

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 2



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

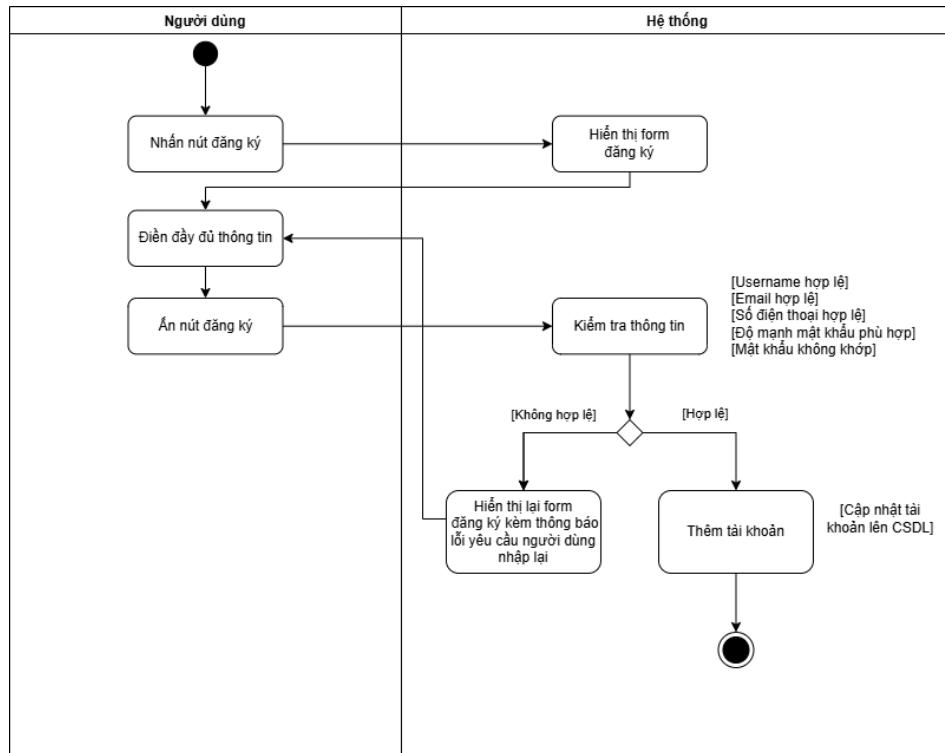
<b>1 Activity Diagram</b>	<b>4</b>
1.1 Quản lý truy cập và xác thực . . . . .	4
1.2 Đăng ký chương trình . . . . .	7
1.3 Tạo chương trình học . . . . .	9
1.4 Thiết lập lịch trình cho sinh viên . . . . .	12
1.5 Đánh giá . . . . .	15
1.6 Phân tích và báo cáo . . . . .	17
1.7 Thông báo và nhắn tin . . . . .	19
1.8 Quản lý tài liệu . . . . .	24
<b>2 Sequence Diagrams</b>	<b>27</b>
2.1 Quản lý truy cập và xác thực . . . . .	27
2.2 Đăng ký chương trình . . . . .	32
2.3 Tạo chương trình học . . . . .	33
2.4 Thiết lập lịch trình cho sinh viên . . . . .	36
2.5 Đánh giá . . . . .	39
2.6 Phân tích và báo cáo . . . . .	43
2.7 Thông báo và nhắn tin . . . . .	45
2.8 Quản lý tài liệu . . . . .	50
<b>3 Class Diagram</b>	<b>56</b>
3.1 Quản lý truy cập và xác thực . . . . .	57
3.2 Đăng ký chương trình . . . . .	58
3.3 Tạo chương trình học . . . . .	59
3.4 Thiết lập lịch trình cho sinh viên . . . . .	60
3.5 Đánh giá . . . . .	61
3.6 Phân tích và báo cáo . . . . .	62
3.7 Thông báo và nhắn tin . . . . .	63
3.8 Quản lý tài liệu . . . . .	64
<b>4 Development view</b>	<b>66</b>
4.1 Quản lý truy cập và xác thực . . . . .	66
4.2 Đăng ký chương trình . . . . .	67
4.3 Tạo chương trình học . . . . .	68
4.4 Thiết lập lịch trình cho sinh viên . . . . .	69
4.5 Đánh giá . . . . .	70
4.6 Phân tích và báo cáo . . . . .	71
4.7 Thông báo và nhắn tin . . . . .	72
4.8 Quản lý tài liệu . . . . .	74
<b>5 Bảng phân công công việc</b>	<b>75</b>

## Revision History

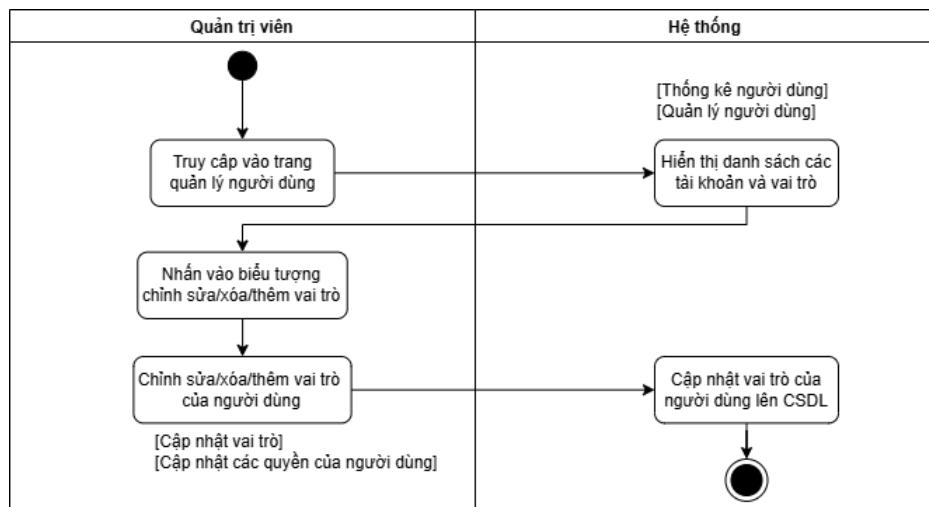
Revision	Date	Author	Description
1.0	9/2025	Trần Đăng Khoa	Use-case diagram + use-case scenario: Quản lý xác thực và đăng nhập
1.0	9/2025	Nguyễn Lê Khôi Nguyên	Use-case diagram + use-case scenario: Đăng ký chương trình học
1.0	9/2025	Nguyễn Đức Dũng	Use-case diagram + use-case scenario: Tạo chương trình 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

## 1 Activity Diagram

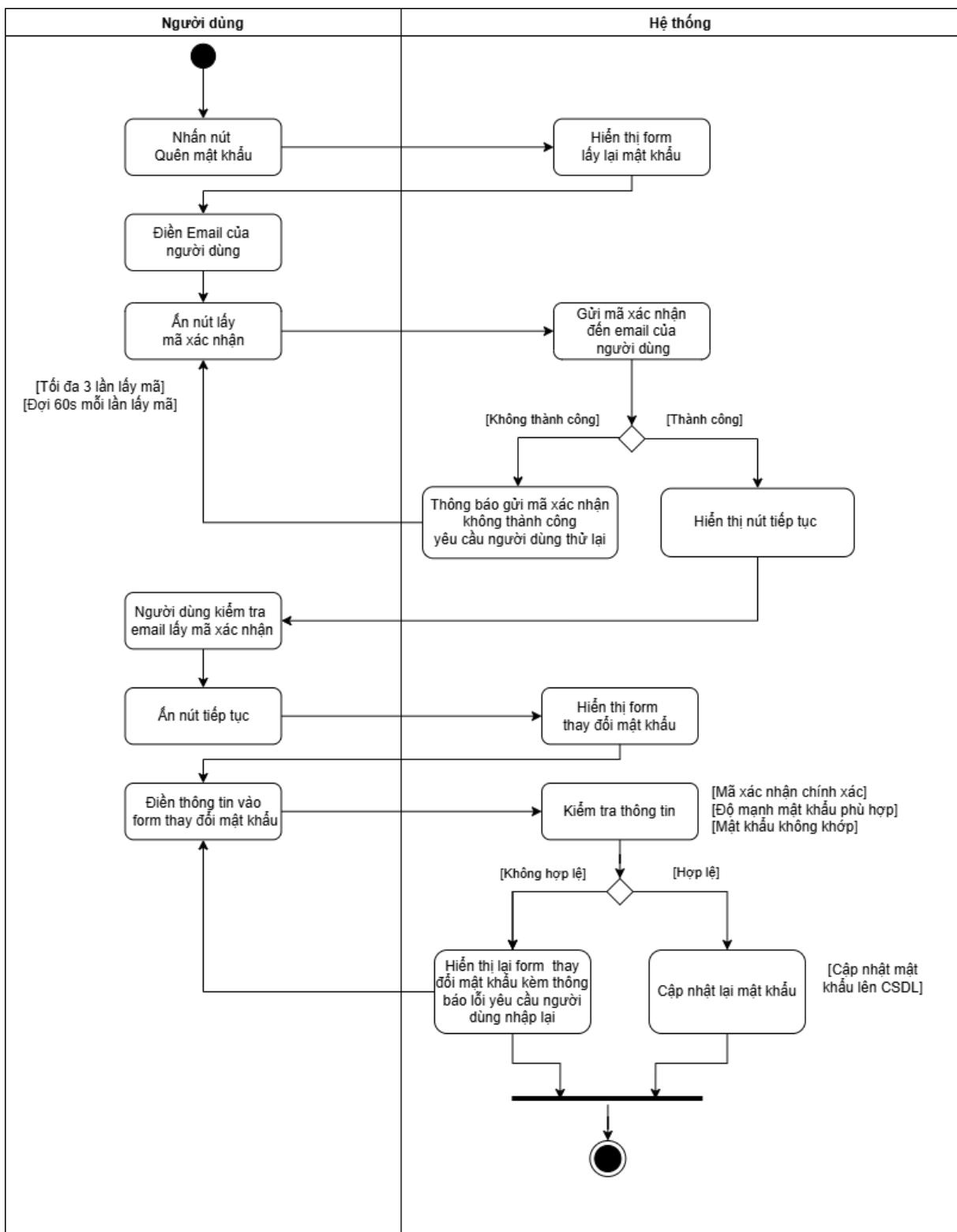
### 1.1 Quản lý truy cập và xác thực



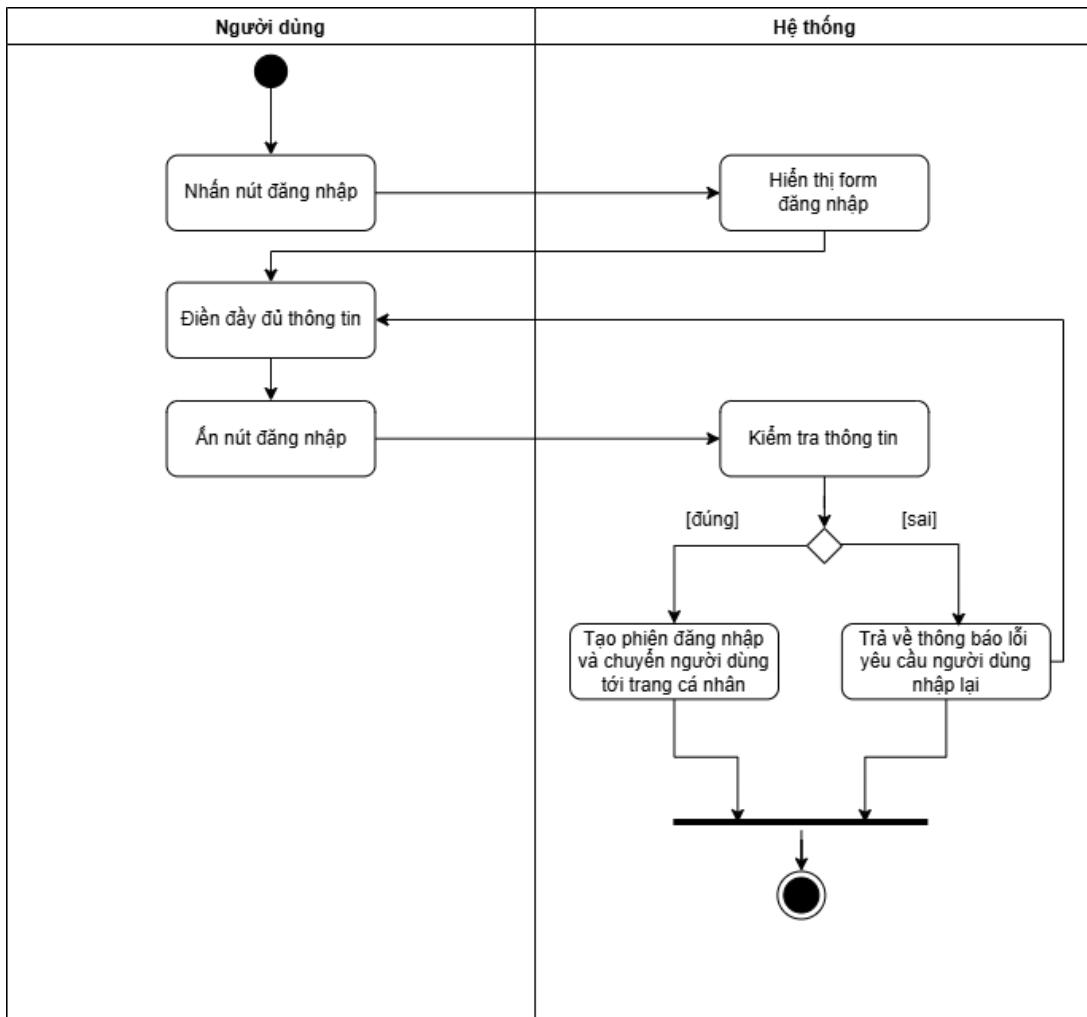
Hình 1.1: Sơ đồ hoạt động cho use-case "Đăng ký"



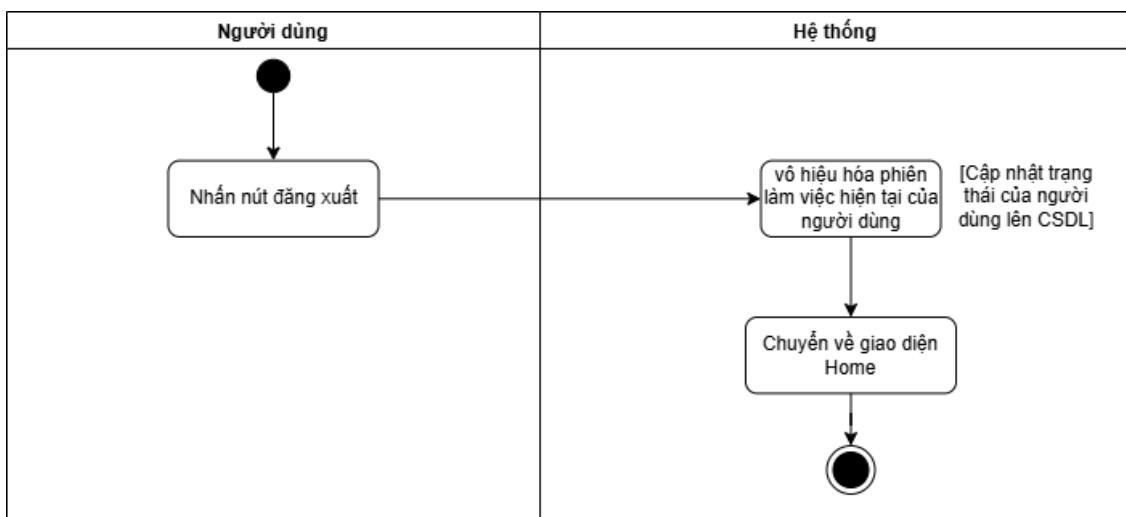
Hình 1.2: Sơ đồ hoạt động cho use-case "Quản lý tài khoản"



Hình 1.3: Sơ đồ hoạt động cho use-case "Đặt lại mật khẩu"

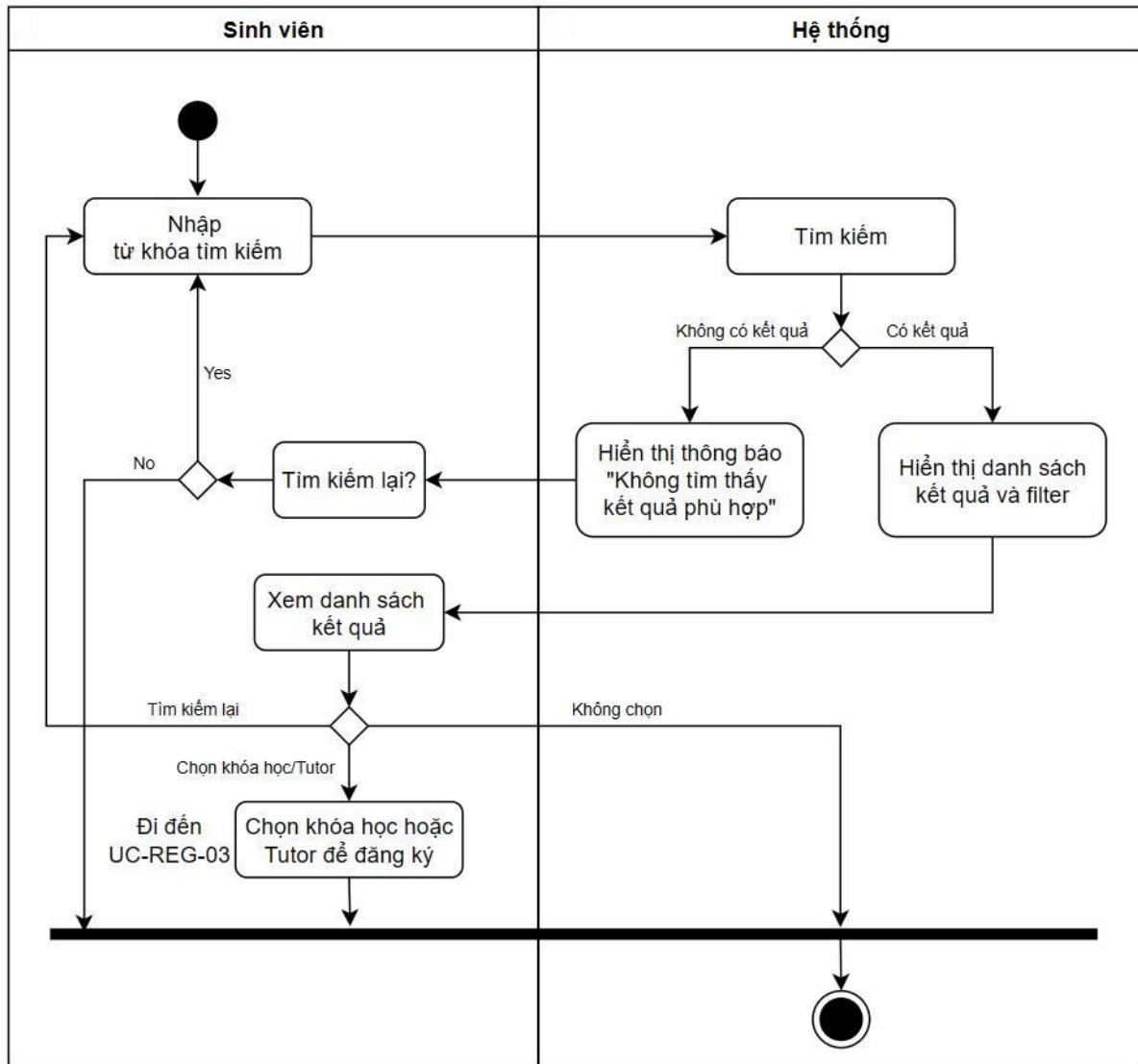


Hình 1.4: Sơ đồ hoạt động cho use-case "Đăng nhập"

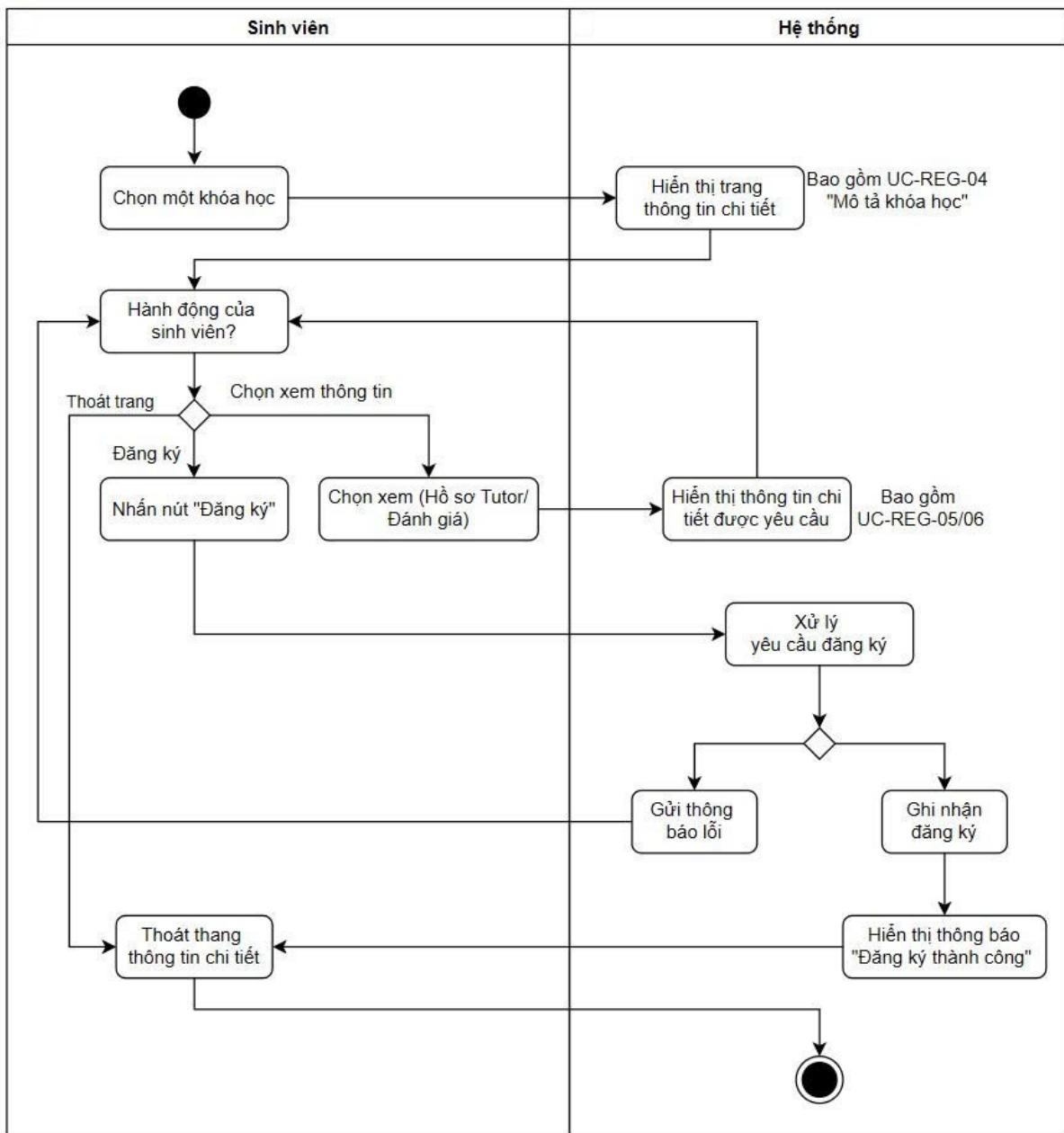


Hình 1.5: Sơ đồ hoạt động cho use-case "Đăng xuất"

## 1.2 Đăng ký chương trình

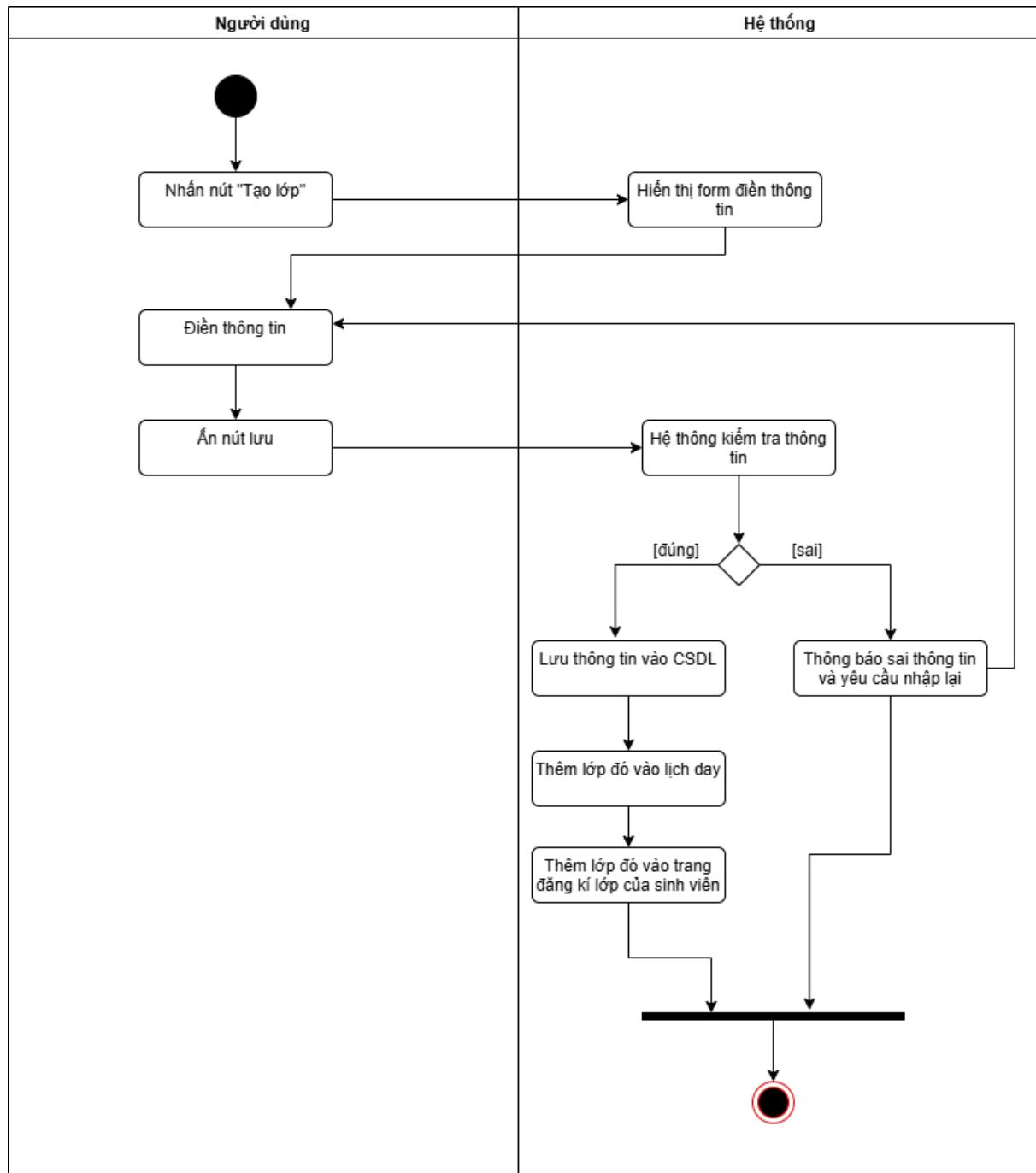


Hình 1.6: Sơ đồ hoạt động cho use-case "Tìm kiếm chương trình"

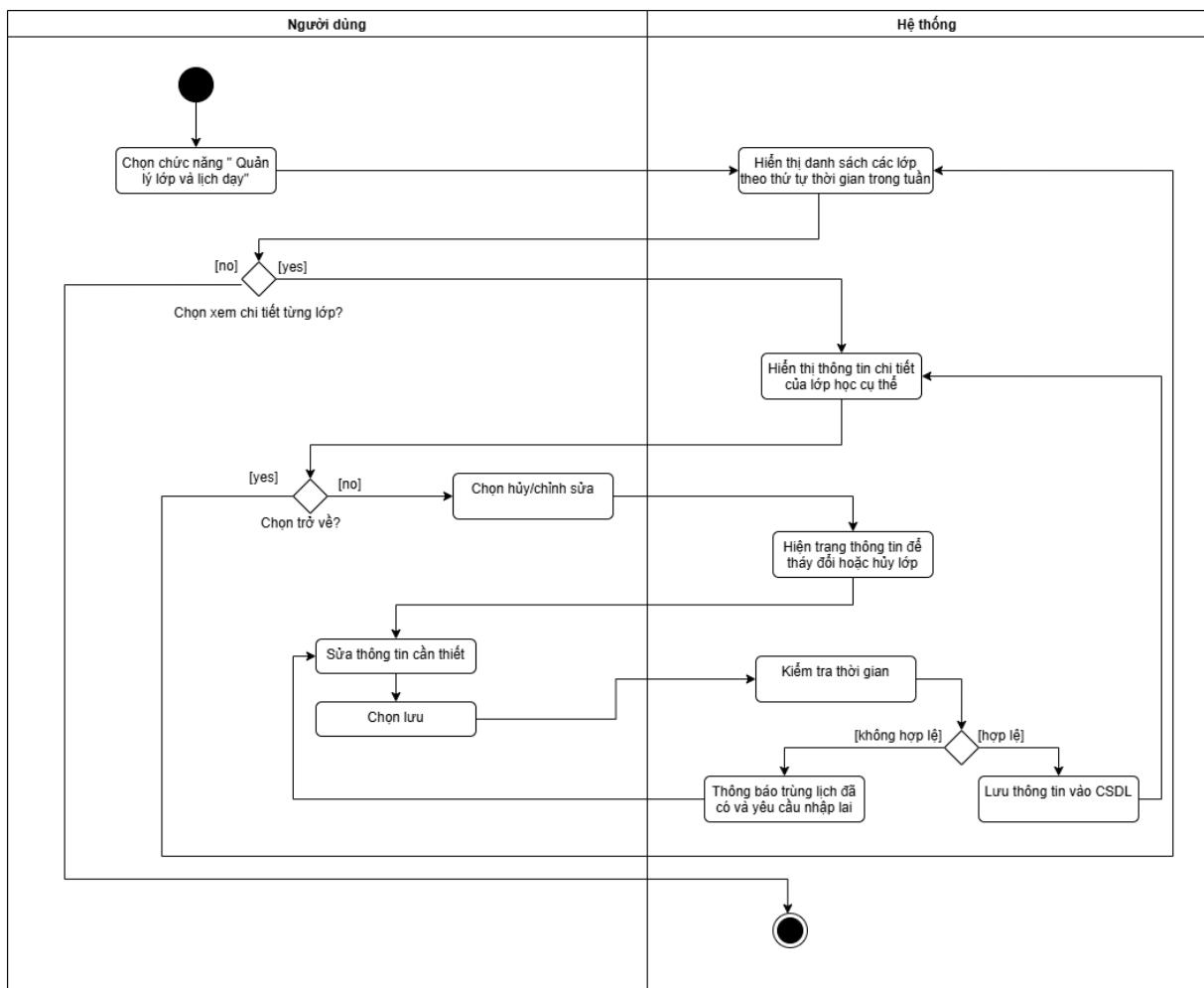


Hình 1.7: Sơ đồ hoạt động cho use-case "Đăng ký chương trình"

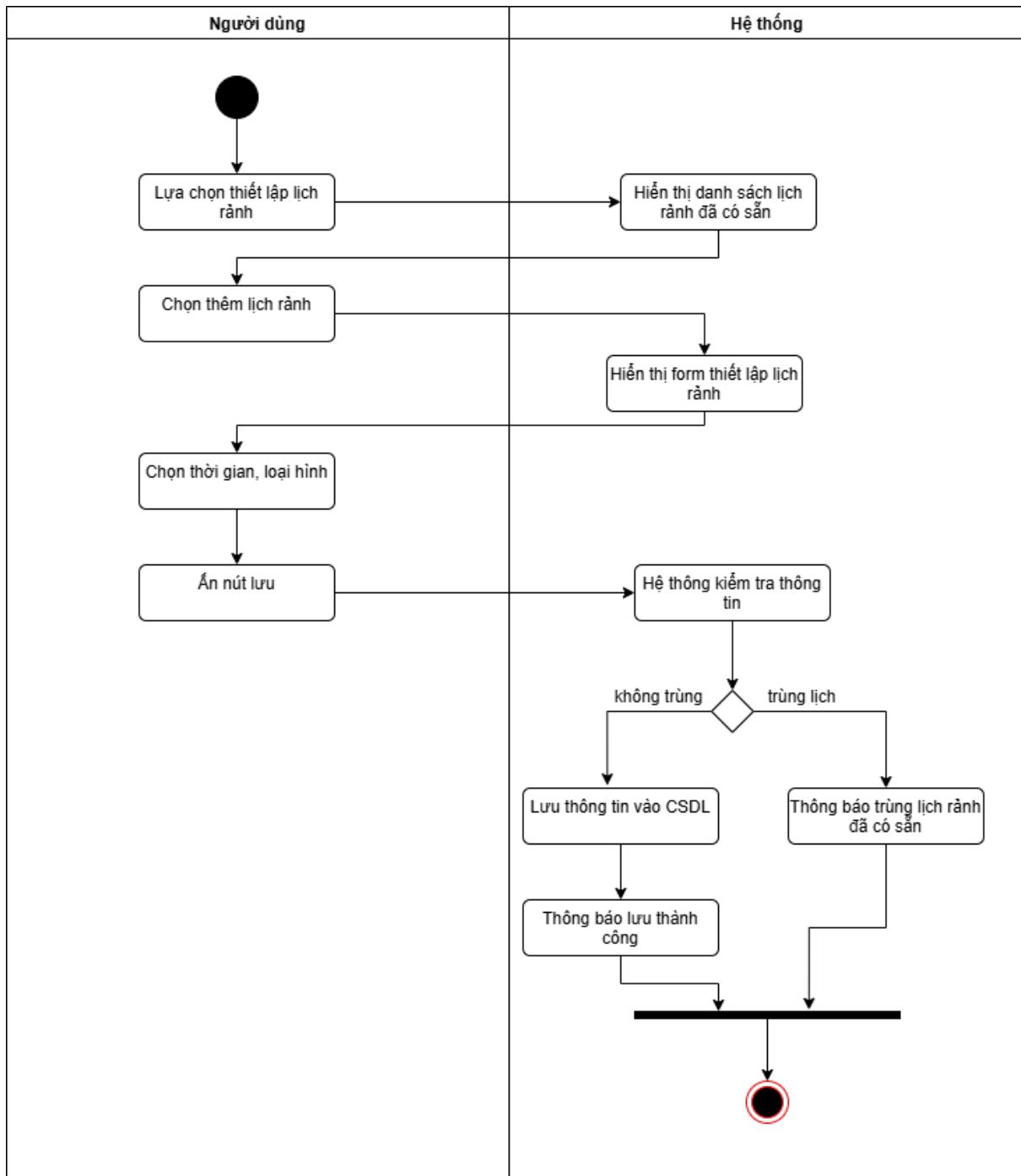
### 1.3 Tạo chương trình học



Hình 1.8: Sơ đồ use-case cho chức năng "Tạo lớp học"

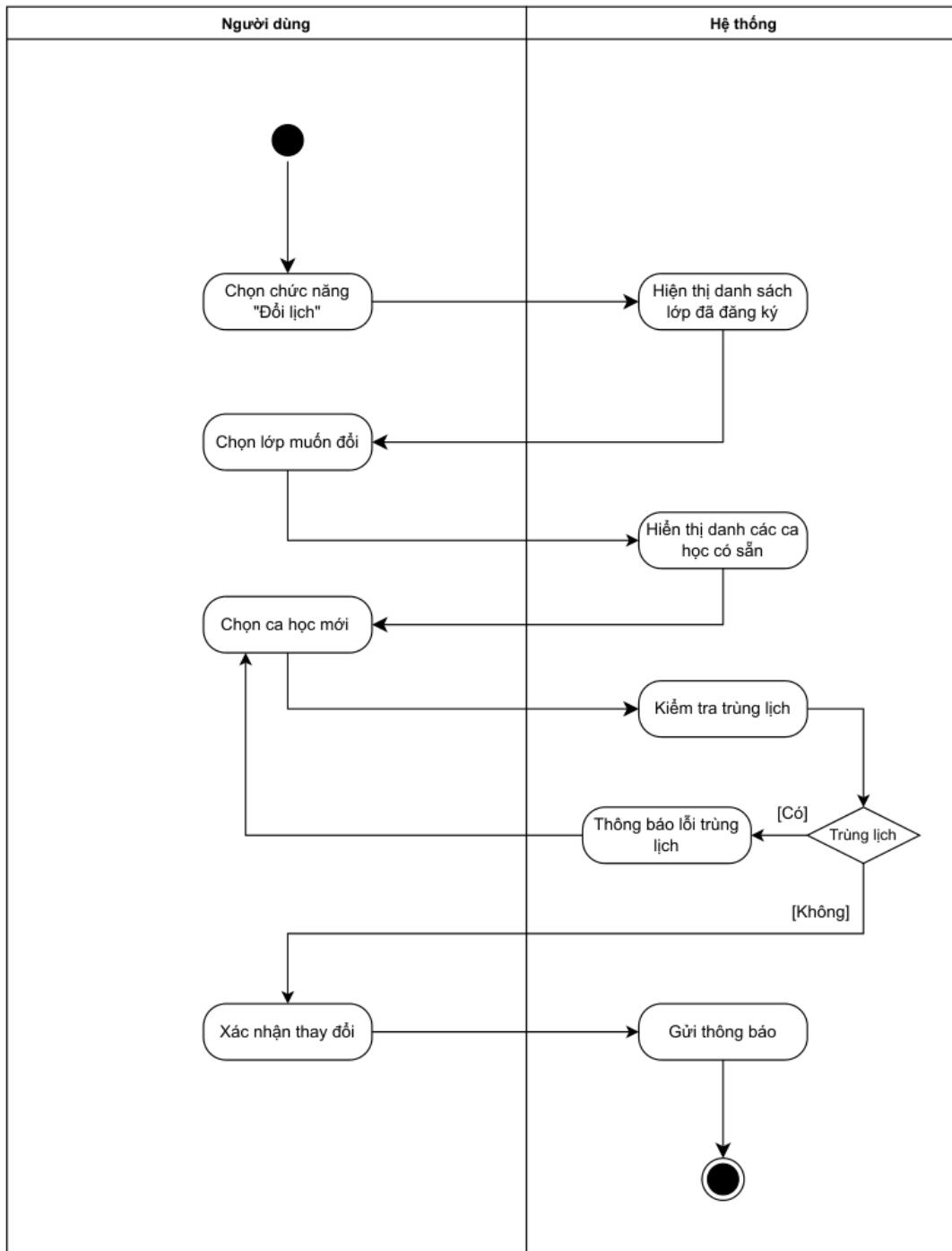


Hình 1.9: Sơ đồ use-case cho chức năng "Quản lí lớp học"

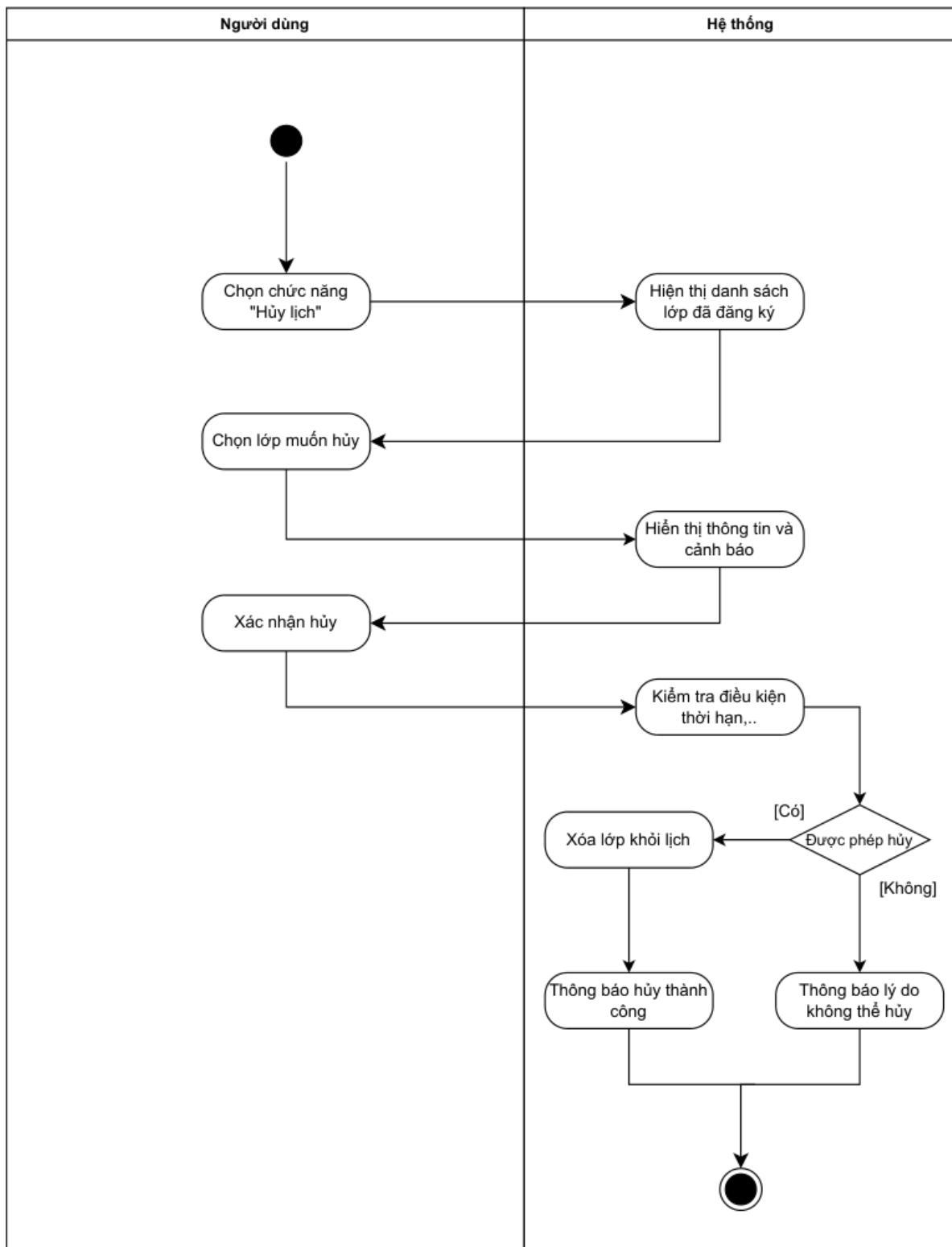


Hình 1.10: Sơ đồ use-case cho chức năng "Lịch rành"

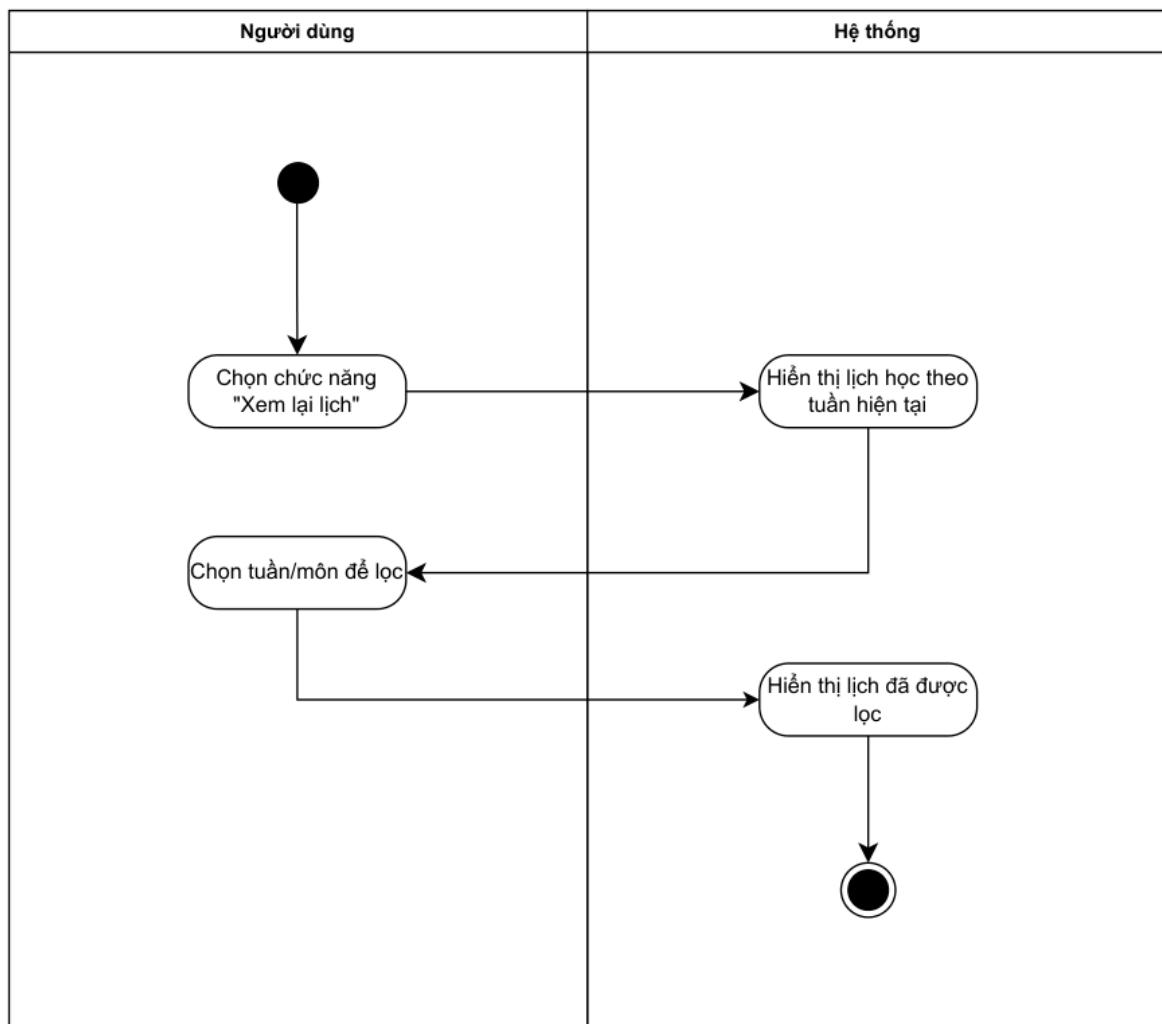
#### 1.4 Thiết lập lịch trình cho sinh viên



Hình 1.11: Sơ đồ hoạt động cho use-case "Đổi lịch học" cho sinh viên

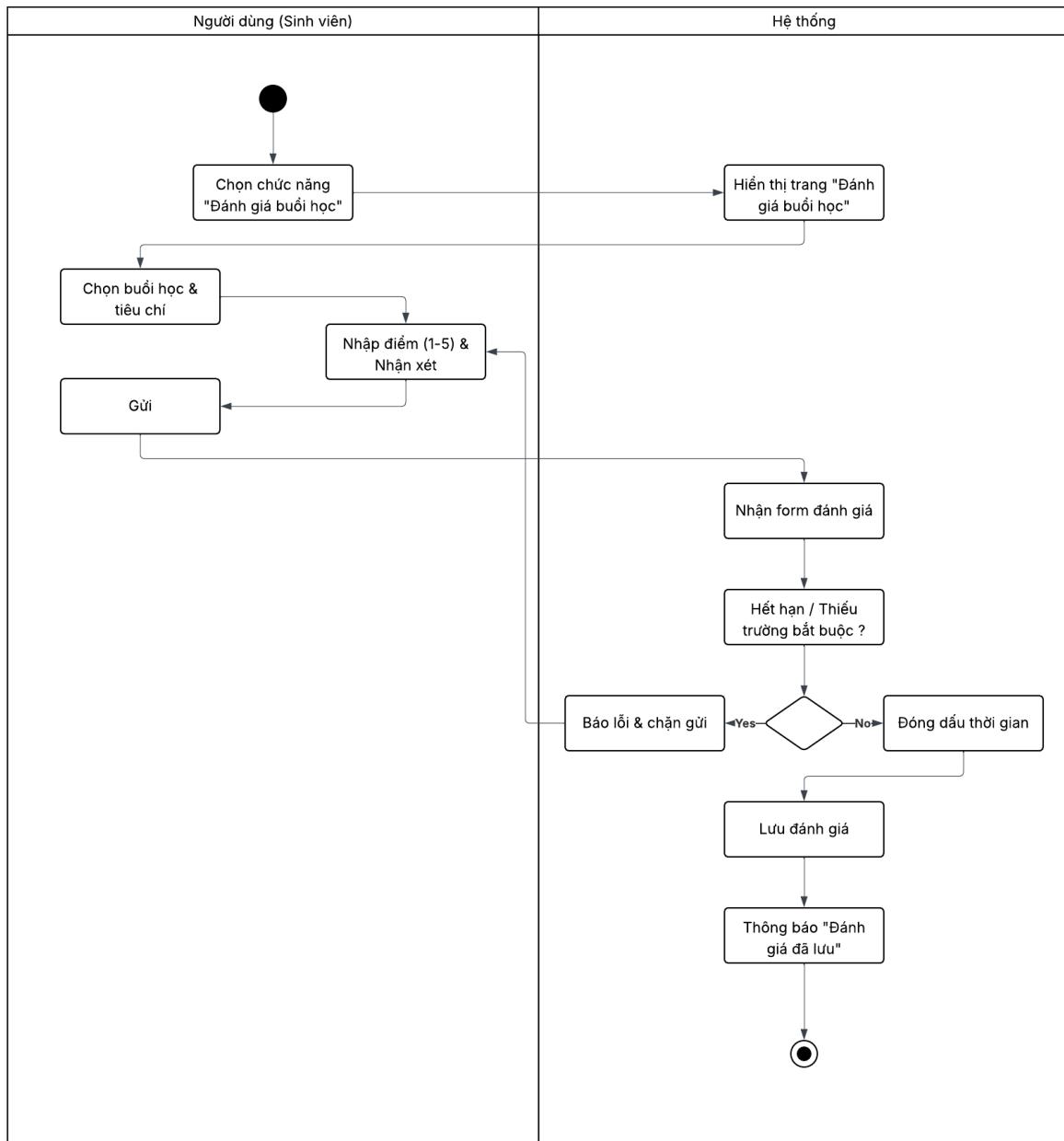


Hình 1.12: Sơ đồ hoạt động cho use-case "Hủy lịch học" cho sinh viên



Hình 1.13: Sơ đồ hoạt động cho use-case "Xem lại lịch học" cho sinh viên

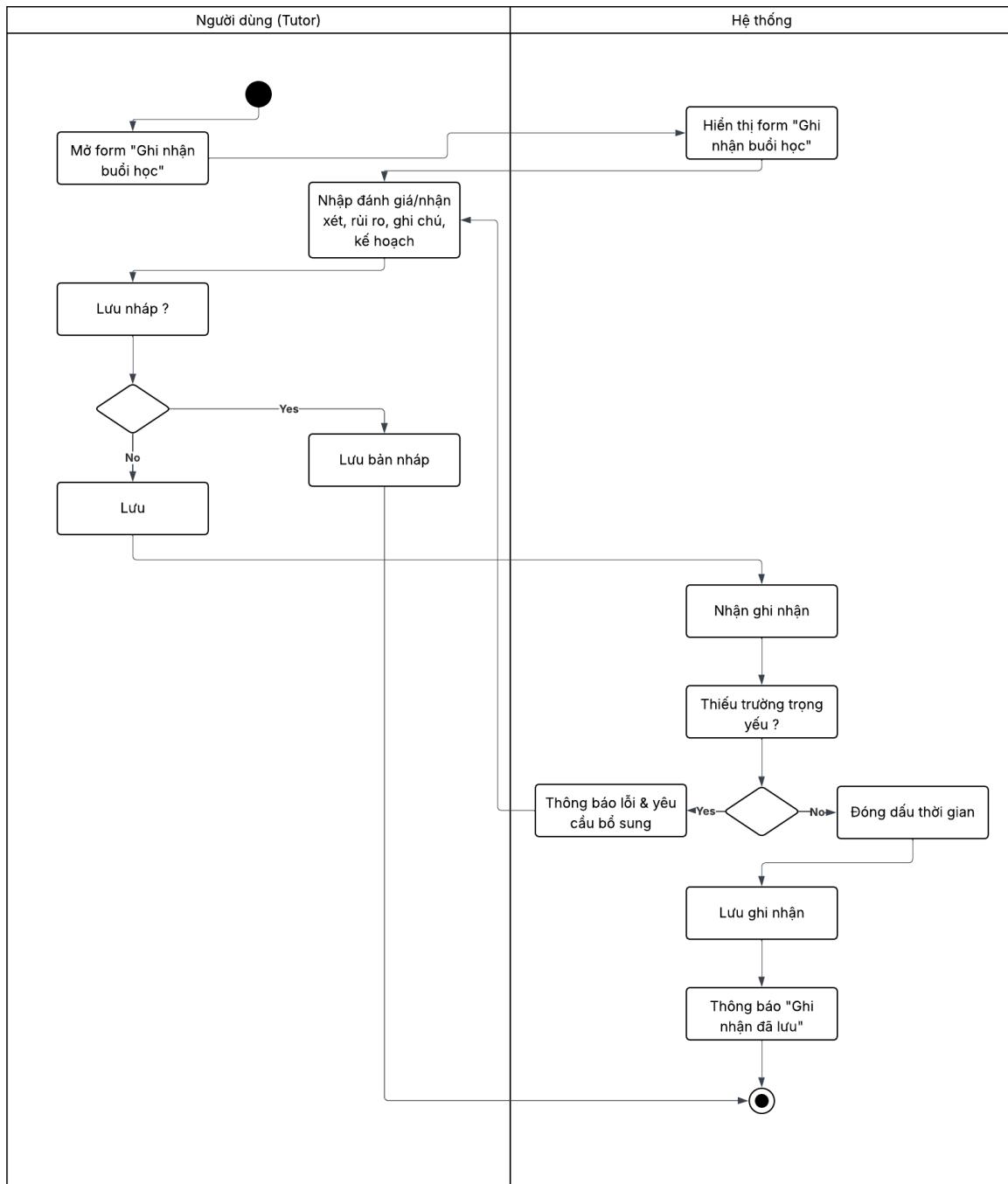
## 1.5 Đánh giá



Hình 1.14: 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.



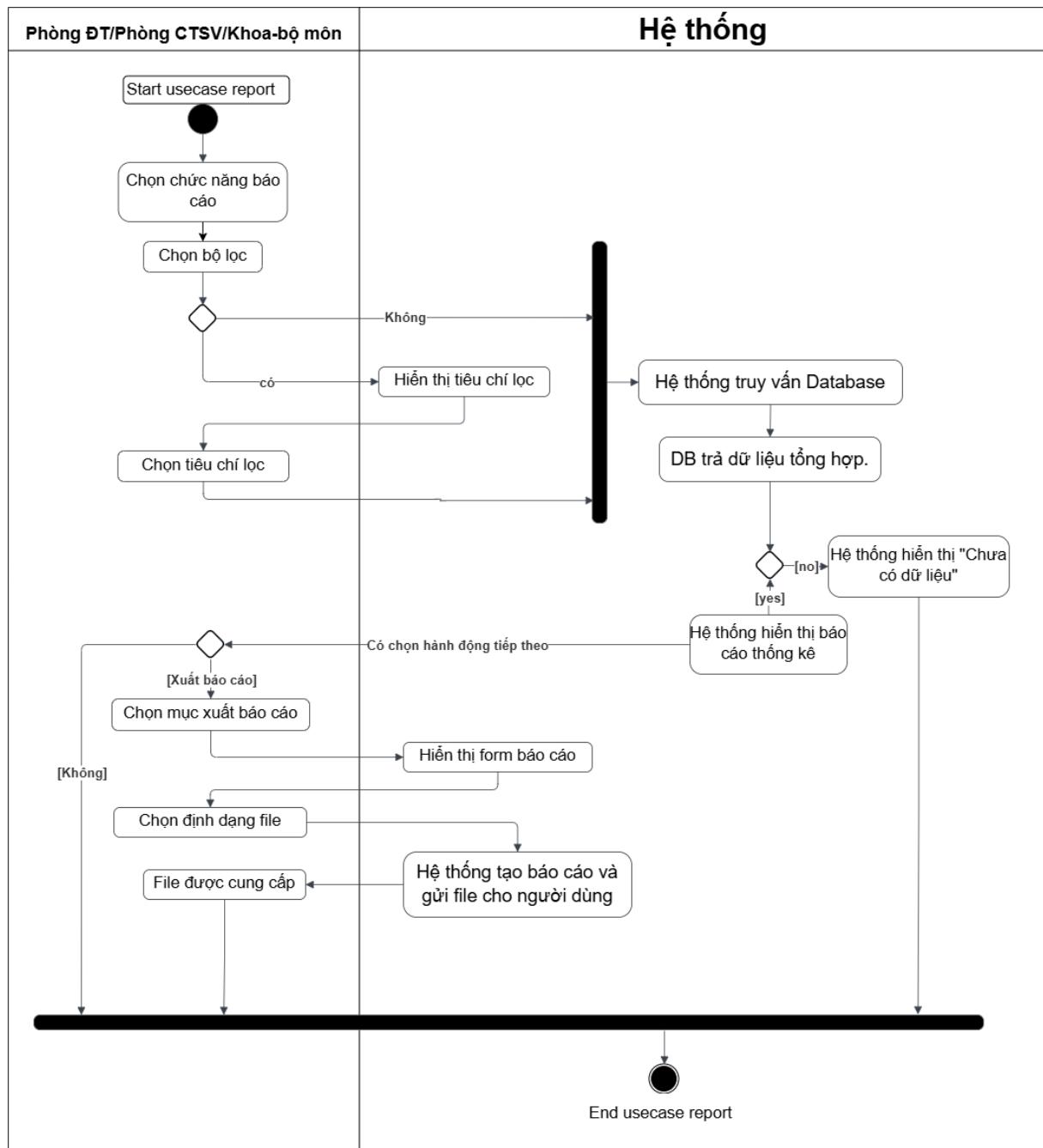
Hình 1.15: Sơ đồ hoạt động cho use-case "Ghi nhận tiến bộ sinh viên" của Tutor

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

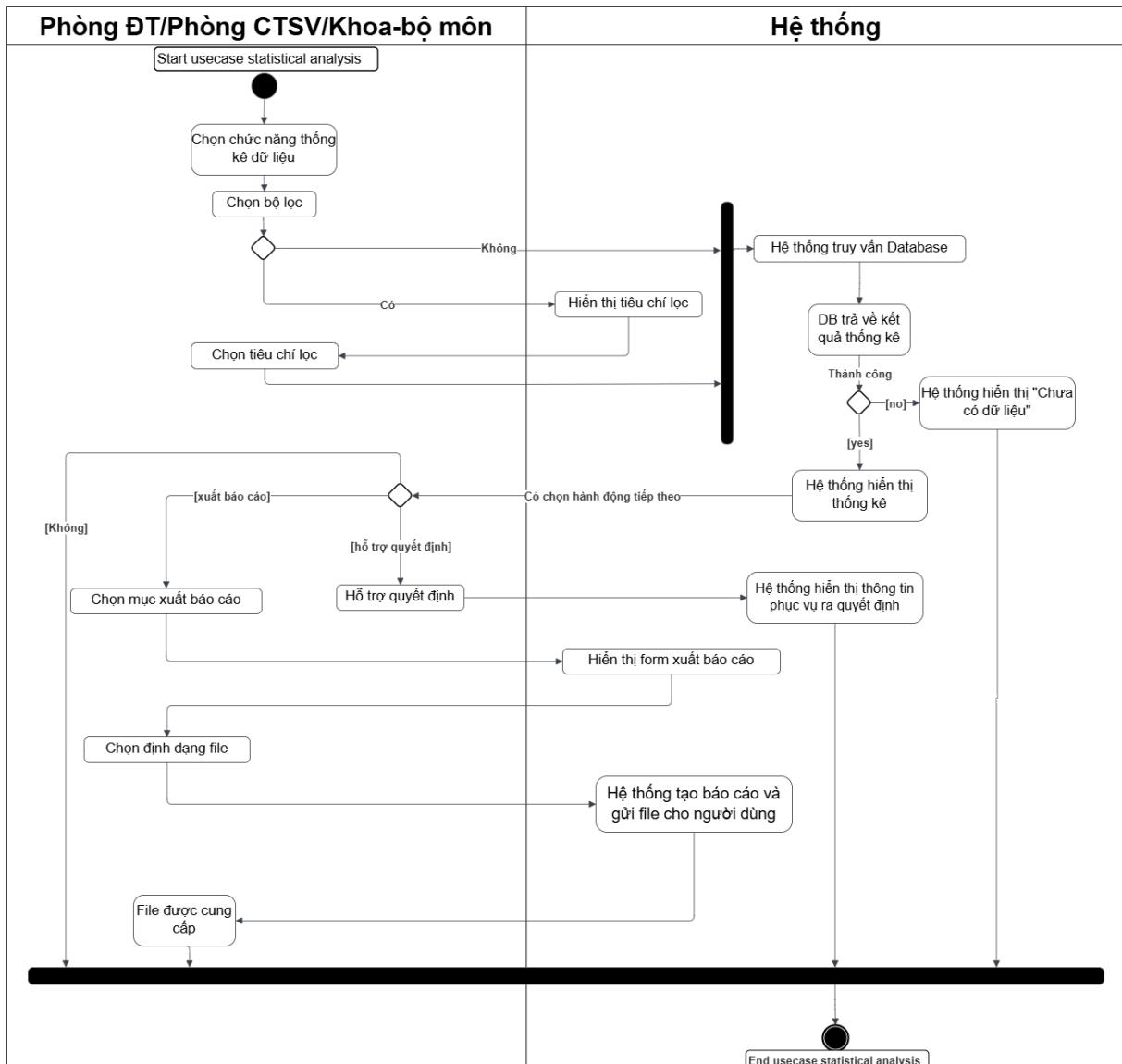
Luồng chuẩn: Tutor mở form “Ghi nhận buổi học” → nhập đánh giá/nhận xét, rủi ro, ghi chú, kế hoạch → lưu bản nháp hoặc lưu chính thức. Nếu thiếu trường trọng yếu, hệ thống báo lỗi; khi lưu thành công, kích hoạt cập nhật tổng hợp.



## 1.6 Phân tích và báo cáo

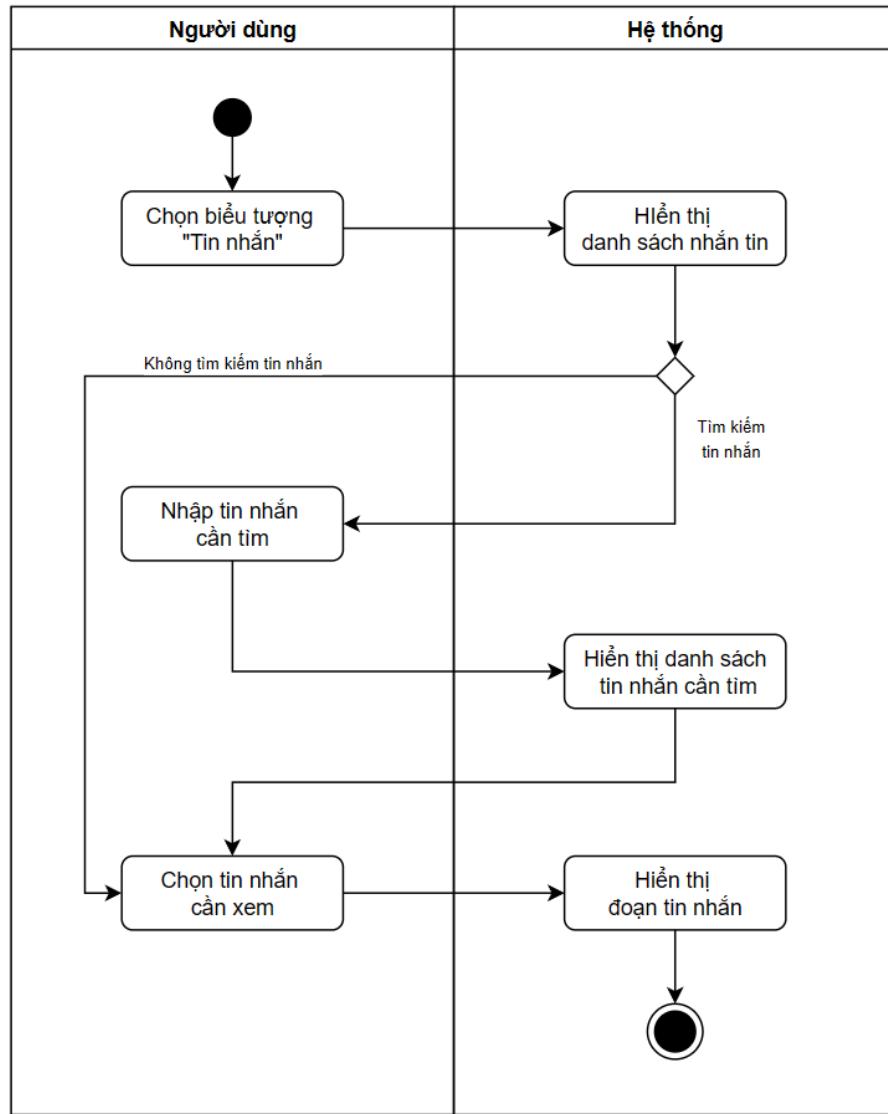


Hình 1.16: Sơ đồ hoạt động cho use-case "Phân tích và báo cáo"



Hình 1.17: Sơ đồ hoạt động cho use-case "Tạo báo cáo"

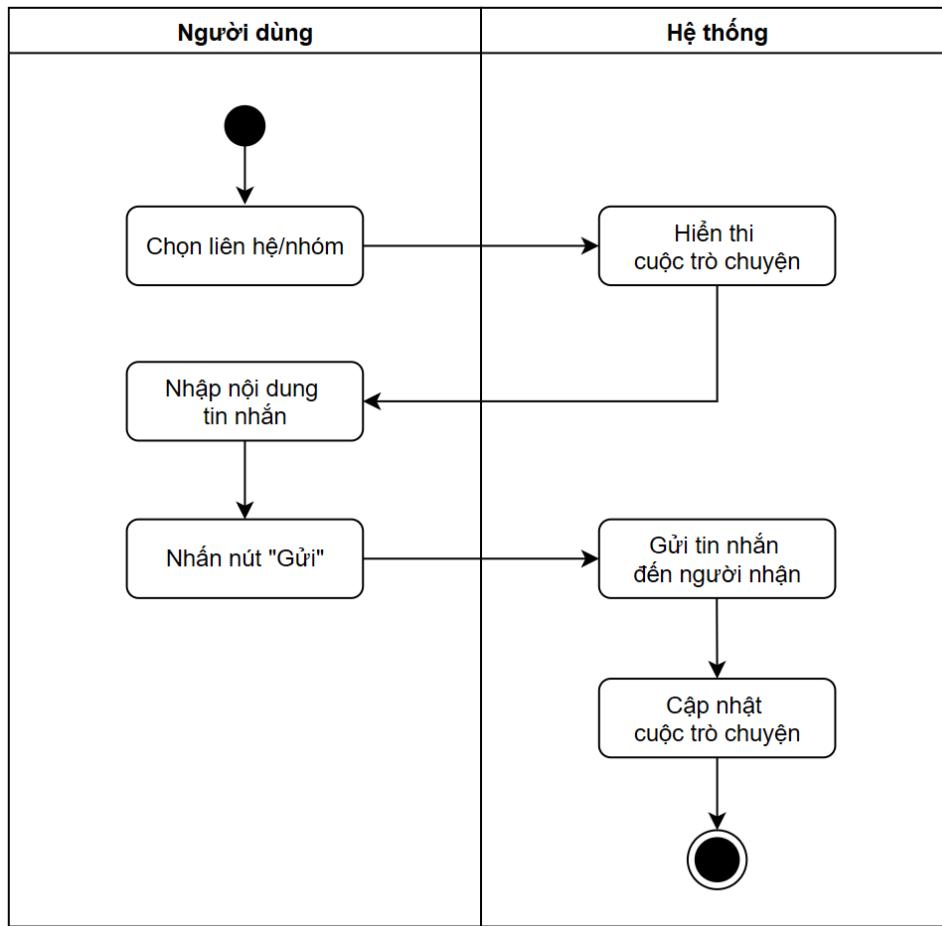
## 1.7 Thông báo và nhắn tin



Hình 1.18: Sơ đồ hoạt động cho use-case "Xem thông báo và tin nhắn"

### Mô tả hoạt động Activity Diagram: Xem thông báo và tin nhắn

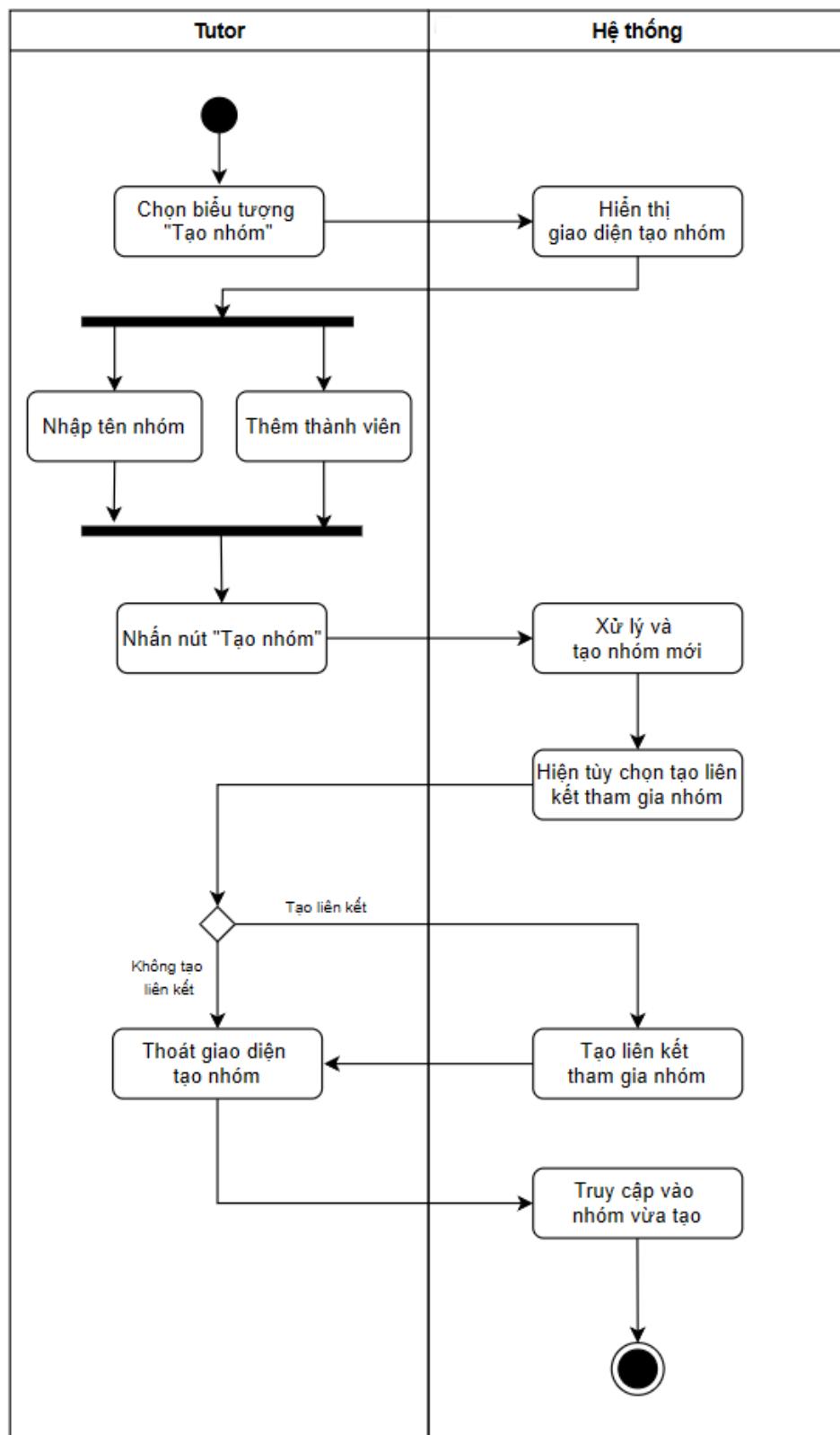
1. **Người dùng** chọn biểu tượng "Tin nhắn" trên giao diện ứng dụng.
2. **Hệ thống** hiển thị danh sách các cuộc trò chuyện (nhắn tin) hiện có.
3. Nếu **người dùng** muốn tìm tin nhắn, bấm vào thanh tìm kiếm và nhập từ khóa cần tìm.
4. **Hệ thống** truy vấn, trả về danh sách các tin nhắn phù hợp với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



Hình 1.19: Sơ đồ hoạt động cho use-case "Gửi tin nhắn"

### Mô tả hoạt động Activity Diagram: Gửi tin nhắn

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

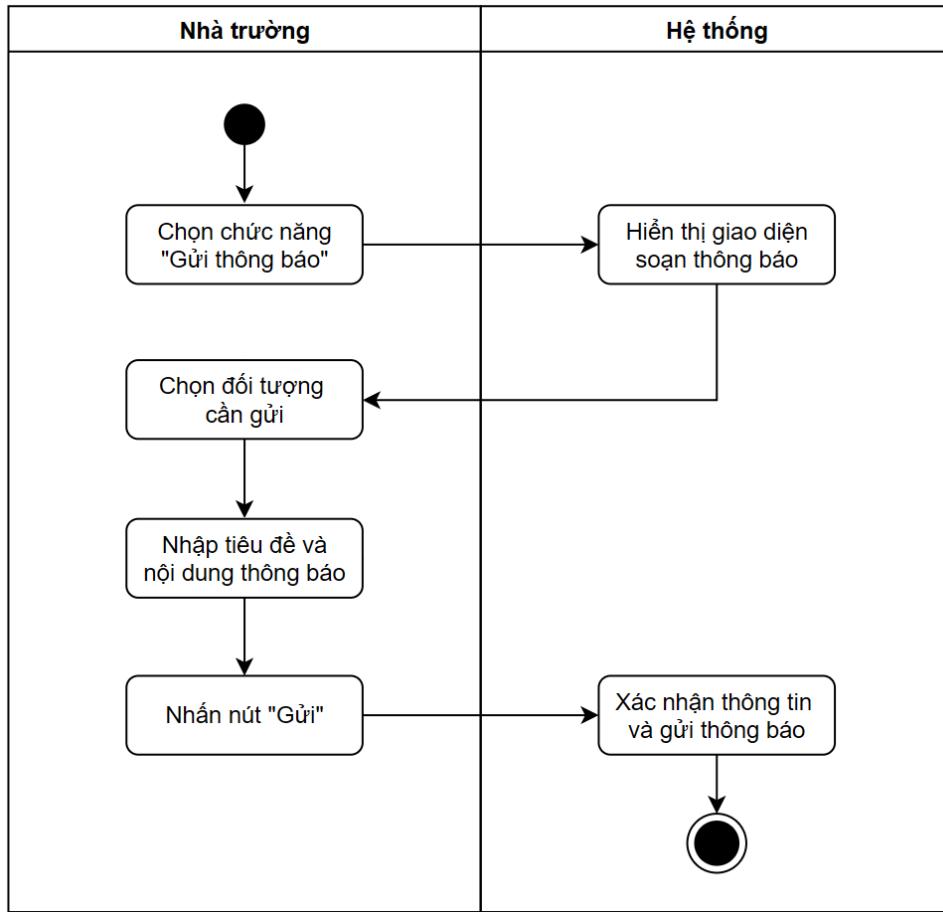


Hình 1.20: Sơ đồ hoạt động cho use-case "Tạo nhóm"



### Mô tả hoạt động Activity Diagram: Tạo nhóm

1. **Tutor** chọn biểu tượng "Tạo nhóm" trên ứng dụng.
2. **Hệ thống** hiển thị giao diện cho phép nhập thông tin nhóm.
3. **Tutor** thực hiện hai thao tác song song: nhập tên nhóm và thêm thành viên.
4. Sau khi hoàn tất, **Tutor** nhấn nút "Tạo nhóm".
5. **Hệ thống** kiểm tra, xử lý và tạo nhóm mới trong hệ thống.
6. **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.

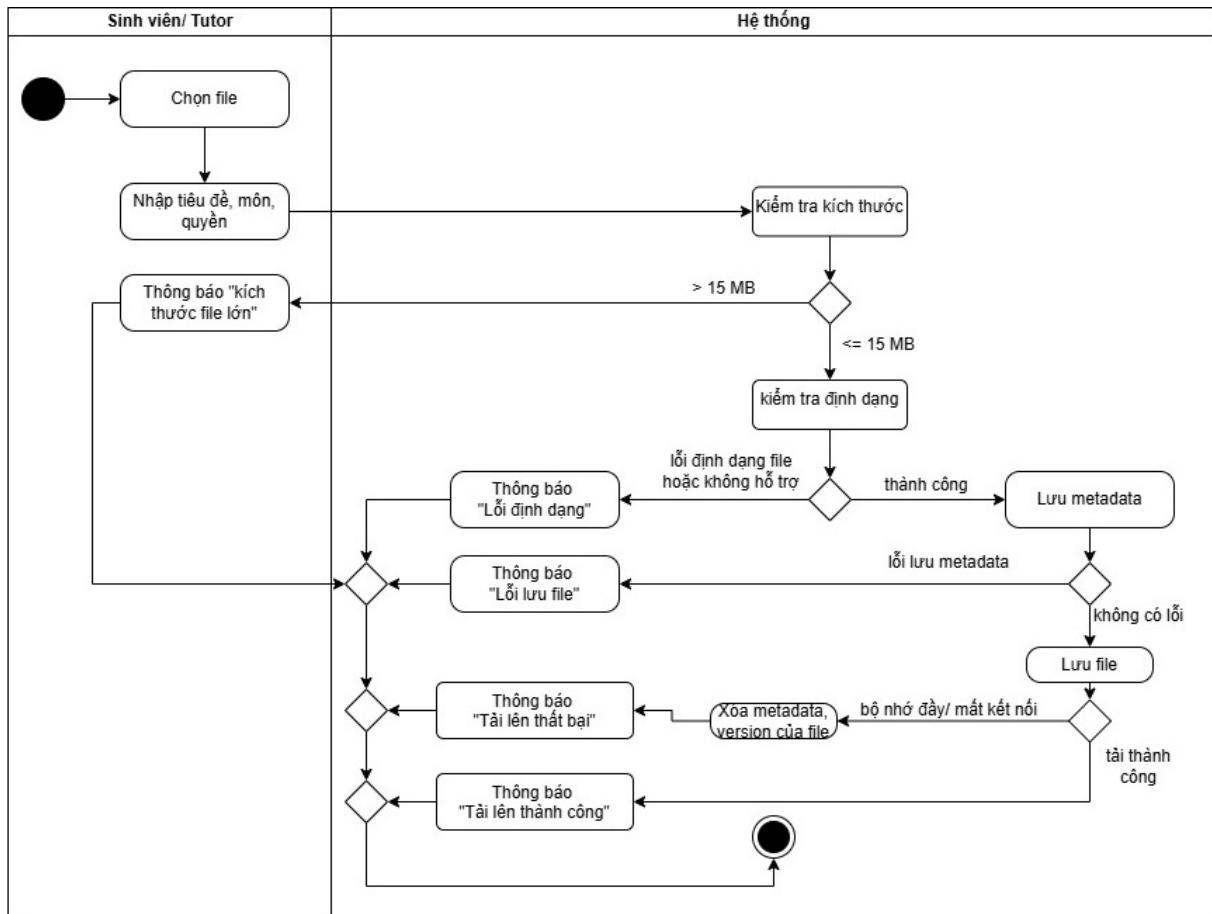


Hình 1.21: Sơ đồ hoạt động cho use-case "Gửi thông báo"

### Mô tả hoạt động Activity Diagram: Gửi thông báo

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

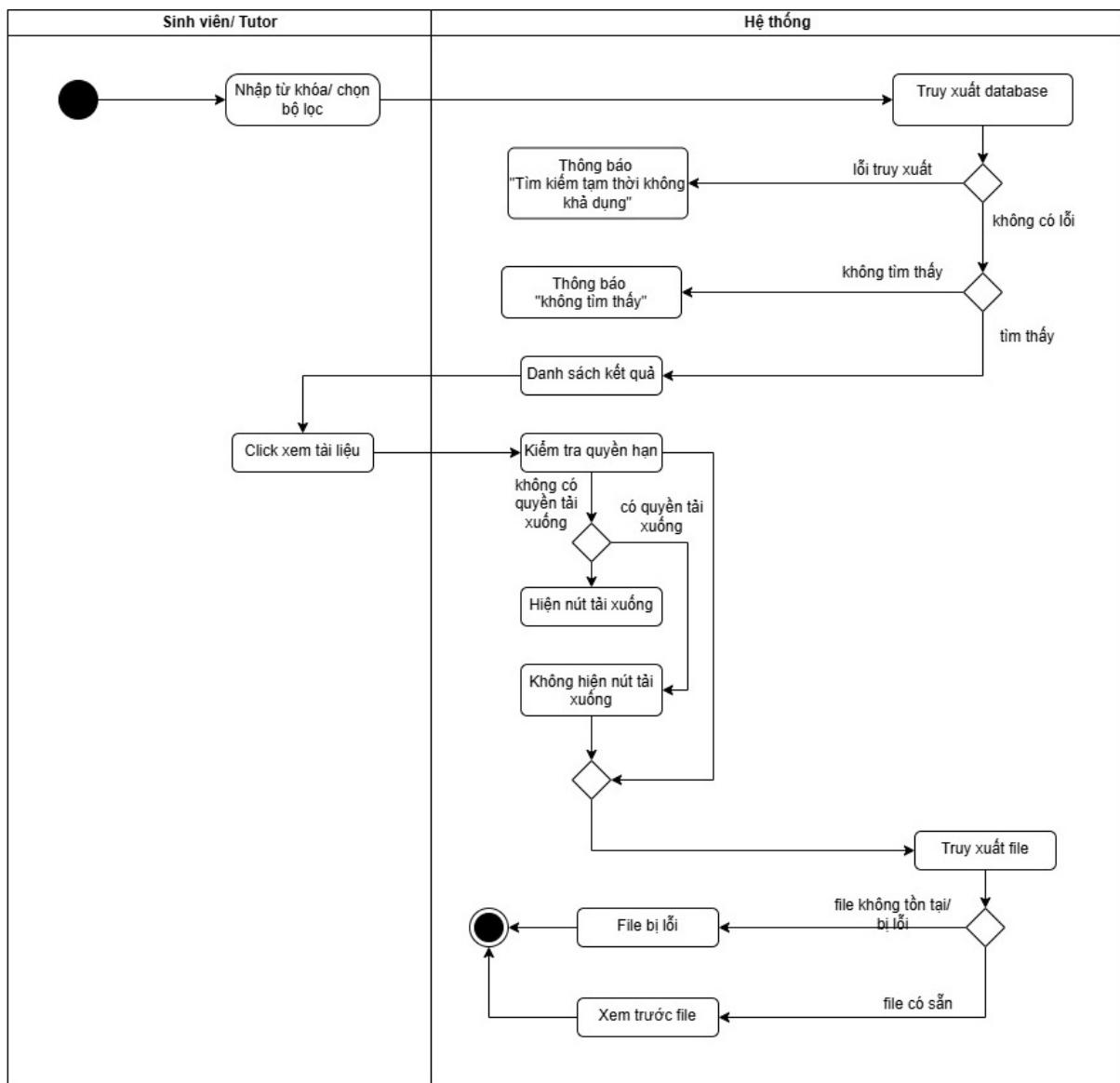
## 1.8 Quản lý tài liệu



Hình 1.22: Sơ đồ hoạt động cho use-case "Tải tài liệu lên"

### Mô tả hoạt động Activity Diagram: 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 phía người dùng.

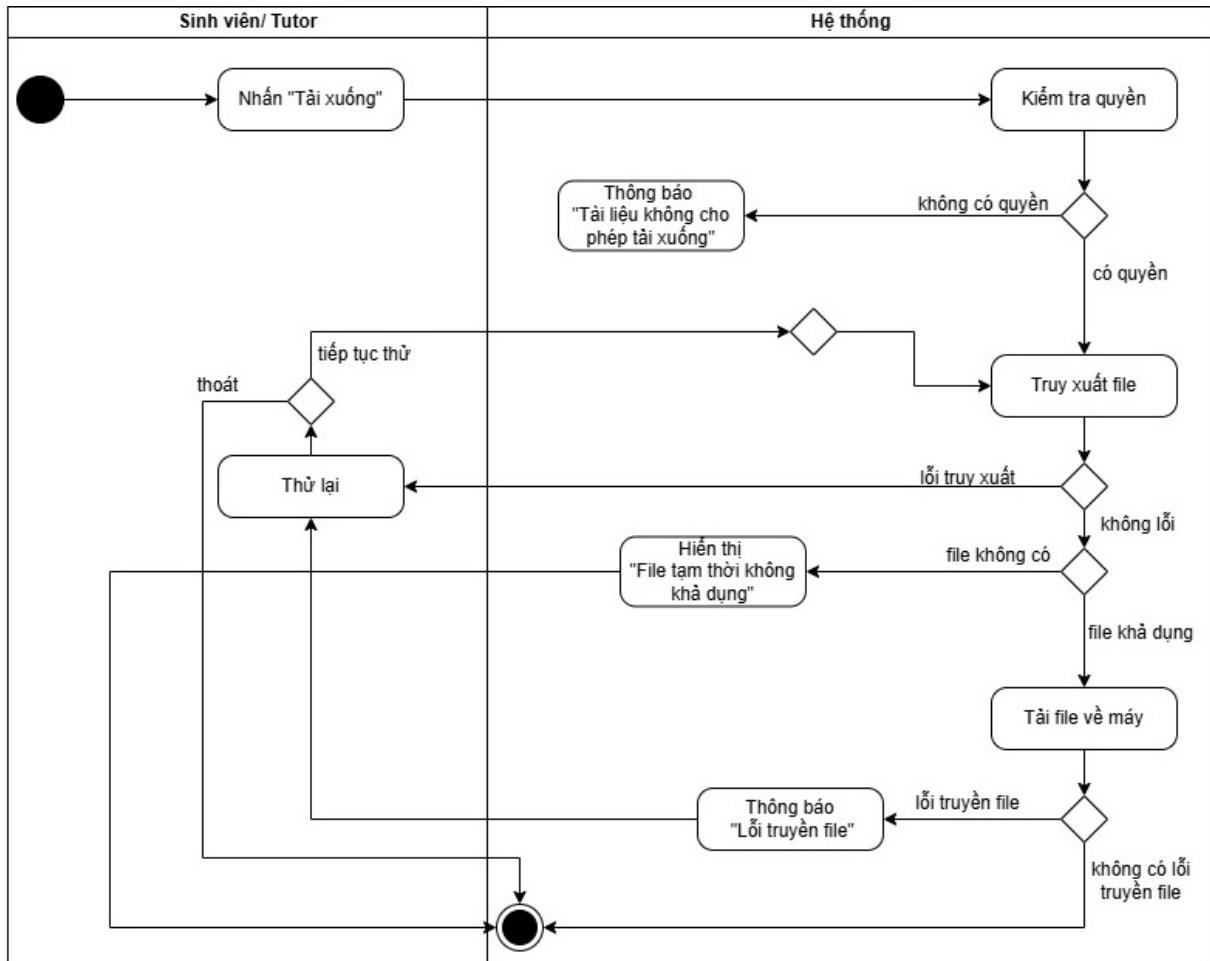


Hình 1.23: Sơ đồ hoạt động cho use-case "Tìm kiếm và xem trước"

### Mô tả hoạt động Activity Diagram: Việc tìm kiếm và xem trước tài liệu

1. **Người dùng** nhập từ khóa trên thanh "Tìm kiếm" và có thể chọn bộ lọc.
2. **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.
3. 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.
4. **Hệ thống** sau đó trả về kết quả và hiển thị giao diện xem trước tài liệu cho **người dùng**.
5. Giao diện xem trước có thêm nút tải xuống nếu tài liệu cho phép tải về.

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



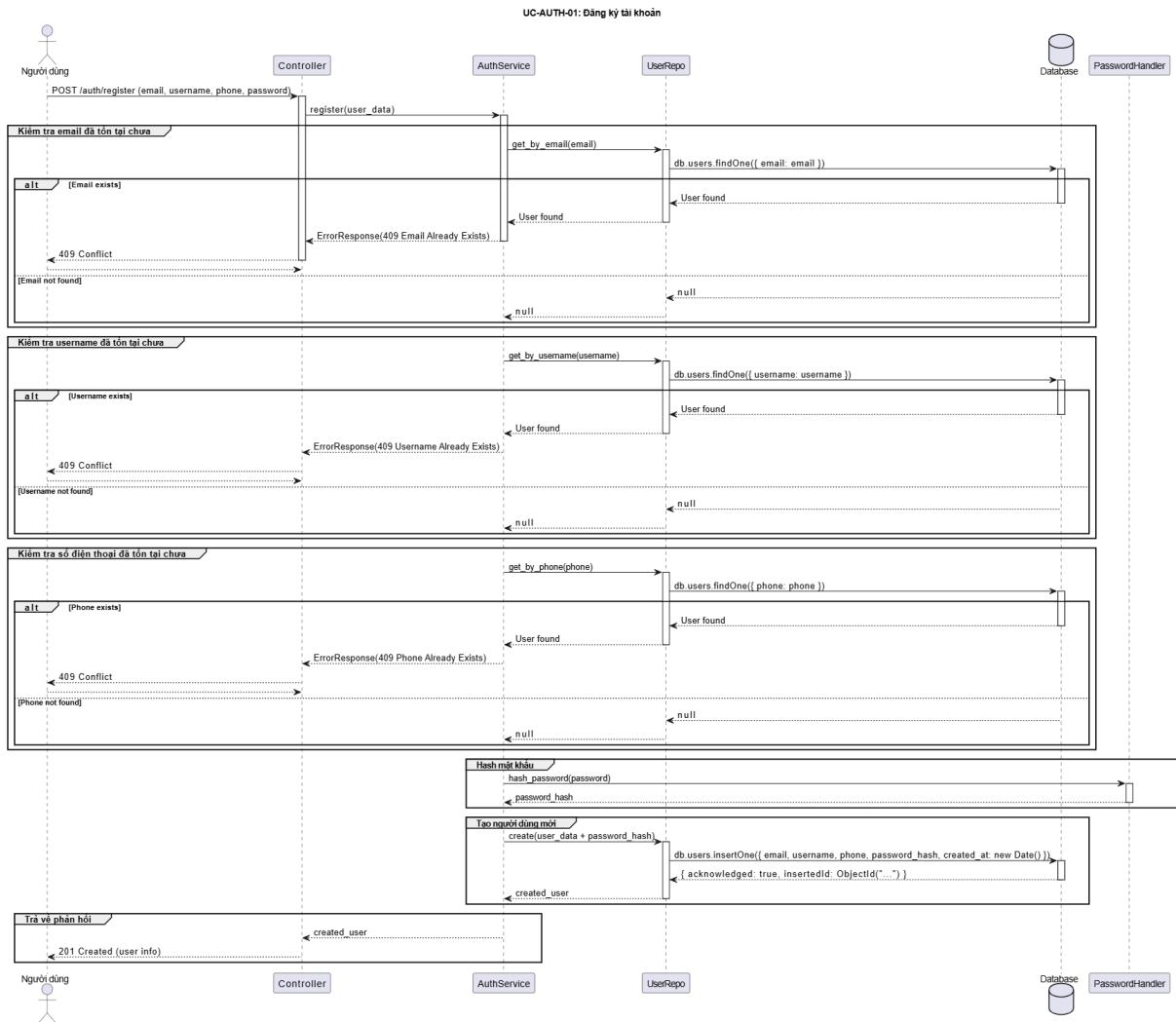
Hình 1.24: Sơ đồ hoạt động cho use-case "Tải tài liệu xuống"

### Mô tả hoạt động Activity Diagram: Việc tải tài liệu xuống

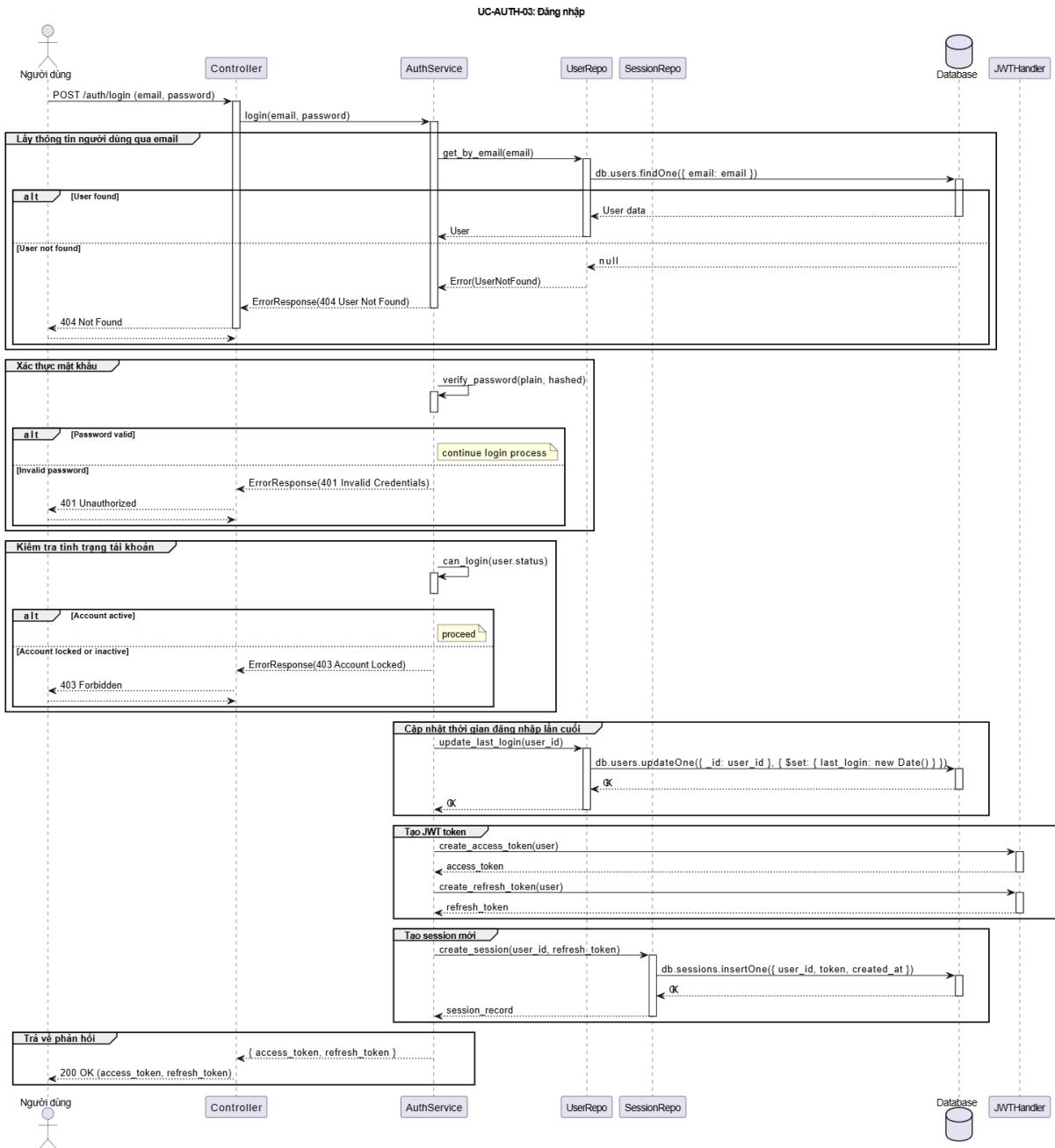
1. **Người dùng** chọn biểu tượng "Tải xuống" trên giao diện ứng dụng.
2. Sau đó **hệ thống** kiểm tra quyền tải xuống rồi thực hiện truy xuất file cần tải xuống.
3. Nếu có phát sinh lỗi, **hệ thống** sẽ hỏi xem **người dùng** muốn tiếp tục tải xuống lại nữa không, nếu người dùng muốn thì sẽ tiếp tục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

## 2 Sequence Diagrams

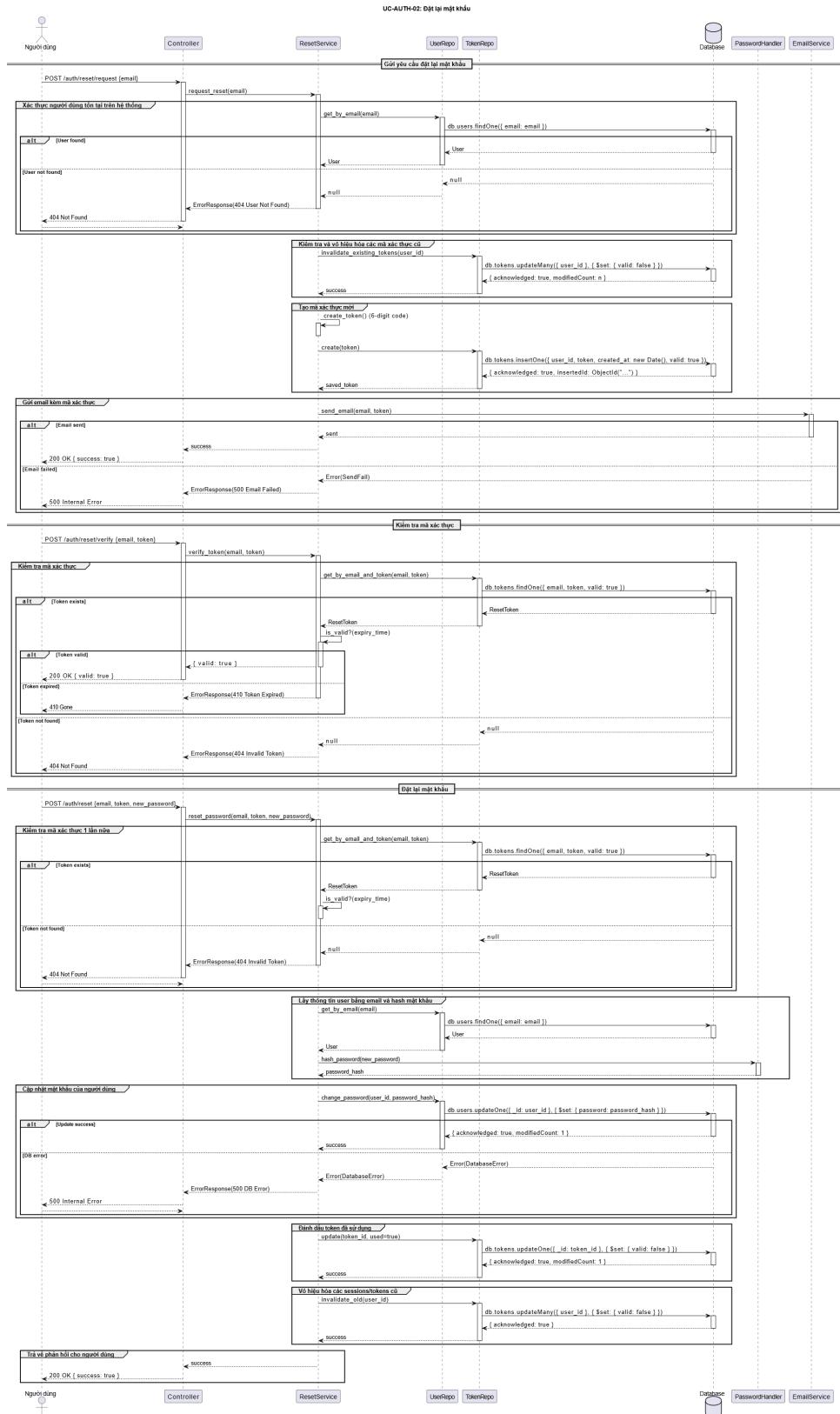
### 2.1 Quản lý truy cập và xác thực



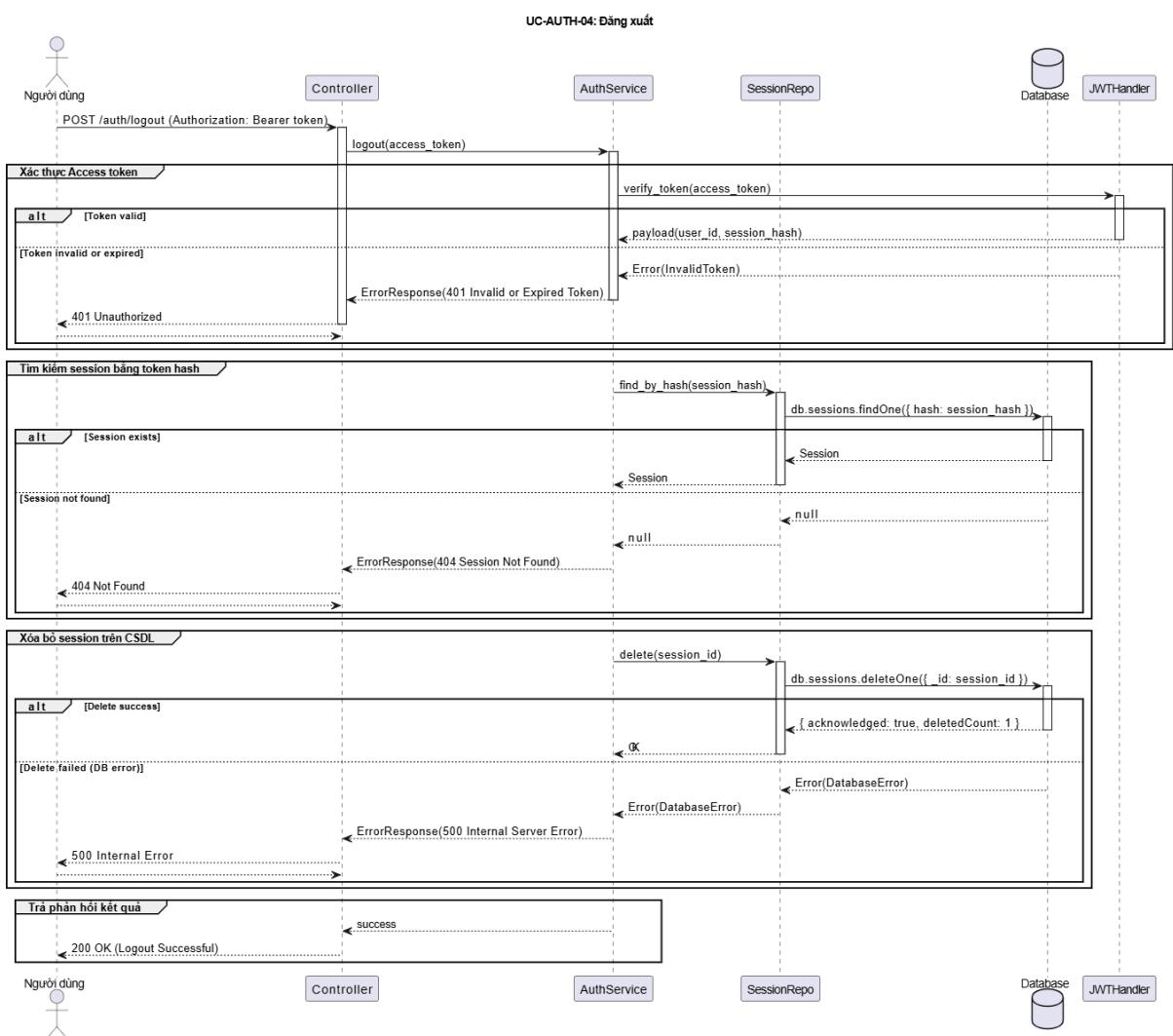
Hình 2.1: Sơ đồ tuần tự cho use-case "Quản lý tài khoản"



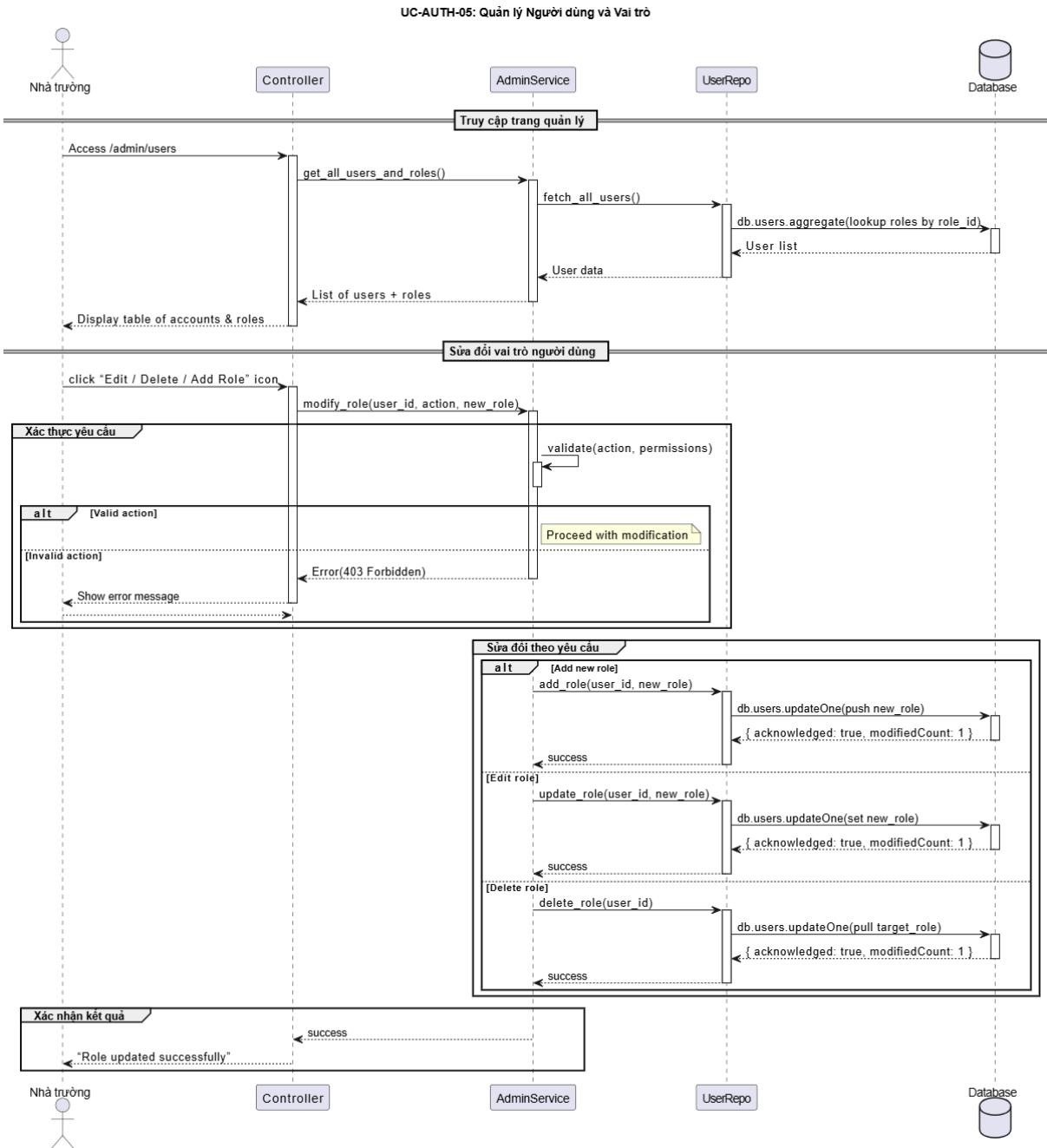
Hình 2.2: Sơ đồ tuần tự cho use-case "Đăng nhập"



Hình 2.3: Sơ đồ tuần tự cho use-case "Đặt lại mật khẩu"

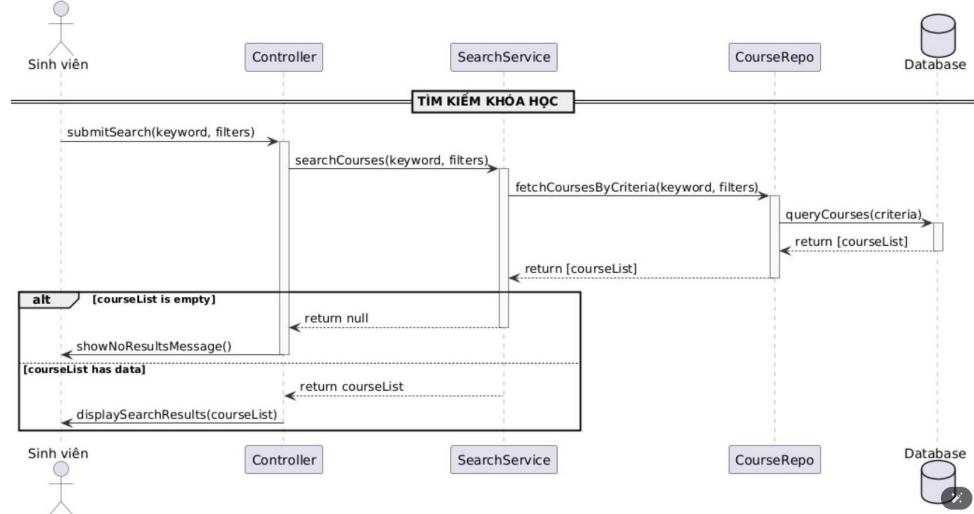


Hình 2.5: Sơ đồ tuần tự cho use-case "Đăng xuất"

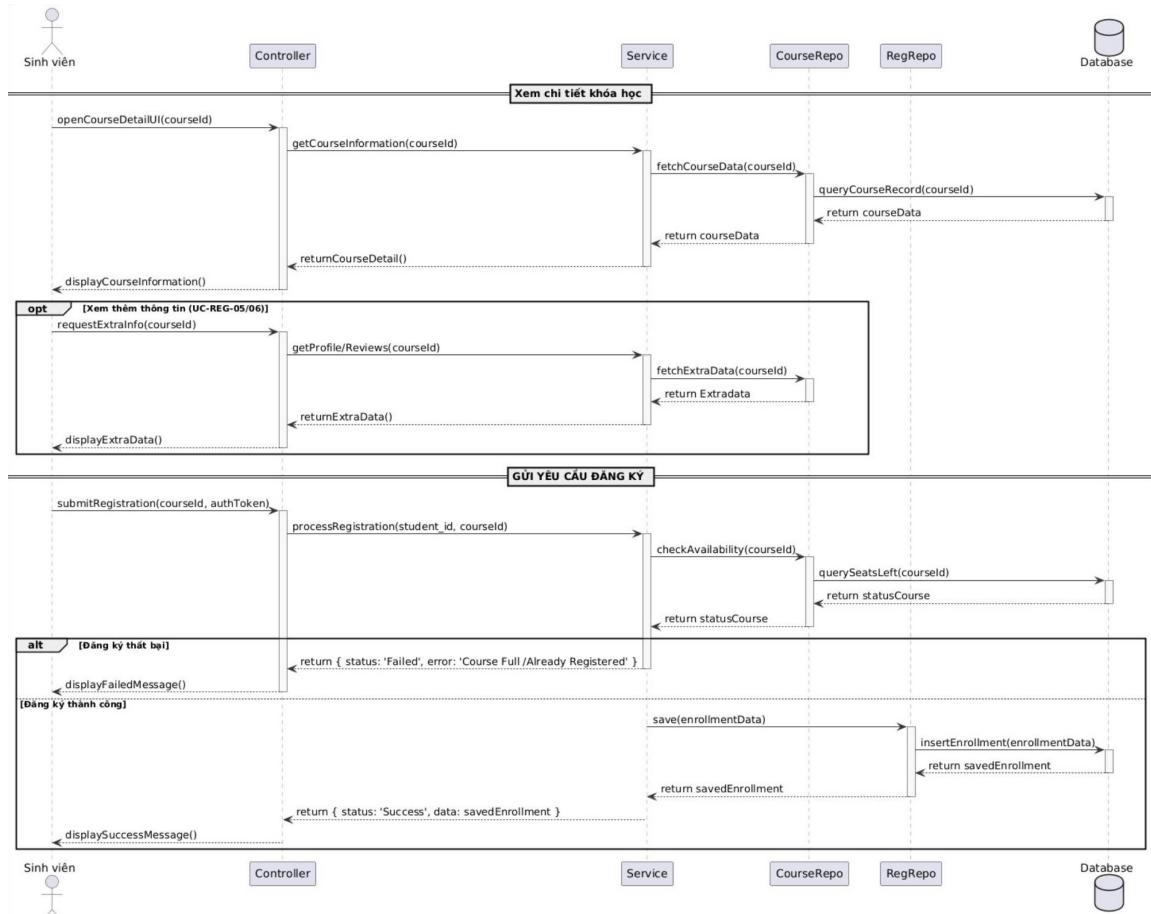


Hình 2.4: Sơ đồ tuần tự cho use-case "Quản lý người dùng"

## 2.2 Đăng ký chương trình

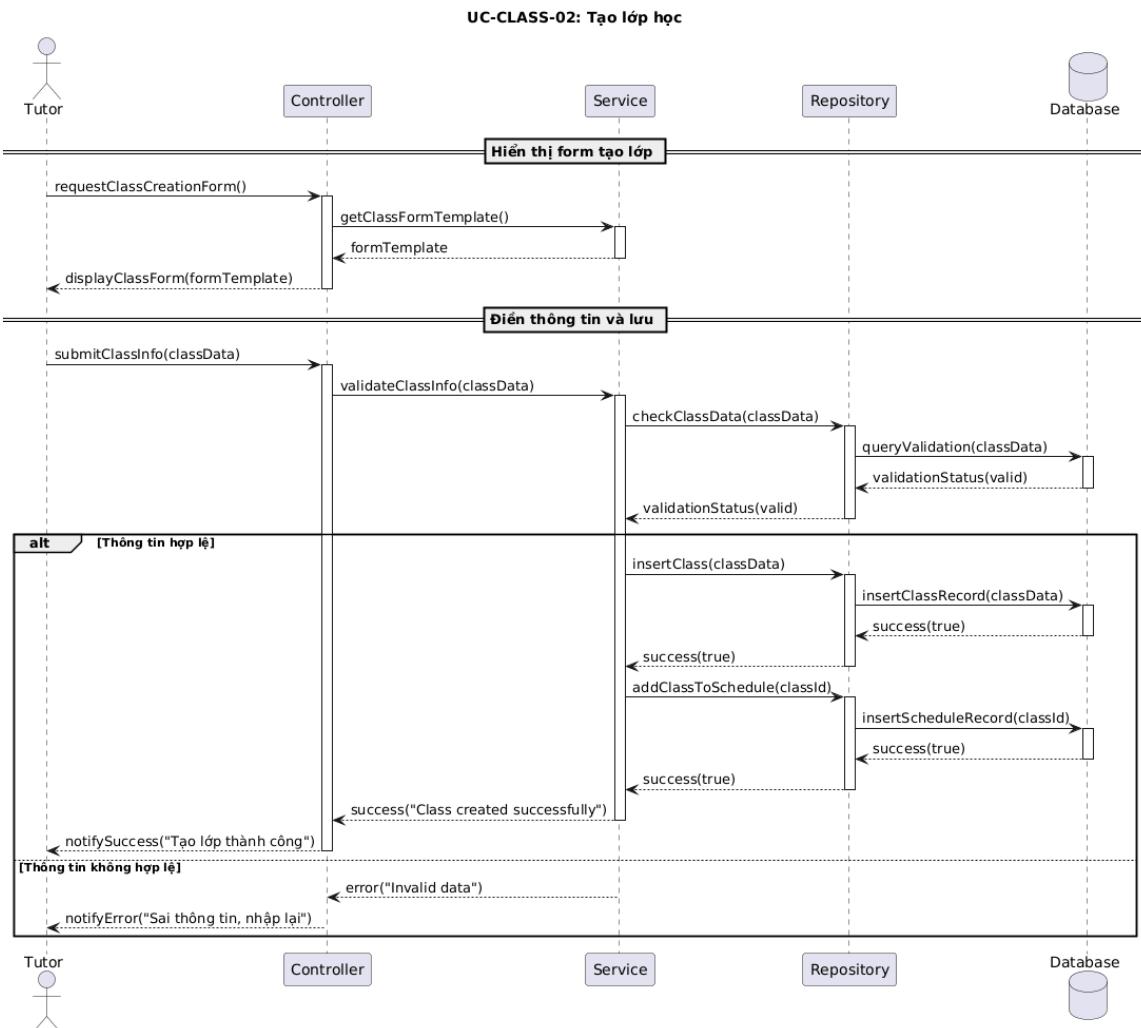


Hình 2.6: Sơ đồ tuần tự cho use-case "Tìm kiếm chương trình"

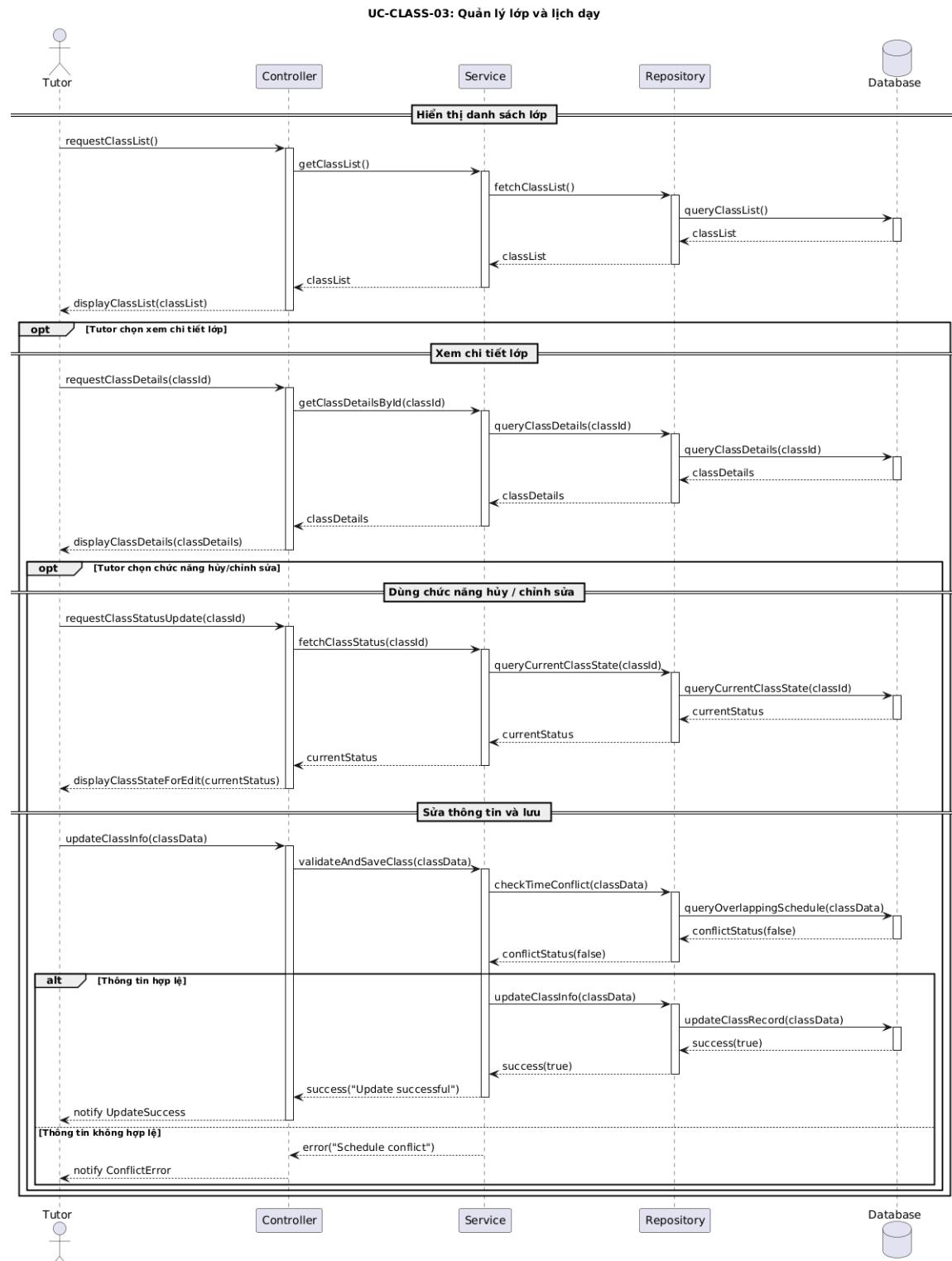


Hình 2.7: Sơ đồ tuần tự cho use-case "Đăng ký chương trình"

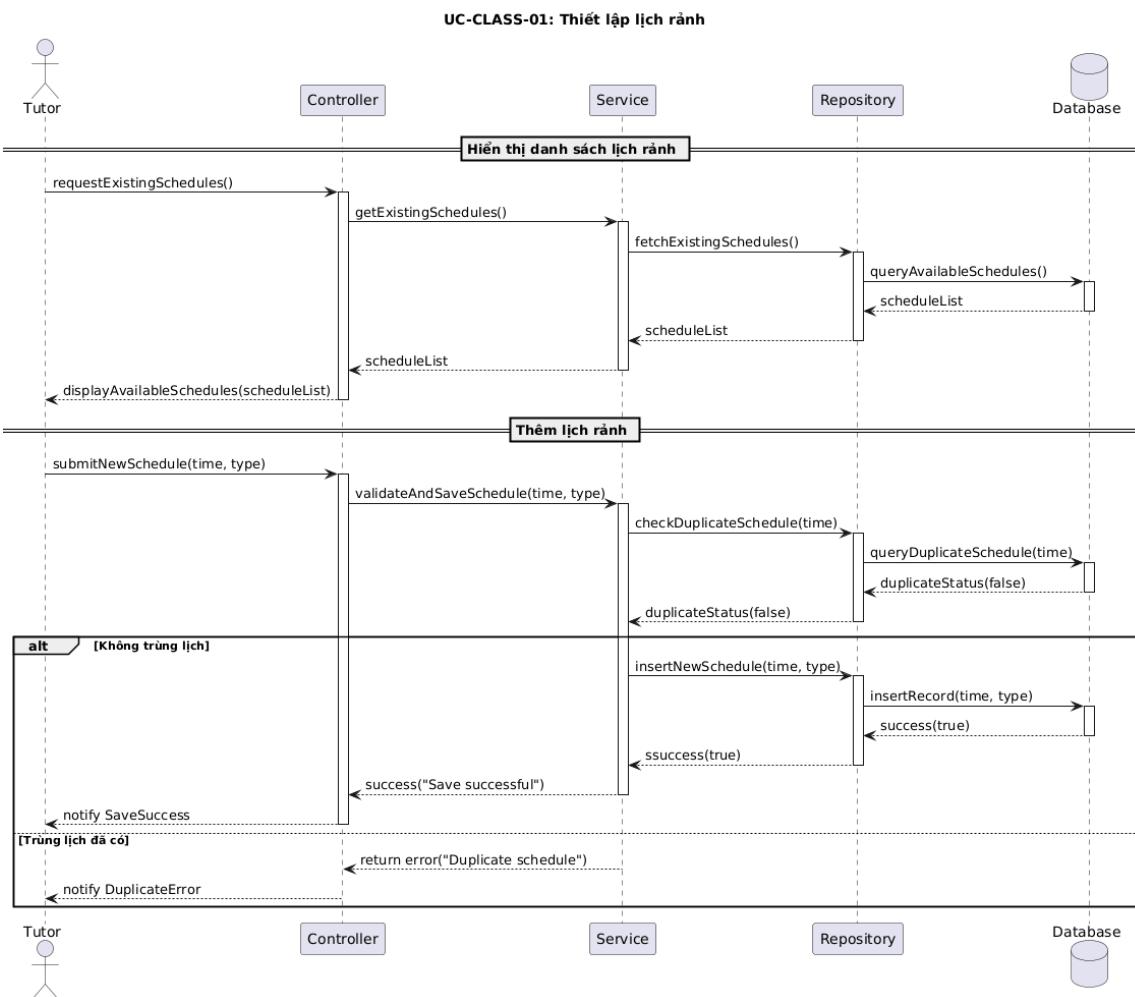
### 2.3 Tạo chương trình học



Hình 2.8: Sơ đồ tuần tự cho use-case "Tạo lớp học"

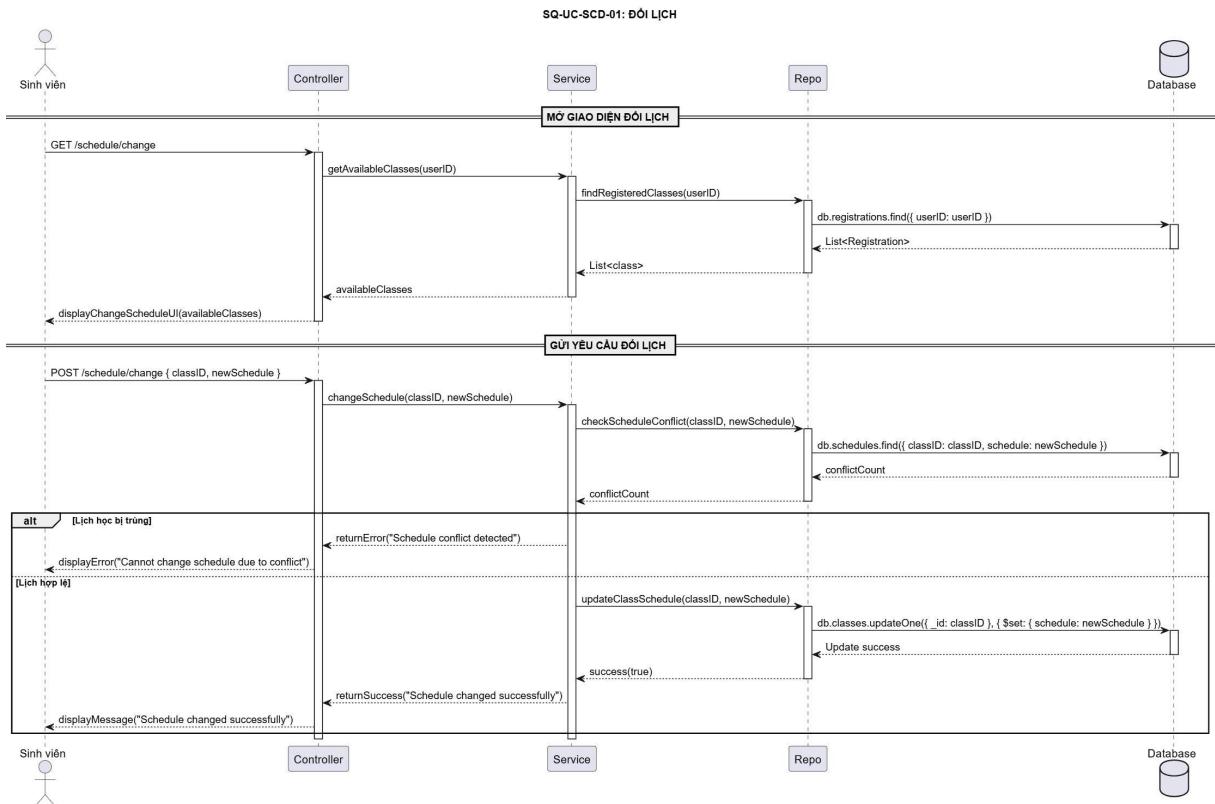


Hình 2.9: Sơ đồ tuần tự cho use-case "Quản lý lớp học"



Hình 2.10: Sơ đồ tuần tự cho use-case "Quản lý lịch rảnh"

## 2.4 Thiết lập lịch trình cho sinh viên



Hình 2.11: Sơ đồ tuần tự cho use-case "Đổi lịch học" cho sinh viên"

Use-case “Đổi lịch học” gồm hai pha chính: (1) mở giao diện đổi lịch và (2) gửi yêu cầu đổi lịch.

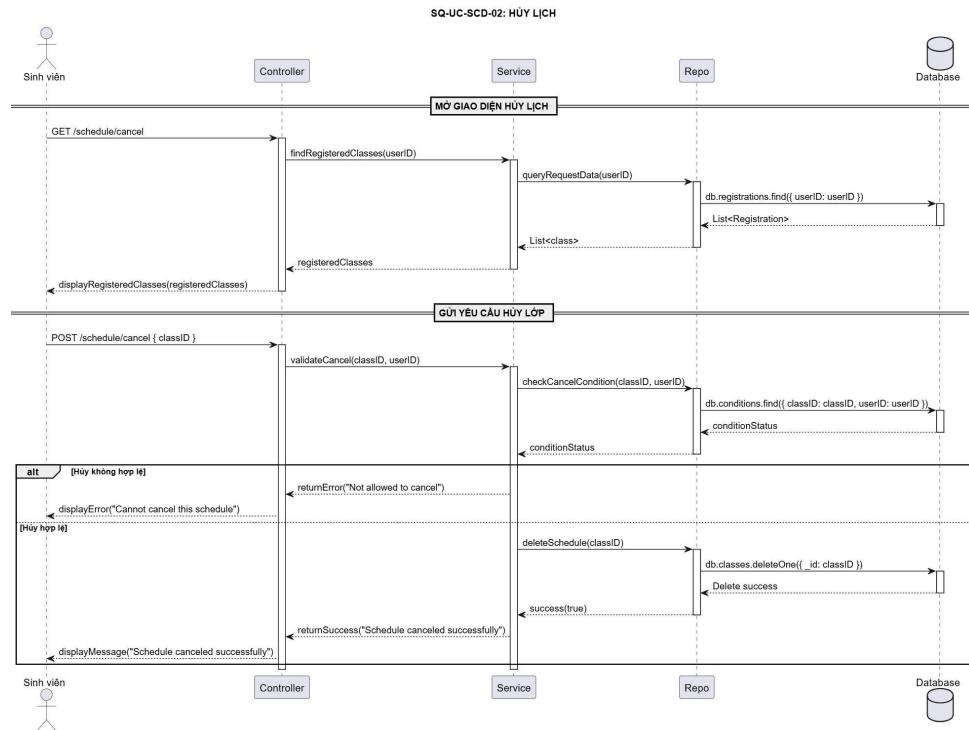
### Mở giao diện đổi lịch.

- Sinh viên gửi yêu cầu HTTP GET `/schedule/change` tới hệ thống.
- `ScheduleEndpoint` nhận yêu cầu và gọi phương thức `getAvailableClasses(userId)` của `ScheduleService`.
- `ScheduleService` gọi tiếp `findRegistrationsByUser(userId)` trên `ScheduleRepository` để truy vấn danh sách các lớp mà sinh viên đã đăng ký từ cơ sở dữ liệu `db.registrations`.
- `ScheduleRepository` trả về danh sách `List<Registration>` cho `ScheduleService`, sau đó service chuyển đổi thành danh sách lớp có thể đổi và trả lại cho `ScheduleEndpoint`.
- Controller trả dữ liệu về phía giao diện và hiển thị màn hình *Change Schedule UI* cho phép sinh viên chọn lớp và ca học mới.

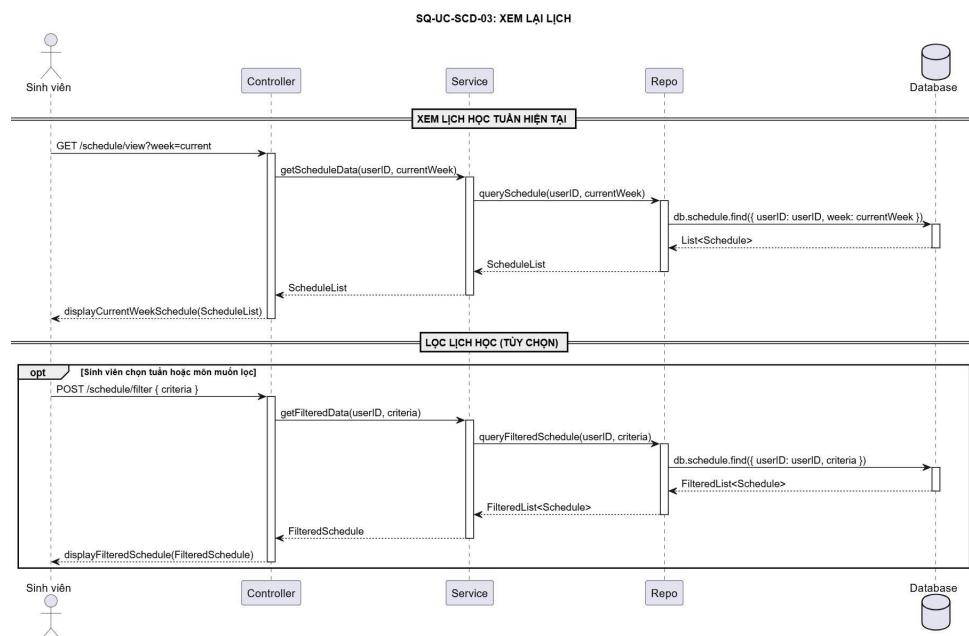


### Gửi yêu cầu đổi lịch.

- Sau khi chọn lớp và ca học mới, sinh viên gửi yêu cầu POST /schedule/change {classId, newSchedule}.
- ScheduleEndpoint gọi changeSchedule(userId, classId, newSchedule) trên ScheduleService.
- Service gọi checkScheduleConflict(classId, newSchedule). Bên trong đó, ScheduleRepository thực hiện truy vấn db.schedules.find({classId, schedule: newSchedule}) để kiểm tra trùng lịch và trả về conflictCount.
- Nếu conflictCount > 0, service trả kết quả ConditionStatus với isValid = false và lý do “Schedule conflict detected”. Controller hiển thị thông báo lỗi “Cannot change schedule due to conflict” cho sinh viên.
- Nếu không có xung đột, ScheduleService gọi updateClassSchedule(classId, newSchedule) trên ScheduleRepository để cập nhật ca học mới. Repository thực hiện thao tác cập nhật (ví dụ db.classes.updateOne({\_id: classId}, {\$set: {schedule: newSchedule}})) và trả về success = true.
- Service tạo ConditionStatus với isValid = true, lý do “Schedule changed successfully” và trả về cho controller. Cuối cùng, giao diện hiển thị thông báo “Schedule changed successfully” cho sinh viên.

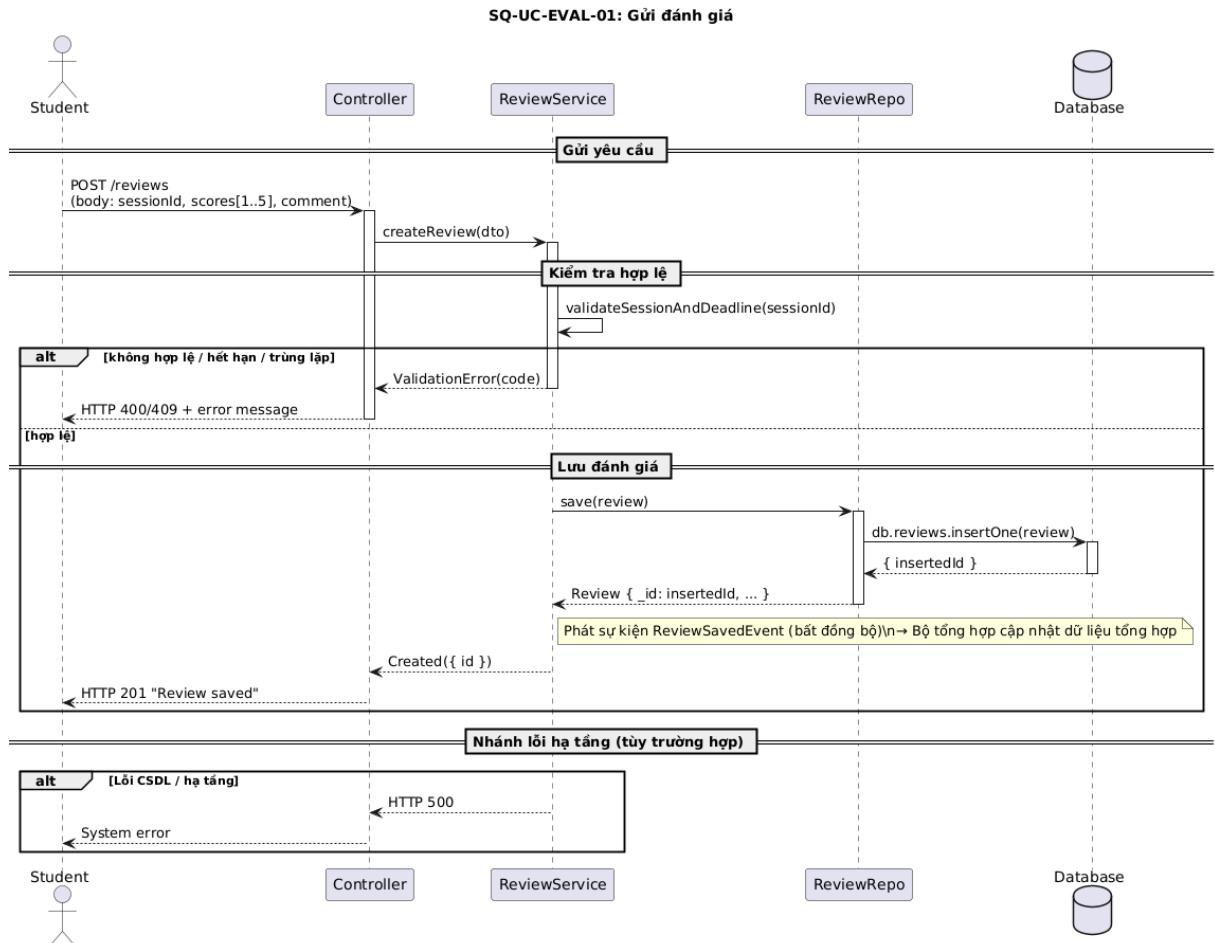


Hình 2.12: Sơ đồ tuần tự cho use-case "Hủy lịch học" cho sinh viên"



Hình 2.13: Sơ đồ tuần tự cho use-case "Xem lại lịch học" cho sinh viên"

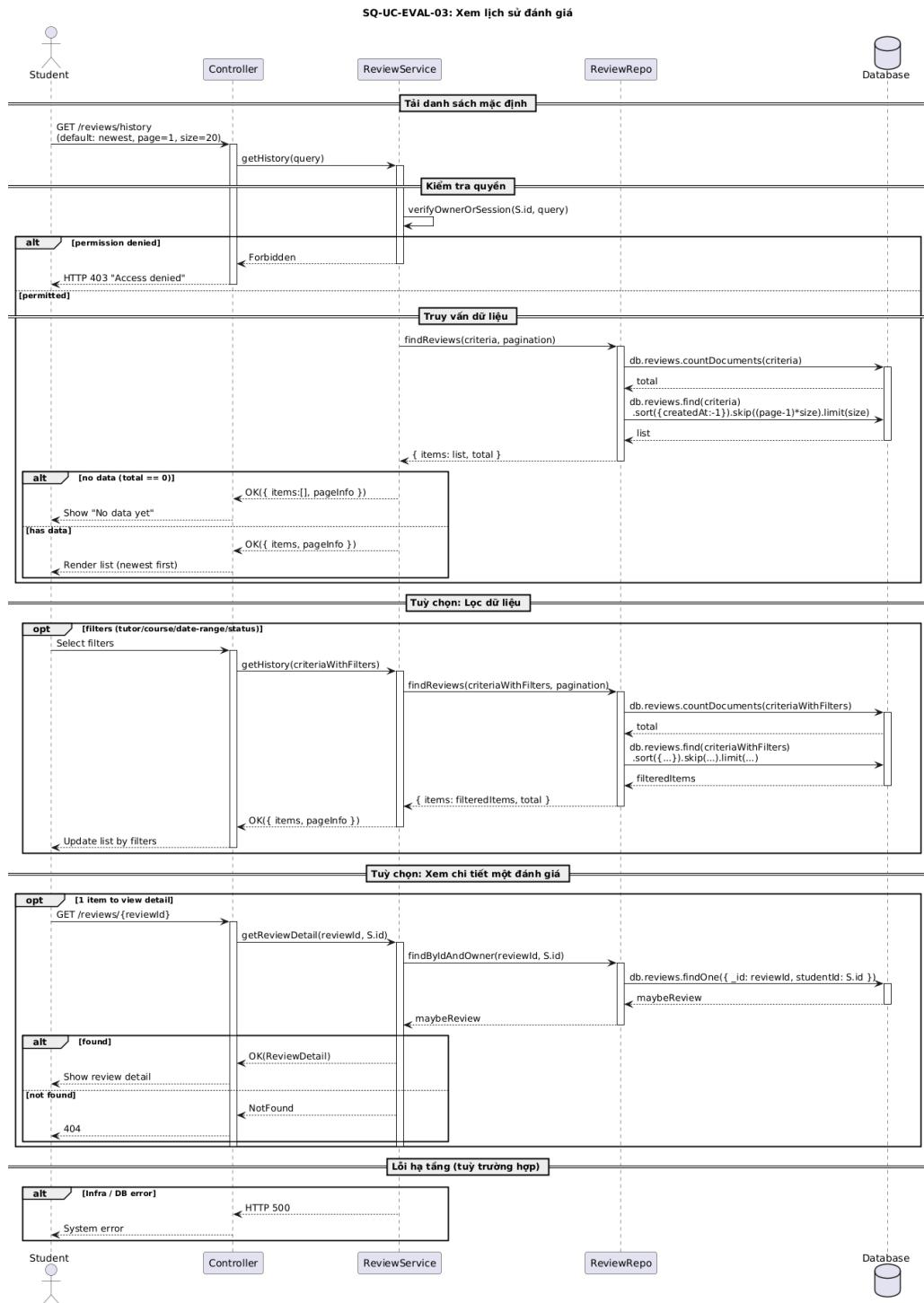
## 2.5 Đánh giá



Hình 2.14: Sơ đồ tuần tự cho use-case "Gửi đánh giá"

### Mô tả sơ đồ tuần tự "Gửi đánh giá"

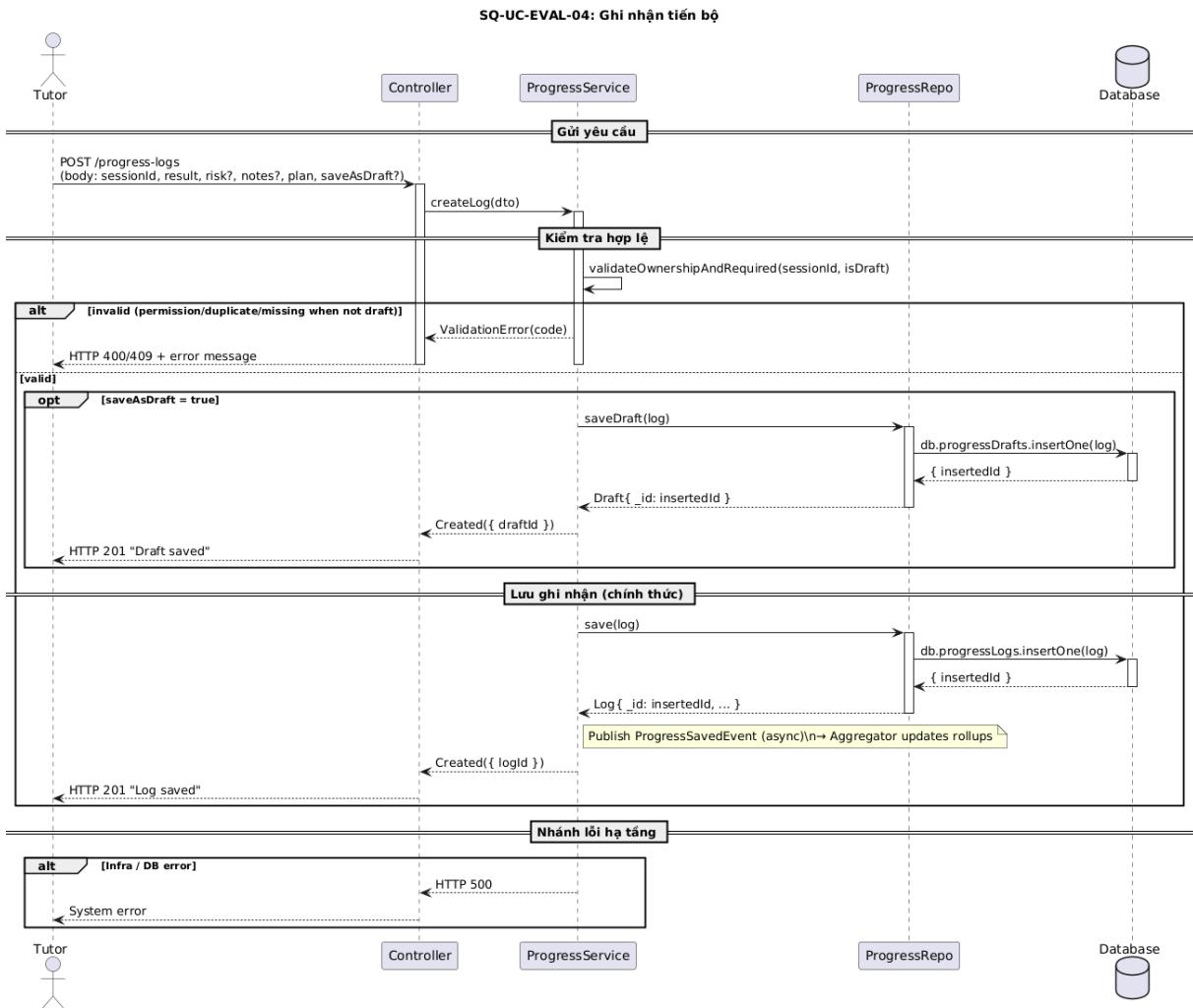
Controller gọi `ReviewService.createReview(dto)` → `validateSessionAndDeadline(sessionId)`  
→ lưu vào `db.reviews` → phát `ReviewSavedEvent` cho bộ tổng hợp.



Hình 2.15: Sơ đồ tuần tự cho use-case "Xem lịch sử đánh giá"

### Mô tả sơ đồ tuần tự "Xem lịch sử đánh giá"

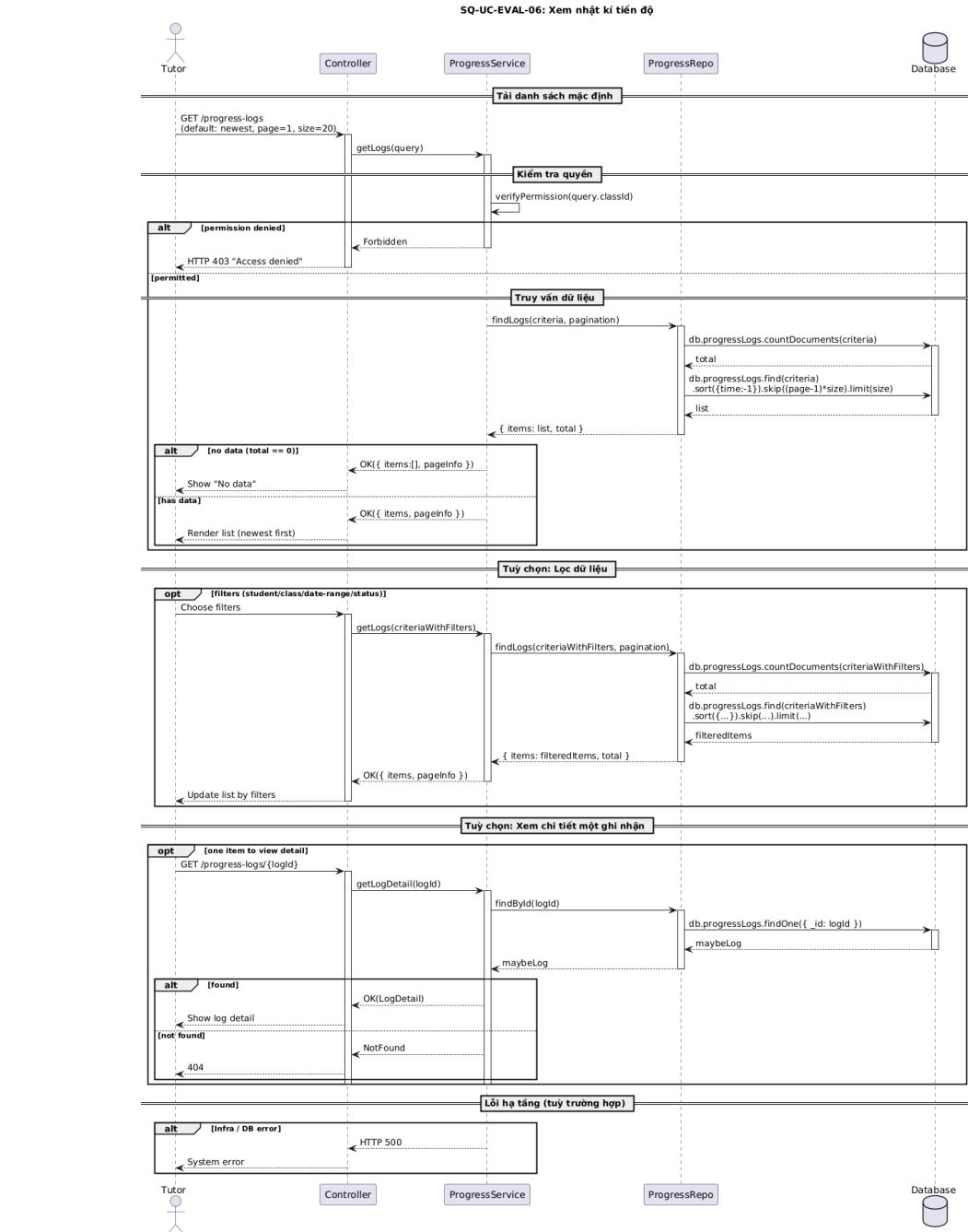
Controller nhận GET /reviews/history → ReviewService.verifyOwnerOrSession() → ReviewRepo.findReviews() truy vấn db.reviews với sắp xếp createdAt:-1 và phân trang.



Hình 2.16: 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.

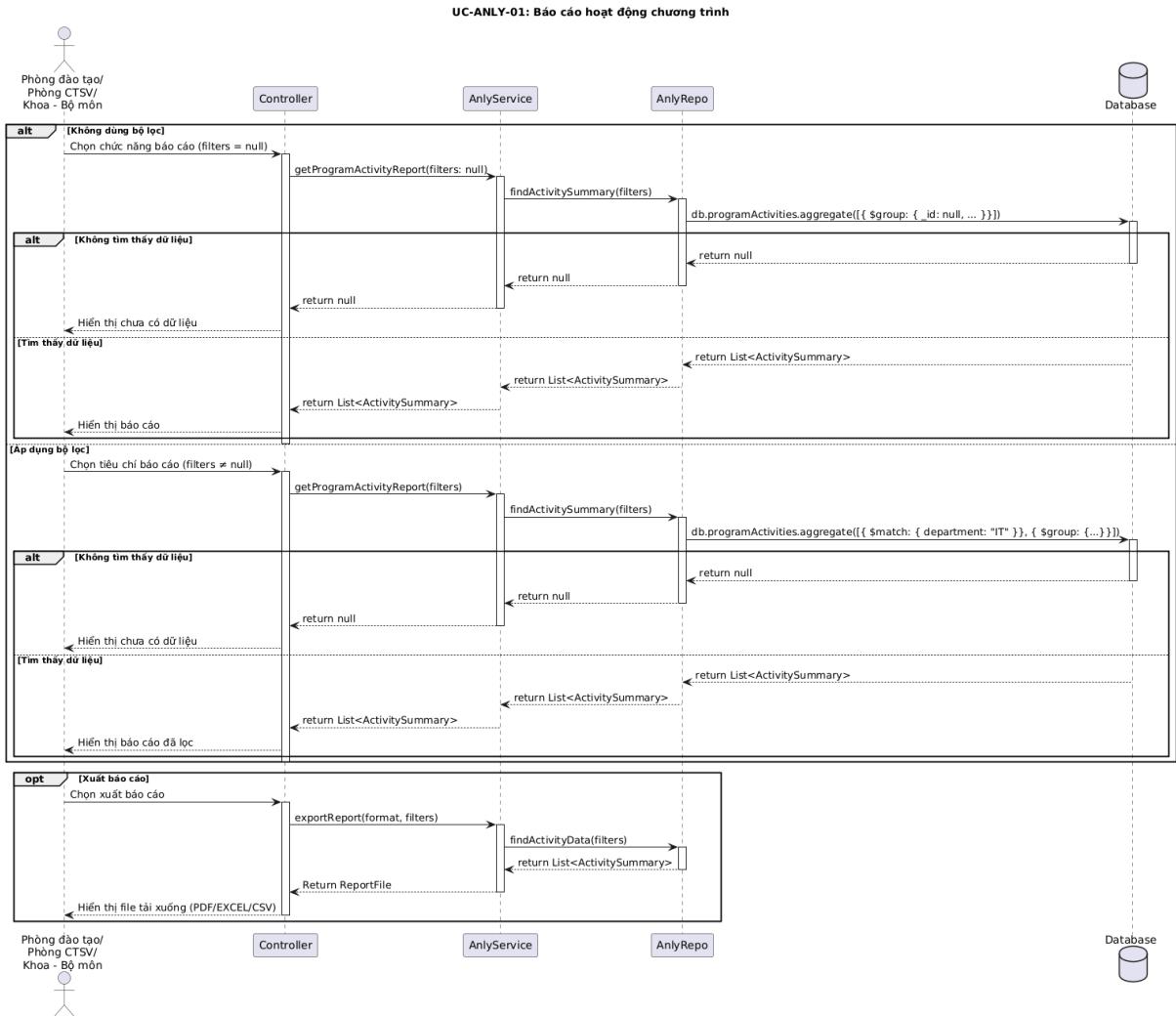


Hình 2.17: Sơ đồ tuần tự cho use-case "Xem nhật ký tiến độ"

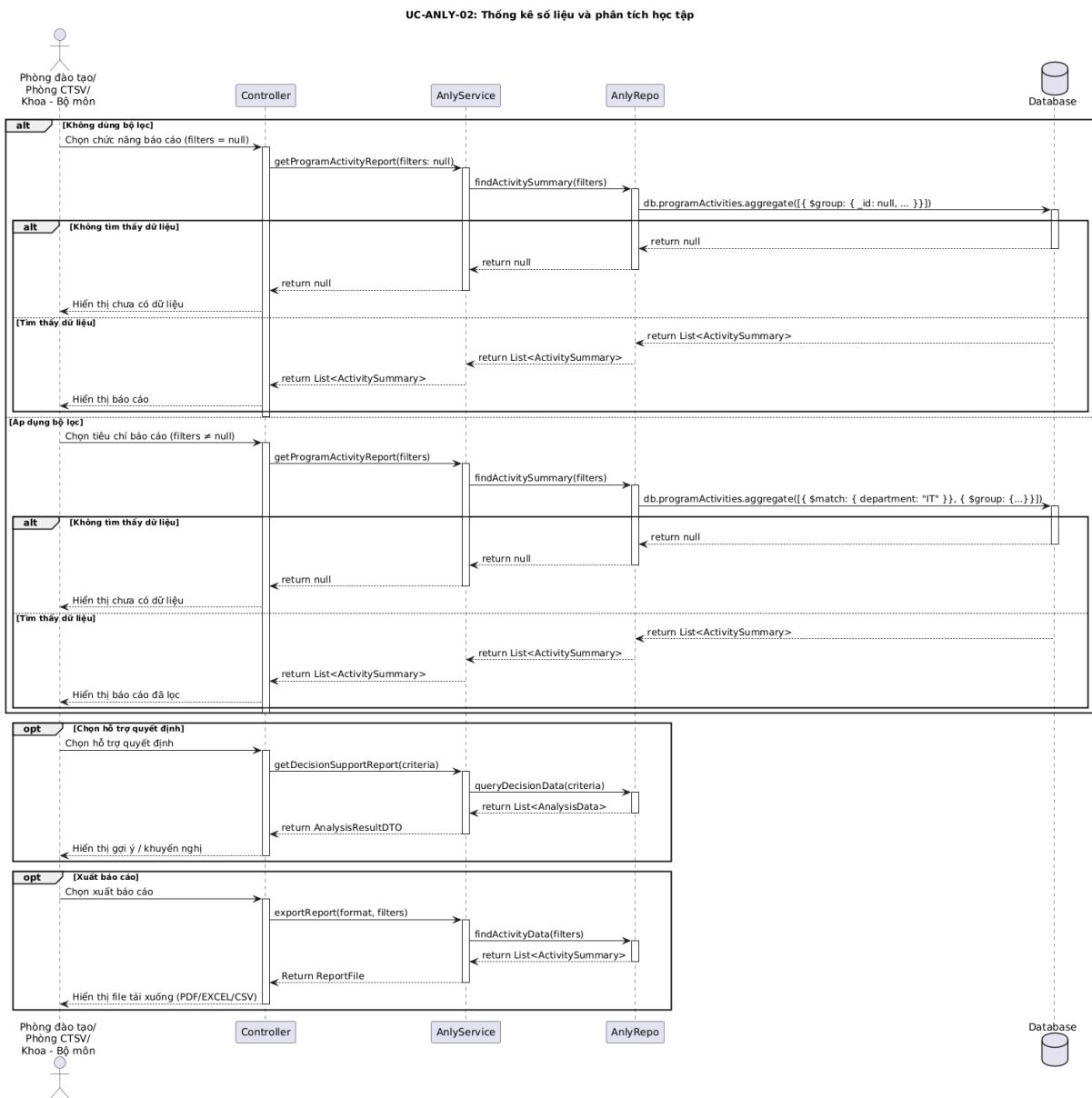
### Mô tả sơ đồ tuần tự "Xem nhật ký tiến độ"

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.

## 2.6 Phân tích và báo cáo

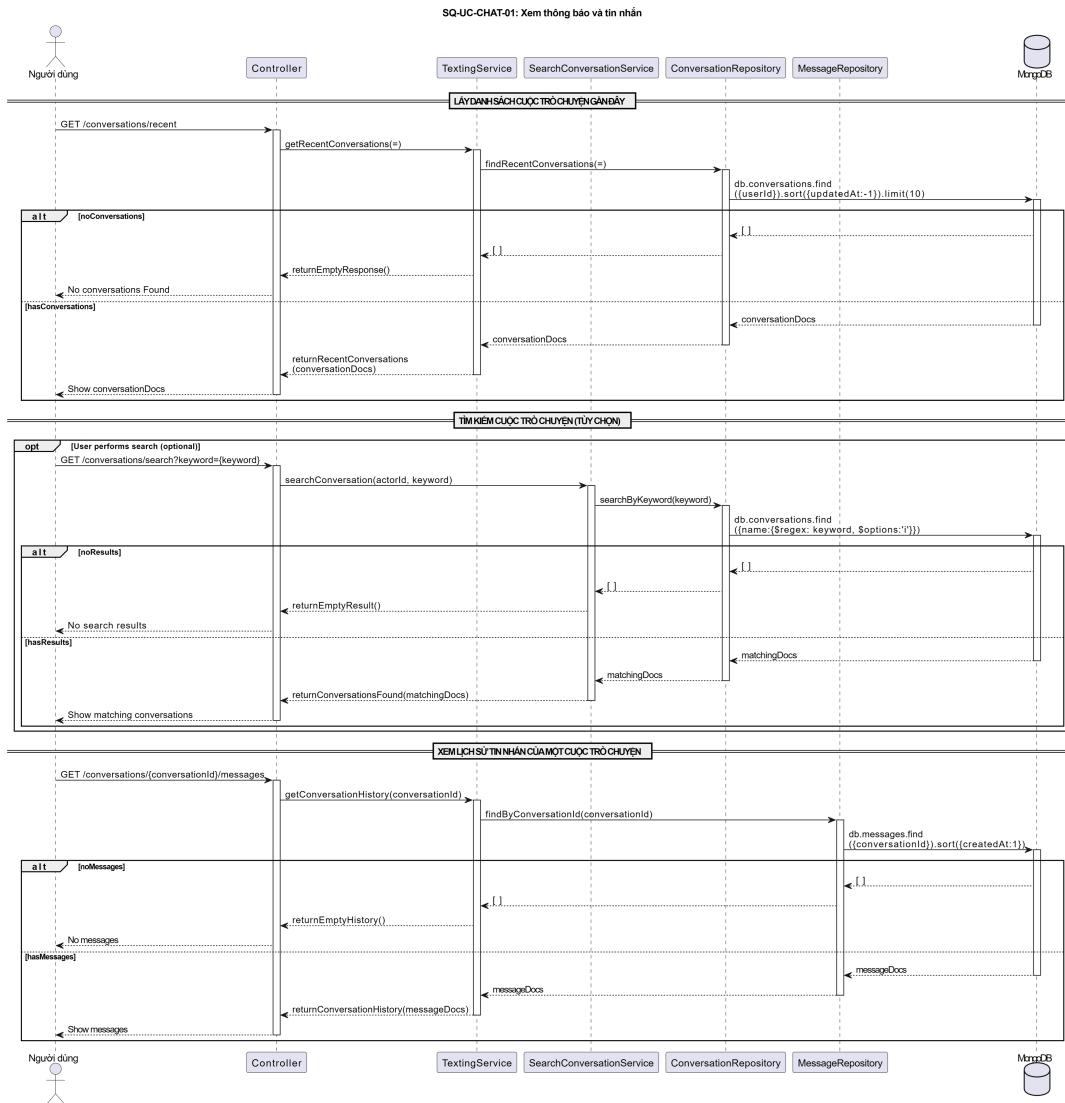


Hình 2.18: Sơ đồ tuần tự cho use-case "Báo cáo hoạt động chương trình"



Hình 2.19: Sơ đồ tuần tự cho use-case "Thống kê số liệu và phân tích học tập"

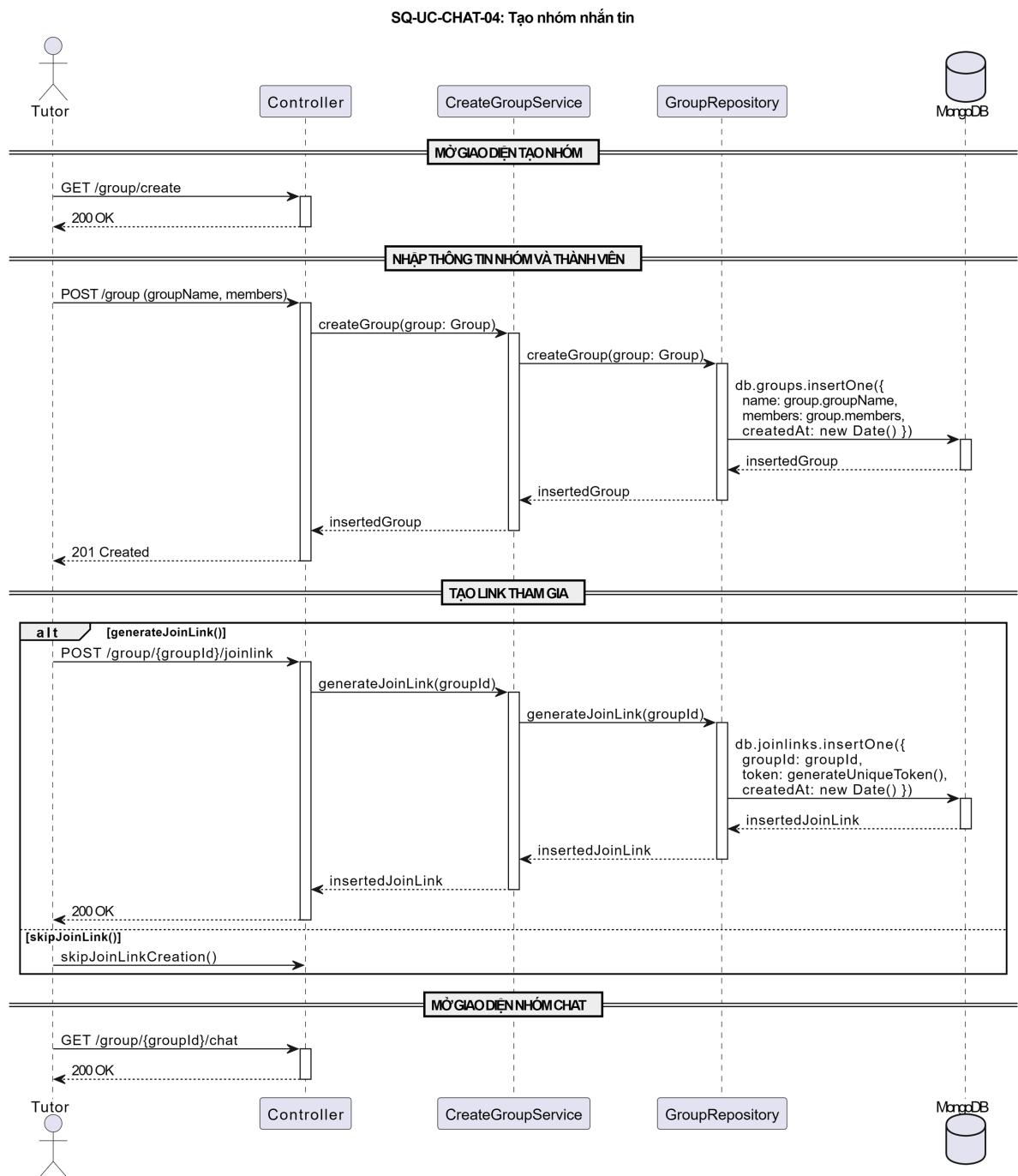
## 2.7 Thông báo và nhắn tin



Hình 2.20: Sơ đồ tuần tự cho use-case "Xem tin nhắn và thông báo"

### Mô tả: 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.

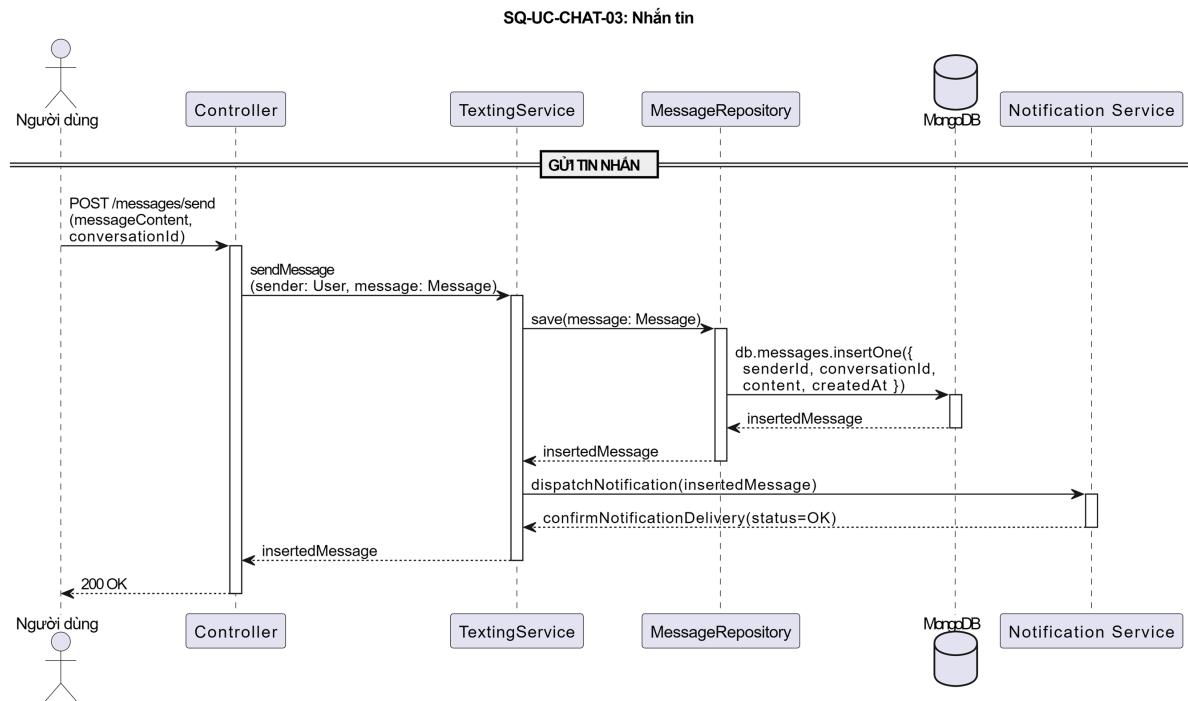


Hình 2.21: Sơ đồ tuần tự cho use-case "Tạo nhóm nhắn tin"



### Mô tả: Tạo nhóm nhắn tin

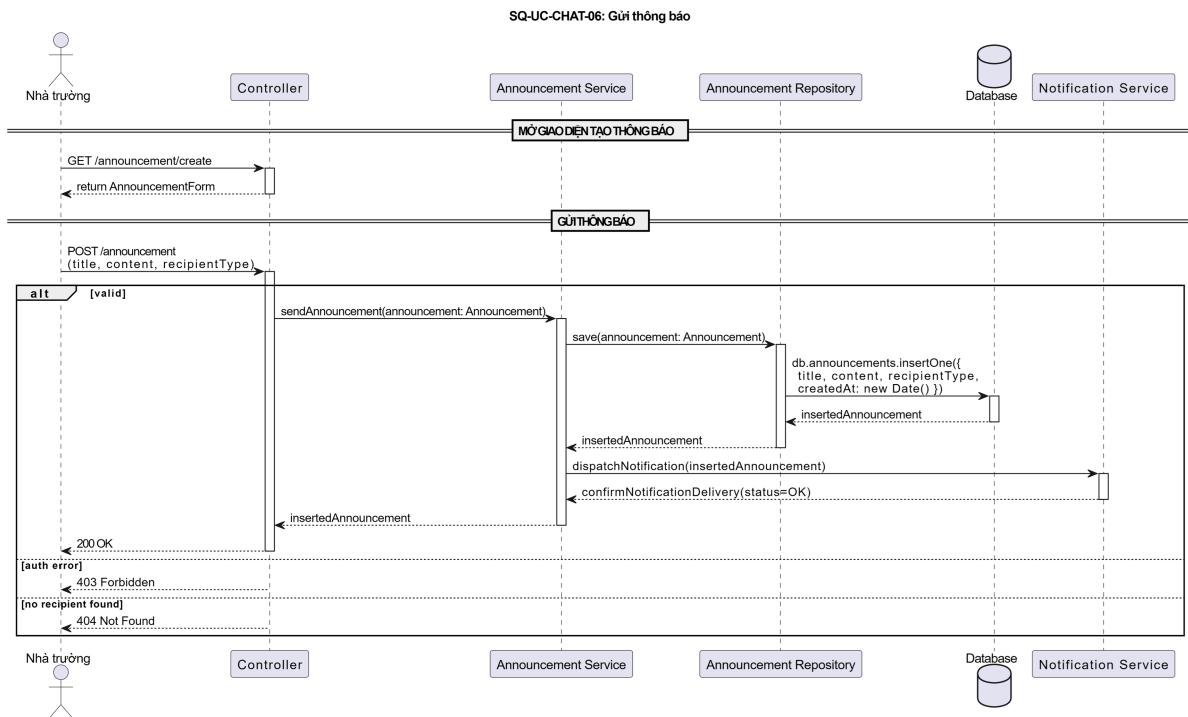
- a. Tutor gửi yêu cầu mở giao diện tạo nhóm (GET /group/create) và nhận phản hồi xác nhận.
- b. Tutor điền thông tin tên nhóm và danh sách thành viên rồi gửi lên (POST /group), Controller tiếp nhận chuyển cho CreateGroupService.
- c. Dịch vụ này gọi GroupRepository, lưu group mới vào DB (ghi nhận groupName,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



Hình 2.22: Sơ đồ tuần tự cho use-case "Nhắn tin"

### Mô tả: 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.

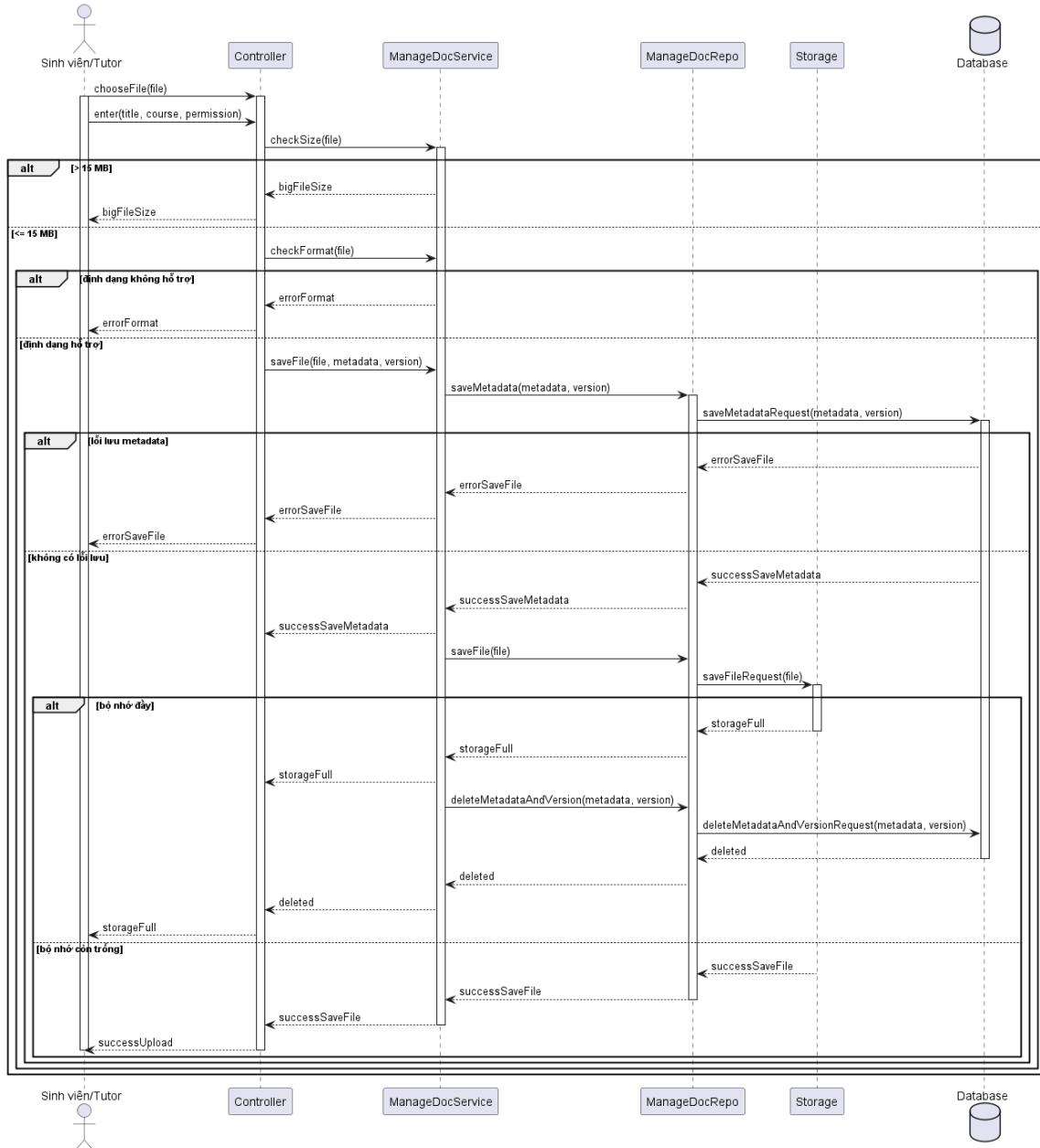


Hình 2.23: Sơ đồ tuần tự cho use-case "Gửi thông báo"

### Mô tả: 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).

## 2.8 Quản lý tài liệu



Hình 2.24: 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":

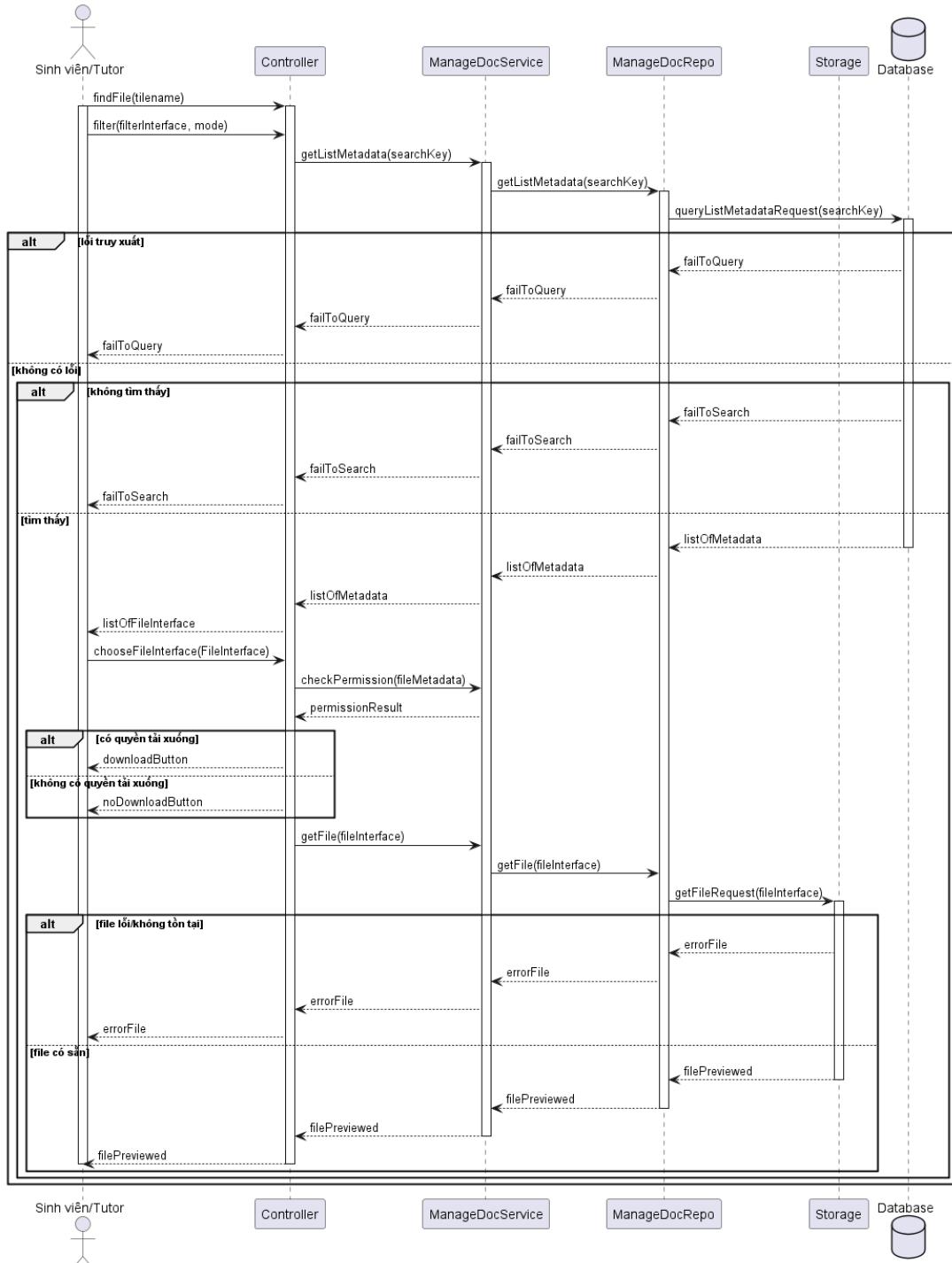
- Người dùng (Sinh viên/Tutor) chọn file và nhập thông tin tài liệu, gửi yêu cầu lên Controller.
- Controller kiểm tra kích thước file, nếu lớn hơn 15MB thì trả về lỗi, nếu hợp lệ thì tiếp tục kiểm tra định dạng file.
- Nếu định dạng không hỗ trợ, trả về lỗi cho người dùng. Nếu hợp lệ, Controller yêu 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.

### Mô tả: Tải tài liệu lên hệ thống



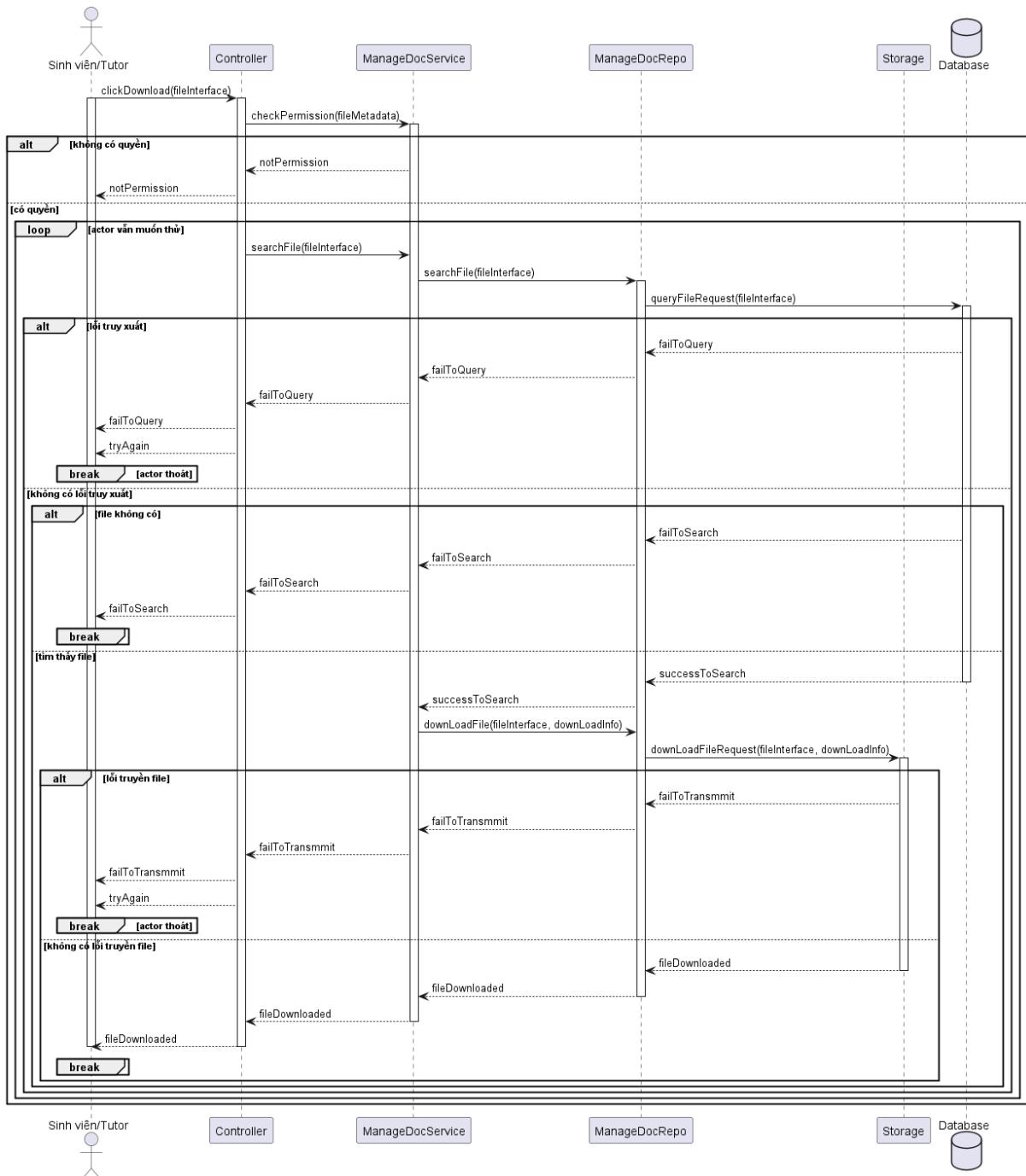
Hình 2.25: Sơ đồ tuần tự cho use-case "Tìm kiếm và xem tài liệu"

#### 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 2.26: 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.

### 3 Class Diagram

Tổng quan cấu trúc backend để vẽ class diagram

Các thành phần chính

1. **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.
2. **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.
3. **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.
4. **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).
5. **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.
6. **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

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



### 3.1 Quản lý truy cập và xác thực



### 3.2 Đăng ký chương trình

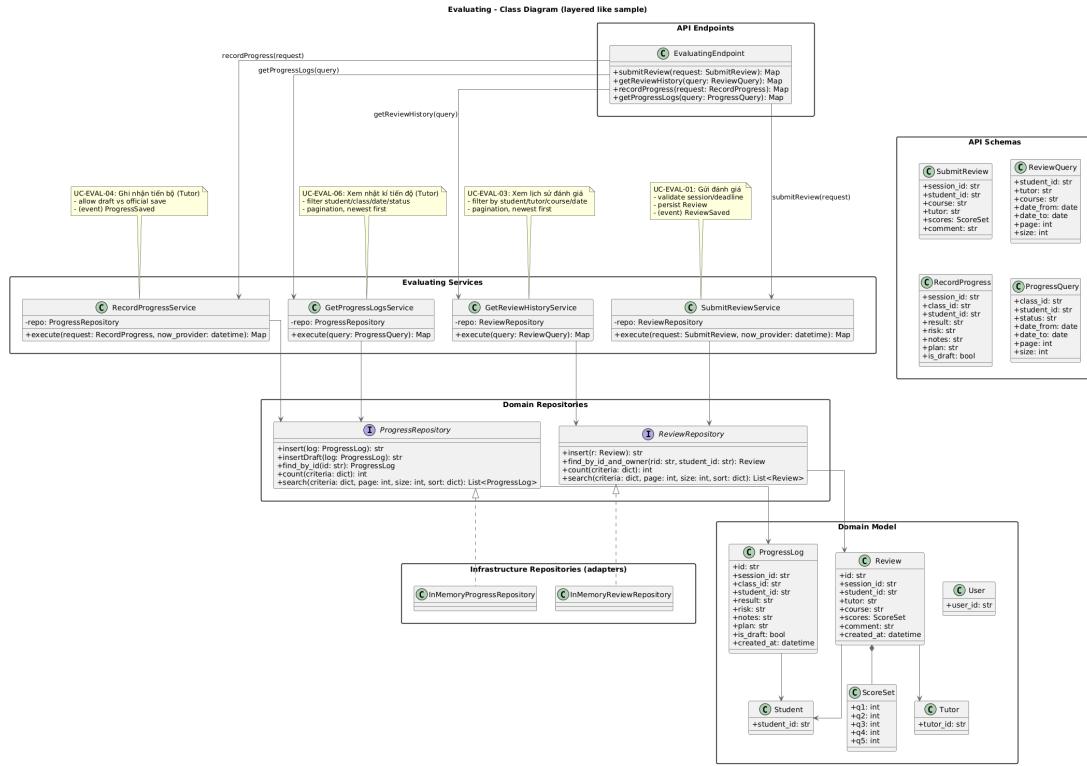


### 3.3 Tạo chương trình học



### 3.4 Thiết lập lịch trình cho sinh viên

### 3.5 Đánh giá



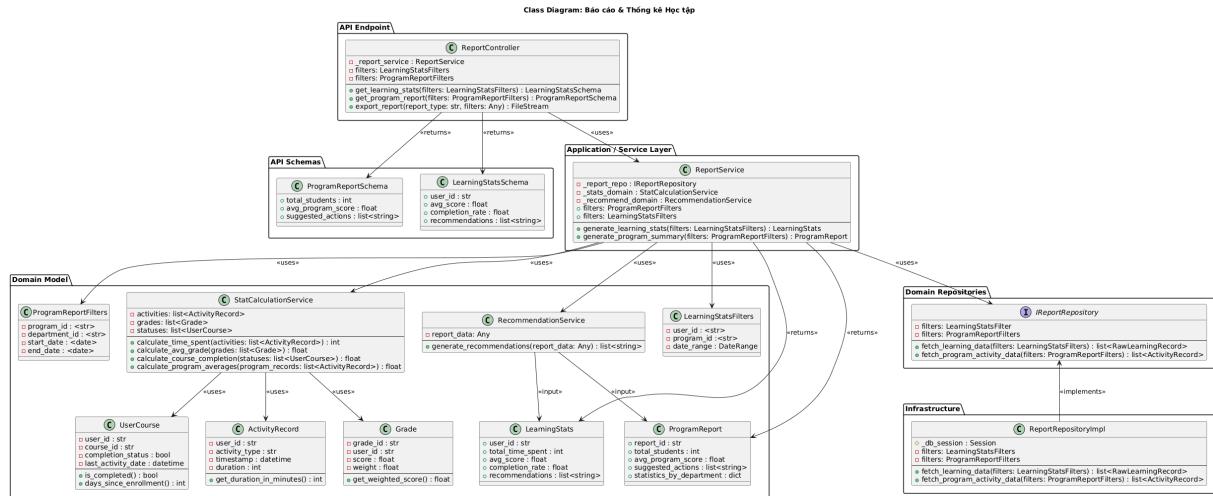
Hình 3.1: Sơ đồ lớp cho use-case "Đánh giá"

### 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`.

### 3.6 Phân tích và báo cáo



Hình 3.2: Sơ đồ lớp cho phân tích và báo cáo

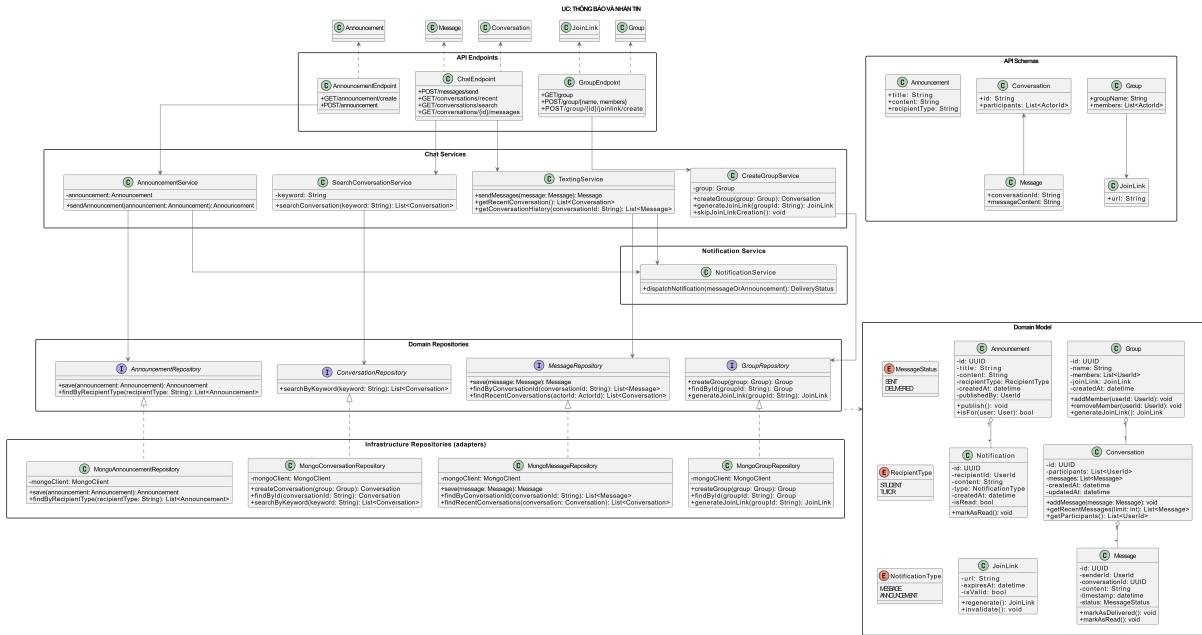
#### Mô tả sơ đồ lớp "Phân tích và báo cáo"

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 DTOs:** `LearningStatsFilters` và `ProgramReportFilters` được sử dụng để định nghĩa phạm vi báo cáo.



### 3.7 Thông báo và nhắn tin



Hình 3.3: Sơ đồ lớp cho use-case Thông báo và nhắn tin

Mô tả sơ bộ cho class diagram của use-case "Thông báo và Nhắn tin"

- **API Schemas:** Định nghĩa chuẩn cấu trúc dữ liệu truyền nhận trong API, bao gồm các đối tượng như *Group* (nhóm), *JoinLink* (liên kết tham gia nhóm), *Conversation* (cuộc hội thoại), *Message* (tin nhắn), và *Announcement* (thông báo). Mục đích là đảm bảo tính nhất quán và rõ ràng trong giao tiếp giữa client và server.
  - **API Endpoints:** Cung cấp các điểm truy cập RESTful cho các chức năng chính như tạo và quản lý nhóm, gửi và nhận tin nhắn, xem lịch sử hội thoại, tạo và quản lý thông báo. Các endpoint này xử lý xác thực đầu vào, ánh xạ và chuyển tiếp dữ liệu đến các lớp nghiệp vụ thích hợp.
  - **Chat Services:** Chịu trách nhiệm thực thi nghiệp vụ liên quan tới chat và nhóm, bao gồm tạo nhóm, sinh liên kết mời tham gia, tìm kiếm các cuộc hội thoại theo từ khóa, gửi tin nhắn, và phát thông báo đến nhóm hoặc cá nhân. Các service này chứa logic nghiệp vụ cốt lõi và phối hợp với các repository để lưu trữ hoặc lấy dữ liệu.
  - **Notification Service:** Là lớp dịch vụ độc lập chuyên biệt để quản lý và phân phối các thông báo hay tin nhắn đến đúng người nhận, đảm bảo tính kịp thời và chính xác trong việc truyền tải thông tin.
  - **Domain Model:** Bao gồm các lớp biểu diễn thực thể dữ liệu lõi trong hệ thống, như *Conversation*, *Message*, *Group*, *JoinLink*, *Announcement*, và *Notification*. Các lớp này định nghĩa thuộc tính dữ liệu và chứa các phương thức nội bộ phục vụ cho việc quản lý trạng thái, các thao tác cập nhật hoặc kiểm tra tính hợp lệ.

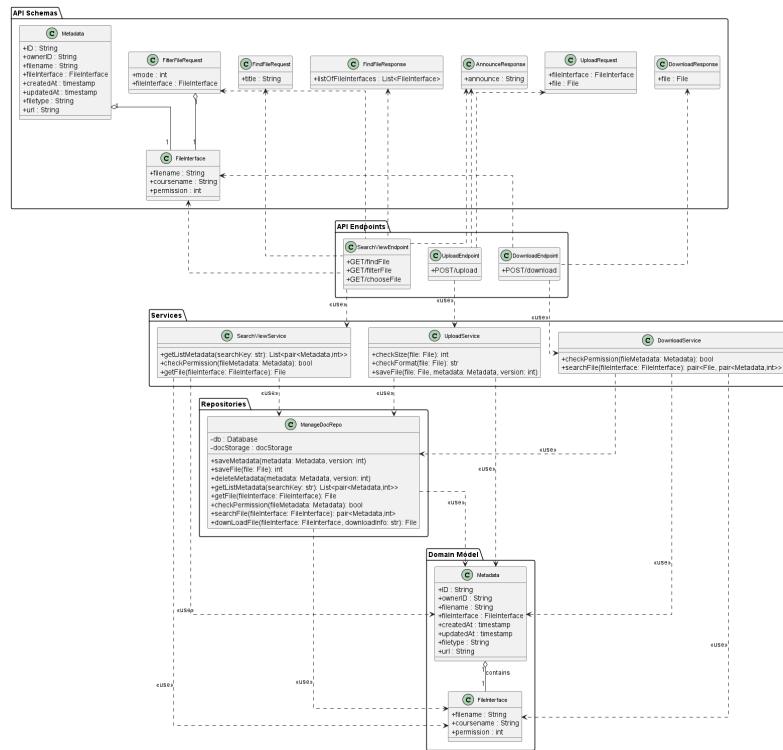


- **Domain Repositories:** Định nghĩa các interface (giao diện) cung cấp các phương thức truy vấn và thao tác dữ liệu cho từng thư mục domain, phục vụ cho các service tương ứng. Việc tách biệt này giúp đảm bảo nguyên tắc đóng mở và hỗ trợ dễ dàng thay đổi trong triển khai cụ thể mà không ảnh hưởng logic nghiệp vụ.
- **Infrastructure Repositories:** Các cài đặt cụ thể của các interface repository ở lớp domain, sử dụng MongoDB để lưu trữ, tìm kiếm và cập nhật dữ liệu thực tế. Điều này đóng vai trò làm cầu nối giữa domain và hệ thống lưu trữ vật lý bên ngoài.
- **Mối liên hệ giữa các lớp:**
  - + Các *API Endpoints* nhận request từ client và gọi tới các *Chat Services* tương ứng.
  - + Các service này sử dụng *Domain Repositories* để truy xuất hoặc cập nhật dữ liệu, đồng thời gọi *Notification Service* để gửi thông báo hoặc tin nhắn đến người dùng.
  - + Lớp *Infrastructure Repositories* triển khai thực tế các thao tác dữ liệu của các repository interfaces.
  - + *Domain Models* liên kết chặt chẽ với các repository, định nghĩa các thực thể và hành vi xử lý nội bộ dữ liệu.

### 3.8 Quản lý tài liệu

«««< HEAD

=====



Hình 3.4: 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.

- **Mối liên hệ giữa các lớp đối tượng:**

- 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.

»»»> 29bfdb1dbef7ca0fa62cd724fbf39e48315dd64

## 4 Development view

### 4.1 Quản lý truy cập và xác thực



#### 4.2 Đăng ký chương trình

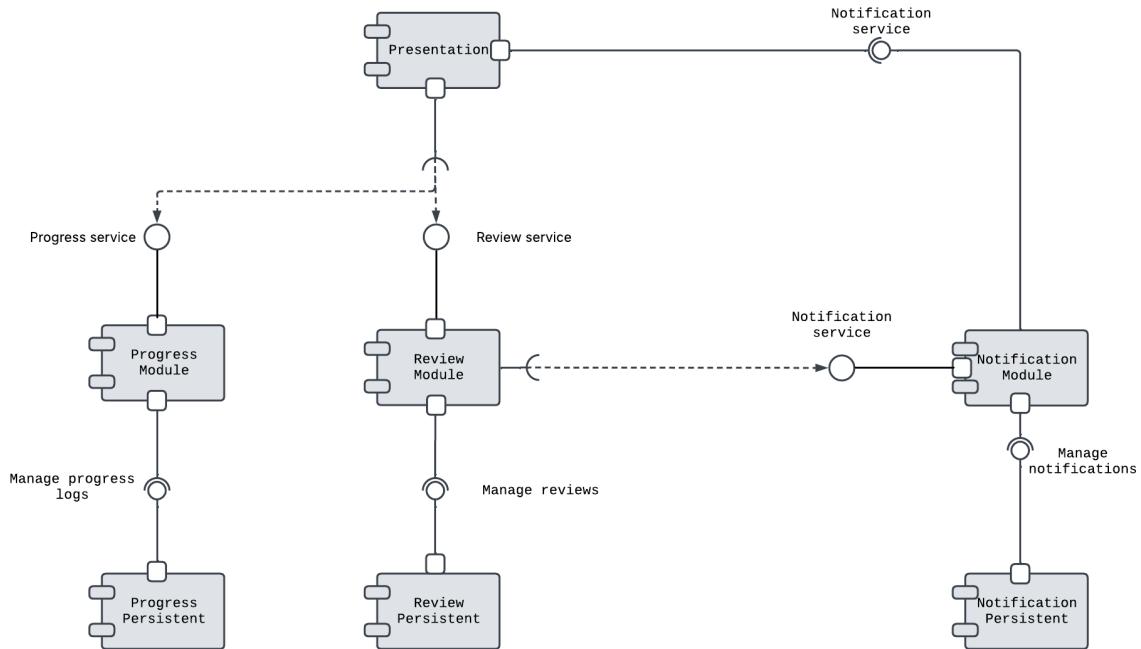


#### 4.3 Tạo chương trình học



#### 4.4 Thiết lập lịch trình cho sinh viên

## 4.5 Đánh giá



Hình 4.1: Sơ đồ thành phần cho use-case "Đánh giá"

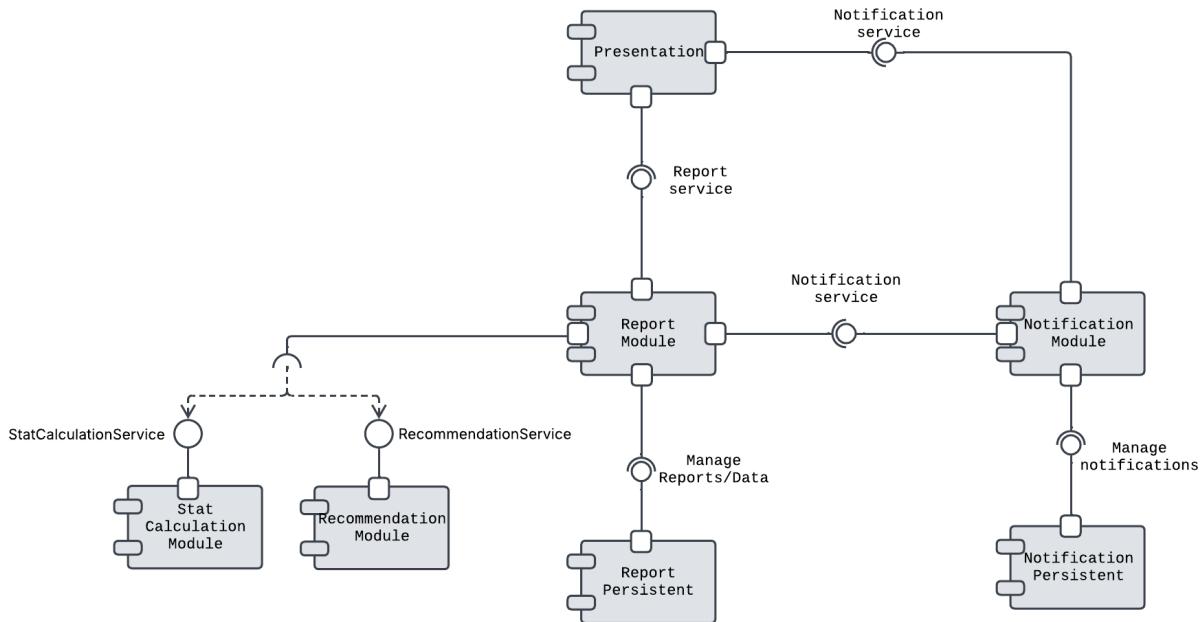
### Mô tả sơ đồ thành phần "Đánh giá"

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).



## 4.6 Phân tích và báo cáo

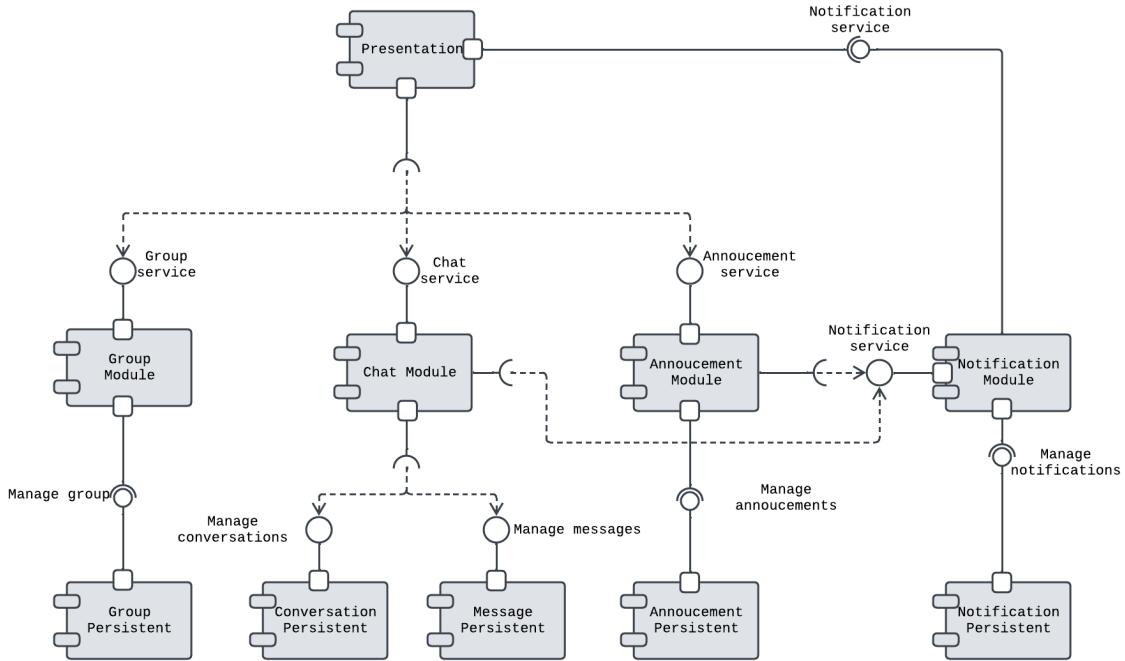


Hình 4.2: Sơ đồ thành phần cho use-case "Phân tích và báo cáo"

#### 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.

#### 4.7 Thông báo và nhắn tin



Hình 4.3: 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 **Annoucement 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.



#### 4.8 Quản lý tài liệu

## 5 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%