

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

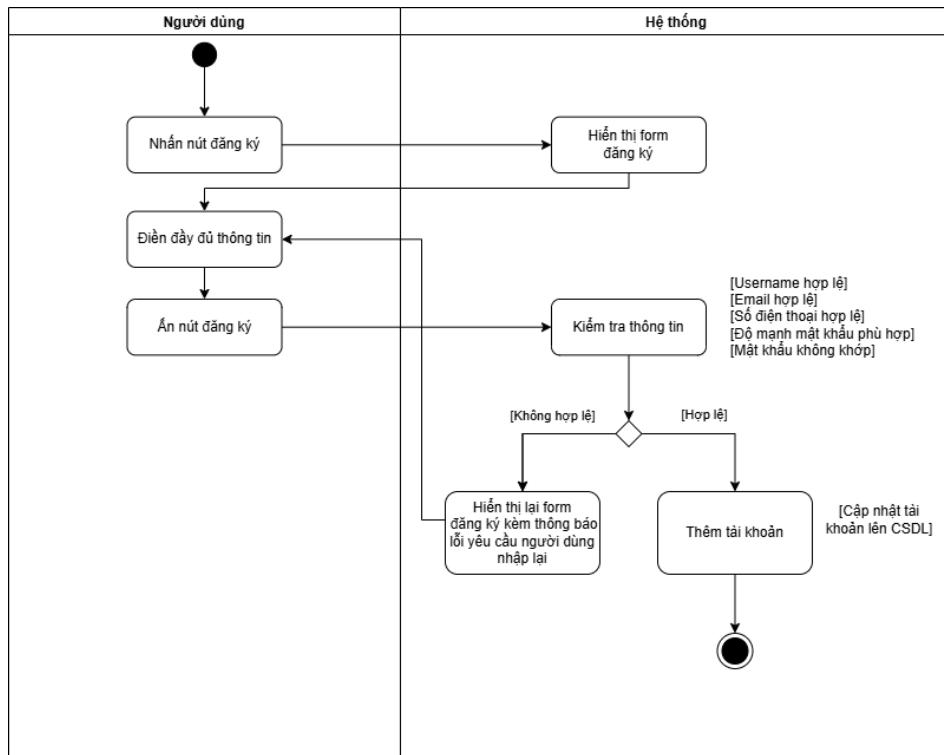
|  |           |
|--|-----------|
| <b>1 Activity Diagram</b>                        | <b>4</b>  |
| 1.1 Quản lý truy cập và xác thực . . . . .       | 4         |
| 1.2 Đăng ký chương trình . . . . .               | 10        |
| 1.3 Tạo chương trình học . . . . .               | 14        |
| 1.4 Thiết lập lịch trình cho sinh viên . . . . . | 20        |
| 1.5 Đánh giá . . . . .                           | 23        |
| 1.6 Phân tích và báo cáo . . . . .               | 25        |
| 1.7 Thông báo và nhắn tin . . . . .              | 27        |
| 1.8 Quản lý tài liệu . . . . .                   | 32        |
| <b>2 Sequence Diagrams</b>                       | <b>35</b> |
| 2.1 Quản lý truy cập và xác thực . . . . .       | 35        |
| 2.2 Đăng ký chương trình . . . . .               | 42        |
| 2.3 Tạo chương trình học . . . . .               | 45        |
| 2.4 Thiết lập lịch trình cho sinh viên . . . . . | 51        |
| 2.5 Đánh giá . . . . .                           | 54        |
| 2.6 Phân tích và báo cáo . . . . .               | 58        |
| 2.7 Thông báo và nhắn tin . . . . .              | 60        |
| 2.8 Quản lý tài liệu . . . . .                   | 65        |
| <b>3 Class Diagram</b>                           | <b>68</b> |
| 3.1 Quản lý truy cập và xác thực . . . . .       | 69        |
| 3.2 Đăng ký chương trình . . . . .               | 70        |
| 3.3 Tạo chương trình học . . . . .               | 71        |
| 3.4 Thiết lập lịch trình cho sinh viên . . . . . | 72        |
| 3.5 Đánh giá . . . . .                           | 73        |
| 3.6 Phân tích và báo cáo . . . . .               | 74        |
| 3.7 Thông báo và nhắn tin . . . . .              | 75        |
| 3.8 Quản lý tài liệu . . . . .                   | 76        |
| <b>4 Development view</b>                        | <b>78</b> |
| 4.1 Quản lý truy cập và xác thực . . . . .       | 78        |
| 4.2 Đăng ký chương trình . . . . .               | 79        |
| 4.3 Tạo chương trình học . . . . .               | 80        |
| 4.4 Thiết lập lịch trình cho sinh viên . . . . . | 81        |
| 4.5 Đánh giá . . . . .                           | 82        |
| 4.6 Phân tích và báo cáo . . . . .               | 83        |
| 4.7 Thông báo và nhắn tin . . . . .              | 84        |
| 4.8 Quản lý tài liệu . . . . .                   | 86        |
| <b>5 Bảng phân công công việc</b>                | <b>87</b> |

## Revision History

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

## 1 Activity Diagram

### 1.1 Quản lý truy cập và xác thực

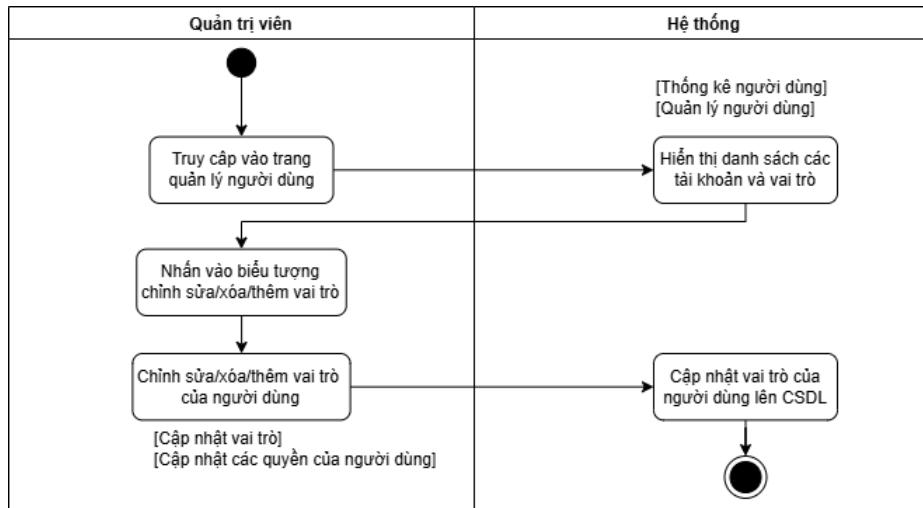


Hình 1.1: Sơ đồ hoạt động cho use-case "Đăng ký"

#### Mô tả hoạt động Activity Diagram: Đăng ký

1. **Người dùng** chọn nút “Đăng ký”.
2. **Hệ thống** hiển thị form đăng ký.
3. **Người dùng** nhập đầy đủ các thông tin yêu cầu như tên, email, số điện thoại, username và mật khẩu.
4. **Người dùng** nhấn nút “Đăng ký”.
5. **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.
6. Nếu thông tin không hợp lệ: hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.

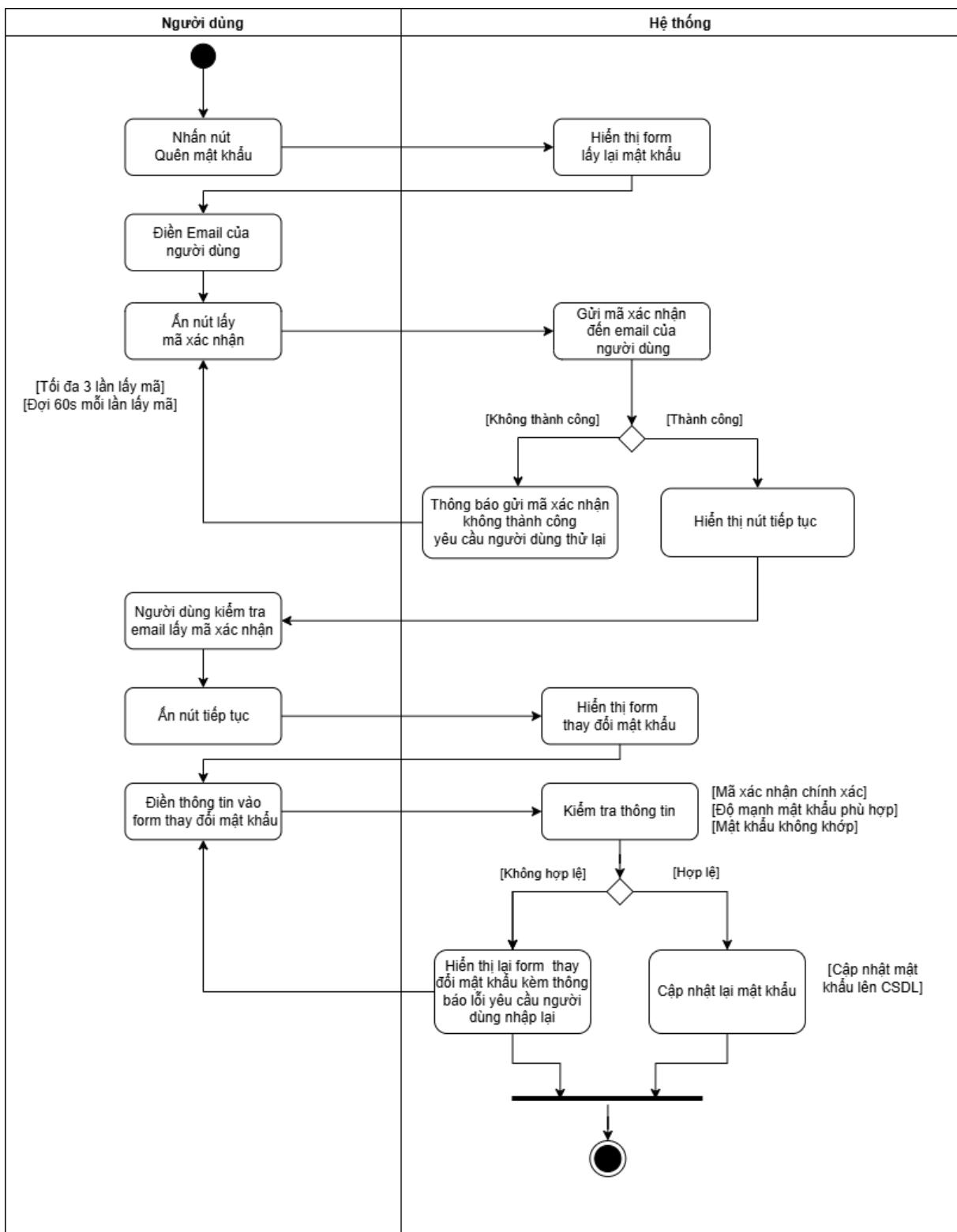
7. Nếu hợp lệ: hệ thống tạo tài khoản mới và lưu vào cơ sở dữ liệu.



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

### Mô tả hoạt động Activity Diagram: Quản lý tài khoản

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

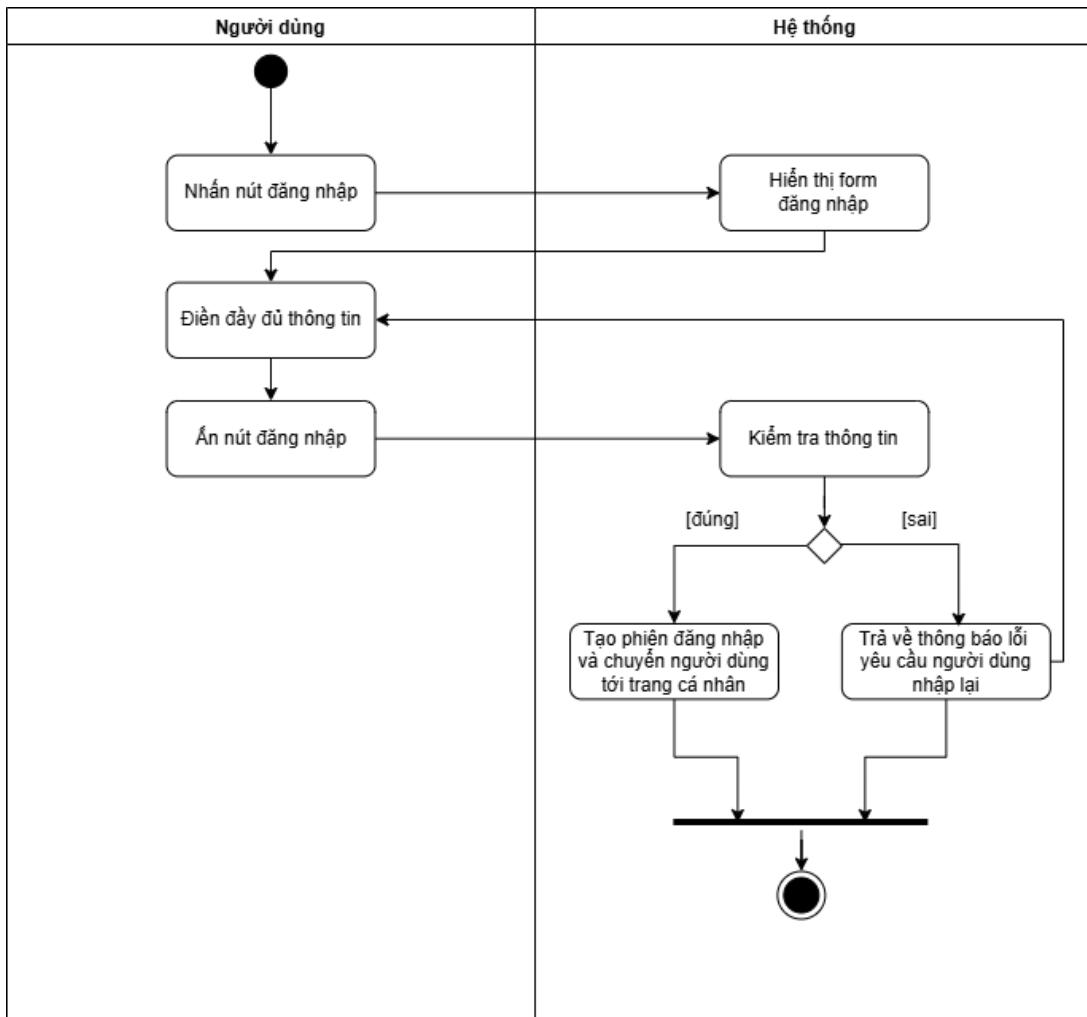


Hình 1.3: Sơ đồ hoạt động cho use-case "Đặt lại mật khẩu"



### Mô tả hoạt động Activity Diagram: Đặt lại mật khẩu

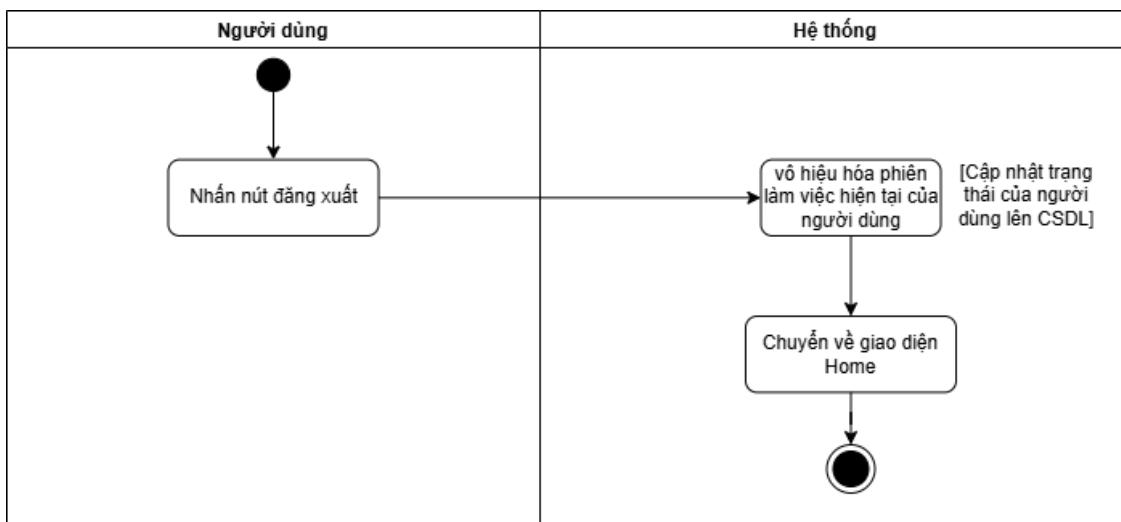
1. **Người dùng** nhấn nút “Quên mật khẩu”.
2. **Hệ thống** hiển thị form yêu cầu nhập email.
3. **Người dùng** điền email của mình.
4. **Người dùng** nhấn nút “Lấy mã xác nhận”.
5. **Hệ thống** gửi mã xác nhận đến email của người dùng.
6. Nếu gửi mã thất bại: hệ thống hiển thị thông báo và yêu cầu người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



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

### Mô tả hoạt động Activity Diagram: Đăng nhập

1. **Người dùng** nhấn nút “Đăng nhập” trên giao diện hệ thống.
2. **Hệ thống** hiển thị form đăng nhập.
3. **Người dùng** nhập thông tin đăng nhập gồm email/username và mật khẩu.
4. **Người dùng** nhấn nút “Đăng nhập”.
5. **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.
6. **Hệ thống** chuyển người dùng đến trang cá nhân.

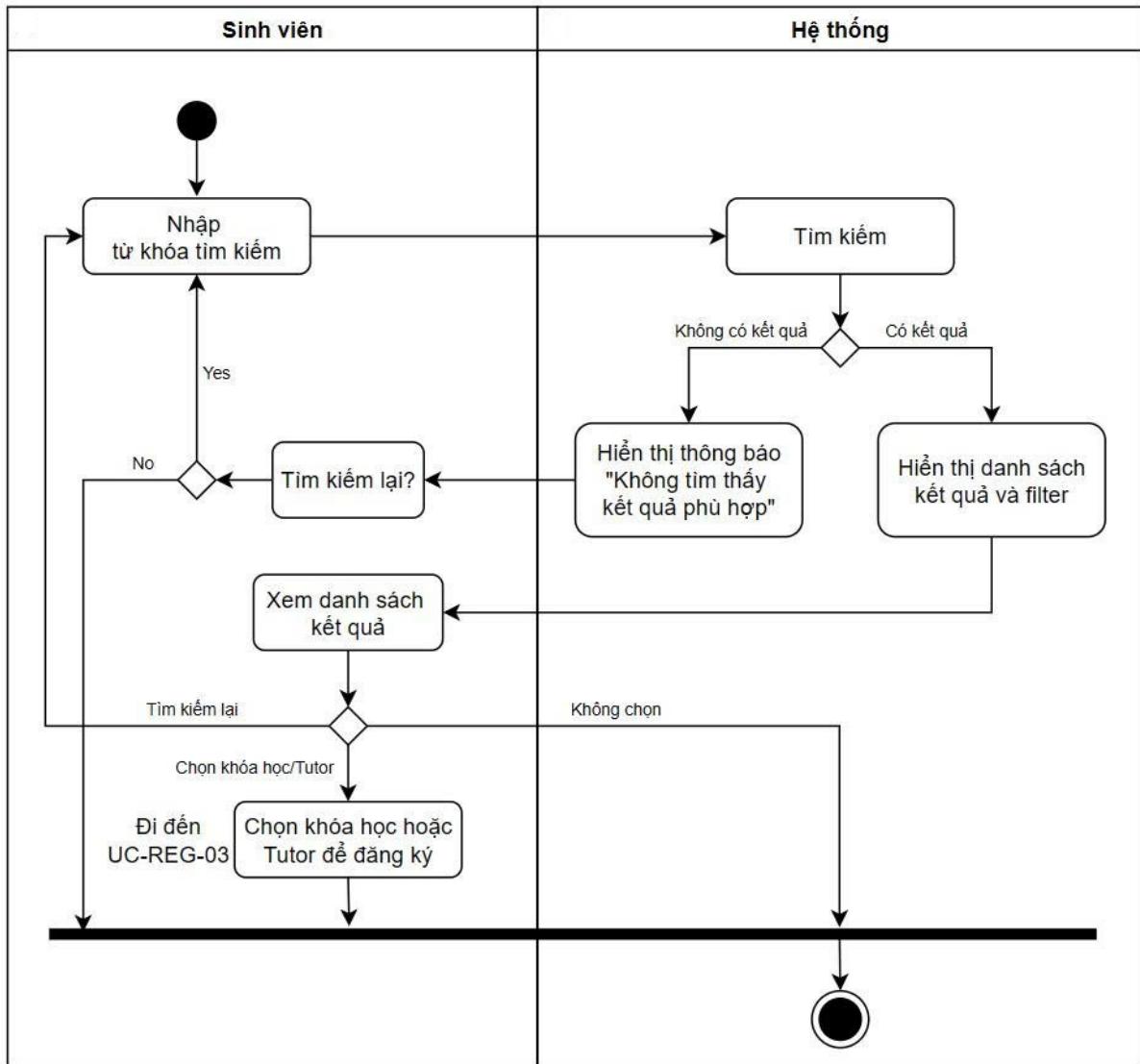


Hình 1.5: Sơ đồ hoạt động cho use-case "Đăng xuất"

### Mô tả hoạt động Activity Diagram: Đăng xuất

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

## 1.2 Đăng ký chương trình



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

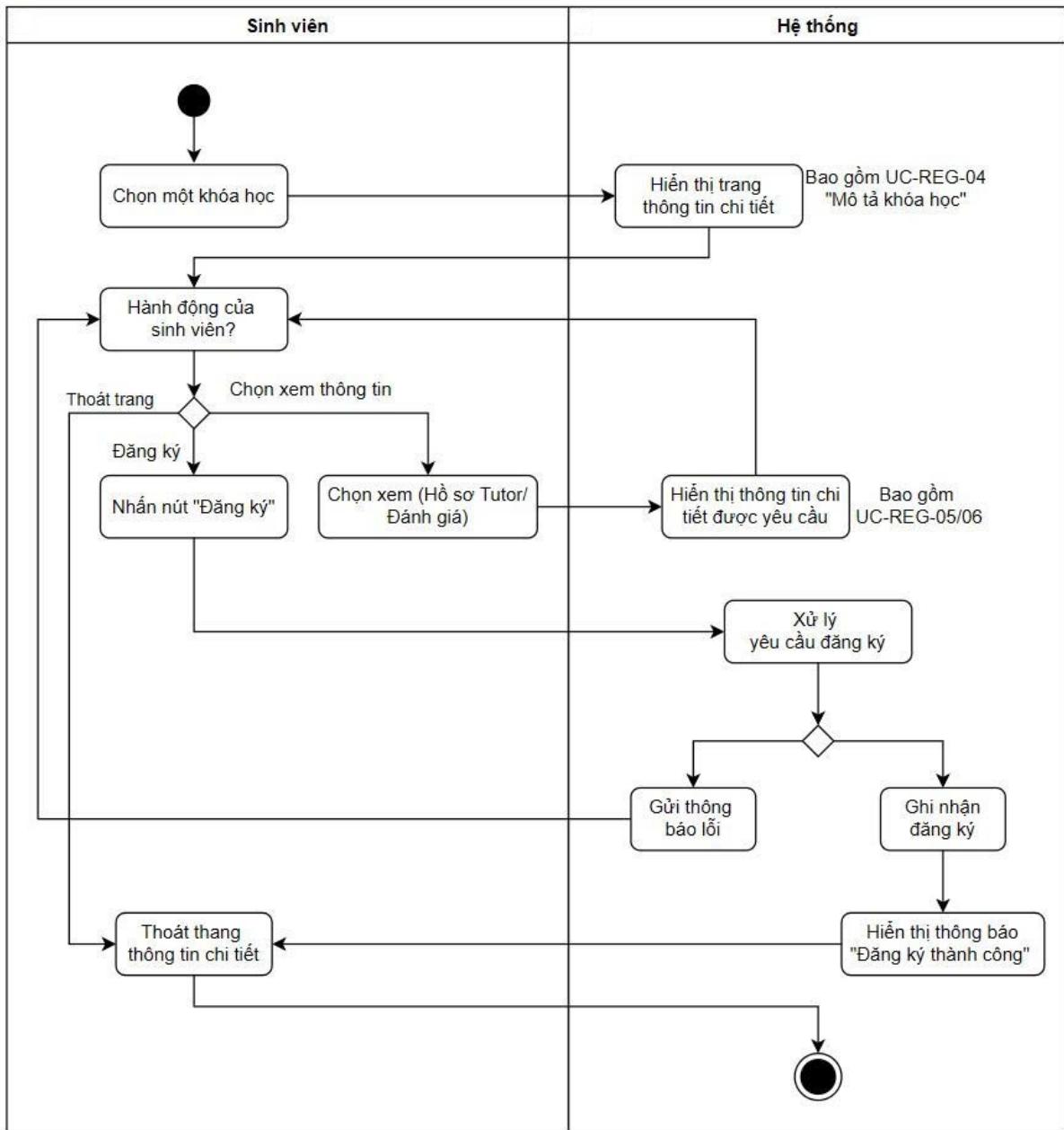
### Đặc tả luồng sự kiện

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

1. **Bắt đầu:** Luồng được kích hoạt khi Sinh viên bắt đầu tìm kiếm.
2. **Nhập liệu:** Sinh viên thực hiện hành động "Nhập từ khóa tìm kiếm".
3. **Hệ thống Xử lý:** Hệ thống tiếp nhận từ khóa và thực hiện nghiệp vụ "Tìm kiếm".
4. **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.
5. **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.
6. **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.
7. **Kết thúc:** Luồng dừng lại khi sinh viên thoát hoặc chuyển sang một Use Case khác (UC-REG-03).



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

### Đặc tả luồng sự kiện

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).



3. **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.

4. **Xử lý Đăng ký:** Hệ thống tiếp nhận yêu cầu và thực hiện "**Xử lý yêu cầu đăng ký**".

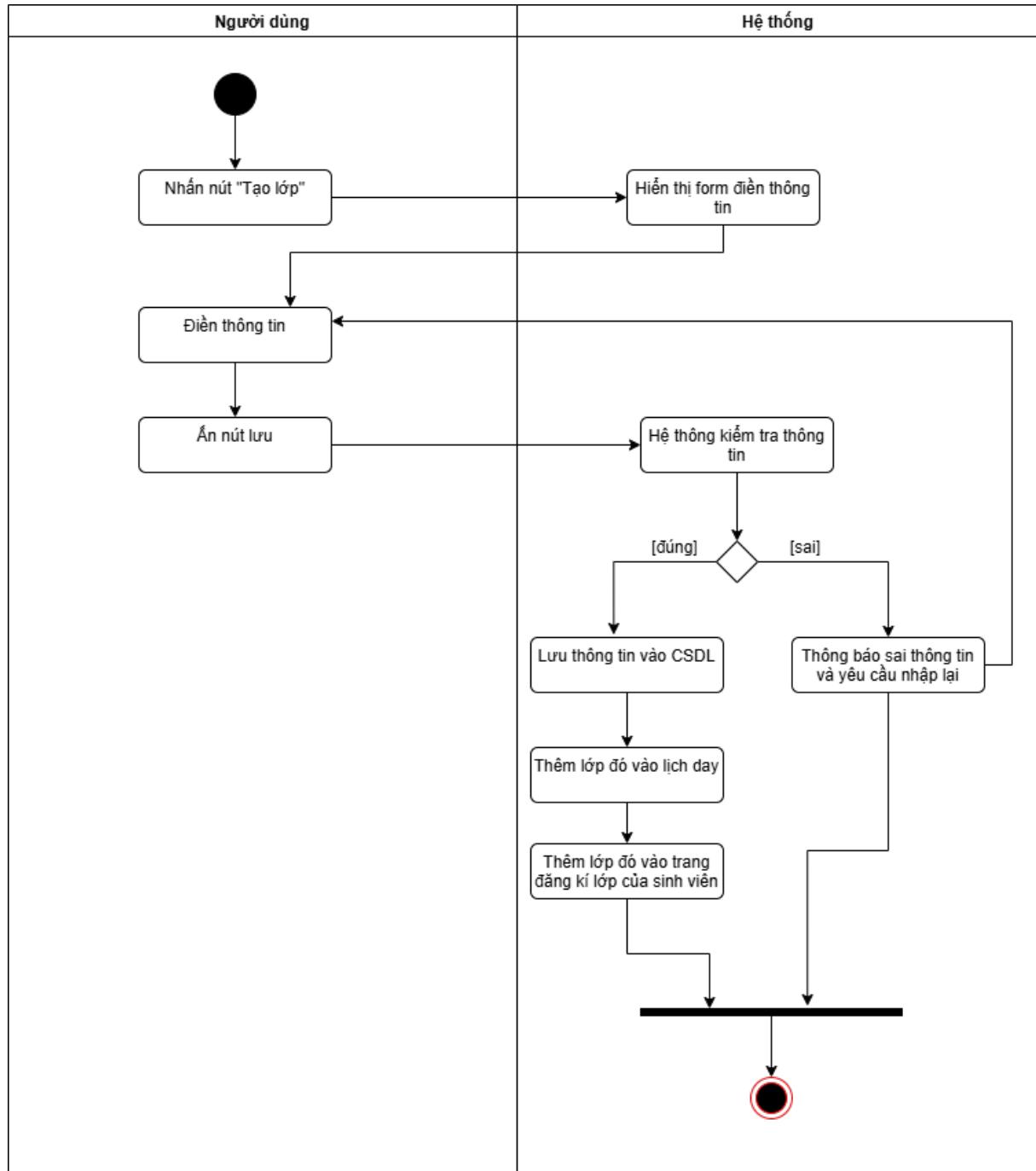
5. **Rẽ nhánh (Kết quả Đăng ký):** Hệ thống kiểm tra kết quả xử lý nghiệp vụ:

- **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.

6. **Hoàn tất:** Hệ thống "**Hiển thị thông báo 'Đăng ký thành công'**" cho Sinh viên.

7. **Kết thúc:** Luồng kết thúc sau khi sinh viên đăng ký thành công.

### 1.3 Tạo chương trình học



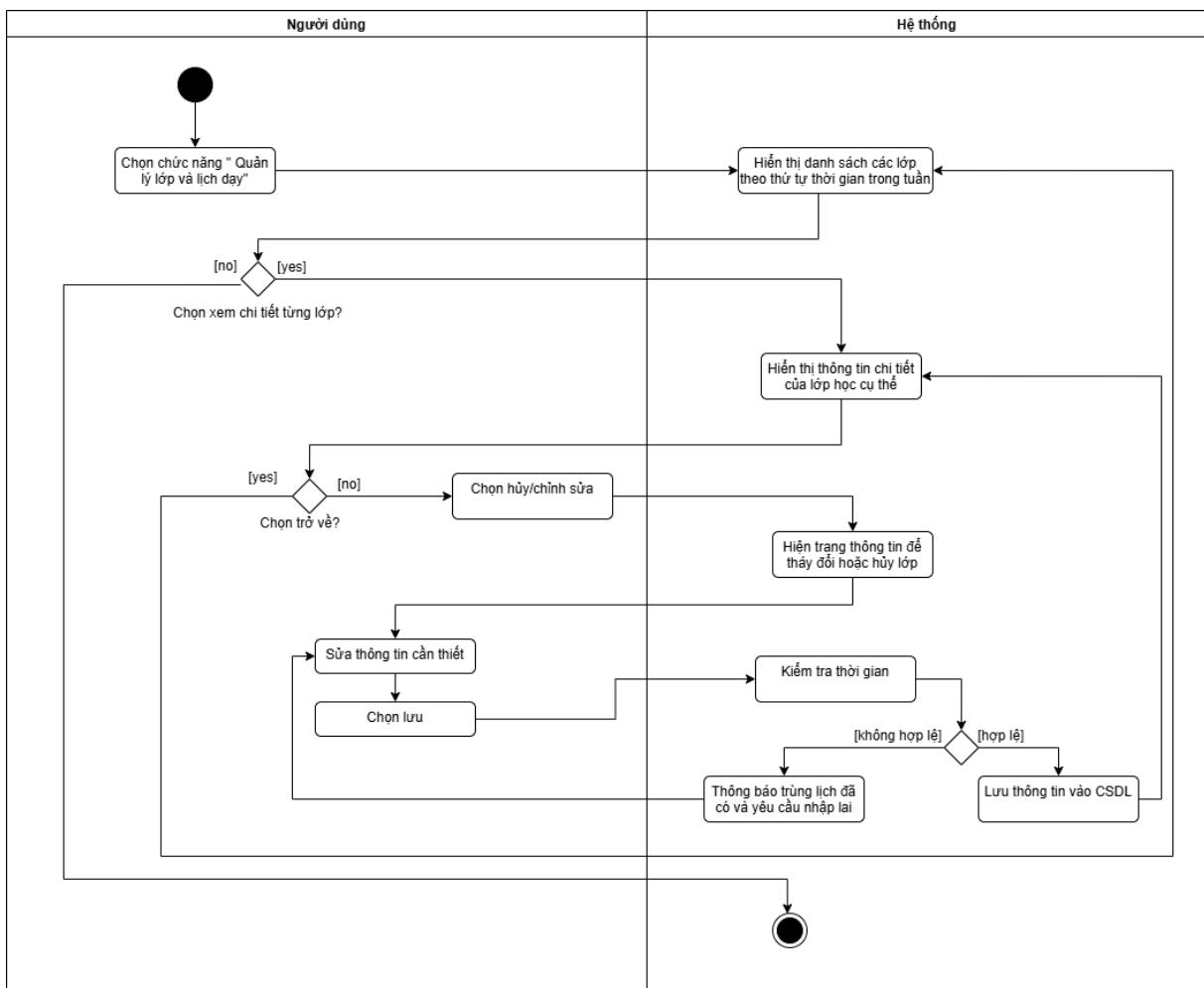
Hình 1.8: Sơ đồ use-case cho chức năng "Tạo lớp học"

Activity diagram - Tạo lớp học mô tả quy trình **tạo lớp học** giữa hai tác nhân: **Người dùng** và **Hệ thống**. Luồng hoạt động diễn ra như sau:

- 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.
- Sau khi hoàn tất, 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.



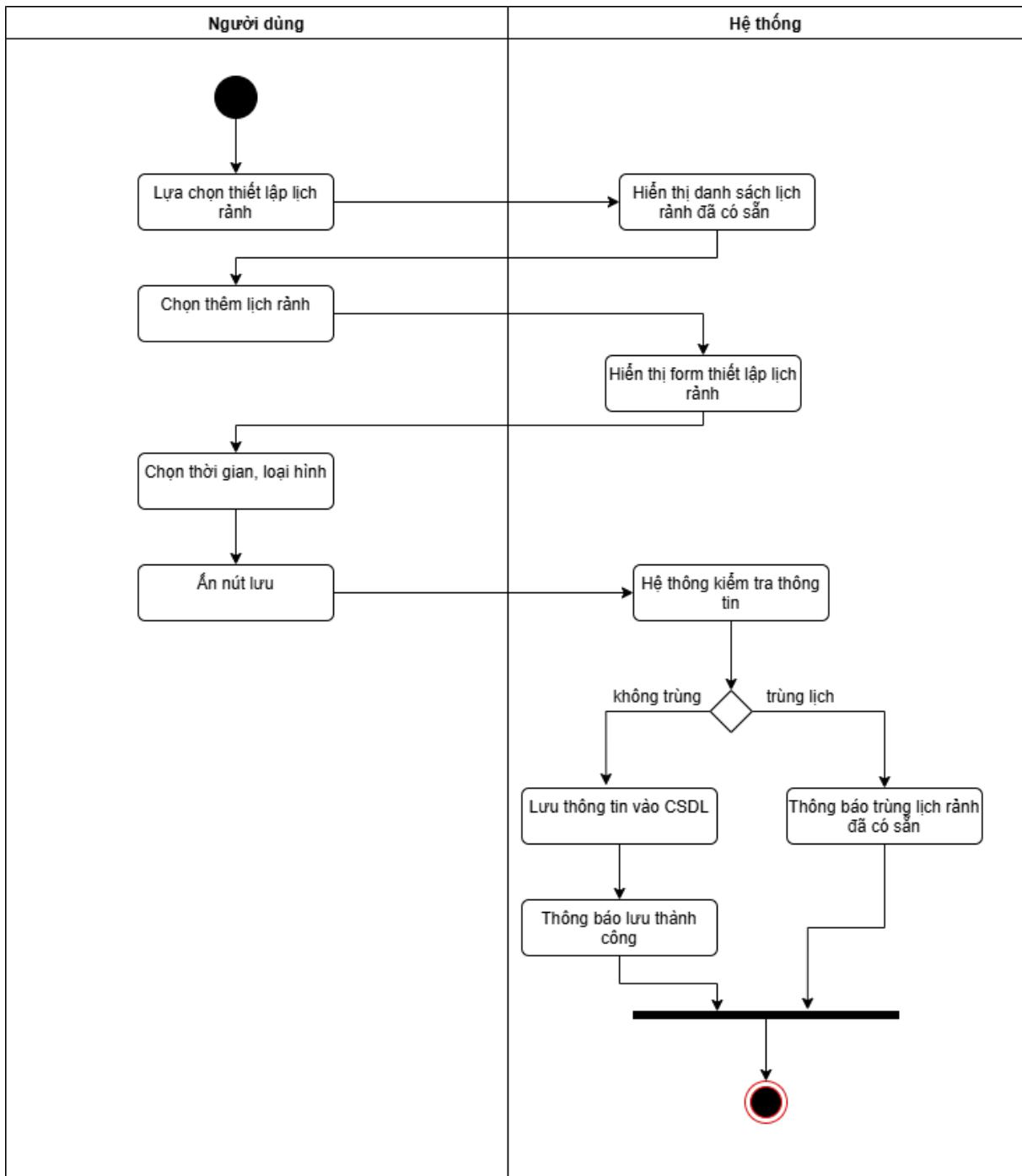
Hình 1.9: Sơ đồ use-case cho chức năng "Quản lý lớp học"

Activity diagram - Quản lý lớp và lịch dạy mô tả quy trình người dùng (tutor) tương tác với hệ thống để quản lý các lớp học và lịch dạy. Quy trình được mô tả như sau:

- 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 trở 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 1.10: Sơ đồ use-case cho chức năng "Lịch rảnh"

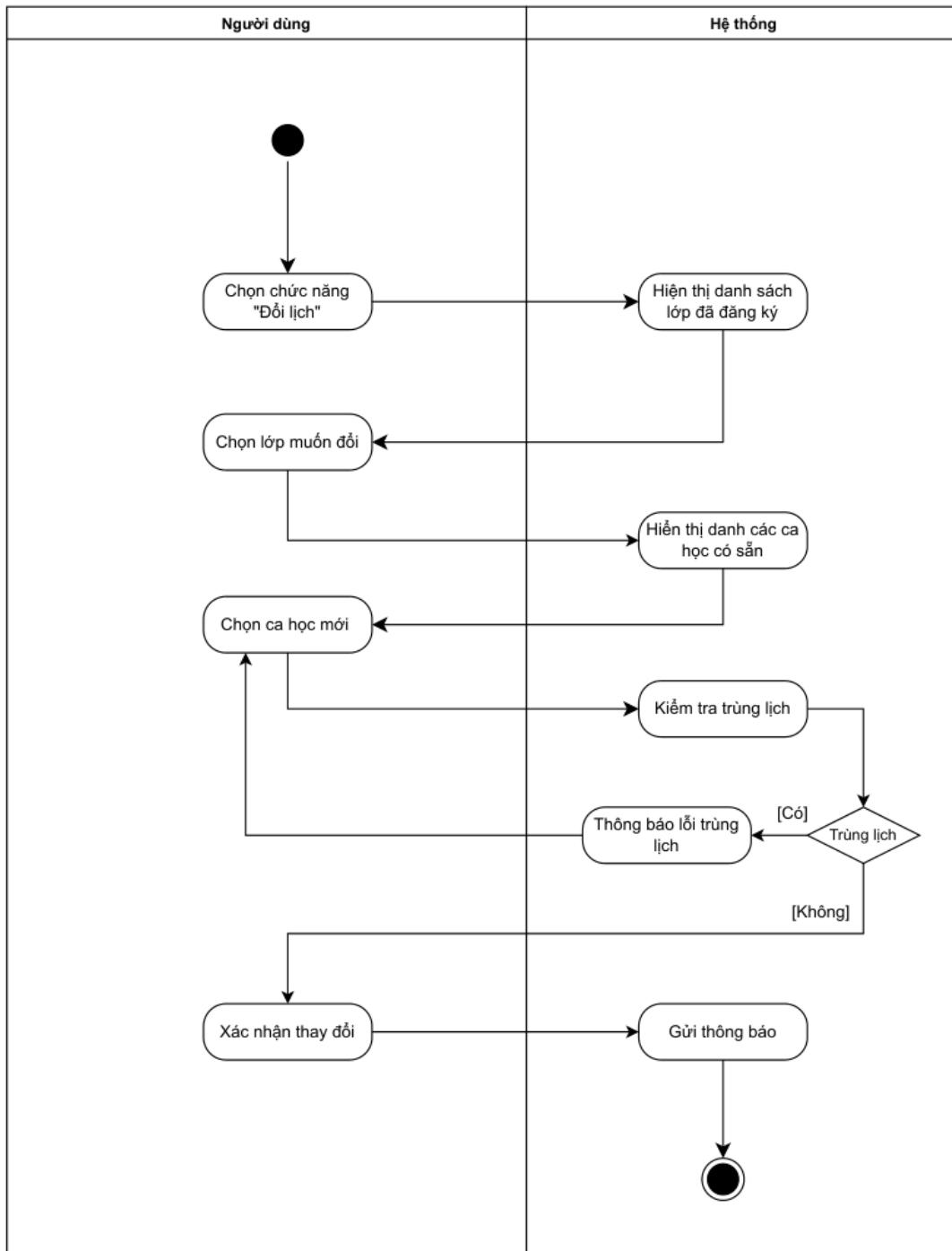
Activity diagram - Thiết lập lịch rảnh mô tả quy trình người dùng thiết lập lịch rảnh thông qua hệ thống. Quy trình bao gồm hai tác nhân chính: **Người dùng** và **Hệ thống**. Luồng hoạt động diễn ra như sau:

- 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.

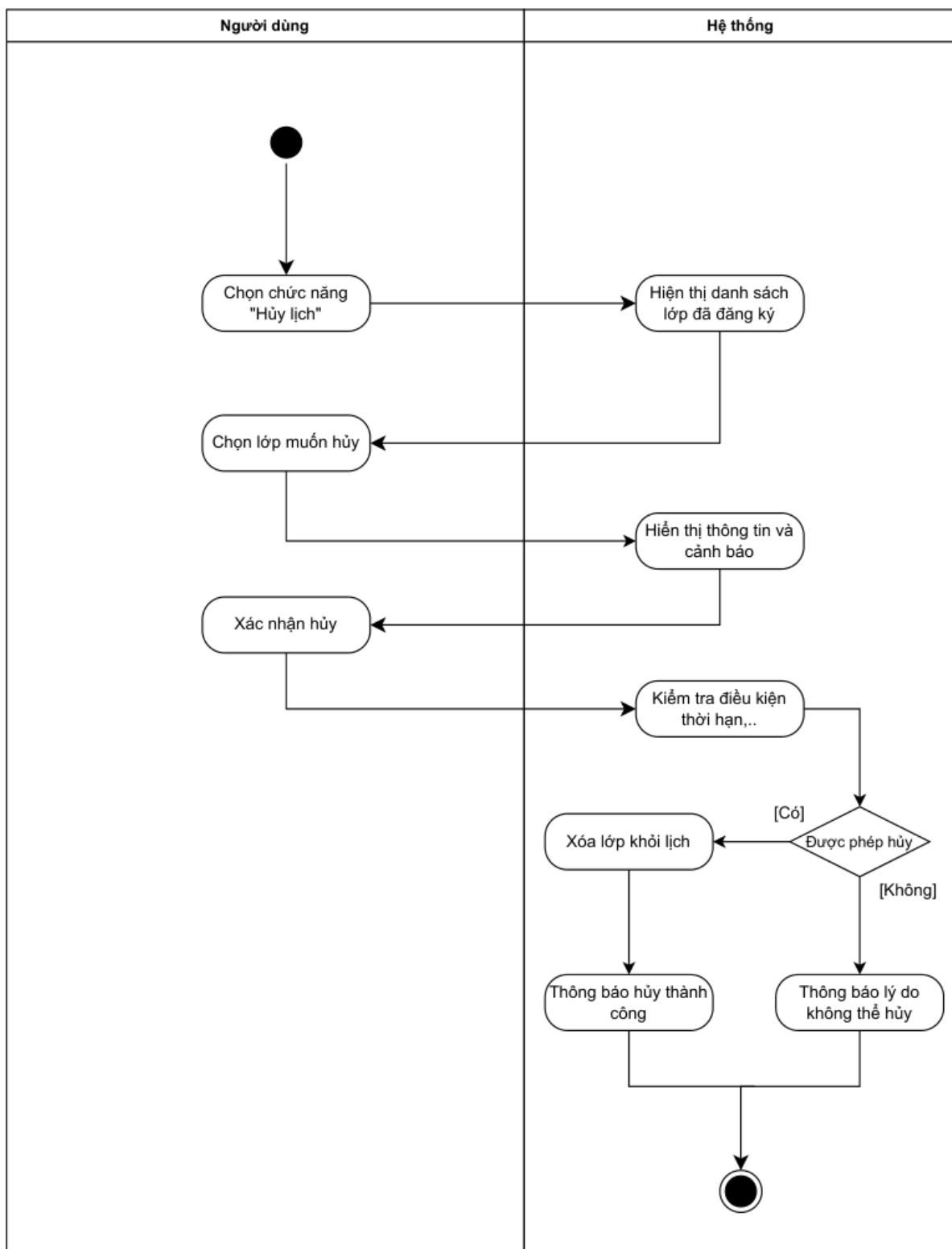


- Người dùng chọn thao tác *thêm lịch rảnh*.
- Hệ thống hiển thị form thiết lập lịch rảnh mới.
- Người dùng chọn thời gian và loại hình phù hợp.
- Người dùng nhấn nút *Lưu*.
- Hệ thống thực hiện kiểm tra thông tin:
  - Nếu không trùng lịch đã tồn tại:
    - \* Hệ thống lưu thông tin vào cơ sở dữ liệu.
    - \* Hệ thống gửi thông báo lưu thành công.
  - Nếu trùng lịch 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.

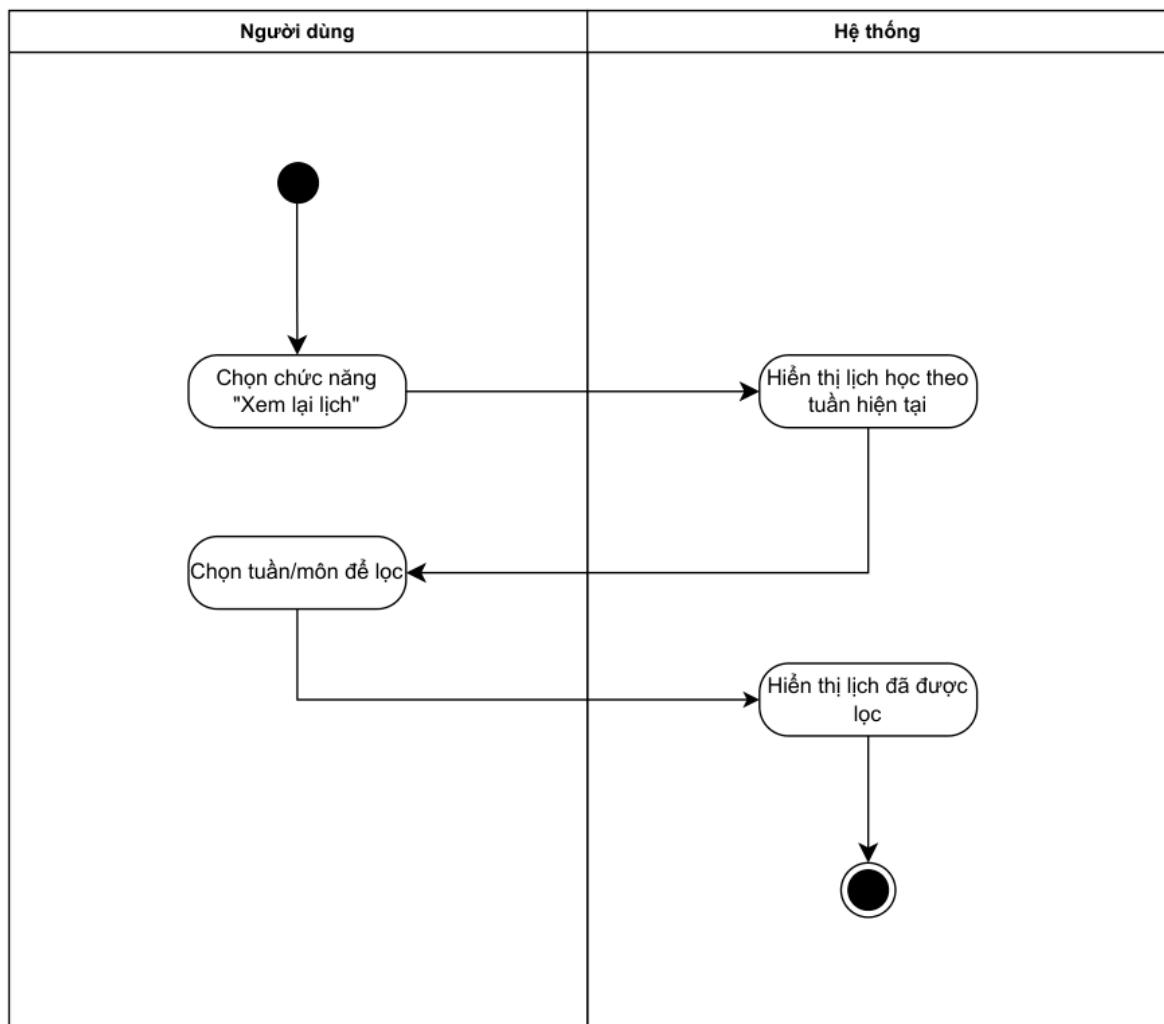
#### 1.4 Thiết lập lịch trình cho sinh viên



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

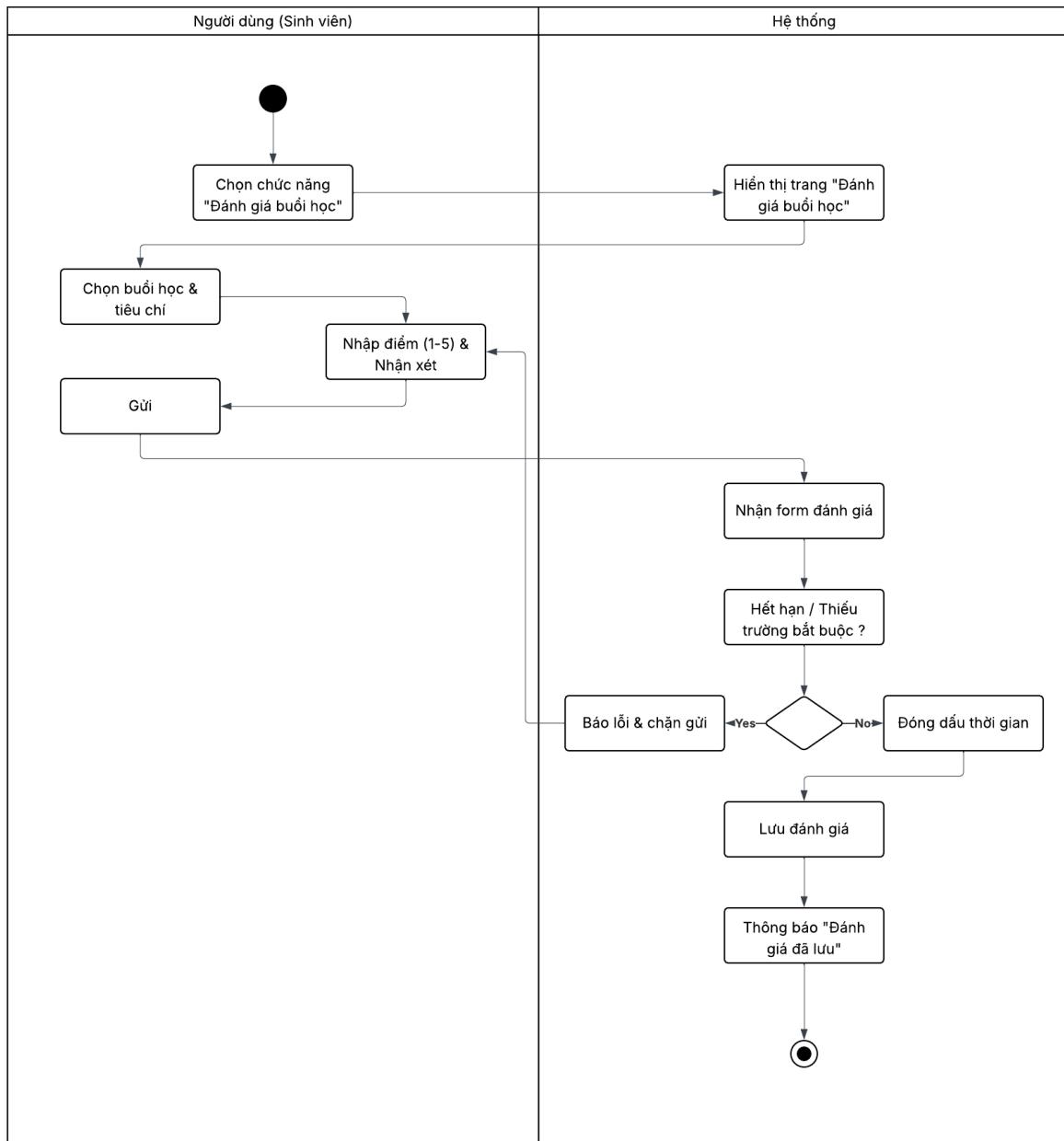


Hình 1.12: Sơ đồ hoạt động cho use-case "Hủy lịch học" cho sinh viên



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

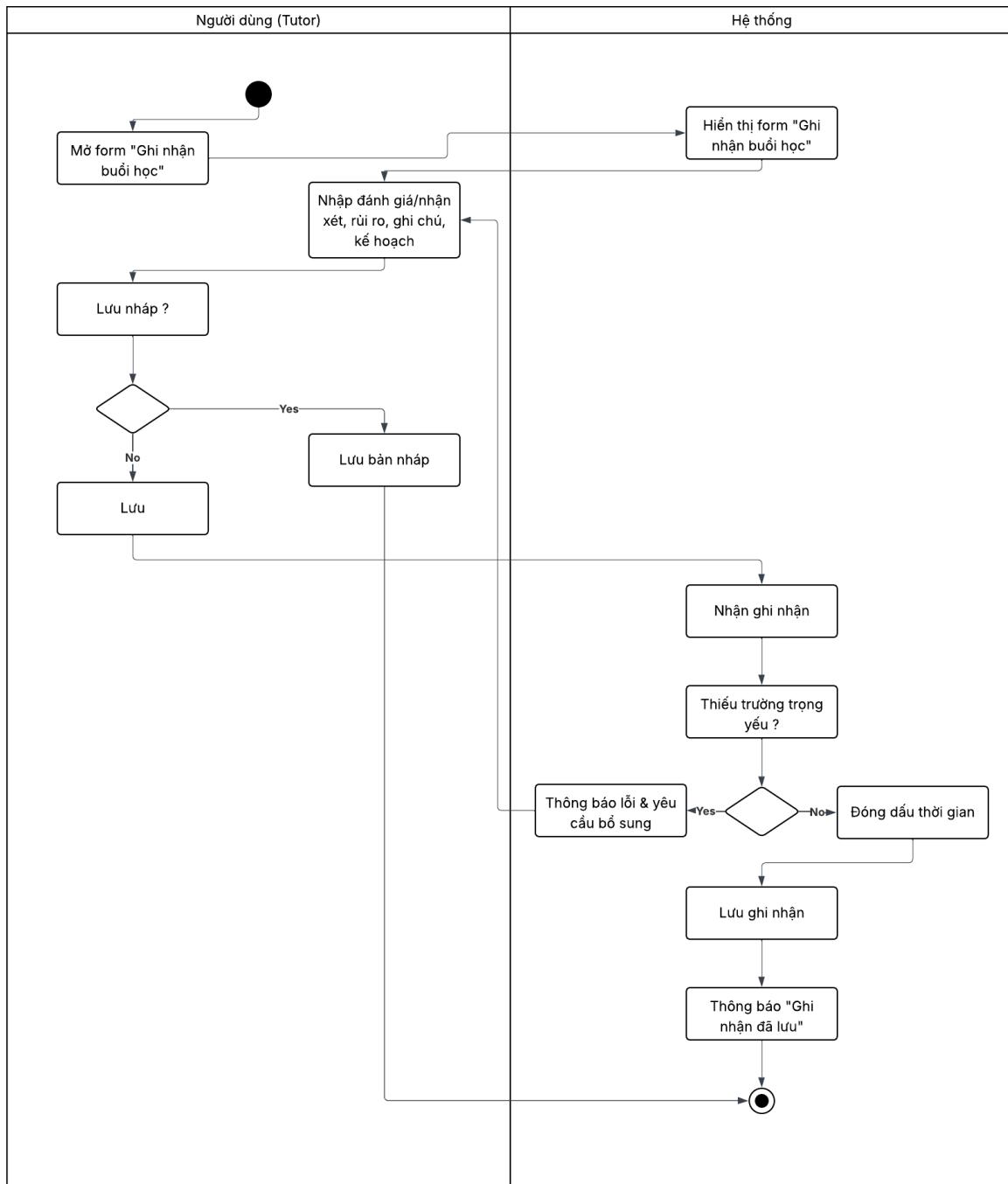
## 1.5 Đánh giá



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.

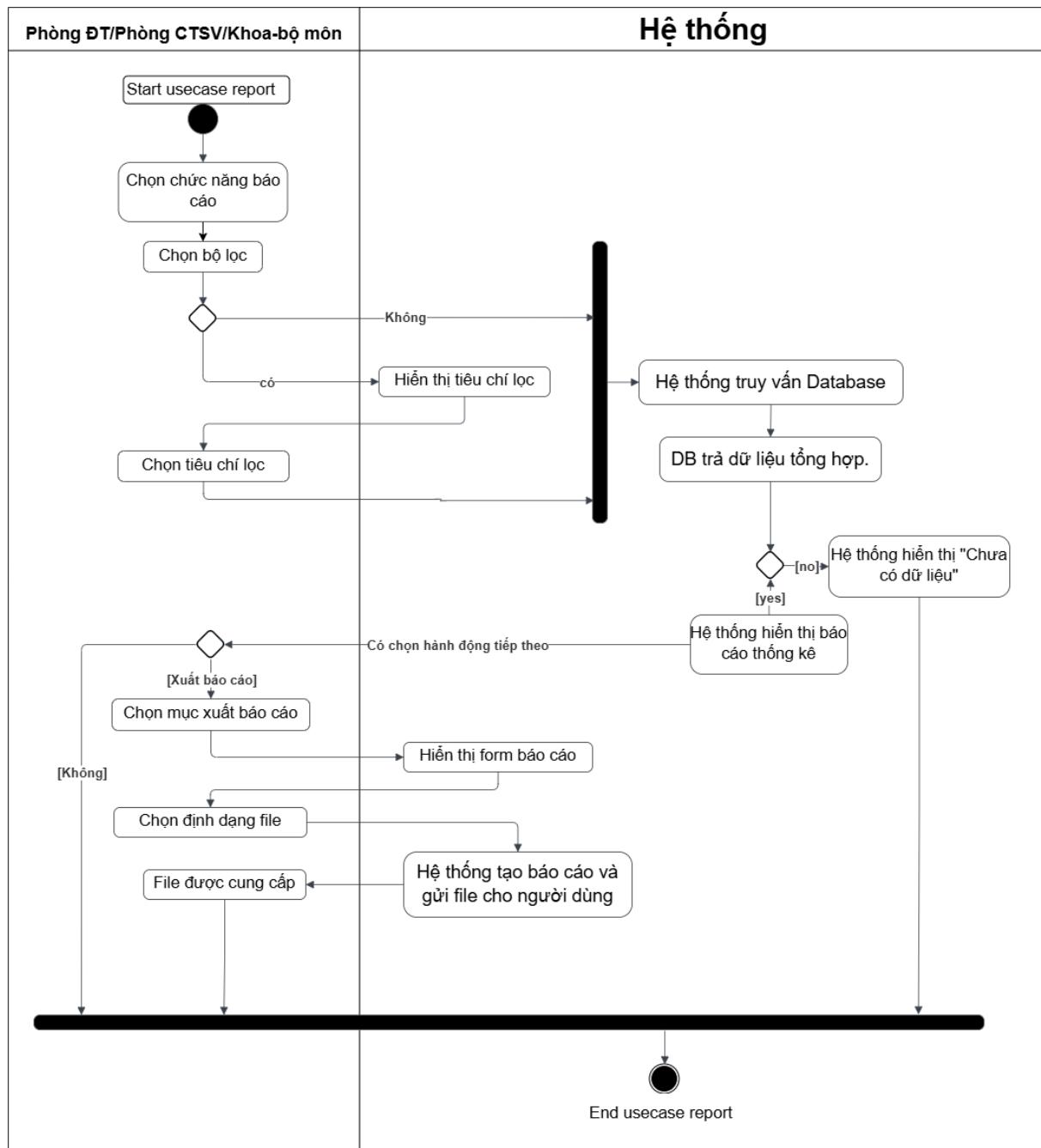


Hình 1.15: Sơ đồ hoạt động cho use-case "Ghi nhận tiến bộ sinh viên" của Tutor

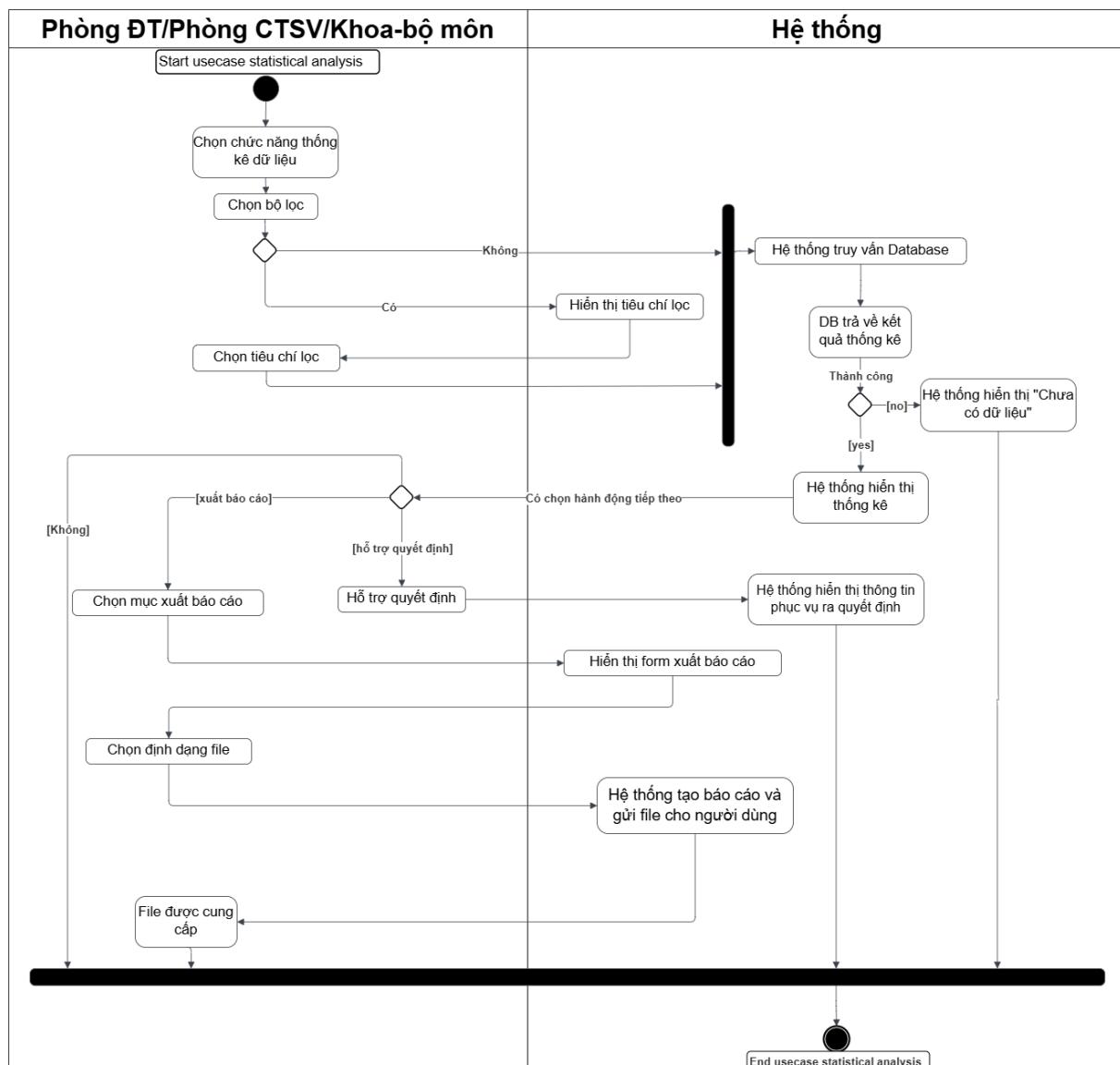
### Mô tả sơ đồ hoạt động "Ghi nhận tiến bộ sinh viên"

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

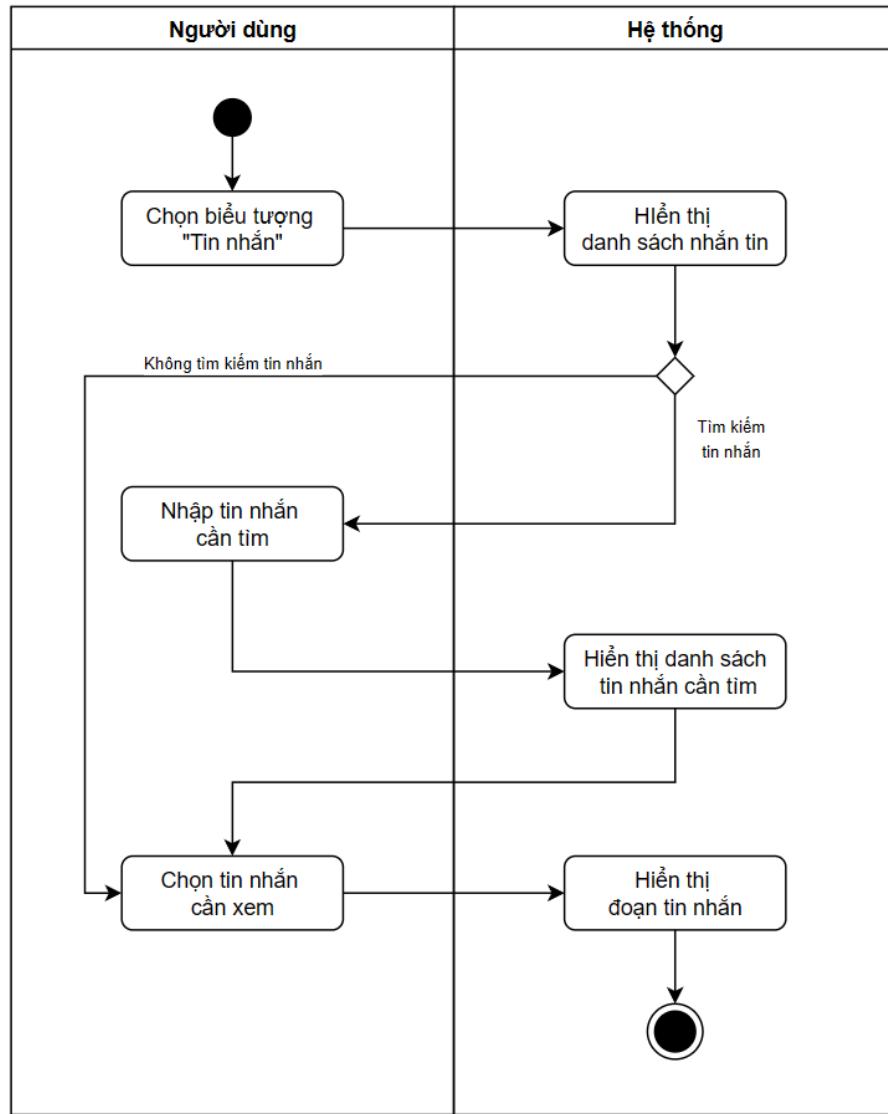


Hình 1.16: Sơ đồ hoạt động cho use-case "Phân tích và báo cáo"



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

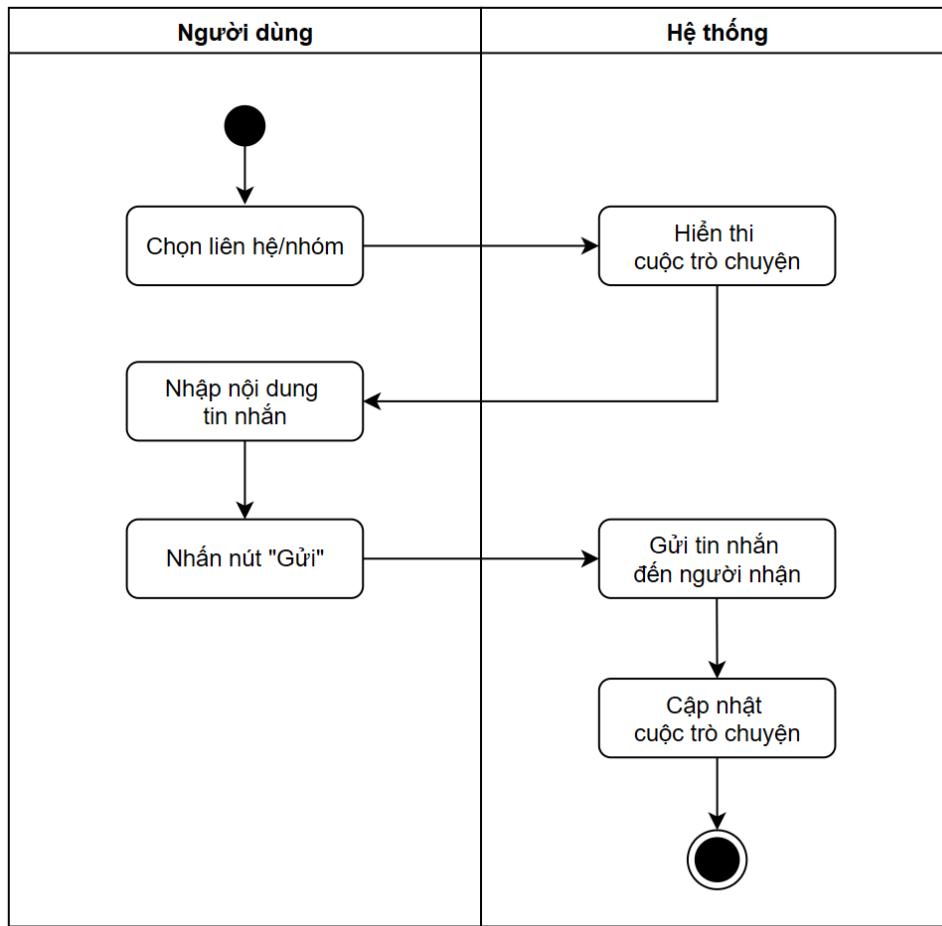
## 1.7 Thông báo và nhắn tin



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

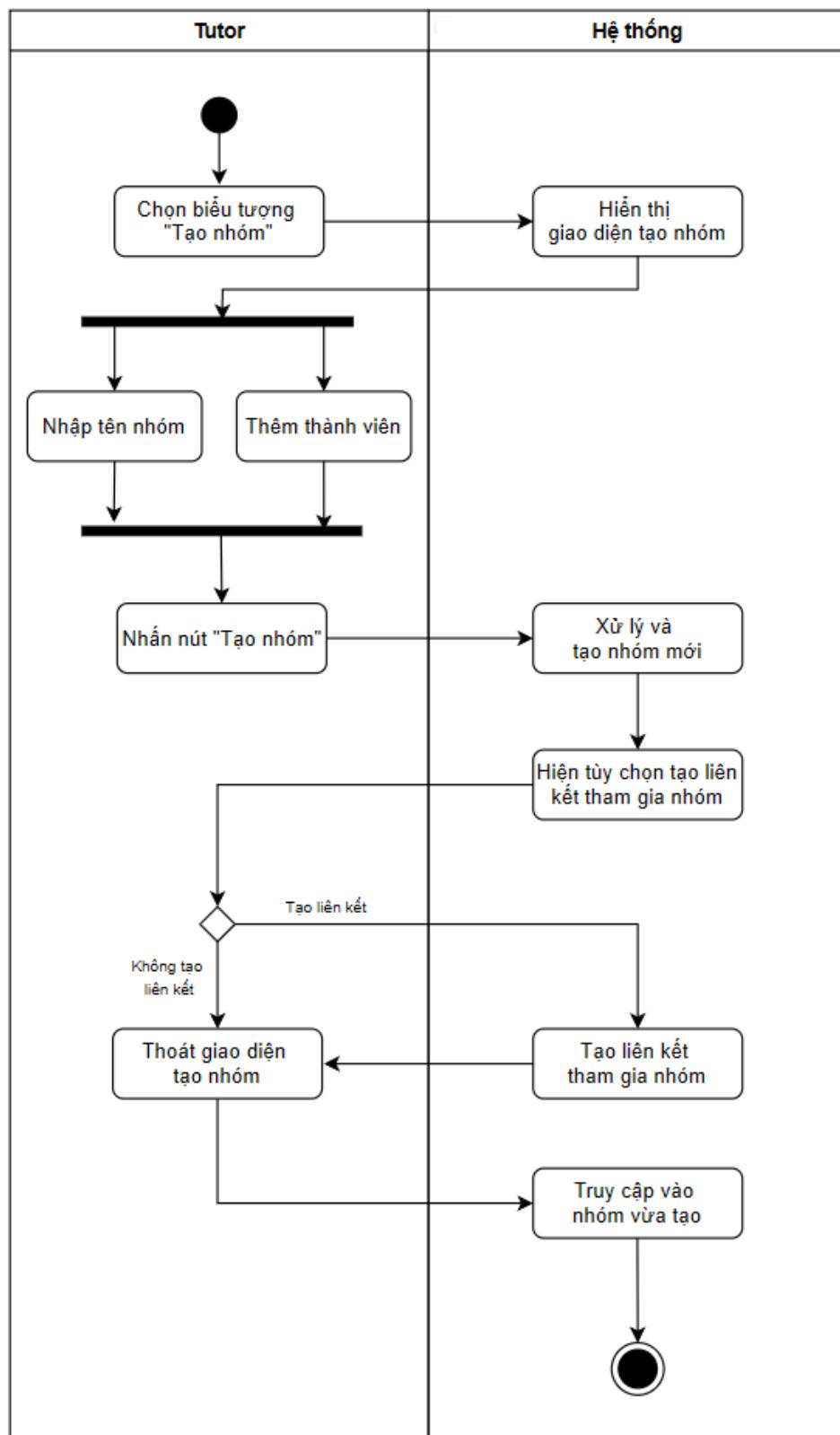
1. **Người dùng** chọn biểu tượng "Tin nhắn" trên giao diện ứng dụng.
2. **Hệ thống** hiển thị danh sách các cuộc trò chuyện (nhắn tin) hiện có.
3. Nếu **người dùng** muốn tìm tin nhắn, bấm vào thanh tìm kiếm và nhập từ khóa cần tìm.
4. **Hệ thống** truy vấn, trả về danh sách các tin nhắn phù hợp với 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.

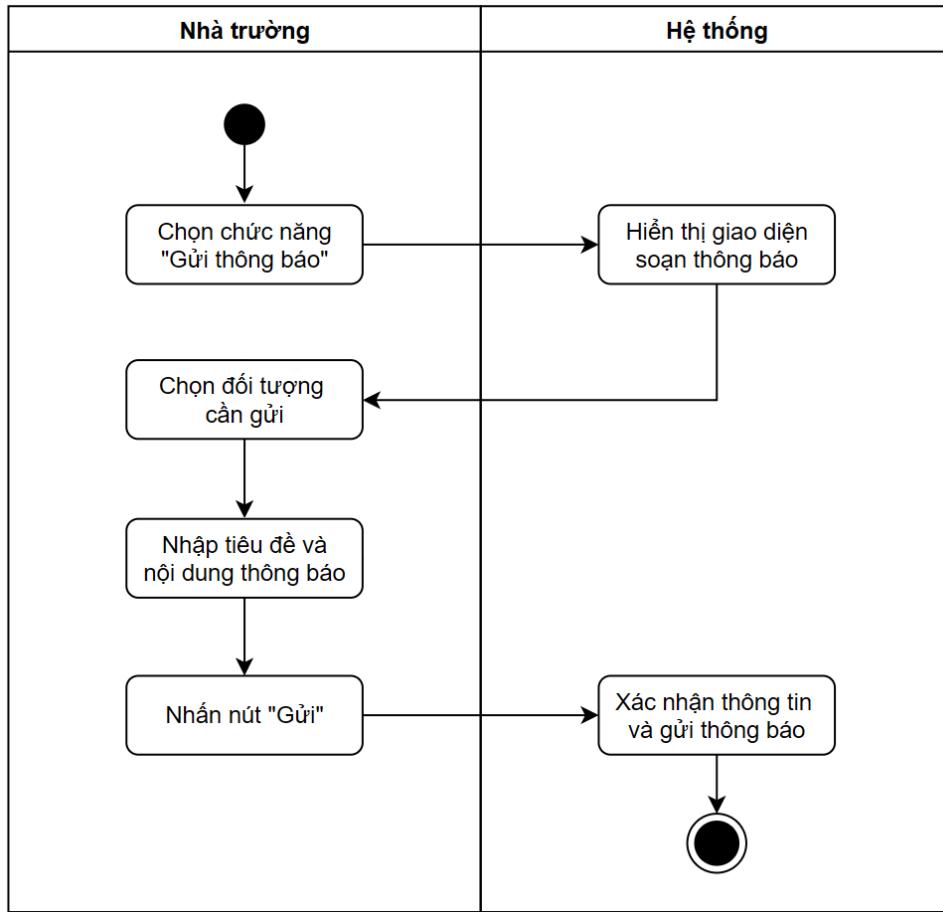


Hình 1.20: Sơ đồ hoạt động cho use-case "Tạo nhóm"



### 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.

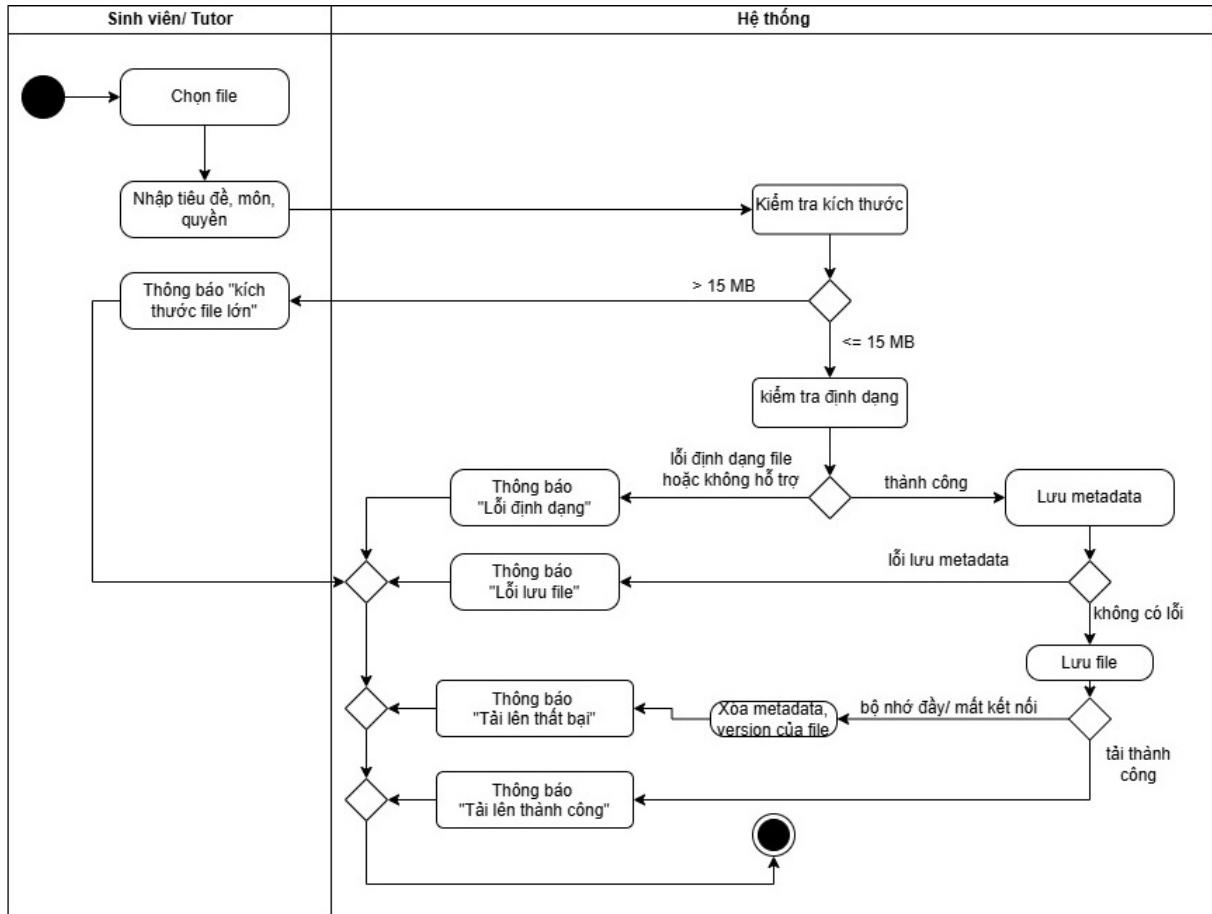


Hình 1.21: Sơ đồ hoạt động cho use-case "Gửi thông báo"

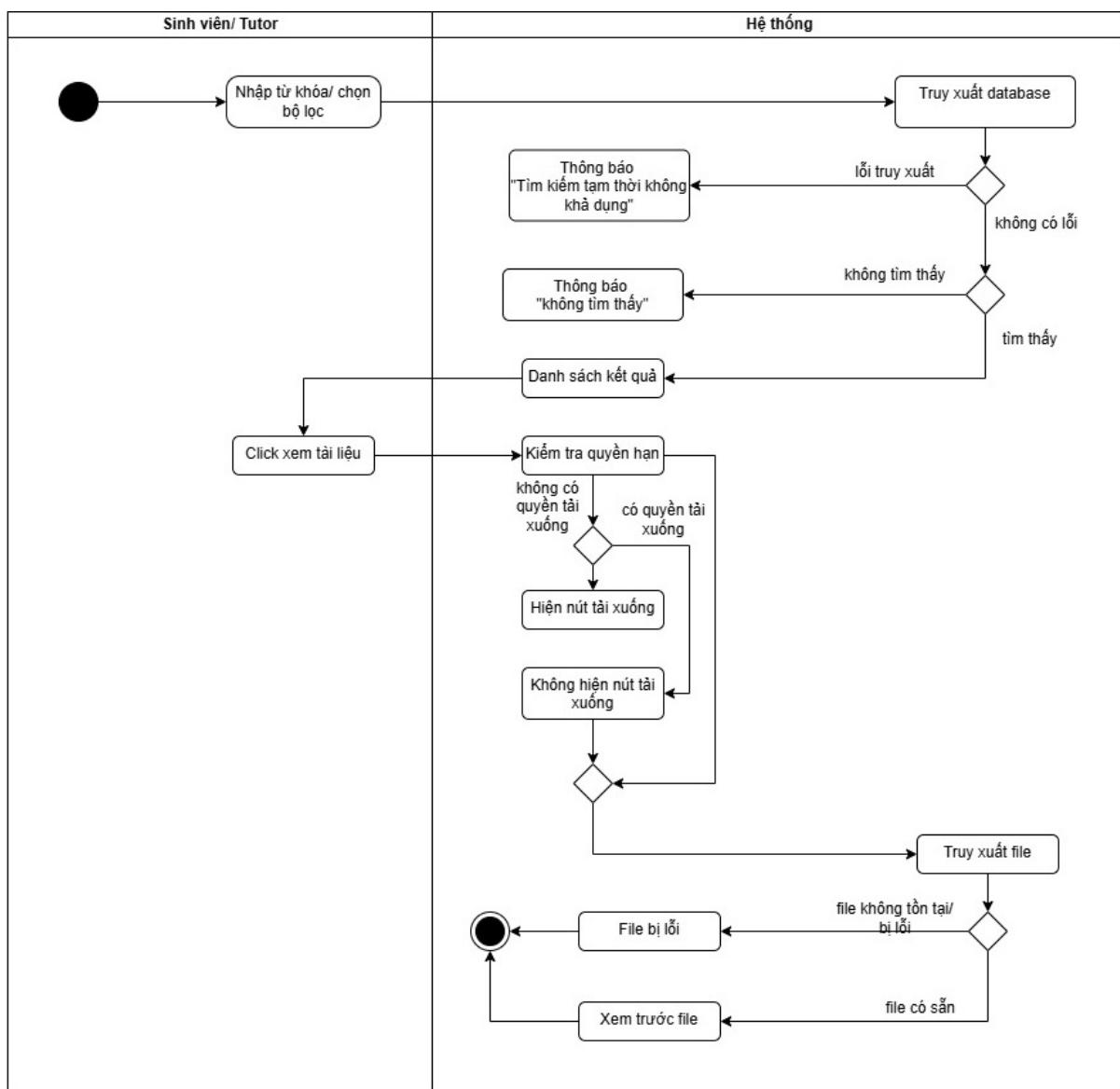
### Mô tả hoạt động Activity Diagram: Gửi thông báo

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.

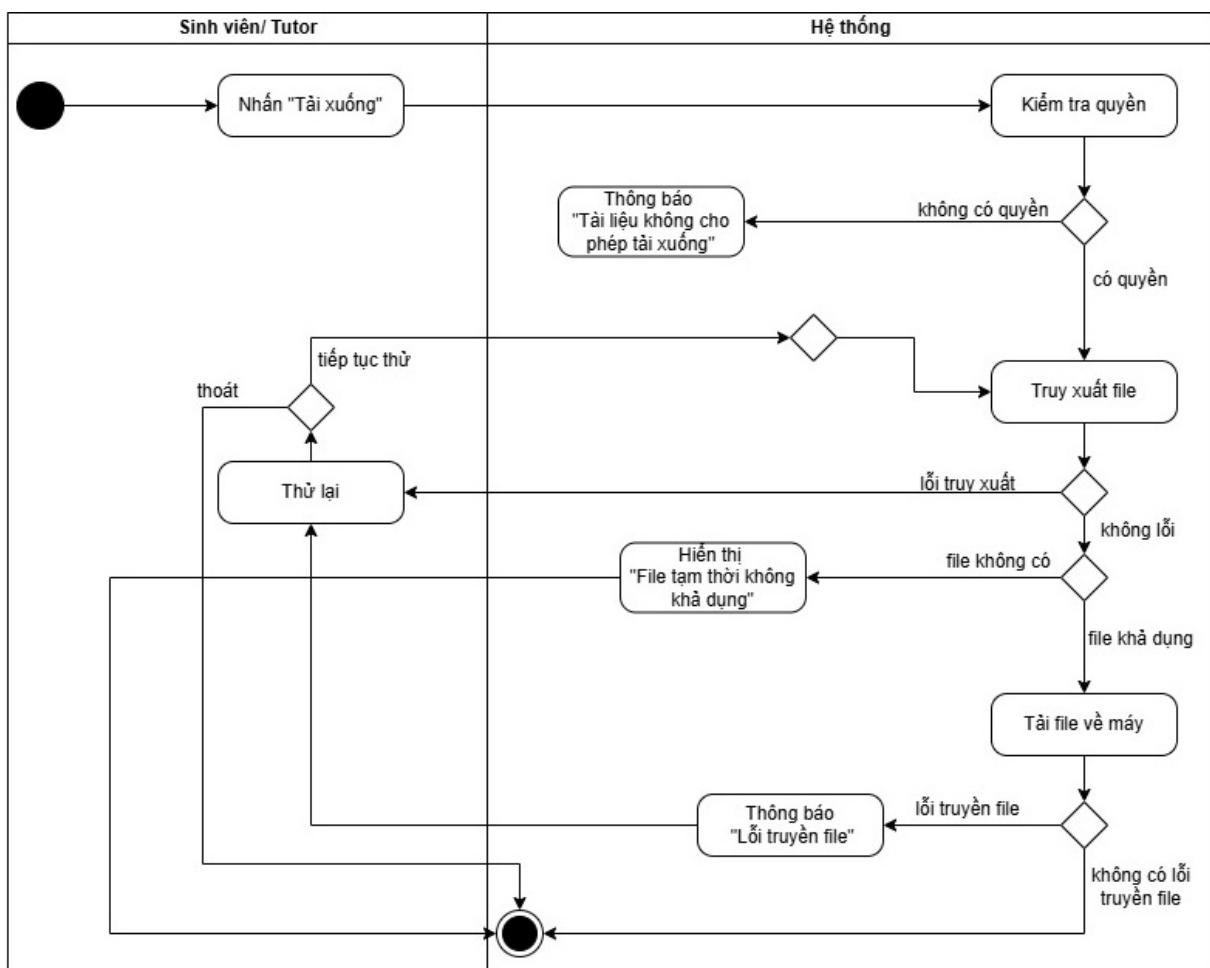
## 1.8 Quản lý tài liệu



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



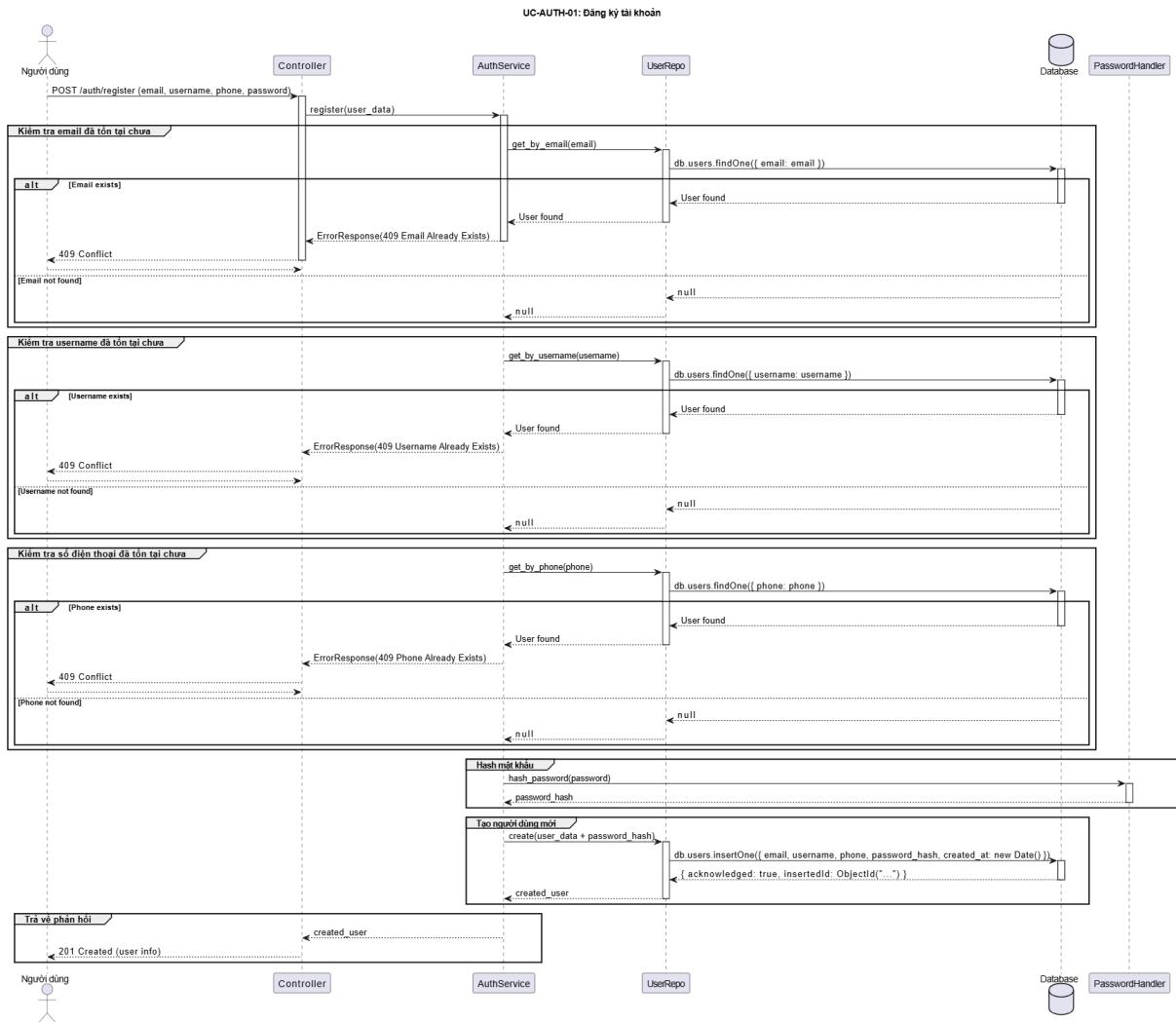
Hình 1.23: Sơ đồ hoạt động cho use-case "Tìm kiếm và xem trước"



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

## 2 Sequence Diagrams

### 2.1 Quản lý truy cập và xác thực

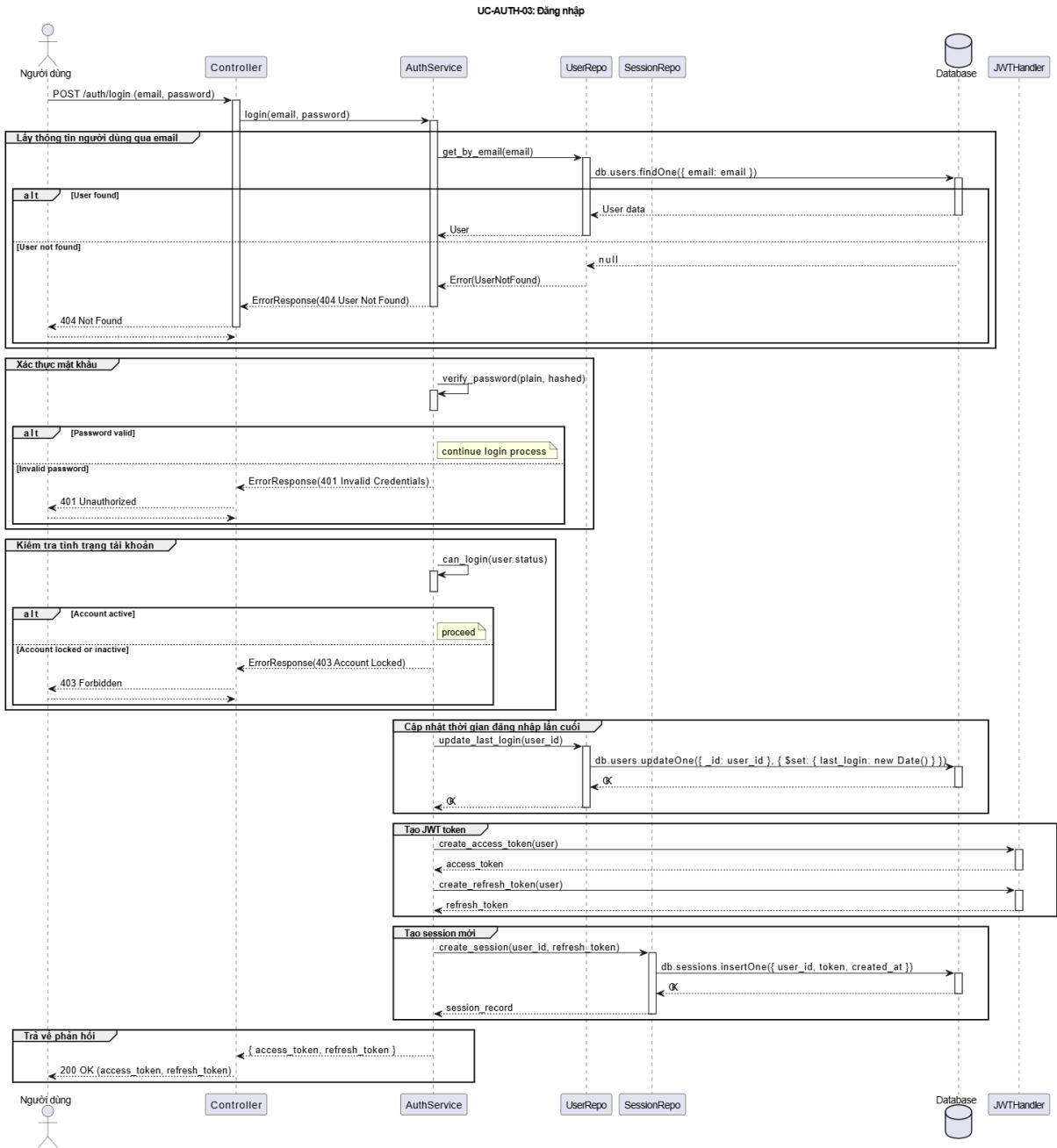


Hình 2.1: Sơ đồ tuần tự cho use-case "Quản lý tài khoản"

### Mô tả: 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.

- f. Sau khi lưu thành công, hệ thống phản hồi thông báo đăng ký thành công.
- g. Nếu dữ liệu không hợp lệ: trả về thông báo lỗi và yêu cầu nhập lại.



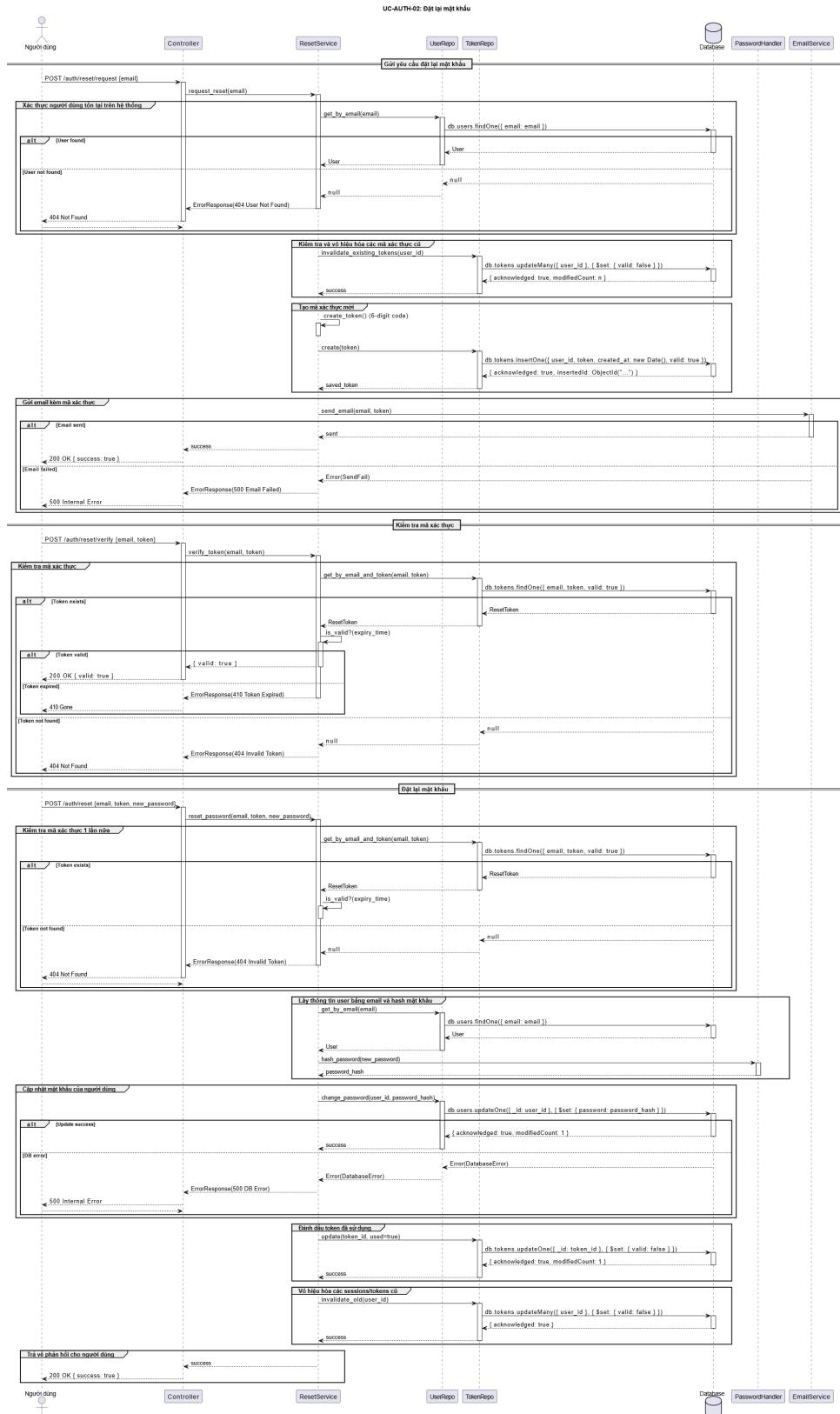
Hình 2.2: Sơ đồ tuần tự cho use-case "Đăng nhập"

### Mô tả: Đăng nhập

- a. Người dùng gửi yêu cầu hiển thị form đăng nhập qua GET /auth/login. Controller trả về giao diện đăng nhập.



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



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

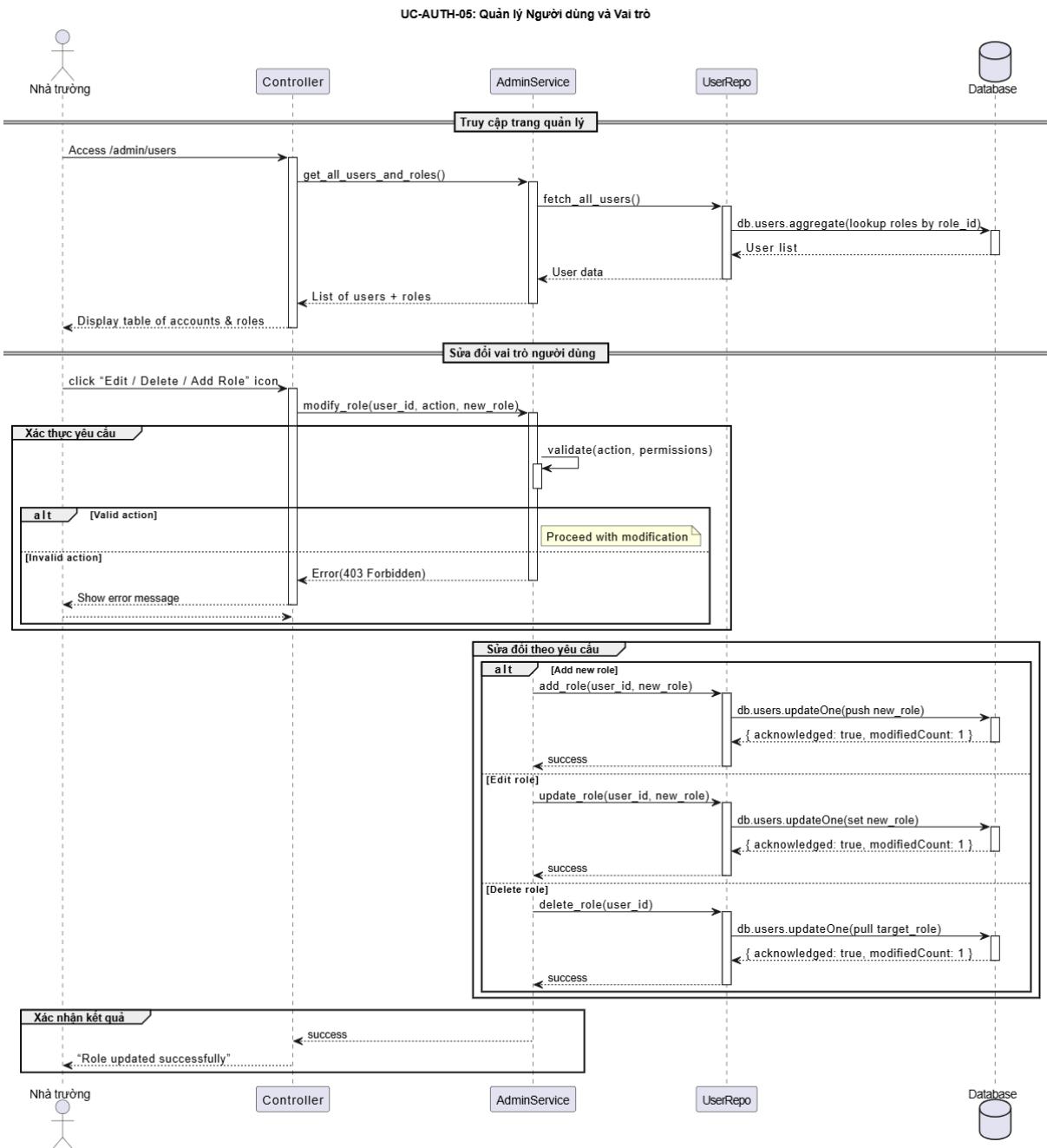


### Mô tả: Đặt lại mật khẩu

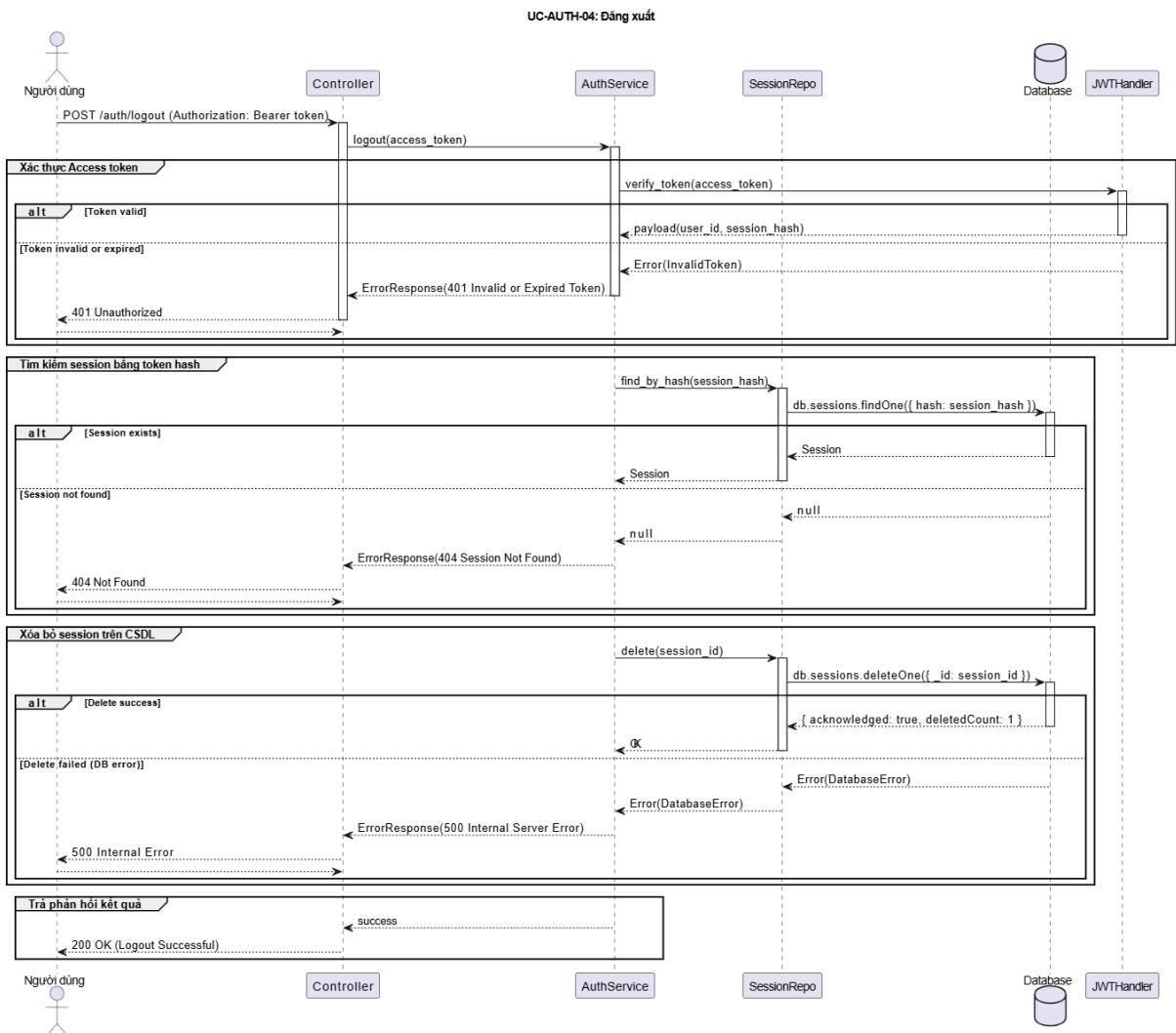
- a. Người dùng gửi yêu cầu lấy mã khôi phục qua POST /auth/reset/request. Controller gọi ResetService để kiểm tra email và tạo mã đặt lại mới.
- b. ResetService lưu mã vào TokenRepo và yêu cầu EmailService gửi mã xác thực đến người dùng.
- c. Người dùng gửi mã xác thực để kiểm tra qua POST /auth/reset/verify.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

### Mô tả: Quản lý người dùng và vai trò

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



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

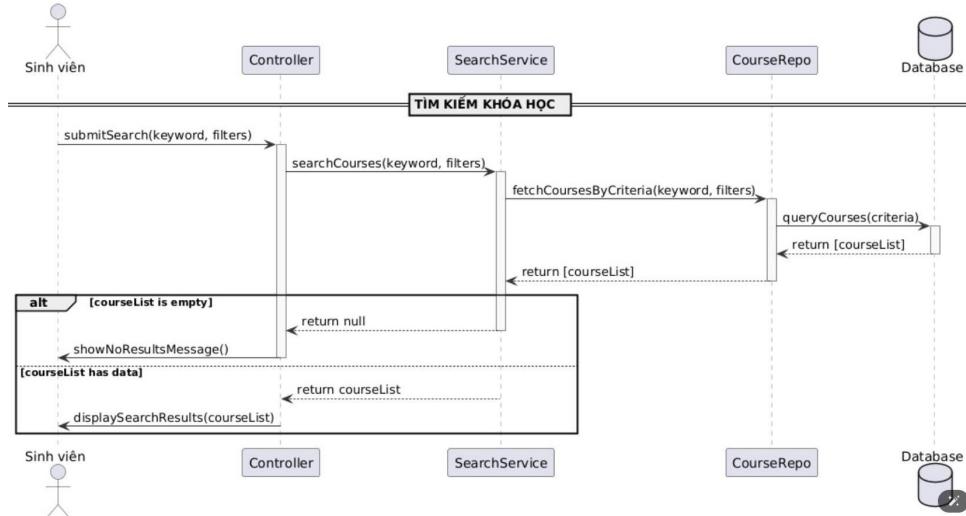


Hình 2.5: Sơ đồ tuần tự cho use-case "Đăng xuất"

### Mô tả: 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.

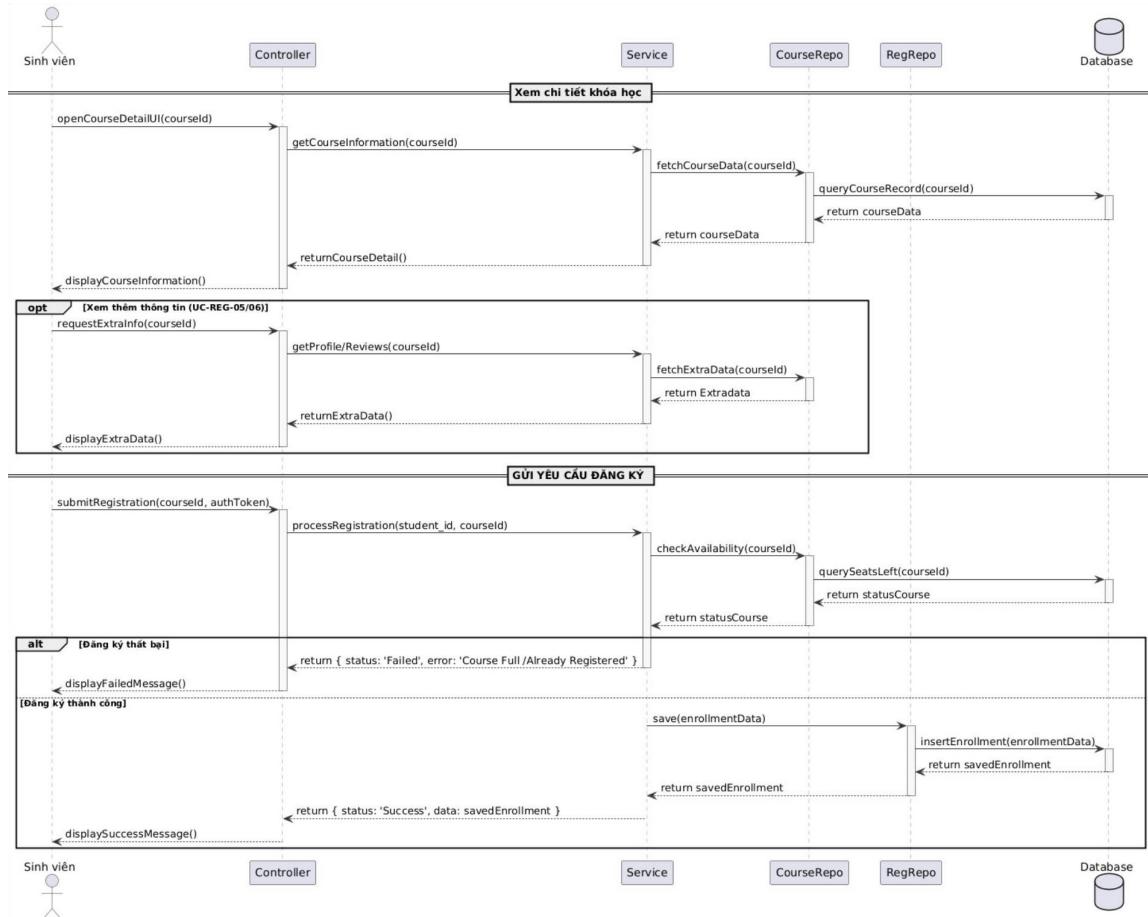
## 2.2 Đăng ký chương trình



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

### Đặc tả

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

### Đặc tả luồng sự kiện

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).

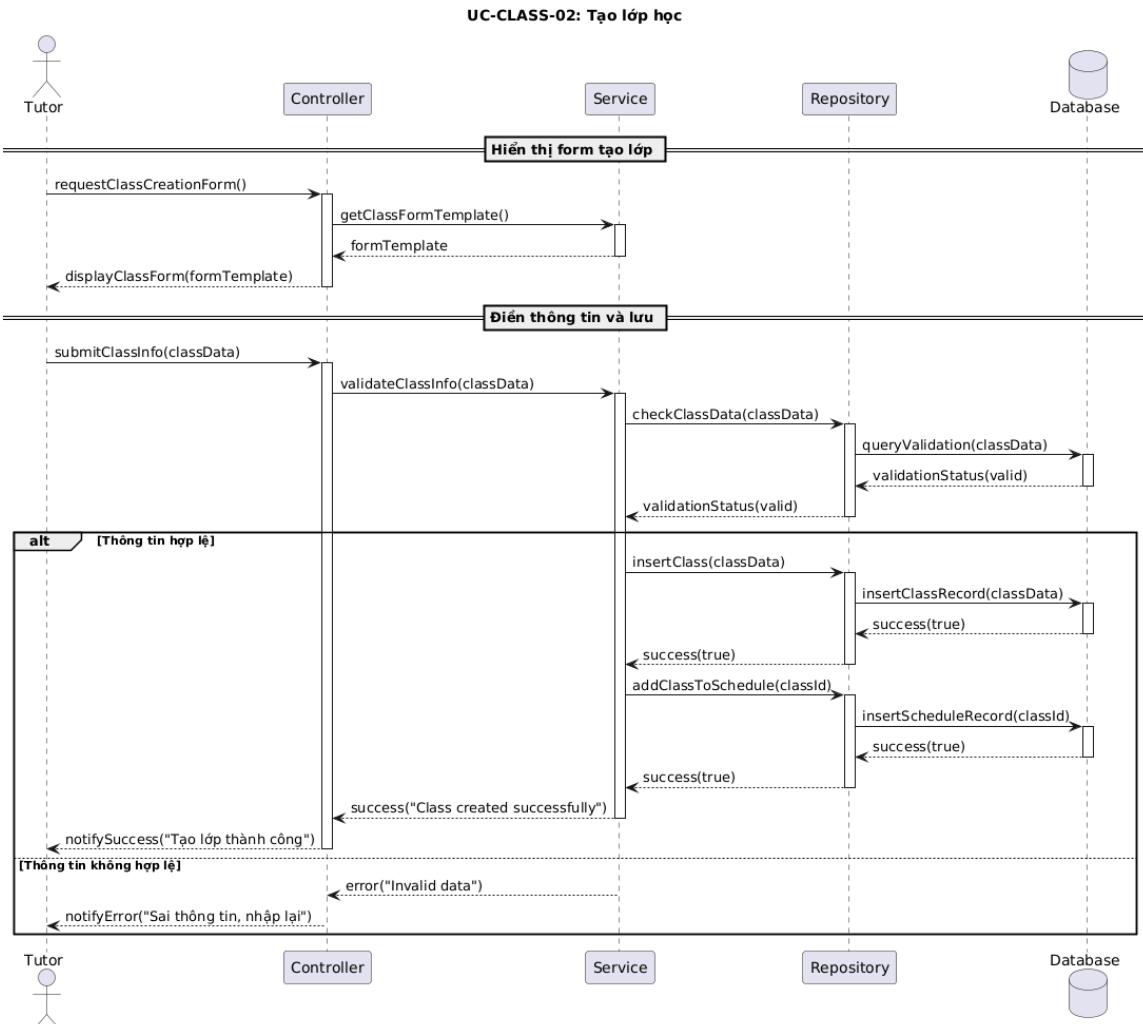
#### Luồng 2: Gửi yêu cầu đăng ký (RegisterProgramService)

- Khởi tạo:** Sinh viên gửi submitRegistration đến Controller.



- **Gọi Service:** Controller gọi processRegistration của Service (đại diện cho 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 lượt đăng ký mới vào Database.

### 2.3 Tạo chương trình học



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

Sequence diagram UC-CLASS-02 mô tả quá trình tutor yêu cầu tạo một lớp học mới, hệ thống hiển thị form nhập liệu, kiểm tra dữ liệu và lưu thông tin lớp vào cơ sở dữ liệu. Quy trình có sự tương tác giữa năm thành phần: *Tutor*, *Controller*, *Service*, *Repository* và *Database*. Chi tiết như sau:

#### 1. 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)`.



## 2. Điều thông tin và lưu

### 2.1 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)`.

### 2.2 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.

### 2.3 Xử lý theo điều kiện Trường hợp 1: Thông tin hợp lệ (`validationStatus = valid`)

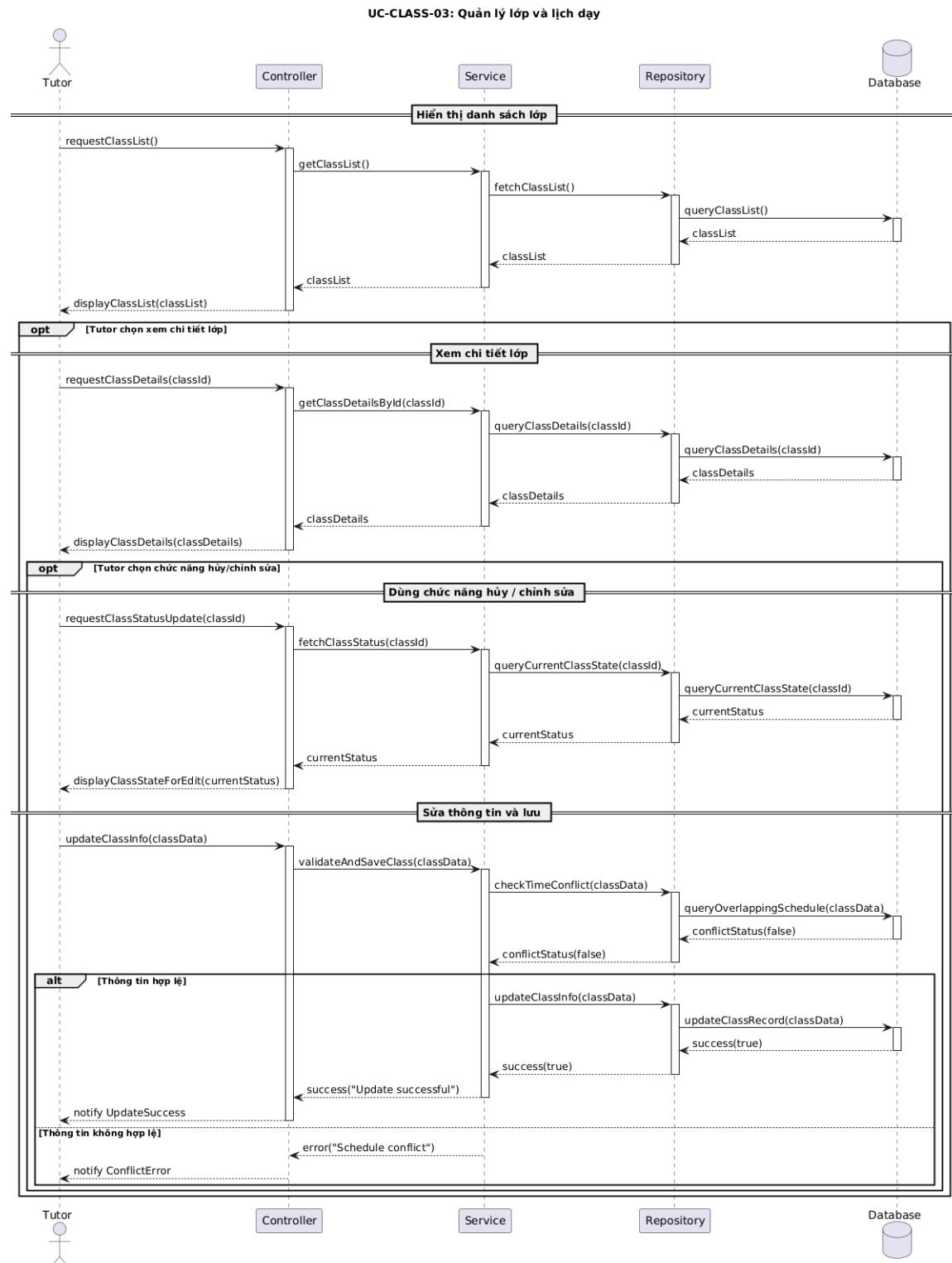
- Service gọi `insertClass(classData)` đến Repository để lưu lớp học.
- Repository thực hiện lưu dữ liệu lớp bằng `insertClassRecord(classData)`.
- Database trả về `success(true)` cho Repository.
- Repository tiếp tục thực hiện `insertScheduleRecord(classId)` để thêm lớp học vào lịch giảng dạy.
- Database 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")`.

## 3. 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 2.9: Sơ đồ tuần tự cho use-case "Quản lý lớp học"

Sequence diagram UC-CLASS-03 minh họa quy trình tutor quản lý lớp học và lịch dạy thông qua các

hành động: hiển thị danh sách lớp, xem chi tiết lớp, chỉnh sửa–huỷ lớp và cập nhật thông tin lớp.

1. 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)`.

2. 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)`.

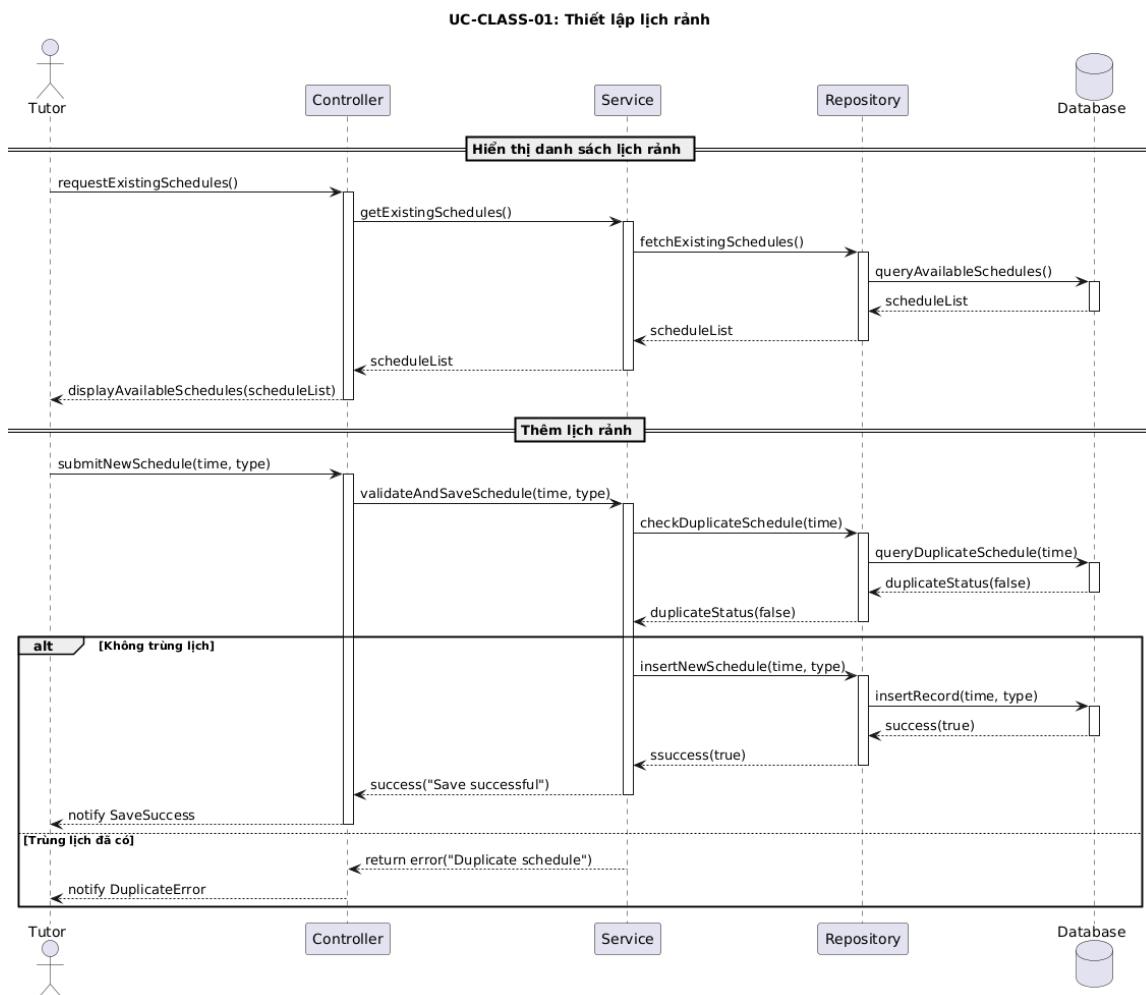
3. Dùng chức năng huỷ / chỉnh sửa (tùy chọn)

- 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)`.

4. 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 `error("Schedule conflict")`.
  - Controller hiển thị thông báo lỗi: `notify_ConflictError`.



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

Sequence diagram UC-CLASS-01 mô tả quá trình tutor xem danh sách lịch rảnh và thêm một lịch rảnh mới. Quy trình bao gồm sự tương tác giữa năm thành phần: *Tutor*, *Controller*, *Service*, *Repository* và *Database*. Chi tiết như sau:

#### 1. 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)`.

## 2. Thêm lịch rảnh mới

### 2.1 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)`.

### 2.2 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.

### 2.3 Xử lý theo điều kiện

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

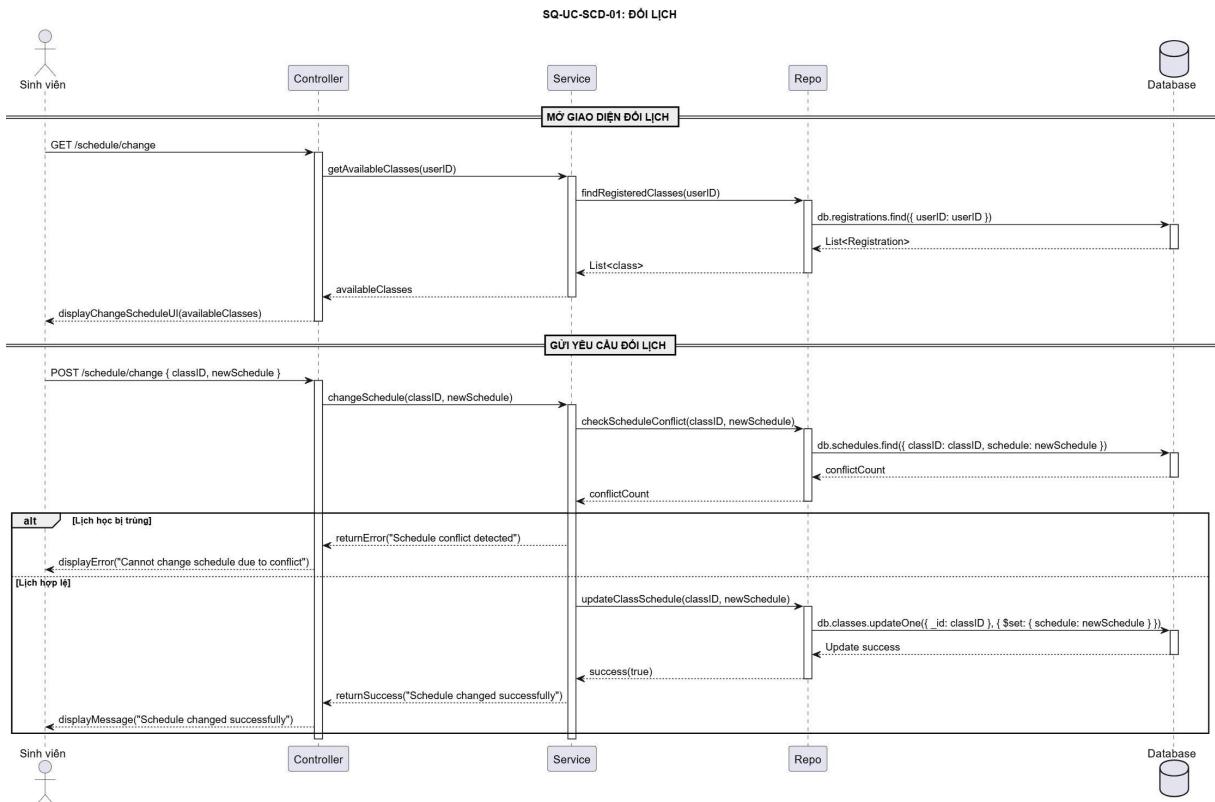
- Service gọi `insertNewSchedule(time, type)` đến Repository để lưu lịch.
- Repository thực hiện thêm bản ghi bằng `insertRecord(time, type)`.
- Database trả về kết quả thành công.
- Repository trả về `success(true)` cho Service.
- Service phản hồi Controller bằng `success("Save successful")`.
- Controller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

3. 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.

## 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.

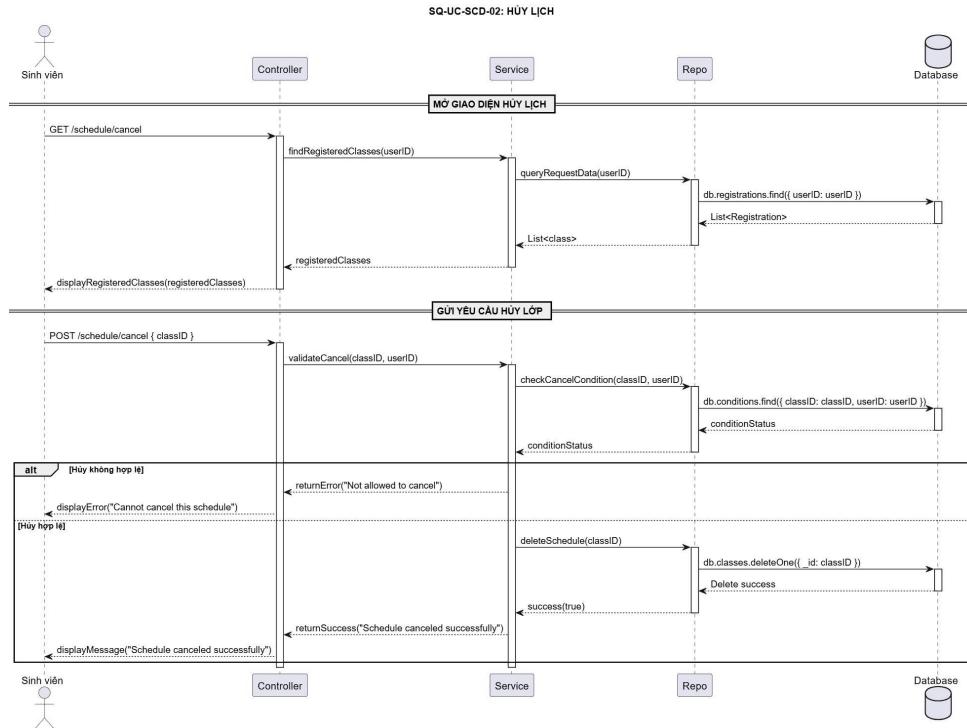
### Mở giao diện đổi lịch.

- 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.

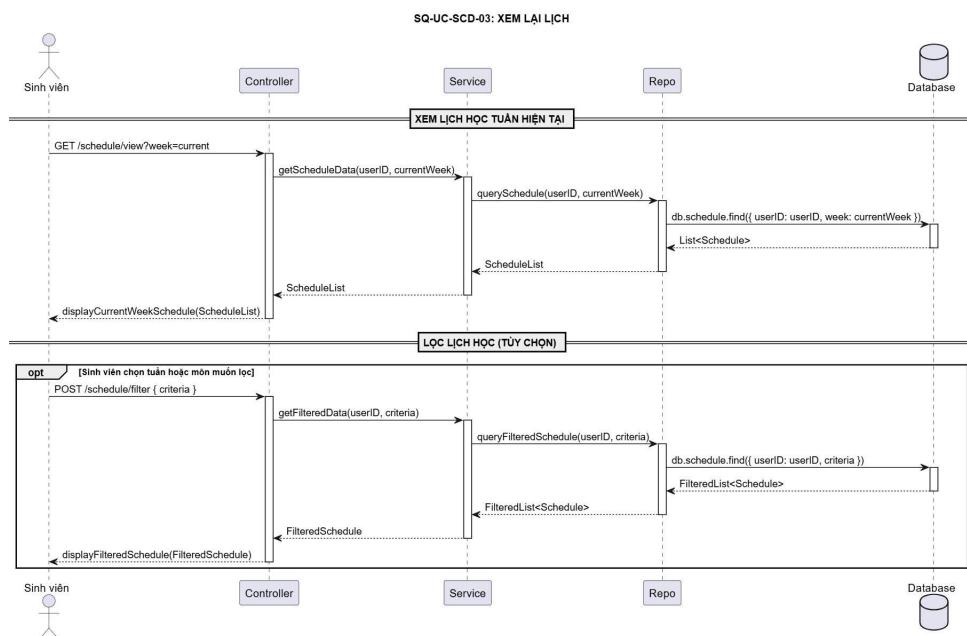


### Gửi yêu cầu đổi lịch.

- 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.

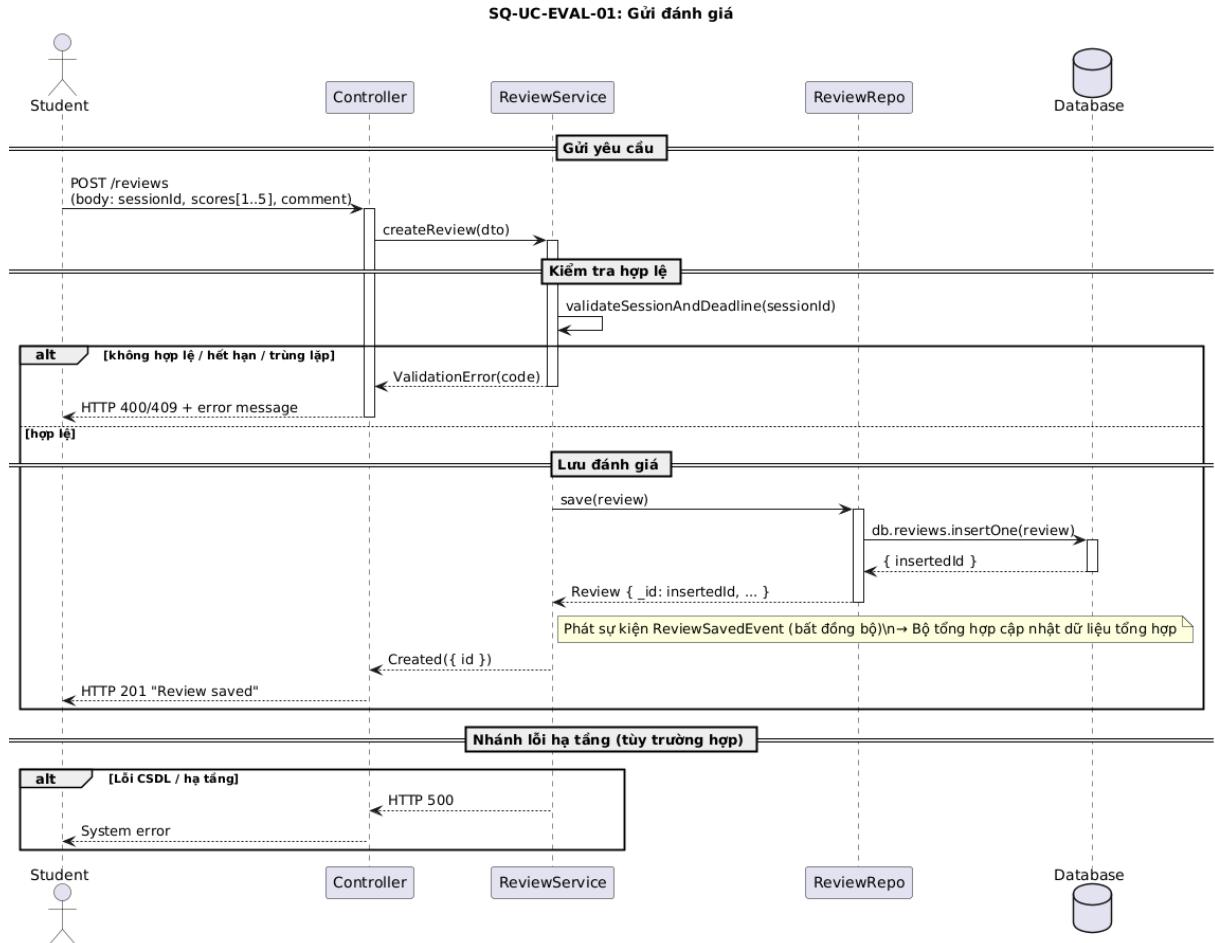


Hình 2.12: Sơ đồ tuần tự cho use-case "Hủy lịch học" cho sinh viên"



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

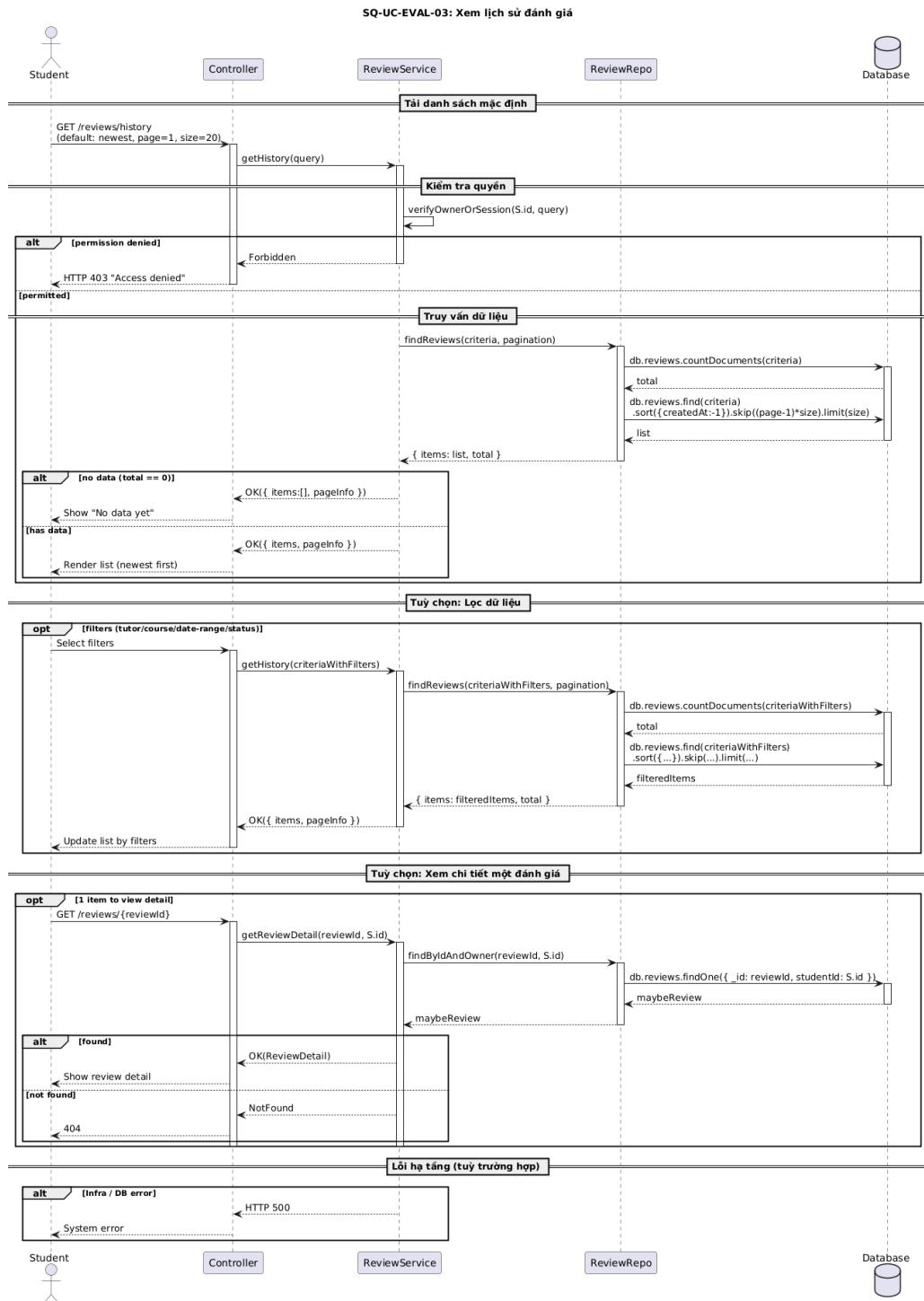
## 2.5 Đánh giá



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

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

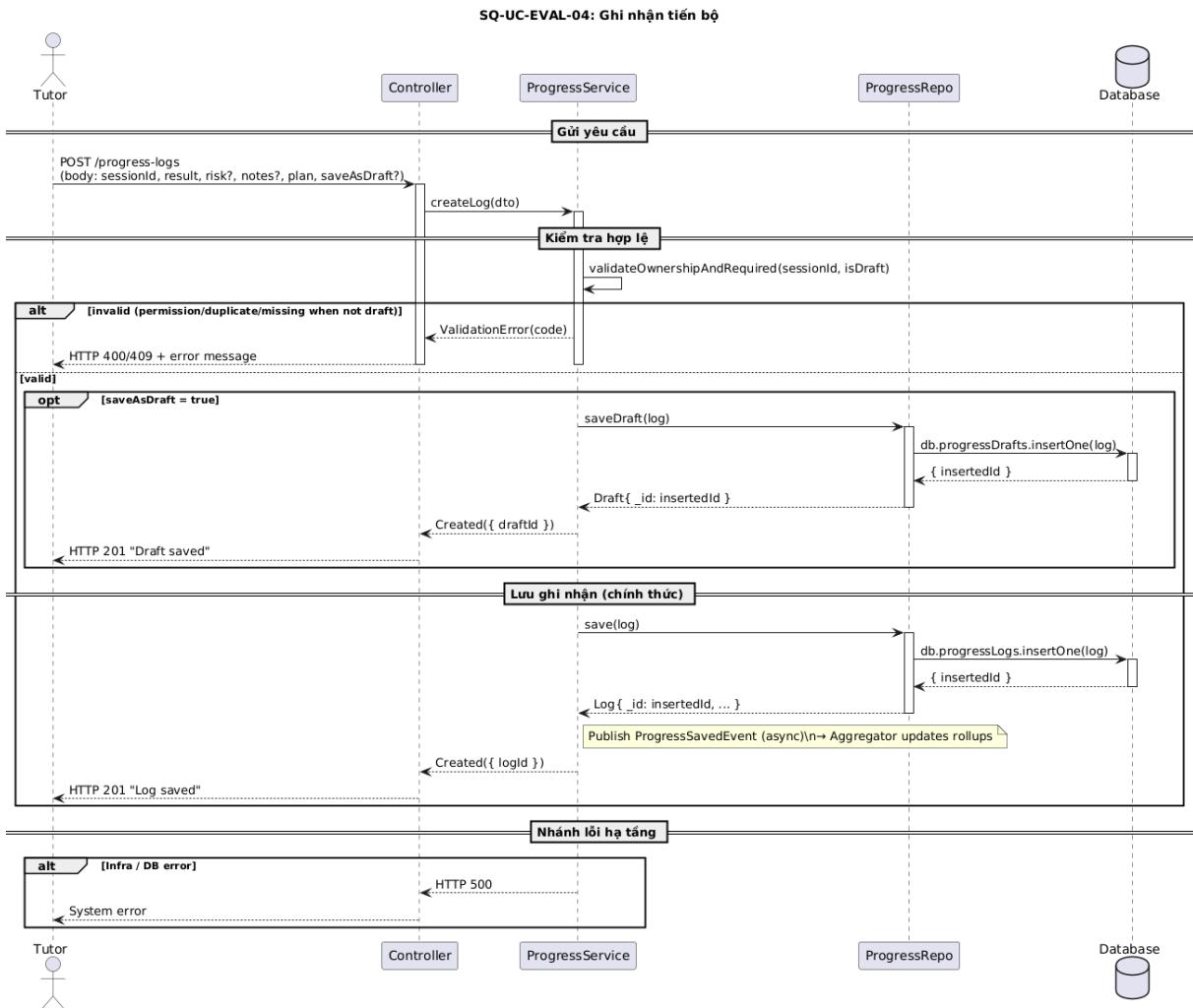
Controller gọi `ReviewService.createReview(dto)` → `validateSessionAndDeadline(sessionId)` → lưu vào `db.reviews` → phát `ReviewSavedEvent` cho bộ tổng hợp.



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"

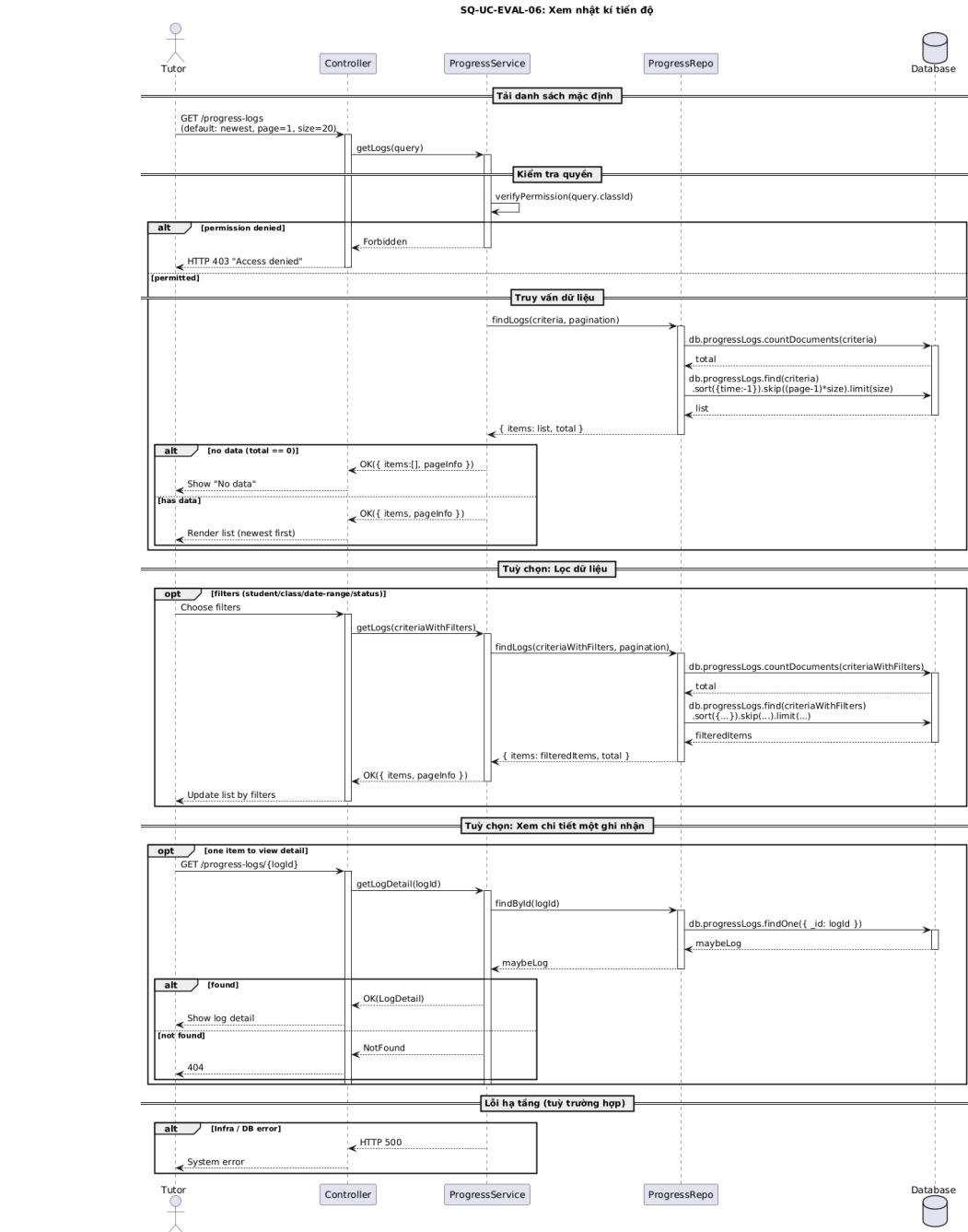
Controller nhận GET /reviews/history → ReviewService.verifyOwnerOrSession() → ReviewRepo.findReviews() truy vấn db.reviews với sắp xếp createdAt:-1 và phân trang.



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.

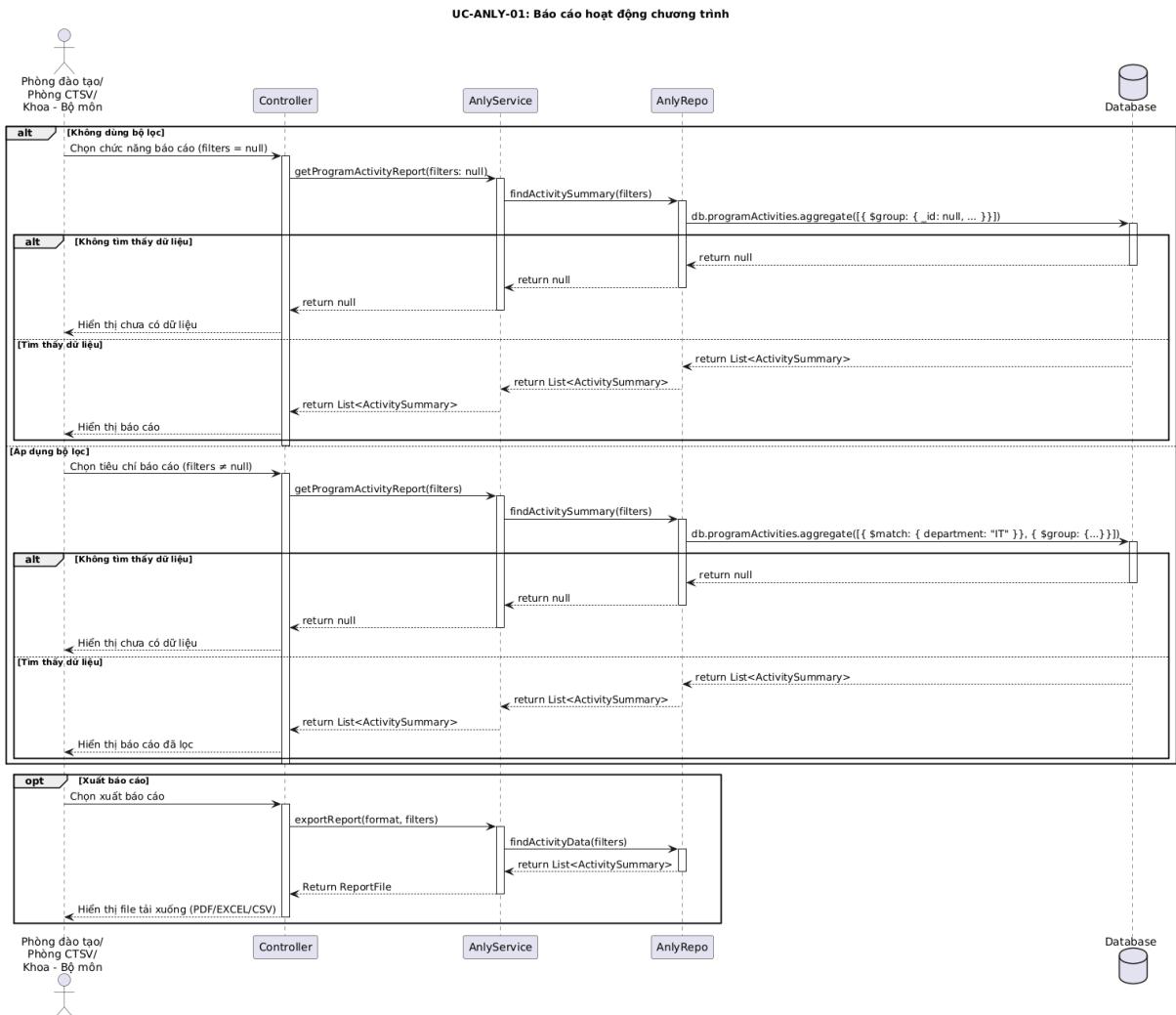


Hình 2.17: Sơ đồ tuần tự cho use-case "Xem nhật ký tiến độ"

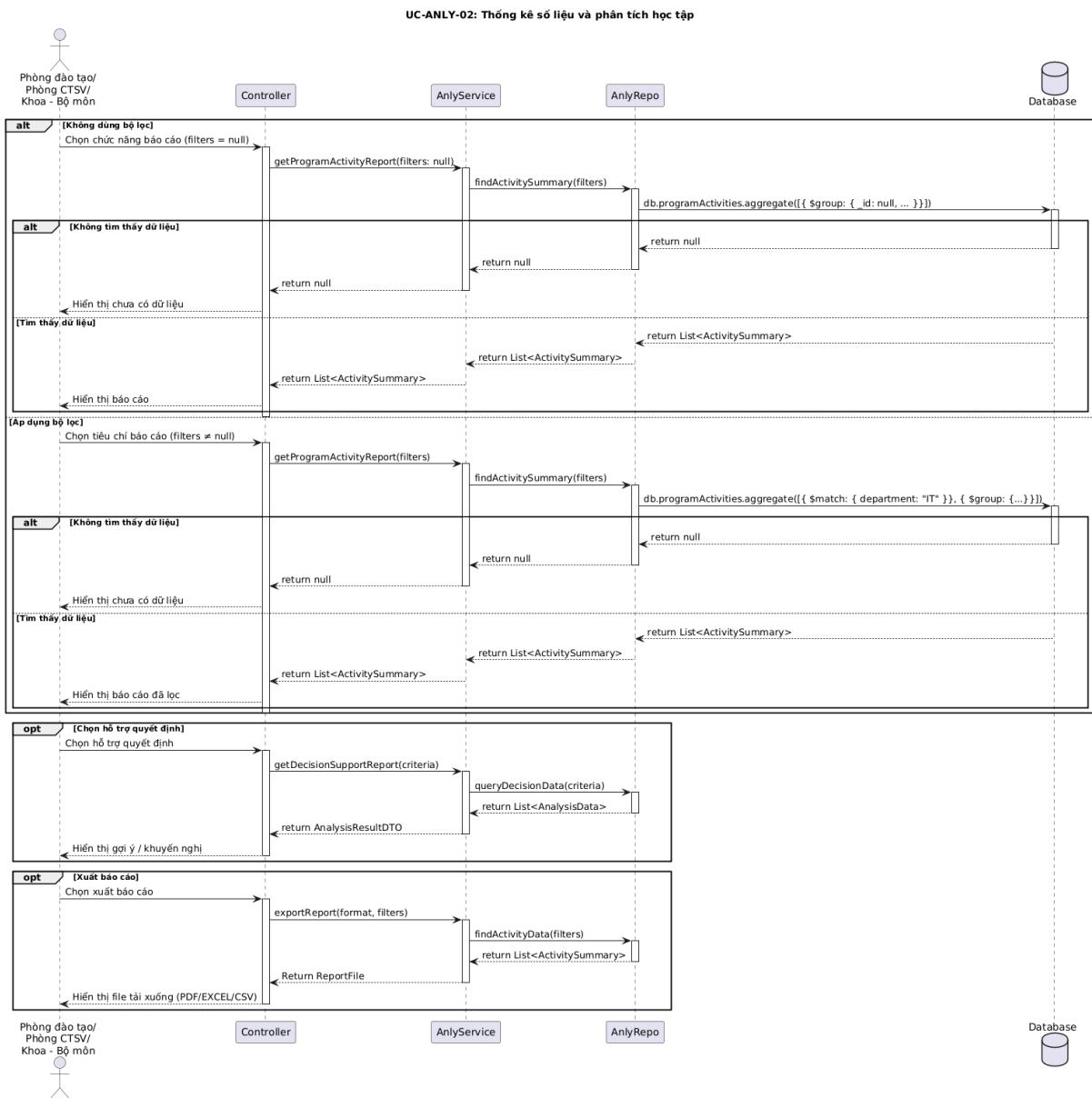
### Mô tả sơ đồ tuần tự "Xem nhật ký tiến độ"

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"

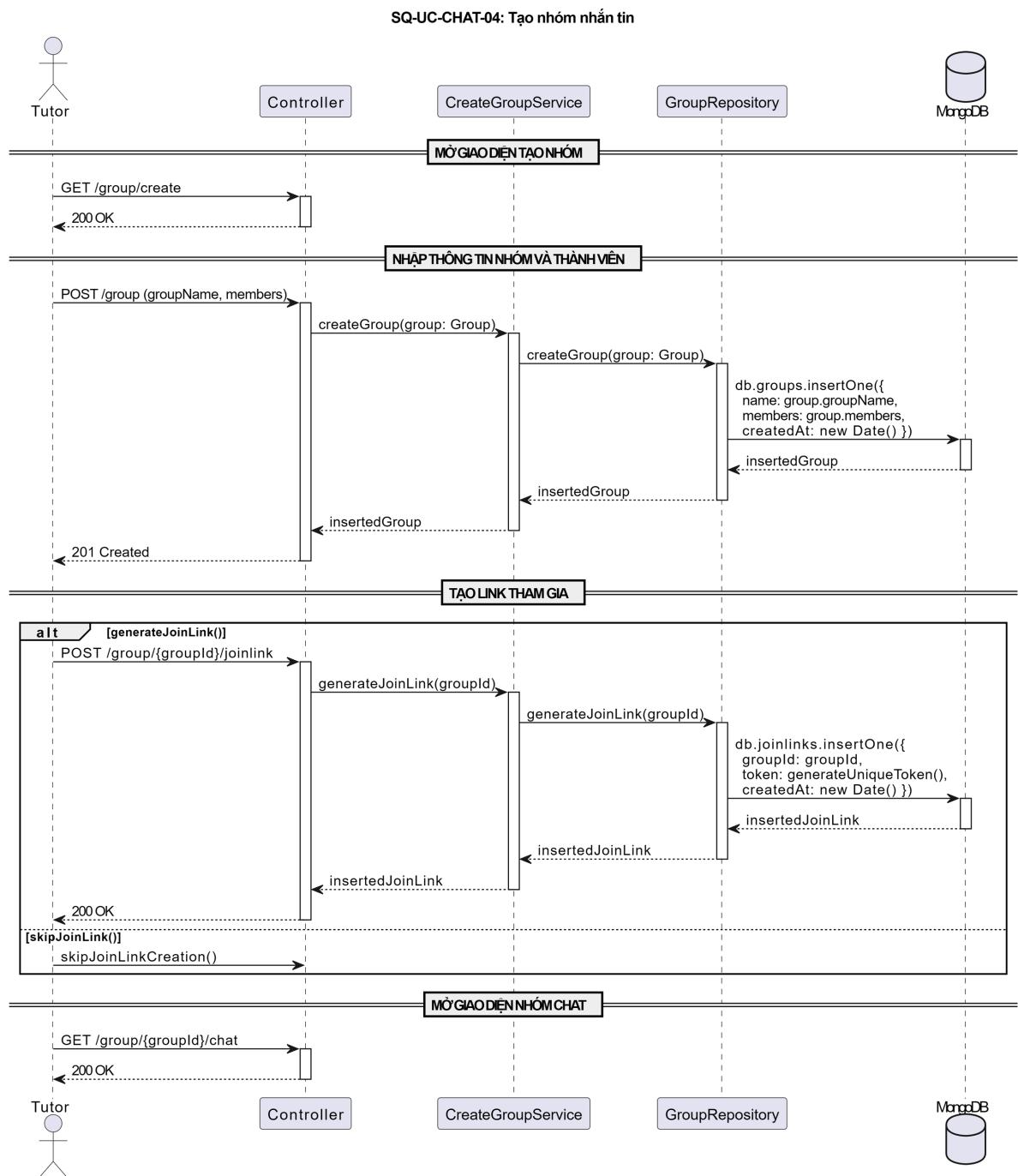
## 2.7 Thông báo và nhắn tin



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.

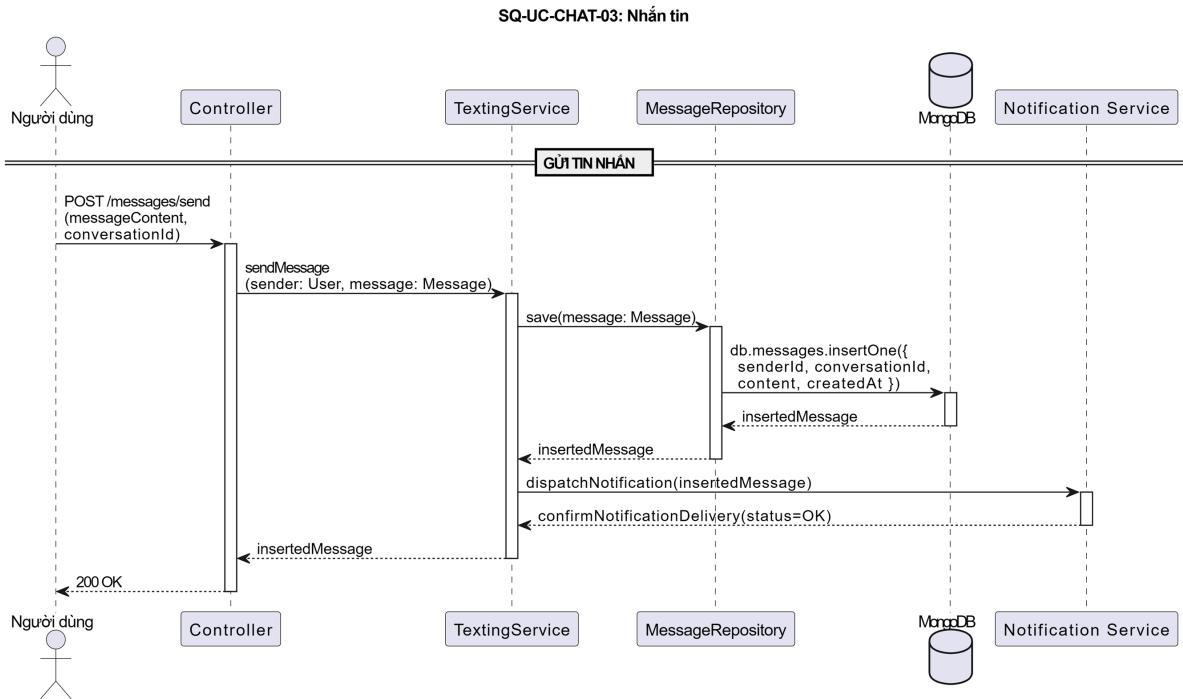


Hình 2.21: Sơ đồ tuần tự cho use-case "Tạo nhóm nhắn tin"



### Mô tả: Tạo nhóm nhắn tin

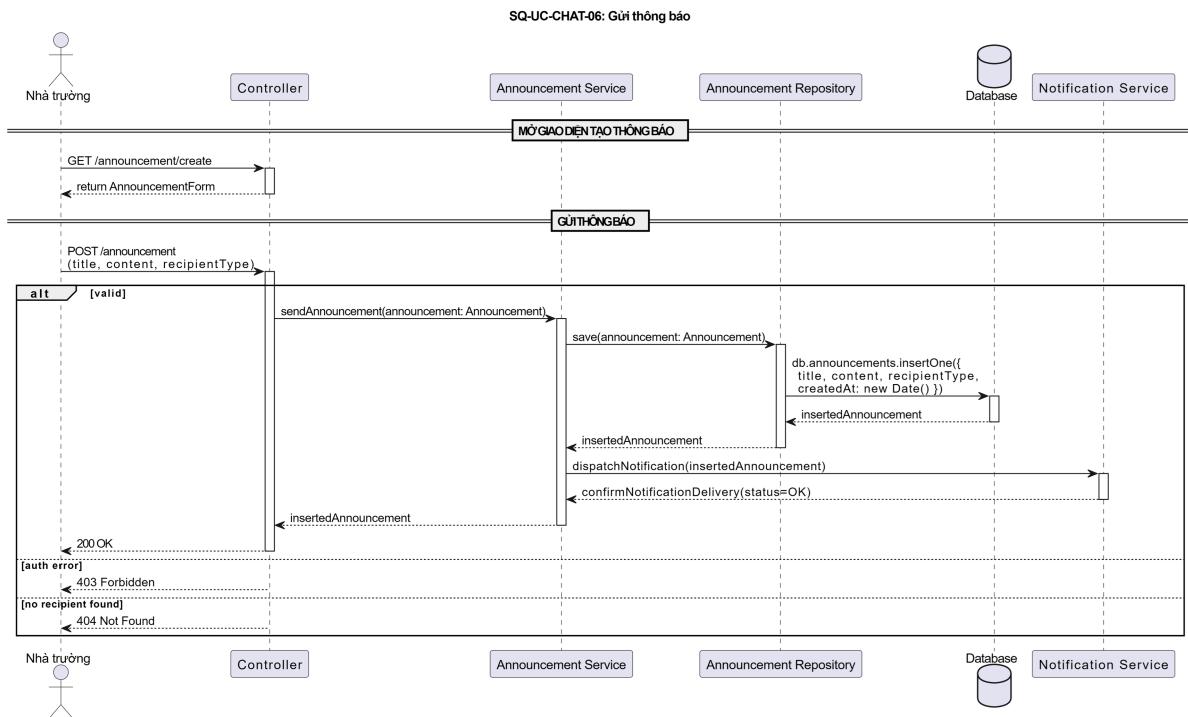
- a. Tutor gửi yêu cầu mở giao diện tạo nhóm (GET /group/create) và nhận phản hồi xác nhận.
- b. Tutor điền thông tin tên nhóm và danh sách thành viên rồi gửi lên (POST /group), Controller tiếp nhận chuyển cho CreateGroupService.
- c. Dịch vụ này gọi GroupRepository, lưu group mới vào DB (ghi nhận groupName, 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.

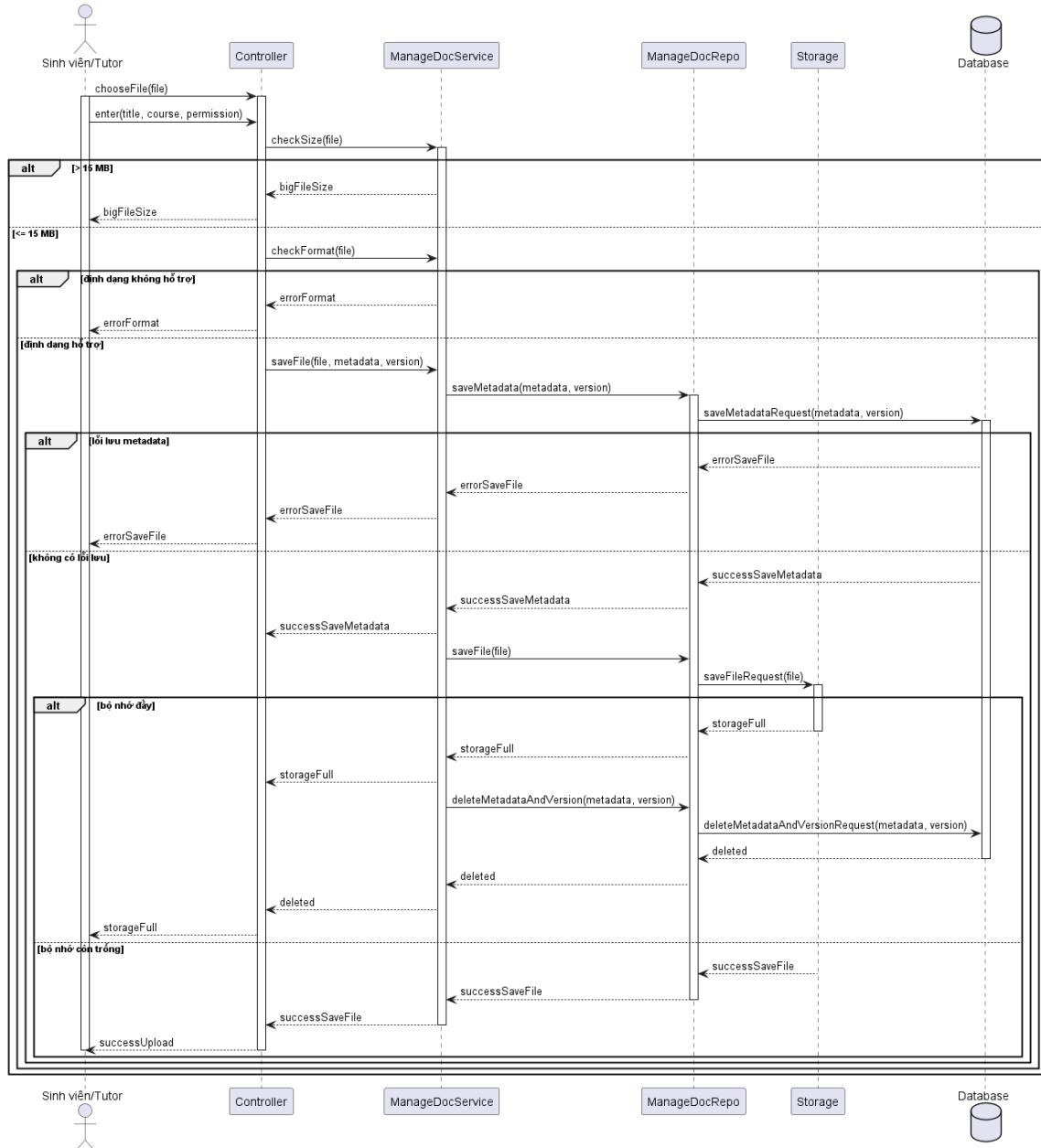


Hình 2.23: Sơ đồ tuần tự cho use-case "Gửi thông báo"

### Mô tả: Gửi thông báo

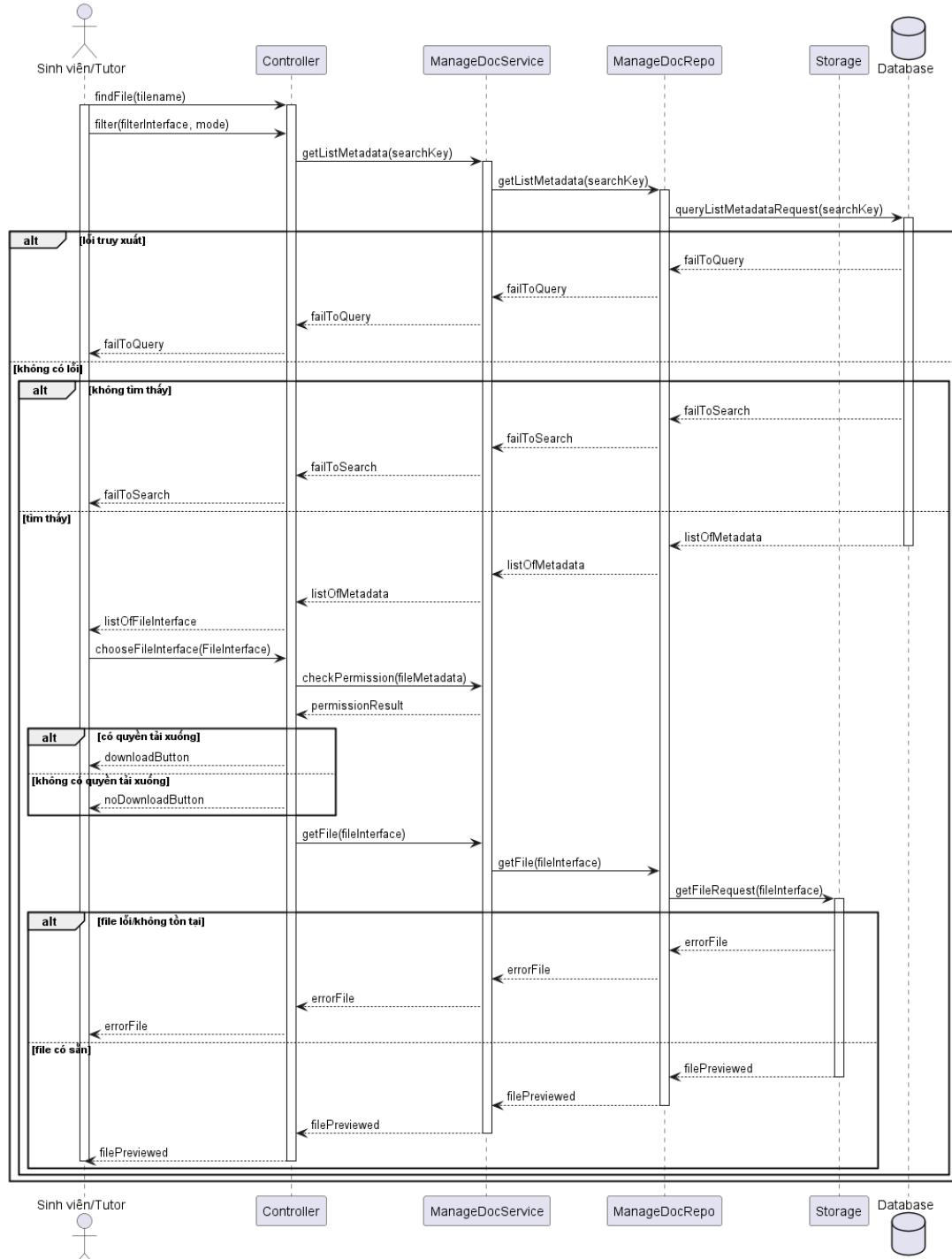
- Nhà trưởng mở giao diện tạo thông báo (`GET /announcement/create`), nhận lại form nhập liệu.
- Gửi thông báo mới (`POST /announcement`) với tiêu đề, nội dung, loại đối tượng nhận: Controller gửi tới AnnouncementService.
- AnnouncementService xử lý, lưu vào AnnouncementRepository, ghi nhận các thông 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

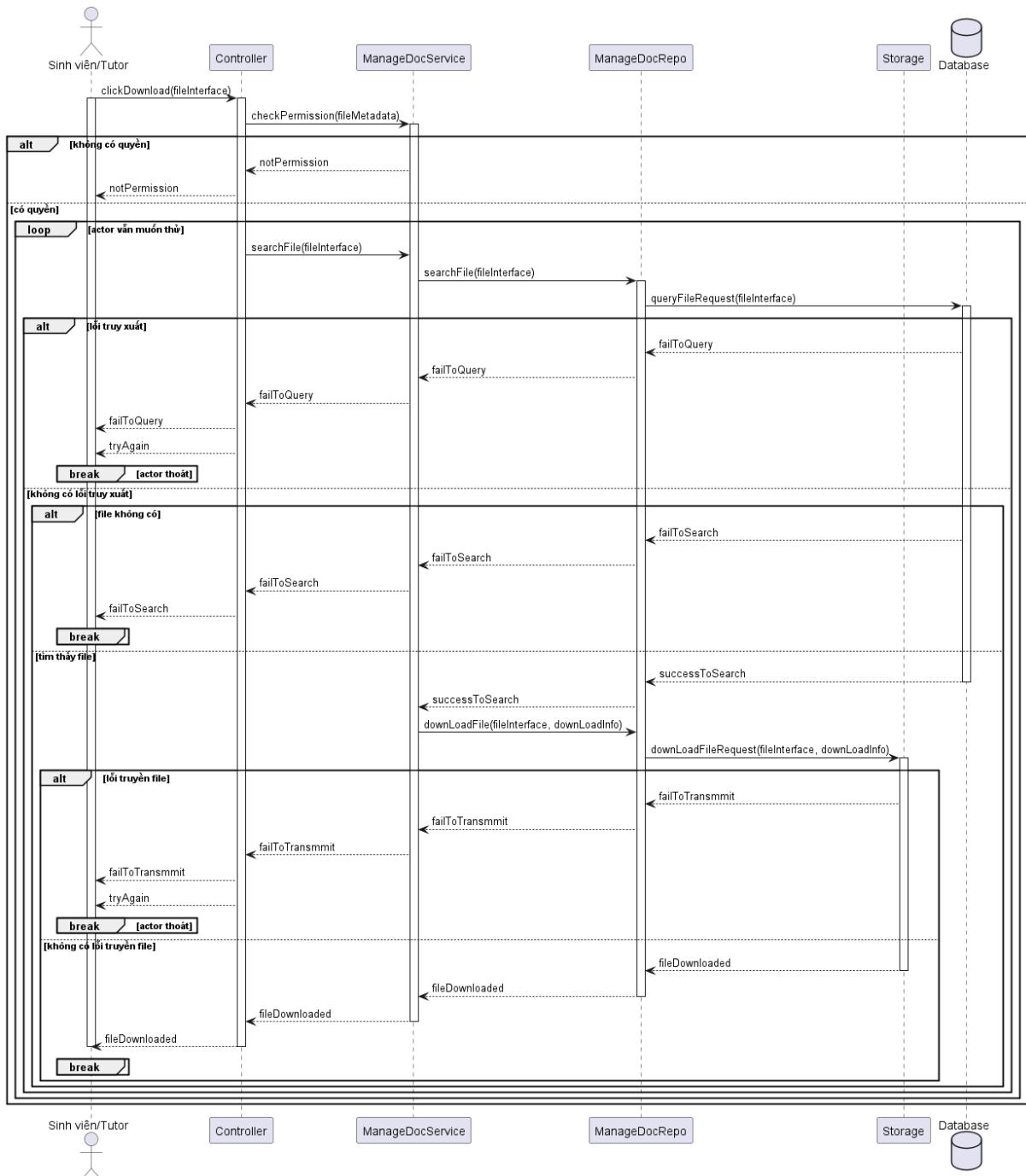


Hình 2.24: Sơ đồ tuần tự cho use-case "Tải tài liệu lên hệ thống"

### Mô tả: Tải tài liệu lên hệ thống



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



Hình 2.26: Sơ đồ tuần tự cho use-case "Tải tài liệu xuống hệ thống"

### 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

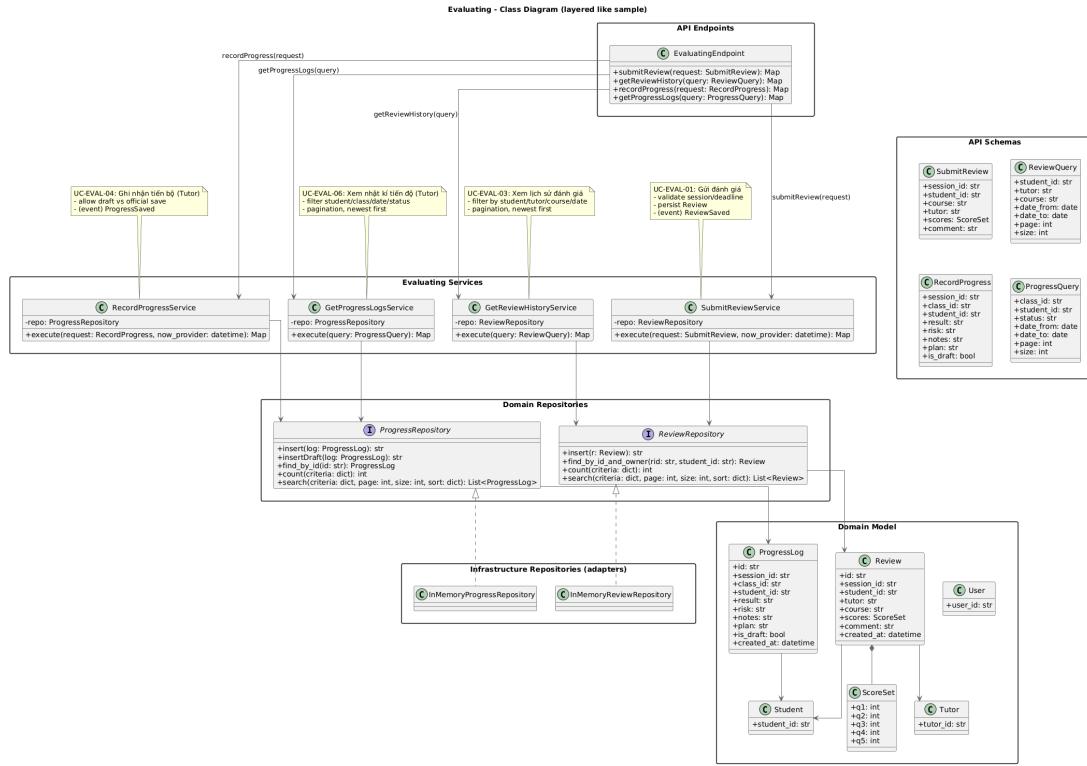


### 3.3 Tạo chương trình học



### 3.4 Thiết lập lịch trình cho sinh viên

### 3.5 Đánh giá



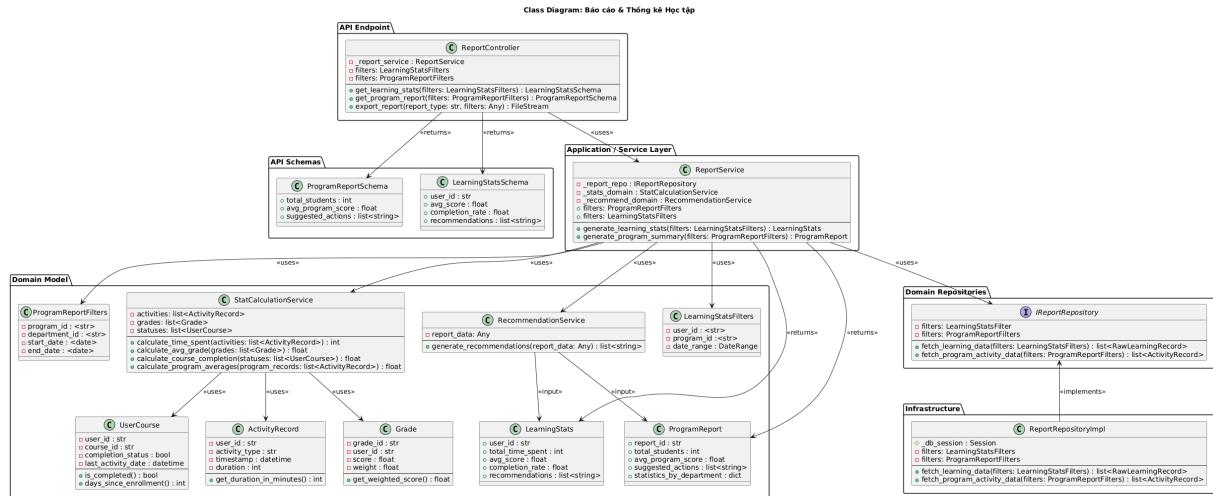
Hình 3.1: Sơ đồ lớp cho use-case "Đánh giá"

#### 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:** EvaluatorEndpoint 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



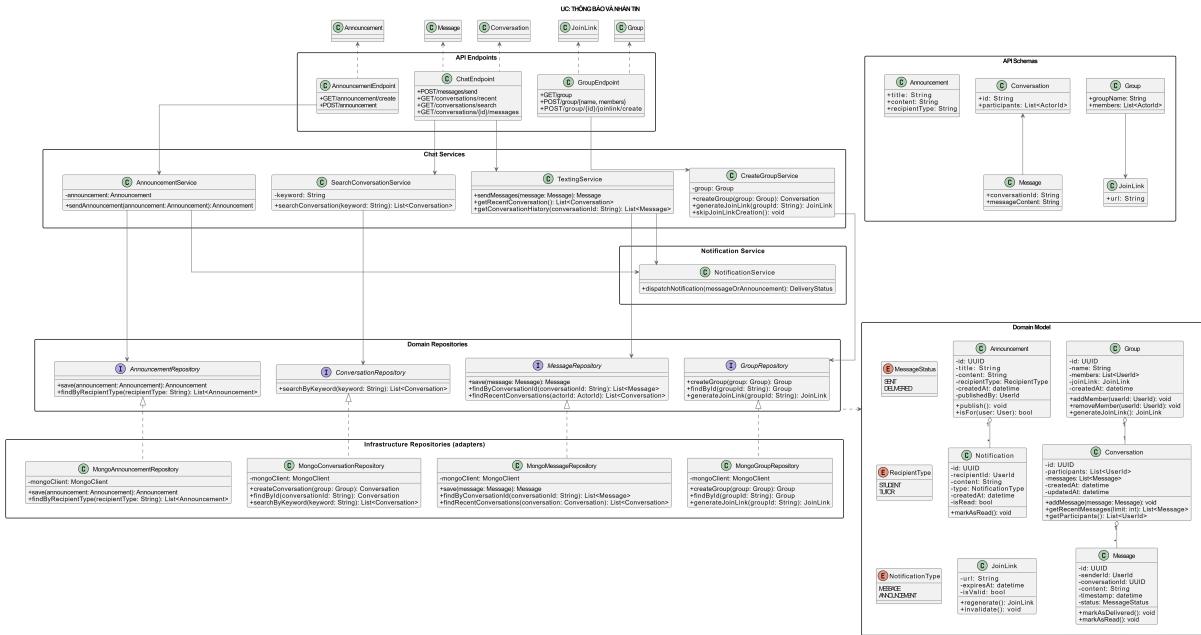
Hình 3.2: Sơ đồ lớp cho phân tích và báo cáo

#### 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



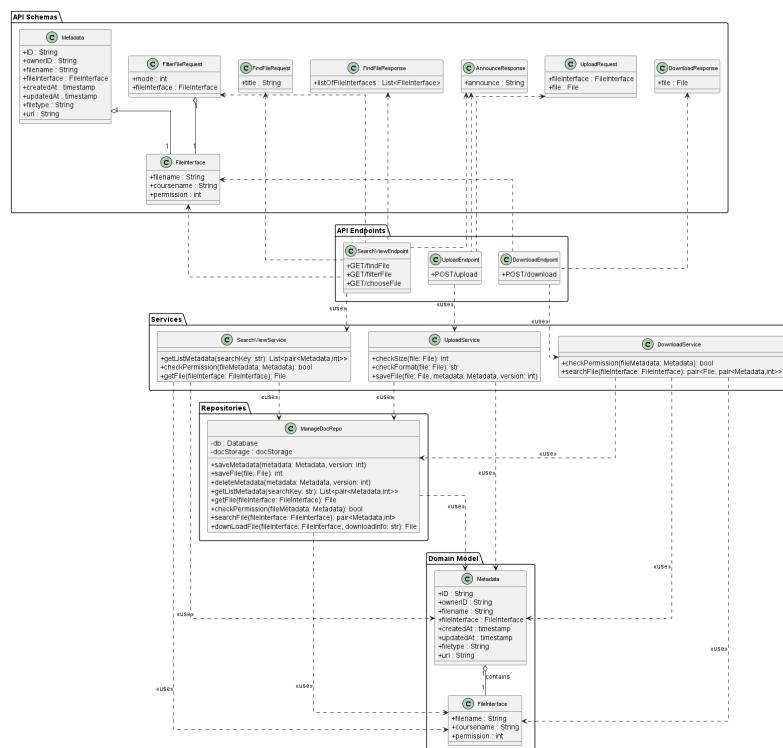
Hình 3.3: Sơ đồ lớp cho use-case Thông báo và Nhắn tin

#### 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



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.



## 4 Development view

### 4.1 Quản lý truy cập và xác thực



#### 4.2 Đăng ký chương trình

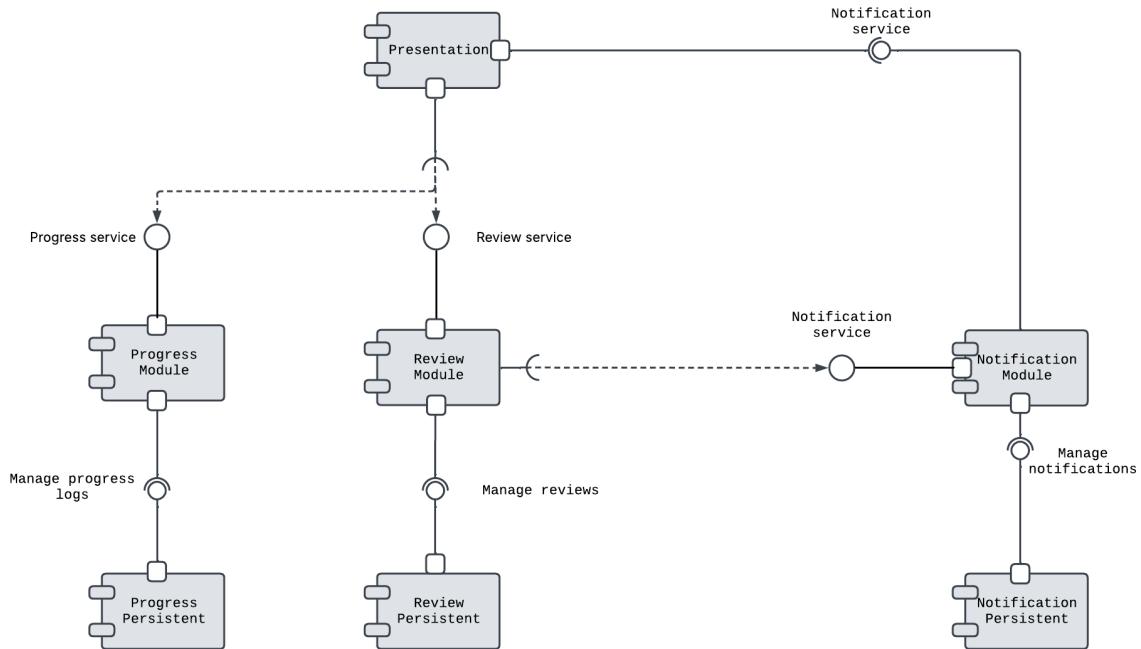


#### 4.3 Tạo chương trình học



#### 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"

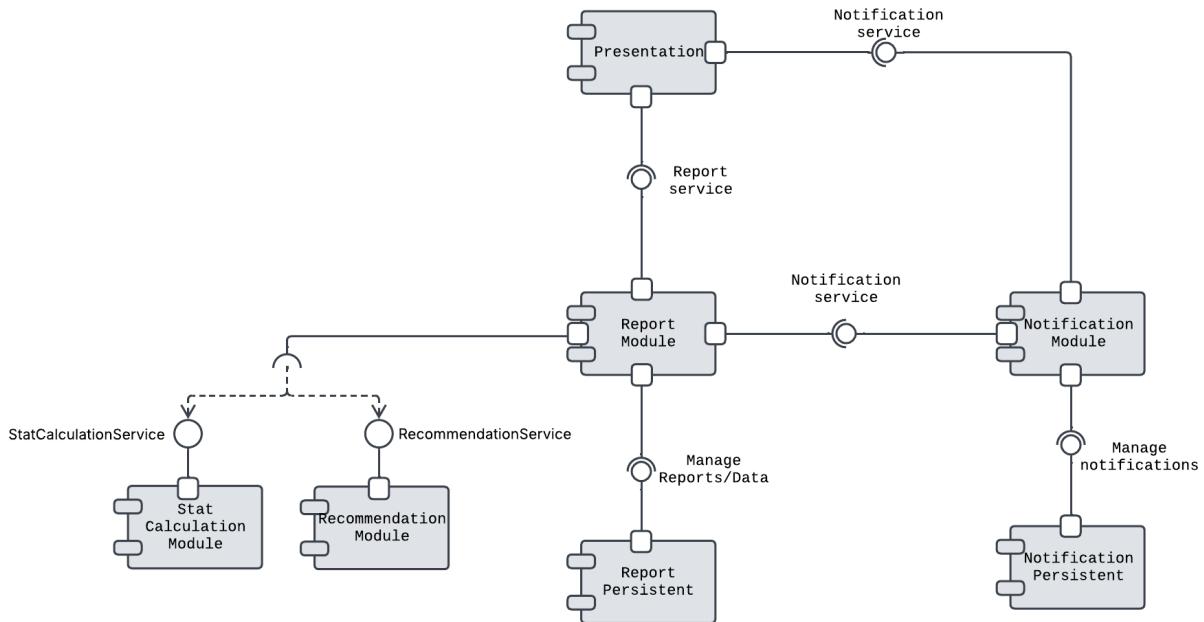
### Mô tả sơ đồ thành phần "Đánh giá"

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



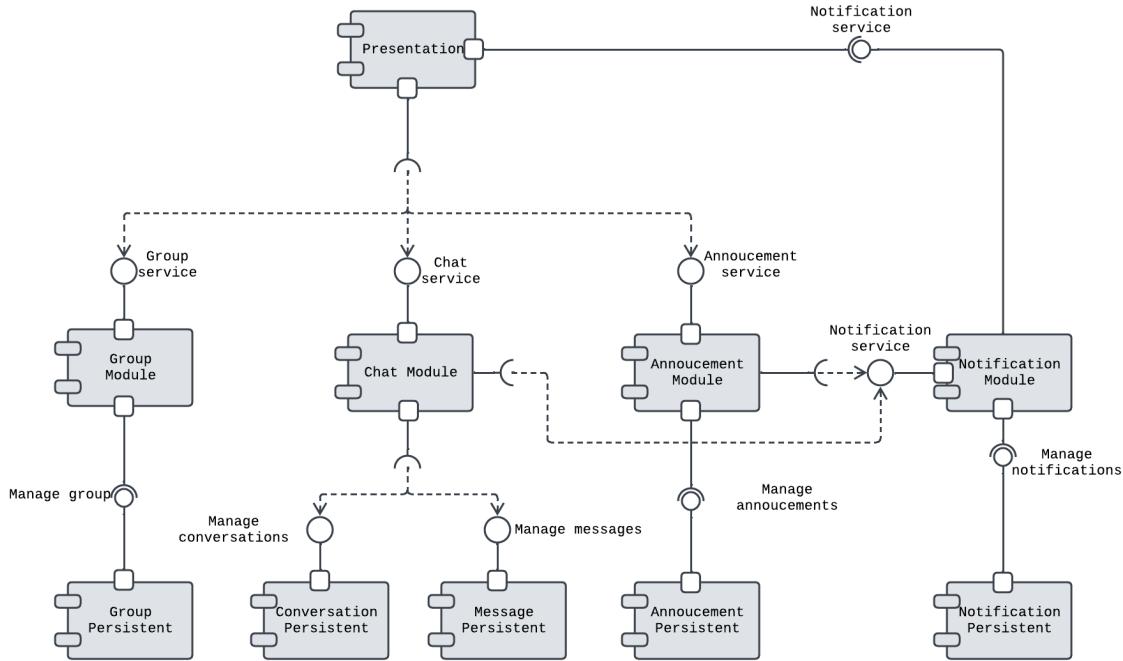
Hình 4.2: Sơ đồ thành phần cho use-case "Phân tích và báo cáo"

#### 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 **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.



#### 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,<br>User story<br>Use-case: Thông báo và tin nhắn | 100%              |
| 2   | Trần Đăng Khoa        | 2311645 | Use-case:<br>Quản lý truy cập và xác thực<br>Giao diện người dùng                   | 100%              |
| 3   | Nguyễn Lê Khôi Nguyên | 2312366 | Use-case:<br>Đăng ký chương trình   | 100%              |
| 4   | Võ Đức Cao Thượng     | 2313405 | Use-case:<br>Phân tích và báo cáo   | 100%              |
| 5   | Lương Trí Thịnh       | 2314057 | Use-case:<br>Đánh giá   | 100%              |
| 6   | Nguyễn Hà Viết Thông  | 2313339 | Use-case:<br>Thiết lập lịch trình cho sinh viên                                     | 100%              |
| 7   | Nguyễn Đức Dũng       | 2210582 | Use-case:<br>Tạo chương trình học   | 100%              |
| 8   | Vũ Anh Tuấn           | 2313771 | Use-case:<br>Quản lý tài liệu   | 100%              |