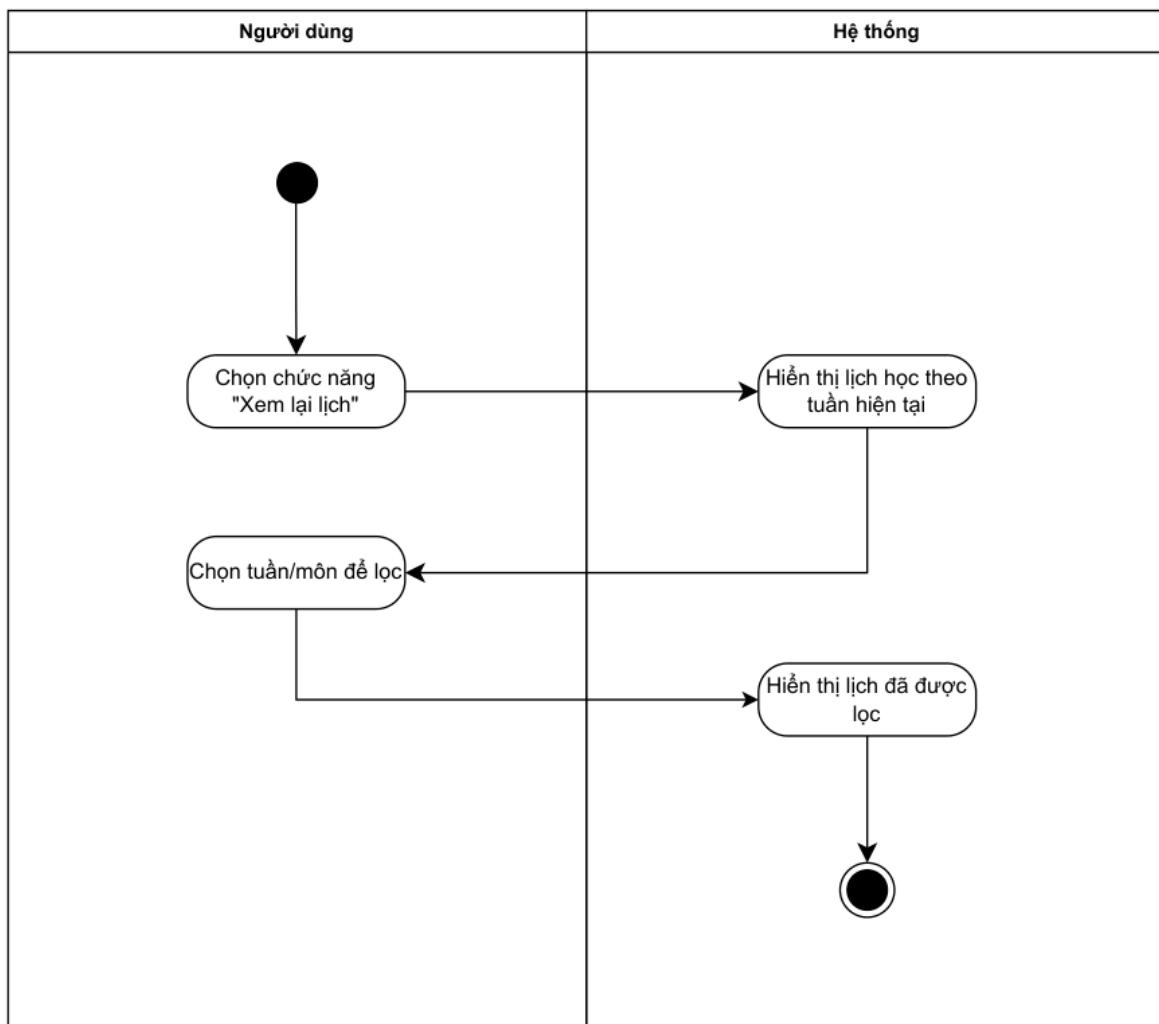


Hình 4.12: Sơ đồ hoạt động cho use-case "Hủy lịch học" cho sinh viên



Mô tả sơ đồ hoạt động: Hủy lịch học

- **Người dùng** chọn chức năng “**Hủy lịch**”.
- **Hệ thống** hiển thị danh sách các lớp đã đăng ký.
- **Người dùng** chọn lớp muốn hủy.
- **Hệ thống** hiển thị thông tin chi tiết và các cảnh báo liên quan (ví dụ: thời hạn hủy, điều kiện học vụ).
- **Người dùng** xác nhận thao tác hủy.
- **Hệ thống kiểm tra điều kiện hủy** (thời hạn, trạng thái lớp, quy định nhà trường, ...).
 - Nếu **không được phép hủy**: hệ thống hiển thị thông báo lý do không thể hủy.
 - Nếu **được phép hủy**: hệ thống xóa lớp khỏi lịch học của sinh viên và thông báo hủy thành công.

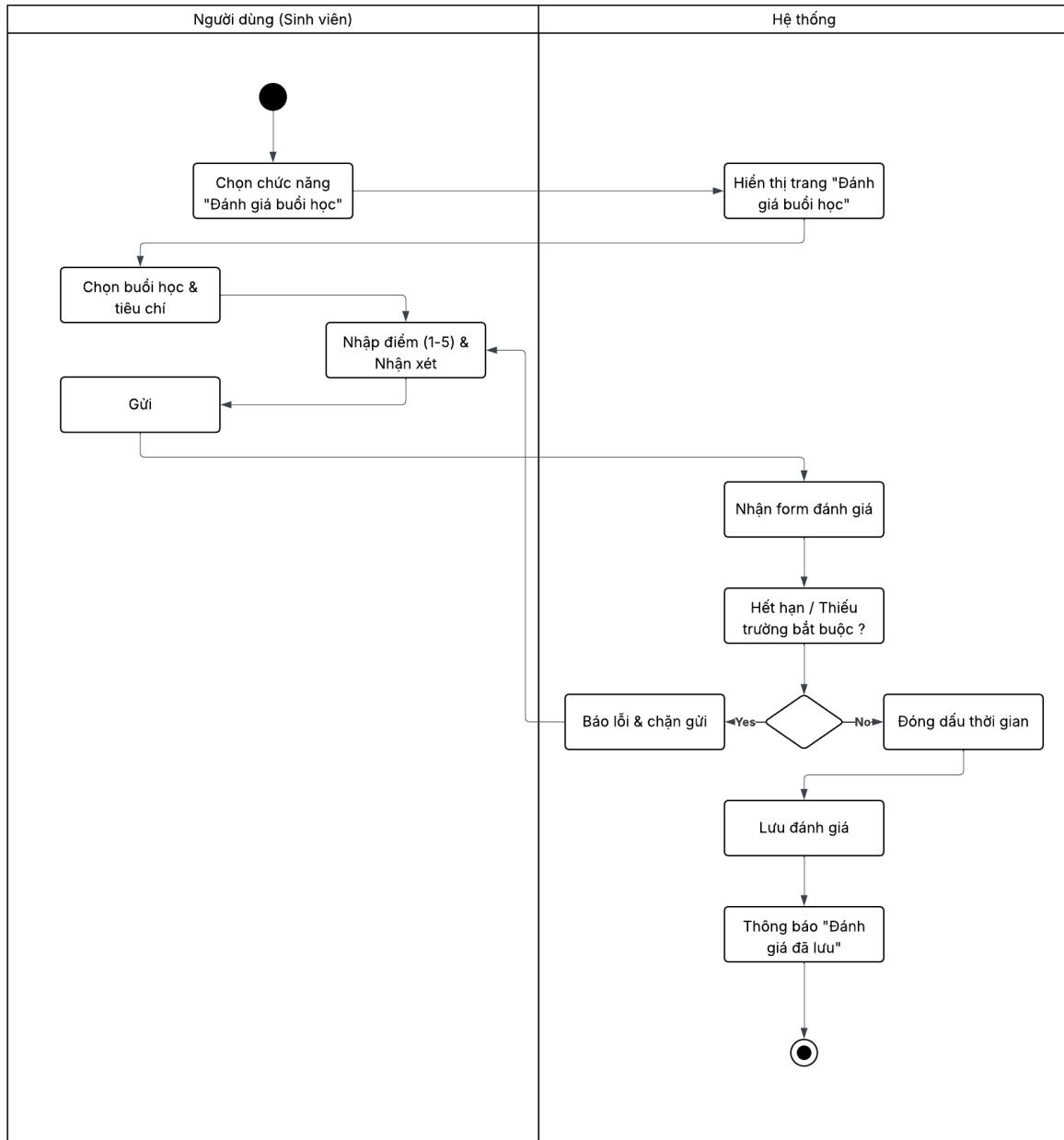


Hình 4.13: Sơ đồ hoạt động cho use-case "Xem lại lịch học" cho sinh viên

Mô tả sơ đồ hoạt động: Xem lại lịch học

- **Người dùng** chọn chức năng “**Xem lại lịch**”.
- **Hệ thống** mặc định hiển thị **lịch học theo tuần hiện tại**.
- Nếu **người dùng** muốn xem lịch theo tuần khác hoặc lọc theo môn học, **người dùng** chọn **tuần/môn cần lọc**.
- **Hệ thống** áp dụng tiêu chí lọc và hiển thị **lịch học đã được lọc** tương ứng.

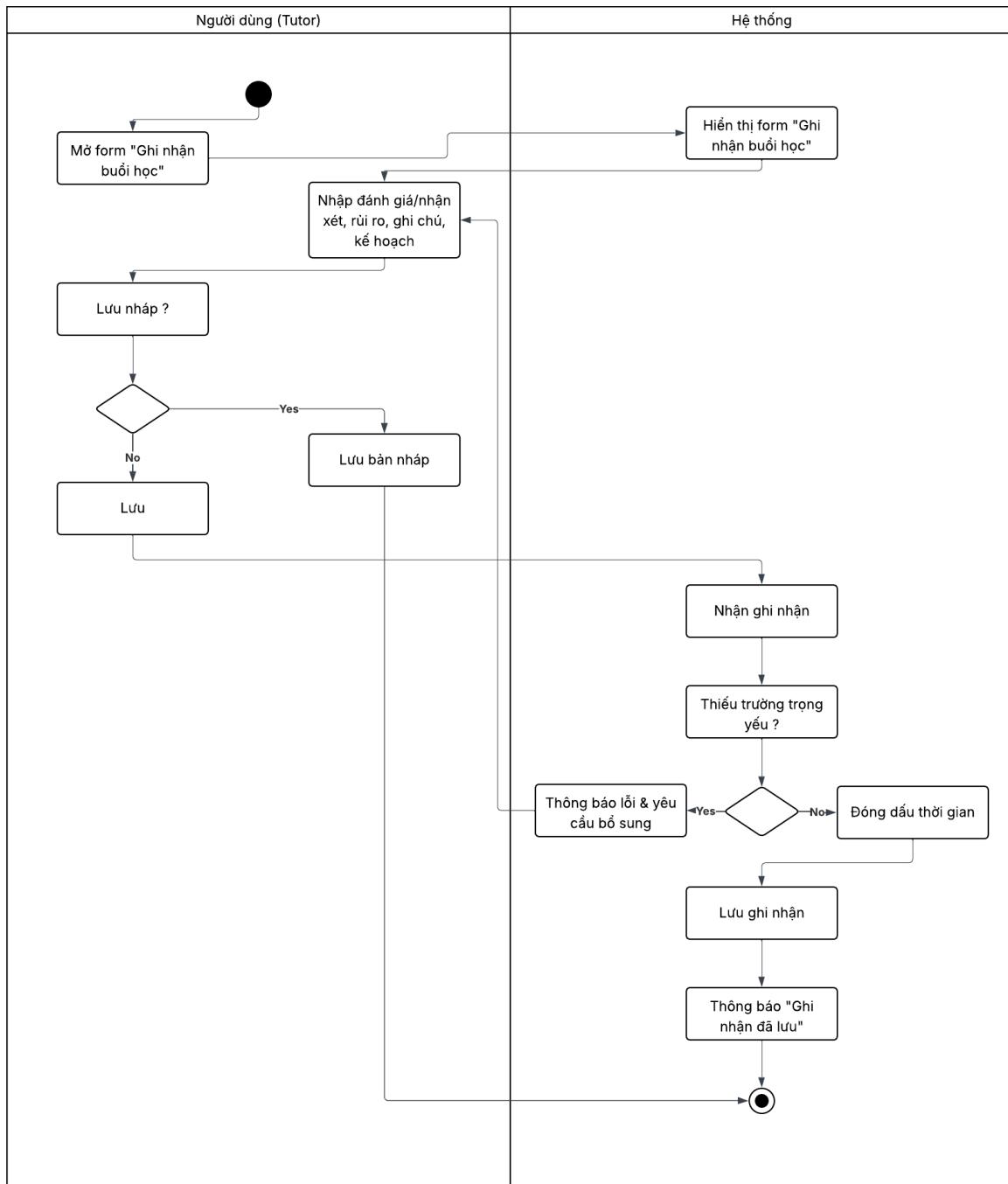
4.5 Đánh giá



Hình 4.14: Sơ đồ hoạt động cho use-case "Gửi đánh giá Tutor" của sinh viên

Mô tả sơ đồ hoạt động: Gửi đánh giá Tutor

Luồng chuẩn: Sinh viên vào trang “Đánh giá buổi học” → chọn tiêu chí, nhập điểm và nhận xét → gửi; hệ thống kiểm tra hạn/chứng thực, lưu và cập nhật tổng hợp. Nếu quá hạn/thiếu trường bắt buộc, hệ thống chặn gửi.

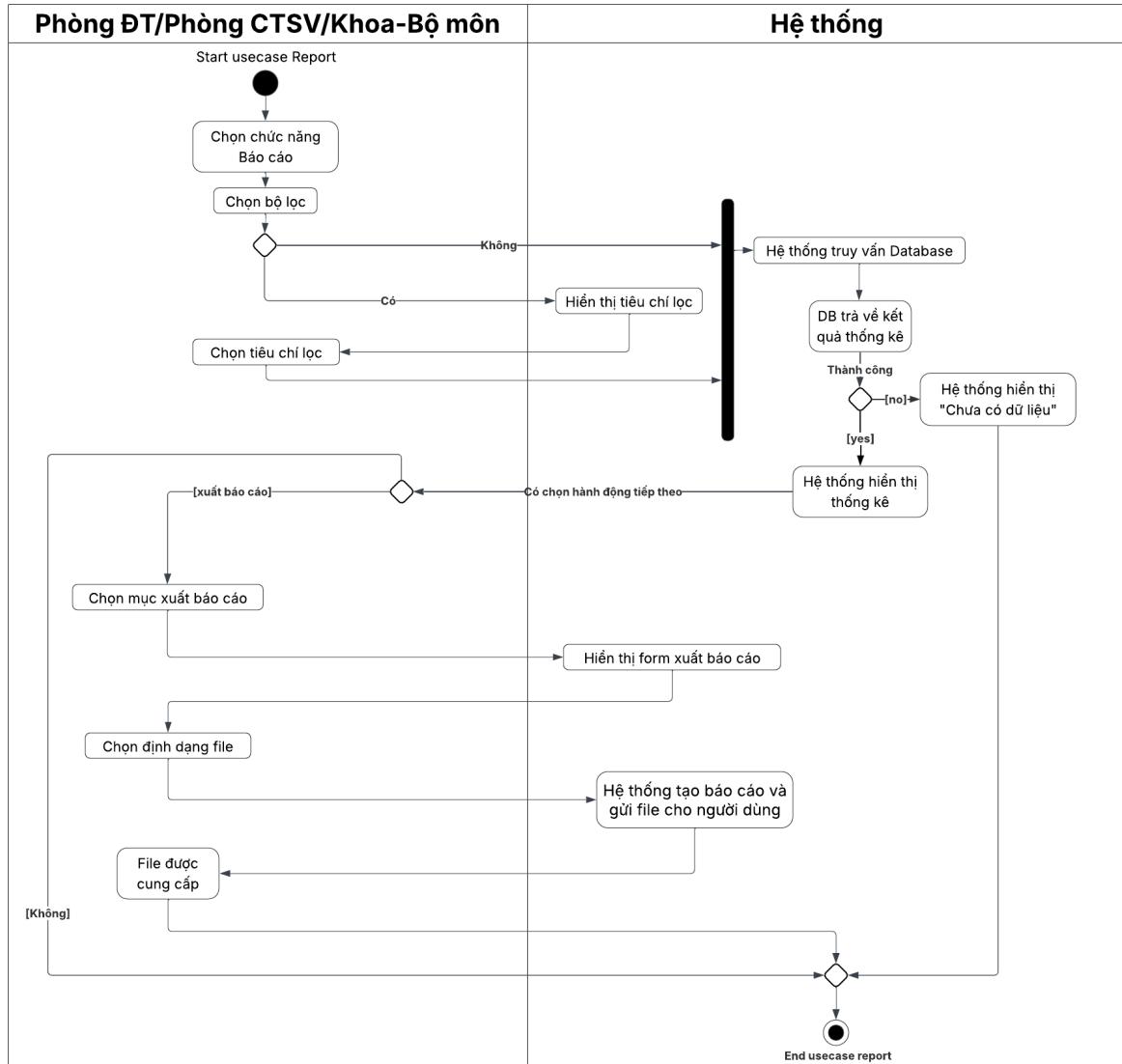


Hình 4.15: Sơ đồ hoạt động cho use-case "Ghi nhận tiến bộ sinh viên" của Tutor

Mô tả sơ đồ hoạt động: Ghi nhận tiến bộ sinh viên

Luồng chuẩn: Tutor mở form “Ghi nhận buổi học” → nhập đánh giá/nhận xét, rủi ro, ghi chú, kế hoạch → lưu bản nháp hoặc lưu chính thức. Nếu thiếu trường trọng yếu, hệ thống báo lỗi; khi lưu thành công, kích hoạt cập nhật tổng hợp.

4.6 Phân tích và báo cáo

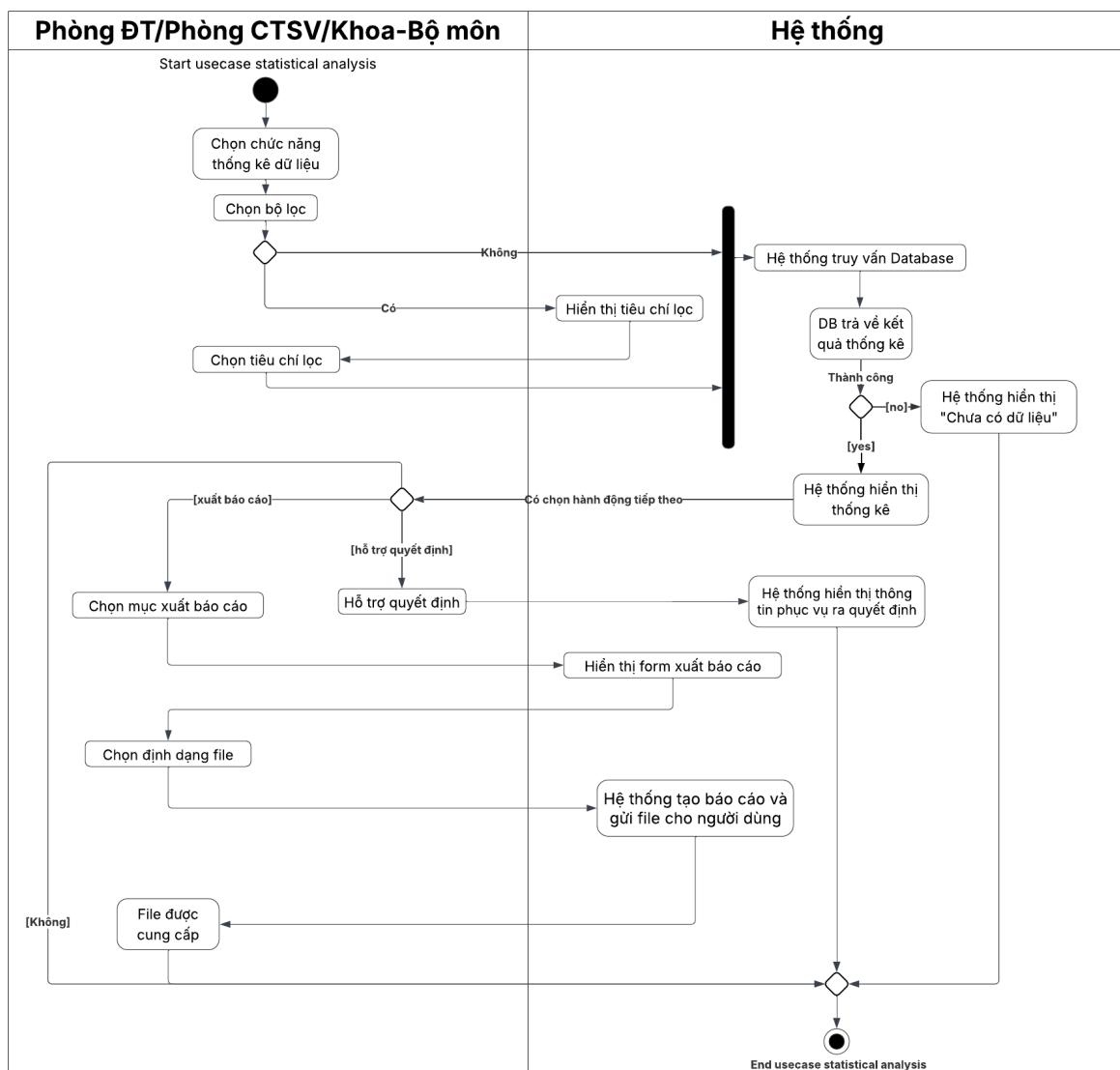


Hình 4.16: Sơ đồ hoạt động cho use-case "Phân tích và báo cáo"



Mô tả sơ đồ hoạt động: Phân tích và báo cáo

- Luồng được kích hoạt khi **Người dùng** (Phòng ĐT/Phòng CTSV/Khoa - Bộ môn) chọn chức năng "*Báo cáo*".
- **Người dùng** chọn bộ lọc cần thiết (có thể bỏ qua).
- **Hệ thống** hiển thị chi tiết tiêu chí lọc (nếu có) và thực hiện nghiệp vụ "Truy vấn Database để lấy dữ liệu tổng hợp".
- Sau khi truy vấn, **Hệ thống** rẽ nhánh dựa trên kết quả:
 - Nếu "Không có dữ liệu": **Hệ thống** hiển thị thông báo "*Chưa có dữ liệu*". Luồng kết thúc.
 - Nếu "Có dữ liệu": **Hệ thống** hiển thị báo cáo thống kê đã tổng hợp". Luồng chuyển đến bước tiếp theo.
- Sau khi có dữ liệu **Người dùng** chọn hành động tiếp theo:
 - Xuất báo cáo: **Người dùng** chọn mục xuất và định dạng file.
 - * **Hệ thống** nhận form báo cáo.
 - * **Hệ thống** tạo báo cáo và gửi file cho **Người dùng**.
 - * Luồng kết thúc.
 - Không có hành động khác: Luồng kết thúc ngay sau khi báo cáo được hiển thị.
- **Kết thúc:** Luồng dừng lại khi báo cáo đã được hiển thị hoặc file đã được xuất.



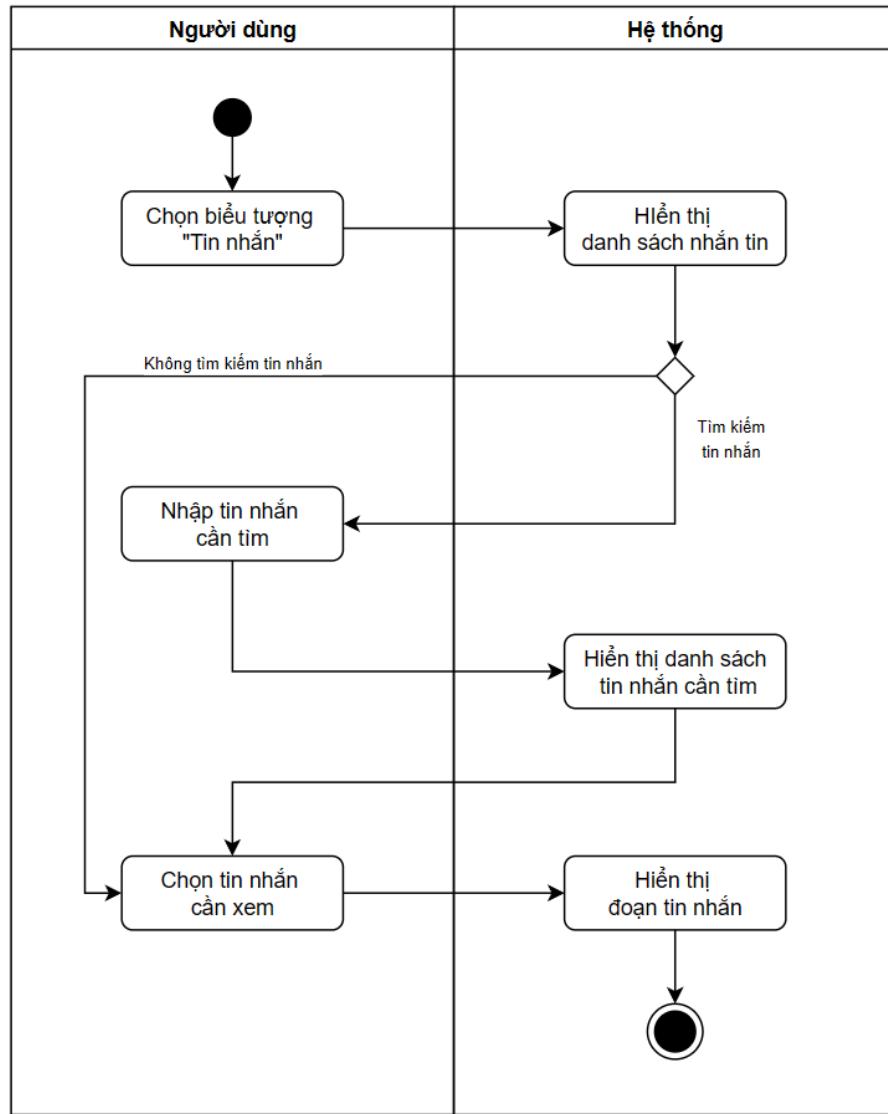
Hình 4.17: Sơ đồ hoạt động cho use-case "Tạo báo cáo"



Mô tả sơ đồ hoạt động: Tạo báo cáo

- Luồng được kích hoạt khi **Người dùng** (Phòng ĐT/Phòng CTSV/Khoa - Bộ môn) chọn chức năng "*Thống kê dữ liệu*".
- **Người dùng** chọn bộ lọc cần thiết (có thể bỏ qua).
- **Hệ thống** hiển thị chi tiết tiêu chí lọc (nếu có) và thực hiện nghiệp vụ "*Truy vấn Database để lấy dữ liệu tổng hợp*".
- Sau khi truy vấn, **Hệ thống** rẽ nhánh dựa trên kết quả:
 - Nếu "Không có dữ liệu": **Hệ thống** hiển thị thông báo "*Chưa có dữ liệu*". Luồng kết thúc.
 - Nếu "Có dữ liệu": **Hệ thống** hiển thị báo cáo thống kê đã tổng hợp. Luồng chuyển đến bước tiếp theo.
- Sau khi có dữ liệu: **Người dùng** chọn hành động tiếp theo:
 - Hỗ trợ ra quyết định: **Hệ thống** hiển thị thông tin phục vụ ra quyết định. Luồng kết thúc.
 - Xuất báo cáo: **Người dùng** chọn mục xuất và định dạng file.
 - * **Hệ thống** nhận form báo cáo.
 - * **Hệ thống** tạo báo cáo và gửi file cho **Người dùng**.
 - * Luồng kết thúc.
 - Không có hành động khác: Luồng kết thúc ngay sau khi báo cáo được hiển thị.
- **Kết thúc:** Luồng dừng lại khi báo cáo đã được hiển thị, thông tin ra quyết định đã được cung cấp, hoặc file đã được xuất.

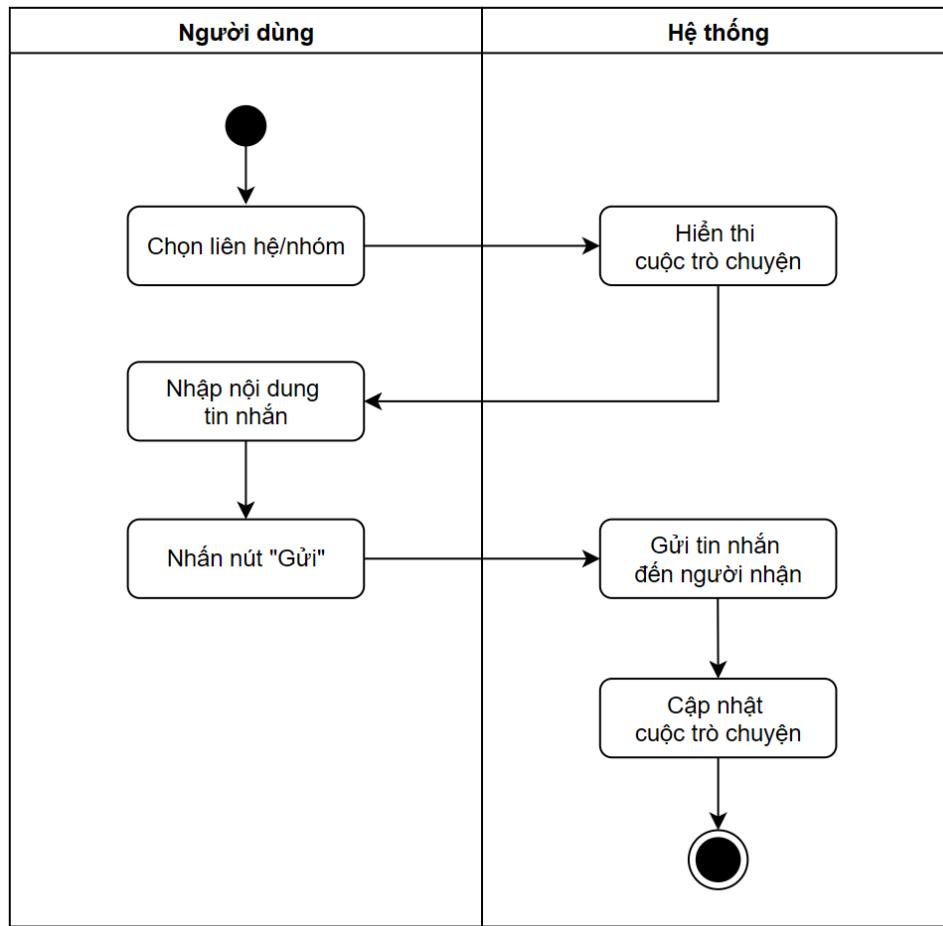
4.7 Thông báo và nhắn tin



Hình 4.18: Sơ đồ hoạt động cho use-case "Xem thông báo và tin nhắn"

Mô tả sơ đồ hoạt động: Xem thông báo và tin nhắn

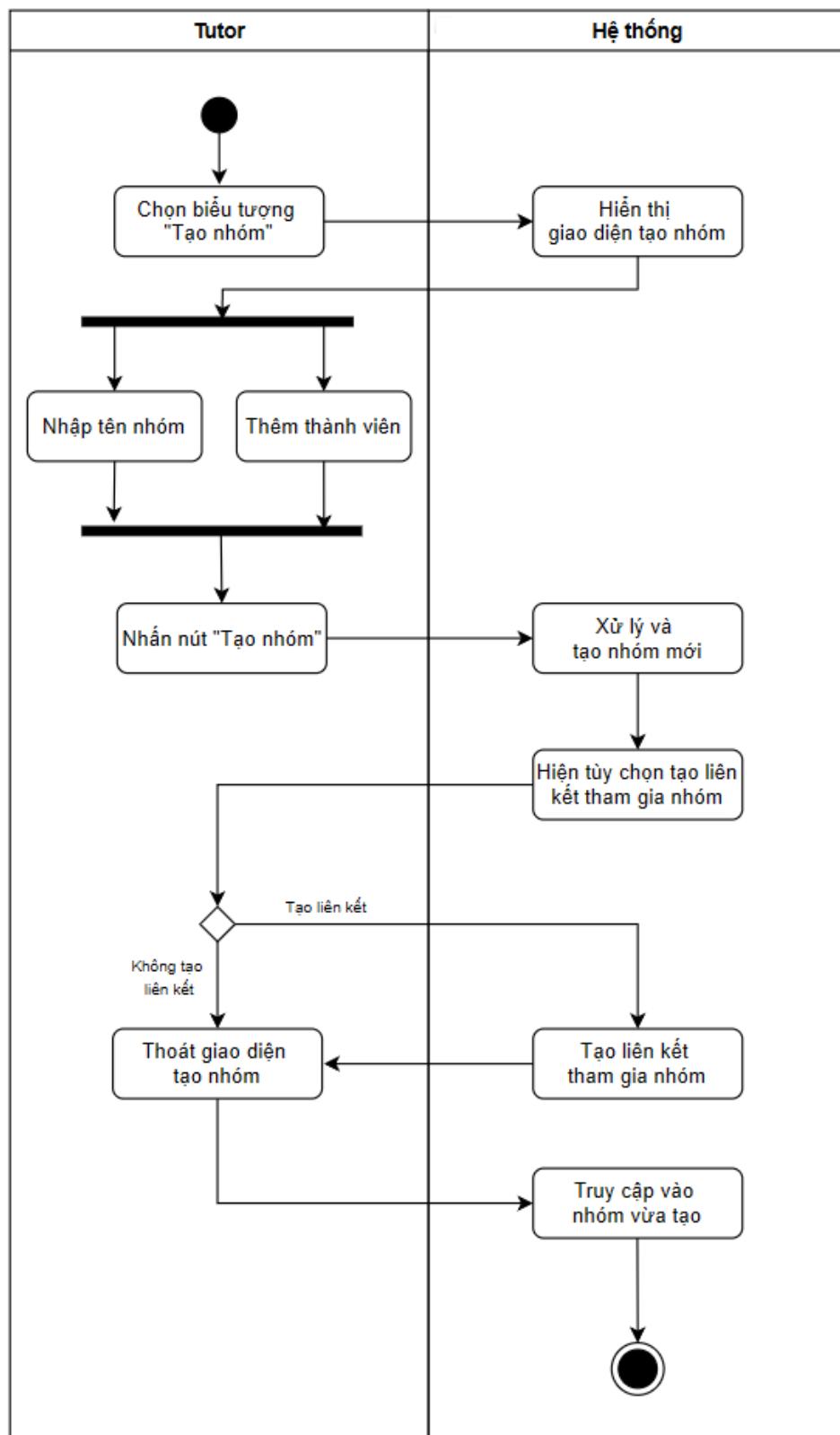
- **Người dùng** chọn biểu tượng "Tin nhắn" trên giao diện ứng dụng.
- **Hệ thống** hiển thị danh sách các cuộc trò chuyện (nhắn tin) hiện có.
- Nếu **người dùng** muốn tìm tin nhắn, bấm vào thanh tìm kiếm và nhập từ khóa cần tìm.
- **Hệ thống** truy vấn, trả về danh sách các tin nhắn phù hợp với nội dung tìm kiếm.
- **Người dùng** chọn một tin nhắn cụ thể trong danh sách kết quả.
- **Hệ thống** hiển thị chi tiết đoạn tin nhắn đã chọn cho người dùng.



Hình 4.19: Sơ đồ hoạt động cho use-case "Gửi tin nhắn"

Mô tả sơ đồ hoạt động: Gửi tin nhắn

- **Người dùng** chọn liên hệ (cá nhân hoặc nhóm) trong danh sách bạn bè/nhóm.
- **Hệ thống** hiển thị giao diện nội dung cuộc trò chuyện với liên hệ/nhóm đó.
- **Người dùng** nhập nội dung tin nhắn muốn gửi.
- **Người dùng** nhấn nút "Gửi".
- **Hệ thống** nhận nội dung tin nhắn, thực hiện gửi đến người nhận/nhóm phù hợp.
- **Hệ thống** đồng thời cập nhật hội thoại (cả bên gửi lẫn bên nhận) hiển thị tin nhắn vừa gửi.

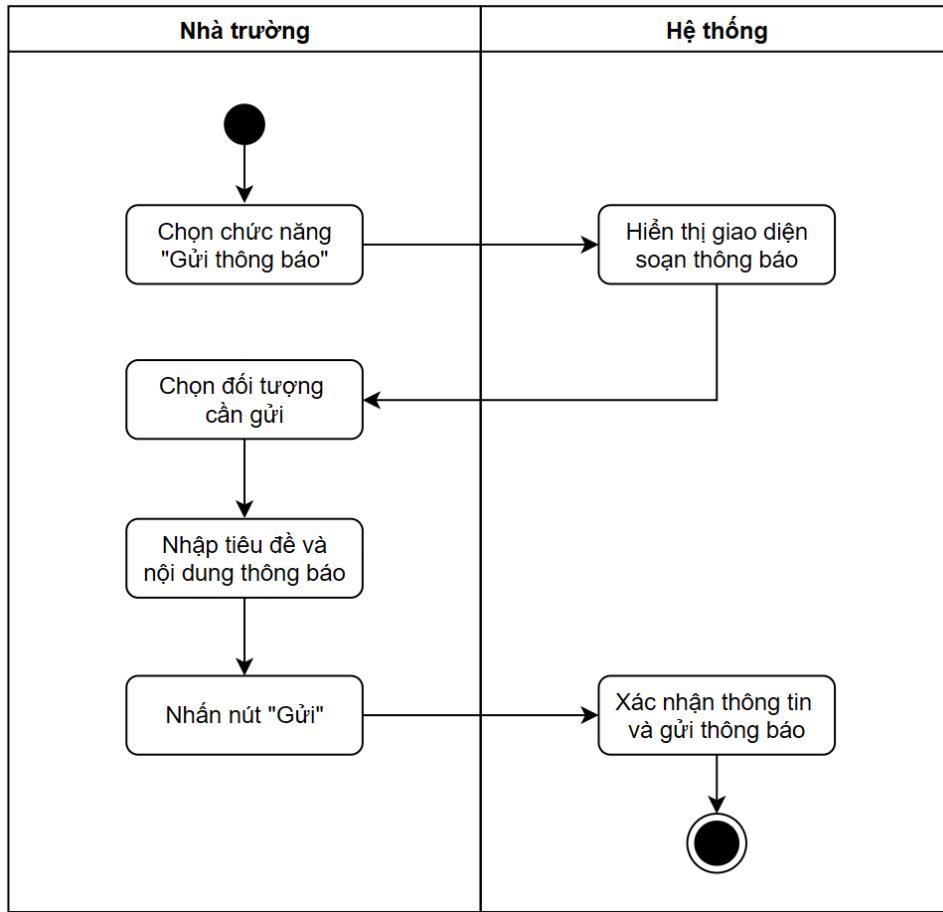


Hình 4.20: Sơ đồ hoạt động cho use-case "Tạo nhóm"



Mô tả sơ đồ hoạt động: Tạo nhóm

- **Tutor** chọn biểu tượng "Tạo nhóm" trên ứng dụng.
- **Hệ thống** hiển thị giao diện cho phép nhập thông tin nhóm.
- **Tutor** thực hiện hai thao tác song song: nhập tên nhóm và thêm thành viên.
- Sau khi hoàn tất, **Tutor** nhấn nút "Tạo nhóm".
- **Hệ thống** kiểm tra, xử lý và tạo nhóm mới trong hệ thống.
- **Hệ thống** cung cấp tùy chọn cho tutor có muốn tạo liên kết tham gia nhóm hay không:
 - Nếu không tạo liên kết: Giao diện tạo nhóm kết thúc, tutor có thể truy cập vào nhóm mới.
 - Nếu tạo liên kết: Hệ thống sinh liên kết, lưu trữ và trả lại cho tutor, rồi kết thúc tại giao diện nhóm vừa tạo.

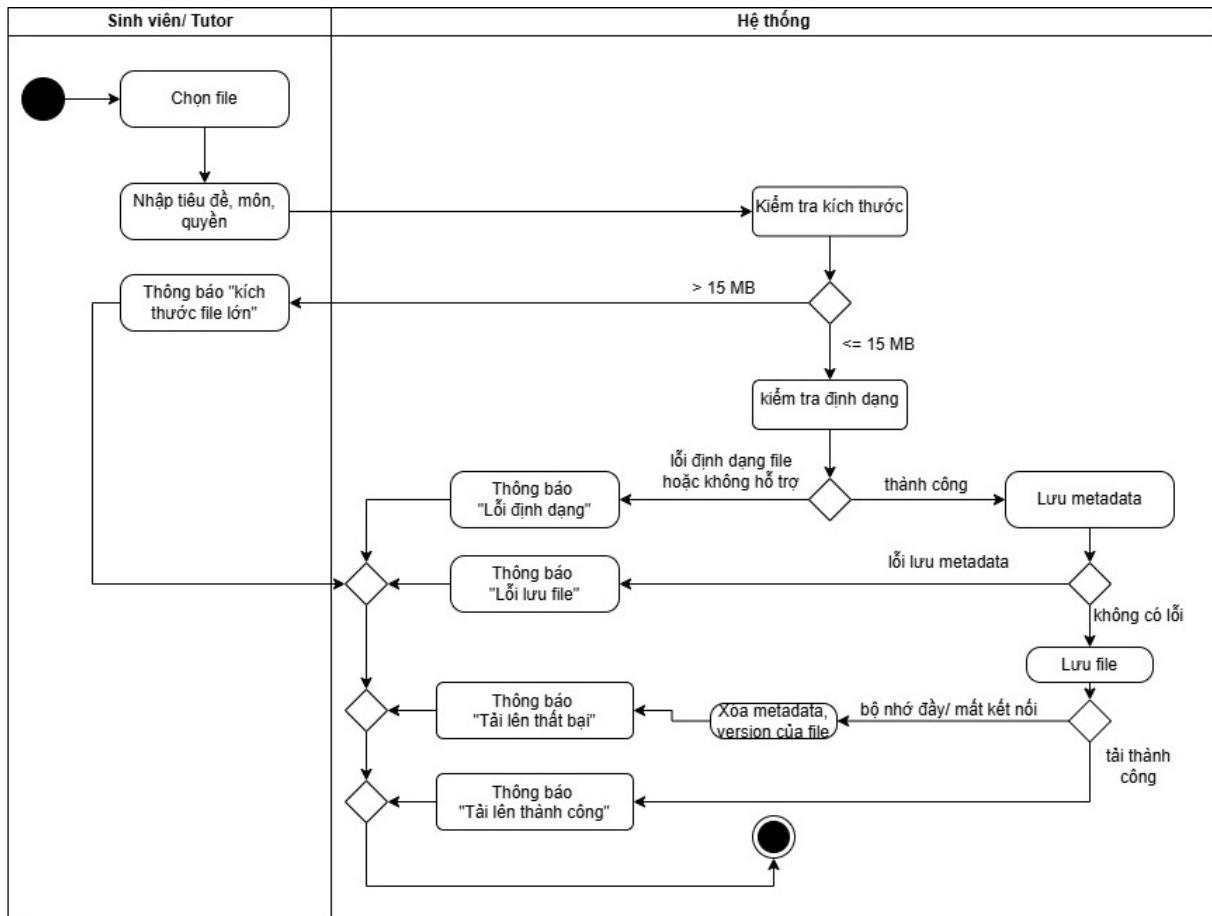


Hình 4.21: Sơ đồ hoạt động cho use-case "Gửi thông báo"

Mô tả sơ đồ hoạt động: Gửi thông báo

- **Nhà trường** chọn biểu tượng "Gửi thông báo" trên thanh điều hướng.
- **Hệ thống** nhận yêu cầu, hiển thị giao diện soạn thảo thông báo, cho phép nhập thông tin cần thiết.
- **Nhà trường** chọn đối tượng cần gửi đi sau đó nhập tiêu đề và nội dung thông báo mong muốn gửi đi.
- Sau khi điền đủ thông tin, **nhà trường** nhấn nút "Gửi".
- **Hệ thống** xác nhận lại thông tin, thực hiện việc gửi thông báo đến các đối tượng đã chọn và kết thúc quy trình.

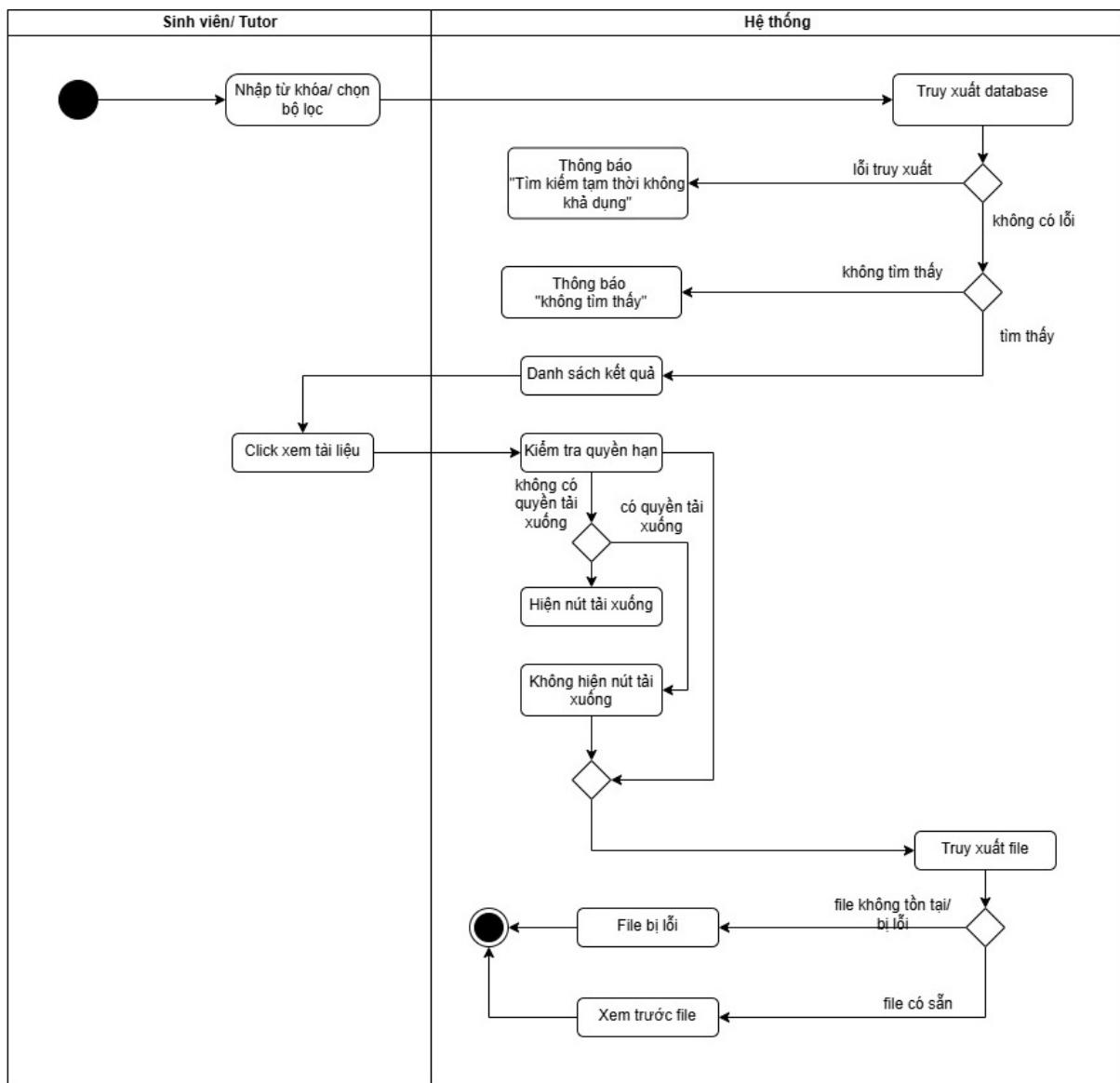
4.8 Quản lý tài liệu



Hình 4.22: Sơ đồ hoạt động cho use-case "Tải tài liệu lên"

Mô tả sơ đồ hoạt động: Tải tài liệu lên

- Người dùng chọn biểu tượng "Tải xuống" trên giao diện ứng dụng.
- Sau đó nhập những thông tin cần thiết cho tệp sẽ tải lên.
- **Hệ thống** kiểm tra kích thước và định dạng tệp tin để quyết định xem có cho người dùng tải lên hay không.
- Nếu hợp lệ thì chuẩn bị tải tệp lên **hệ thống**. Trong quá trình tải sẽ phát sinh sự cố nên hệ thống sẽ báo lỗi tới người dùng khi xảy ra.
- Nếu **hệ thống** tải lên thành công sẽ báo về phái người dùng.

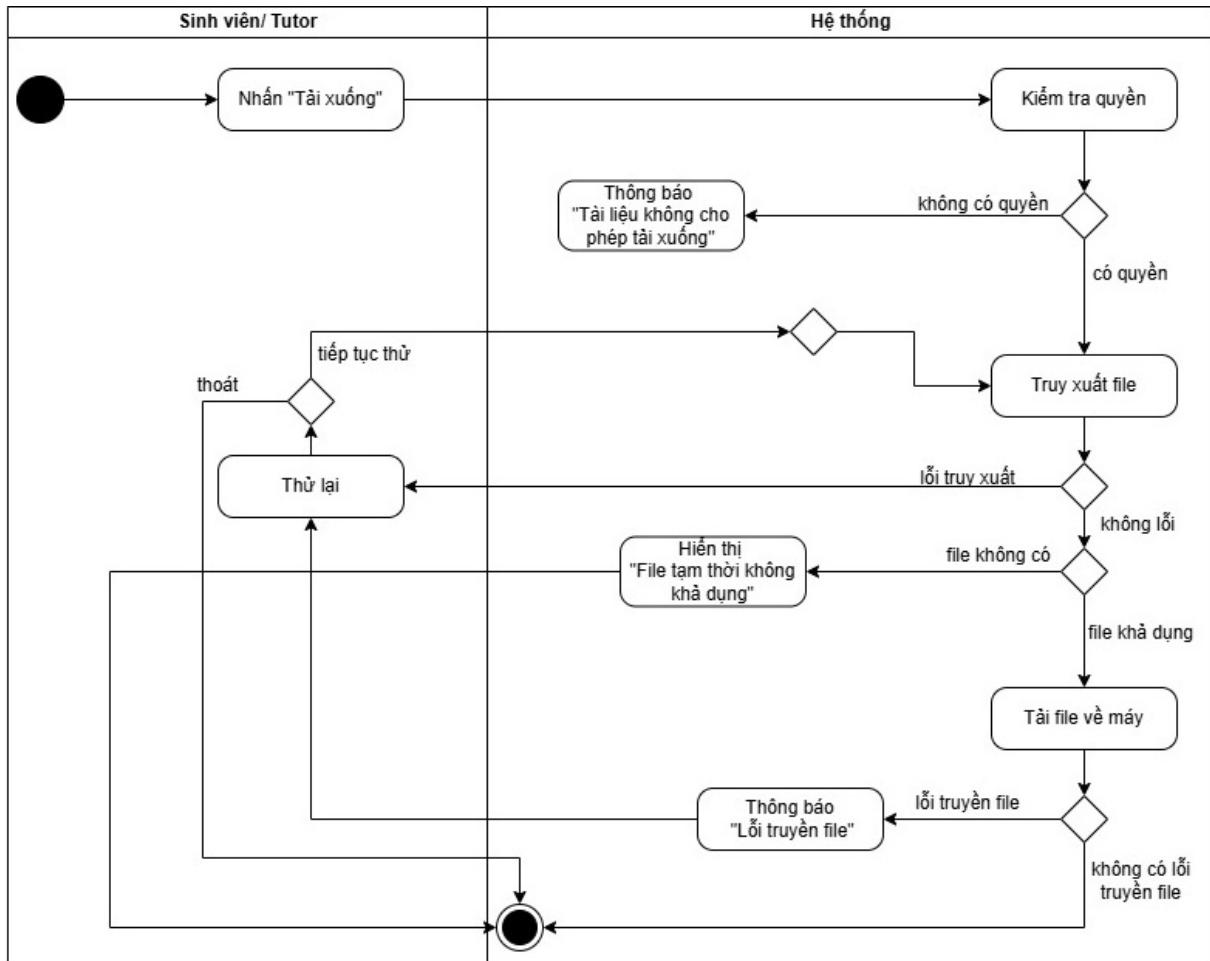


Hình 4.23: Sơ đồ hoạt động cho use-case "Tìm kiếm và xem trước"

Mô tả sơ đồ hoạt động: Việc tìm kiếm và xem trước tài liệu

- **Người dùng** nhập từ khóa trên thanh "Tìm kiếm" và có thể chọn bộ lọc.
- **Hệ thống** sau đó thực hiện truy vấn dựa trên yêu cầu từ **người dùng**, nếu tìm thấy sẽ trả về danh sách các thông tin của tài liệu về phía **người dùng** và đợi **người dùng** bấm chọn.
- Khi người dùng chọn một thông tin tài liệu, **hệ thống** thực hiện các kiểm tra cần thiết và truy xuất bộ nhớ để lấy dữ liệu xem trước.
- **Hệ thống** sau đó trả về kết quả và hiển thị giao diện xem trước tài liệu cho **người dùng**.
- Giao diện xem trước có thêm nút tải xuống nếu tài liệu cho phép tải về.

- Trong quá trình truy xuất, **hệ thống** cũng sẽ đảm bảo xử lý lỗi nếu phát sinh.



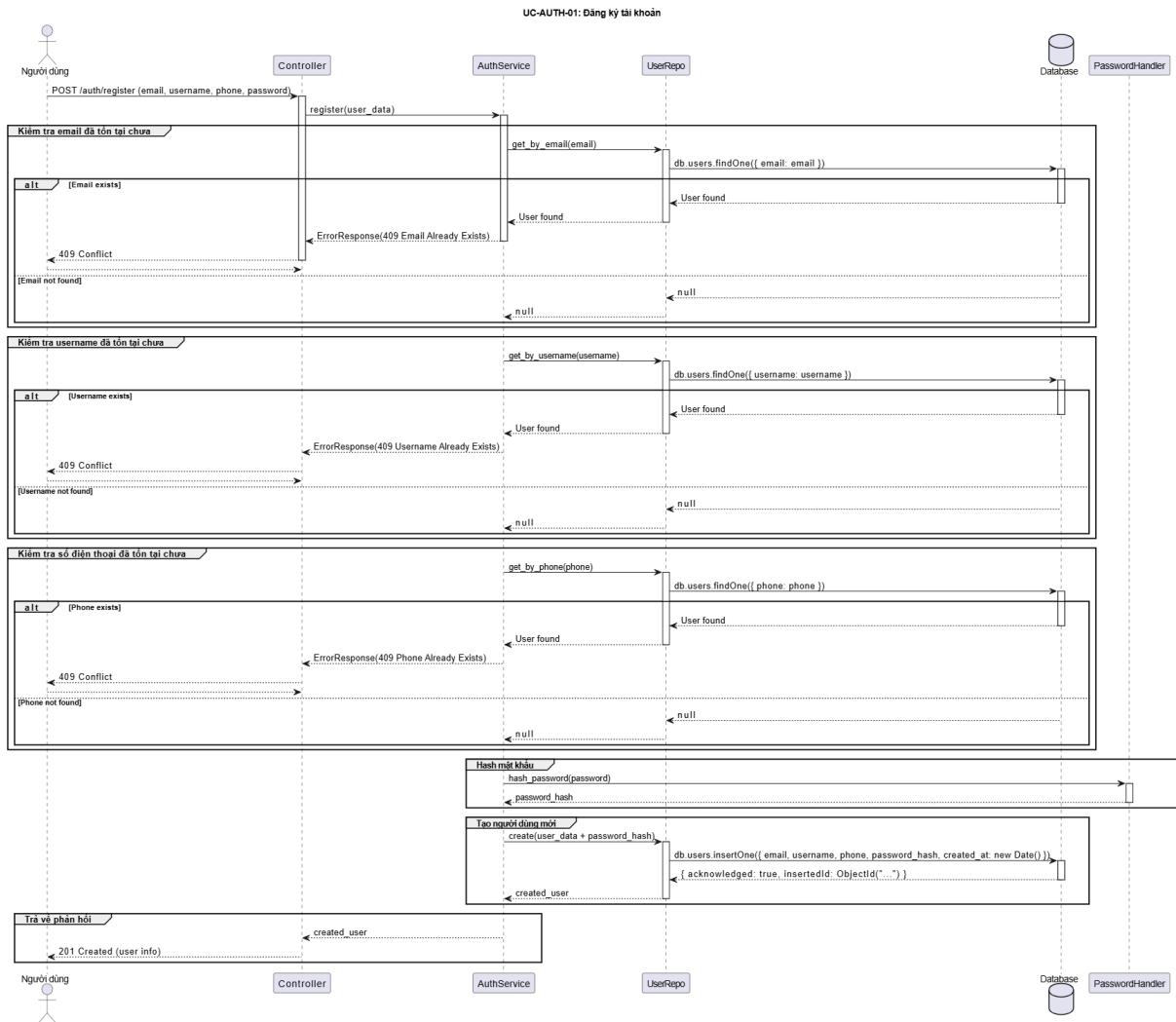
Hình 4.24: Sơ đồ hoạt động cho use-case "Tải tài liệu xuống"

Mô tả sơ đồ hoạt động: Việc tải tài liệu xuống

- Người dùng** chọn biểu tượng "Tải xuống" trên giao diện ứng dụng.
- Sau đó **hệ thống** kiểm tra quyền tải xuống rồi thực hiện truy xuất file cần tải xuống.
- Nếu có phát sinh lỗi, **hệ thống** sẽ hỏi xem **người dùng** muốn tiếp tục tải xuống lại nữa không, nếu người dùng muốn thì sẽ tiếp tục thử tải lại, còn nếu không thì về màn hình chính.
- Khi việc tải diễn ra thành công, file sẽ nằm trên máy **người dùng**.

5 Sequence Diagrams

5.1 Quản lý truy cập và xác thực

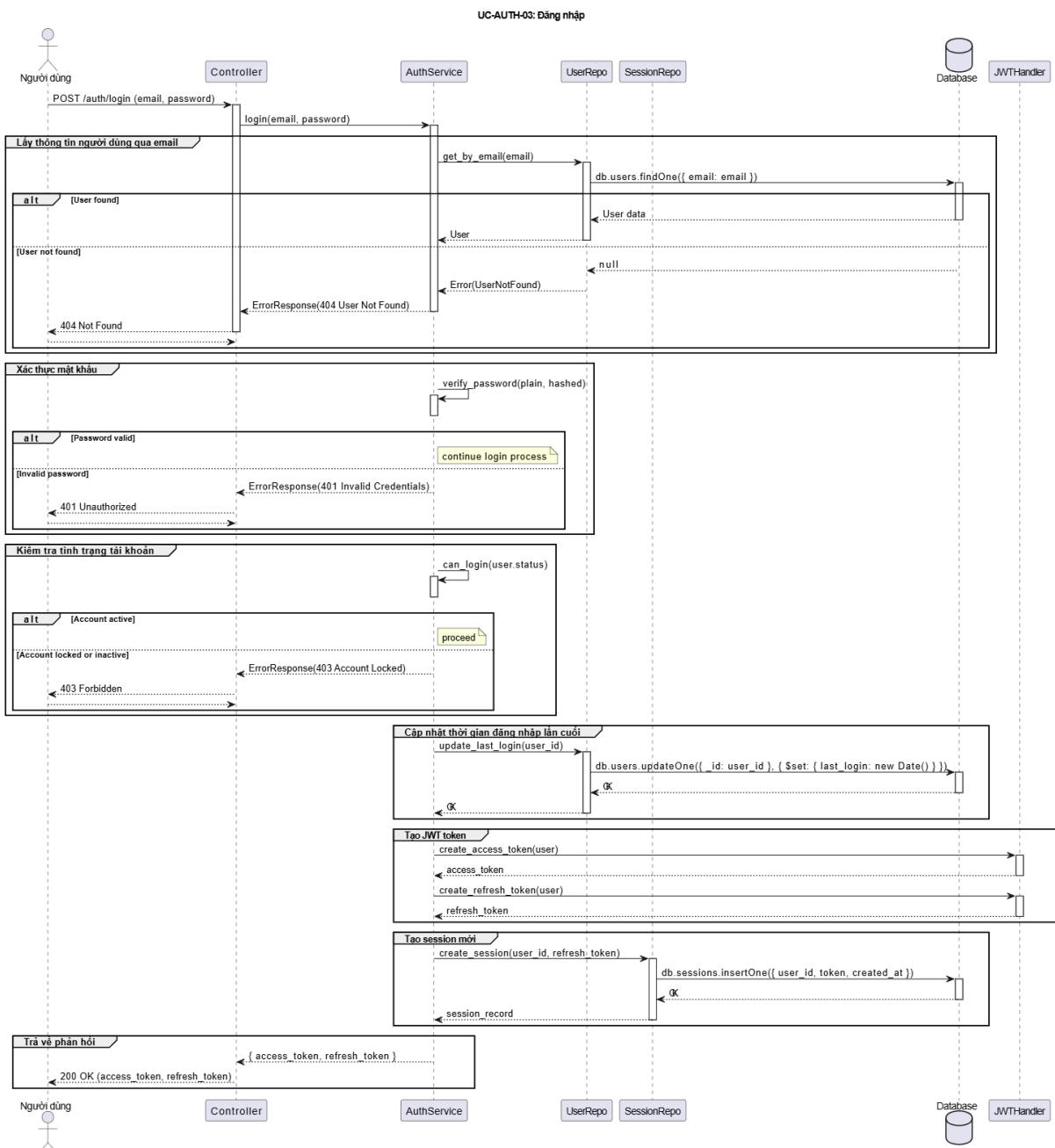


Hình 5.1: Sơ đồ tuần tự cho use-case "Quên mật khẩu"



Mô tả sơ đồ tuần tự: Đăng ký

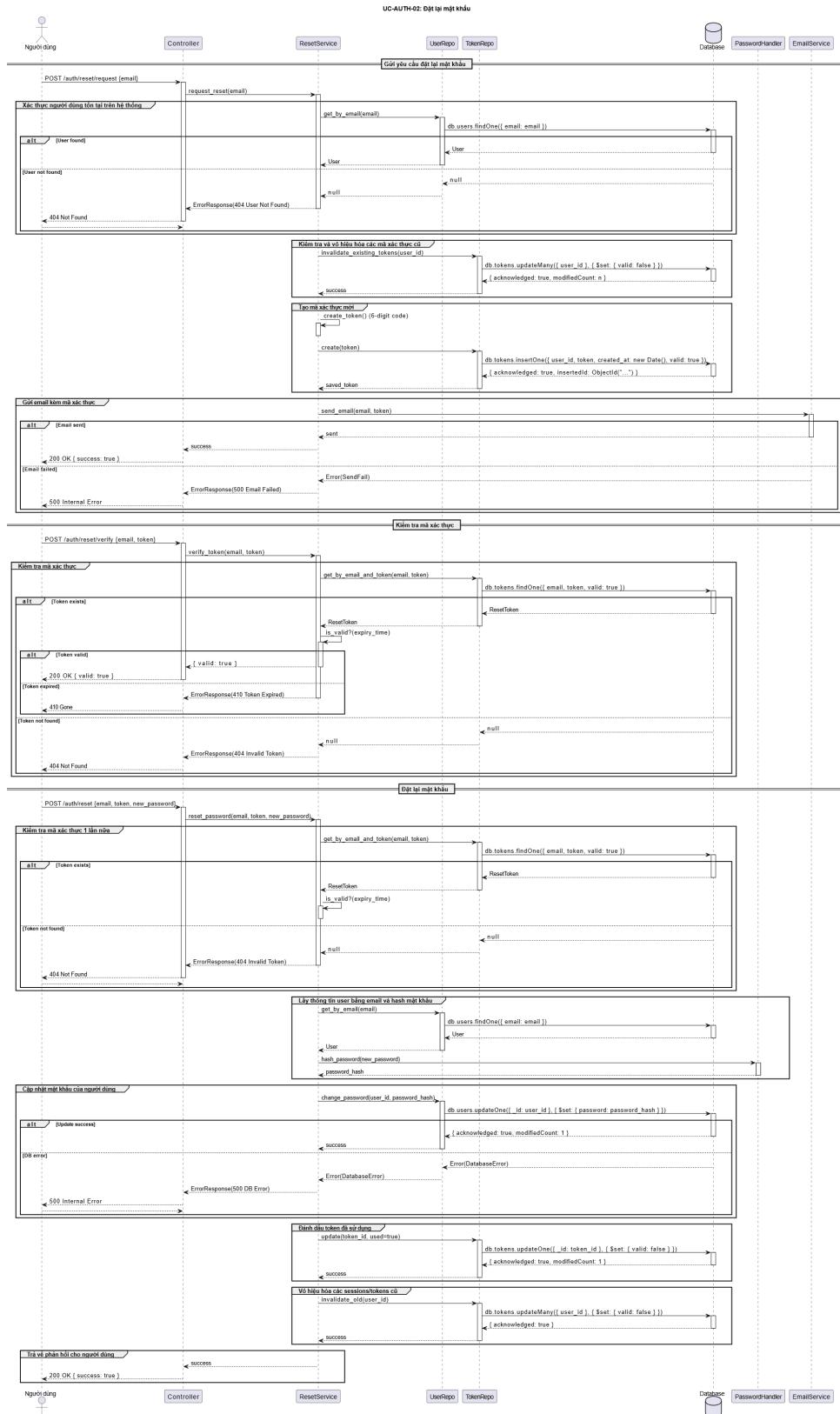
- Người dùng gửi yêu cầu mở giao diện đăng ký qua GET `/auth/register`. Controller trả về form đăng ký.
- Người dùng điền đầy đủ thông tin và gửi yêu cầu tạo tài khoản POST `/auth/register`.
- Controller chuyển dữ liệu sang AuthService để xử lý.
- AuthService thực hiện kiểm tra hợp lệ: email duy nhất, username chưa tồn tại, mật khẩu đủ mạnh.
- Dịch vụ lưu tài khoản mới vào UserRepository và ghi nhận trạng thái ban đầu cho người dùng.
- Sau khi lưu thành công, hệ thống phản hồi thông báo đăng ký thành công.
- Nếu dữ liệu không hợp lệ: trả về thông báo lỗi và yêu cầu nhập lại.



Hình 5.2: Sơ đồ tuần tự cho use-case "Đăng nhập"

Mô tả sơ đồ tuần tự: Đăng nhập

- Người dùng gửi yêu cầu hiển thị form đăng nhập qua GET /auth/login. Controller trả về giao diện đăng nhập.
- Người dùng nhập email/username và mật khẩu, gửi yêu cầu đăng nhập POST /auth/login.
- Controller tiếp nhận yêu cầu, gọi AuthService để xác thực thông tin.
- AuthService truy vấn UserRepository để kiểm tra tài khoản, mật khẩu, trạng thái hoạt động.
- Nếu thông tin hợp lệ: AuthService tạo phiên đăng nhập, sinh JWT hoặc session phù hợp và trả về Controller.
- Controller phản hồi lại người dùng kèm theo token/phiên và chuyển hướng tới trang cá nhân.
- Nếu thông tin không hợp lệ: hệ thống trả về thông báo lỗi xác thực và yêu cầu thử lại.

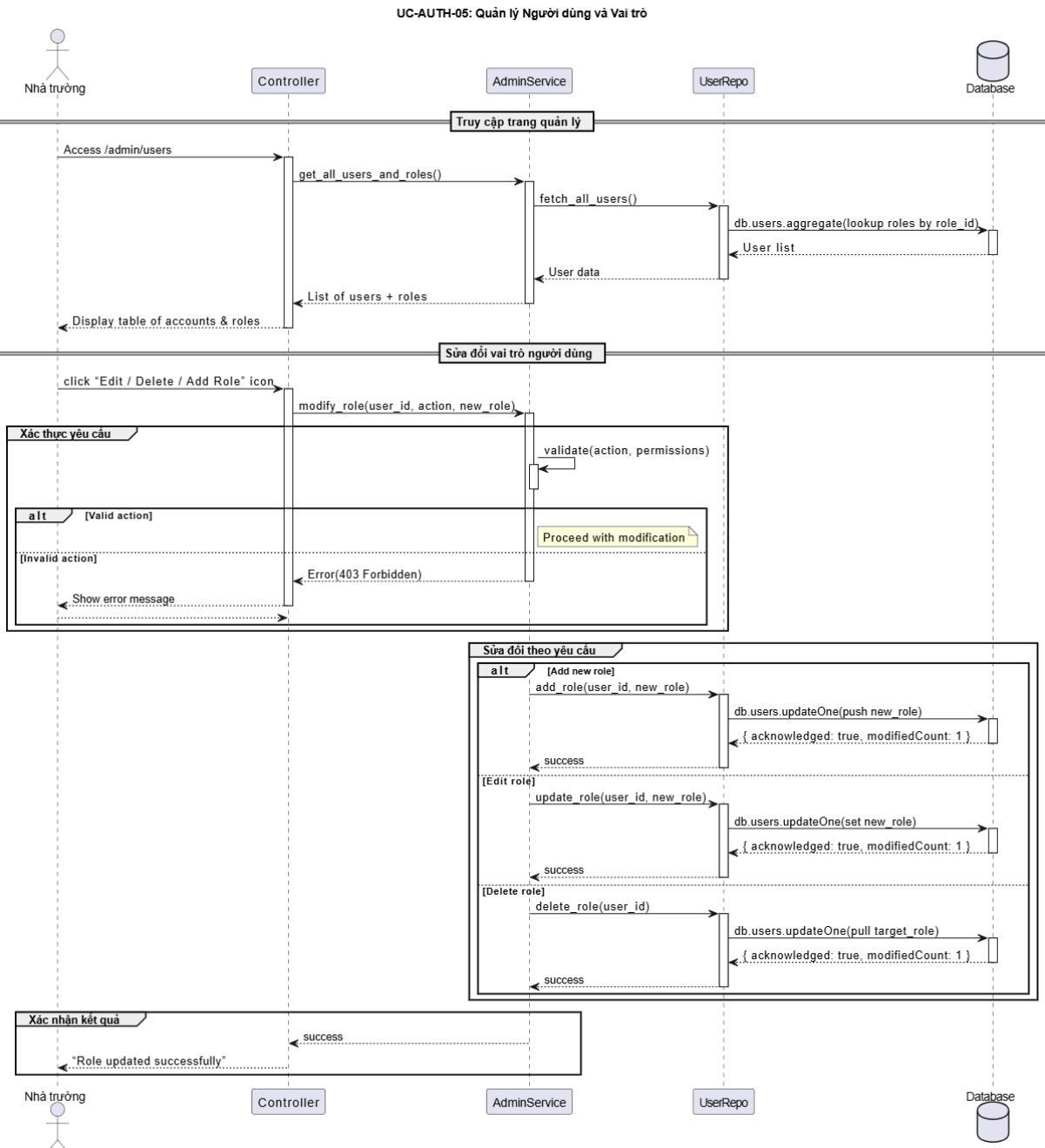


Hình 5.3: Sơ đồ tuần tự cho use-case "Đặt lại mật khẩu"



Mô tả sơ đồ tuần tự: Đặt lại mật khẩu

- Người dùng gửi yêu cầu lấy mã khôi phục qua POST /auth/reset/request. Controller gọi ResetService để kiểm tra email và tạo mã đặt lại mới.
- ResetService lưu mã vào TokenRepo và yêu cầu EmailService gửi mã xác thực đến người dùng.
- Người dùng gửi mã xác thực để kiểm tra qua POST /auth/reset/verify. ResetService xác minh mã và phản hồi kết quả.
- Khi mã hợp lệ, người dùng gửi mật khẩu mới qua POST /auth/reset. ResetService cập nhật mật khẩu trong UserRepo.
- ResetService vô hiệu hóa các mã đặt lại còn hiệu lực và phản hồi thành công cho người dùng.

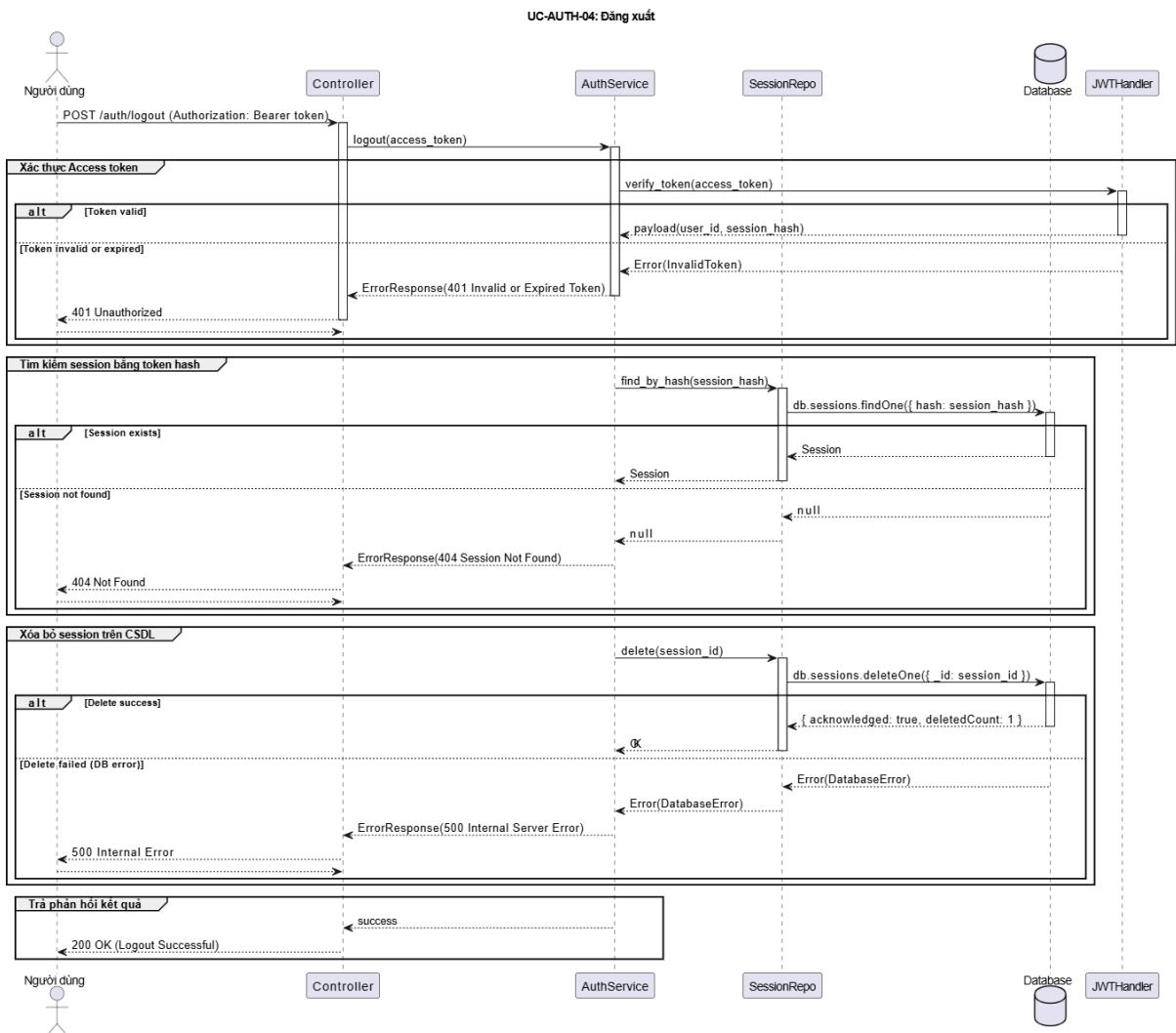


Hình 5.4: Sơ đồ tuần tự cho use-case "Quản lý người dùng"



Mô tả sơ đồ tuần tự: Quản lý người dùng và vai trò

- Nhà trường truy cập trang quản lý người dùng qua GET `/admin/users`. Controller gọi AdminService để lấy danh sách người dùng và vai trò.
- AdminService truy vấn UserRepository và trả dữ liệu người dùng về Controller.
- Khi người dùng quản trị chọn chỉnh sửa vai trò, Controller gửi yêu cầu `modify_role(user_id, action, role)` đến AdminService.
- AdminService kiểm tra quyền thực hiện và thao tác tương ứng (thêm/sửa/xóa vai trò) thông qua UserRepository.
- Sau khi cập nhật thành công, Controller phản hồi kết quả cho Nhà trường.

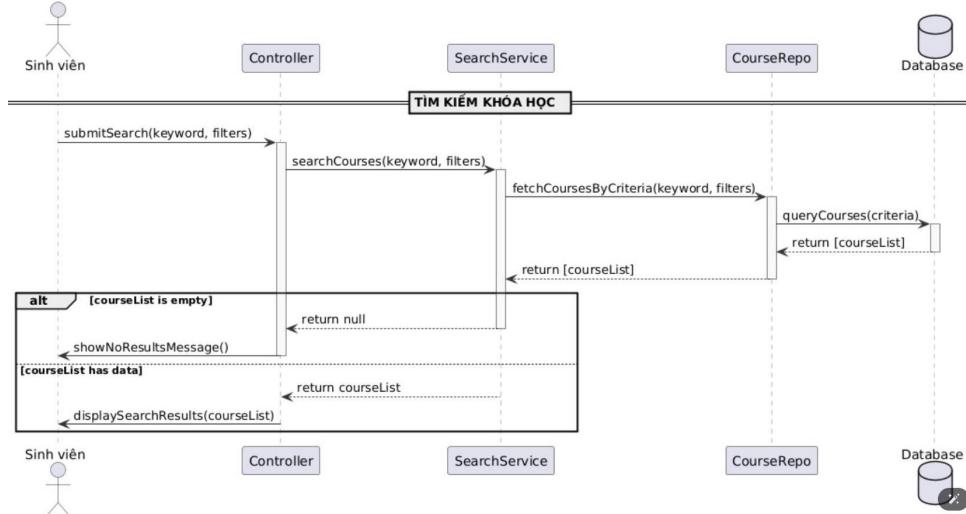


Hình 5.5: Sơ đồ tuần tự cho use-case "Đăng xuất"

Mô tả sơ đồ tuần tự: Đăng xuất

- Người dùng gửi yêu cầu đăng xuất qua POST /auth/logout.
- Controller tiếp nhận và gọi AuthService để hủy phiên làm việc hiện tại.
- AuthService vô hiệu hóa token hoặc xóa session khỏi SessionRepository.
- Hệ thống phản hồi xác nhận đăng xuất thành công và chuyển người dùng về giao diện Home.

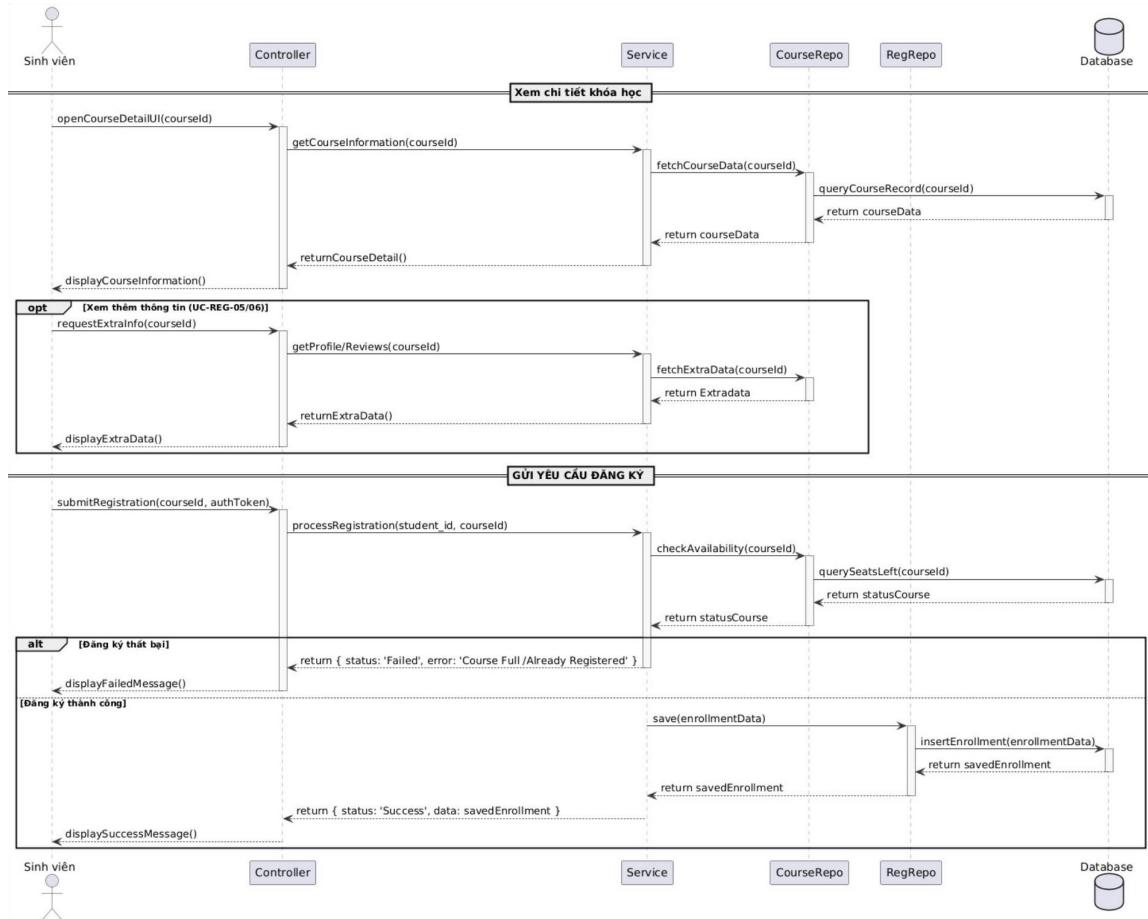
5.2 Đăng ký chương trình



Hình 5.6: Sơ đồ tuần tự cho use-case "Tìm kiếm chương trình"

Mô tả sơ đồ tuần tự: Tìm kiếm chương trình

- Khởi tạo:** Sinh viên gửi yêu cầu `searchCourses(query)` đến Controller.
- Gọi Service:** Controller ủy quyền xử lý cho Service bằng cách gọi `execute(query)`.
- Truy xuất Dữ liệu:** Service gọi `findByCriteria(query)` đến Repo. Repo thực thi `queryCourses` xuống Database và nhận về dữ liệu thô.
- Trả về Dữ liệu (Entity):** Repo trả `List<Course>` về cho Service, và Service trả tiếp về cho Controller.
- Xử lý :** Controller tự gọi hàm nội bộ `mapToSummary` để chuyển đổi danh sách Course (Entity) thành `List<CourseSummary>` trước khi gửi về cho người dùng.
- Hiển thị Kết quả (Khối alt):**
 - Nếu danh sách rỗng (`courseList is empty`), Controller gọi `displayNoResults()` để thông báo cho Sinh viên.
 - Nếu có dữ liệu (`has data`), Controller gọi `displaySearchResults()` với danh sách kết quả trả về.



Hình 5.7: Sơ đồ tuần tự cho use-case "Đăng ký chương trình"

Mô tả sơ đồ tuần tự: Đăng ký chương trình

Sơ đồ này mô tả hai luồng nghiệp vụ riêng biệt, được xử lý bởi hai Service con.

Luồng 1: Xem chi tiết khóa học (GetCourseDetailsService)

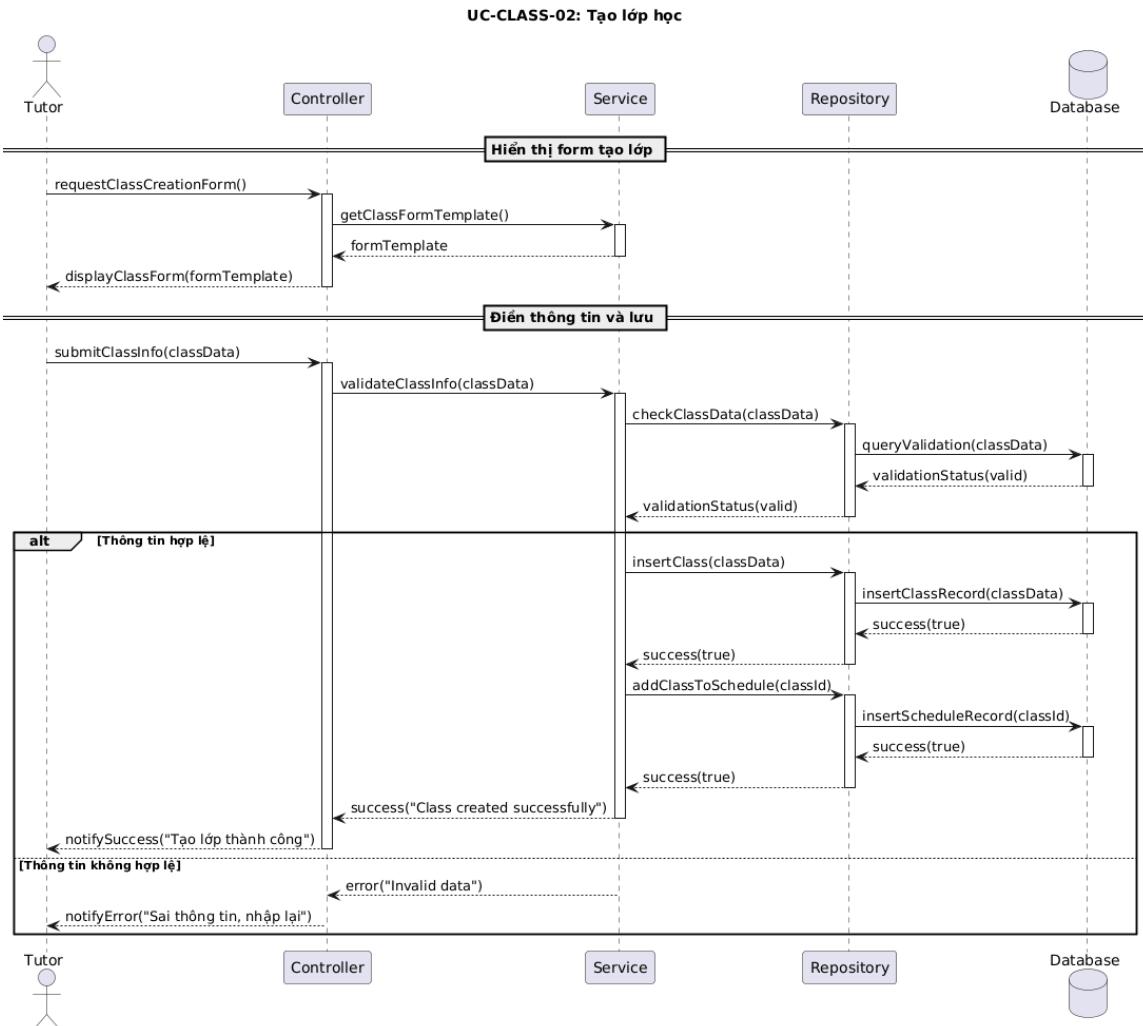
- Khởi tạo:** Sinh viên yêu cầu openCourseDetailUI từ Controller.
- Gọi Service:** Controller gọi getCourseInformation từ Service .
- Truy xuất Dữ liệu:** Service gọi fetchCourseData từ CourseRepo, CourseRepo truy vấn Database để lấy dữ liệu.
- Trả về:** Dữ liệu được trả về qua các tầng và Controller hiển thị thông tin cho Sinh viên.
- Luồng tùy chọn (Khối opt):** Mô tả luồng không bắt buộc để xem thêm thông tin (ví dụ: đánh giá).



Luồng 2: Gửi yêu cầu đăng ký (RegisterProgramService)

- **Khởi tạo:** Sinh viên gửi `submitRegistration` đến Controller.
- **Gọi Service:** Controller gọi `processRegistration` của Service (`RegisterProgramService`).
- **Kiểm tra nghiệp vụ:** Service thực hiện hai lần kiểm tra quan trọng:
 - **Kiểm tra 1:** Gọi `findEnrollment` đến `RegRepo` để kiểm tra sinh viên đã đăng ký môn này chưa.
 - **Kiểm tra 2:** Gọi `checkAvailability` đến `CourseRepo` để kiểm tra khóa học có còn chỗ hay không.
- **Xử lý Kết quả (Khối alt):**
 - **Thất bại:** Nếu một trong hai kiểm tra trên thất bại (đã đăng ký hoặc hết chỗ), Service trả thông báo lỗi về Controller để hiển thị.
 - **Thành công:** Nếu hợp lệ, Service gọi `save(enrollmentData)` đến `RegRepo` để ghi lượu đăng ký mới vào Database.

5.3 Tạo chương trình học



Hình 5.8: Sơ đồ tuần tự cho use-case "Tạo lớp học"

Mô tả sơ đồ tuần tự: Tạo lớp học

Hiển thị form tạo lớp

- Tutor gửi yêu cầu `requestClassCreationForm()` đến Controller để hiển thị form tạo lớp.
- Controller gọi phương thức `getClassFormTemplate()` đến Service.
- Service truy cập Repository để lấy mẫu form (nếu cần).
- Service gửi lại `formTemplate` cho Controller.
- Controller hiển thị form tạo lớp cho Tutor bằng `displayClassForm(formTemplate)`.



Điền thông tin và lưu

Tutor gửi thông tin lớp:

- Tutor nhập thông tin lớp học và gửi dữ liệu đến Controller thông qua `submitClassInfo(classData)`.
- Controller chuyển dữ liệu này cho Service bằng phương thức `validateClassInfo(classData)`.

Kiểm tra tính hợp lệ

- Service gọi `checkClassData(classData)` đến Repository để kiểm tra dữ liệu.
- Repository truy vấn cơ sở dữ liệu qua `queryValidation(classData)`.
- Database trả về kết quả kiểm tra (`validationStatus`).
- Repository gửi `validationStatus` (valid/invalid) về Service.

Xử lý theo điều kiện

Trường hợp 1: Thông tin hợp lệ (`validationStatus = valid`)

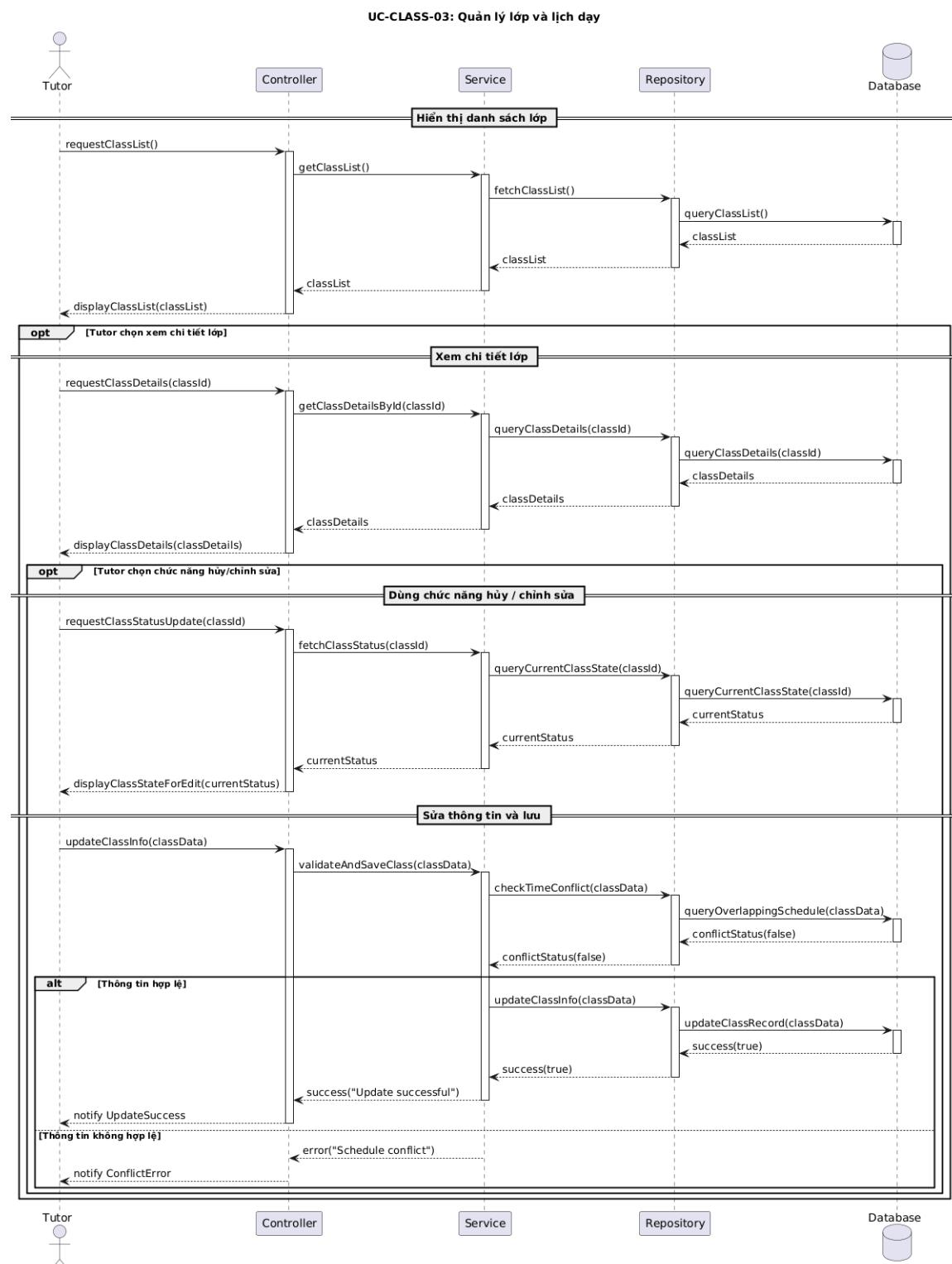
- Service gọi `insertClass(classData)` đến Repository để lưu lớp học.
- Repository thực hiện lưu dữ liệu lớp bằng `insertClassRecord(classData)`.
- Database trả về `success(true)` cho Repository.
- Repository tiếp tục thực hiện `insertScheduleRecord(classId)` để thêm lớp học vào lịch giảng dạy.
- Database phản hồi `success(true)`.
- Repository trả kết quả thành công cho Service.
- Service phản hồi Controller bằng `success("Class created successfully")`.
- Controller thông báo kết quả cho Tutor qua `notifySuccess("Tạo lớp thành công")`.

Trường hợp 2: Thông tin không hợp lệ (`validationStatus = invalid`)

- Service trả về lỗi cho Controller bằng `error("Invalid data")`.
- Controller thông báo lỗi cho Tutor bằng `notifyError("Sai thông tin, nhập lại")`.

Kết thúc

Quy trình kết thúc khi hệ thống phản hồi kết quả cho Tutor, bao gồm: tạo lớp thành công hoặc thông báo lỗi yêu cầu nhập lại thông tin.



Hình 5.9: Sơ đồ tuần tự cho use-case "Quản lý lớp học"



Mô tả sơ đồ tuần tự: Quản lý lớp học

Hiển thị danh sách lớp

- Tutor gửi yêu cầu `requestClassList()` đến Controller.
- Controller gọi `getClassList()` ở Service.
- Service tiếp tục gọi `fetchClassList()` đến Repository.
- Repository truy vấn CSDL bằng `queryClassList()`.
- Danh sách lớp `classList` được trả về theo chiều ngược lại qua Repository → Service → Controller.
- Controller hiển thị danh sách lớp cho Tutor: `displayClassList(classList)`.

Xem chi tiết lớp (tùy chọn)

- Tutor chọn một lớp để xem chi tiết và gọi `requestClassDetails(classId)`.
- Controller gọi `getClassDetailsById(classId)` ở Service.
- Service gọi tiếp `queryClassDetails(classId)` tại Repository.
- Repository truy vấn database và trả về thông tin chi tiết lớp `classDetails`.
- Thông tin được chuyển ngược lại lên Controller.
- Controller hiển thị chi tiết lớp: `displayClassDetails(classDetails)`.

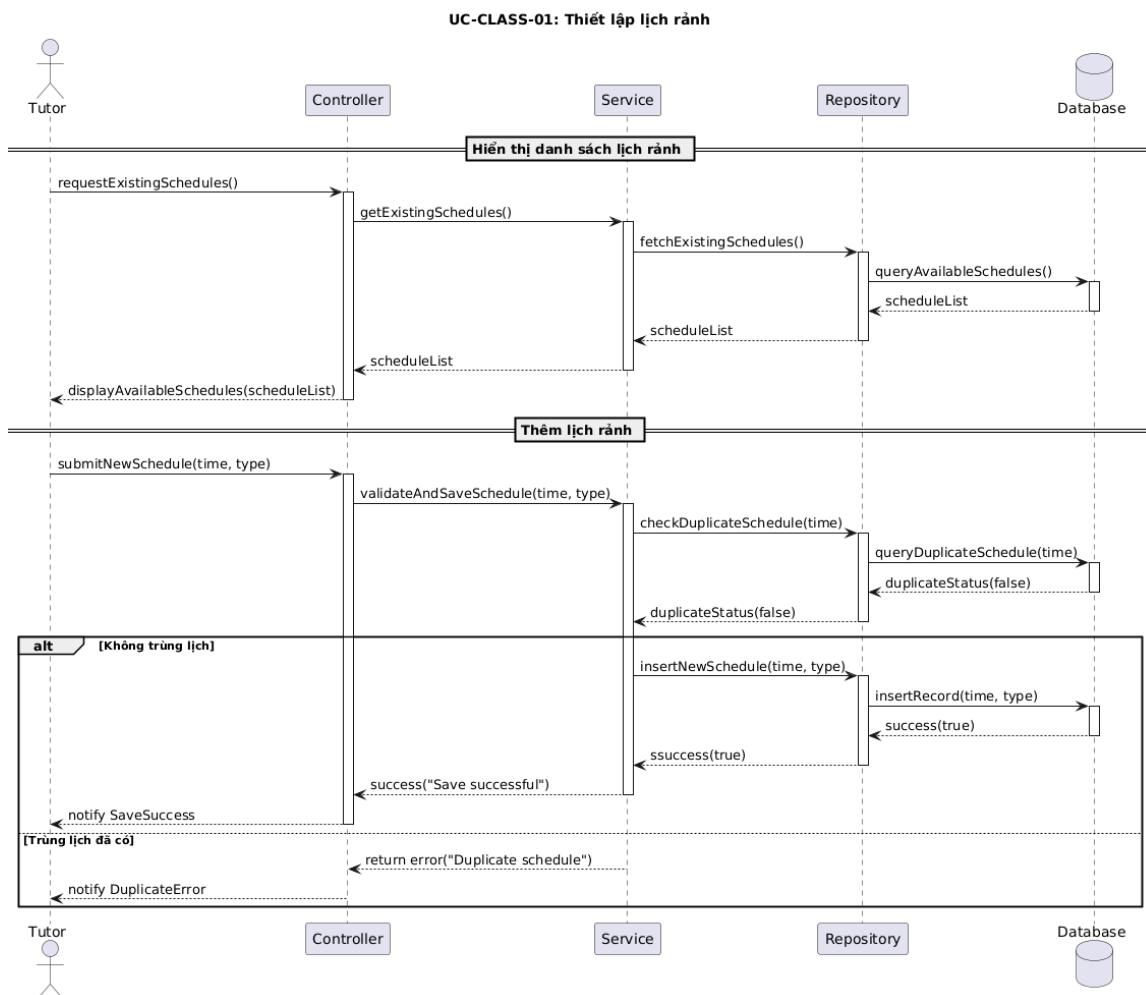
Dùng chức năng huỷ / chỉnh sửa (tùy chọn)

- Tutor yêu cầu chỉnh sửa/hủy thông tin lớp qua `requestClassStatusUpdate(classId)`.
- Controller gọi `fetchClassStatus(classId)` tại Service.
- Service gọi Repository với `queryCurrentClassState(classId)`.
- Repository kiểm tra database và trả về trạng thái hiện tại `currentStatus`.
- Trạng thái được truyền lên lại Controller.
- Controller hiển thị màn hình chỉnh sửa: `displayClassStateForEdit(currentStatus)`.

Sửa thông tin và lưu

- Tutor gửi yêu cầu lưu thay đổi: `updateClassInfo(classData)` đến Controller.
- Controller gọi Service: `validateAndSaveClass(classData)`.
- Service thực hiện kiểm tra trùng lịch bằng `checkTimeConflict(classData)`.
- Repository thực hiện truy vấn `queryOverlappingSchedule(classData)`.
- Nếu trả về `conflictStatus = false` nghĩa là không trùng lịch:

- Service gọi Repository lưu dữ liệu: `updateClassRecord(classData)`.
- Repository trả về `success(true)`.
- Controller thông báo thành công: `notify_UpdateSuccess`.
- Nếu có trùng lịch:
 - Service trả về lỗi `"Schedule conflict"`.
 - Controller hiển thị thông báo lỗi: `notify_ConflictError`.



Hình 5.10: Sơ đồ tuần tự cho use-case "Quản lý lịch rảnh"

Mô tả sơ đồ tuần tự: Quản lý lịch rảnh

Hiển thị danh sách lịch rảnh

- Tutor gửi yêu cầu `requestExistingSchedules()` đến Controller để lấy danh sách lịch rảnh.
- Controller gọi `getExistingSchedules()` đến Service.

- Service tiếp tục gọi `fetchExistingSchedules()` đến Repository.
- Repository truy vấn cơ sở dữ liệu thông qua `queryAvailableSchedules()`.
- Database trả về danh sách lịch rảnh (`scheduleList`) cho Repository.
- Repository gửi `scheduleList` về Service.
- Service trả danh sách lịch rảnh lại cho Controller.
- Controller hiển thị thông tin cho Tutor bằng lời gọi `displayAvailableSchedules(scheduleList)`.

Thêm lịch rảnh mới

1. Tutor gửi yêu cầu
 - Tutor gửi yêu cầu thêm lịch rảnh mới thông qua `submitNewSchedule(time, type)` đến Controller.
 - Controller chuyển yêu cầu đến Service qua phương thức `validateAndSaveSchedule(time, type)`.
2. Kiểm tra trùng lịch
 - Service gọi `checkDuplicateSchedule(time)` đến Repository để kiểm tra lịch trùng.
 - Repository truy vấn Database bằng `queryDuplicateSchedule(time)`.
 - Database trả về trạng thái trùng lịch (`duplicateStatus`).
 - Repository gửi `duplicateStatus (true/false)` về Service.
3. Xử lý theo điều kiện

Trường hợp 1: Không trùng lịch (`duplicateStatus = false`)

- Service gọi `insertNewSchedule(time, type)` đến Repository để lưu lịch.
- Repository thực hiện thêm bản ghi bằng `insertRecord(time, type)`.
- Database trả về kết quả thành công.
- Repository trả về `success(true)` cho Service.
- Service phản hồi Controller bằng `success("Save successful")`.
- Controller thông báo kết quả thành công cho Tutor qua `notify SaveSuccess`.

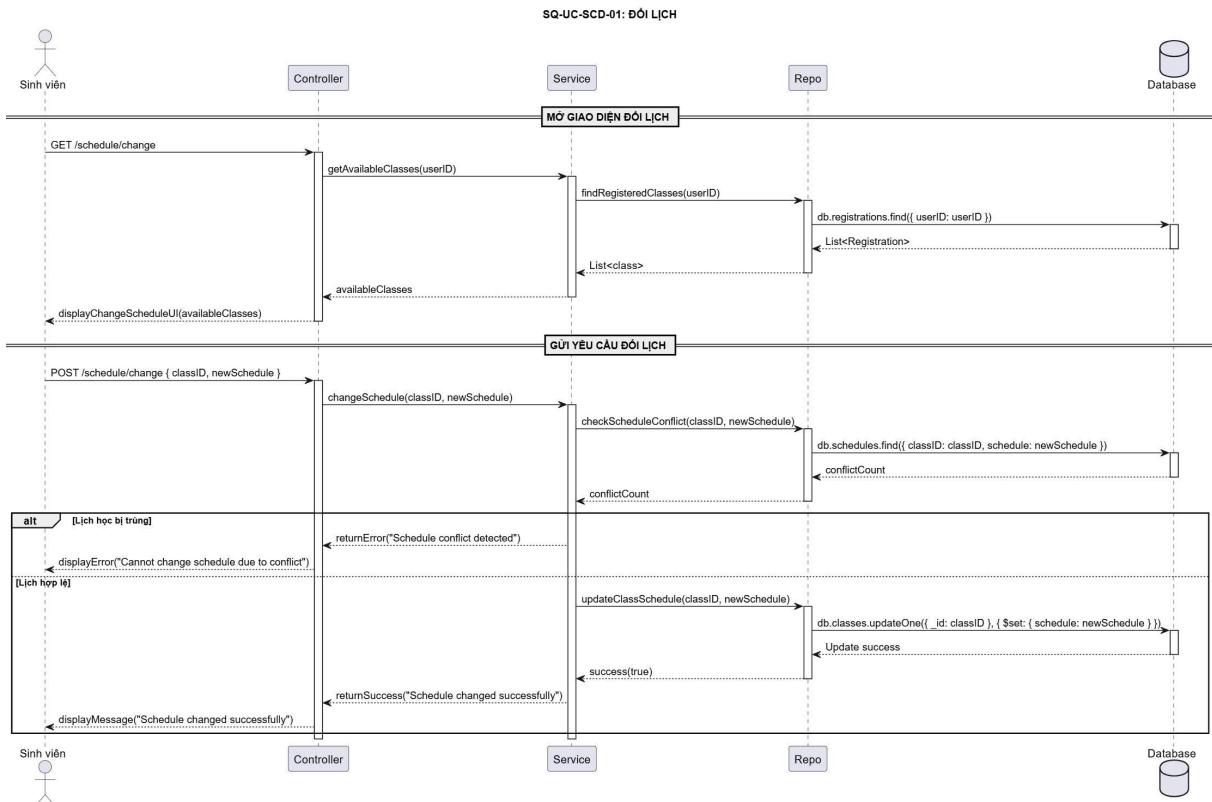
Trường hợp 2: Trùng lịch (`duplicateStatus = true`)

- Service trả về lỗi `return error("Duplicate schedule")` cho Controller.
- Controller thông báo cho Tutor thông qua `notify DuplicateError`.

Kết thúc

Quy trình kết thúc khi hệ thống phản hồi kết quả cho Tutor, bao gồm: thêm lịch thành công hoặc thông báo lỗi do trùng lịch.

5.4 Thiết lập lịch trình cho sinh viên



Hình 5.11: Sơ đồ tuần tự cho use-case "Đổi lịch học" cho sinh viên"

Mô tả sơ đồ tuần tự: Đổi lịch học

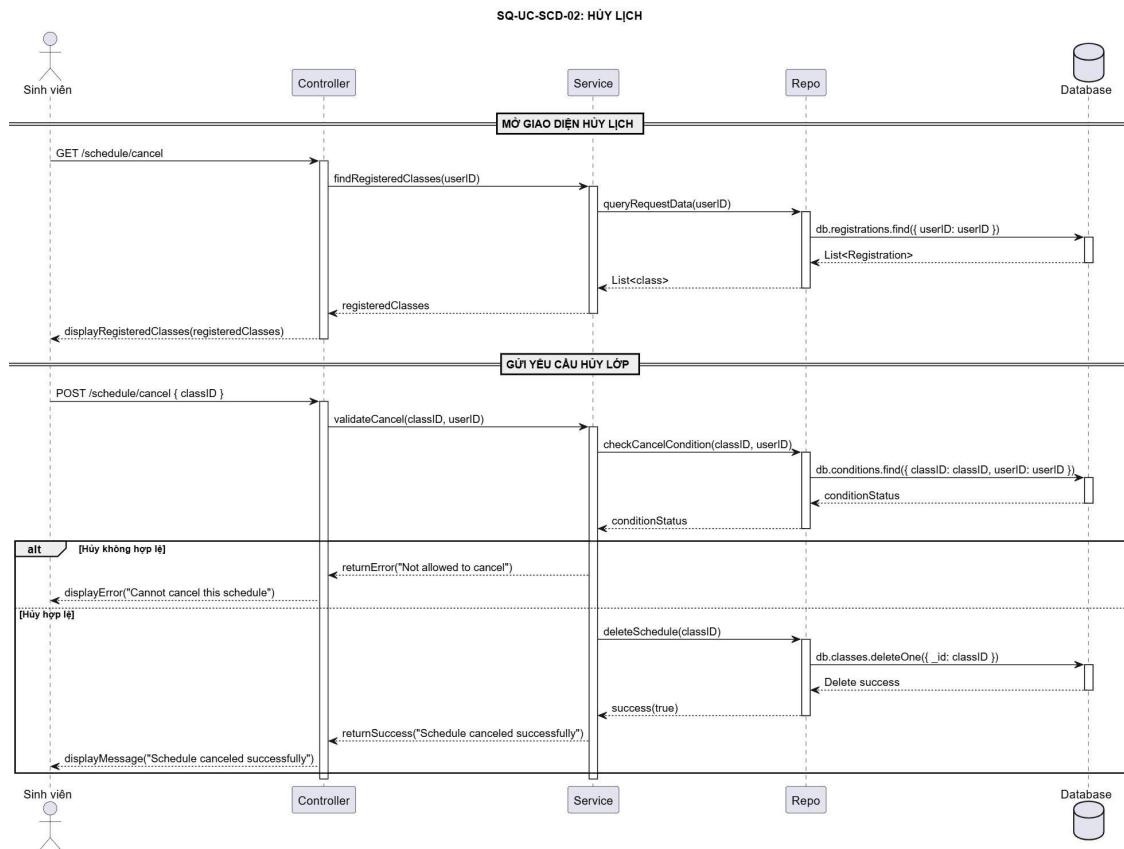
Bước 1: Chọn chức năng đổi lịch.

- Sinh viên chọn chức năng đổi lịch trên giao diện, tương ứng với thông điệp `requestChangeSchedule()` gửi đến Controller.
- Controller gọi phương thức `getAvailableClasses(userID)` của Service để lấy danh sách lớp mà sinh viên đã đăng ký.
- Service tiếp tục gọi `findRegisteredClasses(userID)` trên Repo.
- Repo truy vấn `db.registrations.find({ userID: userID })` để lấy danh sách đăng ký và trả về `List<Registration>` cho Service.
- Service chuyển danh sách này thành `availableClasses` và trả về cho Controller.
- Controller gửi lại kết quả cho sinh viên thông qua thông điệp `responseAvailableClasses(availableClasses)` và giao diện hiển thị danh sách lớp có thể đổi.



Bước 2: Chọn lớp cần đổi và gửi yêu cầu đổi lịch.

- Sau khi chọn lớp và ca học mới, sinh viên gửi yêu cầu `submitChangeRequest(classID, newSchedule)` tới Controller.
- Controller gọi `changeSchedule(classID, newSchedule)` trên Service.
- Service trước hết gọi `checkScheduleConflict(classID, newSchedule)`. Bên trong, Repo thực hiện truy vấn `db.schedules.find({ classID: classID, schedule: newSchedule })` để kiểm tra trùng lịch và trả về `conflictCount`.
- Biểu đồ sử dụng khối alt để tách hai trường hợp:
 - **Trường hợp lịch học bị trùng:** nếu `conflictCount > 0`, Service trả về thông điệp `returnError("Schedule conflict detected")` cho Controller. Controller hiển thị thông báo lỗi cho sinh viên thông qua `showError("Cannot change schedule due to conflict")`.
 - **Trường hợp lịch học hợp lệ:** nếu không có xung đột, Service gọi hàm `updateClassSchedule(classID, newSchedule)` trên Repo. Repo cập nhật bản ghi tương ứng trong cơ sở dữ liệu và trả về kết quả `success(true)`.
 - Sau khi cập nhật thành công, Service gửi lại thông điệp `responseSuccess("Schedule changed successfully")` cho Controller.
 - Controller hiển thị thông báo thành công cho sinh viên thông qua `showMessage("Schedule change completed successfully")`.



Hình 5.12: Sơ đồ tuần tự cho use-case "Hủy lịch học" cho sinh viên"

Mô tả sơ đồ tuần tự: Hủy lịch học

Bước 1: Chọn chức năng hủy lịch.

- Sinh viên chọn chức năng hủy lịch trên giao diện, tương ứng với thông điệp `cancelSchedule(request)` gửi đến Controller.
- Controller gọi `findRegisterClasses(userID)` trên Service để lấy danh sách các lớp mà sinh viên đã đăng ký.
- Service tiếp tục gọi `queryRequestData(userID)` trên Repo.
- Repo truy vấn `db.registrations.find({ userID: userID })` trên cơ sở dữ liệu và nhận về `List<Registration>`.
- Danh sách này được trả ngược lại cho Service, sau đó Service gửi thông điệp `showRegisteredClasses` cho Controller.
- Controller hiển thị danh sách lớp cho sinh viên thông qua lời gọi `hiểnThiDanhSáchLớp()` ở phía giao diện.

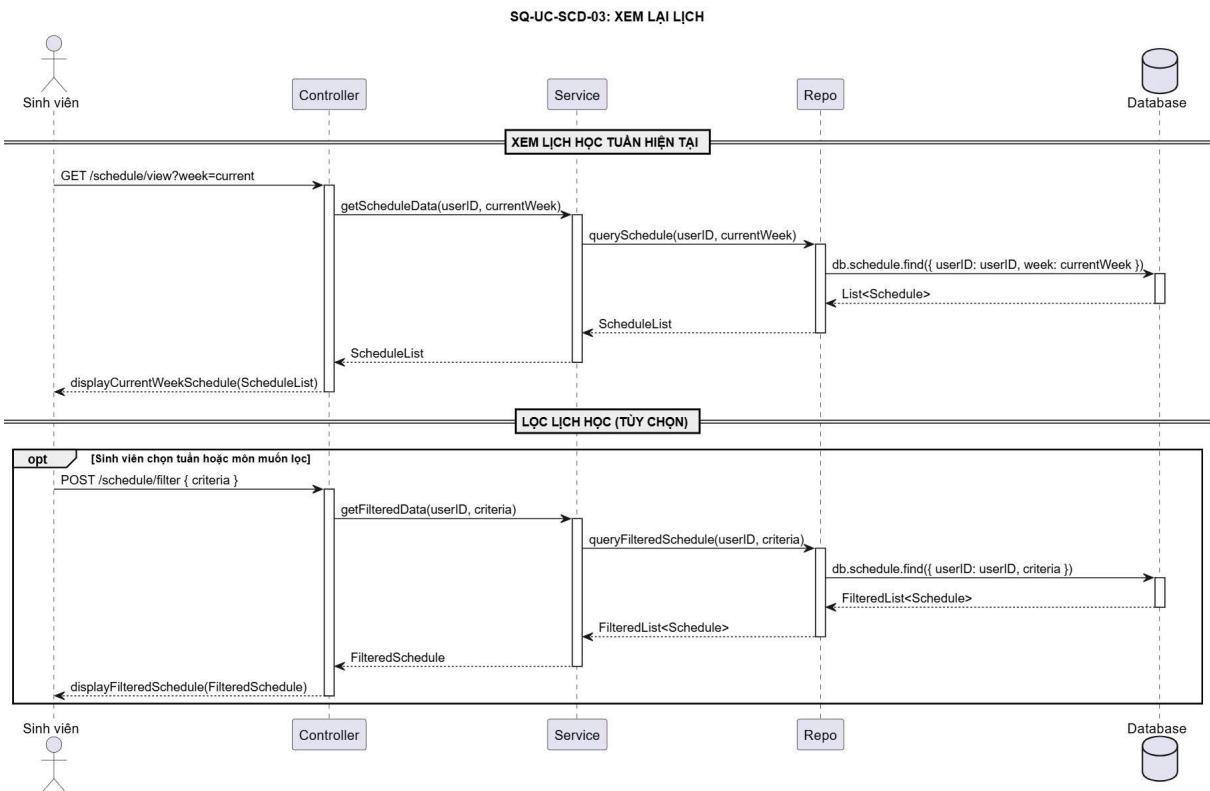


Bước 2: Chọn lớp cần hủy và xử lý yêu cầu.

- Sinh viên chọn một lớp trong danh sách, gửi thông điệp `selectClassToCancel(classID)` tới Controller.
- Controller gọi `validateCancel(classID, userID)` trên Service để kiểm tra xem lớp này có được phép hủy hay không.
- Service gọi tiếp `checkCancelCondition(classID, userID)` trên Repo.
- Repo truy vấn `db.conditions.find({ classID: classID, userID: userID })` và trả về `conditionStatus` phản ánh kết quả kiểm tra điều kiện hủy.

Sau khi nhận được `conditionStatus`, biểu đồ sử dụng khối alt để mô tả hai nhánh xử lý:

- Nhánh “**Hủy không hợp lệ**”:
 - Nếu điều kiện không thỏa (ví dụ đã quá hạn hủy), Service gửi thông điệp Error: `NotAllowed` về cho Controller.
 - Controller hiển thị thông báo lỗi cho sinh viên qua lời gọi `showError("Cannot cancel this schedule")`.
- Nhánh “**Hủy hợp lệ**”:
 - Nếu việc hủy được phép, Service gọi `deleteSchedule(classID)` trên Repo.
 - Repo thực hiện thao tác xóa trong cơ sở dữ liệu (ví dụ `db.classes.deleteOne({_id: classID})`) và nhận phản hồi `Delete success`.
 - Kết quả `success` được gửi ngược lại cho Service, sau đó Service trả về `ScheduleCancelResponse(success)` cho Controller.
 - Cuối cùng, Controller hiển thị thông báo `showMessage("Schedule canceled successfully")` để xác nhận hủy lịch thành công cho sinh viên.



Hình 5.13: Sơ đồ tuần tự cho use-case "Xem lại lịch học" cho sinh viên"

Mô tả sơ đồ tuần tự: Xem lại lịch học

Bước 1: Xem lịch học tuần hiện tại.

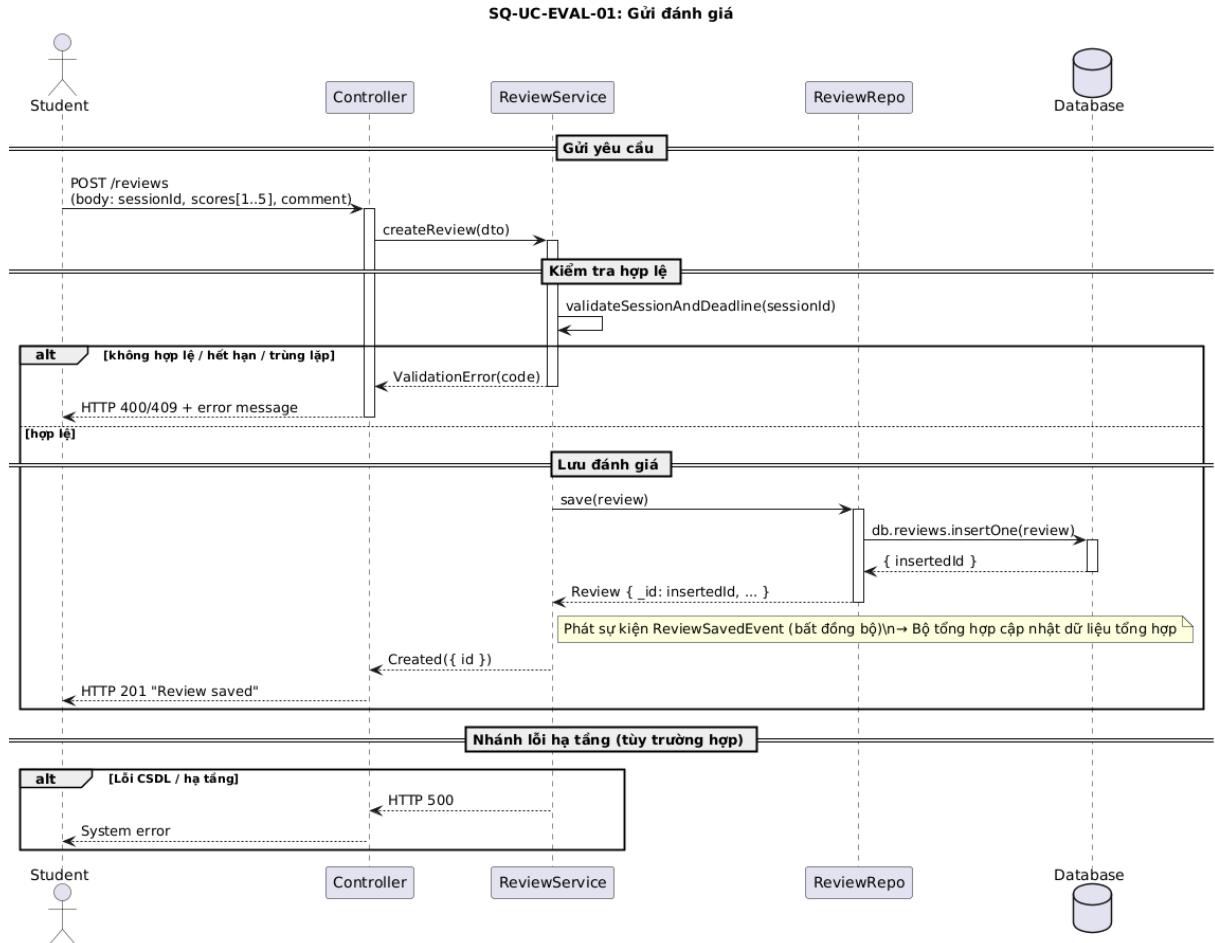
- Sinh viên yêu cầu xem lịch hiện tại thông qua thông điệp `getSchedule(userID, currentWeek)` gửi đến Controller.
- Controller chuyển tiếp yêu cầu sang Service bằng lời gọi `getScheduleData(userID, currentWeek)`.
- Service gọi `querySchedule(userID, currentWeek)` trên Repo để truy vấn dữ liệu lịch học.
- Repo truy vấn cơ sở dữ liệu và nhận về `List<Schedule>`.
- Danh sách này lần lượt được trả ngược về Service và Controller thông qua thông điệp `ScheduleList`.
- Cuối cùng, Controller hiển thị lịch học tuần hiện tại cho sinh viên thông qua lời gọi `displayCurrentWeekSchedule()`.



Bước 2: Lọc lịch học (tùy chọn).

- Khi sinh viên muốn xem lịch theo tuần khác hoặc lọc theo môn học, sinh viên gửi yêu cầu `getFilteredSchedule(userID, criteria)` tới Controller. (Khối opt trong sơ đồ cho biết bước này là tùy chọn.)
- Controller gọi `getFilteredData(userID, criteria)` trên Service.
- Service gọi `queryFilteredSchedule(userID, criteria)` trên Repo.
- Repo truy vấn cơ sở dữ liệu với điều kiện lọc tương ứng, ví dụ `db.schedule.find({ userID: userID, criteria })`, và nhận về `FilteredList<Schedule>`.
- Danh sách đã lọc được trả lại cho Service rồi cho Controller dưới dạng `FilteredSchedule`.
- Controller hiển thị kết quả cho sinh viên thông qua lời gọi `displayFilteredSchedule()`, cho phép sinh viên xem lịch học đã được lọc theo tuần hoặc môn mong muốn.

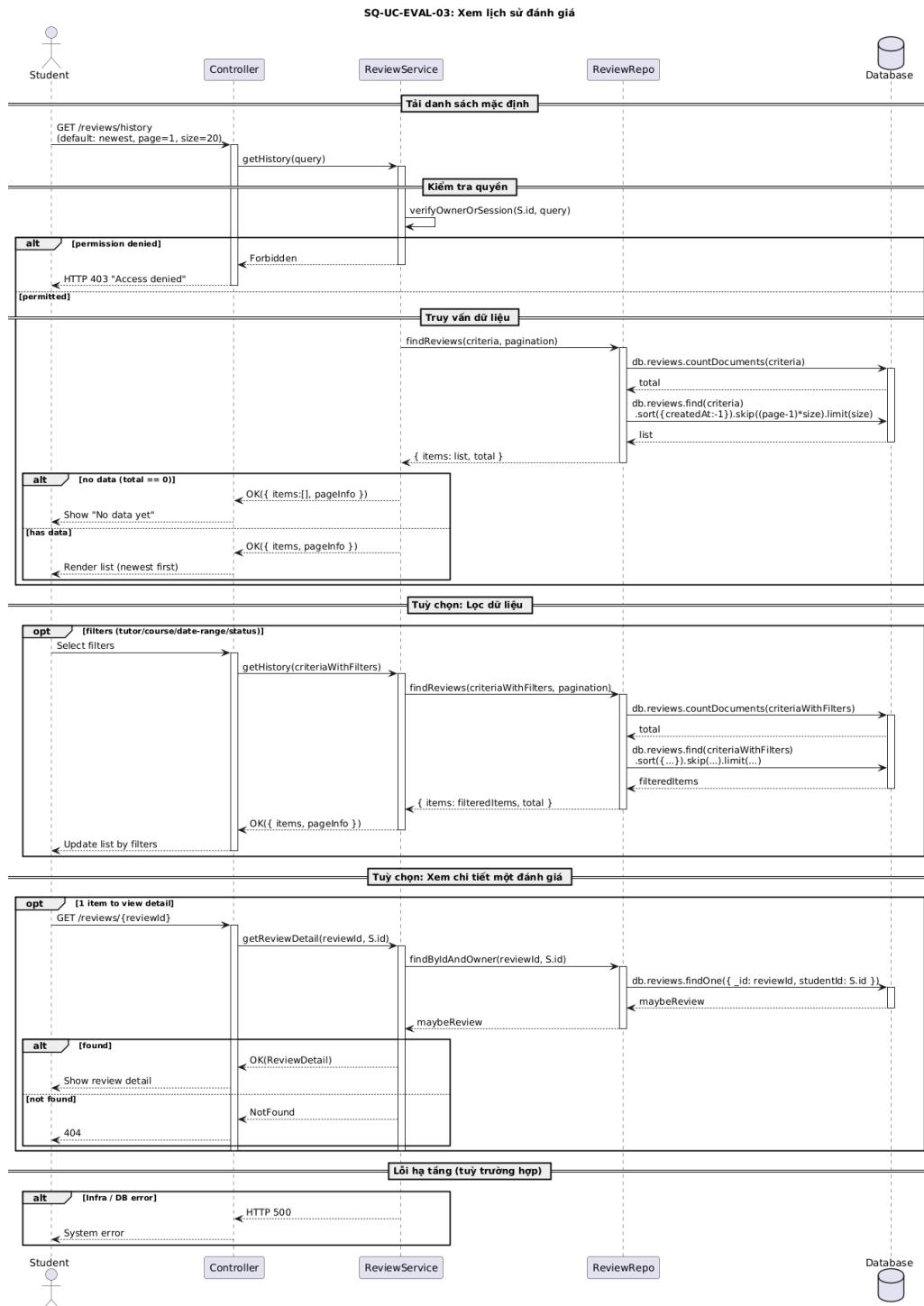
5.5 Đánh giá



Hình 5.14: Sơ đồ tuần tự cho use-case "Gửi đánh giá"

Mô tả sơ đồ tuần tự: Gửi đánh giá

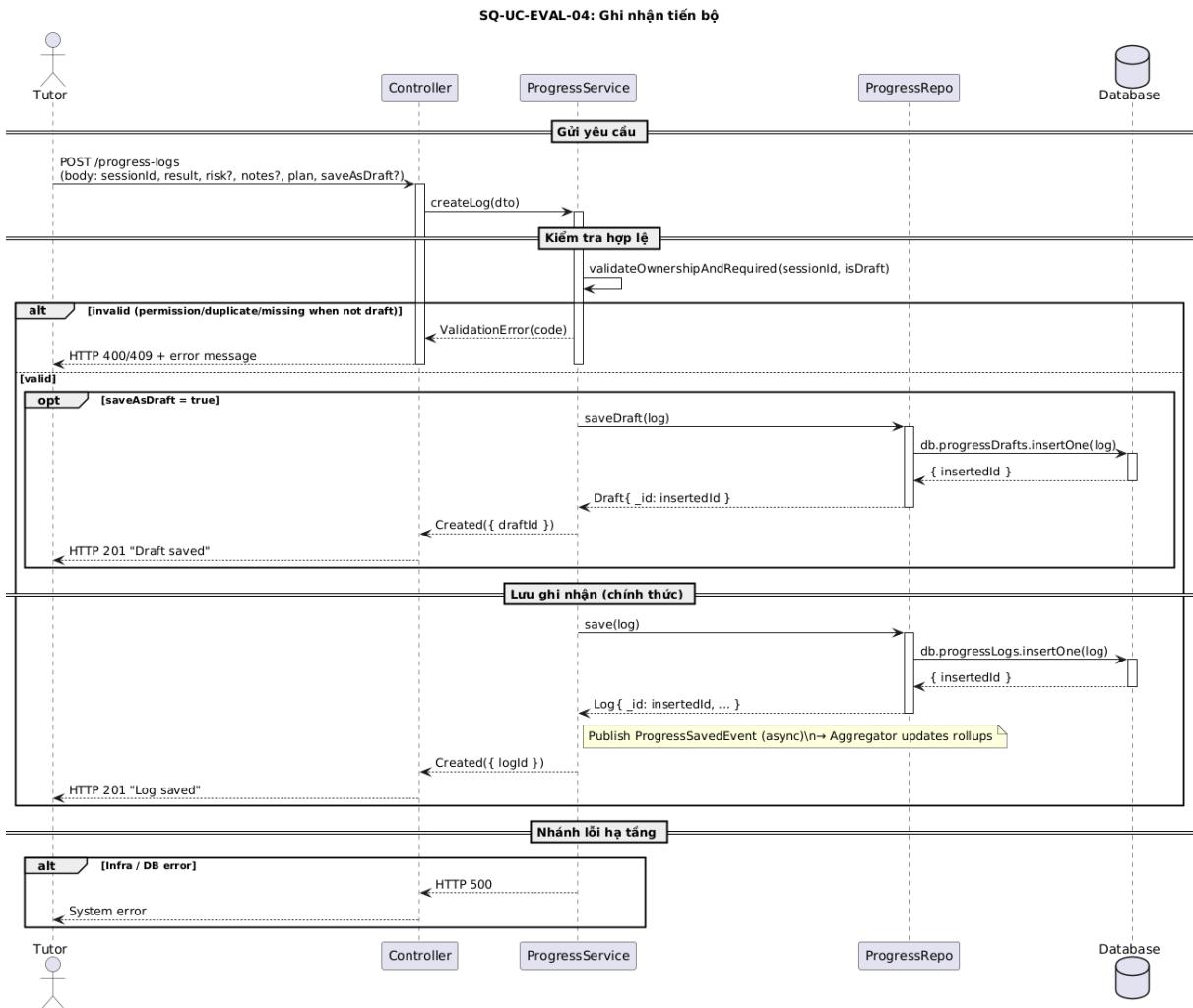
Controller gọi `ReviewService.createReview(dto)` → `validateSessionAndDeadline(sessionId)` → lưu vào `db.reviews` → phát `ReviewSavedEvent` cho bộ tổng hợp.



Hình 5.15: Sơ đồ tuần tự cho use-case "Xem lịch sử đánh giá"

Mô tả sơ đồ tuần tự: Xem lịch sử đánh giá

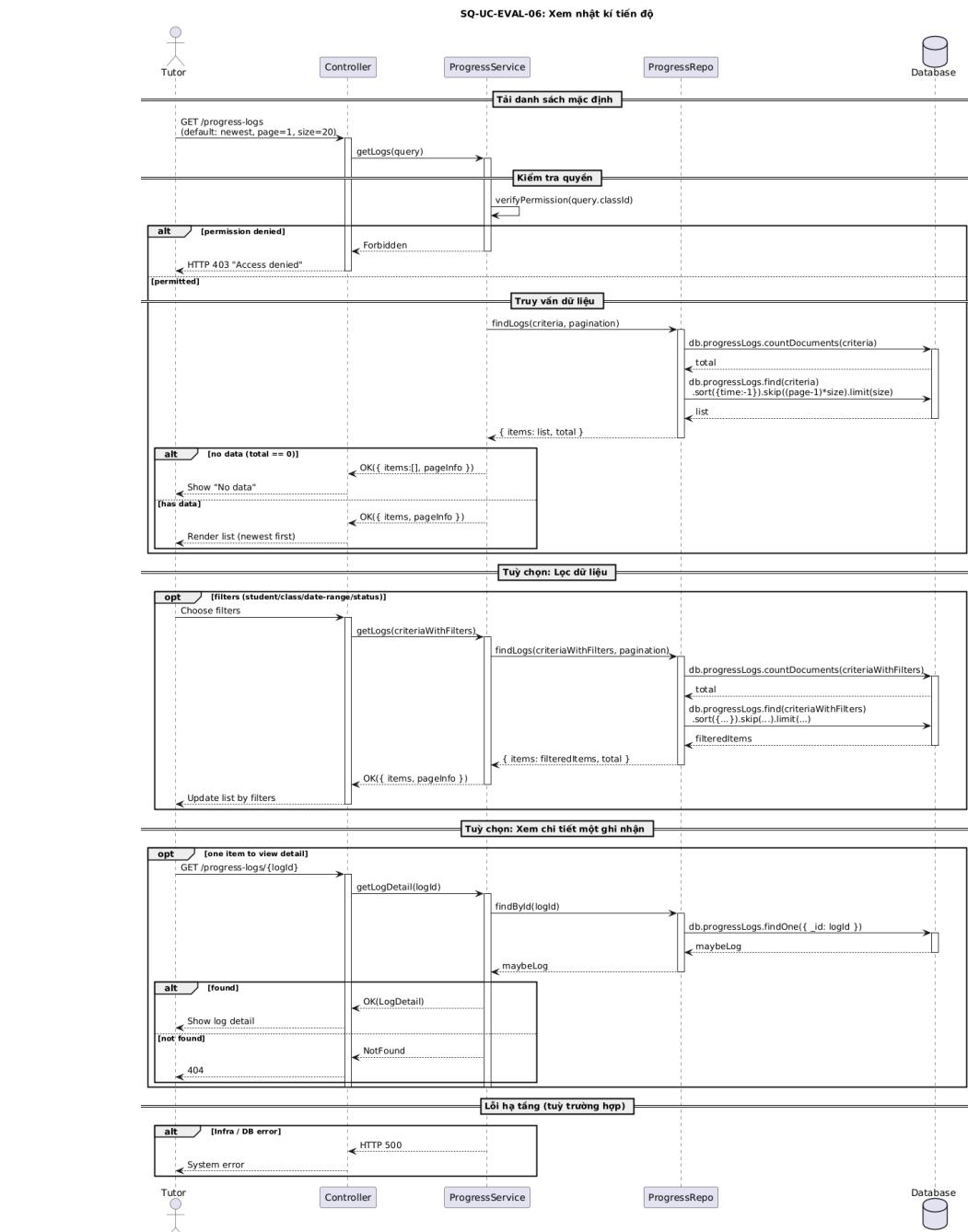
Controller nhận GET /reviews/history → ReviewService.verifyOwnerOrSession() → ReviewRepo.findReviews() truy vấn db.reviews với sắp xếp createdAt:-1 và phân trang.



Hình 5.16: Sơ đồ tuần tự cho use-case "Ghi nhận tiến bộ"

Mô tả sơ đồ tuần tự: Ghi nhận tiến bộ

Controller gọi `ProgressService.createLog(dto)` → `validateOwnershipAndRequired()` → lưu draft hoặc log vào DB; khi lưu chính thức, phát `ProgressSavedEvent` để bộ tổng hợp cập nhật số liệu.

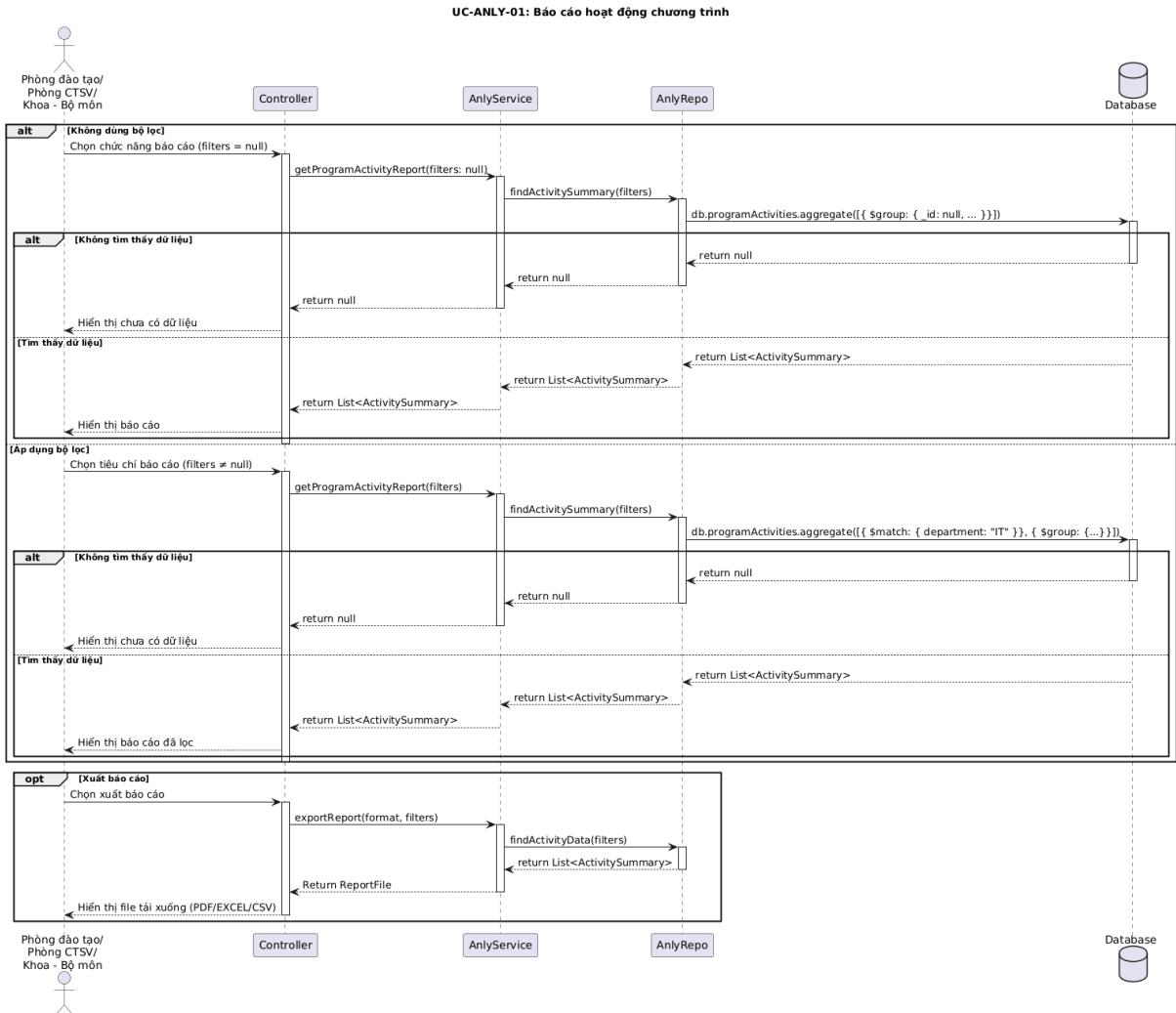


Hình 5.17: Sơ đồ tuần tự cho use-case "Xem nhật ký tiến độ"

Mô tả sơ đồ tuần tự: Xem nhật ký tiến độ

Controller nhận GET /progress-logs → ProgressService.verifyPermission() → ProgressRepo.findLogs(criteria,pagination) truy vấn db.progressLogs với sort(time:-1) và phân trang. Trường hợp không dữ liệu trả về danh sách rỗng.

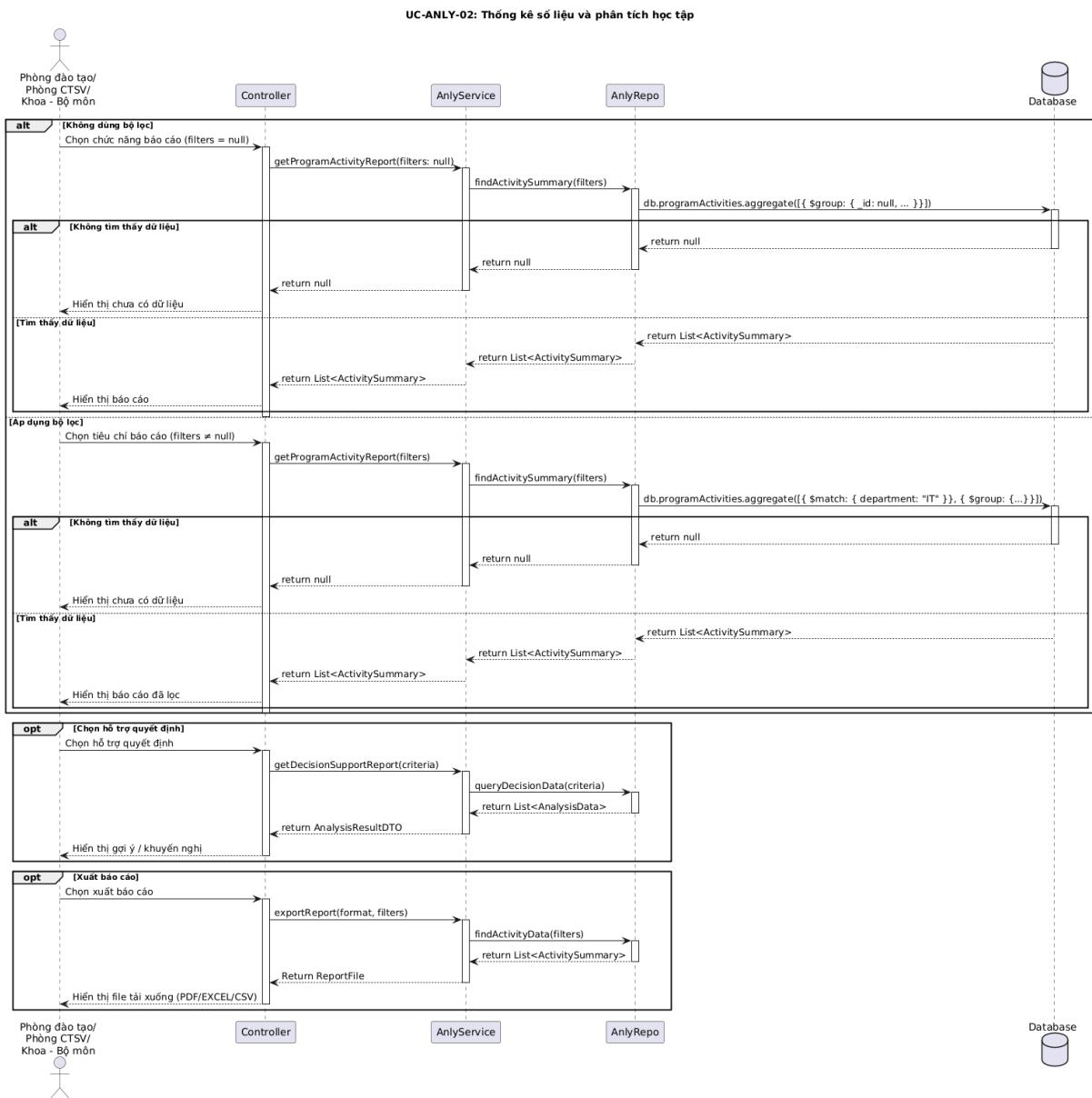
5.6 Phân tích và báo cáo



Hình 5.18: Sơ đồ tuần tự cho use-case "Báo cáo hoạt động chương trình"

Mô tả sơ đồ tuần tự: Báo cáo hoạt động chương trình

Controller gọi ReportService.generate_program_summary(filters) để lấy báo cáo chương trình và dữ liệu tổng hợp → sau đó ReportService gọi ReportRepositoryImpl.fetch_program_activity_data(filters) để truy xuất dữ liệu từ Database.

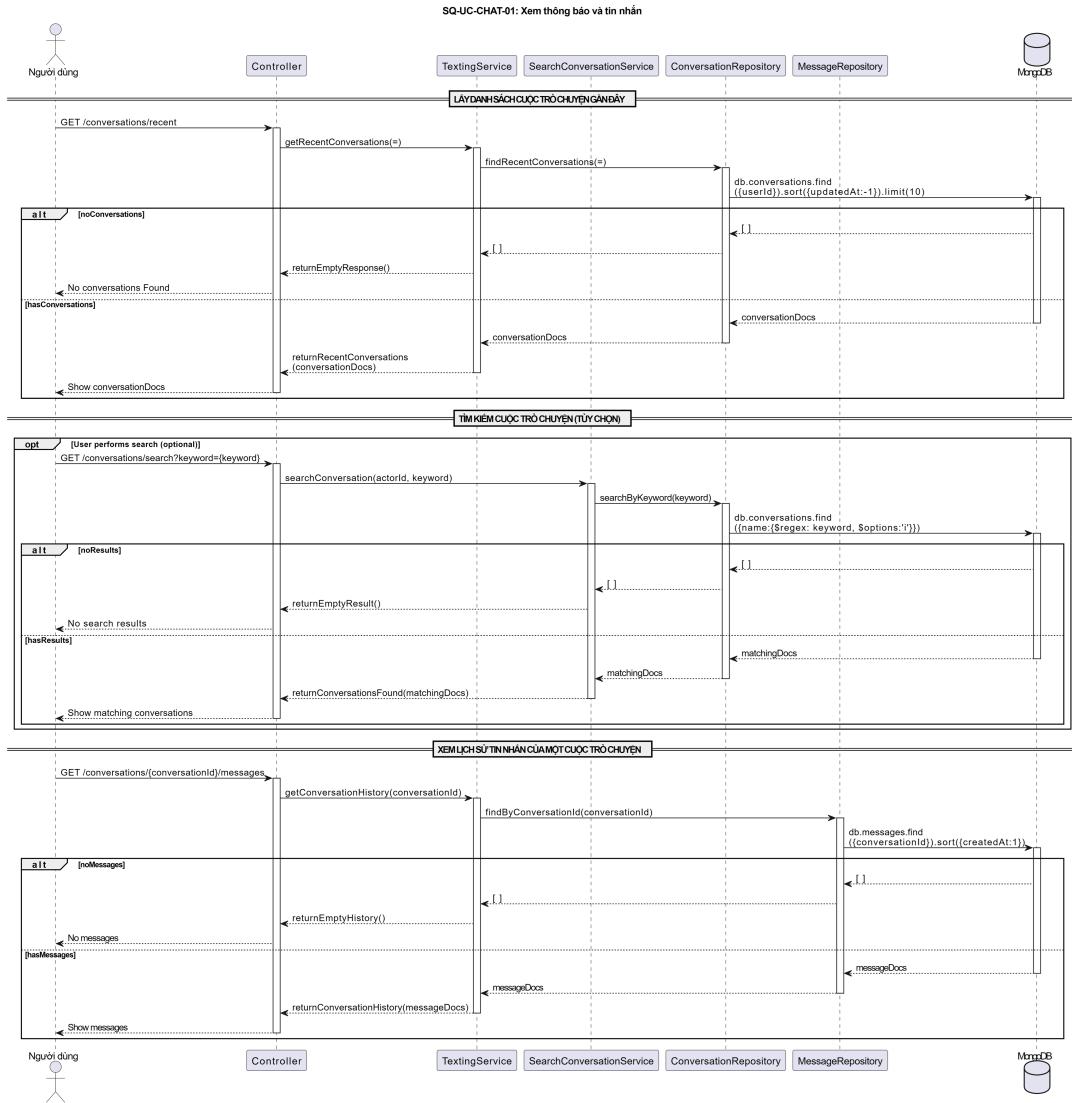


Hình 5.19: Sơ đồ tuần tự cho use-case "Thống kê số liệu và phân tích học tập"

Mô tả sơ đồ tuần tự: Thống kê số liệu và phân tích học tập

Controller gọi ReportService.generate_learning_stats(filters) để lấy thống kê học tập → sau đó ReportService gọi ReportRepositoryImpl.fetch_learning_data(filters) để truy xuất dữ liệu thô từ Database → đồng thời kích hoạt các luồng Hỗ trợ ra quyết định và Xuất báo cáo (nếu được chọn).

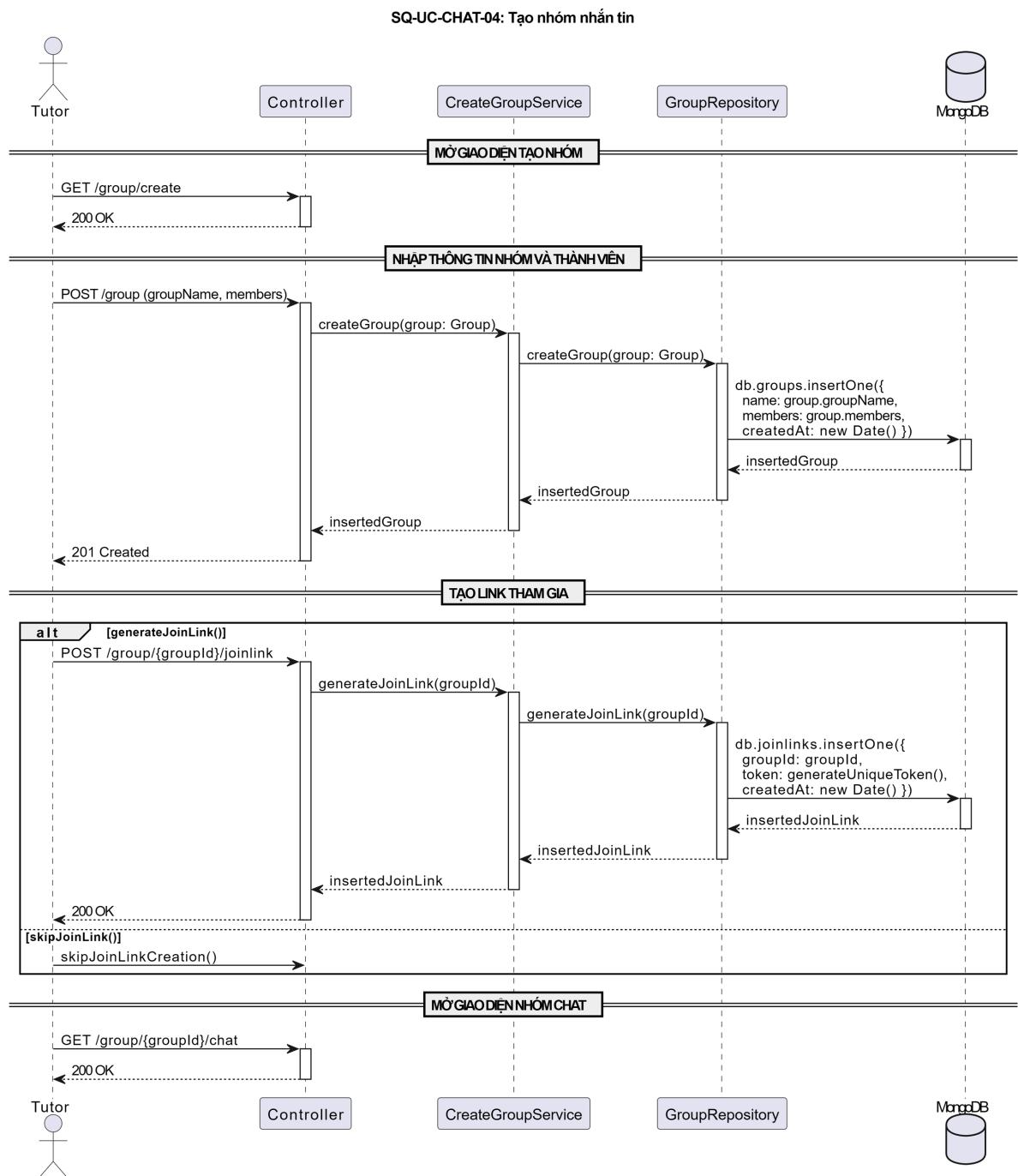
5.7 Thông báo và nhắn tin



Hình 5.20: Sơ đồ tuần tự cho use-case "Xem tin nhắn và thông báo"

Mô tả sơ đồ tuần tự: Xem thông báo và tin nhắn

- Người dùng lấy danh sách các cuộc trò chuyện gần đây qua GET /conversations/recent: Controller gọi TextingService, truy vấn ConversationRepository, trả về các hội thoại gần nhất.
- Người dùng thực hiện tìm kiếm hội thoại với từ khóa qua GET /conversations/search?keyword Controller gọi SearchConversationService, truy ConversationRepository với điều kiện keyword, trả về kết quả phù hợp (hoặc phản hồi rỗng).
- Người dùng xem lịch sử tin nhắn cuộc trò chuyện qua GET /conversations/{id}/messages: Controller gọi TextingService, repository truy vấn message theo conversationId, trả về lịch sử tin nhắn.

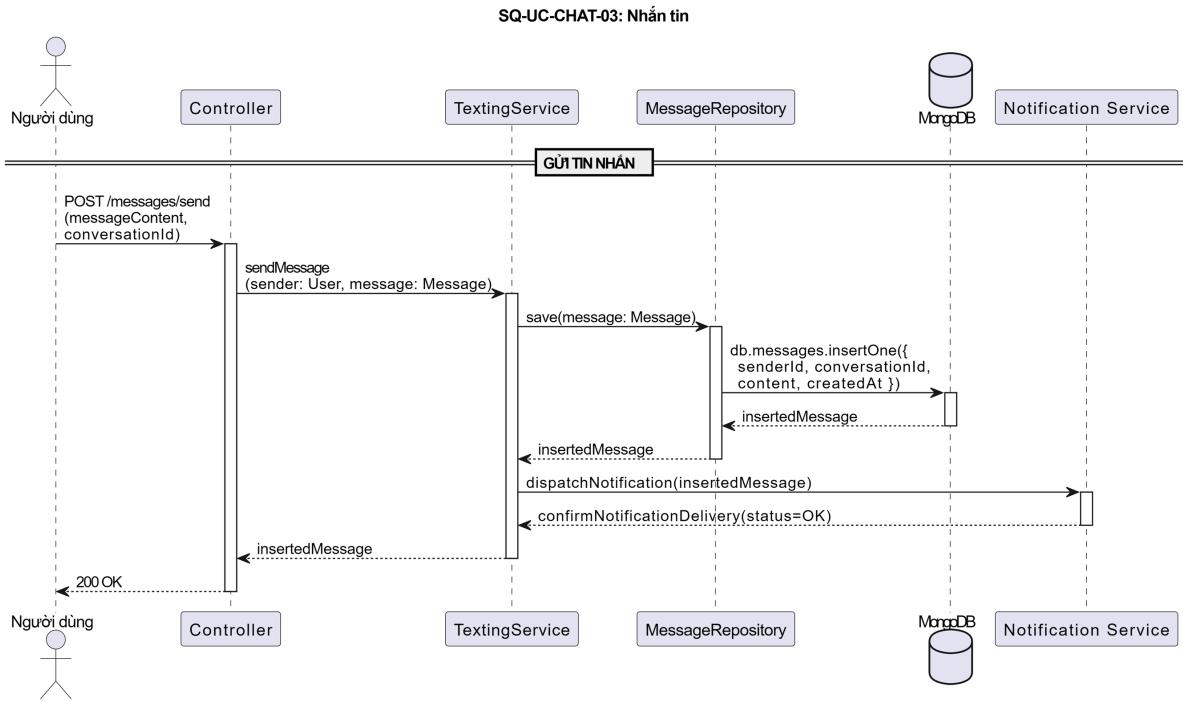


Hình 5.21: Sơ đồ tuần tự cho use-case "Tạo nhóm nhắn tin"



Mô tả sơ đồ tuần tự: Tạo nhóm nhắn tin

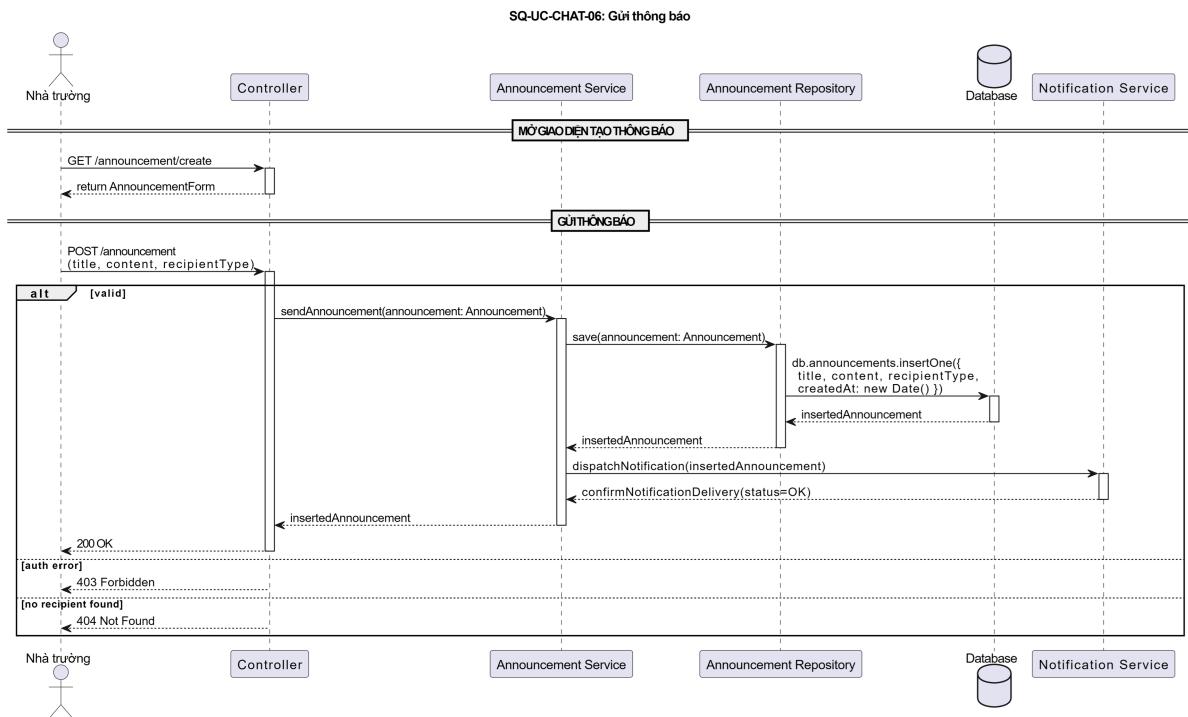
- Tutor gửi yêu cầu mở giao diện tạo nhóm (GET /group/create) và nhận phản hồi xác nhận.
- Tutor điền thông tin tên nhóm và danh sách thành viên rồi gửi lên (POST /group), Controller tiếp nhận chuyển cho CreateGroupService.
- Dịch vụ này gọi GroupRepository, lưu group mới vào DB (ghi nhận groupName, thành viên, ngày tạo).
- Kết quả lưu thành công được trả về (201 Created).
- Tutor có thể tạo liên kết tham gia bằng cách gửi (POST /group/{groupId}/joinlink), dịch vụ sẽ sinh token link, lưu vào DB.
- Nếu bỏ qua, hệ thống ghi nhận thao tác và không tạo link.
- Cuối cùng, tutor truy xuất giao diện chat nhóm vừa tạo.



Hình 5.22: Sơ đồ tuần tự cho use-case "Nhắn tin"

Mô tả sơ đồ tuần tự: Nhắn tin

- Người dùng gửi yêu cầu POST nhắn tin mới, gồm nội dung và conversationId.
- Controller tiếp nhận chuyển đến TextingService, dịch vụ này khởi tạo message object và lưu thông qua MessageRepository vào DB.
- Sau khi lưu thành công, dịch vụ tiếp tục gọi NotificationService để gửi thông báo tới các thành viên hội thoại vừa nhận tin nhắn.
- Luồng kết thúc khi hệ thống trả về phản hồi xác nhận gửi thành công cho người dùng.

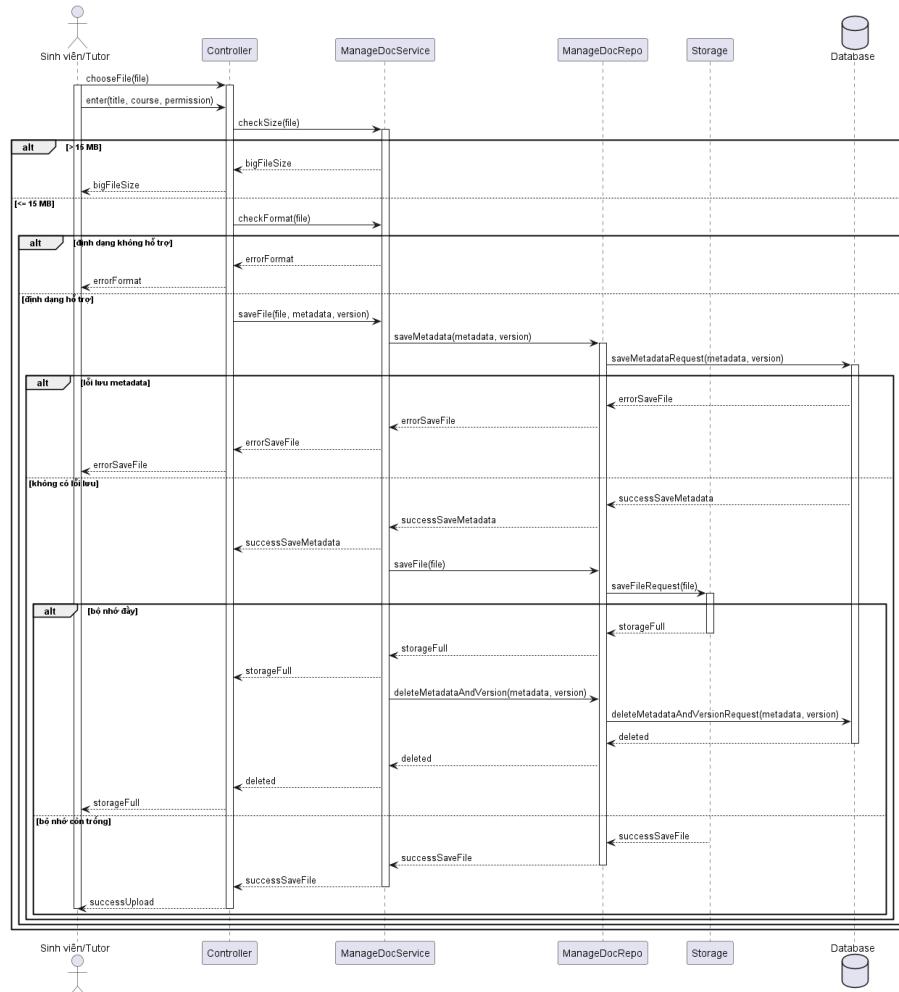


Hình 5.23: Sơ đồ tuần tự cho use-case "Gửi thông báo"

Mô tả sơ đồ tuần tự: Gửi thông báo

- Nhà trưởng mở giao diện tạo thông báo (GET /announcement/create), nhận lại form nhập liệu.
- Gửi thông báo mới (POST /announcement) với tiêu đề, nội dung, loại đối tượng nhận: Controller gửi tới AnnouncementService.
- AnnouncementService xử lý, lưu vào AnnouncementRepository, ghi nhận các thông tin về thông báo.
- Thông tin thông báo mới được chuyển tiếp đến NotificationService để phân phối đến danh sách người nhận thỏa mãn điều kiện (recipientType).
- Hệ thống trả về các trạng thái: gửi thành công (200 OK); lỗi xác thực (403 Forbidden); không có người nhận phù hợp (404 Not Found).

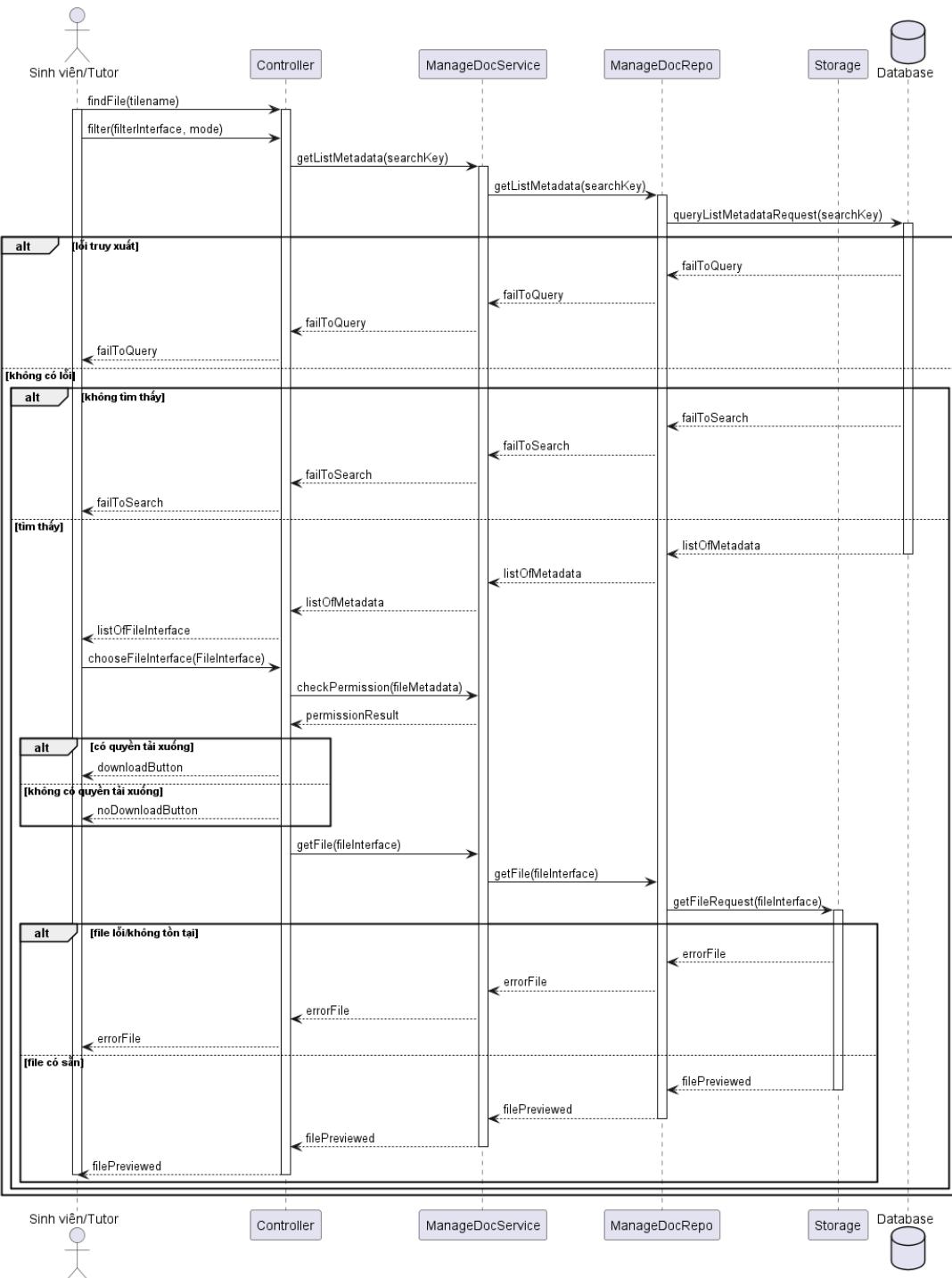
5.8 Quản lý tài liệu



Hình 5.24: Sơ đồ tuần tự cho use-case "Tải tài liệu lên hệ thống"

Mô tả sơ đồ tuần tự: Tải tài liệu lên hệ thống

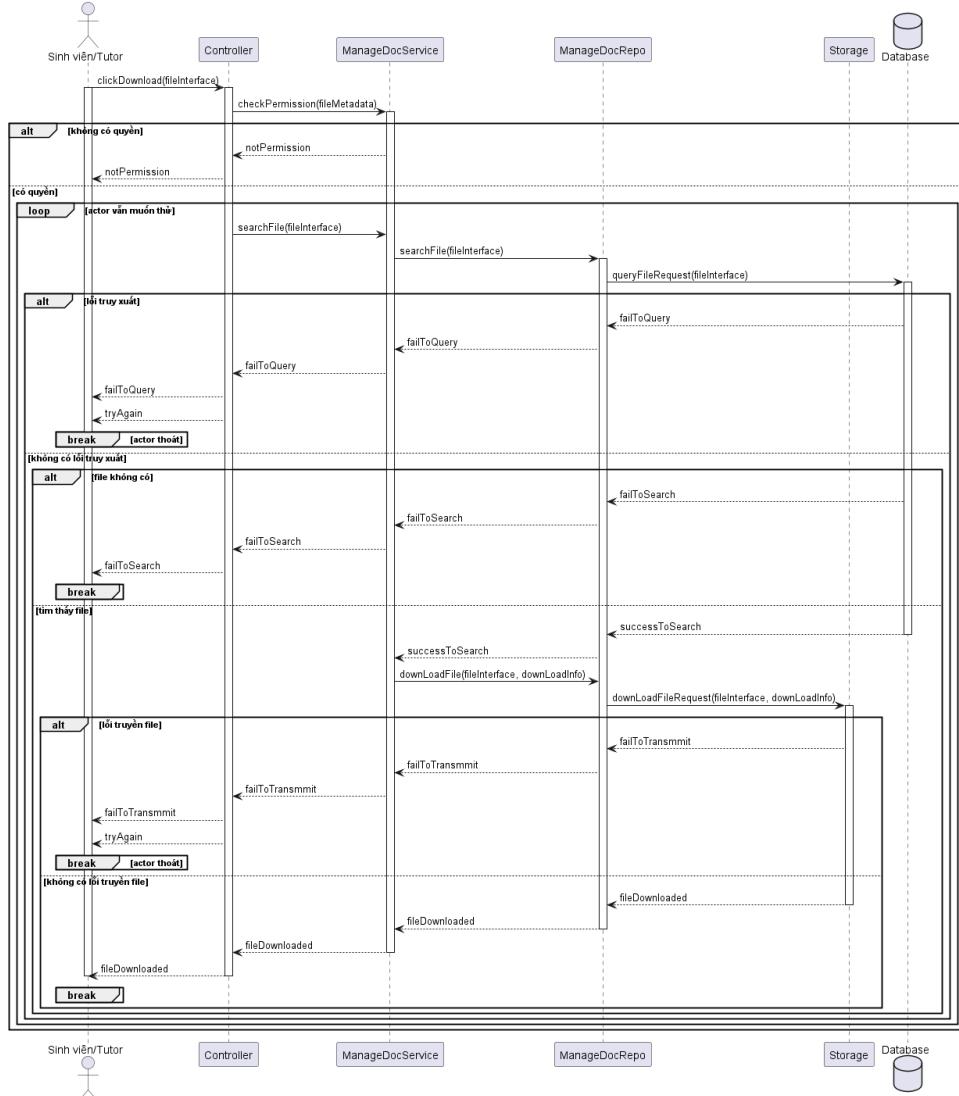
- Người dùng (Sinh viên/Tutor) chọn file và nhập thông tin tài liệu, gửi yêu cầu lên Controller.
- Controller kiểm tra kích thước file, nếu lớn hơn 15MB thì trả về lỗi, nếu hợp lệ thì tiếp tục kiểm tra định dạng file.
- Nếu định dạng không hỗ trợ, trả về lỗi cho người dùng. Nếu hợp lệ, Controller yêu cầu Service lưu file và metadata.
- Service lưu metadata qua Repo xuống Database. Nếu lưu metadata lỗi, trả về lỗi cho người dùng.
- Nếu lưu metadata thành công, Service tiếp tục lưu file vào Storage. Nếu bộ nhớ đầy, hệ thống xóa metadata vừa lưu và báo lỗi bộ nhớ đầy cho người dùng.
- Nếu lưu file thành công, trả về thông báo thành công cho người dùng.



Hình 5.25: Sơ đồ tuần tự cho use-case "Tìm kiếm và xem tài liệu"

Mô tả sơ đồ tuần tự: Tìm kiếm và xem tài liệu

- Người dùng gửi yêu cầu tìm kiếm tài liệu tới Controller, có thể kèm theo bộ lọc.
- Controller gọi Service để lấy danh sách metadata tài liệu phù hợp, Service truy vấn Repo và Database.
- Nếu truy vấn lỗi hoặc không tìm thấy tài liệu, trả về lỗi cho người dùng.
- Nếu tìm thấy, trả về danh sách tài liệu cho người dùng. Người dùng chọn tài liệu muốn xem.
- Controller kiểm tra quyền truy cập của người dùng với tài liệu đó.
- Nếu có quyền, hiển thị nút tải xuống; nếu không, ẩn nút tải xuống.
- Controller yêu cầu Service lấy file để xem trước, Service truy vấn Repo và Storage.
- Nếu file lỗi hoặc không tồn tại, trả về lỗi. Nếu thành công, trả về file xem trước cho người dùng.



Hình 5.26: Sơ đồ tuần tự cho use-case "Tải tài liệu xuống hệ thống"

Mô tả sơ đồ tuần tự: Tải tài liệu xuống hệ thống

- Người dùng nhấn tải xuống tài liệu, Controller kiểm tra quyền truy cập.
- Nếu không có quyền, trả về thông báo lỗi.
- Nếu có quyền, Controller và Service thực hiện truy vấn file qua Repo và Database.
- Nếu truy vấn lỗi hoặc không tìm thấy file, trả về lỗi cho người dùng.
- Nếu tìm thấy, Service yêu cầu Repo tải file từ Storage.
- Nếu quá trình truyền file lỗi, trả về lỗi và cho phép người dùng thử lại.
- Nếu tải file thành công, trả về file cho người dùng.

6 Giao diện người dùng

6.1 Giao diện hệ thống của Admin

The screenshot displays the Turto Admin Dashboard, which is a comprehensive system for managing student activities. The dashboard features a dark header bar with the Turto logo and a light-colored main area divided into several sections:

- Top Bar:** Shows system status (System Online, 324 users online, +18% this month), a search bar, and a navigation bar with "Hôm nay" and "Administrator".
- Welcome Section:** A green banner with "Wellcom Back!" and a message about system activity.
- Key Metrics:** Four cards showing total users (1.248), total lecturers (156), total students (1.092), and total events (3.456).
- Attendance Status:** A section titled "Trạng thái buổi học" showing completion rates for three categories: Hoàn thành (3.420), Đang diễn ra (24), and Chờ xác nhận (12).
- Recent Activity:** A section titled "Hoạt động gần đây" listing recent interactions with users like Nguyễn Văn A, Trần Thị B, Lê Văn C, and Phạm Thị D.
- Popular Lecturers:** A section titled "Gia sư xuất sắc" listing top lecturers: Nguyễn Thị Hương (1), Trần Văn Minh (2), and Lê Thị Lan (3), along with their ratings and the number of events.
- Utility Buttons:** Buttons for "Thêm người dùng" (Add User), "Xem báo cáo" (View Report), "AI Insights", and "Cài đặt" (Settings).
- Footer:** Includes copyright information (© 2025 Turto. All rights reserved.), links to Privacy Policy and Terms of Service, and a footer note (Last updated: 29/10/2025).

Hình 6.1: Giao diện tổng quan của Admin

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH TRƯỜNG ĐẠI HỌC BÁCH KHOA



Quản lý người dùng

Tìm kiếm... Vai trò Trạng thái

Người dùng	Email / Username	Vai trò	Trạng thái	Đăng nhập gần nhất	Thao tác
Nguyễn Văn Admin	admin@turo.com	Admin	Hoạt động	2025-10-29 14:30	Chỉnh sửa Xoá
Trần Thị Bình	tutor1@turo.com	Gia sư	Hoạt động	2025-10-29 10:15	Chỉnh sửa Xoá
Lê Văn Student	student1@turo.com	Học viên	Hoạt động	2025-10-28 16:45	Chỉnh sửa Xoá
Phạm Thị Dung	tutor2@turo.com	Gia sư	Không hoạt động	2025-10-20 09:00	Chỉnh sửa Xoá

Hiển thị 1-4 trong tổng số 1,248 người dùng

© 2025 Turo. All rights reserved. | [Privacy Policy](#) [Terms of Service](#) v1.0.0 Last updated: 29/10/2025

Hình 6.2: Giao diện quản lý người dùng

Phân tích học tập

Tuần Tháng Năm

Môn học	Điểm TB	Số lượng học viên
Mathematics	8.5/10	320 học viên
Physics	8.2/10	280 học viên
Chemistry	7.9/10	250 học viên
English	8.7/10	300 học viên
Computer Science	9.1/10	250 học viên

© 2025 Turo. All rights reserved. | [Privacy Policy](#) [Terms of Service](#) v1.0.0 Last updated: 29/10/2025

Hình 6.3: Giao diện phân tích học tập

The screenshot shows the Turto Admin Panel interface. On the left, there's a sidebar with various menu items like 'Tổng quan', 'Người dùng', 'Thống kê', 'Báo cáo & Phân tích', and 'Phản hồi'. The main content area is titled 'AI Insights & Recommendations' and includes a sub-section 'Trợ lý AI Turto'. It displays several cards with AI-generated insights:

- Dự đoán người dùng mới tháng sau: +15%
- Môn học tiềm năng nhất: AI/ML
- Thời điểm booking cao nhất: 18:00-20:00
- Tỷ lệ hài lòng dự kiến: 94%

Below this, there are sections for 'Khuyến nghị' (Recommendations) and 'Dự đoán xu hướng' (Predicted trends), each containing three cards with metrics like 'Tăng cường giá sư môn Toán' (+25%), 'Cải thiện chất lượng môn Hóa học' (+10%), and 'Tối ưu lịch học buổi tối' (+15%).

Hình 6.4: Giao diện AI hỗ trợ phân tích

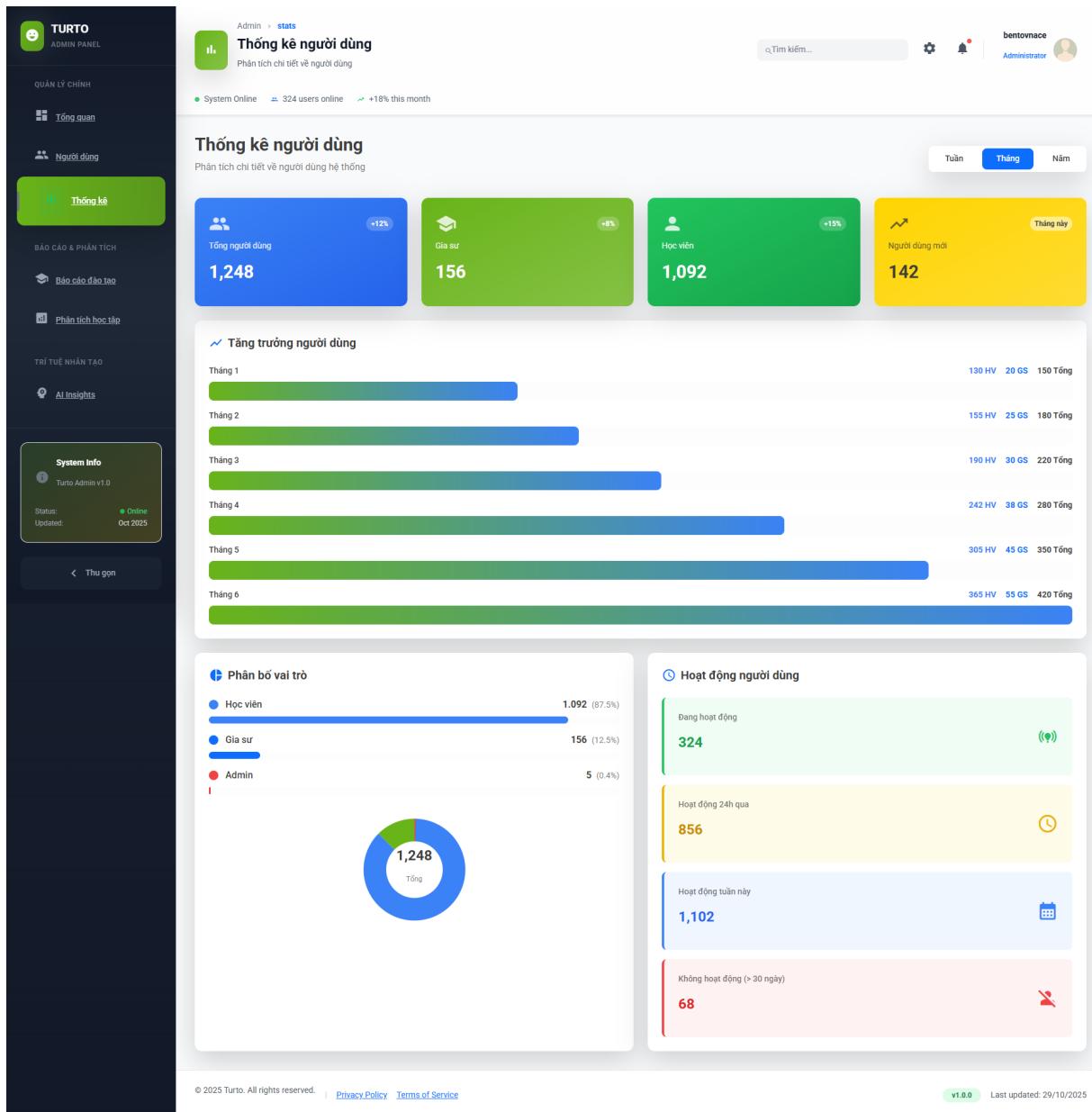
The screenshot shows the Turto Admin Panel interface. The sidebar includes 'Tổng quan', 'Người dùng', 'Thống kê', 'Báo cáo & Phân tích', and 'Phản hồi'. The main content area is titled 'Báo cáo đào tạo' and includes a 'Bộ lọc' (Filter) section with fields for 'Từ ngày' (01/01/2025), 'Đến ngày' (29/10/2025), 'Khoa' (Tất cả các khoa), and 'Ngành' (Tất cả các ngành). Below this, there are four large green cards with metrics:

- Tổng buổi học: 3.456
- Hoàn thành: 3.204
- Tổng giờ học: 6.912h
- Điểm TB: 8.5/10

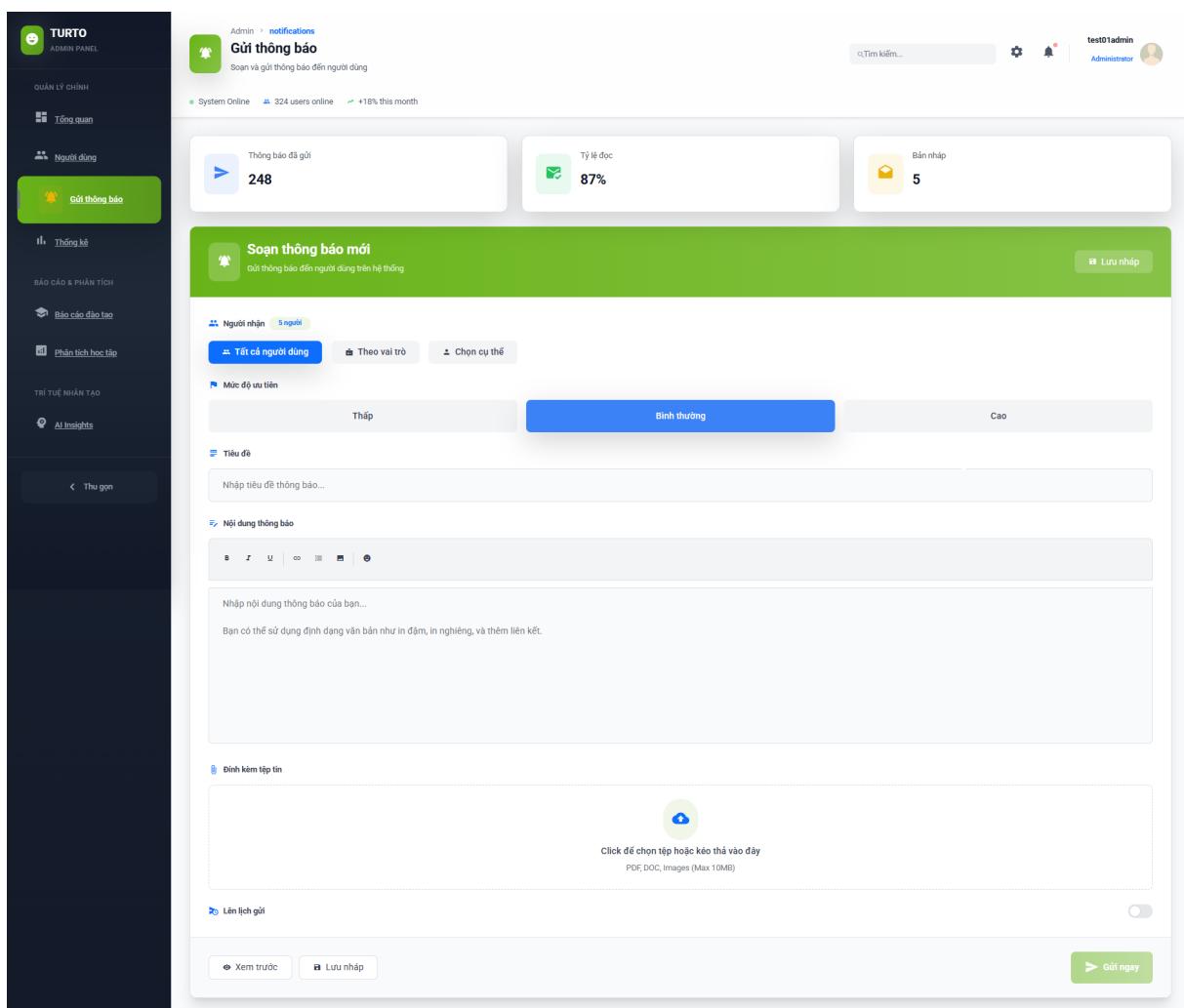
Underneath these cards is a section titled 'Thống kê theo khoa' (Statistics by department) showing data for four departments:

Khoa	Số buổi học	Số học viên	Tỷ lệ hoàn thành
Khoa Công nghệ thông tin	1.200	450	92%
Khoa Kinh tế	850	320	92%
Khoa Kỹ thuật	780	290	92%
Khoa Ngoại ngữ	626	232	92%

Hình 6.5: Giao diện báo cáo đào tạo



Hình 6.6: Giao diện thống kê



Hình 6.7: Giao diện gửi thông báo

6.2 Giao diện người dùng của Tutor

The screenshot displays the Turto Tutor Dashboard interface. On the left, a dark sidebar menu includes: TỔNG QUAN (Dashboard, Thông báo, Học viên của tôi, Lịch dạy, Thông kê, Tài khoản, Hồ sơ, Cài đặt), and a user profile for 'bentovnace' (Gia sư). The main dashboard area has a green header bar with 'Xin chào!' and a message about 3 upcoming classes. Below are several cards:

- Tổng học viên:** 24 (18 đang hoạt động)
- Tổng giờ dạy:** 234 (12 giờ tuần này)
- Đánh giá trung bình:** 4.8 ★★ (45 đánh giá)
- Buổi học sắp tới:** 8 (4 đang diễn ra)

Tiến độ buổi học:

- Hoàn thành: 144
- Đang diễn ra: 4
- Sắp tới: 8

Lịch học hôm nay:

- Nguyễn Văn A: Toán • 2 giờ (Hôm nay, 14:00) - Trực tuyến
- Trần Thị B: Vật lý • 1.5 giờ (Hôm nay, 16:30) - Trực tiếp
- Lê Văn C: Hóa học • 2 giờ (Ma, 09:00) - Trực tuyến

Học viên xuất sắc:

- Nguyễn Văn A: 24 buổi học, Tiến độ 85%
- Trần Thị B: 20 buổi học, Tiến độ 78%
- Lê Văn C: 18 buổi học, Tiến độ 92%

Đánh giá gần đây:

- Phạm Thị D: ★★★★☆, Giảng dạy rất nhiệt tình và dễ hiểu! 2 ngày trước
- Hoàng Văn E: ★★★★☆, Tốt, nhưng cần thêm ví dụ thực tế. 3 ngày trước
- Võ Thị F: ★★★★★, Excellent! Rất hài lòng. 5 ngày trước

Hình 6.8: Giao diện Dashboard của Tutor

Hình 6.9: Giao diện hồ sơ gia sư

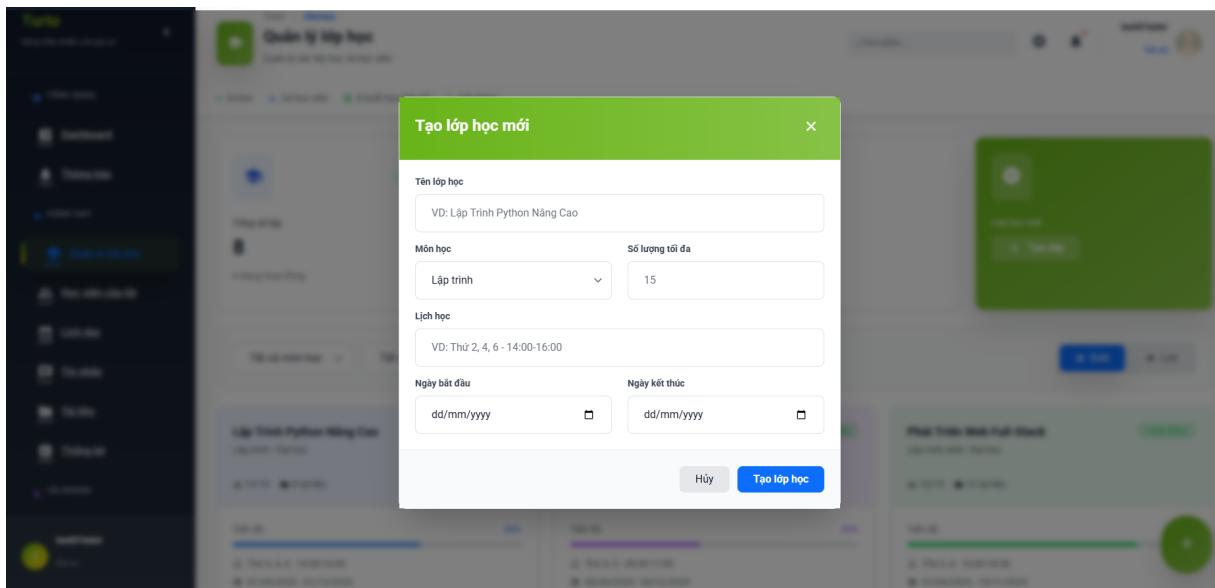
Hình 6.10: Giao diện tài liệu giảng dạy

Giờ	Th 2 27	Th 3 28	Th 4 29	Th 5 30	Th 6 31	Th 7 1	CN 2
08:00							
09:00				Hóa học Lê Văn C 09:00-11:00 Chờ xác nhận			
10:00						Tiếng Anh Hoàng Văn E 10:00-11:30	
11:00							
12:00							
13:00							
14:00			Toán Nguyễn Văn A 14:00-16:00	Toán Phạm Thị D 14:00-16:00			
15:00							
16:00							
17:00							
18:00							
19:00							
20:00							

Hình 6.11: Giao diện lịch dạy học

Hình 6.12: Giao diện quản lý lớp học

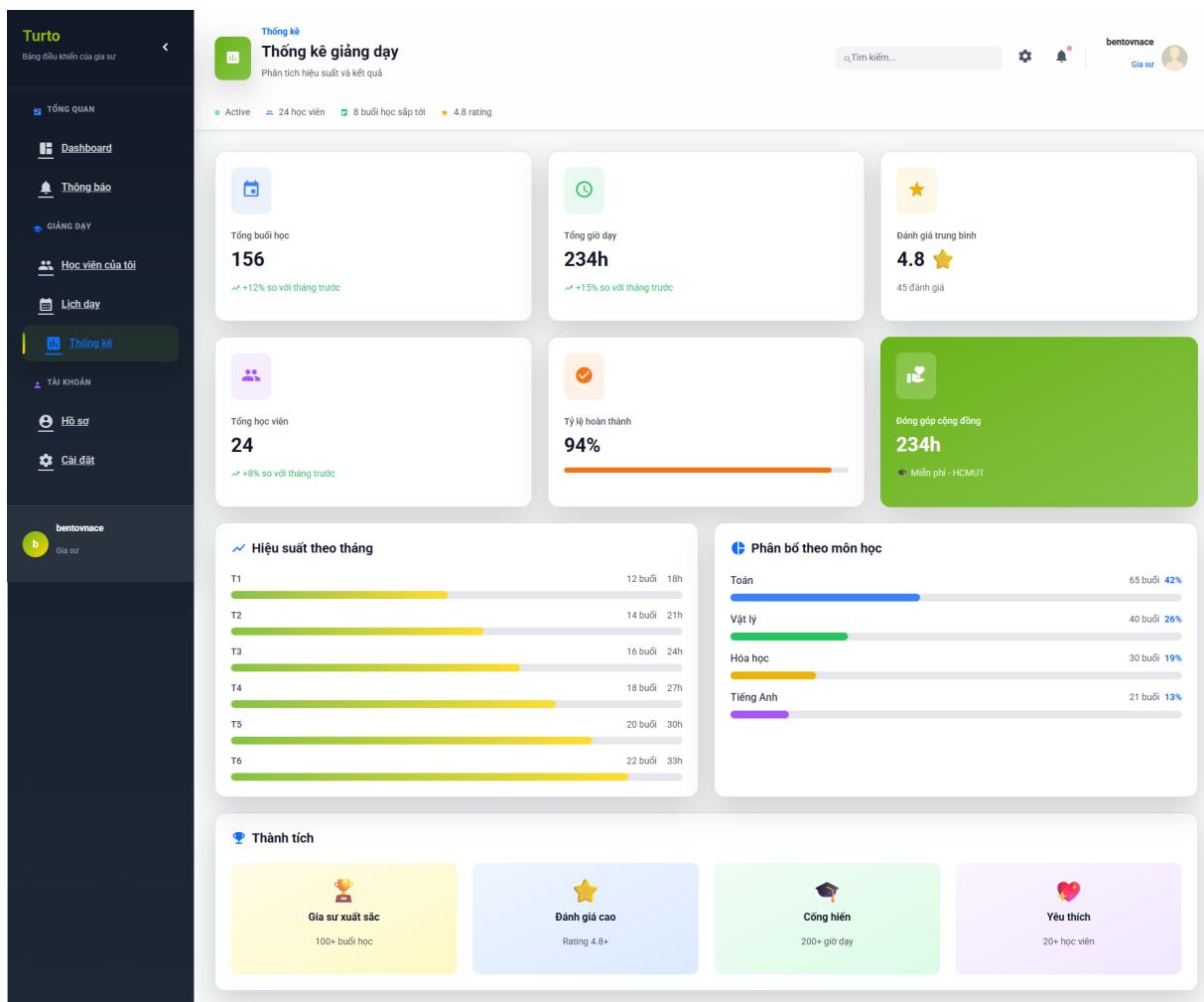
Hình 6.13: Giao diện xem lớp học



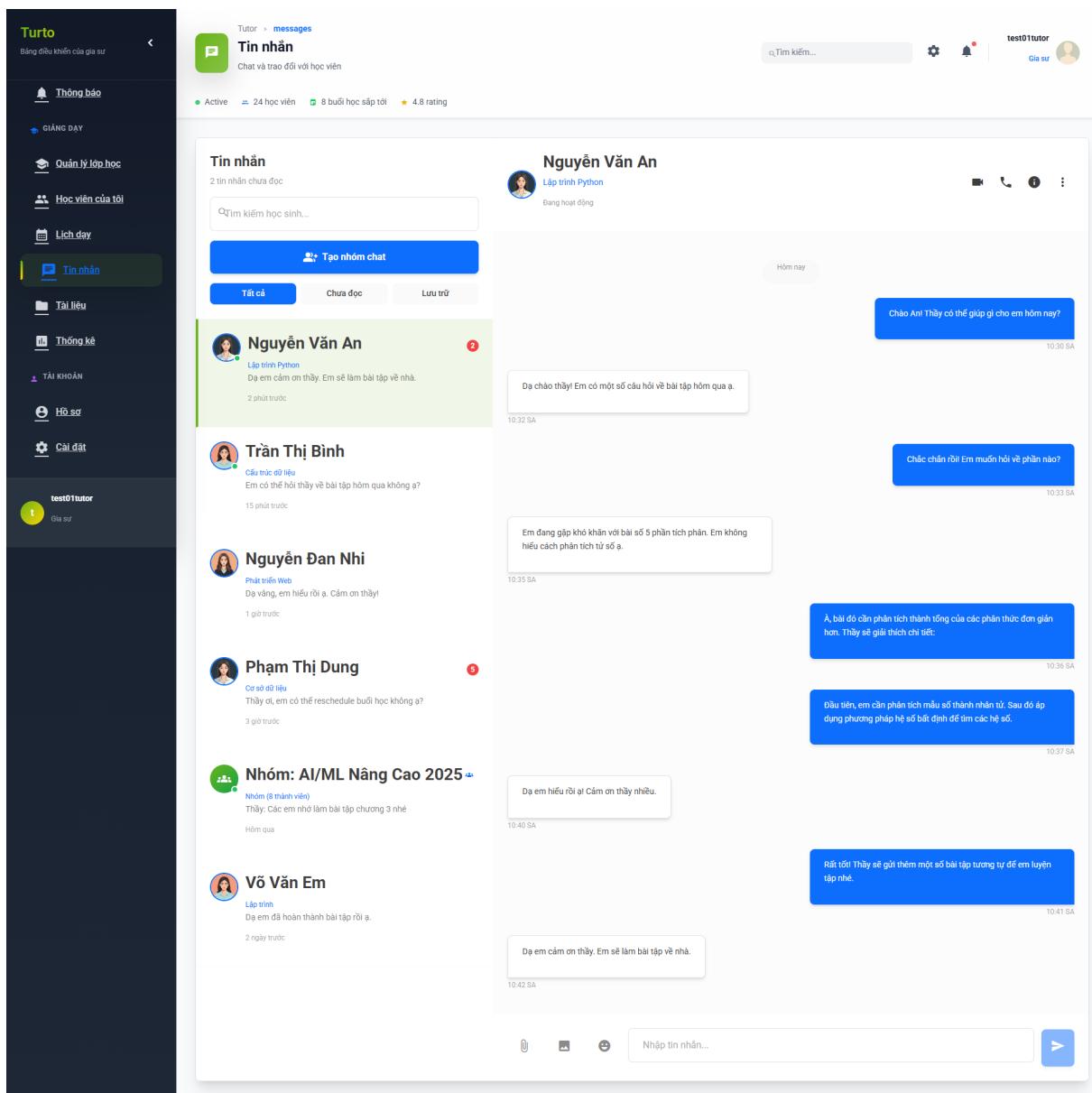
Hình 6.14: Giao diện tạo lớp học mới

HỌC VIÊN	MÔN HỌC	SỐ BUỔI HỌC	BUỔI HỌC CĂN NHẤT	TIẾN ĐỘ	TRẠNG THÁI	HÀNH ĐỘNG
Nguyễn Văn A ID: 1	Toán	24	2 ngày trước	<div style="width: 85%;">85%</div>	Hoạt động
Trần Thị B ID: 2	Vật lý	20	1 ngày trước	<div style="width: 78%;">78%</div>	Hoạt động
Lê Văn C ID: 3	Hóa học	18	3 ngày trước	<div style="width: 92%;">92%</div>	Hoạt động
Phạm Thị D ID: 4	Toán	15	1 tuần trước	<div style="width: 65%;">65%</div>	Không hoạt động
Hoàng Văn E ID: 5	Tiếng Anh	22	1 ngày trước	<div style="width: 88%;">88%</div>	Hoạt động

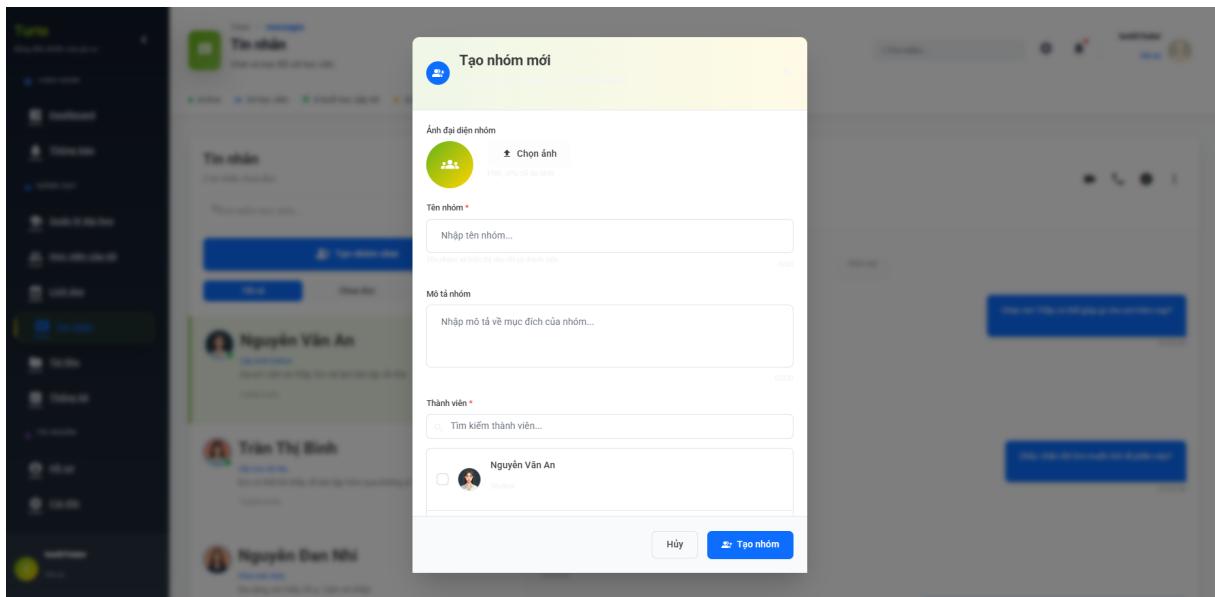
Hình 6.15: Giao diện quản lý học viên



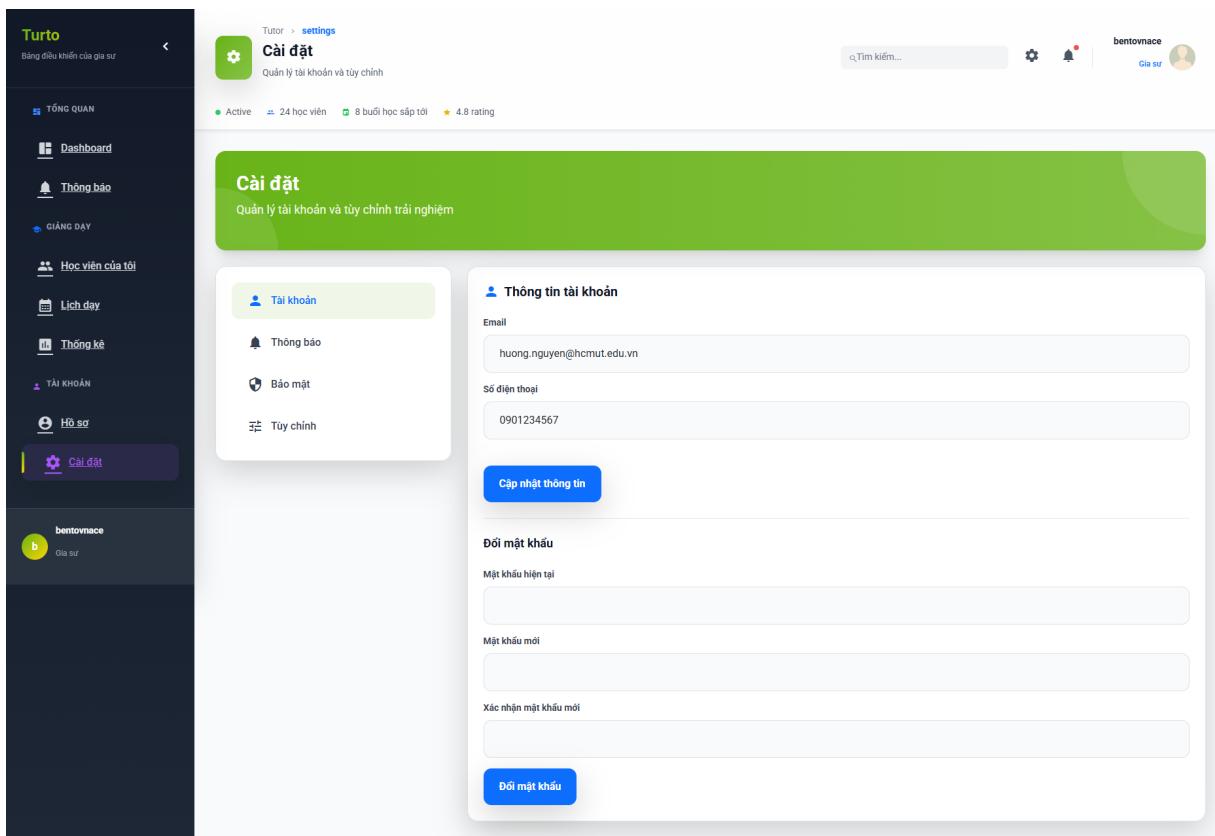
Hình 6.16: Giao diện thống kê



Hình 6.17: Giao diện trò chuyện



Hình 6.18: Giao diện tạo nhóm chat



Hình 6.19: Giao diện cài đặt

The screenshot displays the Turto application's notification center. At the top, there is a header bar with the title "Thông báo" and a subtitle "Cập nhật mới nhất về buổi học và hoạt động". Below the header, there are four main notification categories:

- Chưa đọc:** 3 mới
- Tổng thông báo:** 6
- Học viên mới:** 1
- Danh giá mới:** 1

Below these categories, there is a filter bar with options: Tất cả, Chưa đọc (3), Đã đọc, Học viên mới, Danh giá, Nhắc nhở. Under each category, there is a list of notifications with details such as the user's name, subject, and timestamp.

- Học viên mới đăng ký:** Nguyễn Văn A đã đăng ký học với bạn môn Toán (5 phút trước). Includes links Xem chi tiết and Danh dấu đã đọc.
- Nhắc nhở buổi học:** Buổi học với Trần Thị B sẽ bắt đầu trong 30 phút (15 phút trước). Includes links Xem chi tiết and Danh dấu đã đọc.
- Đánh giá mới:** Lê Văn C đã đánh giá 5★ cho buổi học vừa rồi (1 giờ trước). Includes links Xem chi tiết and Danh dấu đã đọc.
- Buổi học hoàn thành:** Bạn đã hoàn thành buổi học với Phạm Thị D (2 giờ trước). Includes link Xem chi tiết.
- Thay đổi lịch học:** Hoàng Văn E đã yêu cầu đổi lịch buổi học ngày mai (3 giờ trước). Includes link Xem chi tiết.
- Cập nhật hệ thống:** Turto vừa ra mắt tính năng tạo lịch dạy mới (1 ngày trước). Includes link Xem chi tiết.

Hình 6.20: Giao diện thông báo

6.3 Giao diện người dùng của Sinh viên

Xin chào, bentovnace!
Đây là bảng điều khiển của bạn hôm nay.

Tổng buổi học: 24 Giờ học: 48h Sắp tới: 5 Tin nhắn mới: 2

Thao tác nhanh:

- Tim gia sư: Duyệt gia sư có sẵn
- Xem hồ sơ: Quản lý thông tin cá nhân
- Xem lịch học: Quản lý lớp học của bạn

Lớp học sắp tới:

Môn học	Giảng viên	Thời gian	Địa điểm	Thao tác
Giải tích 2	Với GS. Nguyễn Văn A	T2, 20/05/2025 (7:30 - 9:30 Sáng)	Phòng A301, Tòa Khoa học	Trực tuyến Thêm vào lịch
Vật lý 1	Với GS. Trần Thị B	T3, 21/05/2025 (1:00 - 3:00 Chiều)	Phòng A301, Tòa Khoa học	Trực tiếp Xem chi tiết

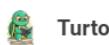
Gia sư đề xuất:

Nguyễn Thị Hương Chuyên gia Toán ★ 4.9 (120) 250+ học viên Xem hồ sơ Chat	Trần Thị Bình Chuyên gia Vật lý ★ 4.8 (98) 180+ học viên Xem hồ sơ Chat	Lê Thị Lan Chuyên gia Hóa học ★ 5.0 (75) 150+ học viên Xem hồ sơ Chat	Phạm Thị Dung Giáo viên Tiếng Anh ★ 4.7 (150) 300+ học viên Xem hồ sơ Chat
--	--	--	---

Tin nhắn gần đây:

- Nguyễn Thị Hương**
Đừng quên ôn lại Chương 3 trước buổi học tối nhé...
2 phút trước
- Trần Thị Bình**
Bài tập vật lý của bạn tiến bộ tốt lắm! Cố gắng lên.
1 giờ trước

Hình 6.21: Giao diện trang chủ của Sinh viên



Tìm gia sư hoàn hảo cho bạn

Kết nối với các gia sư chuyên nghiệp để đạt được mục tiêu học tập

Bộ lọc

Tìm kiếm

Môn học

- Tất cả
- Toán
- Vật lý
- Hóa học
- Tiếng Anh
- Tin học
- Sinh học

Đánh giá tối thiểu

- ★ 4.5+
- ★ 4+
- ★ 3.5+

Xóa tất cả bộ lọc

Hiển thị 6 gia sư

Sắp xếp: Đánh giá cao nhất

Nguyễn Thị Hương
Mathematics

★ 4.9 (120)

Chuyên gia Giải tích và Toán cao cấp với hơn 5 năm kinh nghiệm.

✉ Miễn phí - Cộng đồng HCMUT 150

Xem hồ sơ

Trần Thị Bình
Physics

★ 4.8 (98)

Chuyên về Điện tử học và Vật lý lượng tử.

✉ Miễn phí - Cộng đồng HCMUT 120

Xem hồ sơ

Lê Thị Lan
Chemistry

★ 5 (75)

Chuyên gia Hóa hữu cơ với nền tảng nghiên cứu.

✉ Miễn phí - Cộng đồng HCMUT 95

Xem hồ sơ

Phạm Thị Dung
English

★ 4.7 (150)

Chuyên luyện thi IELTS và TOEFL.

✉ Miễn phí - Cộng đồng HCMUT 200

Xem hồ sơ

Võ Thị Mai
Computer Science

★ 4.9 (110)

Chuyên gia Cấu trúc dữ liệu, Thuật toán và Học máy.

✉ Miễn phí - Cộng đồng HCMUT 130

Xem hồ sơ

Hoàng Thị Hương
Biology

★ 4.6 (85)

Chuyên gia Sinh học phân tử và Di truyền học.

✉ Miễn phí - Cộng đồng HCMUT 100

Xem hồ sơ

< 1 2 3 >

Hình 6.22: Giao diện tìm gia sư

140



Turto
Trang chủ
Gia sư
Lịch học

[Trang chủ](#) / [Gia sư](#) / [Nguyễn Thị Hương](#)

Nguyễn Thị Hương

Chuyên gia Khoa học Máy tính

Cựu sinh viên HOMUT

★ 4.9 (74 đánh giá)

120 học viên

Miễn phí - Cộng đồng HCMUT

Đặt buổi học
Gửi tin nhắn

Giới thiệu

Với hơn 5 năm kinh nghiệm trong giảng dạy, tôi chuyên về phương pháp học tập cá nhân hóa. Mục tiêu của tôi là làm cho các chủ đề phức tạp trở nên dễ hiểu và dễ tiếp cận. Tôi có bằng Thạc sĩ Khoa học Máy tính từ HCMUT và có thành tích đã được chứng minh trong việc nâng cao điểm số và sự tự tin của học viên.

Bằng cấp

- Thạc sĩ Khoa học Máy tính
- Cử nhân Kỹ thuật Phần mềm HCMUT
- Chứng chỉ Java Professional

Chuyên môn

Cấu trúc dữ liệu
Thuật toán
Java
Học máy
Python

Đánh giá từ học viên

Nguyễn Thị Mai
15/09/2025

★★★★★

Cô Hương là một gia sư tuyệt vời! Cô giải thích các khái niệm phức tạp một cách dễ hiểu. Điểm số của em đã cải thiện đáng kể kể từ khi bắt đầu học với cô.

Trần Văn Bình
10/09/2025

★★★★★

Rất kiên nhẫn và am hiểu. Rất khuyên dùng cho bất kỳ ai đang gặp khó khăn với thuật toán!

Lê Thị Cúc
28/08/2025

★★★★★

Gia sư xuất sắc. Cô đã giúp em chuẩn bị cho kỳ thi cuối kỳ và em đã đạt điểm cao!

Lịch trống

Tháng 10, 2025

CN	T2	T3	T4	T5	T6	T7
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Khung giờ trống

7:30 AM - 9:30 AM

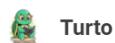
2:00 PM - 4:00 PM

Đặt buổi học ngay

Cần trợ giúp? [Liên hệ hỗ trợ](#)

Hình 6.23: Giao diện xem hồ sơ gia sư

141



Đặt buổi học

Lên lịch buổi học của bạn với Nguyễn Thị Hương

Thông tin gia sư

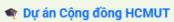


Nguyễn Thị Hương
Mathematics
★ 4.9

 Miễn phí
Công đồng HCMUT

Tóm tắt đặt lịch

Ngày	2025-10-16
Giờ	Chưa chọn
Thời lượng	1 giờ
Hình thức	Trực Tuyến
Chi phí	 Miễn phí

 Dự án Cộng đồng HCMUT
Tất cả buổi học đều miễn phí cho sinh viên

Loại buổi học



Học trực tuyến
Qua video call



Học trực tiếp
Tại địa điểm thỏa thuận

Xác nhận đặt lịch

Khi xác nhận, bạn đồng ý với [Điều khoản & Điều kiện](#) của chúng tôi

Chọn ngày & giờ

Tháng 10, 2025

CN	T2	T3	T4	T5	T6	T7
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Khung giờ trống

7:30 - 9:00	9:30 - 11:00
14:00 - 15:30	16:00 - 17:30

Thời lượng buổi học

1 giờ

Ghi chú bổ sung (Tùy chọn)

Chú đề hoặc câu hỏi cụ thể bạn muốn thảo luận?

Hình 6.24: Giao diện đặt buổi học



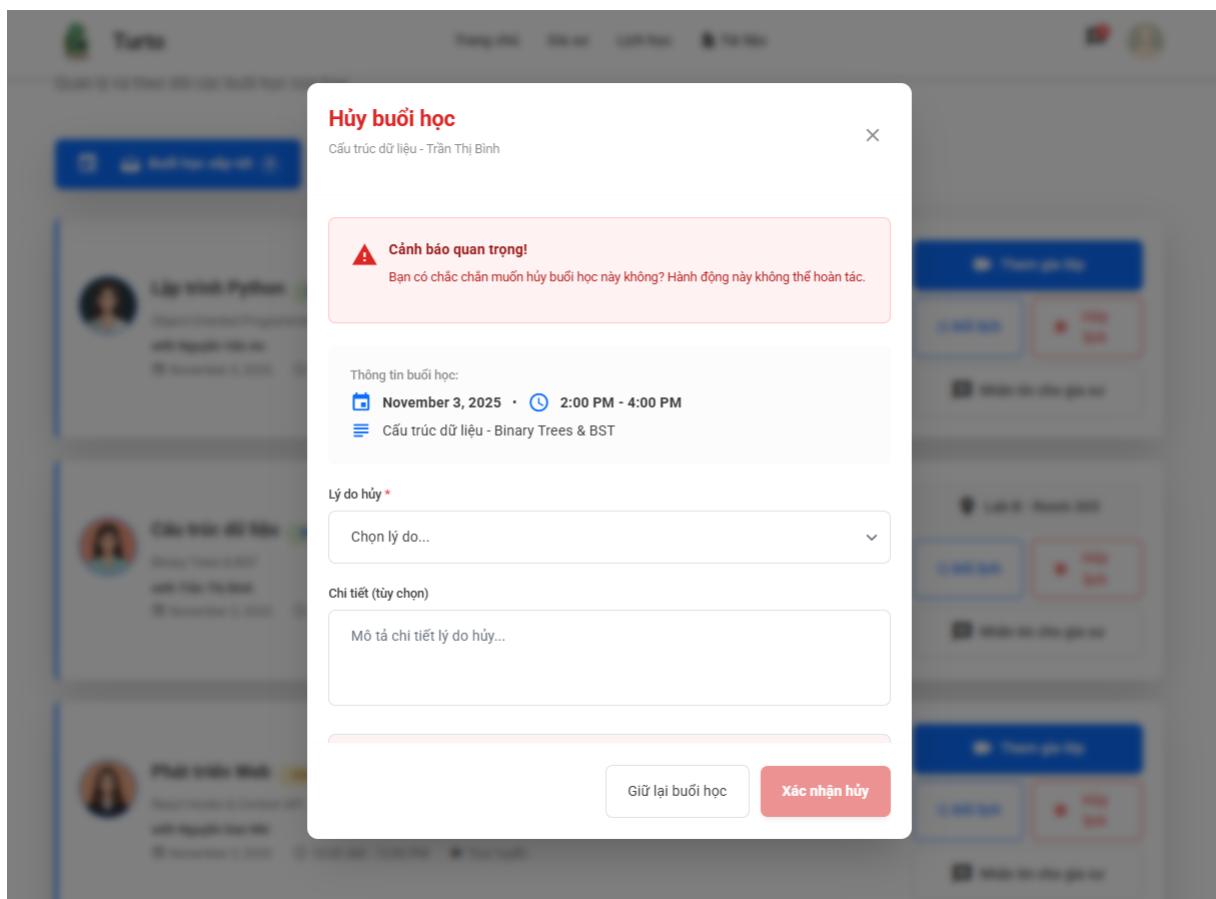
Lịch học của tôi

Quản lý và theo dõi các buổi học của bạn

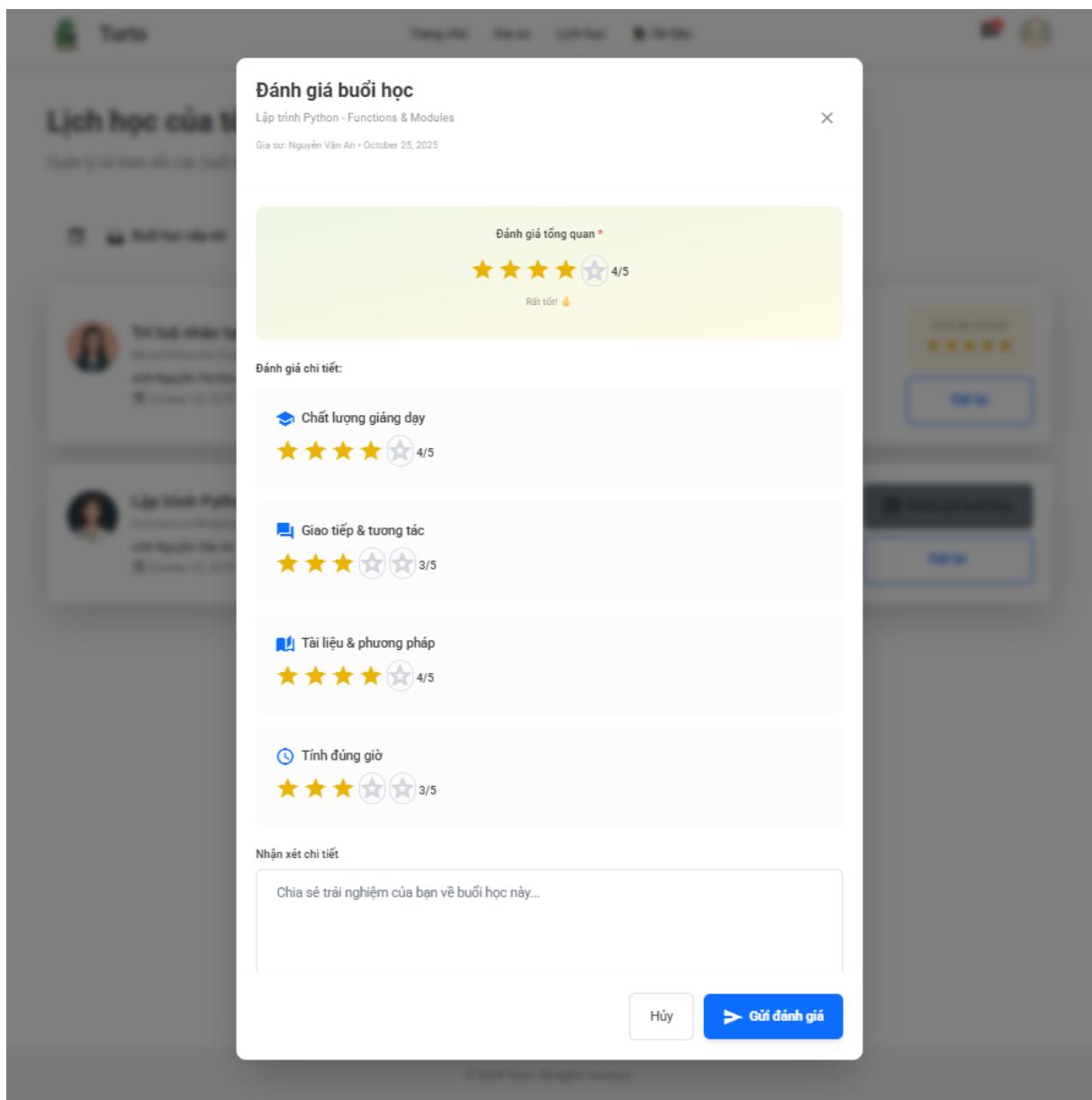
Số lượng	Mô tả	Thời gian	Đánh giá
4	Buổi học sắp tới		
2	Buổi học đã qua		

Hình 6.25: Giao diện lịch học

Hình 6.26: Giao diện đổi lịch học



Hình 6.27: Giao diện hủy lịch học



Hình 6.28: Giao diện đánh giá buổi học

Tài liệu học tập

Xem tài liệu từng lớp học mà bạn đã đăng ký

Lập trình cơ bản

Nguyễn Văn An

12 tài liệu

Cấu trúc dữ liệu & Giải thuật

Trần Thị Bình

15 tài liệu

Phát triển Web Full-Stack

Nguyễn Đan Nhi

18 tài liệu

Cơ sở dữ liệu nâng cao

Phạm Thị Dung

10 tài liệu

Trí tuệ nhân tạo I

Nguyễn Thị Kim Anh

8 tài liệu

Lập trình cơ bản

Giảng viên: Nguyễn Văn An

Đăng ký từ: 1/9/2025

4

Tài liệu

Bộ lọc

Loại file

Tất cả định dạng

PDF

Word

PowerPoint

Video

ZIP

Reset

🔍

Tài liệu gần đây

Bài 1 - Giới thiệu lập trình Python.pdf

Nguyễn Văn An

2.4 MB 2025-10-28

Xem Tải

Code examples - Variables and Data Types.zip

Nguyễn Văn An

1.2 MB 2025-10-27

Xem Tải

Bài tập tuần 1 - Cú pháp cơ bản.docx

Nguyễn Văn An

850 KB 2025-10-26

Xem Tải

Tất cả tài liệu

Bài 1 - Giới thiệu lập trình Python.pdf

Nguyễn Văn An

2.4 MB 2025-10-28

Xem Tải

Code examples - Variables and Data Types.zip

Nguyễn Văn An

1.2 MB 2025-10-27

Xem Tải

Bài tập tuần 1 - Cú pháp cơ bản.docx

Nguyễn Văn An

850 KB 2025-10-26

Xem Tải

Video hướng dẫn - Setup môi trường.mp4

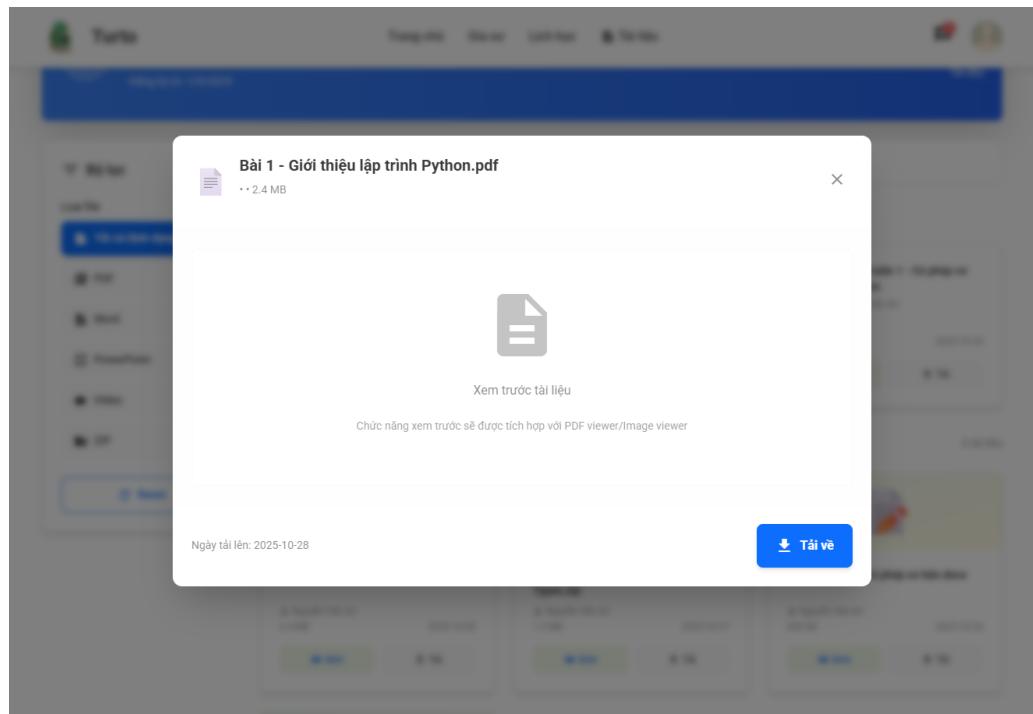
Nguyễn Văn An

45.3 MB 2025-10-25

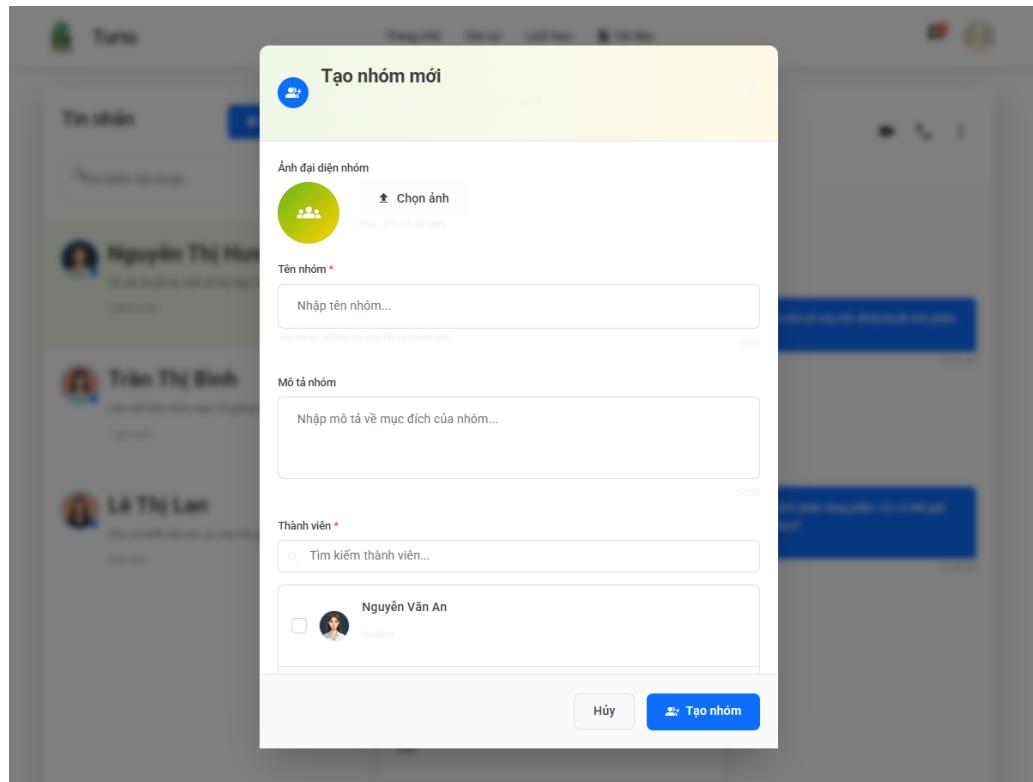
Xem Tải

Hình 6.29: Giao diện tài liệu

146



Hình 6.30: Giao diện xem tài liệu



Hình 6.31: Giao diện tạo nhóm trò chuyện



Tin nhắn • Tạo nhóm

Nguyễn Thị Hương Trực tuyến

Chào em! Cô có thể giúp gì cho em hôm nay?

10:30 SA

Nguyễn Thị Hương 2

Cô sẽ chuẩn bị một số bài tập thực hành cho b...

2 phút trước

Trần Thị Bình

Làm tốt lắm hôm nay! Cô gắng luyện tập nhé.

1 giờ trước

Lê Thị Lan

Cho cô biết nếu em có câu hỏi gì về bài tập v...

Hôm qua

Chắc chắn rồi! Em muốn biết cụ thể vấn đề gì?

10:33 SA

Em đang gặp khó khăn với tích phân tùng phần. Cô có thể giải thích lại khái niệm này không ạ?

10:35 SA

Được! Tích phân tùng phần dựa trên quy tắc tích cho đạo hàm. Công thức là $\int u \, dv = uv - \int v \, du$.

10:36 SA

Cô sẽ chuẩn bị một số bài tập thực hành cho buổi học tiếp theo nhé.

10:38 SA

Nhập tin nhắn... ▶

Hình 6.32: Giao diện trò chuyện



Turto

Trang chủ Gia sư Lịch học





Nguyen Van A
Computer Science • 3rd Year

Tổng buổi học **24**

Giờ học **42**

Tham gia **January 2025**

Đánh giá **★ 4.8**

[Chỉnh sửa hồ sơ](#)

Thông tin cá nhân

Email: nguyen.van.a@gmail.com

Điện thoại: +84 123 456 789

Trường đại học: HCMUT

Thành tích



First Session
Completed your first learning session



10 Sessions Streak
Attended 10 consecutive sessions



Top Reviewer
Provided 5+ helpful reviews



Excellence Award
Maintained 4.5+ average rating

Khóa học đã đăng ký

Calculus 2
with Nguyễn Văn An

9/12 buổi học hoàn thành

Tiếp theo: Mon, Oct 16 - 7:30 AM

75% Complete

Physics 1
with Trần Thị Bình

6/10 buổi học hoàn thành

Tiếp theo: Tue, Oct 17 - 1:00 PM

60% Complete

Chemistry
with Lê Hoàng Cường

3/8 buổi học hoàn thành

Tiếp theo: Wed, Oct 18 - 4:00 PM

40% Complete

Thống kê học tập

Tuần này	4 buổi học
Tháng này	12 buổi học
Đánh giá TB	★ 4.8
Tỷ lệ hoàn thành	92%

Hình 6.33: Giao diện hồ sơ học sinh



6.4 Liên kết kiểm thử giao diện người dùng

Trang giao diện người dùng - **Turto Program**: [Tại đây](#). Đăng xuất tài khoản mặc định sau đó đăng nhập vào một trong các tài khoản dưới đây:

Danh sách các tài khoản kiểm thử:

- Admin: admin@turto.com - Password: Admin@123
- Tutor: tutor@turto.com - Password: Admin@123
- Sinh viên: student@turto.com - Password: Admin@123

7 Class Diagram

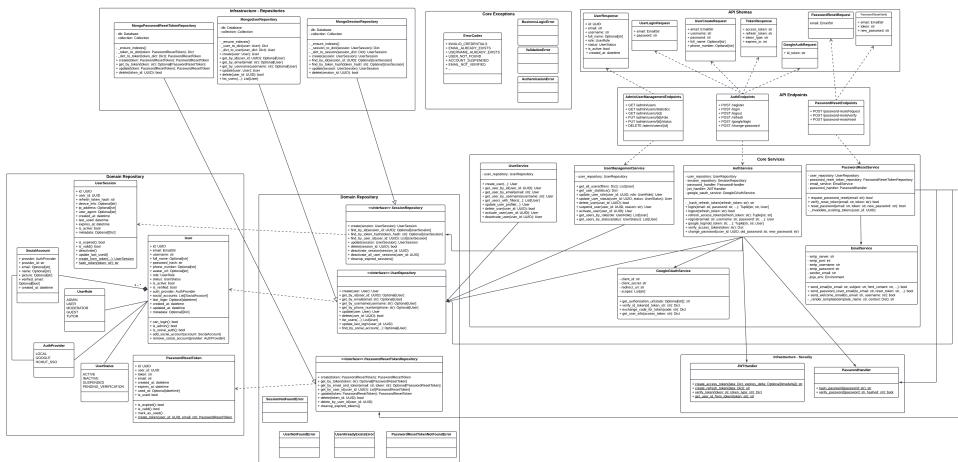
Tổng quan cấu trúc backend để vẽ class diagram

Các thành phần chính

- **API Layer:** Định nghĩa các điểm truy cập (endpoint), nhận và xử lý request từ client, trả dữ liệu ra dạng schema phù hợp. Phụ trách xác thực đầu vào và trích xuất thông tin để chuyển xuống các lớp bên dưới.
- **Core Service Layer:** Chứa logic nghiệp vụ chính (business logic). Sử dụng các service để xử lý các trường hợp thực tế (đăng ký, đăng nhập, đổi mật khẩu, quản lý thông tin người dùng, ...), đồng thời phối hợp với các lớp hạ tầng hoặc gọi các dịch vụ ngoài.
- **Domain Model Layer:** Khai báo các đối tượng cốt lõi (User, Session, Token, SocialAccount), các kiểu dữ liệu enum cho trạng thái/quyền và chứa các phương thức kiểm tra hoặc thao tác lên dữ liệu domain.
- **Domain Repository Interface Layer:** Định nghĩa các interface (giao diện) cho việc đọc/ghi và truy xuất dữ liệu mà các service sử dụng. Tách biệt rõ phần abstract (giao diện) và phần concrete (thực thi).
- **Infrastructure Layer:** Chứa thực thi thực tế các repository (ví dụ: MongoDB, SQL, Redis, ...), các adapter cho dịch vụ ngoài như Email, Google Login hoặc thực hiện việc hash password/JWT.
- **External Services Layer:** Các adapter cho dịch vụ bên thứ ba hoặc nằm ngoài hệ thống (email, oauth, xác thực bên ngoài).

Luồng xử lý tổng quát

- Request từ client gửi lên **API Endpoint**, đầu vào được xác thực thông qua **Schema**.
- **API Endpoint** chuyển dữ liệu đến **Service** thích hợp.
- **Service** thực hiện xử lý logic nghiệp vụ, sử dụng các hàm trong domain model để kiểm tra dữ liệu hoặc trạng thái.
- **Service** gọi đến **Repository Interface** để truy xuất hoặc thay đổi dữ liệu.
- **Repository Interface** được adapter hạ tầng (**Infrastructure**) thực thi kết nối thực tế với database hoặc dịch vụ ngoài.
- Kết quả được trả về qua các tầng ngược lại về **API Layer**, sau đó trả response cho client.



Hình 7.1: Sơ đồ lớp cho use-case Đăng nhập/Đăng ký/Đăng xuất/Quản lý tài khoản

Mô tả sơ đồ lớp: Hệ thống Xác thực và Quản lý Người dùng

- **Domain Models:** Đây là lõi dữ liệu của hệ thống, định nghĩa các thực thể chính liên quan đến người dùng và quá trình xác thực:
 - *User*: đại diện hồ sơ người dùng, gồm thông tin định danh (email, username), trạng thái tài khoản (active, suspended), vai trò truy cập (admin, tutor, user), và tập các phương thức kiểm tra quyền.
 - *UserSession*: mô tả phiên đăng nhập, lưu refresh token đã mã hoá, thông tin thiết bị, thời điểm tạo và hết hạn. Hỗ trợ kiểm tra tính hợp lệ và hủy phiên.
 - *PasswordResetToken*: token phục hồi mật khẩu kèm thời gian hết hạn, trạng thái đã dùng, và phương thức xác minh hợp lệ.
 - Các enum như *UserRole*, *UserStatus*, *AuthProvider* đảm bảo rằng mọi giá trị trạng thái trong hệ thống đều rõ ràng, nhất quán.
 - **Domain Repositories:** Các interface quy định hợp đồng truy cập dữ liệu cho tầng nghiệp vụ. Một số chức năng chính:
 - *UserRepository*: tạo, truy vấn, cập nhật, xoá người dùng; tìm theo email, username hoặc tài khoản mạng xã hội.
 - *SessionRepository*: quản lý toàn bộ phiên đăng nhập; truy vấn theo người dùng hoặc token; vô hiệu hóa phiên.
 - *PasswordResetTokenRepository*: tạo, tìm kiếm và xoá token đặt lại mật khẩu; dọn dẹp token hết hạn.

Việc phân tách này giúp logic nghiệp vụ không phụ thuộc vào cơ chế lưu trữ cụ thể.

- **Infrastructure Repositories:** Là cài đặt cụ thể của các interface repository sử dụng MongoDB. Các lớp như *MongoUserRepository*, *MongoSessionRepository* và *MongoPasswordResetTokenRepository* thực hiện chuyển đổi qua lại giữa object domain và tài liệu MongoDB, xử lý index, và thực thi các thao tác CRUD.

- **Infrastructure Security:** Bao gồm các tiện ích bảo mật hỗ trợ xác thực:

- *PasswordHandler*: băm và kiểm chứng mật khẩu.
- *JWTHandler*: tạo, xác minh và giải mã access token và refresh token theo chuẩn JWT.

Hai lớp này cung cấp nền tảng an toàn cho toàn bộ luồng đăng nhập.

- **Core Services:** Đây là tầng xử lý nghiệp vụ của toàn hệ thống xác thực, bao gồm:

- *AuthService*: chịu trách nhiệm đăng ký, đăng nhập, đăng xuất, làm mới token, xác thực Google OAuth, đổi mật khẩu.
- *UserService*: cung cấp các thao tác cơ bản liên quan hồ sơ người dùng.
- *UserManagementService*: dành cho quản trị, hỗ trợ xem thống kê, thay đổi vai trò và trạng thái tài khoản.
- *PasswordResetService*: quản lý toàn bộ quy trình cấp và xác minh token đặt lại mật khẩu, gửi email và cập nhật mật khẩu mới.
- *EmailService*: gửi email hệ thống theo template (welcome, reset password).
- *GoogleOAuthService*: tích hợp với Google, xử lý xác minh ID token và truy xuất thông tin người dùng.

Tầng này sử dụng các repository, công cụ bảo mật và dịch vụ email để cung cấp hành vi hoàn chỉnh.

- **Core Exceptions:** Nhóm ngoại lệ chuẩn hoá các lỗi nghiệp vụ thường gặp như sai mật khẩu, email đã tồn tại, tài khoản bị khóa, chưa xác minh email... nhằm giúp API trả về lỗi thống nhất và dễ debug.
- **API Schemas:** Định nghĩa cấu trúc đầu vào và đầu ra cho các API, như *UserCreateRequest*, *UserLoginRequest*, *TokenResponse*, *PasswordResetRequest*... Nhờ đó, dữ liệu truyền từ client luôn tuân theo schema rõ ràng và dễ kiểm tra.
- **API Endpoints:** Các điểm truy cập RESTful tương ứng với các hành động chính:

- *AuthEndpoints*: đăng ký, đăng nhập, đăng xuất, làm mới token, đăng nhập Google.
- *AdminUserManagementEndpoints*: quản lý người dùng ở cấp quản trị, bao gồm thay đổi role/status và thống kê.
- *PasswordResetEndpoints*: quy trình cấp và xác minh token đặt lại mật khẩu.

Mỗi endpoint ánh xạ dữ liệu từ request vào tầng services và trả về schema response tương ứng.

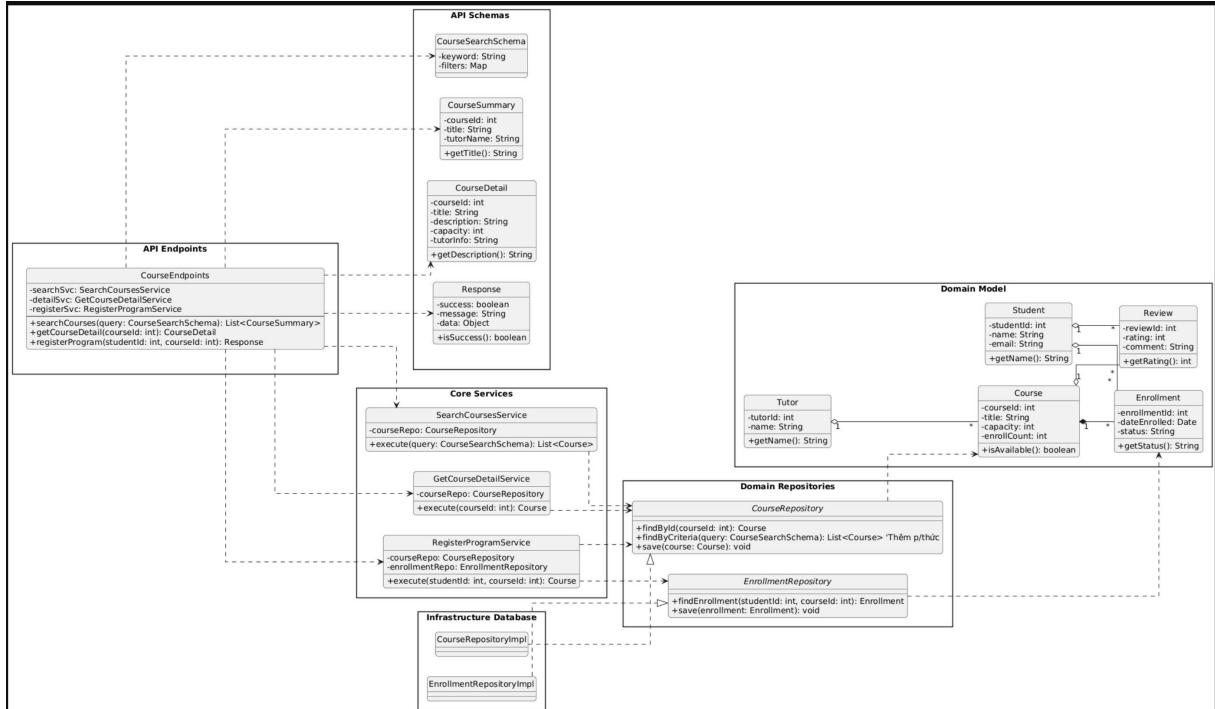


- **Mối quan hệ giữa các lớp:**

- API Endpoints nhận request từ client, ánh xạ request sang schema rồi gọi service tương ứng.
- Core Services thực thi nghiệp vụ và tương tác với các repository để truy xuất hoặc cập nhật dữ liệu.
- Infrastructure Repositories thực thi truy vấn dữ liệu thực tế lên MongoDB.
- Domain Models biểu diễn dữ liệu lõi và các hành vi liên quan.
- Infrastructure Security được sử dụng xuyên suốt trong quá trình xác thực và cấp token.

7.2 Đăng ký chương trình

Sơ đồ lớp mô tả kiến trúc phần mềm của phân hệ "Đăng ký Chương trình". Hệ thống được thiết kế theo Layered Architecture, tuân thủ các nguyên tắc của Clean Architecture, bao gồm 4 tầng chính: API, Core, Domain, và Infrastructure.



Hình 7.2: Sơ đồ lớp cho phân hệ "Đăng ký Chương trình"

Mô tả sơ đồ lớp: Đăng ký Chương trình

- API Endpoints:** Đại diện cho các điểm truy cập RESTful của phân hệ. Lớp `CourseEndpoints` đóng vai trò *coordinator*: tiếp nhận request từ client, kiểm tra và ánh xạ dữ liệu đầu vào theo API Schemas, sau đó điều phối gọi các service nghiệp vụ chuyên biệt (`SearchCoursesService`, `GetCourseDetailService`, `RegisterProgramService`). Nhờ đó, tầng API tập trung vào *giao tiếp* và *điều phối*, không chứa logic nghiệp vụ chi tiết.
- API Schemas:** Định nghĩa các cấu trúc dữ liệu truyền/nhận giữa client và server (Data Transfer Objects). Mục tiêu là tách biệt mô hình dữ liệu dùng cho API khỏi mô hình dữ liệu trong Domain, giúp hạn chế rò rỉ chi tiết triển khai ra phía ngoài. Một số lớp tiêu biểu gồm:
 - `CourseSearchSchema`: dữ liệu truy vấn phục vụ chức năng tìm kiếm.
 - `CourseSummary`, `CourseDetail`: dữ liệu cho các khung nhìn tổng quan và chi tiết khóa học.
 - `Response`: chuẩn hóa cấu trúc phản hồi (mã lỗi, thông báo, payload dữ liệu).



- **Core Services:** Đây là nơi hiện thực phần *business logic* cốt lõi. Các service được thiết kế tách riêng từng nhóm nghiệp vụ:

- `SearchCoursesService`: xử lý nghiệp vụ *đọc* cho chức năng tìm kiếm và lọc khóa học.
- `GetCourseDetailService`: xử lý nghiệp vụ *đọc chi tiết*, lấy thông tin đầy đủ của một khóa học.
- `RegisterProgramService`: xử lý nghiệp vụ *ghi*, bao gồm kiểm tra điều kiện và ghi nhận đăng ký khóa học.

Các service này phối hợp với **Domain Repositories** thông qua interface, giúp giảm phụ thuộc vào tầng lưu trữ cụ thể.

- **Domain Model:** Bao gồm các lớp thực thể lõi như `Course`, `Enrollment`, `Student`, `Tutor`, `Review`. Các lớp này tập trung mô tả:

- Thuộc tính, trạng thái của các đối tượng trong miền bài toán.
- Các hàm nghiệp vụ nội tại, ví dụ `Course.isAvailable()` để kiểm tra trạng thái còn chỗ hay không.

Nhờ đó, *quy tắc nghiệp vụ* được đặt gần dữ liệu của nó, thay vì phân tán trong controller hoặc tầng truy xuất dữ liệu.

- **Domain Repositories:** Định nghĩa các interface đóng vai trò *hợp đồng* truy xuất dữ liệu cho Domain, chẳng hạn:

- `CourseRepository`
- `EnrollmentRepository`
- `TutorRepository`, `ReviewRepository`, ...

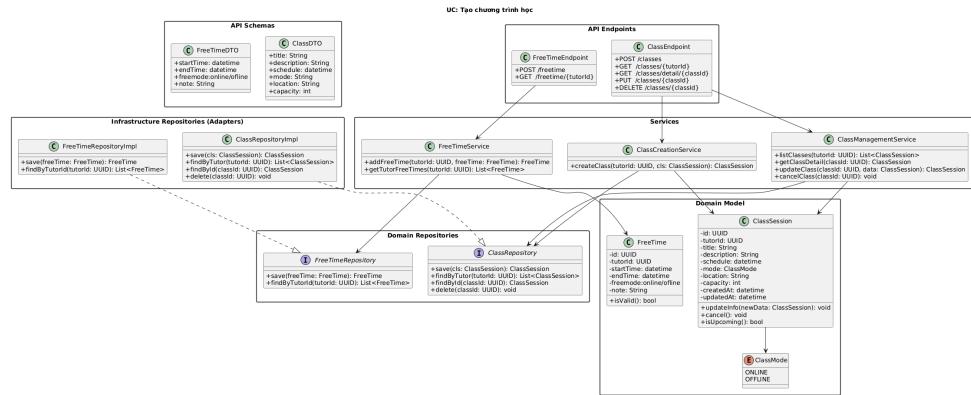
Tầng **Core Services** chỉ phụ thuộc vào các *hợp đồng* này, không phụ thuộc vào chi tiết lưu trữ. Cách tổ chức này hỗ trợ mạnh cho việc kiểm thử (dễ mocking/stubbing) và thay thế công nghệ lưu trữ mà không ảnh hưởng logic nghiệp vụ.

- **Infrastructure Database:** Chứa các lớp hiện thực cụ thể (implementation) của **Domain Repositories**, chẳng hạn `CourseRepositoryImpl`, `EnrollmentRepositoryImpl`. Các lớp này:

- Chịu trách nhiệm ánh xạ giữa thực thể Domain và cấu trúc lưu trữ vật lý.
- Thực thi các truy vấn thực tế đến MongoDB hoặc hệ quản trị cơ sở dữ liệu tương ứng.

Có thể xem đây là *cầu nối* giữa mô hình miền (Domain Model) và lớp lưu trữ dữ liệu thật.

7.3 Tạo chương trình học



Hình 7.3: Class diagram - UC:Tạo chương trình học

Mô tả sơ đồ lớp: Tạo chương trình học

Class diagram trên mô tả kiến trúc nhiều tầng của module “Tạo chương trình học”, bao gồm các thành phần: API Schemas, API Endpoints, Services, Domain Model, Domain Repositories và Infrastructure Adapters.

1. API Schemas

Hai lớp API Schemas được sử dụng để trao đổi dữ liệu giữa client và server.

FreeTime

- startTime:** datetime – thời gian bắt đầu rảnh.
- endTime:** datetime – thời gian kết thúc rảnh.
- freemode:** online/offline – hình thức dạy.
- note:** String – ghi chú của tutor.

Class

- title:** String – tiêu đề lớp học.
- description:** String – mô tả nội dung lớp.
- schedule:** datetime – lịch dạy.
- mode:** String – hình thức ONLINE/OFFLINE.
- location:** String – địa điểm học.
- capacity:** int – số lượng học viên tối đa.



2. API Endpoints

Các endpoint điều phối request và gọi đến service tương ứng.

FreeTimeEndpoint

- POST /freetime – thêm lịch rảnh.
- GET /freetime/{tutorId} – lấy danh sách lịch rảnh của tutor.

ClassEndpoint

- POST /classes – tạo lớp học mới.
- GET /classes/{tutorId} – lấy danh sách lớp theo tutor.
- GET /classes/detail/{classId} – xem chi tiết lớp học.
- PUT /classes/{classId} – cập nhật thông tin lớp.
- DELETE /classes/{classId} – huỷ lớp học.

3. Services

Services chứa toàn bộ logic nghiệp vụ chính của hệ thống.

FreeTimeService

- **addFreeTime(tutorId, freeTime)** – thêm khoảng thời gian rảnh.
- **getTutorFreeTimes(tutorId)** – lấy danh sách free time.

ClassCreationService

- **createClass(tutorId, cls)** – tạo lớp học mới, kiểm tra logic domain.

ClassManagementService

- **listClasses(tutorId)** – lấy danh sách lớp của tutor.
- **getClassDetail(classId)** – xem chi tiết lớp.
- **updateClass(classId, data)** – cập nhật thông tin lớp.
- **cancelClass(classId)** – huỷ lớp học.



4. Domain Model

Hai thực thể chính biểu diễn dữ liệu và hành vi nghiệp vụ.

FreeTime

- **id:** UUID.
- **tutorId:** UUID.
- **startTime, endTime:** datetime – khoảng thời gian rảnh.
- **freemode:** String – online/offline.
- **note:** String.
- **isValid(): bool** – kiểm tra logic hợp lệ của thời gian rảnh.

ClassSession

- **id:** UUID.
- **tutorId:** UUID.
- **title:** String – tiêu đề lớp.
- **description:** String – mô tả lớp học.
- **schedule:** datetime – lịch học.
- **mode:** ClassMode – ONLINE/OFFLINE.
- **location:** String – địa điểm học.
- **capacity:** int – số lượng tối đa.
- **createdAt, updatedAt:** datetime.

Hành vi của ClassSession:

- **updateInfo(newData)** – cập nhật nội dung lớp.
- **cancel()** – huỷ lớp.
- **isUpcoming()** – kiểm tra lớp còn trong tương lai.

ClassMode

- ONLINE.
- OFFLINE.



5. Domain Repositories

Các interface trừu tượng cho tầng domain.

FreeTimeRepository

- save(freeTime): FreeTime.
- findByTutorId(tutorId): List<FreeTime>.

ClassRepository

- save(cls): ClassSession.
- findByTutor(tutorId): List<ClassSession>.
- findById(classId): ClassSession.
- delete(classId): void.

6. Infrastructure Repositories (Adapters)

Hiện thực các interface domain, kết nối với cơ sở dữ liệu.

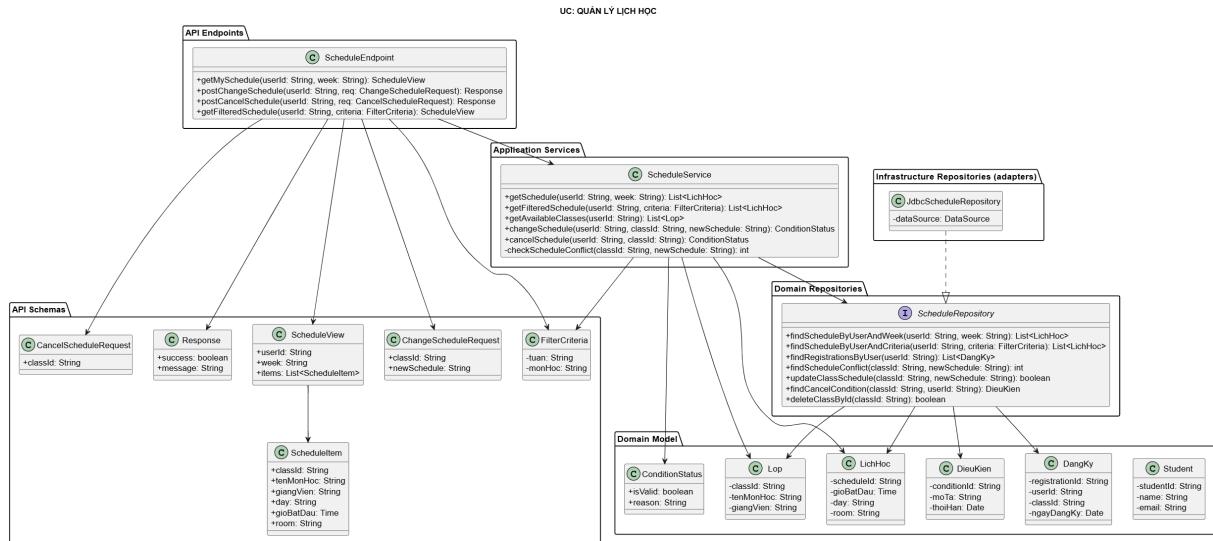
FreeTimeRepositoryImpl

- save(freeTime).
- findByTutorId(tutorId).

ClassRepositoryImpl

- save(cls).
- findByTutor(tutorId).
- findById(classId).
- delete(classId).

7.4 Thiết lập lịch trình cho sinh viên



Hình 7.4: Sơ đồ lớp cho use-case "Xem lại lịch học" cho sinh viên

Mô tả sơ đồ lớp: Xem lại lịch học

Biểu đồ lớp cho use-case “Quản lý lịch học” được thiết kế theo kiến trúc nhiều lớp, tách biệt rõ phần giao tiếp API, lớp dịch vụ nghiệp vụ, lớp truy cập dữ liệu và mô hình miền.

Lớp API Endpoint

- **ScheduleEndpoint:** là lớp Controller chịu trách nhiệm tiếp nhận và xử lý yêu cầu HTTP từ phía client.
 - `getMySchedule(userId, week): ScheduleView` – trả về lịch học theo tuần.
 - `postChangeSchedule(userId, req: ChangeScheduleRequest): Response` – tiếp nhận và xử lý yêu cầu đổi lịch.
 - `postCancelSchedule(userId, req: CancelScheduleRequest): Response` – xử lý yêu cầu hủy lịch.
 - `getFilteredSchedule(userId, criteria: FilterCriteria): ScheduleView` – lấy lịch học theo tiêu chí lọc.

Lớp dịch vụ nghiệp vụ (Application Services)

- **ScheduleService:** hiện thực logic nghiệp vụ cho toàn bộ chức năng lịch học.
 - `getSchedule(userId, week): List<LichHoc>` – lấy lịch học theo tuần.
 - `getFilteredSchedule(userId, criteria): List<LichHoc>` – lấy lịch theo tiêu chí lọc.
 - `getAvailableClasses(userId): List<Lop>` – lấy danh sách lớp mà sinh viên đã đăng ký, dùng cho màn hình đổi/hủy lịch.



- `changeSchedule(userId, classId, newSchedule)`: `ConditionStatus` – kiểm tra trùng lịch và cập nhật ca học mới.
- `cancelSchedule(userId, classId)`: `ConditionStatus` – kiểm tra điều kiện hủy và xóa lớp khỏi lịch.
- `checkScheduleConflict(classId, newSchedule)`: `int` – phương thức nội bộ kiểm tra số lượng xung đột lịch trong cơ sở dữ liệu.

Lớp kho lưu trữ miền (Domain Repositories)

- **ScheduleRepository** (interface): định nghĩa các thao tác dữ liệu mà ScheduleService sử dụng.
 - `findScheduleByUserAndWeek(userId, week)`: `List<LichHoc>`
 - `findScheduleByUserAndCriteria(userId, criteria)`: `List<LichHoc>`
 - `findRegistrationsByUser(userId)`: `List<DangKy>`
 - `findScheduleConflict(classId, newSchedule)`: `int`
 - `updateClassSchedule(classId, newSchedule)`: `boolean`
 - `findCancelCondition(classId, userId)`: `DieuKien`
 - `deleteClassById(classId)`: `boolean`

Lớp repository hạ tầng (Infrastructure Repository)

- **JdbcScheduleRepository**: hiện thực interface ScheduleRepository bằng công nghệ truy cập dữ liệu cụ thể (ví dụ JDBC/ORM).
 - Thuộc tính `dataSource`: `DataSource` dùng để quản lý kết nối tới cơ sở dữ liệu.

Mô hình miền (Domain Model)

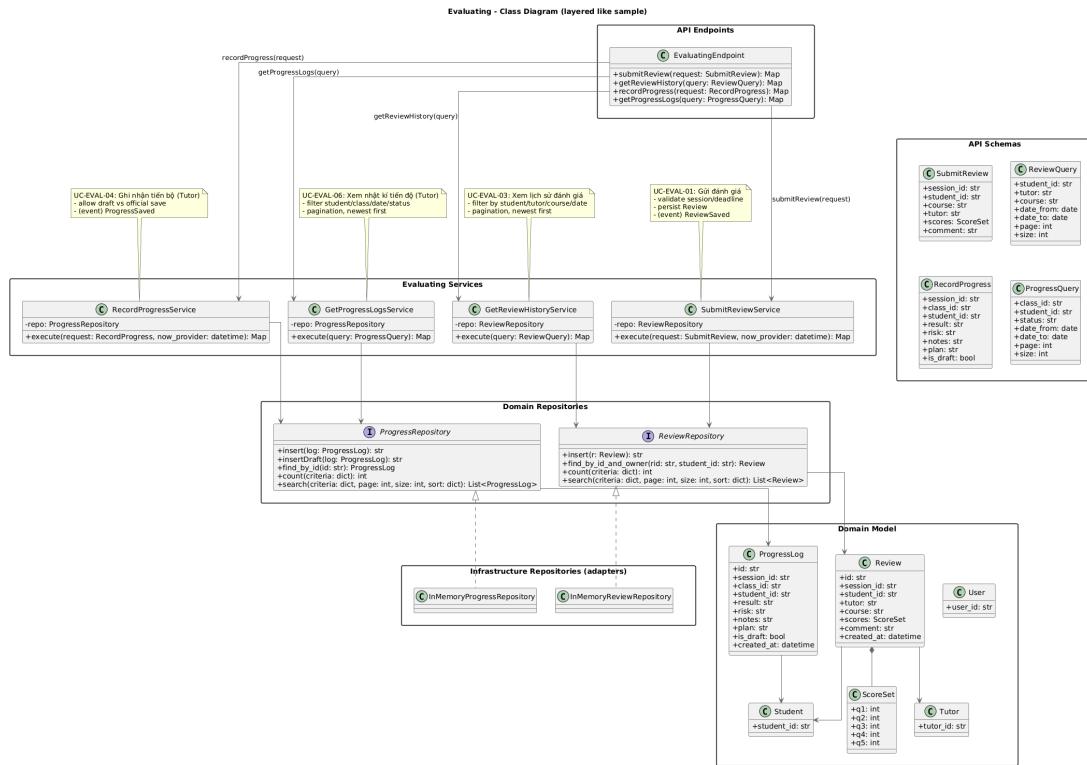
- **Student**: thông tin sinh viên.
 - `studentId`: `String`, `name`: `String`, `email`: `String`.
- **Lop**: lớp học/nhóm học phần.
 - `classId`: `String`, `tenMonHoc`: `String`, `giangVien`: `String`.
- **LichHoc**: một mục lịch học cụ thể.
 - `scheduleId`: `String`, `gioBatDau`: `Time`, `day`: `String`, `room`: `String`.
- **DangKy**: bản ghi đăng ký lớp của sinh viên.
 - `registrationId`: `String`, `userId`: `String`, `classId`: `String`, `ngayDangKy`: `Date`.
- **DieuKien**: thông tin điều kiện áp dụng cho việc hủy/đổi lớp.
 - `conditionId`: `String`, `moTa`: `String`, `thoiHan`: `Date`.
- **ConditionStatus**: kết quả kiểm tra điều kiện nghiệp vụ.

- **isValid**: boolean – cho biết điều kiện có hợp lệ hay không.
- **reason**: String – lý do thành công hoặc thất bại.

API Schemas

- **ChangeScheduleRequest**
 - classId: String, newSchedule: String.
- **CancelScheduleRequest**
 - classId: String.
- **FilterCriteria**
 - tuan: String, monHoc: String.
- **ScheduleView**
 - userId: String, week: String, items: List<ScheduleItem>.
- **ScheduleItem**
 - classId: String, tenMonHoc: String, giangVien: String, day: String, gioBatDau: Time, room: String.
- **Response**
 - success: boolean, message: String.

7.5 Đánh giá



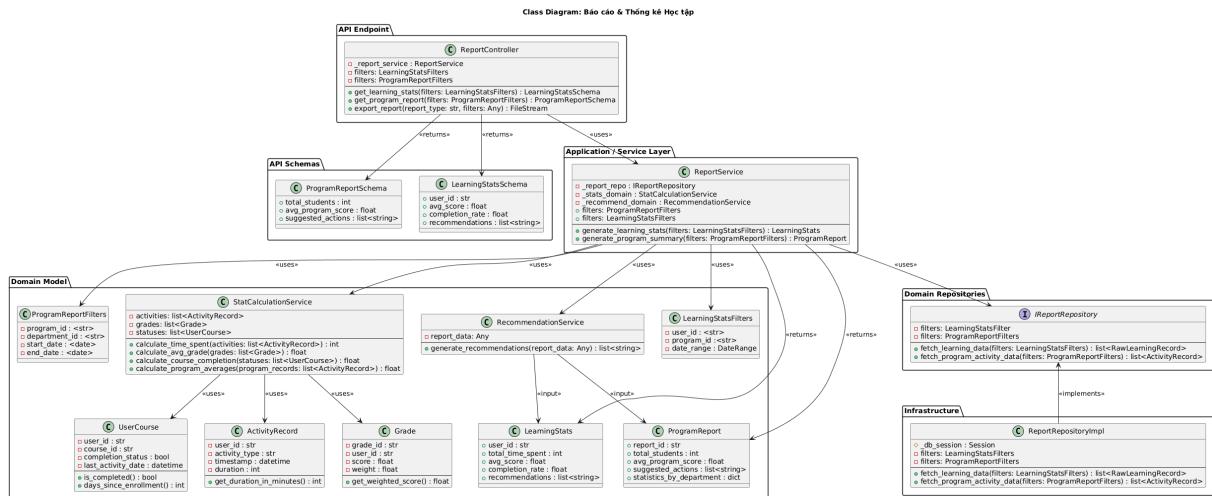
Hình 7.5: Sơ đồ lớp cho use-case "Đánh giá"

Mô tả sơ đồ lớp: Đánh giá

Mô hình lớp (layered) tách rõ *API Endpoints/Controllers*, *Evaluating Services (Use Cases)*, *Domain Repositories (Ports)*, *Infrastructure Repositories (Adapters)* và *Domain Model*:

- **API:** `EvaluatingEndpoint` nhận các request: `submitReview`, `getReviewHistory`, `recordProgress`, `getProgressLogs`.
- **Services:** `SubmitReviewService`, `GetReviewHistoryService`, `RecordProgressService`, `GetProgressLogsService`.
- **Ports (Repos):** `ReviewRepo`, `ProgressRepo` cung cấp `insert`, `findById`, `search`, `count`.
- **Adapters:** hiện thực trong *Infrastructure* (ví dụ `InMemoryReviewRepo`, `InMemoryProgressRepo`) phục vụ demo/test.
- **Domain:** `Review`, `ProgressLog`, `ScoreSet`, `User`, `Student`, `Tutor` và các **DTOs** `SubmitReviewDTO`, `ReviewQuery`, `RecordProgressDTO`, `ProgressQuery`.

7.6 Phân tích và báo cáo



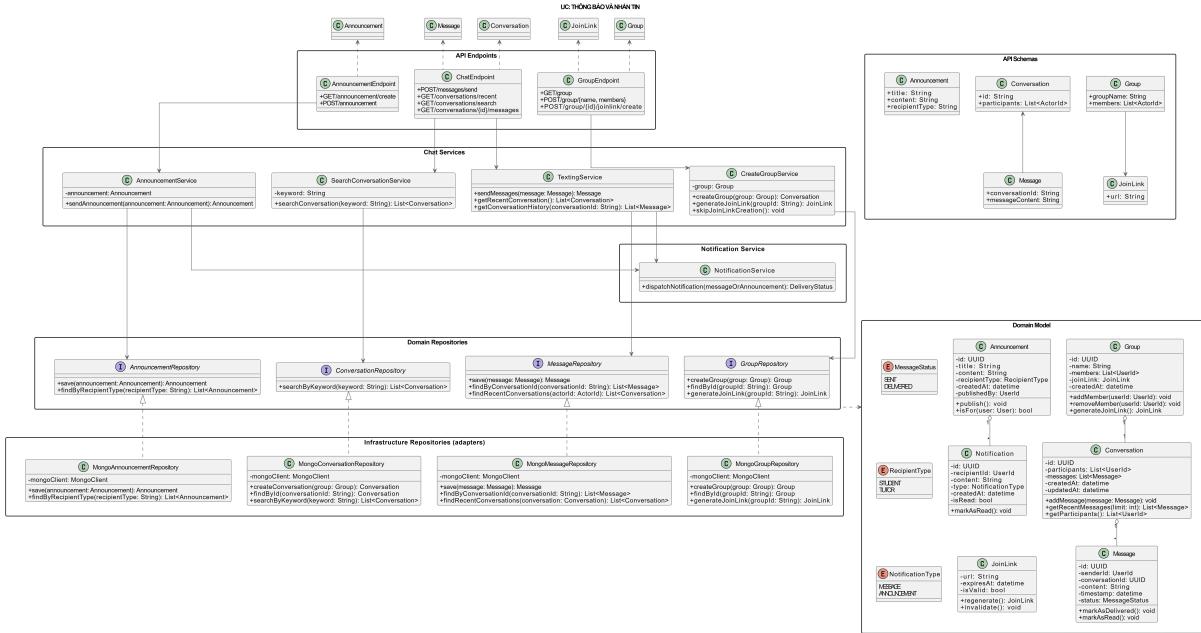
Hình 7.6: Sơ đồ lớp cho phân tích và báo cáo

Mô tả sơ đồ lớp: Phân tích và báo cáo

Mô hình lớp (layered) của hệ thống Báo cáo & Thống kê Học tập được thiết kế theo kiến trúc phân lớp, tách biệt rõ ràng giữa các tầng *Presentation (API Endpoints/Controllers)*, *Application/Service Layer (Use Cases)*, *Domain Repositories (Ports)*, *Infrastructure Repositories (Adapters)* và *Domain Model*:

- API Endpoints:** `ReportController` là bộ điều khiển chính, nhận các request: `get_learning_stats`, `get_program_report`, `export_report`.
- API Schemas:** Các đối tượng truyền dữ liệu cho API, bao gồm `LearningStatsSchema` và `ProgramReportSchema`.
- Application Services:** `ReportService` là lớp dịch vụ ứng dụng, chứa logic điều phối chính thông qua các phương thức: `generate_learning_stats`, `generate_program_summary`.
- Domain Repositories:** Định nghĩa các cổng truy cập dữ liệu (Interface) mà `ReportService` phụ thuộc, cụ thể là `IReportRepository`, cung cấp các phương thức truy vấn dữ liệu thô: `fetch_learning_data`, `fetch_program_activity_data`.
- Infrastructure:** Chứa các lớp triển khai cụ thể của Port, ví dụ `ReportRepositoryImpl`, chịu trách nhiệm tương tác trực tiếp với cơ sở dữ liệu (`_db_session`) để thực hiện các thao tác truy vấn.
- Domain Model:** Bao gồm các thực thể cốt lõi và dịch vụ nghiệp vụ:
 - Domain Services:** `StatCalculationService` (thực hiện tính toán) và `RecommendationService` (thực hiện phân tích và đề xuất).
 - Entities/Value Objects:** `LearningStats`, `ProgramReport`, `ActivityRecord`, `Grade`, `UserCourse`.
 - Filter:** `LearningStatsFilters` và `ProgramReportFilters` được sử dụng để định nghĩa phạm vi báo cáo.

7.7 Thông báo và nhắn tin



Hình 7.7: Sơ đồ lớp cho use-case Thông báo và Nhắn tin

Mô tả sơ đồ lớp: Thông báo và Nhắn tin

- Domain Models:** Đây là các thực thể cốt lõi của hệ thống chat và thông báo, đại diện cho dữ liệu và trạng thái nội bộ:
 - Group:** mô tả nhóm trò chuyện, bao gồm tên nhóm, danh sách thành viên, thời điểm tạo và các hành vi như thêm/xóa thành viên hoặc tạo liên kết tham gia.
 - Conversation:** biểu diễn một cuộc hội thoại (nhóm hoặc cá nhân), chứa danh sách người tham gia, thời gian cập nhật tin nhắn và các phương thức giúp quản lý trạng thái hội thoại.
 - Message:** đại diện tin nhắn được gửi trong một cuộc hội thoại; gồm người gửi, nội dung, thời điểm gửi, trạng thái đã đọc, và các phương thức xác thực hoặc cập nhật trạng thái.
 - Announcement:** thông báo được gửi từ một nhóm hoặc hệ thống, bao gồm nội dung, người tạo, thời điểm phát hành, thời gian hết hạn và các thuộc tính giúp quản lý vòng đời thông báo.
 - JoinLink:** liên kết tham gia nhóm, gồm token mời, ngày hết hạn, giới hạn số lần sử dụng và các phương thức kiểm tra tính hợp lệ.
 - Notification:** thông báo phân phối đến người dùng (ví dụ: có tin nhắn mới, được tag, mời vào nhóm). Lưu trạng thái đã gửi, đã nhận, đã đọc cùng phương thức cập nhật.

- **Domain Repositories:** Các interface quy định việc truy vấn và lưu trữ dữ liệu cho tầng nghiệp vụ. Một số repository chính:
 - **GroupRepository:** tạo và cập nhật nhóm, truy vấn danh sách nhóm theo người dùng, quản lý thành viên và join link.
 - **ConversationRepository:** truy vấn hội thoại theo người dùng, tìm kiếm theo từ khóa, cập nhật thời điểm hoạt động gần nhất.
 - **MessageRepository:** tạo, lưu trữ và truy vấn lịch sử tin nhắn.
 - **AnnouncementRepository:** quản lý thông báo nhóm, truy xuất theo thời gian hoặc theo nhóm.
 - **NotificationRepository:** lưu trạng thái thông báo, phục vụ quá trình đồng bộ giữa các thiết bị.

Việc tách các repository giúp logic nghiệp vụ không phụ thuộc vào cơ chế lưu trữ dữ liệu cụ thể.

- **Infrastructure Repositories:** Là cài đặt cụ thể của các interface repository sử dụng MongoDB. Các lớp MongoGroupRepository, MongoMessageRepository và MongoConversationRepository thực hiện ánh xạ giữa object domain và tài liệu MongoDB.
- **Core Services:** Tầng xử lý nghiệp vụ của hệ thống thông báo và nhắn tin, bao gồm:
 - **ChatService:** tạo nhóm, quản lý join link, tạo hội thoại, gửi tin nhắn, tìm kiếm tài nguyên chat và quản lý thành viên.
 - **MessageService:** điều phối việc gửi tin nhắn, cập nhật trạng thái đã đọc và tải lịch sử tin nhắn.
 - **AnnouncementService:** tạo, cập nhật và phân phối thông báo nhóm.
 - **NotificationService:** chịu trách nhiệm phát thông báo hệ thống, đồng bộ trạng thái thông báo qua WebSocket hoặc push notification.

Các service này phối hợp với domain repositories và đóng vai trò trung tâm trong các luồng xử lý chat và thông báo.

- **API Schemas:** Định nghĩa cấu trúc dữ liệu đầu vào và đầu ra của API, như CreateGroupRequest, MessageSendRequest, AnnouncementCreateRequest, NotificationResponse... Nhờ đó việc trao đổi dữ liệu giữa client và server luôn rõ ràng và có kiểm soát.
- **API Endpoints:** Các điểm truy cập RESTful ứng với các hành động chính của hệ thống:
 - **ChatEndpoints:** gửi tin nhắn, tải lịch sử tin nhắn, tìm kiếm tin nhắn, xem tin nhắn.
 - **GroupEndpoints:** tạo nhóm, cập nhật thông tin nhóm, lấy danh sách nhóm, tạo join link, tham gia nhóm.
 - **AnnouncementEndpoints:** tạo và gửi thông báo.

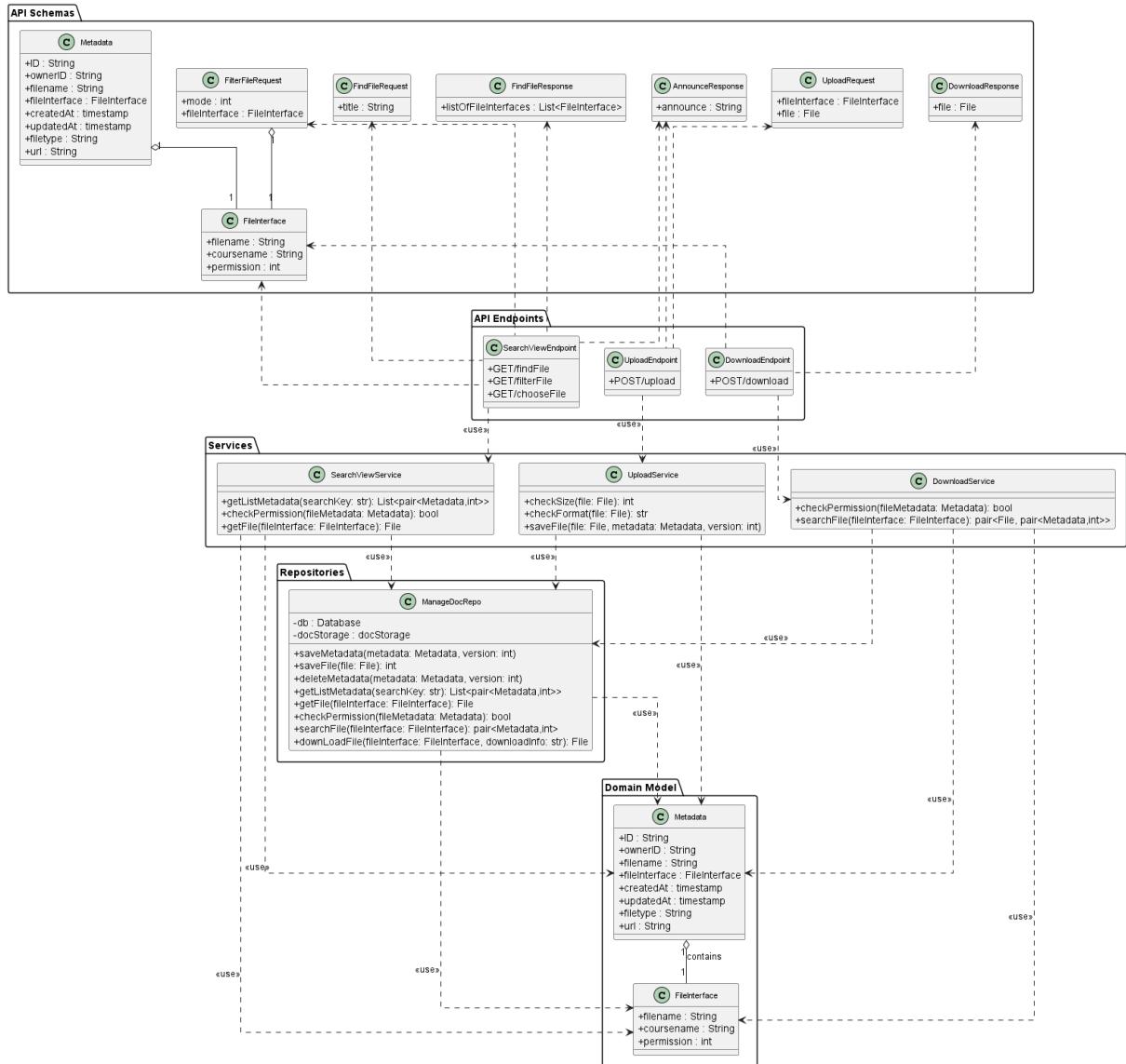
Mỗi endpoint ánh xạ request thành schema phù hợp rồi gọi service tương ứng.



- **Mối quan hệ giữa các lớp:**

- API Endpoints nhận request từ client, ánh xạ sang schema và gọi service tương ứng.
- Core Services thực thi nghiệp vụ và tương tác với các repository để truy vấn hoặc cập nhật dữ liệu.
- Infrastructure Repositories thực hiện truy vấn dữ liệu thực tế lên MongoDB.
- Domain Models biểu diễn dữ liệu lõi và các hành vi liên quan.
- NotificationService phối hợp với MessageService và AnnouncementService để phân phối thông báo và đồng bộ trạng thái.
- API Schemas đảm bảo dữ liệu vào/ra từ client luôn nhất quán và dễ kiểm tra.

7.8 Quản lý tài liệu



Hình 7.8: Sơ đồ lớp cho use-case Quản lý tài liệu

Mô tả sơ đồ lớp: Quản lý tài liệu

- Sơ đồ lớp mô tả các thành phần chính của hệ thống quản lý tài liệu, bao gồm các lớp thuộc API Endpoints, API Schemas, Services, Domain Model và Repositories.
- API Endpoints:**
 - Gồm các lớp như `DownloadEndpoint`, `UploadEndpoint`, `SearchViewEndpoint`.
 - Các lớp này đại diện cho các điểm cuối API, cung cấp các phương thức như tải lên, tải xuống, tìm kiếm, lọc và chọn tài liệu.

- **API Schemas:**

- Định nghĩa các lớp dữ liệu trao đổi giữa client và server như `FindFileRequest`, `FindFileResponse`, `AnnounceResponse`, `UploadRequest`, `DownloadResponse`.
- Các lớp `FileInterface` và `Metadata` mô tả thông tin file và metadata liên quan.
- Các lớp này giúp chuẩn hóa dữ liệu đầu vào/đầu ra cho các endpoint.

- **Services:**

- Gồm các lớp `UploadService`, `SearchViewService`, `DownloadService`.
- Các service này thực hiện các nghiệp vụ như kiểm tra file, lưu file, kiểm tra quyền truy cập, tìm kiếm và tải file.

- **Domain Model:**

- Định nghĩa các lớp cốt lõi như `FileInterface` và `Metadata`, mô tả thông tin chi tiết về tài liệu và metadata liên quan.
- Các lớp này là nền tảng cho việc lưu trữ và xử lý dữ liệu tài liệu trong hệ thống.

- **Repositories:**

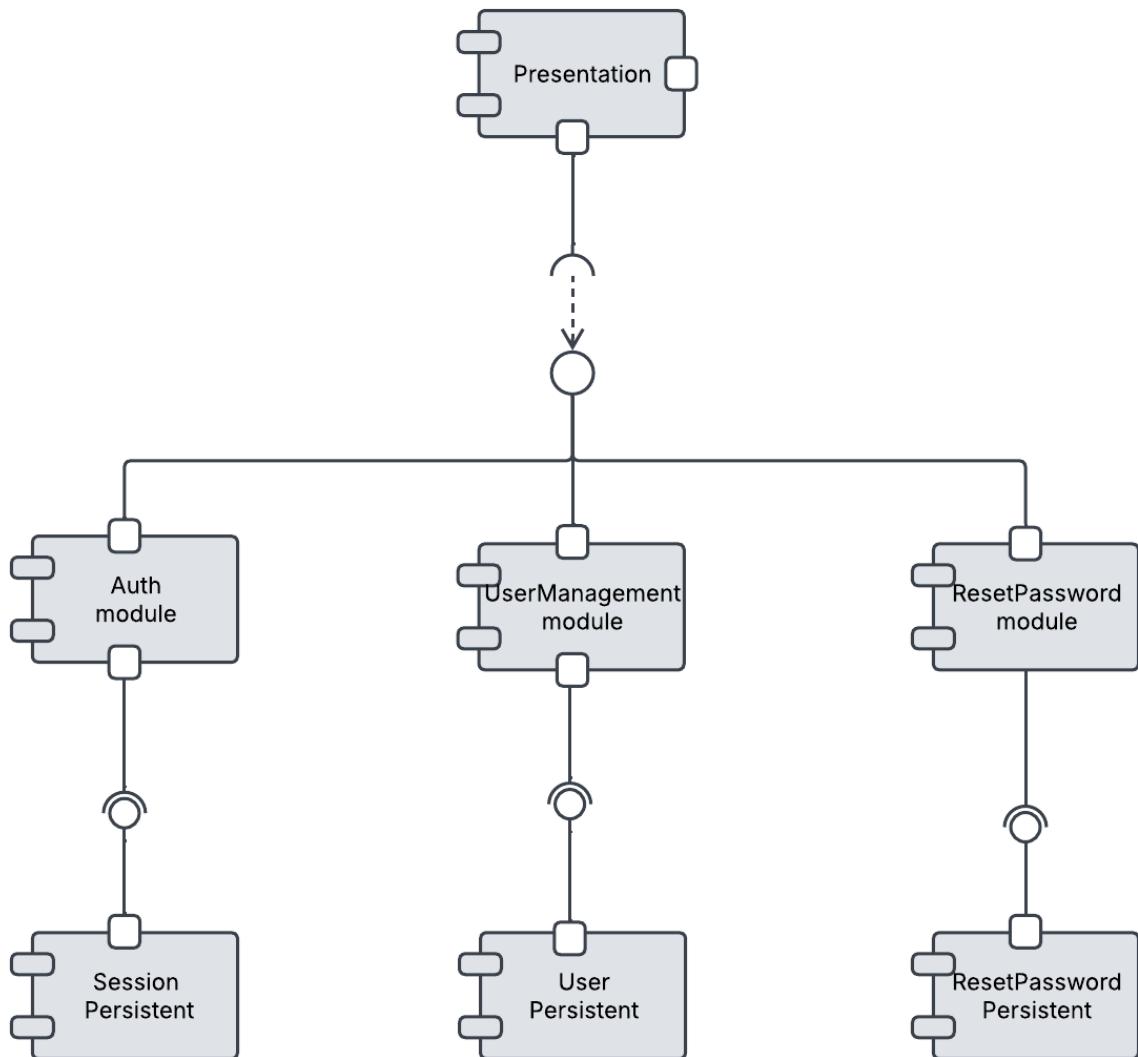
- Lớp `ManageDocRepo` có các hàm bao bọc các hàm tương ứng của cơ sở dữ liệu và kho lưu trữ tài liệu.
- Cung cấp các phương thức lưu, truy xuất và tải file từ kho lưu trữ và cơ sở dữ liệu thông qua các hàm bao bọc.

- **Luồng xử lý:**

- Các API Endpoints nhận request từ client và gọi trực tiếp các Service để xử lý logic nghiệp vụ.
- Các Service này sử dụng các phương thức của Repositories để truy xuất hoặc cập nhật dữ liệu trong cơ sở dữ liệu và kho lưu trữ.
- Domain Model liên kết chặt chẽ với các repository, định nghĩa các thực thể và hành vi xử lý nội bộ dữ liệu.
- Các lớp dữ liệu (API Schemas) được sử dụng để trao đổi thông tin giữa client và server, đồng thời làm cầu nối giữa các tầng của hệ thống.

8 Development view

8.1 Quản lý truy cập và xác thực



Hình 8.1: Component diagram cho use-case "Hệ thống Xác thực và Quản lý Người dùng"

Mô tả sơ đồ thành phần: Hệ thống Xác thực và Quản lý Người dùng

Sơ đồ thành phần của hệ thống Xác thực và Quản lý người dùng được tổ chức theo ba lớp chính, đảm bảo sự phân tách nhiệm vụ (separation of concerns), giảm phụ thuộc và thuận tiện mở rộng:

- **Giao diện người dùng (UI):**

- **Presentation:** Chịu trách nhiệm nhận yêu cầu từ phía người dùng (đăng nhập, đăng ký, cập nhật hồ sơ, đặt lại mật khẩu), kiểm tra dữ liệu đầu vào và chuyển tiếp request xuống các module nghiệp vụ tương ứng.



- **Logic nghiệp vụ (Business Logic):**

- **Auth Module:** Đảm nhiệm các chức năng:

- * *Login:* Xác thực thông tin người dùng, kiểm tra mật khẩu, sinh phiên đăng nhập (session/token).
 - * *Logout:* Thu hồi hoặc vô hiệu hóa phiên đăng nhập hiện tại.
 - * *Register:* Tạo tài khoản mới, kiểm tra trùng lặp email/username.

Module này giao tiếp với **Session Persistent** để lưu token/phiên và truy vấn thông tin đăng nhập.

- **UserManagement Module:** Thực hiện nghiệp vụ quản lý thông tin người dùng:

- * Cập nhật hồ sơ cá nhân
 - * Quản lý trạng thái tài khoản
 - * Thay đổi mật khẩu khi đã đăng nhập

Thành phần này truy xuất dữ liệu từ **User Persistent** để đọc/ghi thông tin tài khoản.

- **ResetPassword Module:** Tiếp nhận và xử lý các yêu cầu đặt lại mật khẩu:

- * Sinh mã khôi phục (token)
 - * Xác thực mã khôi phục
 - * Thiết lập lại mật khẩu mới

Module này gọi tới **ResetPassword Persistent** để lưu và kiểm tra mã khôi phục.

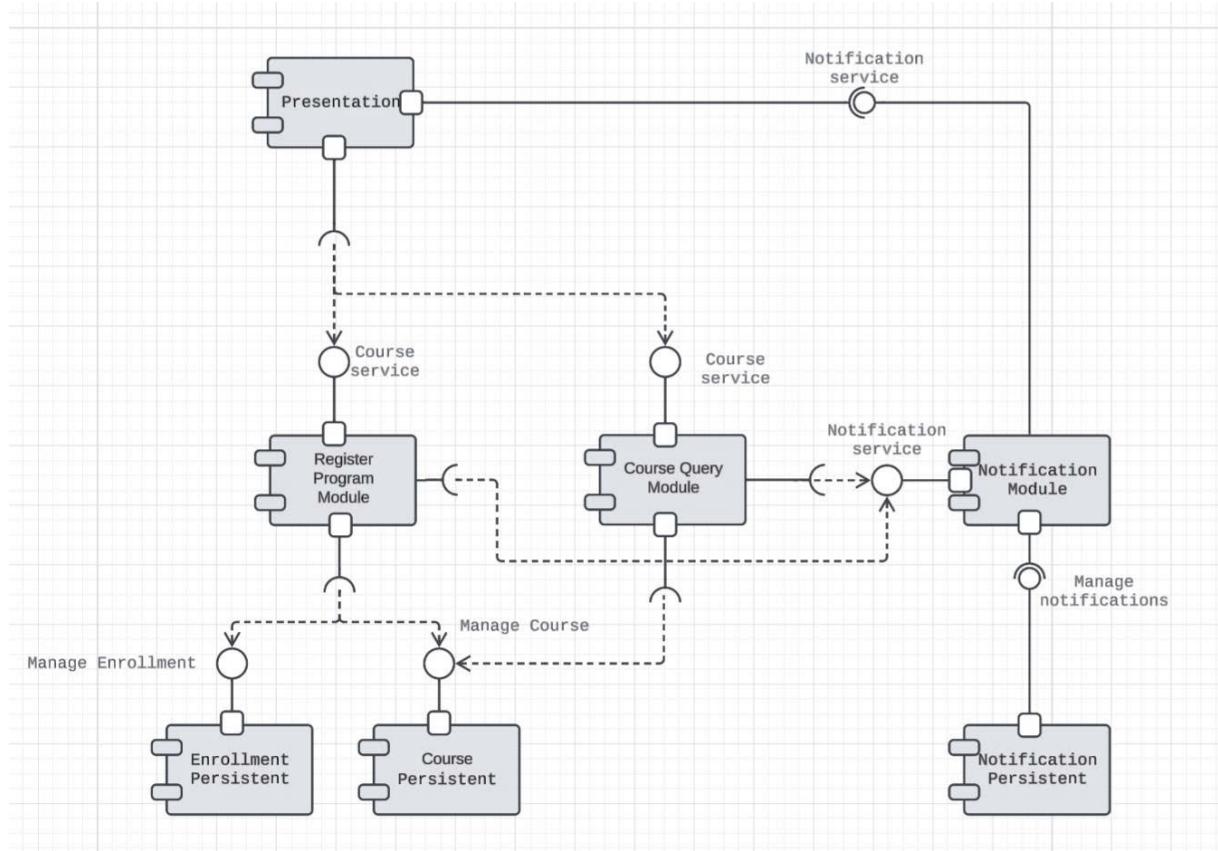
- **Tầng lưu trữ (Persistence Layer):**

- **Session Persistent:** Lưu phiên đăng nhập, token xác thực, trạng thái hoạt động của phiên.
 - **User Persistent:** Lưu trữ thông tin tài khoản, hồ sơ người dùng, trạng thái (active/suspended).
 - **ResetPassword Persistent:** Lưu mã đặt lại mật khẩu và thời gian hiệu lực, phục vụ xác thực trong quy trình khôi phục.

Luồng tương tác (Interfaces):

- Thành phần **Presentation** gửi yêu cầu tương ứng xuống các module nghiệp vụ: Auth, UserManagement và ResetPassword.
- Các module nghiệp vụ xử lý logic, truy cập dữ liệu qua các thành phần lưu trữ tương ứng (Session, User, ResetPassword).
- Tầng Persistence phản hồi dữ liệu giúp các module hoàn thành quy trình xác thực, quản lý và đặt lại mật khẩu.

8.2 Đăng ký chương trình



Hình 8.2: Component diagram - Kiến trúc Module Phân tầng

Mô tả sơ đồ thành phần: Đăng ký chương trình

Tầng Presentation

- [Presentation]: Đại diện cho toàn bộ giao diện người dùng (UI).
- Giao tiếp (Ports):
 - Yêu cầu (Requires): Port Course service để thực hiện các nghiệp vụ chính.
 - Cung cấp (Provides): Port Notification service (để nhận và hiển thị thông báo).

Tầng Application Modules

Tầng này chứa các module nghiệp vụ (gộp chung API và Core Services).

- [Course Query Module]: Hiện thực các use-case nghiệp vụ "đọc" (tìm kiếm, xem chi tiết).
 - Cung cấp (Provides): Port Course service (cho Presentation).
 - Yêu cầu (Requires): Port Manage Course (từ tầng Persistence) và port Notification service.

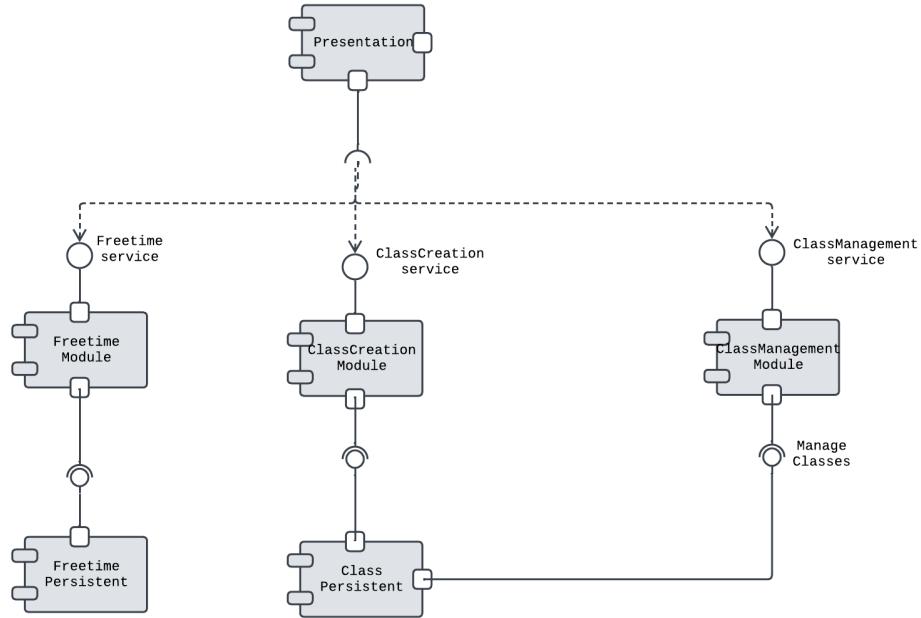
- **[Register Program Module]**: Hiện thực use-case nghiệp vụ "ghi" (đăng ký).
 - **Cung cấp (Provides)**: Port Course service (cho Presentation).
 - **Yêu cầu (Requires)**: Port Manage Course và Manage Enrollment (từ tầng Persistence).
 - **Phụ thuộc (Dependency)**: Phụ thuộc vào Notification Module (nét đứt) để gửi thông báo.
- **[Notification Module]**: Chịu trách nhiệm điều phối và phát dịch vụ thông báo.
 - **Cung cấp (Provides)**: Port Notification service.
 - **Yêu cầu (Requires)**: Port Notification service (từ Presentation) và port Manage notifications (từ tầng Persistence).

Tầng Persistence Modules

Đây là tầng truy xuất dữ liệu (Infrastructure), là các adapter truy cập DB.

- **[Course Persistent]**: Triển khai và cung cấp (provides) port Manage Course.
- **[Enrollment Persistent]**: Triển khai và cung cấp (provides) port Manage Enrollment.
- **[Notification Persistent]**: Triển khai và cung cấp (provides) port Manage notifications.

8.3 Tạo chương trình học



Hình 8.3: Component - Tạo chương trình học

Mô tả sơ đồ thành phần: Tạo chương trình học

Giao diện người dùng (UI)

Presentation: Đảm nhiệm việc tiếp nhận yêu cầu từ người dùng liên quan đến tạo lớp học, quản lý lớp và quản lý thời gian rảnh. Thành phần này kiểm tra dữ liệu đầu vào và chuyển tiếp yêu cầu xuống các dịch vụ nghiệp vụ tương ứng.

Logic nghiệp vụ (Business Logic)

Freetime Module: Xử lý các nghiệp vụ liên quan đến thời gian rảnh của Tutor như lấy danh sách thời gian rảnh, cập nhật hoặc kiểm tra xung đột. Module này giao tiếp với *Freetime Persistent* để truy xuất dữ liệu thời gian rảnh đã sẵn có.

ClassCreation Module: Đảm nhiệm quy trình tạo lớp học mới, bao gồm kiểm tra tính hợp lệ, ràng buộc lịch và khởi tạo dữ liệu lớp. Thành phần này tương tác với *Class Persistent* để lưu thông tin lớp mới.

ClassManagement Module: Quản lý các lớp hiện có của tutor: cập nhật thông tin lớp, thay đổi giáo viên phụ trách, điều chỉnh lịch dạy và tra cứu danh sách lớp. Module này cũng làm việc với *Class Persistent* để đọc/ghi dữ liệu.



Tầng lưu trữ (Persistence Layer)

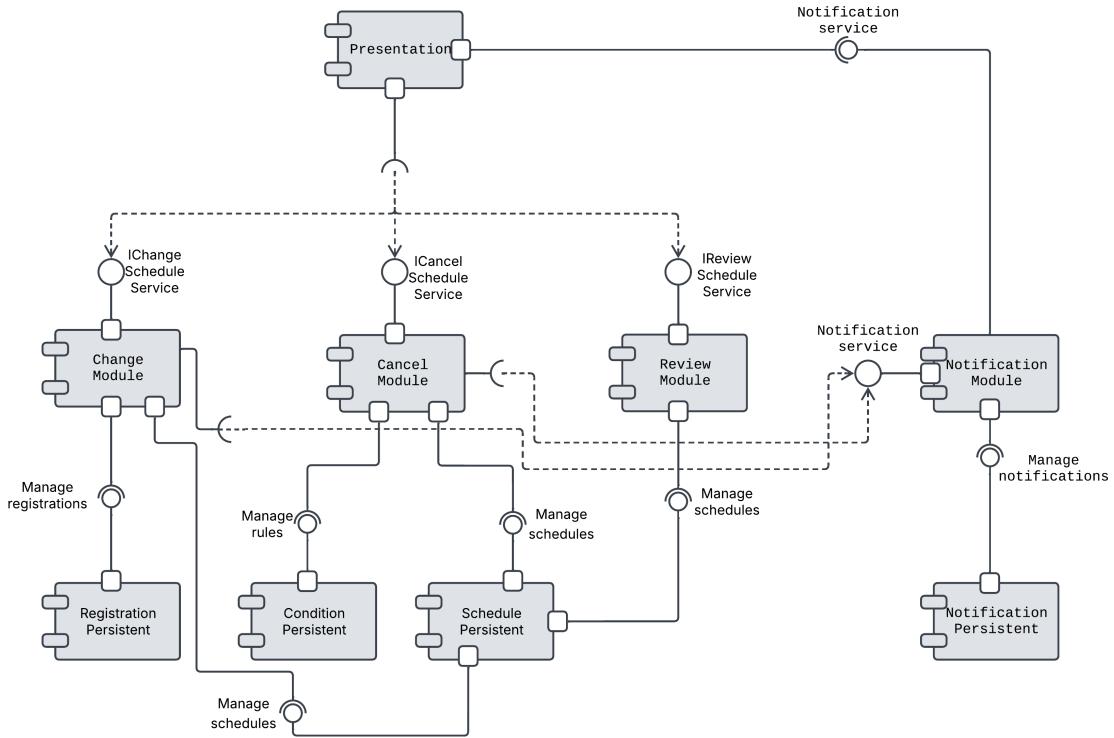
Freetime Persistent: Lưu trữ thông tin thời gian rảnh, phục vụ cho sinh viên tham khảo lịch.

Class Persistent: Lưu trữ dữ liệu các lớp học: lịch, giáo viên, trạng thái lớp và các thông tin quản lý liên quan.

Luồng tương tác (Interfaces)

- Presentation là điểm vào của hệ thống, gửi yêu cầu xuống các module nghiệp vụ: *Freetime*, *Class-Creation* và *ClassManagement*.
- Các module nghiệp vụ xử lý logic tương ứng và truy cập tầng lưu trữ để lấy hoặc lưu dữ liệu.
- Tầng Persistence phản hồi dữ liệu giúp các module hoàn thành nghiệp vụ như kiểm tra lịch, tạo lớp mới hoặc cập nhật thông tin lớp.

8.4 Thiết lập lịch trình cho sinh viên



Hình 8.4: Sơ đồ thành phần cho use-case "Xem lại lịch học" cho sinh viên

Mô tả sơ đồ thành phần: Xem lại lịch học

Hình 8.4 mô tả kiến trúc thành phần của chức năng *Quản lý lịch học* ở góc nhìn *development view*. Các thành phần được tổ chức thành ba tầng chính: tầng trình bày (Presentation), các module nghiệp vụ (Change/Cancel/Review) và các thành phần lưu trữ (Persistent), cùng với module Thông báo.

Thành phần Presentation

- **Presentation:** đại diện cho giao diện người dùng của hệ thống. Nó cung cấp màn hình cho ba use-case: đổi lịch, hủy lịch và xem lại lịch.
- Presentation *phụ thuộc* vào ba interface dịch vụ: **IChangeScheduleService**, **ICcancelScheduleService** và **IReviewScheduleService**. Các quan hệ phụ thuộc này được thể hiện bằng các đường nét đứt từ Presentation đến ba vòng tròn interface tương ứng.
- Đồng thời, Presentation cũng kết nối với *Notification service* bên ngoài để nhận và hiển thị các thông báo liên quan đến lịch học cho sinh viên (ví dụ thông báo đổi lịch thành công, hủy lịch thành công).



Các module nghiệp vụ

- **Change Module:**

- Hiện thực interface **IChangeScheduleService** (thể hiện bằng socket ở thành phần Change Module gắn với ball của interface).
- Chịu trách nhiệm xử lý yêu cầu *đổi lịch*: kiểm tra đăng ký hiện có, cập nhật ca học mới vào lịch và phát sinh sự kiện thông báo nếu cần.
- Sử dụng **Registration Persistent** để quản lý danh sách đăng ký lớp (connector “Manage registrations”).
- Sử dụng **Schedule Persistent** để cập nhật thông tin lịch học (connector “Manage schedules”).

- **Cancel Module:**

- Hiện thực interface **ICancelScheduleService**.
- Xử lý yêu cầu *hủy lịch*: kiểm tra điều kiện hủy, xác định lớp có được phép hủy hay không và nếu hợp lệ thì xóa lịch tương ứng.
- Kết nối với **Registration Persistent** (đọc các lớp đã đăng ký), **Condition Persistent** (đọc các quy tắc/thời hạn hủy – connector “Manage rules”) và **Schedule Persistent** (xóa mục lịch – connector “Manage schedules”).
- Các thao tác hủy thành công có thể phát sinh sự kiện tới **Notification Module** thông qua interface *Notification service* (đường nét đứt từ Cancel Module sang vòng tròn Notification service).

- **Review Module:**

- Hiện thực interface **IReviewScheduleService**.
- Đảm nhiệm chức năng *xem lại lịch học*: lấy lịch theo tuần hiện tại hoặc theo tiêu chí lọc (tuần/môn học).
- Sử dụng **Schedule Persistent** thông qua interface “Manage schedules” để truy vấn dữ liệu lịch học.

Các thành phần lưu trữ (Persistent)

Mỗi thành phần Persistent tương ứng với một *kho dữ liệu* riêng, cung cấp các interface truy cập dữ liệu cho các module nghiệp vụ:

- **Registration Persistent:**

- Lưu trữ thông tin đăng ký lớp của sinh viên.
- Cung cấp interface *Manage registrations* cho Change Module và Cancel Module đọc/ghi danh sách đăng ký.



- **Condition Persistent:**

- Lưu trữ các điều kiện/quy tắc áp dụng cho việc hủy hoặc đổi lịch (thời hạn, trạng thái lớp, v.v.).
- Cung cấp interface *Manage rules* cho Cancel Module để kiểm tra điều kiện hủy.

- **Schedule Persistent:**

- Lưu trữ thông tin lịch học theo tuần, ca học, phòng học.
- Cung cấp interface *Manage schedules* cho Change Module, Cancel Module và Review Module để đọc/ghi dữ liệu lịch.

- **Notification Persistent:**

- Lưu trữ các thông báo đã gửi cho người dùng (ví dụ thông báo đổi lịch/hủy lịch).
- Cung cấp interface *Manage notifications* cho Notification Module.

Notification Module và Notification service

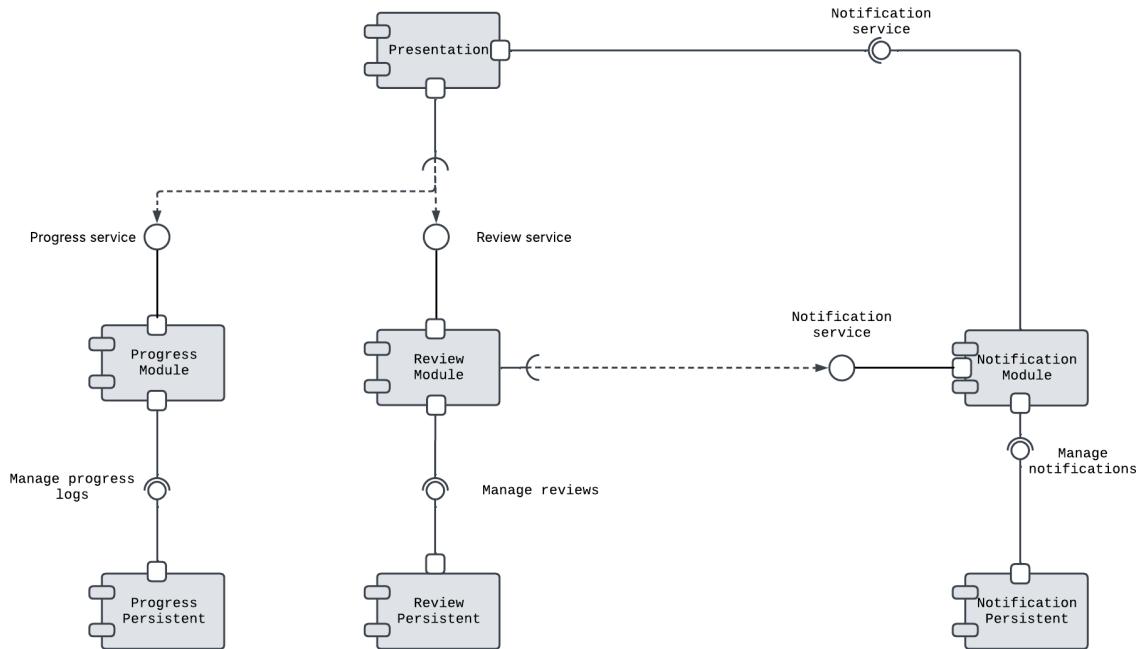
- **Notification Module:**

- Nhận các yêu cầu gửi thông báo từ các module nghiệp vụ thông qua interface *Notification service* (ball-socket ở bên phải hình).
 - Thực hiện logic gửi và lưu thông báo, sử dụng **Notification Persistent** để ghi nhận lịch sử thông báo (connector “Manage notifications”).
- Interface *Notification service* cũng được kết nối tới thành phần Presentation, cho phép UI đăng ký/nhận thông báo liên quan tới lịch học của sinh viên.

Nhìn chung, biểu đồ thành phần này cho thấy rõ:

- Presentation chỉ phụ thuộc vào các interface dịch vụ trừu tượng, không phụ thuộc trực tiếp vào lớp triển khai hay tầng lưu trữ.
- Mỗi module nghiệp vụ (Change, Cancel, Review) được tách riêng, truy cập dữ liệu thông qua các Persistent tương ứng, giúp việc mở rộng hoặc thay đổi logic từng phần trở nên dễ dàng hơn.
- Cơ chế thông báo được gom vào một module độc lập (Notification Module), nhận sự kiện từ các module lịch học và quản lý lưu trữ thông báo tập trung.

8.5 Đánh giá



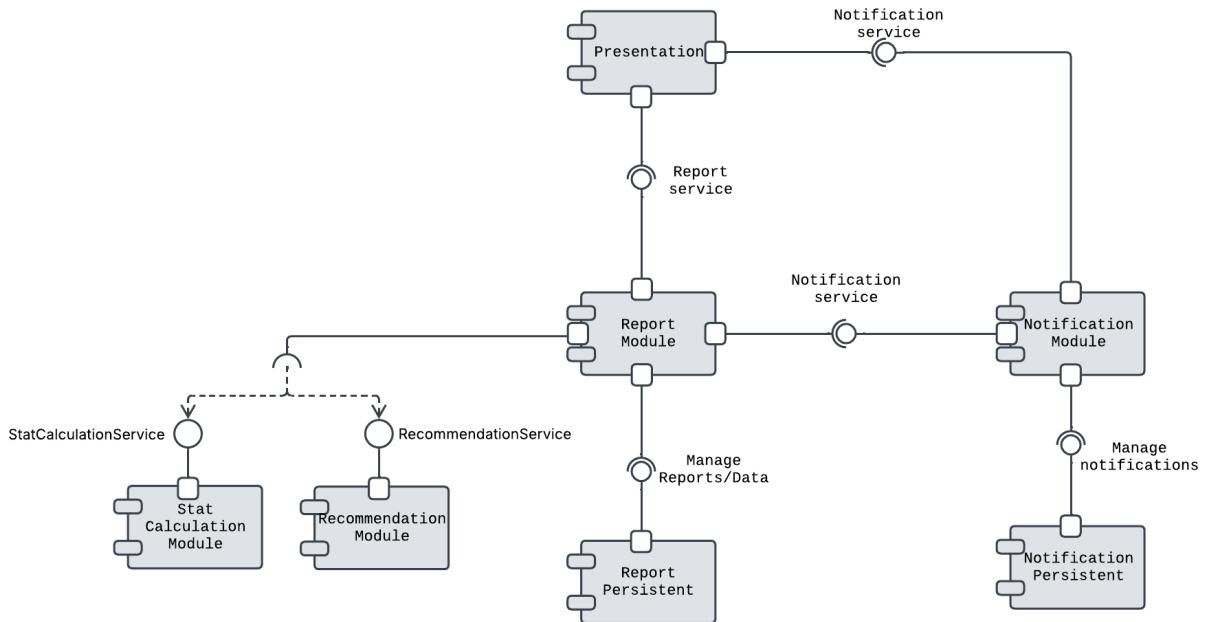
Hình 8.5: Sơ đồ thành phần cho use-case "Đánh giá"

Mô tả sơ đồ thành phần: Đánh giá

Ánh xạ development view ở mức module:

- **Presentation:** REST controllers/routers của Evaluating; giao tiếp *Notification service* (async) để bắn thông báo sau khi lưu.
- **Review Module & Progress Module:** hiện thực use-case và domain service; phụ thuộc **Domain Ports (Repositories)**.
- **Review Persistent & Progress Persistent:** adapters truy cập DB (ORM/DAO); triển khai các port *ReviewRepo*, *ProgressRepo*.
- **Notification Module/Persistent:** phát và lưu trạng thái thông báo liên quan (nếu cấu hình).

8.6 Phân tích và báo cáo

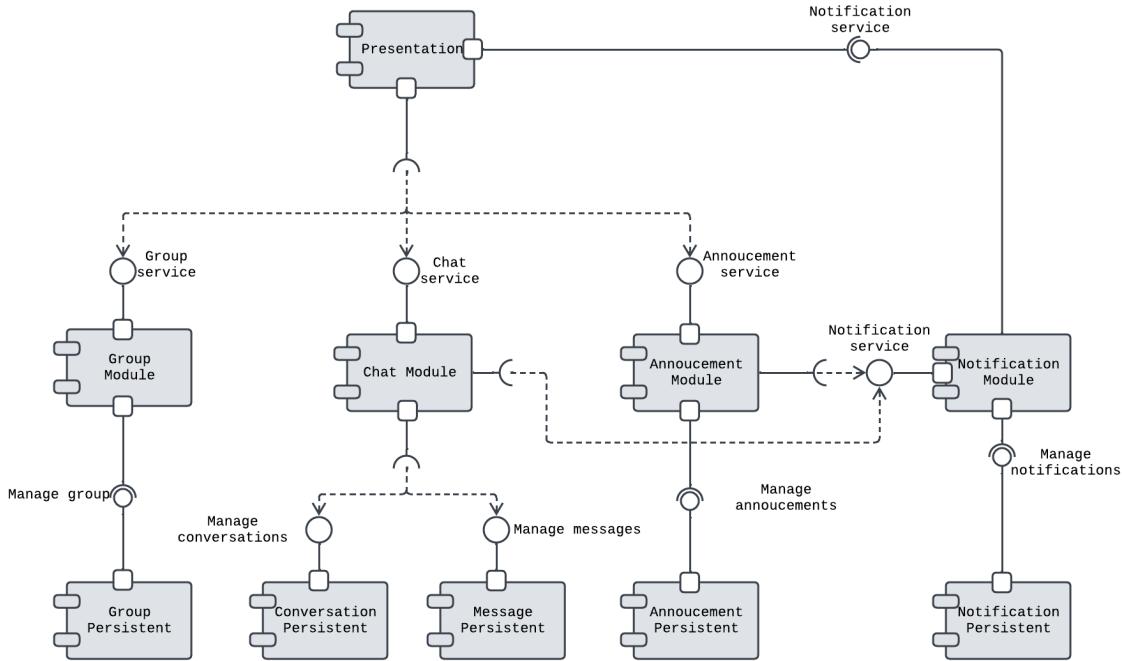


Hình 8.6: Sơ đồ thành phần cho use-case "Phân tích và báo cáo"

Mô tả sơ đồ thành phần: Phân tích và báo cáo

- Presentation:** Chứa các REST controllers/routers của hệ thống Báo cáo & Thống kê (ReportController); giao tiếp Notification service (async) để bắn thông báo sau khi hoàn tất các tác vụ nặng (ví dụ: xuất file báo cáo).
- Review Module:** Hiện thực các use-case và domain service chính (ReportService), chịu trách nhiệm điều phối toàn bộ quá trình tạo báo cáo và thống kê. Nó phụ thuộc vào các Domain Ports (Repositories).
- Stat Calculation Module & Recommendation Module:** Hiện thực các Domain Service chuyên biệt (ví dụ: StatCalculationService, RecommendationService) để thực hiện các thuật toán tính toán và phân tích đề xuất phức tạp. Chúng được Report Module sử dụng.
- Report Persistent:** Là adapter truy cập DB (ORM/DAO) cho dữ liệu báo cáo; triển khai các port như IReportRepository để lấy dữ liệu thô (RawLearningRecord, ActivityRecord).
- Notification Module/Persistent:** Phát dịch vụ Notification service để các module khác sử dụng (ví dụ: Report Module, Presentation) và lưu trạng thái thông báo liên quan (nếu cấu hình) trong Notification Persistent.

8.7 Thông báo và nhắn tin



Hình 8.7: Component diagram cho use-case "Thông báo và Nhắn tin"

Mô tả sơ đồ thành phần: Thông báo và Nhắn tin

Sơ đồ thành phần hệ thống **Thông báo và Nhắn tin** được phân chia làm ba lớp chức năng chính như sau:

- **Giao diện người dùng (UI):**
 - **Presentation:** Là thành phần giao tiếp với người dùng cuối, nhận thao tác từ phía giao diện và chuyển tiếp các yêu cầu dịch vụ (request) tới các module xử lý nghiệp vụ bên dưới.
- **Logic nghiệp vụ (Business Logic):**
 - **Group Module:** Thực hiện các nghiệp vụ về quản lý nhóm như tạo nhóm, quản lý thành viên, phân quyền. Module này sử dụng dịch vụ lưu trữ **Group Persistent**.
 - **Chat Module:** Chịu trách nhiệm gửi/nhận tin nhắn, quản lý hội thoại, truy xuất lịch sử nhắn tin. Kết nối với **Message Persistent** và **Conversation Persistent** để lưu trữ; gửi notification khi có tin nhắn mới qua **Notification Module**.
 - **Announcement Module:** Xử lý việc gửi thông báo của Nhà trường, quản lý các thông báo hệ thống, lưu trữ tại **Announcement Persistent** và gọi **Notification Module** để gửi thông báo cho người dùng.
 - **Notification Module:** Nhận thông tin từ các module nghiệp vụ (chat, nhóm, thông báo), tạo và quản lý thông báo, lưu vào **Notification Persistent**, đồng thời trả lại trạng thái gửi về các module liên quan.



- **Tầng lưu trữ (Persistence Layer):**

- **Group Persistent:** Lưu trữ, truy xuất dữ liệu về nhóm.
- **Conversation Persistent, Message Persistent:** Lưu trữ hội thoại và tin nhắn hỗ trợ tìm kiếm, truy vấn nhanh.
- **Announcement Persistent:** Lưu các thông báo của hệ thống.
- **Notification Persistent:** Ghi nhận, cập nhật trạng thái gửi thông báo, phục vụ tra cứu lịch sử gửi/nhận.

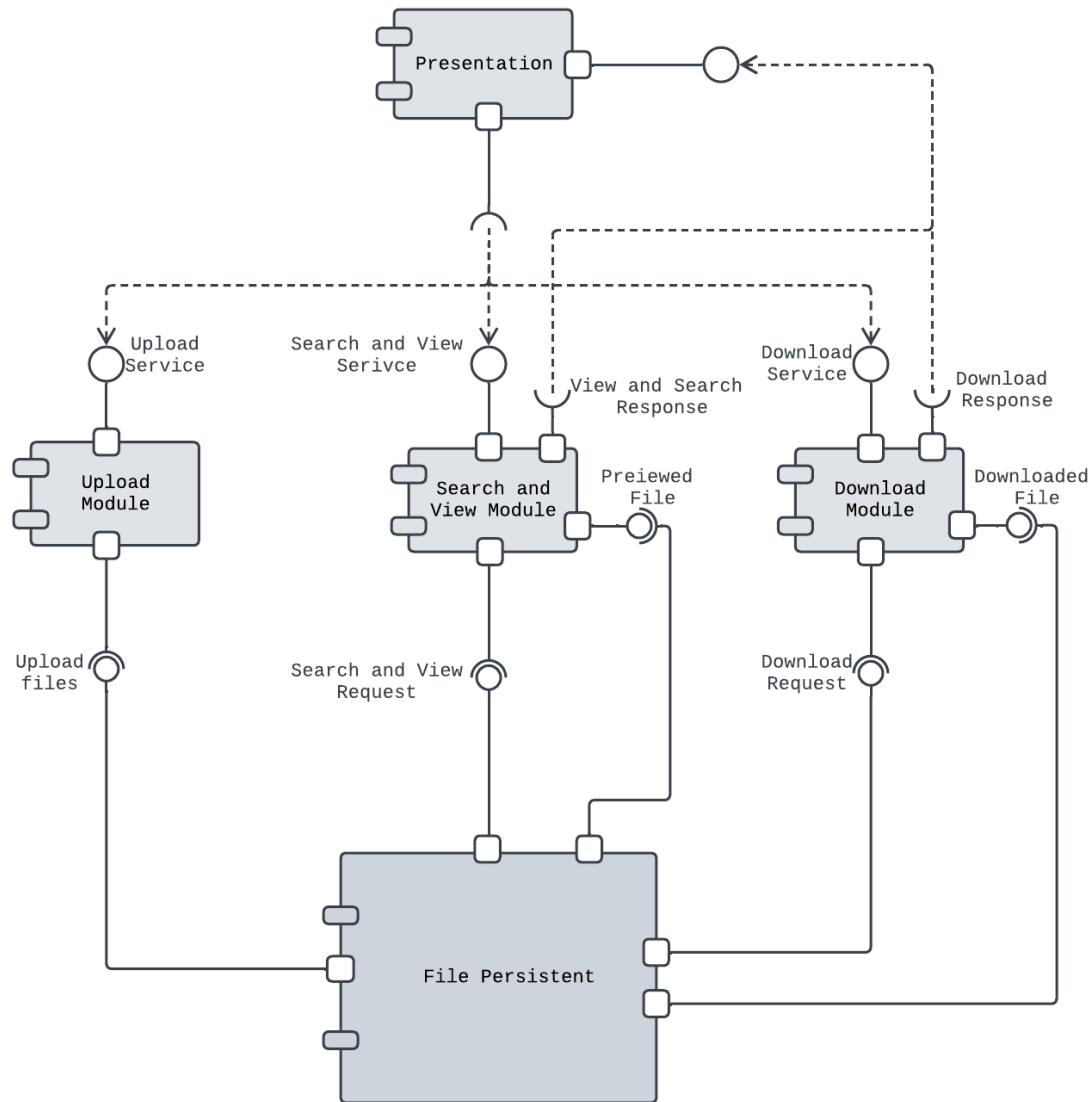
Luồng tương tác (Interfaces):

- Giao diện Presentation gửi request đến các module nghiệp vụ (**Group, Chat, Announcement**).
- Các module nghiệp vụ thực hiện logic, truy xuất/lưu trữ dữ liệu thông qua các thành phần tầng lưu trữ và đẩy sự kiện sang **Notification Module** khi cần thông báo cho người dùng.
- **Notification Module** ghi nhận trạng thái gửi, cập nhật vào **Notification Persistent**, đồng thời trả phản hồi lại nếu cần.

Sơ đồ phân tách trách nhiệm rõ ràng (separation of concerns), giảm phụ thuộc giữa các thành phần (loose coupling) và thuận lợi mở rộng hoặc bảo trì về sau.



8.8 Quản lý tài liệu



Hình 8.8: Component diagram cho use-case "Thông báo và nhắn tin"

Mô tả sơ đồ thành phần: Quản lý tài liệu

Sơ đồ thành phần hệ thống **Quản lý tài liệu** được phân chia làm ba lớp chức năng chính như sau:

- **Giao diện người dùng (UI):**

Presentation: Đóng vai trò giao tiếp với người dùng cuối, nhận thao tác từ phía người dùng và chuyển tiếp các yêu cầu yêu cầu (request) tới các module xử lý nghiệp vụ bên dưới.

- **Logic nghiệp vụ (Business Logic):**

- **Upload Module:** Thực hiện các thao tác kiểm tra và xử lý yêu cầu tải tài liệu từ phía người dùng.
- **Search and View Module:** Thực hiện truy xuất tài liệu từ yêu cầu của người dùng, sau đó trả về bản xem trước của tài liệu đó về phía người dùng, cũng như xử lý các lỗi khi xảy ra trong quá trình thao tác.
- **Download Module:** Thực hiện các truy xuất thông tin tài liệu để lấy được file tài liệu theo yêu cầu từ người dùng, sau đó tải file này về máy người dùng, và có xử lý các lỗi trong quá trình truyền file.

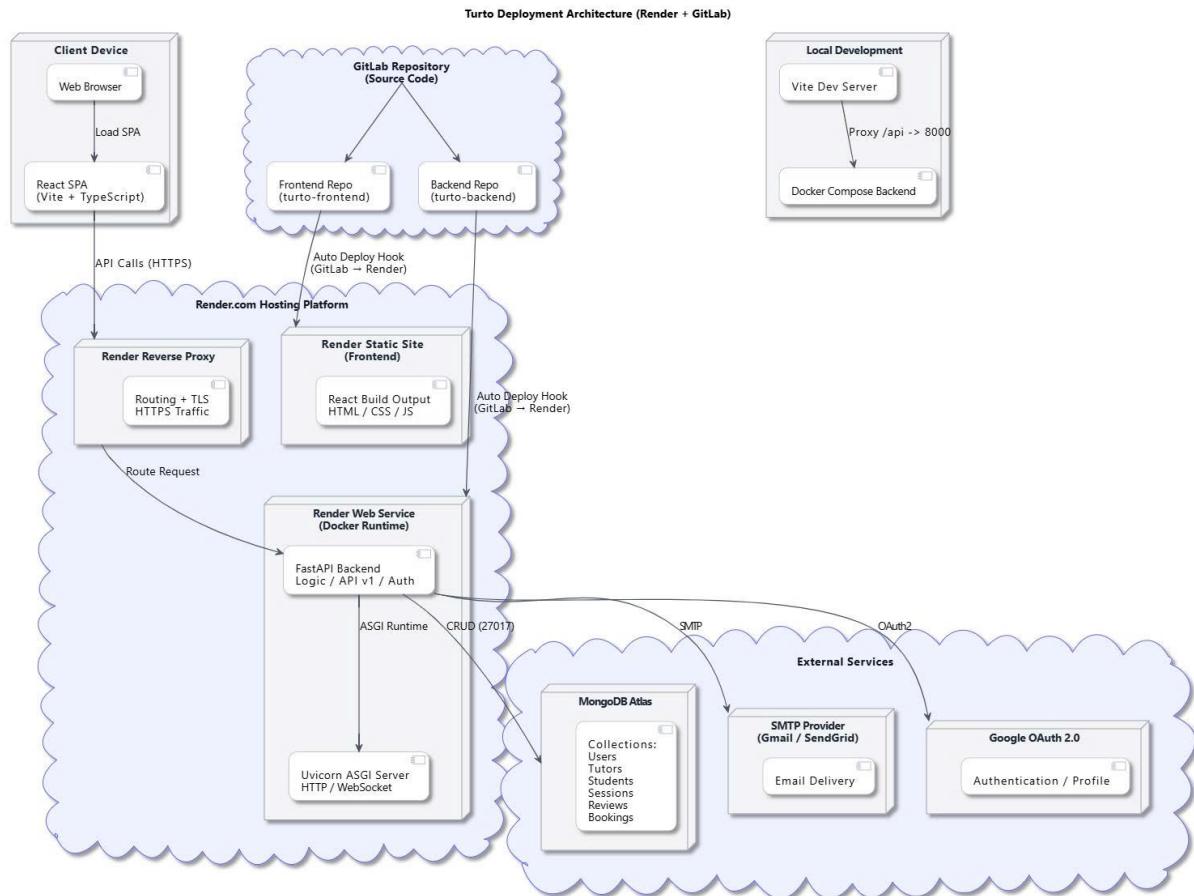
- **Tầng lưu trữ (Persistence Layer):**

File Persistent: Lưu trữ để truy xuất dữ liệu của tài liệu, phục vụ cho quá trình tìm kiếm, xem trước và tải xuống tài liệu của người dùng.

Luồng tương tác (Interfaces):

- Giao diện Presentation gửi request đến các module nghiệp vụ (Upload, Search and View, Download).
- Các module nghiệp vụ thực hiện logic, truy xuất/lưu trữ dữ liệu thông qua các thành phần tầng lưu trữ File Persistent.
- Module File Persistent phản hồi dữ liệu cần thiết cho các module nghiệp vụ Search and View, Download.
- Các module Search and View, Download này sau đó trả về kết quả xử lý (file tài liệu, bản xem trước) về Presentation để phản hồi người dùng.

9 Deployment View của hệ thống Turto



Hình 9.1: Sơ đồ triển khai hệ thống

9.1 Thiết bị Người dùng (Client Device)

Người dùng truy cập Turto thông qua trình duyệt web. Trình duyệt sẽ tải ứng dụng React Single Page Application (SPA), được xây dựng bằng Vite và TypeScript. Toàn bộ giao tiếp giữa SPA và backend diễn ra qua các API sử dụng giao thức HTTPS.

9.2 Kho mã nguồn GitLab và CI/CD

Hệ thống mã nguồn được quản lý trong hai repository tách biệt:

- **turto-frontend:** chứa mã nguồn React/Vite.
- **turto-backend:** chứa mã nguồn backend FastAPI.

Mỗi lần push code lên GitLab, một *Auto Deploy Hook* được kích hoạt để Render.com tự động build và triển khai (deploy) dịch vụ tương ứng.



9.3 Nền tảng Render.com (Môi trường Production)

9.3.1 Render Static Site (Frontend)

Render cung cấp dịch vụ lưu trữ tĩnh cho frontend. Kết quả build của React (HTML, CSS, JavaScript) được phục vụ trực tiếp cho người dùng.

9.3.2 Render Reverse Proxy

Reverse Proxy của Render đảm nhiệm:

- Định tuyến request từ client
- Xử lý bảo mật TLS/HTTPS
- Chuyển tiếp các request API từ SPA đến backend

9.3.3 Render Web Service (Backend)

Backend được triển khai dưới dạng một dịch vụ Docker (Dockerized Web Service). Bên trong service bao gồm:

- Ứng dụng FastAPI (xử lý logic, API v1, xác thực)
- Môi trường ASGI
- Máy chủ Uvicorn phục vụ HTTP và WebSocket

Backend kết nối đến MongoDB Atlas, SMTP Provider và Google OAuth để thực thi các chức năng hệ thống.

9.4 Các Dịch vụ Bên ngoài (External Services)

9.4.1 MongoDB Atlas

Toàn bộ dữ liệu của Turto được lưu trữ trên MongoDB Atlas. Các collection chính bao gồm:

- Users
- Tutors
- Students
- Sessions
- Reviews
- Bookings

9.4.2 Dịch vụ SMTP

SMTP Provider (Gmail hoặc SendGrid) được sử dụng để gửi email xác thực và các thông báo từ hệ thống.



9.4.3 Google OAuth 2.0

Hệ thống hỗ trợ đăng nhập bằng Google thông qua giao thức OAuth 2.0. Backend thực hiện trao đổi token và lấy thông tin người dùng để xác thực.

9.5 Môi trường Phát triển Cục bộ (Local Development)

9.5.1 Frontend

Ứng dụng React chạy bằng Vite Dev Server ở môi trường local. Các request tới đường dẫn /api được proxy sang backend chạy tại port 8000.

9.5.2 Backend

Backend chạy thông qua Docker Compose, mô phỏng gần giống môi trường production. Lập trình viên có thể chạy FastAPI và MongoDB bản local cho việc kiểm thử.

Tóm tắt

Kiến trúc triển khai của Turto được xây dựng theo mô hình cloud-first, sử dụng Render.com để lưu trữ frontend và chạy backend trong môi trường Docker. GitLab CI/CD đảm nhiệm việc tự động hóa triển khai. Các dịch vụ như MongoDB Atlas, SMTP và Google OAuth giúp hệ thống đảm bảo tính mở rộng, ổn định và bảo mật. Môi trường phát triển local được cấu hình tương tự production để giảm sai lệch khi triển khai thực tế.



10 Sử dụng Generative AI

10.1 Sử dụng AI trong phát triển use-case "Thông báo và nhắn tin"

Trong quá trình phát triển use-case "Thông báo và nhắn tin", em đã sử dụng Generative AI để hỗ trợ trong việc tạo nội dung và ý tưởng. Cụ thể:

- **Tạo ý tưởng sơ bộ cho use-case:** Sử dụng AI để đề xuất một vài tính năng chính và kịch bản có thể có trong use-case này để có một cái nhìn tổng quan hơn về những gì chính cần phát triển.
- **Chỉnh sửa và gợi ý hoàn thiện Activity Diagram:** Đưa ra sơ đồ về luồng hoạt động chính (phần Activity Diagram) và nhờ AI gợi ý một số chi tiết nhằm hoàn thiện hơn các bước trong luồng hoạt động.
- **Cải thiện Sequence Diagram:** Đưa ra Sequence Diagram để AI code lại bằng mã plantUML để tối ưu và dễ đọc hơn.
- **Gợi ý phát triển Class Diagram:** Trình bày ý tưởng ban đầu về các lớp và mối quan hệ giữa chúng, sau đó nhờ AI đề xuất các cải tiến để làm rõ hơn cấu trúc dữ liệu và các thuộc tính cần thiết.
- **Gợi ý phân cấp Development View:** Sử dụng AI để đề xuất cách tổ chức các module và thành phần trong hệ thống Component Diagram.



11 Bảng phân công công việc

STT	Họ và Tên	MSSV	Công việc được giao	Mức độ hoàn thành
1	Huỳnh Trung Kiên	2311730	Sơ đồ use-case hệ thống tổng quát, User story Use-case: Thông báo và tin nhắn	100%
2	Trần Đăng Khoa	2311645	Use-case: Quản lý truy cập và xác thực Giao diện người dùng	100%
3	Nguyễn Lê Khôi Nguyên	2312366	Use-case: Đăng ký chương trình	100%
4	Võ Đức Cao Thượng	2313405	Use-case: Phân tích và báo cáo	100%
5	Lương Trí Thịnh	2314057	Use-case: Đánh giá	100%
6	Nguyễn Hà Viết Thông	2313339	Use-case: Thiết lập lịch trình cho sinh viên	100%
7	Nguyễn Đức Dũng	2210582	Use-case: Tạo chương trình học	100%
8	Vũ Anh Tuấn	2313771	Use-case: Quản lý tài liệu	100%