

ĐẠI HỌC PHENIKAA  
TRƯỜNG CÔNG NGHỆ THÔNG TIN PHENIKAA

---



**HỌC PHẦN: PHÂN TÍCH VÀ THIẾT KẾ PHẦN MỀM**  
**ĐỀ TÀI: Phát triển website mạng xã hội**

Giảng viên hướng dẫn: Mai Thúy Nga

Nhóm 7: Bùi Thu Trang - 23010297

Trịnh Thị Lan Nhi - 23010280

Lớp tín chỉ: CSE703095-1-1-25(N02)

**Hà Nội, tháng 11 năm 2025**

## BẢNG PHÂN CÔNG CÔNG VIỆC

STT	Mã SV	Họ tên	Công việc	Hoàn thành	Điểm
1.	23010297	Bùi Thu Trang	<ul style="list-style-type: none"> <li>- Lên kế hoạch, phân công công việc, kiểm tra tiến độ, tổng hợp kết quả</li> <li>- Viết code triển khai API cho tất cả các tính năng, quản lý Session, Auth key, xây dựng logic tải lên cho Post media, và API</li> <li>- Thiết kế và triển khai CSDL.</li> <li>- Viết báo cáo chương 4, 5</li> </ul>	100%	10
2.	23010280	Trịnh Thị Lan Nhi	<ul style="list-style-type: none"> <li>- Frontend Developer (UI&amp;UX) : Xây dựng giao diện dựa trên thiết kế, tích hợp API so người 1 cung cấp.</li> <li>- Viết báo cáo chương 1, 2, 3</li> </ul>	90%	9

# MỤC LỤC

<b>BẢNG PHÂN CÔNG CÔNG VIỆC.....</b>	2
<b>MỤC LỤC.....</b>	3
<b>DANH MỤC HÌNH ẢNH.....</b>	6
<b>DANH MỤC BẢNG BIỂU.....</b>	8
<b>GIẢI THÍCH THUẬT NGỮ.....</b>	10
<b>LỜI GIỚI THIỆU.....</b>	12
<b>CHƯƠNG 1. TỔNG QUAN VỀ PHẦN MỀM.....</b>	13
1.1.Mô tả bài toán.....	13
1.2. Yêu cầu bài toán.....	14
1.2.1. Chuẩn bị dữ liệu.....	14
1.2.2. Thực hiện quy trình.....	15
1.3. Yêu cầu nghiệp vụ.....	15
1.3.1. Mô tả quy trình nghiệp vụ.....	15
1.3.2. Các yêu cầu nghiệp vụ.....	16
1.4. Phân tích Use Case các chức năng chính.....	18
1.4.1. Sơ đồ Use Case các chức năng chính .....	18
1.4.2. Các tác nhân hệ thống.....	19
1.4.3. Các nhóm Use Case.....	19
1.4.4. Bảng ánh xạ yêu cầu nghiệp vụ với các Use Case.....	21
<b>CHƯƠNG 2. KIẾN TRÚC TỔNG THỂ CỦA HỆ THỐNG.....</b>	22
2.1. Sơ đồ kiến trúc tổng thể.....	22
2.1.1. Sơ đồ.....	22
2.1.2. Mô tả sơ đồ .....	22
2.1.3. Giới thiệu các công nghệ sử dụng.....	23
2.2. Hệ quản trị cơ sở dữ liệu.....	24
2.2.1. Giới thiệu .....	24
2.2.2. Thành phần chính kiến trúc (của Postgres).....	25
2.2.3. Ưu và nhược điểm.....	25
2.3. Server.....	26
2.3.1. Giới thiệu về Next.js 15 .....	26
2.3.2. Lý do sử dụng.....	26
2.4. Web Client.....	27

2.4.1. Giới thiệu về Web Client.....	27
2.4.2. Các thành phần cơ bản trong Web Client.....	27
2.4.3. Lý do sử dụng.....	28
2.5. Lớp Dữ liệu và Dịch vụ (Data & Services Layer).....	28
2.5.1. Dịch vụ Tin nhắn Real-time: Stream API.....	28
2.5.2. Dịch vụ Lưu trữ File: UploadThing.....	29
<b>CHƯƠNG 3. ĐẶC TẢ CÁC CHỨC NĂNG .....</b>	<b>29</b>
3.1. Nhóm chức năng Quản lý Truy cập (Tác nhân: Khách).....	29
3.1.1. UC-1.1 – Đăng ký tài khoản.....	29
3.1.2. UC-1.2 – Đăng nhập hệ thống.....	31
3.2 Nhóm chức năng Tương tác Nội dung (Tác nhân: Người dùng).....	33
3.2.1. UC-2.1 – Xem nội dung bảng tin.....	33
3.2.2. UC-2.2 – Đăng bài viết mới.....	35
3.2.3. UC-2.3 – Bình luận bài viết.....	38
3.2.4. UC-2.4 - Thích bài viết.....	39
3.2.5. UC-2.5 - Lưu bài viết.....	41
3.2.6. UC-2.6 – Gửi tin nhắn (DM).....	42
3.2.7. UC-2.7 - Báo cáo vi phạm.....	45
3.2.8. UC-2.8 - Quản lý hồ sơ cá nhân.....	47
3.2.9. UC-2.9 - Quản lý bài viết cá nhân.....	48
3.2.10. UC- 2.10 - Xem thông báo.....	50
3.3. Nhóm chức năng Quản trị Hệ thống (Tác nhân: Quản trị viên).....	52
3.3.1. UC-3.1 – Quản lý tài khoản người dùng.....	52
3.3.2. UC-3.2 – Quản lý nội dung vi phạm.....	53
<b>CHƯƠNG 4. THIẾT KẾ PHẦN MỀM .....</b>	<b>56</b>
4.1 Thiết kế cơ sở dữ liệu.....	56
4.1.1. Lược đồ cơ sở dữ liệu cho các nhóm chức năng.....	56
4.1.2. Danh sách các bảng: Số thứ tự, tên bảng và ý nghĩa bảng.....	56
4.1.3. Chi tiết bảng.....	57
4.2. Thiết kế API.....	60
4.2.1. Cấu trúc thư mục API.....	60
4.2.2. Danh sách các API.....	60
4.2.3. Thiết kế chi tiết API.....	61
4.3. Thiết kế chức năng.....	64
4.3.1. Cấu trúc chung.....	64

4.3.2. Thiết kế một số chức năng.....	66
<b>CHƯƠNG 5. CÀI ĐẶT VÀ KIỂM THỬ .....</b>	<b>81</b>
5.1.Cài đặt.....	81
5.1.1. Cấu trúc thư mục mã nguồn.....	81
5.1.2. Yêu cầu môi trường & Hướng dẫn cài đặt.....	82
5.1.3.Ảnh xạ chức năng và file cài đặt.....	82
5.2. Kiểm thử .....	83
5.2.1. Giới thiệu.....	83
5.2.2. Quy trình kiểm thử.....	84
5.2.3. Thực hiện kiểm thử.....	85
<b>KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>88</b>

## **DANH MỤC HÌNH ẢNH**

Hình 1.1. Sơ đồ Use-case.....	18
Hình 2.1. Sơ đồ khung trúc tổng thể của hệ thống.....	22
Hình 3.1. Giao diện đăng ký dành cho khách.....	30
Hình 3.2. Giao diện đăng nhập dành cho khách bằng tài khoản.....	32
Hình 3.3. Giao diện đăng nhập dành cho khách bằng tài khoản Google.....	32
Hình 3.4. Giao diện bảng tin dành cho người dùng.....	34
Hình 3.5. Giao diện bảng tin những người đã follow.....	35
Hình 3.6. Giao diện đăng bài viết mới.....	37
Hình 3.7. Giao diện chi tiết bài đăng.....	37
Hình 3.8. Giao diện bài đăng theo chủ đề.....	38
Hình 3.9. Giao diện bình luận bài viết.....	39
Hình 3.10. Giao diện trước khi thích bài viết.....	40
Hình 3.11. Giao diện sau khi thích bài viết.....	40
Hình 3.12. Giao diện sau khi lưu bài viết.....	41
Hình 3.13. Giao diện gửi tin nhắn (DM).....	43
Hình 3.14. Giao diện tạo cuộc hội thoại.....	44
Hình 3.15. Giao diện tìm kiếm người dùng nhắm tin.....	44
Hình 3.16. Giao diện hộp thoại bình chọn trong tin nhắn.....	45
Hình 3.17. Giao diện sửa hồ sơ bản thân.....	48
Hình 3.18. Giao diện xóa bài viết của người dùng.....	50
Hình 3.19. Giao diện thông báo.....	51
Hình 3.20. Giao diện xóa tài khoản người dùng.....	53
Hình 3.22. Giao diện xóa bình luận của người dùng.....	54
Hình 3.23. Giao diện xóa bài viết của người dùng.....	55
Hình 4.1. Lược đồ cơ sở dữ liệu.....	56
Hình 4.2. Sơ đồ lớp phân tích chung.....	65
Hình 4.3. Sơ đồ trình tự chung.....	65
Hình 4.4. Sơ đồ lớp phân tích của chức năng đăng nhập.....	66
Hình 4.5. Sơ đồ trình tự của chức năng đăng nhập.....	67
Hình 4.6. Sơ đồ lớp phân tích của chức năng đăng ký.....	68
Hình 4.7. Sơ đồ trình tự của chức năng đăng ký.....	68
Hình 4.8. Sơ đồ lớp phân tích của chức năng quản lý bài đăng.....	69

Hình 4.9. Sơ đồ trình tự của chức năng quản lý bài đăng.....	71
Hình 4.10. Sơ đồ lớp phân tích của chức năng bình luận bài viết.....	71
Hình 4.11. Sơ đồ trình tự của chức năng bình luận bài viết.....	73
Hình 4.12. Sơ đồ lớp phân tích của chức năng thích/ lưu bài viết.....	73
Hình 4.13. Sơ đồ trình tự của chức năng thích/ lưu bài viết.....	74
Hình 4.14. Sơ đồ lớp phân tích của chức năng gửi tin nhắn (DM).....	75
Hình 4.15. Sơ đồ trình tự của chức năng gửi tin nhắn (DM).....	75
Hình 4.16. Sơ đồ lớp phân tích của chức năng cập nhật hồ sơ cá nhân.....	76
Hình 4.17. Sơ đồ trình tự của chức năng cập nhật hồ sơ cá nhân.....	77
Hình 4.18. Sơ đồ lớp phân tích của chức năng quản lý người dùng.....	78
Hình 4.19. Sơ đồ trình tự của chức năng quản lý người dùng.....	78
Hình 4.20. Sơ đồ lớp phân tích của chức năng quản lý nội dung vi phạm.....	79
Hình 4.21. Sơ đồ trình tự của chức năng quản lý nội dung vi phạm.....	80
Hình 5.1. Quy trình kiểm thử.....	84

## DANH MỤC BẢNG BIỂU

Bảng 1.1. Bảng mô tả lược đồ cơ sở dữ liệu.....	15
Bảng 1.2. Bảng ánh xạ yêu cầu nghiệp vụ với usecase .....	22
Bảng 2.1. Công nghệ sử dụng.....	24
Bảng 3.1. Đặc tả chức năng Đăng kí .....	31
Bảng 3.2. Đặc tả chức năng Đăng nhập.....	33
Bảng 3.3. Đặc tả chức năng Xem bảng tin .....	35
Bảng 3.4. Đặc tả chức năng Đăng bài viết mới .....	38
Bảng 3.5. Đặc tả chức năng Bình luận bài viết.....	39
Bảng 3.6. Đặc tả chức năng Thích bài viết.....	40
Bảng 3.7. Đặc tả chức năng Lưu bài viết.....	41
Bảng 3.8. Đặc tả chức năng Gửi tin nhắn .....	45
Bảng 3.9. Đặc tả chức năng Báo cáo vi phạm .....	47
Bảng 3.10. Quản lý hồ sơ cá nhân.....	48
Bảng 3.11. Đặc tả Use case Quản lý bài viết cá nhân.....	50
Bảng 3.12. Đặc tả usecase Xem thông báo.....	51
Bảng 3.13. Đặc tả chức năng Quản lý tài khoản người dùng.....	53
Bảng 3.14. Đặc tả chức năng Quản lý nội dung vi phạm .....	55
Bảng 4.1. Danh sách các bảng sử dụng trong CSDL.....	57
Bảng 4.2. Chi tiết bảng User.....	57
Bảng 4.3. Chi tiết bảng Session .....	58
Bảng 4.4. Chi tiết bảng Auth_key .....	58
Bảng 4.5. Chi tiết bảng Follow .....	58
Bảng 4.6. Chi tiết bảng Post.....	58
Bảng 4.7. Chi tiết bảng Media .....	58
Bảng 4.8. Chi tiết bảng Comments .....	59
Bảng 4.9. Chi tiết bảng Likes.....	59
Bảng 4.10. Chi tiết bảng Bookmarks.....	59
Bảng 4.11. Chi tiết bảng Notifications.....	59
Bảng 4.12. Các hoạt động cơ bản trong REST .....	60
Bảng 4.13. Bảng danh sách các API.....	61
Bảng 4.14. Bảng mô tả API Quản lý Xác thực người dùng.....	62
Bảng 4.15. Bảng mô tả API Quản lý thông tin tài khoản người dùng.....	62
Bảng 4.16. Bảng mô tả API Quản lý bài viết.....	63

Bảng 4.17. Bảng mô tả API Quản lý bình luận .....	63
Bảng 4.18. Bảng mô tả API Quản lý thông báo.....	63
Bảng 4.19. Bảng mô tả API Quản lý tin nhắn .....	64
Bảng 4.20. Bảng mô tả API dành riêng cho quản trị viên.....	64
Bảng 4.21. Bảng mô tả các API phụ trợ.....	64
Bảng 5.1. Chi tiết thư mục logic nghiệp vụ và routing (src/app).....	81
Bảng 5.2. Chi tiết thư mục component và UI (src/components) .....	81
Bảng 5.3. Chi tiết thư mục lõi, dịch vụ và CSDL (src/lib, prisma, auth.ts).....	82
Bảng 5.4. Ánh xạ Use Case và file mã nguồn chính.....	83
Bảng 5.5. Bảng tổng hợp ca kiểm thử cho người dùng (Test Case Summary).....	85
Bảng 5.6. Bảng tổng hợp ca kiểm thử cho Quản trị viên.....	86
Bảng 5.7. Bảng tổng hợp ca kiểm thử phi chức năng.....	86

## GIẢI THÍCH THUẬT NGỮ

Thuật ngữ	Viết đầy đủ	Ý nghĩa
API	Application Programming Interface	API đóng vai trò là một giao diện cho phép các hệ thống hoặc các thành phần phần mềm khác nhau giao tiếp và trao đổi dữ liệu với nhau.
ORM	Object-Relational Mapper	Object- Relational Mapper. Cụ thể là Prisma ORM, lớp giao tiếp giữa Next.js và Postgres.
DM	Direct Message	Tin nhắn truyền theo thời gian thực
ACID	Atomicity, Consistency, Isolation, Durability	Tính nguyên tố, Tính nhất quán, Tính cô lập, Tính bền vững
OAuth	Open Authorization	Open Authorization. Cụ thể là Google OAuth được sử dụng cho chức năng đăng nhập/xác thực.
FCP	First Contentful Paint	FCP đo lường thời gian từ khi trang bắt đầu tải cho đến khi bất kỳ phần nội dung nào của trang (text, hình ảnh, hoặc các phần tử khác) được hiển thị lần đầu tiên trên màn hình trình duyệt của người dùng.
LCP	Largest Contentful Paint	LCP đo lường thời gian mà thành phần nội dung lớn nhất (ví dụ: hình ảnh chính, video, hoặc khối văn bản lớn) hiển thị trên màn hình của người dùng.
WAL	Write-Ahead Log	WAL là một cơ chế ghi nhật ký giao dịch (transaction logging mechanism) được sử dụng trong PostgreSQL.
Server Components		Thành phần của React chạy trên máy chủ (Next.js15) để giảm tải JavaScript cho client.
Server Action		Hàm chạy trên máy chủ, xử lý logic nghiệp vụ mà không cần tạo API routes thủ công.
Real-time		Tính năng giao tiếp theo thời gian thực (ví dụ: tin nhắn DM qua Stream API).
RESTful API Call		Phương thức gọi API tuân theo kiến trúc REST (Representation State Transfer) giữa Client/Stream API và Server.

Optimistic Update		Kỹ thuật cập nhật giao diện ngay lập tức trước khi nhận phản hồi từ máy chủ, giúp trải nghiệm người dùng nhanh hơn.
Full-text Search		Chức năng tìm kiếm toàn văn, được tích hợp và tối ưu trên PostgreSQL.
Validation		Quá trình kiểm tra tính hợp lệ của dữ liệu nhập vào (sử dụng Zod trong dự án).
Session		Phiên đăng nhập của người dùng, được quản lý bởi Lucia Authentication.
Revalidation		Kỹ thuật xác thực lại cache của dữ liệu (của Next.js) để đảm bảo dữ liệu luôn mới.
Guest		Khách truy cập, người chưa đăng nhập.
User		Người dùng đã đăng nhập, thành viên chính thức của cộng đồng.
Admin		Quản trị viên, người có quyền hạn cao nhất để quản lý hệ thống.
Follows		Mối quan hệ "theo dõi" giữa người dùng.
Bookmarks		Chức năng lưu bài viết lại để xem sau.
Test Case		Kịch bản kiểm thử cụ thể để xác minh chức năng hệ thống.
Notifications		Hệ thống thông báo được thiết kế để giữ chân người dùng và thông báo cho họ về các tương tác quan trọng xảy ra trên nền tảng.
Profile		Danh tính và cá nhân hóa của người dùng.

## LỜI GIỚI THIỆU

Trong bối cảnh công nghệ thông tin phát triển như vũ bão, các nền tảng mạng xã hội đã trở thành một phần không thể thiếu trong đời sống, cách chúng ta học tập và chia sẻ thông tin. Tuy nhiên, các nền tảng "đại chúng" quy mô lớn thường khiến nội dung bị phân mảnh, loãng và thiếu tính tập trung. Điều này tạo ra một nhu cầu rõ rệt về các không gian cộng đồng nhỏ hơn, linh hoạt và tập trung vào các chủ đề cụ thể, nơi người dùng có thể kết nối và trao đổi một cách giá trị.

Đáp ứng nhu cầu đó, báo cáo này trình bày chi tiết dự án xây dựng Hệ thống Mạng xã hội DevShareLite — một nền tảng được thiết kế với mục tiêu cung cấp trải nghiệm tương tác nhanh, mượt mà và tập trung vào cộng đồng. Hệ thống cho phép người dùng chia sẻ bài viết, tham gia thảo luận, theo dõi các chủ đề quan tâm, và kết nối trực tiếp với nhau qua tin nhắn riêng tư (DM).

Điểm nổi bật cốt lõi của dự án nằm ở việc áp dụng một kiến trúc kỹ thuật hiện đại và tối ưu. Thay vì sử dụng các mô hình client-server truyền thống, DevShareLite được xây dựng hoàn toàn trên nền tảng Next.js 15 (App Router). Dự án tận dụng triệt để các công nghệ tiên tiến nhất như Server Components để giảm tải cho client và Server Actions để xử lý logic nghiệp vụ một cách an toàn và hiệu quả.

Để đảm bảo tính toàn vẹn dữ liệu và hiệu suất, hệ thống sử dụng PostgreSQL làm cơ sở dữ liệu chính (qua Prisma ORM), Lucia Auth cho việc xác thực an toàn (bao gồm cả Google OAuth), và tích hợp các dịch vụ API bên thứ ba mạnh mẽ như Stream API cho tin nhắn real-time và UploadThing để xử lý file media.

Báo cáo này sẽ trình bày toàn bộ quá trình thực hiện dự án, từ bước phân tích yêu cầu nghiệp vụ, thiết kế kiến trúc hệ thống, lựa chọn công nghệ, cho đến các bước cài đặt và kiểm thử chi tiết để chứng minh tính hiệu quả của mô hình.

# CHƯƠNG 1. TỔNG QUAN VỀ PHẦN MỀM

## 1.1. Mô tả bài toán

- Trong bối cảnh công nghệ thông tin phát triển nhanh chóng, nhu cầu chia sẻ kiến thức, trao đổi kinh nghiệm và học tập trong cộng đồng ngày càng tăng cao. Tuy nhiên, các nền tảng mạng xã hội chung thường quá rộng, nội dung thiếu tập trung, và khó khăn trong việc tìm kiếm, phân loại theo các chủ đề hoặc sở thích cụ thể.

- Từ thực tế đó, nhóm đề xuất xây dựng phần mềm mạng xã hội chia sẻ và kết nối cộng đồng (DevShareLite) — một hệ thống hỗ trợ người dùng trong việc học tập, thảo luận, và chia sẻ các bài viết, kinh nghiệm về nhiều chủ đề. Hệ thống này hướng tới việc tạo ra một cộng đồng người dùng có khả năng trao đổi, lưu trữ tài nguyên, và tìm kiếm nội dung một cách nhanh chóng và thuận tiện.

- Phần mềm sẽ được sử dụng cho nhiều đối tượng người dùng khác nhau với nhiều mục tiêu khác nhau:

+ Đối với Người dùng (User), là những thành viên chính thức của cộng đồng. Sau khi đăng ký tài khoản, họ có thể chủ động tham gia bằng cách đăng bài viết chia sẻ kiến thức, kinh nghiệm hoặc đặt câu hỏi. Các bài viết này sẽ được đăng tải lên hệ thống ngay lập tức mà không cần qua kiểm duyệt. Bên cạnh việc tạo nội dung, người dùng còn có thể bình luận và tương tác với các thành viên khác. Để cá nhân hóa trải nghiệm, hệ thống cho phép họ lưu lại các bài viết yêu thích và theo dõi những tác giả mà họ quan tâm.

+ Với vai trò là Quản trị viên (Admin), họ có quyền hạn cao nhất, chịu trách nhiệm duy trì sự lành mạnh của cộng đồng. Một nhiệm vụ quan trọng của Admin là giám sát nội dung sau khi đã được đăng tải, chủ động xóa các bài đăng, bình luận hoặc tài khoản của người dùng nếu phát hiện vi phạm quy định của nền tảng. Ngoài các quyền quản trị này, Admin có đầy đủ các chức năng của một Người dùng thông thường, bao gồm đăng bài, bình luận và tương tác.

+ Đối với Khách truy cập (Guest), họ có quyền truy cập rất hạn chế. Khi truy cập vào hệ thống, họ chỉ có thể thấy trang Đăng nhập và Đăng ký. Họ không thể xem hay tìm kiếm bất kỳ nội dung nào. Để tham gia sâu hơn vào cộng đồng, chẳng hạn như xem bài viết, đăng bài, hay bình luận, hệ thống sẽ yêu cầu họ phải thực hiện đăng ký tài khoản và trở thành Người dùng chính thức.

- DevShareLite được thiết kế như một mạng xã hội thông thường, tập trung vào việc chia sẻ nội dung. Bài viết được phân loại khoa học theo các chủ đề (tags) do người dùng tự định nghĩa, kết hợp với chức năng tìm kiếm nâng cao theo chủ đề hoặc từ khóa. Nền tảng này

cũng xây dựng một không gian cộng đồng để hỏi – đáp và chia sẻ kinh nghiệm. Hệ thống cung cấp các công cụ cho Admin để xử lý các vi phạm (xóa bài, xóa tài khoản) nhằm duy trì trật tự chung.

- Phần mềm DevShareLite mang lại lợi ích thiết thực cho nhiều nhóm đối tượng. Đối với người dùng, đây là nguồn tài liệu, thông tin đa dạng và là nơi nhận phản hồi nhanh chóng từ những người có chung sở thích. Trong khi đó, những người có kinh nghiệm có thể tận dụng nền tảng để chia sẻ kiến thức và xây dựng uy tín cá nhân. Về phía quản trị viên, phần mềm giúp đơn giản hóa việc quản lý và xử lý các vi phạm sau khi chúng phát sinh. Trên hết, DevShareLite tạo ra một không gian học tập và chia sẻ tri thức tập trung, góp phần thúc đẩy tinh thần hợp tác trong cộng đồng người dùng.

## 1.2. Yêu cầu bài toán

### 1.2.1. Chuẩn bị dữ liệu

Để hệ thống hoạt động ổn định và thuận tiện cho người dùng, việc chuẩn bị dữ liệu ban đầu là bước rất quan trọng.

Cụ thể, phần mềm cần có sẵn các nhóm dữ liệu cơ bản sau:

- Danh mục chủ đề (Topics): Bao gồm các lĩnh vực phổ biến và đa dạng như Du lịch, Âm thực, Thể thao, Công nghệ, Giáo dục, giúp người dùng dễ dàng tìm và chia sẻ bài viết phù hợp.
- Danh mục bài viết (Posts): Hệ thống cần một số bài viết mẫu để hiển thị trên trang chủ, phục vụ kiểm thử giao diện và trải nghiệm người dùng.
- Vai trò người dùng (Roles):
  - o Guest (Khách): Người chưa đăng nhập. Vai trò này chỉ có thể thấy trang Đăng nhập và Đăng ký, không thể xem bất kỳ nội dung nào bên trong.
  - o User (Người dùng): Có thể đăng bài (bài viết được đăng lên ngay không cần duyệt), bình luận, gửi tin nhắn, hoặc lưu bài viết.
  - o Admin (Quản trị viên): Có toàn quyền quản lý hệ thống. Admin có thể xóa bài đăng, xóa bình luận, hoặc xóa tài khoản người dùng vi phạm. Admin cũng sở hữu tất cả các chức năng của một User.

Dữ liệu khởi tạo được lưu trữ trong cơ sở dữ liệu, gồm các bảng chính:

Tên bảng	Mô tả
Users	Lưu thông tin tài khoản người dùng: username, email, mật khẩu, vai trò, trạng thái.
Posts	Lưu bài viết, bao gồm tiêu đề, nội dung, chủ đề, người đăng và ngày đăng.
Comments	Lưu bình luận của người dùng trong từng bài viết.
Messages	Lưu tin nhắn riêng giữa các người dùng.
Bookmarks	Quản lý danh sách bài viết được người dùng lưu lại để xem sau.
Notifications	Thông báo khi có bài viết mới, bình luận, hoặc phản hồi.
Follows	Quản lý mối quan hệ “theo dõi” giữa người dùng.
Sessions	Quản lý phiên đăng nhập của người dùng.
Auth_Keys	Lưu trữ khóa xác thực cho cơ chế đăng nhập và xác thực.
Post_Media	Lưu trữ các tệp đa phương tiện (ảnh, video) được gắn kèm trong bài viết

Bảng 1.1. Bảng mô tả lược đồ cơ sở dữ liệu

### 1.2.2. Thực hiện quy trình

- Đăng ký – Đăng nhập: Người dùng mới tạo tài khoản và đăng nhập vào hệ thống.
- Tương tác với nội dung: Sau khi đăng nhập, người dùng có thể xem bài viết, đăng bài mới (được đăng lên trực tiếp), bình luận hoặc nhắn tin.
- Quản lý nội dung: Admin giám sát nội dung sau khi đã được đăng tải. Nếu phát hiện vi phạm, Admin sẽ tiến hành xử lý (xóa bài, xóa bình luận hoặc tài khoản).

### 1.3. Yêu cầu nghiệp vụ

#### 1.3.1. Mô tả quy trình nghiệp vụ

Phần mềm DevShareLite là một mạng xã hội trực tuyến dành cho cộng đồng, giúp họ chia sẻ bài viết, thảo luận về các chủ đề quan tâm và học hỏi lẫn nhau.

Hệ thống được vận hành thông qua các quy trình nghiệp vụ chính sau:

Bước 1: Khởi tạo và quản trị dữ liệu ban đầu

- Admin đăng nhập vào hệ thống quản trị và khởi tạo dữ liệu ban đầu:
  - o Tạo các tài khoản mẫu cho User và Admin.
  - o Cấu hình quyền truy cập và phân hệ chức năng cho từng vai trò (User và Admin).
- Dữ liệu này được lưu trữ trong cơ sở dữ liệu và có thể chỉnh sửa bất cứ lúc nào.

#### Bước 2: Đăng ký và xác thực tài khoản

- Khách truy cập (Guest) có thể đăng ký tài khoản mới bằng email, tên người dùng và mật khẩu.
- Hệ thống sẽ:
  - o Kiểm tra thông tin trùng lặp (email, username).
  - o Xác thực định dạng email.
  - o Mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu.
- Sau khi đăng ký thành công, người dùng được gán vai trò mặc định là User.

#### Bước 3: Hoạt động tương tác trong mạng xã hội

Sau khi đăng nhập, User có thể:

- o Viết bài chia sẻ (có tiêu đề, nội dung, tag, và hình ảnh minh họa). Bài viết được đăng tải ngay lập tức.
- o Chính sửa hoặc xóa bài viết của chính mình.
- o Bình luận, thích, lưu hoặc chia sẻ bài viết của người khác.
- o Gửi tin nhắn riêng (DM) cho các thành viên khác.
- Hệ thống sẽ:
  - o Tự động ghi nhận các hành động (lượt xem, lượt thích, bình luận...).
  - o Gửi thông báo cho người dùng khi có phản hồi, tin nhắn mới hoặc khi bài viết được quan tâm nhiều.

#### Bước 4: Kiểm duyệt và xử lý nội dung

- Admin có toàn quyền quản lý:
  - o Cấu hình chuyên mục, phân quyền người dùng.
  - o Quản lý báo cáo vi phạm: Admin chủ động theo dõi nội dung và xử lý các trường hợp vi phạm (như xóa bài đăng, xóa bình luận, hoặc xóa/khóa tài khoản người dùng).
  - o Sao lưu dữ liệu định kỳ.

### **1.3.2. Các yêu cầu nghiệp vụ**

❖BR1: Nhóm chức năng chuẩn bị dữ liệu ban đầu

Mục tiêu: Khởi tạo hệ thống ban đầu, tạo các danh mục, tài khoản và phân quyền cần thiết để hệ thống hoạt động.

Mô tả chi tiết: Quản trị viên (Admin) có thể khởi tạo hệ thống bằng cách tạo các danh mục chủ đề và phân quyền người dùng (User, Admin), đồng thời thiết lập cấu hình chung như

banner và logo. Toàn bộ dữ liệu này được lưu trong CSDL, cho phép chỉnh sửa hoặc xóa. Để tiết kiệm thời gian, hệ thống hỗ trợ nhập dữ liệu mẫu từ file Excel, giúp nhanh chóng có đủ cấu trúc và các thông số cơ bản để đi vào hoạt động.

❖ BR2: Nhóm chức năng thực hiện đăng ký

Mục tiêu: Cung cấp cơ chế đăng ký, xác thực và quản lý tài khoản người dùng.

Mô tả chi tiết: Người dùng có thể tự đăng ký tài khoản mới bằng email hoặc username, với tùy chọn nhận email xác nhận. Hệ thống sẽ tự động kiểm tra trùng lặp thông tin, mã hóa mật khẩu để đảm bảo an toàn trước khi lưu vào cơ sở dữ liệu. Chức năng này cũng bao gồm hỗ trợ đăng nhập, khôi phục mật khẩu khi quên và cho phép người dùng tự cập nhật hồ sơ cá nhân. Kết quả là người dùng có thể tự chủ quản lý tài khoản của mình một cách thuận tiện và bảo mật.

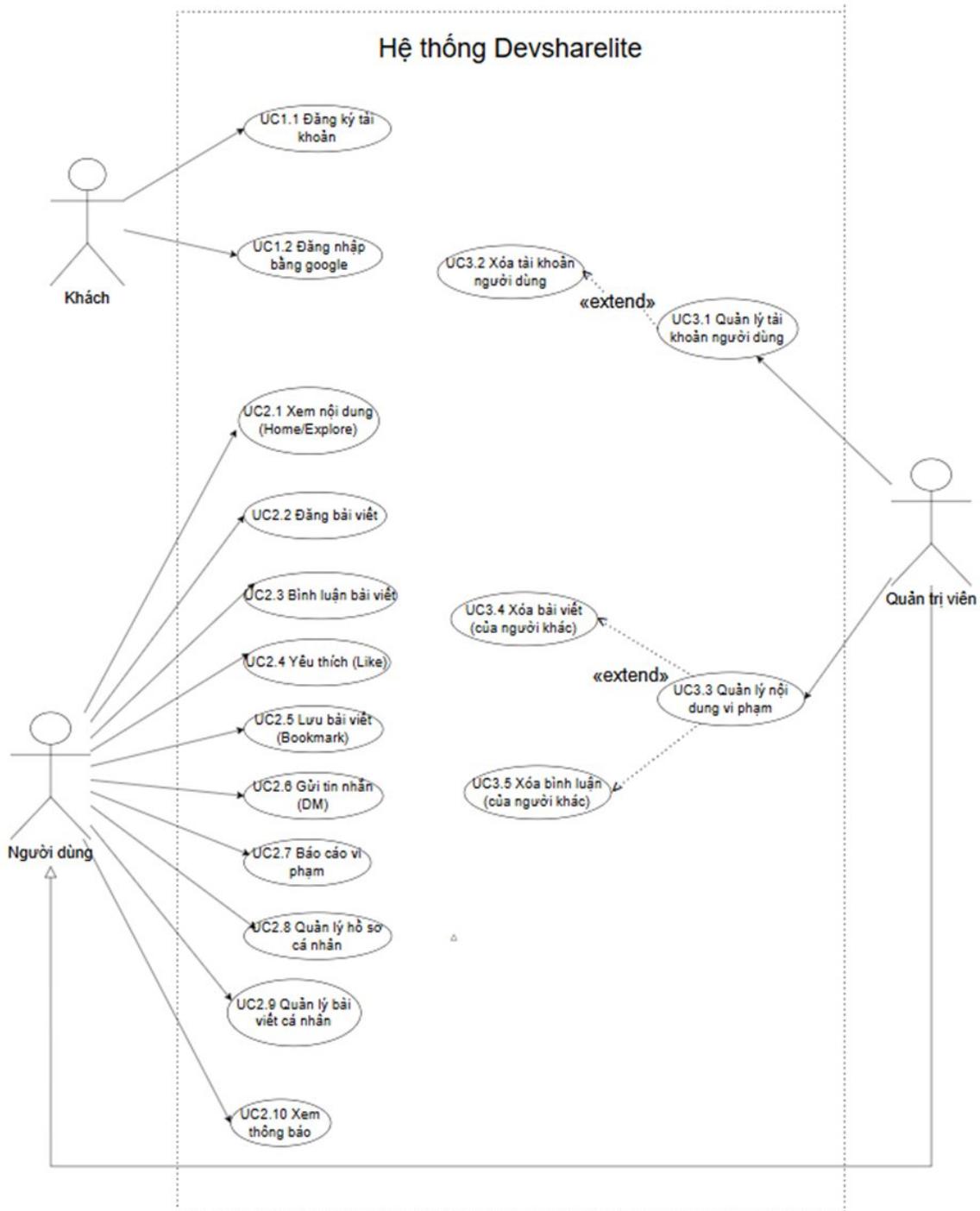
❖ BR3: Nhóm chức năng thực hiện

Mục tiêu: Hỗ trợ người dùng thực hiện các thao tác chính trong diễn đàn: đăng bài, bình luận, tương tác, và nhắn tin.

Mô tả chi tiết: Người dùng có thể chủ động viết, đăng (trực tiếp không cần duyệt), sửa hoặc xóa các bài chia sẻ kinh nghiệm, dự án và hướng dẫn của riêng mình. Họ cũng dễ dàng tương tác với nội dung của người khác thông qua việc bình luận, thích, lưu bài viết, hoặc trao đổi trực tiếp qua hộp thư cá nhân. Về phía hệ thống, nó cung cấp bộ lọc tìm kiếm (theo chủ đề, tác giả, từ khóa), tự động gợi ý các bài viết liên quan và gửi thông báo thời gian thực khi có tương tác mới. Kết quả là tạo ra một môi trường tương tác thuận tiện, giúp tăng cường tính cộng đồng và thúc đẩy hoạt động chia sẻ kiến thức.

## 1.4. Phân tích Use Case các chức năng chính

### 1.4.1. Sơ đồ Use Case các chức năng chính



Hình 1.1. Sơ đồ Use-case

#### **1.4.2. Các tác nhân hệ thống**

- **Khách (Guest)** là những người dùng truy cập hệ thống nhưng chưa đăng nhập. Ở vai trò này, họ không có các quyền cơ bản bao gồm xem các bài viết công khai và sử dụng chức năng tìm kiếm bài viết theo từ khóa. Để có thể tham gia tương tác hoặc đăng bài, họ sẽ cần phải thực hiện việc đăng ký một tài khoản mới.
- **User (Người dùng)** User là những thành viên đã đăng nhập vào hệ thống. Họ có đầy đủ quyền tương tác xã hội, bao gồm việc tạo, chỉnh sửa và xóa các bài viết của chính mình, cũng như thích, bình luận hoặc lưu lại bài viết của người khác. Ngoài ra, họ có thể gửi tin nhắn cá nhân (DM) để trao đổi riêng tư.
- **Admin (Quản trị viên)** Admin là người quản lý toàn bộ hệ thống với quyền hạn cao nhất. Trách nhiệm của họ bao gồm việc quản lý tất cả người dùng và phân quyền truy cập, tạo hoặc chỉnh sửa các chủ đề, và thiết lập cấu hình chung của hệ thống. Admin cũng là người duy nhất có thể xóa bình luận, bài viết, người dùng

#### **1.4.3. Các nhóm Use Case**

##### ❖ Nhóm UC1: Nhóm chức năng Khách và Xác thực

- UC1.1 – Đăng ký tài khoản mới Ca sử dụng này mô tả quá trình Khách (Guest) tạo tài khoản. Người dùng sẽ nhập các thông tin bắt buộc vào biểu mẫu đăng ký. Hệ thống sẽ tự động tiếp nhận, kiểm tra tính hợp lệ và sự trùng lặp của thông tin trong cơ sở dữ liệu trước khi chính thức lưu tài khoản mới.
- UC1.2 – Đăng nhập bằng google Trong UC1.2, Khách (Guest) đã có tài khoản sẽ chọn đăng nhập thông qua dịch vụ của Google. Hệ thống sẽ chuyển hướng người dùng đến trang xác thực của Google. Sau khi xác thực thành công, Google trả về thông tin người dùng và hệ thống sẽ cấp phiên truy cập (session), cho phép người dùng vào sử dụng các chức năng của thành viên.

##### ❖ Nhóm UC2: Nhóm chức năng Người dùng (Tương tác & Quản lý)

- UC2.1 – Xem nội dung (Home/Explore) Ca sử dụng này mô tả việc Khách (Guest) hoặc Người dùng (User) xem các bài viết công khai trên trang chủ (Home) hoặc trang khám phá (Explore). Người dùng có thể duyệt qua danh sách các bài đăng, xem nội dung mà không cần thực hiện hành động cụ thể nào.

- UC2.2 – Đăng bài viết Trong ca sử dụng này, Người dùng (User) thực hiện chức năng chính là tạo nội dung. Họ sẽ soạn thảo một bài viết để chia sẻ kinh nghiệm, bao gồm tiêu đề và nội dung, sau đó đăng tải lên hệ thống.
- UC2.3 – Bình luận bài viết Đây là chức năng tương tác cốt lõi, cho phép Người dùng (User) để lại ý kiến hoặc câu hỏi của mình bằng cách gửi bình luận bên dưới một bài viết. Ngay sau khi gửi, hệ thống sẽ lập tức lưu trữ và hiển thị bình luận đó công khai.
- UC2.4 – Yêu thích (Like) Ca sử dụng này cho phép Người dùng (User) bày tỏ sự ủng hộ hoặc đánh giá cao nội dung. Họ có thể nhấn nút "Thích" (Like) để hệ thống ghi nhận và cập nhật số lượng lượt thích cho bài viết.
- UC2.5 – Lưu bài viết (Bookmark) Ca sử dụng này cho phép Người dùng (User) lưu lại các bài viết mà họ quan tâm. Họ có thể sử dụng chức năng "Lưu bài viết" (Bookmark) để đưa bài viết vào danh sách cá nhân, giúp dễ dàng tìm lại và đọc sau này.
- UC2.6 – Gửi tin nhắn riêng (DM) Chức năng này cho phép giao tiếp riêng tư giữa các thành viên. Hai người dùng bất kỳ có thể khởi tạo một cuộc hội thoại và trao đổi trực tiếp với nhau thông qua hệ thống hộp thư cá nhân (Direct Message), tách biệt hoàn toàn khỏi các khu vực thảo luận công khai.
- UC2.7 – Báo cáo vi phạm Khi một Người dùng (User) phát hiện nội dung vi phạm các quy định (như spam, nội dung không lành mạnh) ở bài viết hoặc bình luận, họ có thể sử dụng chức năng này để gửi báo cáo. Hệ thống sẽ tự động ghi nhận và gửi thông báo đến Quản trị viên (Admin) để xem xét.
- UC2.8 – Quản lý hồ sơ cá nhân Ca sử dụng này cho phép Người dùng (User) tự điều chỉnh hồ sơ của mình. Sau khi đăng nhập, họ có thể truy cập vào trang cá nhân để thay đổi ảnh đại diện (avatar), cập nhật phần giới thiệu về bản thân, hoặc các thông tin liên lạc khác.
- UC2.9 – Quản lý bài viết cá nhân Ca sử dụng này mô tả quyền quản lý nội dung sau khi đăng. Người dùng (User) có toàn quyền truy cập, chỉnh sửa nội dung hoặc xóa các bài viết do chính họ tạo ra thông qua trang quản lý cá nhân.
- UC2.10 – Xem thông báo Ca sử dụng này cho phép Người dùng (User) nhận và xem các thông báo liên quan đến hoạt động của họ. Hệ thống sẽ gửi thông báo khi có người bình luận, yêu thích bài viết, hoặc gửi tin nhắn mới, giúp người dùng cập nhật các tương tác.

#### ❖ Nhóm UC3: Nhóm chức năng Quản trị

- UC3.1 – Quản lý tài khoản người dùng Đây là chức năng dành riêng cho Quản trị viên (Admin). Admin có quyền truy cập vào danh sách tất cả tài khoản trong hệ

thống để xem thông tin và thực hiện các thao tác quản lý. Ca sử dụng này được mở rộng (<<extend>>) bởi UC "Xóa tài khoản người dùng".

- UC3.2 – Xóa tài khoản người dùng Đây là ca sử dụng mở rộng (<<extend>>) từ UC3.1 – Quản lý tài khoản người dùng. Khi một tài khoản vi phạm nghiêm trọng, Admin có thể thực hiện chức năng này để xóa tài khoản đó vĩnh viễn ra khỏi hệ thống.
- UC3.3 – Quản lý nội dung vi phạm Đây là chức năng dành riêng cho Quản trị viên (Admin) để xử lý các báo cáo vi phạm (từ UC2.6). Admin sẽ xem xét các nội dung bị báo cáo (bài viết, bình luận) và đưa ra quyết định xử lý. Ca sử dụng này được mở rộng (<<extend>>) bởi các UC "Xóa bài viết" và "Xóa bình luận".
- UC3.4 – Xóa bài viết (của người khác) Đây là ca sử dụng mở rộng (<<extend>>) từ UC3.3 – Quản lý nội dung vi phạm. Sau khi xem xét báo cáo, nếu một bài viết vi phạm quy định, Admin có quyền xóa bài viết đó khỏi hệ thống.
- UC3.5 – Xóa bình luận (của người khác) Đây là ca sử dụng mở rộng (<<extend>>) từ UC3.3 – Quản lý nội dung vi phạm. Nếu một bình luận chứa nội dung không phù hợp hoặc vi phạm, Admin có quyền xóa bình luận đó.

#### **1.4.4. Bảng ảnh xạ yêu cầu nghiệp vụ với các Use Case**

BR	UC	Tác nhân liên quan	Mô tả tóm tắt
BR1	UC1.1 – Đăng ký tài khoản mới UC1.2 – Đăng nhập bằng google	Guest	Nhóm chức năng cho phép người dùng bên ngoài (Khách) đăng ký tài khoản và xác định trên hệ thống.
BR2	UC2.1 – Xem nội dung (Home/Explore) UC2.2 – Đăng bài viết UC2.3 – Bình luận bài viết UC2.4 – Yêu thích (Like) UC2.5 Lưu bài viết (Bookmark) UC2.6 – Gửi tin nhắn riêng (DM) UC2.7 – Báo cáo vi phạm UC2.8 – Quản lý hồ sơ cá nhân UC2.9 – Quản lý bài viết cá nhân UC2.10 – Xem thông báo	Guest, User, Admin	Hỗ trợ toàn bộ hoạt động xem nội dung, tương tác, chia sẻ và quản lý thông tin cá nhân của người dùng.
BR3	UC3.1 – Quản lý tài khoản người dùng UC3.2 – Xóa tài khoản người dùng	User, Admin	Cung cấp các công cụ cho quản trị viên để duy trì và kiểm

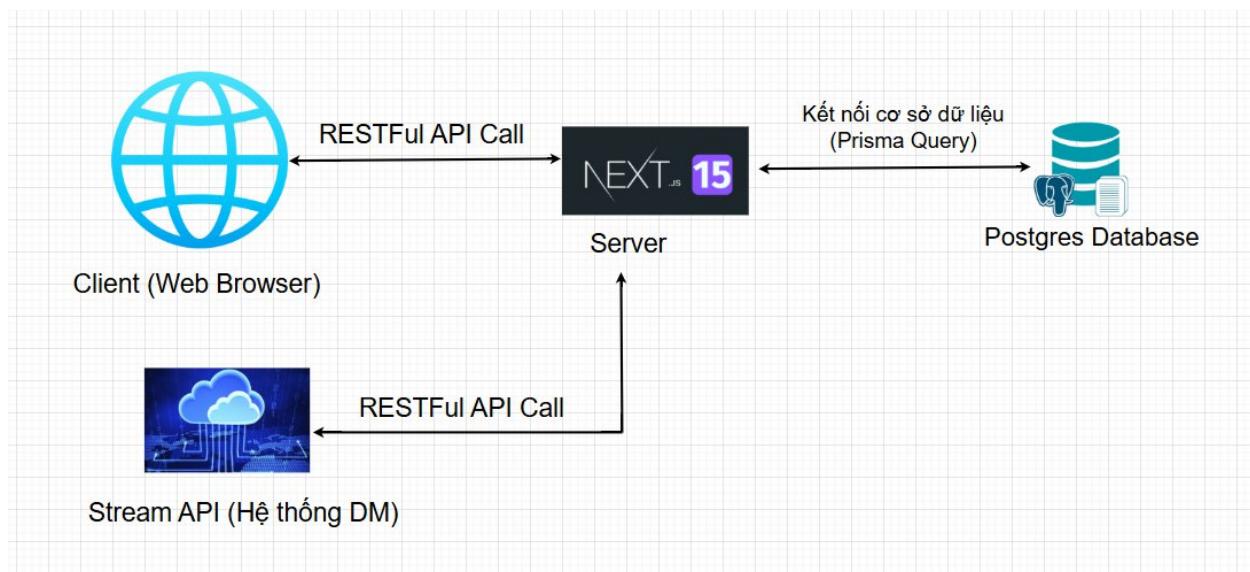
UC3.3 – Quản lý nội dung vi phạm UC3.4 – Xóa bài viết (của người khác) UC3.5 – Xóa bình luận (của người khác)		duyệt hệ thống, bao gồm quản lý người dùng và nội dung.
---	--	---

Bảng 1.2. Bảng ánh xạ yêu cầu nghiệp vụ với usecase

## CHƯƠNG 2. KIẾN TRÚC TỔNG THỂ CỦA HỆ THỐNG

### 2.1. Sơ đồ kiến trúc tổng thể

#### 2.1.1. Sơ đồ



Hình 2.1. Sơ đồ kiến trúc tổng thể của hệ thống

#### 2.1.2. Mô tả sơ đồ

Kiến trúc của DevShareLite là một hệ thống full-stack Next.js 15 hiện đại. Giao diện (Client) sử dụng React, Shadcn UI và TanStack React Query để tối ưu trải nghiệm (như cuộn vô hạn, cập nhật lướt quan), kết hợp React Hook Form/Zod để xác thực biểu mẫu. Phía máy chủ (Server) tận dụng Server Components để render và Server Actions để xử lý logic, dùng Prisma ORM để giao tiếp với Postgres Database (lưu trữ chính) và Lucia Auth để xác thực (hỗ trợ cả Google OAuth). Hệ thống được mở rộng bằng hai dịch vụ bên thứ ba: UploadThing để xử lý file upload và Stream API để cung cấp tính năng chat (DM) real-time.

#### 2.1.3. Giới thiệu các công nghệ sử dụng

Nhóm	Công nghệ	Mục đích sử dụng
------	-----------	------------------

Framework chính	Next.js 15	Framework React fullstack cung cấp Server Components, Server Actions, routing, và rendering.
Giao diện (UI/Frontend)	Tailwind CSS	Framework CSS utility-first để xây dựng giao diện nhanh chóng và tùy biến cao.
	Shadcn UI	Bộ sưu tập các component UI đẹp mắt, dễ tùy biến, xây dựng trên Tailwind CSS.
	React Hook Form & Zod	Quản lý trạng thái biểu mẫu (form) và xác thực dữ liệu (validation) real-time.
	TipTap Editor	Trình soạn thảo văn bản "headless" (không giao diện) mạnh mẽ, hỗ trợ hashtag (#) và mentions (@).
	Themes (Tối/Sáng)	Cung cấp giao diện Tối (Dark mode), Sáng (Light mode) và Tự động (System theme).
Quản lý dữ liệu	TanStack React Query	Quản lý, cache, và đồng bộ hóa dữ liệu máy chủ, hỗ trợ cuộn vô hạn và optimistic updates.
	Server Actions	Xử lý các thay đổi dữ liệu (mutations) trực tiếp từ máy chủ mà không cần tạo API routes thủ công.
Cơ sở dữ liệu	Postgres DB	Hệ quản trị CSDL quan hệ đối tượng (ORDBMS) mạnh mẽ, lưu trữ dữ liệu chính.
	Prisma ORM	Lớp giao tiếp (ORM) an toàn kiểu (type-safe) giữa Next.js và Postgres.
Xác thực	Lucia Authentication	Thư viện xác thực linh hoạt, hỗ trợ session và tích hợp OAuth (Google).

Dịch vụ bên thứ ba	UploadThing	Dịch vụ xử lý file upload, hỗ trợ kéo-thả và copy-paste.
	Stream	Nền tảng API cung cấp hạ tầng cho hệ thống tin nhắn (DM) real-time.
Tính năng cốt lõi	Notifications, Likes, Follows, Comments, Bookmarks	Các tính năng mạng xã hội cơ bản, được xây dựng dựa trên logic của Next.js và CSDL Postgres.
Tìm kiếm	Full-text Search	Chức năng tìm kiếm toàn văn, được tích hợp và tối ưu trên Postgres.
Tối ưu hóa	Advanced Caching & Revalidation	Các kỹ thuật cache và xác thực lại dữ liệu của Next.js để đảm bảo hiệu suất cao.

Bảng 2.1. Công nghệ sử dụng

## 2.2. Hệ quản trị cơ sở dữ liệu



### 2.2.1. Giới thiệu

Hệ thống DevShareLite sử dụng PostgreSQL (Postgres) làm hệ quản trị cơ sở dữ liệu (DBMS). Postgres là một hệ quản trị cơ sở dữ liệu quan hệ đối tượng (ORDBMS) mã nguồn mở, nổi tiếng với độ tin cậy, tính toàn vẹn dữ liệu mạnh mẽ và hiệu suất cao. Nó hỗ trợ đầy đủ các chuẩn SQL và cung cấp nhiều tính năng nâng cao như kiểu dữ liệu tùy chỉnh, lập chỉ mục phức tạp và khả năng mở rộng mạnh mẽ. Trong dự án này, việc tương tác với Postgres được thực hiện thông qua Prisma ORM. Prisma là một ORM (Object-Relational Mapper) thế hệ mới, cung cấp một API an toàn kiểu (type-safe) để thực hiện các truy vấn CSDL, giúp giảm thiểu lỗi và tăng tốc độ phát triển.

### 2.2.2. Thành phần chính kiến trúc (của Postgres)

Kiến trúc của PostgreSQL hoạt động theo mô hình client-server. Các thành phần chính bao gồm:

1. Postmaster (Tiến trình chủ): Đây là tiến trình cha, lắng nghe các kết nối từ client. Khi một client kết nối, Postmaster sẽ tạo ra một tiến trình con (backend process) riêng biệt để xử lý tất cả các truy vấn từ client đó.
2. Backend Process (Tiến trình nền): Mỗi client được phục vụ bởi một tiến trình nền riêng. Tiến trình này nhận truy vấn SQL từ client, phân tích (parse), tối ưu hóa (optimize) và thực thi truy vấn đó.
3. Shared Memory (Bộ nhớ chia sẻ): Một vùng bộ nhớ được chia sẻ giữa tất cả các tiến trình Postgres, chứa các dữ liệu đệm (shared buffers), WAL (Write-Ahead Log) buffers, và thông tin về các khóa (locks).
4. Data Files (Tệp dữ liệu): Nơi dữ liệu thực sự được lưu trữ trên đĩa, được tổ chức thành các bảng, chỉ mục (indexes) và các tệp cấu hình.
5. WAL (Write-Ahead Log): Một cơ chế đảm bảo tính toàn vẹn dữ liệu (ACID). Mọi thay đổi dữ liệu trước khi được ghi vào tệp dữ liệu chính đều được ghi vào WAL. Điều này cho phép phục hồi (recovery) trong trường hợp xảy ra sự cố.

### **2.2.3. *Ưu và nhược điểm***

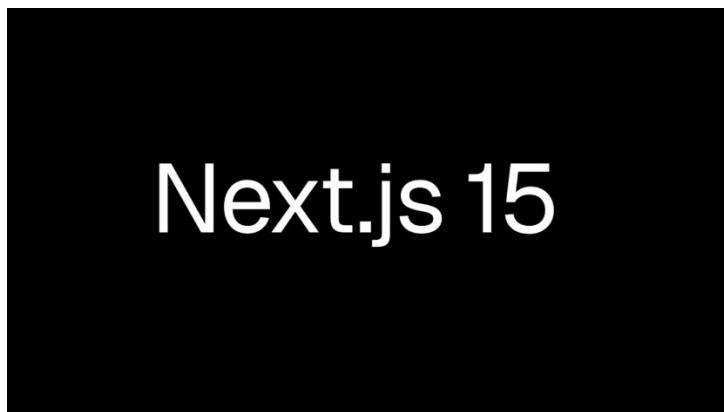
Ưu điểm:

- Độ tin cậy và Ôn định: Nổi tiếng với khả năng hoạt động ổn định trong thời gian dài và đảm bảo tính toàn vẹn dữ liệu (ACID).
- Khả năng mở rộng: Hỗ trợ nhiều kiểu dữ liệu phức tạp (JSON, mảng, hình học) và khả năng tạo chỉ mục nâng cao (bao gồm cả Full-text Search).
- Chuẩn SQL: Tuân thủ chuẩn SQL rất tốt, giúp các truy vấn phức tạp trở nên dễ dàng.
- Mã nguồn mở: Hoàn toàn miễn phí và có một cộng đồng hỗ trợ lớn mạnh.

Nhược điểm:

- Phức tạp trong quản trị: So với các CSDL đơn giản hơn như MySQL hay SQLite, việc cấu hình và tối ưu (tuning) Postgres có thể phức tạp hơn.
- Hiệu suất cho các tác vụ đơn giản: Đối với các ứng dụng chỉ đọc (read-heavy) rất đơn giản, Postgres có thể hơi "nặng ký" hơn so với các giải pháp khác.

## 2.3. Server



### 2.3.1. Giới thiệu về Next.js 15

Máy chủ ứng dụng (Application Server) của DevShareLite được xây dựng hoàn toàn bằng Next.js 15. Next.js là một framework full-stack dựa trên React, cho phép phát triển các ứng dụng web hiệu suất cao.

Khác với các ứng dụng React truyền thống (SPA) chỉ chạy ở client, Next.js cho phép chạy code ở cả client và server. Với phiên bản 15 (và kiến trúc App Router), Next.js giới thiệu các khái niệm cốt lõi là Server Components (thành phần chạy trên máy chủ) và Server Actions (hàm chạy trên máy chủ), cho phép xây dựng ứng dụng nhanh, nhẹ và an toàn hơn.

### 2.3.2. Lý do sử dụng

Việc lựa chọn Next.js 15 làm nền tảng server mang lại nhiều lợi ích chiến lược:

1. Kiến trúc Full-stack Hợp nhất: Chỉ cần một codebase (TypeScript/JavaScript) duy nhất để quản lý cả logic giao diện (frontend) và logic nghiệp vụ (backend). Điều này giúp đồng bộ hóa phát triển và giảm độ phức tạp.
2. Hiệu suất vượt trội: Bằng cách sử dụng Server Components, phần lớn logic và việc render HTML được thực hiện trên máy chủ. Client chỉ nhận về HTML và một lượng JavaScript tối thiểu, giúp tốc độ tải trang (FCP, LCP) cực kỳ nhanh.
3. Trải nghiệm phát triển (DX) tuyệt vời: Server Actions đơn giản hóa việc thay đổi dữ liệu. Thay vì phải viết API routes, client, rồi gọi API, lập trình viên có thể gọi trực tiếp một hàm (Server Action) từ component, giúp code ngắn gọn và an toàn hơn.
4. Tối ưu hóa Cache: Next.js cung cấp các cơ chế cache và "revalidation" (xác thực lại) dữ liệu rất linh hoạt, cho phép kiểm soát chặt chẽ dữ liệu nào là tĩnh, dữ liệu nào cần được cập nhật, giúp giảm tải truy vấn CSDL.

5. Tương thích hoàn hảo với Prisma: Next.js và Prisma là một "cặp đôi" hoàn hảo. Prisma cung cấp type-safety cho các truy vấn CSDL, và Next.js tận dụng điều đó để xây dựng logic backend một cách an toàn và nhanh chóng.

## 2.4. Web Client



### 2.4.1. Giới thiệu về Web Client

Web Client là thành phần chạy trên trình duyệt của người dùng, chịu trách nhiệm hiển thị giao diện và xử lý các tương tác người dùng. Trong kiến trúc của Next.js, Client không còn "nặng nề" như các ứng dụng SPA (Single Page Application) truyền thống. Thay vào đó, nó là một "client mỏng" (thin client), nhận các thành phần đã được render từ máy chủ (Server Components) và chỉ chạy các JavaScript cần thiết cho việc tương tác.

### 2.4.2. Các thành phần cơ bản trong Web Client

- React (với Client Components): Các thành phần cần tương tác (ví dụ: nút Like, Form, menu thả xuống) được đánh dấu là "use client" và chạy trên trình duyệt, sử dụng các hook của React (useState, useEffect).
- Quản lý dữ liệu Client (TanStack React Query): Mặc dù Server Actions xử lý việc thay đổi dữ liệu, TanStack Query đóng vai trò cực kỳ quan trọng trong việc lấy và cache dữ liệu phía client. Nó cung cấp các hook mạnh mẽ (như useQuery, useInfiniteQuery) để fetch dữ liệu, tự động cache, và làm mới dữ liệu nền.
- Xử lý Biểu mẫu (React Hook Form & Zod): Cung cấp trải nghiệm người dùng tốt nhất khi nhập liệu. React Hook Form tối ưu hóa hiệu suất render của form, trong khi Zod đảm bảo dữ liệu nhập vào là chính xác trước khi gửi đi (validation).
- Styling (Tailwind CSS & Shadcn UI): Cung cấp giao diện nhất quán, hiện đại và dễ bảo trì.

- Trình soạn thảo (TipTap): Cung cấp một giao diện soạn thảo văn bản giống như Notion hoặc các trình editor hiện đại, cho phép người dùng dễ dàng định dạng bài viết.

#### **2.4.3. Lý do sử dụng**

1. Trải nghiệm người dùng (UX) tức thì: Nhờ Optimistic Updates (cập nhật lác quan) từ TanStack Query. Khi người dùng nhấn "Like", giao diện sẽ cập nhật ngay lập tức trước khi máy chủ phản hồi, tạo cảm giác hệ thống cực kỳ nhanh.
2. Giảm tải JavaScript: Kiến trúc Server Components của Next.js giúp giảm đáng kể lượng JavaScript cần tải và thực thi trên client, giúp trang web hoạt động mượt mà ngay cả trên các thiết bị cấu hình thấp.
3. Tách biệt logic rõ ràng: Các công nghệ được lựa chọn giúp tách biệt rõ ràng: React Hook Form lo về form, TanStack Query lo về dữ liệu, Tailwind/Shadcn lo về giao diện. Điều này giúp code dễ đọc và dễ bảo trì.
4. Tính năng nâng cao có sẵn: Các tính năng phức tạp như cuộn vô hạn (infinite scrolling) hay kéo-thả file (drag & drop) được hỗ trợ hiệu quả bởi TanStack Query và UploadThing, giảm thời gian phát triển.

### **2.5. Lớp Dữ liệu và Dịch vụ (Data & Services Layer)**



#### **2.5.1. Dịch vụ Tin nhắn Real-time: Stream API**

Giới thiệu: Để xây dựng tính năng tin nhắn trực tiếp (DM) một cách hiệu quả và có khả năng mở rộng, hệ thống tích hợp Stream API.

Vai trò: Stream là một dịch vụ API của bên thứ ba, cung cấp một hạ tầng hoàn chỉnh cho việc chat (DM) real-time. Thay vì tự xây dựng một hệ thống WebSocket phức tạp, DevShareLite ủy thác toàn bộ việc gửi, nhận, và lưu trữ tin nhắn cho Stream.

Lý do sử dụng: Giúp giảm đáng kể độ phức tạp của máy chủ ứng dụng (Next.js), đảm bảo hiệu suất cao, độ trễ thấp cho tin nhắn, và cung cấp các tính năng chat nâng cao (như trạng thái online, "đã xem") mà không tốn công phát triển.

### **2.5.2. Dịch vụ Lưu trữ File: UploadThing**

Giới thiệu: Việc cho phép người dùng tải lên hình ảnh, video, hoặc các tệp tin khác được xử lý bởi dịch vụ UploadThing.

Vai trò: UploadThing là một dịch vụ bên thứ ba chuyên dụng cho việc upload file trong các ứng dụng Next.js. Nó cung cấp các thành phần giao diện (hỗ trợ kéo-thả, copy-paste) và một máy chủ xử lý file an toàn.

Lý do sử dụng: Giúp giảm tải (offload) băng thông và việc xử lý file nặng nề khỏi máy chủ Next.js chính. Nó đơn giản hóa quy trình upload, đảm bảo an toàn và cung cấp các URL file đã được tối ưu (CDN) để truy cập nhanh hơn.

## **CHƯƠNG 3. ĐẶC TẢ CÁC CHỨC NĂNG**

### **3.1. Nhóm chức năng Quản lý Truy cập (Tác nhân: Khách)**

#### **3.1.1. UC-1.1 – Đăng ký tài khoản**

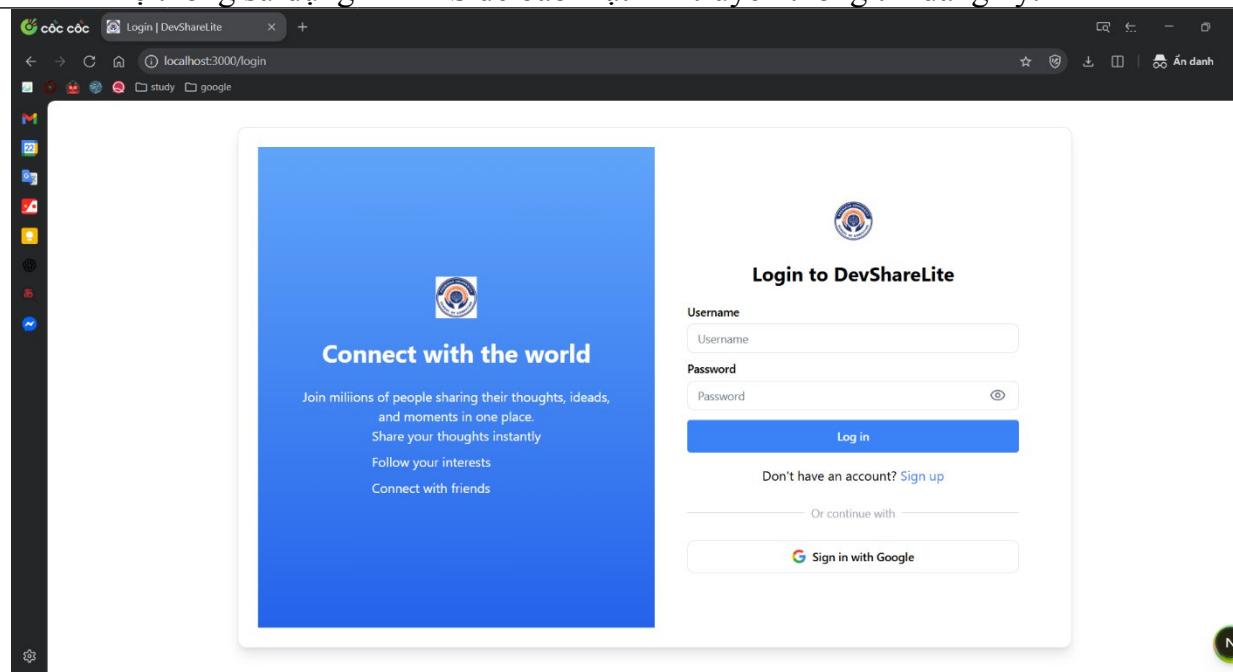
UC-1.1		<b>ĐĂNG KÝ TÀI KHOẢN</b>
Mô tả		Chức năng cho phép Khách (Guest) tạo một tài khoản. Người dùng (User) mới bằng cách cung cấp email, username và mật khẩu để tham gia hệ thống.
Tác nhân		Khách (Guest)
Tiền điều kiện		Khách đang ở trang Đăng ký (/signup).
Hậu điều kiện	Thành công	Một tài khoản User mới được tạo trong CSDL (Postgres). Khách được chuyển hướng đến trang Đăng nhập.
	Lỗi	Khách vẫn ở trang Đăng ký và nhận được thông báo lỗi.

#### **ĐẶC TẢ CHỨC NĂNG**

+ Luồng sự kiện chính/Kịch bản chính

1. Chức năng bắt đầu khi Khách chọn "Đăng ký" từ trang Đăng nhập.
2. Hệ thống hiển thị biểu mẫu (SignUpForm) yêu cầu: Email, Username, Mật khẩu.
3. Khách nhập đầy đủ thông tin. Hệ thống (Zod) tự động kiểm tra định dạng (validation) ngay khi người dùng nhập.
4. Khách nhấn nút "Đăng ký".
5. Hệ thống (Server Action) tiếp nhận thông tin.
6. Hệ thống kiểm tra Email và Username trong CSDL (Postgres) xem có bị trùng không.
7. Nếu thông tin hợp lệ (theo Zod) và không trùng, thực hiện Luồng A1: Đăng ký thành công.

8. Nếu thông tin không hợp lệ, thực hiện Luồng A2: Thông tin không hợp lệ (Validation).
  9. Nếu thông tin bị trùng, thực hiện Luồng A3: Thông tin bị trùng lặp.
- + Luồng con: A1. Đăng ký thành công
1. Hệ thống mã hóa mật khẩu.
  2. Hệ thống lưu tài khoản User mới vào CSDL (Postgres).
  3. Hệ thống chuyển hướng Khách đến trang "Trang chủ" kèm thông báo đăng ký thành công.
  4. Use case kết thúc.
- + Luồng lỗi: A2. Thông tin không hợp lệ (Validation)
1. Hệ thống (Zod) hiển thị thông báo lỗi ngay bên dưới trường nhập (ví dụ: "Email không đúng định dạng", "Mật khẩu quá ngắn").
  2. Use case quay lại Bước 3.
- + Luồng lỗi: A3. Thông tin bị trùng lặp
1. Hệ thống (Server Action) trả về lỗi.
  2. Hệ thống hiển thị thông báo lỗi "Email hoặc Username này đã tồn tại".
  3. Use case quay lại Bước 3.
- + Đầu vào / Đầu ra
- Đầu vào: Email, Username, Password (chuỗi ký tự do người dùng nhập)
  - Đầu ra: Thông báo thành công / lỗi, và chuyển hướng trang tương ứng
- + Ràng buộc và điều kiện đặc biệt
- Mật khẩu phải tối thiểu 8 ký tự, có ít nhất 1 chữ hoa và 1 số.
  - Email phải đúng định dạng (theo RFC 5322).
  - Username không chứa ký tự đặc biệt.
  - Hệ thống sử dụng HTTPS để bảo mật khi truyền thông tin đăng ký.



Bảng 3.1. Đặc tả chức năng Đăng ký

### 3.1.2. UC-1.2 – Đăng nhập hệ thống

UC-1.2	ĐĂNG NHẬP HỆ THỐNG	
Mô tả	Chức năng cho phép Khách đã có tài khoản xác thực và truy cập vào hệ thống. Hệ thống hỗ trợ đăng nhập bằng Email/Mật khẩu và đăng nhập thông qua Google OAuth2.	
Tác nhân	Khách (Guest)	
Tiền điều kiện	Khách đang ở trang Đăng nhập (/login). Tài khoản đã được đăng ký hợp lệ (đã tồn tại trong cơ sở dữ liệu).	
Hậu điều kiện	Thành công	Khách trở thành Người dùng (User), được cấp phiên (session) và chuyển hướng đến trang chủ
	Lỗi	Khách vẫn ở trang Đăng nhập và nhận được thông báo lỗi.
<b>ĐẶC TẢ CHỨC NĂNG</b>		
+ Luồng sự kiện chính/Kịch bản chính (Đăng nhập bằng Email/Password)	<ol style="list-style-type: none"> <li>Chức năng bắt đầu khi Khách truy cập trang Đăng nhập.</li> <li>Hệ thống hiển thị LoginForm yêu cầu Email và Mật khẩu.</li> <li>Khách nhập Email và Mật khẩu.</li> <li>Khách nhấn nút "Đăng nhập".</li> <li>Hệ thống (Server Action) tiếp nhận thông tin.</li> <li>Hệ thống (Lucia Auth) kiểm tra tính chính xác của Email và Mật khẩu đã mã hóa trong CSDL (Postgres).</li> <li>Nếu thông tin hợp lệ, thực hiện Luồng B1: Đăng nhập thành công.</li> <li>Nếu thông tin không hợp lệ, thực hiện Luồng B2: Đăng nhập thất bại.</li> </ol>	
+ Luồng con: B1. Đăng nhập thành công	<ol style="list-style-type: none"> <li>Hệ thống (Lucia Auth) tạo một phiên (session) cho người dùng.</li> <li>Hệ thống lưu session cookie vào trình duyệt của người dùng.</li> <li>Hệ thống chuyển hướng người dùng đến trang chủ với vai trò là Người dùng.</li> <li>Use case kết thúc.</li> </ol>	
+ Luồng lỗi: B2. Đăng nhập thất bại (Thông tin không chính xác)	<ol style="list-style-type: none"> <li>Hệ thống hiển thị thông báo lỗi "Email hoặc mật khẩu không đúng".</li> <li>Người dùng có thể nhập lại thông tin. Use case quay lại Bước 3 của Luồng chính.</li> </ol>	
+ Luồng thay thế: B3. Đăng nhập bằng Google	<ol style="list-style-type: none"> <li>Tại Bước 2, Khách nhấn nút "Đăng nhập với Google" (GoogleSignInButton).</li> <li>Hệ thống chuyển hướng Khách đến trang xác thực của Google (OAuth2).</li> <li>Khách chọn tài khoản và đồng ý.</li> <li>Google chuyển hướng về callback route (/api/auth/callback/google/route.ts).</li> <li>Hệ thống (Lucia Auth) xác thực thông tin từ Google, tìm hoặc tạo mới tài khoản nếu chưa tồn tại.</li> <li>Thực hiện Luồng B1: Đăng nhập thành công.</li> </ol>	
+ Ràng buộc & Yêu cầu phi chức năng		

- Hiệu năng: Thời gian đăng nhập < 2 giây.
  - Bảo mật:
    - Mật khẩu được mã hóa bằng bcrypt.
    - Session lưu bằng cookie bảo mật (HTTPOnly, Secure).
    - Tài khoản bị tạm khóa nếu nhập sai mật khẩu > 5 lần.
  - Tính khả dụng: Giao diện thân thiện, dễ thao tác, hiển thị lỗi rõ ràng.
- Tính tương thích: Hỗ trợ trên các trình duyệt hiện đại (Chrome, Edge, Firefox).

*Hình 3.2. Giao diện đăng nhập dành cho khách bằng tài khoản*

*Hình 3.3. Giao diện đăng nhập dành cho khách bằng tài khoản Google*

*Bảng 3.2. Đặc tả chức năng Đăng nhập*

### 3.2 Nhóm chức năng Tương tác Nội dung (Tác nhân: Người dùng)

#### 3.2.1. UC-2.1 – Xem nội dung bảng tin

UC-2.1		DĂNG BÀI VIẾT MỚI
Mô tả		Chức năng cho phép Người dùng đã đăng nhập tạo và chia sẻ một bài viết mới (bao gồm văn bản, mã nguồn, và hình ảnh) lên hệ thống.
Tác nhân		Người dùng (User)
Tiền điều kiện		Người dùng đã đăng nhập thành công.
Hậu điều kiện	Thành công	Bài viết mới được lưu vào CSDL và hiển thị trên bảng tin.
	Lỗi	Bài viết không được tạo, người dùng nhận thông báo lỗi.
<b>ĐẶC TẢ CHỨC NĂNG</b>		
+ Luồng sự kiện chính/Kịch bản chính <ol style="list-style-type: none"> <li>Chức năng bắt đầu khi Người dùng ở trang chủ (/) và tương tác với PostEditor.</li> <li>Người dùng nhập nội dung văn bản vào trình soạn thảo TipTap.</li> <li>Nếu Người dùng muốn thêm ảnh, thực hiện Luồng C1: Tải ảnh lên.</li> <li>Người dùng nhấn nút "Đăng" (Post).</li> <li>Hệ thống (Client) kiểm tra nội dung (ví dụ: không được rỗng) bằng Zod.</li> <li>Nếu hợp lệ, hệ thống thực hiện Optimistic Update (hiển thị ngay bài đăng mới trên feed) và đồng thời gửi yêu cầu (Server Action) lên máy chủ.</li> <li>Kiểm tra bài viết không rỗng;</li> <li>Kiểm tra kích thước/định dạng hình ảnh (nếu có);</li> <li>Kiểm tra các ràng buộc nội dung khác (nếu có).</li> <li>Hệ thống (Server Action) xác thực người dùng, lưu bài viết (nội dung text và URL ảnh) vào CSDL (Postgres).</li> <li>Nếu lưu thành công, thực hiện Luồng C2: Đăng bài thành công (Phía Server).</li> <li>Nếu lưu thất bại, thực hiện Luồng C3: Đăng bài thất bại.</li> </ol>		
+ Luồng con: C1. Tải ảnh lên <ol style="list-style-type: none"> <li>Người dùng kéo-thả hoặc chọn file ảnh.</li> <li>Hệ thống (Client) gọi dịch vụ UploadThing.</li> <li>UploadThing xử lý, tải file lên và trả về URL của ảnh.</li> <li>Client nhận URL và chèn ảnh vào bản preview của PostEditor (hiển thị thumbnail).</li> <li>PostEditor hiển thị ảnh đã tải lên.</li> <li>Người dùng có thể xóa/ đổi ảnh trước khi nhấn Post</li> <li>Luồng quay lại bước 4 của Luồng chính.</li> </ol>		
+ Luồng con: C2. Đăng bài thành công (Phía Server) <ol style="list-style-type: none"> <li>Server lưu bản ghi bài viết (posts) vào bảng posts (fields: id, user_id, content, created_at, updated_at, ...).</li> <li>Nếu có attachments thì lưu vào bảng post_attachments hoặc tương đương (url,</li> </ol>		

type, post\_id).

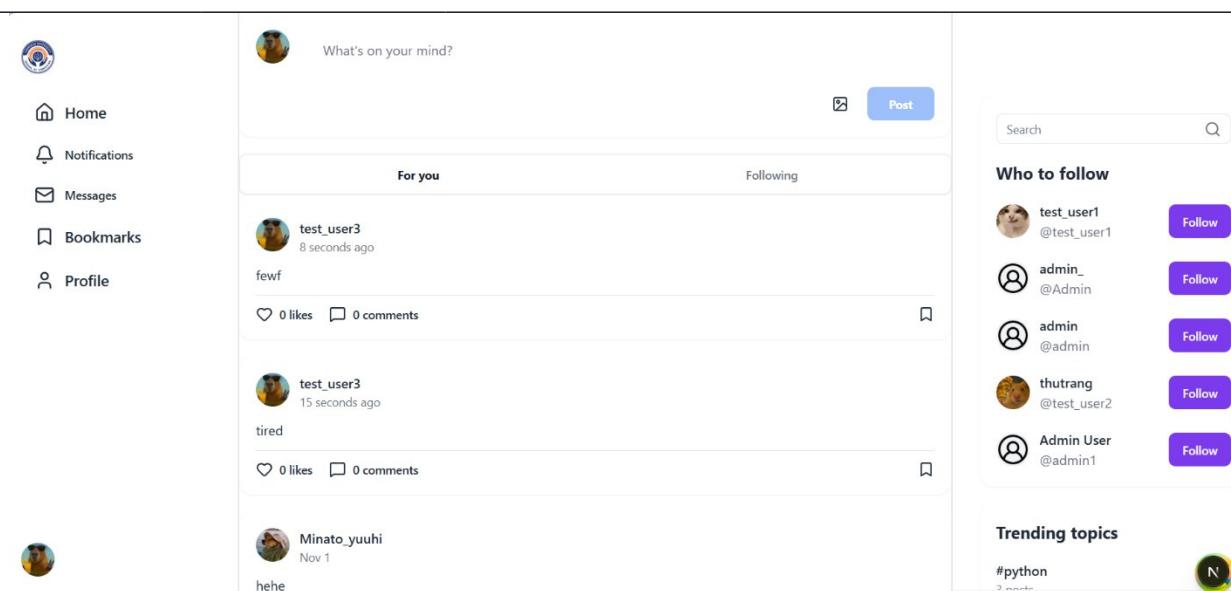
3. Server thực hiện thao tác revalidate cache cho trang feed (nếu dùng Next.js) để đảm bảo các client khác sẽ thấy bài viết mới.
4. Server trả về response success chứa dữ liệu bài viết (id, timestamp, attachments).
5. Client nhận response → cập nhật UI (thay tạm bài viết bằng bài viết có ID thực và timestamp chính xác); làm trống PostEditor.
6. Use case kết thúc.

+ Luồng lỗi: C3. Đăng bài thất bại

1. Server trả về lỗi (HTTP 4xx/5xx) hoặc timeout; nguyên nhân có thể là: lỗi xác thực, lỗi DB, lỗi upload, lỗi quota, v.v.
2. Client nhận lỗi → rollback Optimistic Update (loại bỏ bài viết tạm khỏi feed).
3. Hiển thị thông báo lỗi rõ ràng cho người dùng: ví dụ
4. “Đăng bài thất bại. Vui lòng thử lại.”
5. Nếu do upload: “Upload ảnh thất bại.”
6. Nếu do lượng nội dung vi phạm: “Nội dung chứa từ/đoạn bị cấm.”
7. Người dùng có thể sửa bài và thử đăng lại. Use case kết thúc.

+ Yêu cầu phi chức năng

- Hiệu năng: Đăng bài và phản hồi trong < 2 giây.
- Khả năng sử dụng: Hỗ trợ kéo-thả ảnh, gợi ý hashtag.
- Tính ổn định: Dữ liệu không mất khi mạng chập chờn.
- Bảo mật: Chỉ người dùng đăng nhập mới có quyền đăng.
- Responsive: Hiển thị tốt trên cả máy tính và điện thoại.



Hình 3.4. Giao diện bảng tin dành cho người dùng



Hình 3.5. Giao diện bảng tin những người đã follow

Bảng 3.3. Đặc tả chức năng Xem bảng tin

### 3.2.2. UC-2.2 – Đăng bài viết mới

UC-2.2		ĐĂNG BÀI VIẾT MỚI
Mô tả		Chức năng cho phép Người dùng đã đăng nhập tạo và chia sẻ một bài viết mới (bao gồm văn bản, mã nguồn, và hình ảnh) lên hệ thống.
Tác nhân		Người dùng (User)
Tiền điều kiện		Người dùng đã đăng nhập thành công.
Hậu điều kiện	Thành công	Bài viết mới được lưu vào CSDL và hiển thị trên bảng tin.
	Lỗi	Bài viết không được tạo, người dùng nhận thông báo lỗi.
ĐẶC TẢ CHỨC NĂNG		
+ Luồng sự kiện chính/Kịch bản chính		
1. Chức năng bắt đầu khi Người dùng ở trang chủ (/) và tương tác với PostEditor. 2. Người dùng nhập nội dung văn bản vào trình soạn thảo TipTap. 3. Nếu Người dùng muốn thêm ảnh, thực hiện Luồng C1: Tải ảnh lên. 4. Người dùng nhấn nút "Đăng" (Post). 5. Hệ thống (Client) kiểm tra nội dung (ví dụ: không được rỗng) bằng Zod. 6. Nếu hợp lệ, hệ thống thực hiện Optimistic Update (hiển thị ngay bài đăng mới trên feed) và đồng thời gửi yêu cầu (Server Action) lên máy chủ. 7. Kiểm tra bài viết không rỗng; 8. Kiểm tra kích thước/định dạng hình ảnh (nếu có); 9. Kiểm tra các ràng buộc nội dung khác (nếu có). 10. Hệ thống (Server Action) xác thực người dùng, lưu bài viết (nội dung text và URL)		

ảnh) vào CSDL (Postgres).

11. Nếu lưu thành công, thực hiện Luồng C2: Đăng bài thành công (Phía Server).

12. Nếu lưu thất bại, thực hiện Luồng C3: Đăng bài thất bại.

+ Luồng con: C1. Tải ảnh lên

1. Người dùng kéo-thả hoặc chọn file ảnh.

2. Hệ thống (Client) gọi dịch vụ UploadThing.

3. UploadThing xử lý, tải file lên và trả về URL của ảnh.

4. Client nhận URL và chèn ảnh vào bản preview của PostEditor (hiển thị thumbnail).

5. PostEditor hiển thị ảnh đã tải lên.

6. Người dùng có thể xóa/ đổi ảnh trước khi nhấn Post

7. Luồng quay lại bước 4 của Luồng chính.

+ Luồng con: C2. Đăng bài thành công (Phía Server)

1. Server lưu bản ghi bài viết (posts) vào bảng posts (fields: id, user\_id, content, created\_at, updated\_at, ...).

2. Nếu có attachments thì lưu vào bảng post\_attachments hoặc tương đương (url, type, post\_id).

3. Server thực hiện thao tác revalidate cache cho trang feed (nếu dùng Next.js) để đảm bảo các client khác sẽ thấy bài viết mới.

4. Server trả về response success chứa dữ liệu bài viết (id, timestamp, attachments).

5. Client nhận response → cập nhật UI (thay tạm bài viết bằng bài viết có ID thực và timestamp chính xác); làm trống PostEditor.

6. Use case kết thúc.

+ Luồng lỗi: C3. Đăng bài thất bại

1. Server trả về lỗi (HTTP 4xx/5xx) hoặc timeout; nguyên nhân có thể là: lỗi xác thực, lỗi DB, lỗi upload, lỗi quota, v.v.

2. Client nhận lỗi → rollback Optimistic Update (loại bỏ bài viết tạm khỏi feed).

3. Hiển thị thông báo lỗi rõ ràng cho người dùng: ví dụ

4. “Đăng bài thất bại. Vui lòng thử lại.”

5. Nếu do upload: “Upload ảnh thất bại.”

6. Nếu do lượng nội dung vi phạm: “Nội dung chứa từ/đoạn bị cấm.”

7. Người dùng có thể sửa bài và thử đăng lại. Use case kết thúc.

+ Yêu cầu phi chức năng

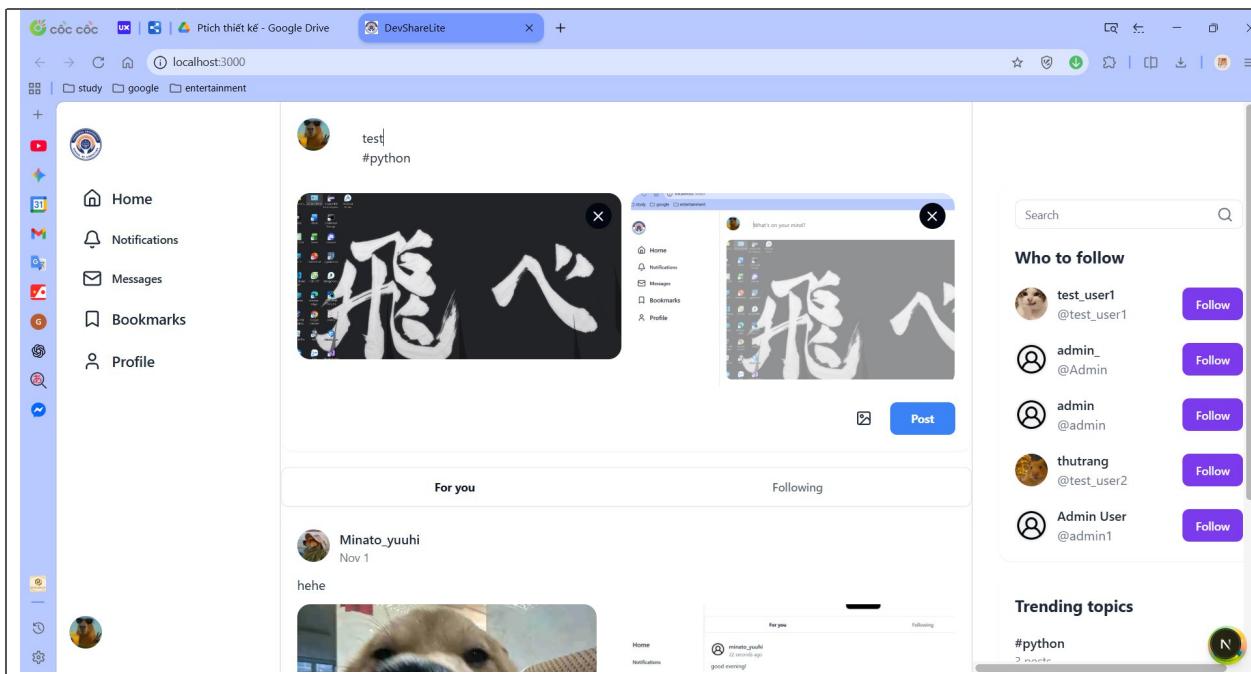
- Hiệu năng: Đăng bài và phản hồi trong < 2 giây.

- Khả năng sử dụng: Hỗ trợ kéo-thả ảnh, gợi ý hashtag.

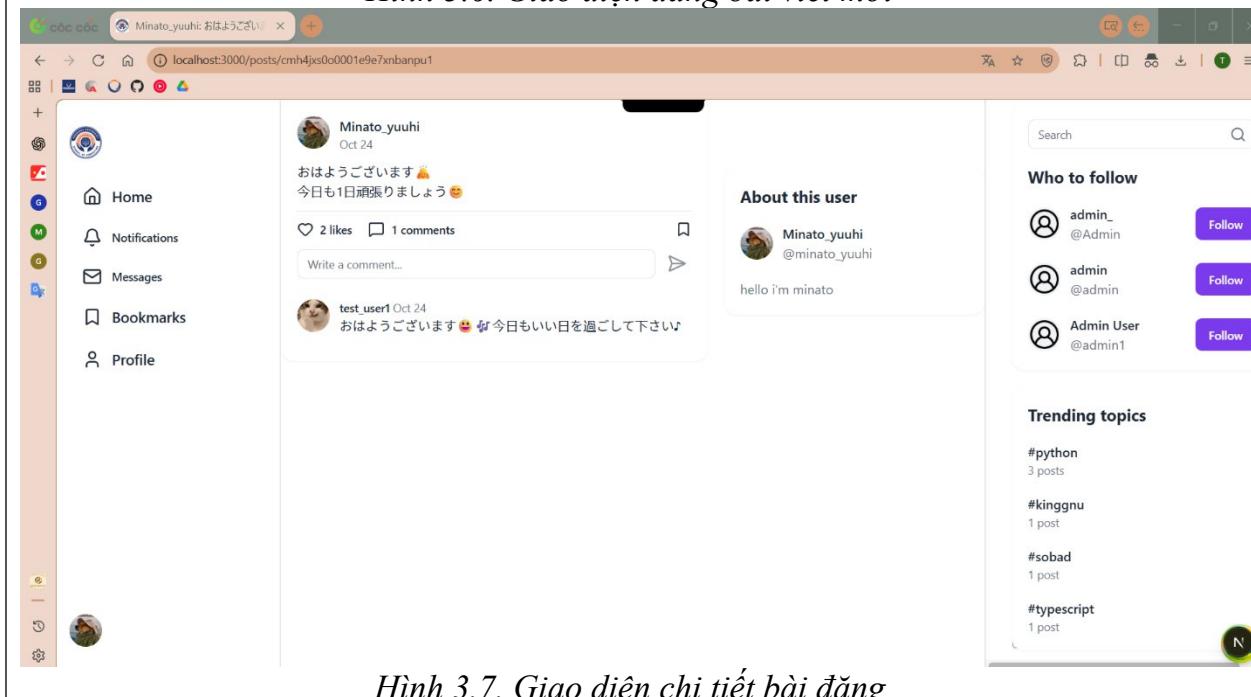
- Tính ổn định: Dữ liệu không mất khi mạng chập chờn.

- Bảo mật: Chỉ người dùng đăng nhập mới có quyền đăng.

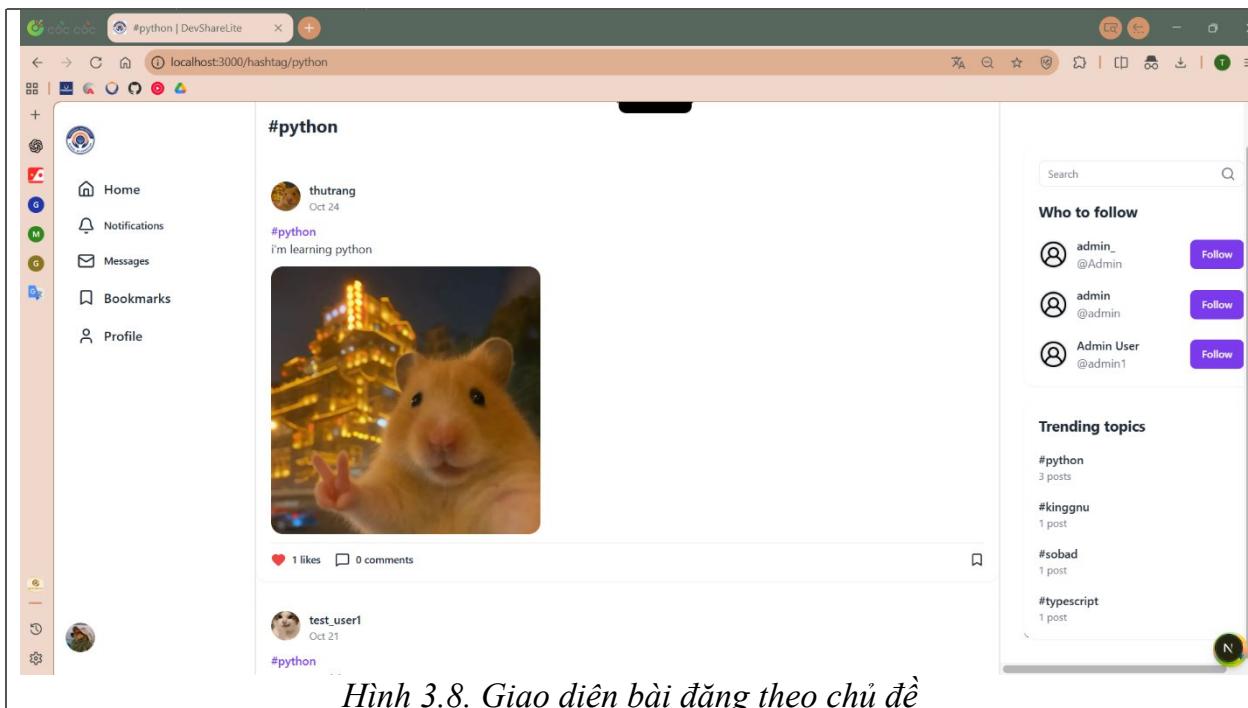
- Responsive: Hiển thị tốt trên cả máy tính và điện thoại.



Hình 3.6. Giao diện đăng bài viết mới



Hình 3.7. Giao diện chi tiết bài đăng



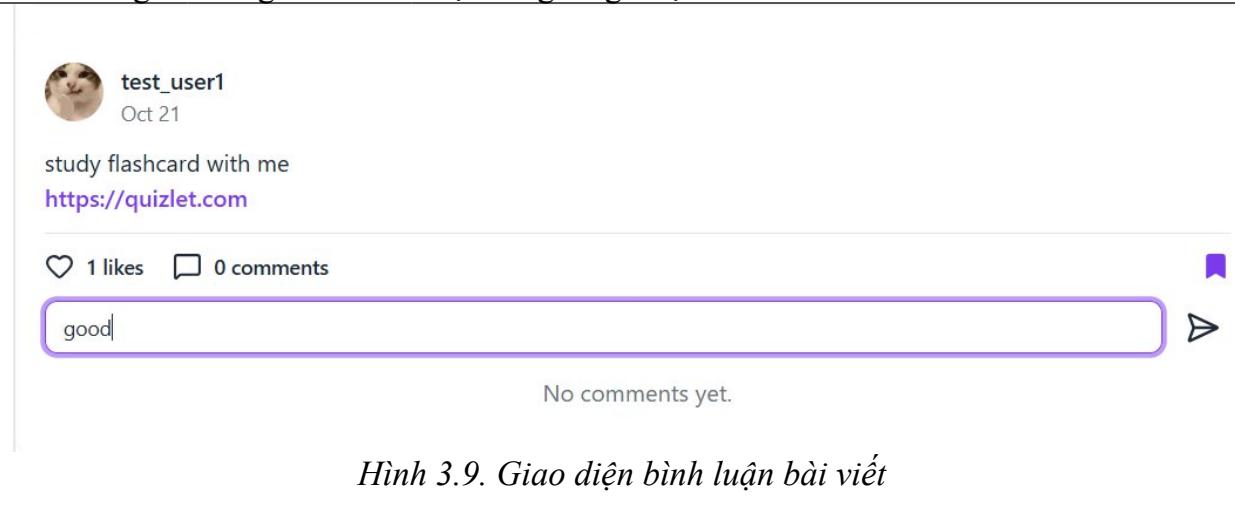
Hình 3.8. Giao diện bài đăng theo chủ đề

Bảng 3.4. Đặc tả chức năng Đăng bài viết mới

### 3.2.3. UC-2.3 – Bình luận bài viết

UC-2.3		BÌNH LUẬN BÀI VIẾT
Mô tả		Chức năng cho phép Người dùng gửi phản hồi, ý kiến hoặc thảo luận về một bài viết cụ thể.
Tác nhân		Người dùng (User)
Tiền điều kiện		Người dùng đã đăng nhập và đang xem một bài viết (/posts/[postId]).
Hậu điều kiện	Thành công	Bình luận mới được lưu vào CSDL và hiển thị dưới bài viết. Bộ cache/ISR được revalidate để dữ liệu đồng bộ. Input bình luận được làm trống.
	Lỗi	Bình luận không được lưu; nếu đã hiển thị tạm thời, hệ thống rollback và thông báo lỗi cho người dùng.
<b>ĐẶC TẢ CHỨC NĂNG</b>		
+ Luồng sự kiện chính/Kịch bản chính		
<ol style="list-style-type: none"> <li>Chức năng bắt đầu khi Người dùng xem chi tiết một bài viết.</li> <li>Hệ thống hiển thị CommentInput ở dưới bài viết.</li> <li>Người dùng nhập nội dung bình luận.</li> <li>Người dùng nhấn nút "Gửi".</li> <li>Hệ thống (Client) thực hiện Optimistic Update (hiển thị ngay bình luận mới).</li> <li>Hệ thống (Server Action) tiếp nhận nội dung, UserID và PostID.</li> <li>Hệ thống lưu bình luận mới vào CSDL (Postgres).</li> <li>Nếu lưu thành công, thực hiện Luồng D1: Gửi thành công (Phía Server).</li> </ol>		

9. Nếu thất bại, thực hiện Luồng D2: Gửi thất bại.
- + Luồng con: D1. Gửi thành công (Phía Server)
1. Server lưu bình luận vào bảng comments với các thông tin: id, user\_id, post\_id, content, created\_at.
  2. Server cập nhật bộ đếm bình luận của bài viết (nếu có).
  3. Server thực hiện revalidate cache hoặc ISR để đồng bộ dữ liệu.
  4. Server trả về response thành công, Client nhận dữ liệu và hiển thị bình luận chính thức.
  5. Input bình luận được xóa nội dung, use case kết thúc.
- + Luồng lỗi: D2. Gửi thất bại
1. Server trả về lỗi (HTTP 4xx/5xx) hoặc timeout.
  2. Client rollback Optimistic Update (xóa bình luận tạm).
  3. Hệ thống hiển thị thông báo lỗi:
  4. “Không thể gửi bình luận. Vui lòng thử lại.”
  5. “Phiên đăng nhập đã hết hạn.”
  6. “Nội dung bình luận không hợp lệ.”
  7. Người dùng có thể sửa nội dung và gửi lại. Use case kết thúc.



Hình 3.9. Giao diện bình luận bài viết

Bảng 3.5. Đặc tả chức năng Bình luận bài viết

### 3.2.4. UC-2.4 - Thích bài viết

UC-2.4		THÍCH BÀI VIẾT
Mô tả		Chức năng cho phép Người dùng thích hoặc bỏ thích một bài viết để thể hiện sự yêu thích và lưu vào danh sách ưa thích.
Tác nhân		Người dùng (User)
Tiền điều kiện		Người dùng đã đăng nhập
Hậu điều kiện	Thành công	Trạng thái Like được cập nhật trong CSDL và UI.
	Lỗi	Trạng thái không thay đổi.

## ĐẶC TẢ CHỨC NĂNG

### + Luồng sự kiện chính/Kịch bản chính

1. Chức năng bắt đầu khi Người dùng xem một bài viết trên bảng tin hoặc trang chi tiết.
2. Hệ thống hiển thị LikeButton với trạng thái hiện tại (đã thích/chưa thích) và số lượt thích.
3. Người dùng nhấn vào LikeButton.
4. Nếu chưa thích, thực hiện Luồng K1: Thích bài viết.
5. Nếu đã thích, thực hiện Luồng K2: Bỏ thích bài viết.

### + Luồng con: K1. Thích bài viết

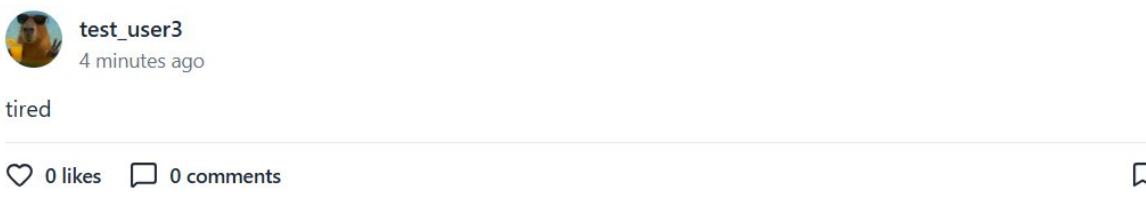
1. Hệ thống (Client) thực hiện Optimistic Update (tăng số lượt thích, đổi màu icon).
2. Hệ thống (Server Action) tạo bản ghi Like mới trong CSDL.
3. Hệ thống tạo thông báo cho tác giả bài viết.
4. Hệ thống revalidate cache của bài viết.
5. Use case kết thúc.

### + Luồng con: K2. Bỏ thích bài viết

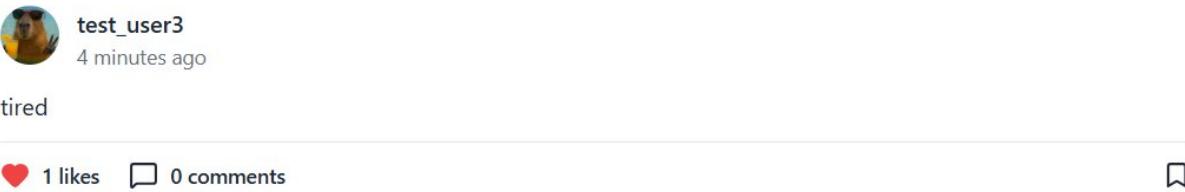
1. Hệ thống (Client) thực hiện Optimistic Update (giảm số lượt thích, đổi màu icon).
2. Hệ thống (Server Action) xóa bản ghi Like trong CSDL.
3. Hệ thống revalidate cache của bài viết.
4. Use case kết thúc.

### + Luồng lỗi: K3. Thao tác thất bại

1. Hệ thống (Client) rollback Optimistic Update.
2. Hệ thống hiển thị thông báo lỗi.
3. Use case kết thúc.



Hình 3.10. Giao diện trước khi thích bài viết



Hình 3.11. Giao diện sau khi thích bài viết

Bảng 3.6. Đặc tả chức năng Thích bài viết

### 3.2.5. UC-2.5 - Lưu bài viết

UC-2.5	LUU BÀI VIẾT (BOOKMARK)			
Mô tả	Chức năng cho phép Người dùng lưu bài viết vào danh sách đánh dấu cá nhân để dễ dàng truy cập lại sau này.			
Tác nhân	Người dùng (User)			
Tiền điều kiện	Người dùng đã đăng nhập			
Hậu điều kiện	Thành công	Bài viết được thêm/xóa khỏi danh sách Bookmark.		
	Lỗi	Trạng thái không thay đổi.		
<b>ĐẶC TẢ CHỨC NĂNG</b>				
+ Luồng sự kiện chính/Kịch bản chính	<ol style="list-style-type: none"> <li>Chức năng bắt đầu khi Người dùng xem một bài viết.</li> <li>Hệ thống hiển thị BookmarkButton với trạng thái hiện tại.</li> <li>Người dùng nhấn vào BookmarkButton.</li> <li>Nếu chưa lưu, thực hiện Luồng L1: Lưu bài viết.</li> <li>Nếu đã lưu, thực hiện Luồng L2: Bỏ lưu bài viết.</li> </ol>			
+ Luồng con: L1. Lưu bài viết	<ol style="list-style-type: none"> <li>Hệ thống (Client) thực hiện Optimistic Update (đổi icon sang trạng thái đã lưu).</li> <li>Hệ thống (Server Action) tạo bản ghi Bookmark trong CSDL.</li> <li>Hệ thống revalidate cache của trang /bookmarks.</li> <li>Use case kết thúc.</li> </ol>			
+ Luồng con: L2. Bỏ lưu bài viết	<ol style="list-style-type: none"> <li>Hệ thống (Client) thực hiện Optimistic Update (đổi icon sang trạng thái chưa lưu).</li> <li>Hệ thống (Server Action) xóa bản ghi Bookmark trong CSDL.</li> <li>Hệ thống revalidate cache của trang /bookmarks.</li> <li>Use case kết thúc.</li> </ol>			
+ Luồng lỗi: L3. Thao tác thất bại	<ol style="list-style-type: none"> <li>Hệ thống (Client) rollback Optimistic Update.</li> <li>Hệ thống hiển thị thông báo lỗi.</li> <li>Use case kết thúc.</li> </ol>			
 <b>test_user3</b> 6 minutes ago  tired  <span style="color: #f08080;">1 likes</span> <span style="color: #f08080;">0 comments</span> <span style="float: right;"></span>				
<i>Hình 3.12. Giao diện sau khi lưu bài viết</i>				

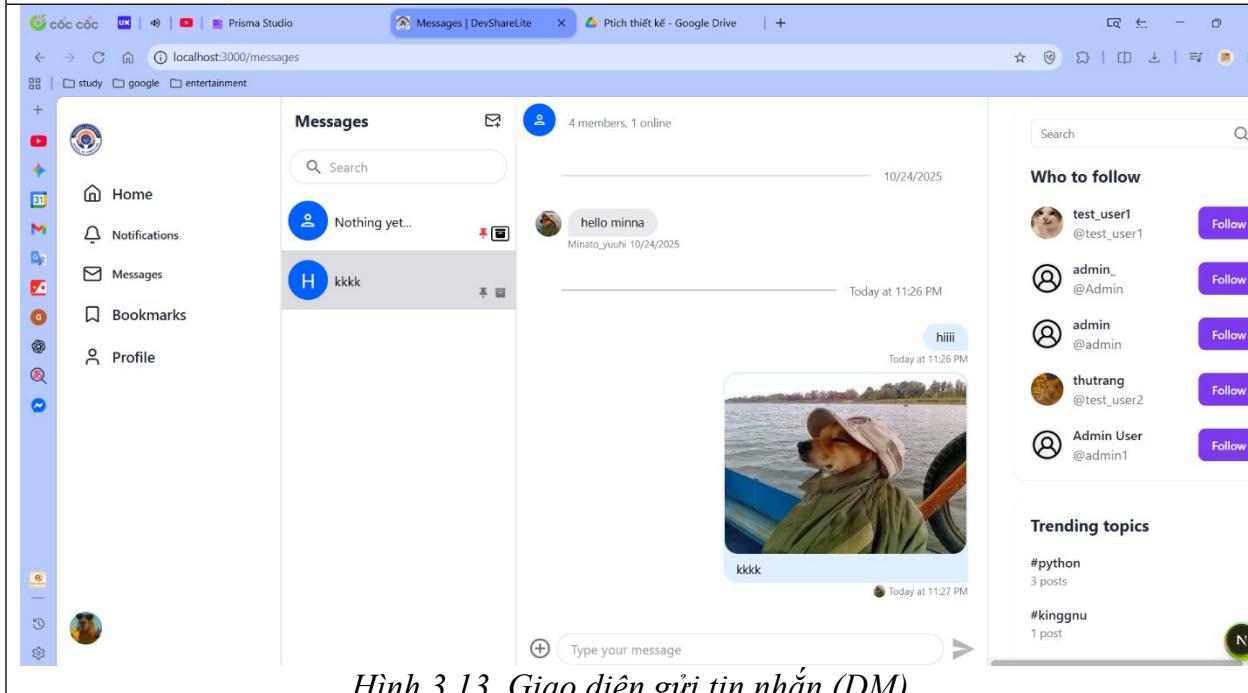
Bảng 3.7. Đặc tả chức năng Lưu bài viết

### 3.2.6. UC-2.6 – Gửi tin nhắn (DM)

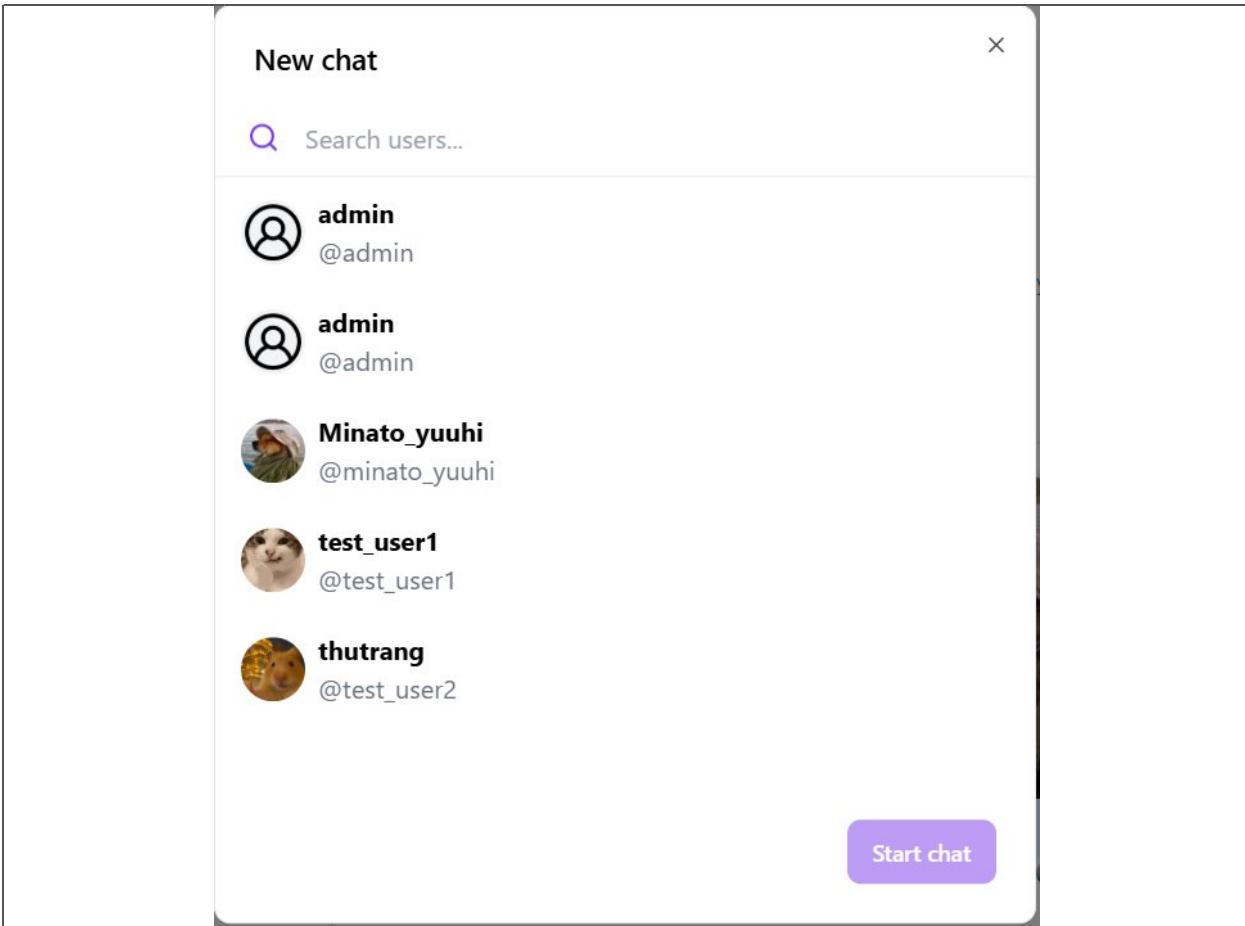
UC-2.6		GỬI TIN NHẮN (DM)
Mô tả		Chức năng cho phép người dùng gửi tin nhắn riêng tư đến một người dùng khác trong hệ thống. Tin nhắn có thể là văn bản hoặc hình ảnh và được truyền theo thời gian thực (real-time) thông qua Stream API. Người dùng có thể bắt đầu cuộc trò chuyện mới hoặc tiếp tục một cuộc hội thoại đã có.
Tác nhân		Người dùng (User)
Tiền điều kiện		Người dùng đã đăng nhập thành công vào hệ thống. Hệ thống đã kết nối ổn định với máy chủ Stream API. Người nhận tin nhắn tồn tại và chưa bị khóa tài khoản.
Hậu điều kiện	Thành công	Tin nhắn được gửi và hiển thị ngay lập tức trong khung chat của cả hai người dùng. Cuộc hội thoại được lưu vào danh sách “Messages”.
	Lỗi	Tin nhắn không được gửi, hệ thống hiển thị thông báo lỗi tương ứng.
<b>ĐẶC TẢ CHỨC NĂNG</b>		
+ Luồng sự kiện chính/Kịch bản chính <ol style="list-style-type: none"> <li>Người dùng truy cập trang /messages hoặc chọn “Nhắn tin” từ hồ sơ của người khác.</li> <li>Hệ thống khởi tạo kết nối với Stream API để thiết lập liên lạc real-time.</li> <li>Hệ thống hiển thị danh sách các cuộc trò chuyện đã có (nếu có).</li> <li>Người dùng chọn một cuộc trò chuyện hoặc nhấn “New Chat” để bắt đầu cuộc trò chuyện mới.</li> <li>Người dùng tìm kiếm và chọn người muốn nhắn tin.</li> <li>Hệ thống hiển thị cửa sổ chat (Chat.tsx) giữa hai người dùng.</li> <li>Người dùng nhập nội dung tin nhắn vào ô nhập văn bản.</li> <li>Người dùng nhấn nút “Gửi” hoặc phím Enter.</li> <li>Hệ thống gửi dữ liệu tin nhắn đến Stream API.</li> <li>Stream API xử lý và gửi tin nhắn đến người nhận theo thời gian thực.</li> <li>Tin nhắn hiển thị ngay lập tức trong giao diện chat của cả hai người.</li> <li>Cuộc hội thoại được cập nhật trong danh sách “Messages”.</li> <li>Use Case kết thúc.</li> </ol>		
+ Luồng phụ (Alternative Flow): <ol style="list-style-type: none"> <li>Người dùng có thể gửi tin nhắn hình ảnh bằng cách chọn biểu tượng hình ảnh và tải lên file.</li> <li>Người dùng có thể gửi emoji hoặc reaction thay vì văn bản.</li> <li>Nếu người dùng đang chat với nhiều người, họ có thể chuyển đổi giữa các cuộc trò chuyện mà không cần tải lại trang.</li> </ol>		

+ Luồng lỗi: Luồng lỗi (Exception Flow):

1. F1: Kết nối đến Stream API thất bại → Hiển thị thông báo “Không thể gửi tin nhắn. Vui lòng kiểm tra kết nối.”
2. F2: Tin nhắn trống → Hệ thống hiển thị cảnh báo “Tin nhắn không được để trống.”
3. F3: Người nhận không tồn tại hoặc bị khóa tài khoản → Hiển thị “Không thể gửi tin nhắn. Người dùng không khả dụng.”
4. F4: Ảnh tải lên vượt giới hạn dung lượng → Hiển thị “Tệp quá lớn, vui lòng chọn ảnh nhỏ hơn 5MB.”



Hình 3.13. Giao diện gửi tin nhắn (DM)



Hình 3.14. Giao diện tạo cuộc hội thoại

cốc cốc | localhost:3000/messages

Messages

3 members, 1 online

Search

No chats here yet...

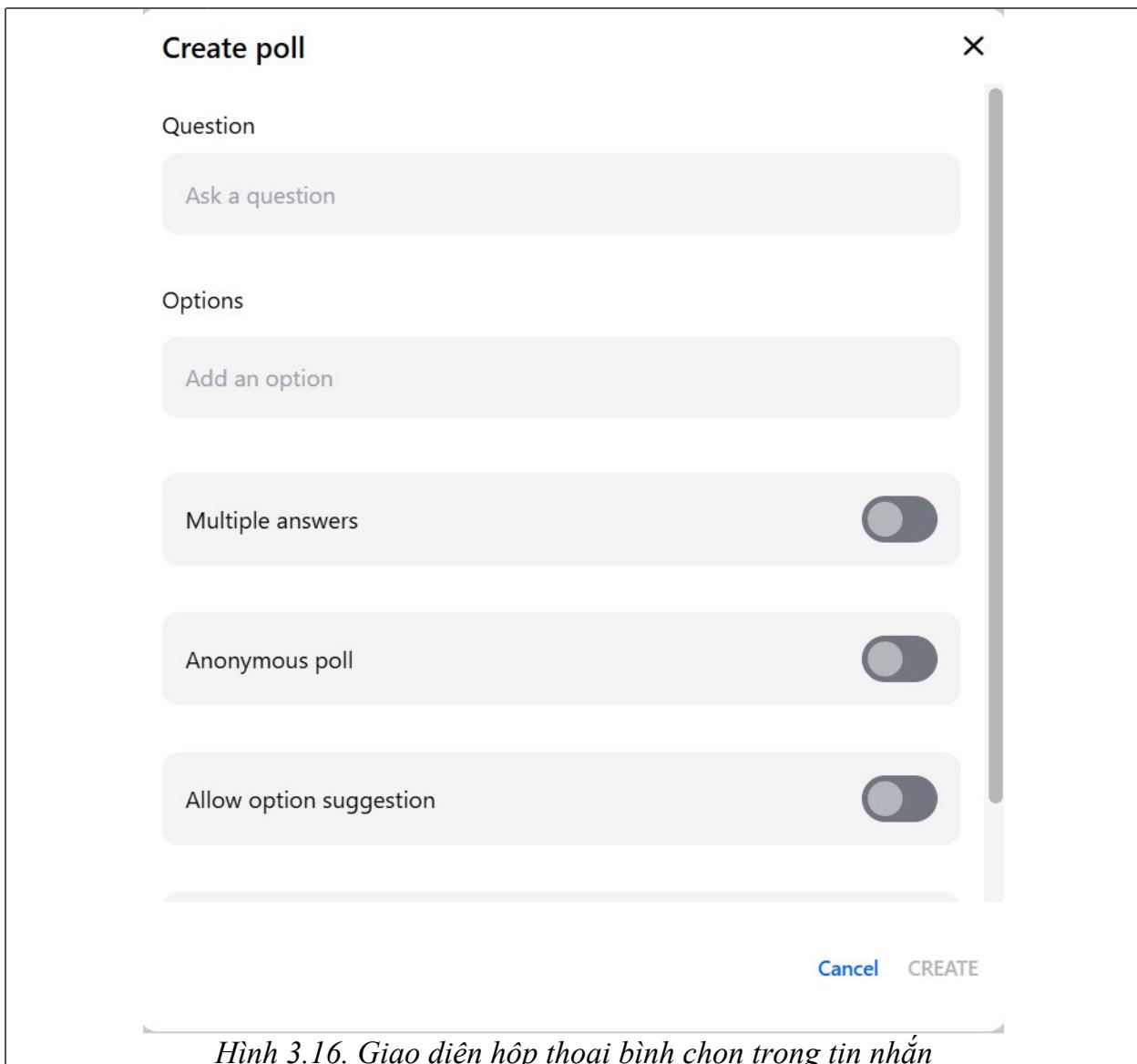
Who to follow

- test\_user1 @test\_user1 Follow
- admin\_ @Admin Follow
- admin @admin Follow
- thutrang @test\_user2 Follow
- Admin User @admin1 Follow

Trending topics

#python

Hình 3.15. Giao diện tìm kiếm người dùng nhắn tin



Hình 3.16. Giao diện hộp thoại bình chọn trong tin nhắn

Bảng 3.8. Đặc tả chức năng Gửi tin nhắn

### 3.2.7. UC-2.7 - Báo cáo vi phạm

UC-2.7	BÁO CÁO VI PHẠM	
Mô tả	Chức năng cho phép Người dùng báo cáo các bài viết hoặc bình luận vi phạm quy định cộng đồng để Quản trị viên xem xét và xử lý.	
Tác nhân	Người dùng (User)	
Tiền điều kiện	Người dùng đã đăng nhập	
Hậu điều kiện	Thành công	Báo cáo được gửi đến hệ thống và lưu vào CSDL để Admin xem xét.
	Lỗi	Báo cáo không được gửi, hiển thị thông báo lỗi.
<b>ĐẶC TẢ CHỨC NĂNG</b>		

+ Luồng sự kiện chính/Kịch bản chính

1. Chức năng bắt đầu khi Người dùng xem một bài viết hoặc bình luận trên hệ thống.
2. Người dùng nhấn vào menu (dấu ba chấm) của bài viết/bình luận.
3. Hệ thống hiển thị menu tùy chọn với mục "Báo cáo".
4. Người dùng chọn "Báo cáo".
5. Nếu báo cáo bài viết, thực hiện Luồng Q1: Báo cáo bài viết.
6. Nếu báo cáo bình luận, thực hiện Luồng Q2: Báo cáo bình luận.

+ Luồng con: Q1. Báo cáo bài viết

- Hệ thống hiển thị Dialog "Báo cáo bài viết" với các lý do:
  - Spam hoặc quảng cáo
  - Nội dung bạo lực hoặc đồi trụy
  - Thông tin sai lệch
  - Ngôn từ gây thù ghét
  - Quấy rối hoặc bắt nạt
  - Khác (cho phép nhập lý do)
- Người dùng chọn một hoặc nhiều lý do.
- Người dùng có thể thêm mô tả chi tiết (tùy chọn).
- Người dùng nhấn nút "Gửi báo cáo".
- Hệ thống (Client) xác thực dữ liệu bằng Zod.
- Hệ thống (Server Action) lưu báo cáo vào CSDL (bảng Report) với thông tin:
  - reporterId (ID người báo cáo)
  - postId (ID bài viết bị báo cáo)
  - reason (Lý do)
  - description (Mô tả chi tiết)
  - status (Trạng thái: PENDING)
  - createdAt (Thời gian báo cáo)

7. Thực hiện Luồng Q3: Gửi báo cáo thành công.

+ Luồng con: Q2. Báo cáo bình luận

8. Hệ thống hiển thị Dialog "Báo cáo bình luận" với các lý do tương tự Q1.
9. Người dùng chọn lý do và có thể thêm mô tả.
10. Người dùng nhấn nút "Gửi báo cáo".
11. Hệ thống (Client) xác thực dữ liệu bằng Zod.
12. Hệ thống (Server Action) lưu báo cáo vào CSDL (bảng Report) với:
  - reporterId
  - commentId (ID bình luận bị báo cáo)
  - reason
  - description
  - status (PENDING)
  - createdAt

13. Thực hiện Luồng Q3: Gửi báo cáo thành công.

+ Luồng con: Q3. Gửi báo cáo thành công

1. Hệ thống lưu báo cáo vào CSDL.
2. Hệ thống tạo thông báo cho Admin (nếu có hệ thống thông báo admin).

- 3. Hệ thống đóng Dialog và hiển thị thông báo "Cảm ơn bạn đã báo cáo."
  - 4. Chúng tôi sẽ xem xét trong thời gian sớm nhất."
  - 5. Use case kết thúc.
- + Luồng lỗi: Q4. Gửi báo cáo thất bại
1. Nếu có lỗi khi lưu vào CSDL hoặc kết nối thất bại.
  2. Hệ thống hiển thị thông báo lỗi "Không thể gửi báo cáo. Vui lòng thử lại sau."
  3. Người dùng có thể thử gửi lại hoặc đóng Dialog.
  4. Use case kết thúc.
- + Luồng lỗi: Q5. Người dùng đã báo cáo trước đó
1. Hệ thống kiểm tra xem người dùng đã báo cáo nội dung này chưa.
  2. Nếu đã báo cáo, hiển thị thông báo "Bạn đã báo cáo nội dung này trước đó."
  3. Chúng tôi đang xem xét."
  4. Use case kết thúc.

Bảng 3.9. *Đặc tả chức năng Báo cáo vi phạm*

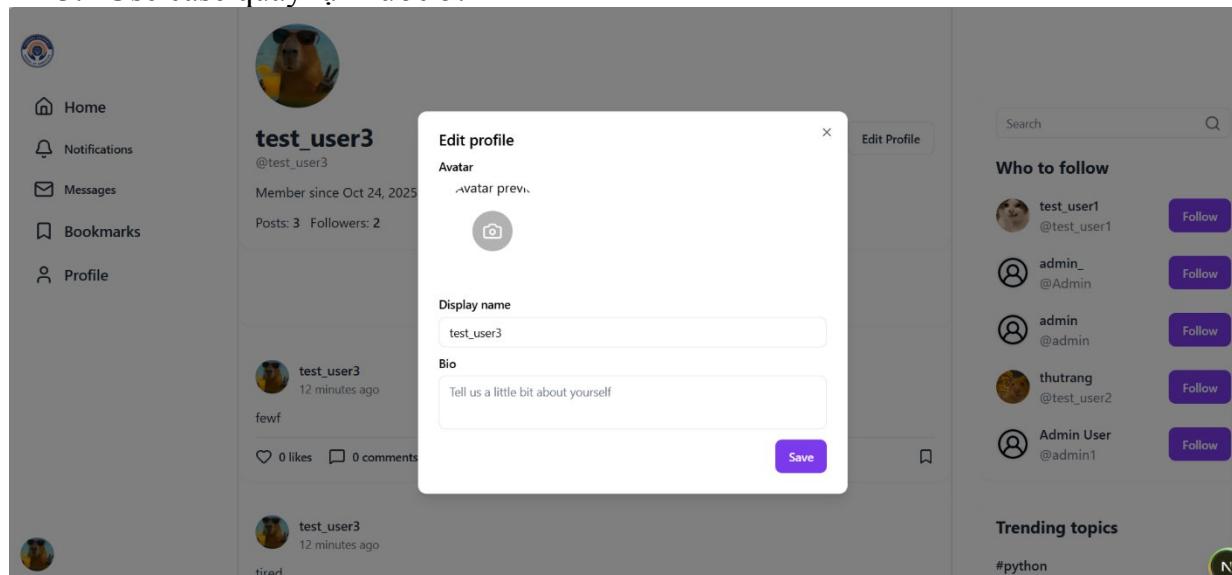
### 3.2.8. UC-2.8 - Quản lý hồ sơ cá nhân

UC-2.8		QUẢN LÍ HỒ SƠ CÁ NHÂN
Mô tả		Chức năng cho phép Người dùng chỉnh sửa thông tin hồ sơ cá nhân như ảnh đại diện, ảnh bìa, bio, và thông tin liên hệ.
Tác nhân		Người dùng (User)
Tiền điều kiện		Người dùng đã đăng nhập thành công
Hậu điều kiện	Thành công	Thông tin hồ sơ được cập nhật trong CSDL và hiển thị ngay.
	Lỗi	Thông tin không được cập nhật, hiển thị thông báo lỗi.

#### ĐẶC TẢ CHỨC NĂNG

- + Luồng sự kiện chính/Kịch bản chính
1. Chức năng bắt đầu khi Người dùng truy cập trang hồ sơ của chính mình
  2. (/users/[username]).
  3. Hệ thống hiển thị nút "Chỉnh sửa hồ sơ" (EditProfileButton).
  4. Người dùng nhấn nút "Chỉnh sửa hồ sơ".
  5. Hệ thống hiển thị Dialog chỉnh sửa (EditProfileDialog) với các trường:
    - Tên hiển thị (Display Name)
    - Bio (giới thiệu bản thân)
    - Ảnh đại diện (Avatar)
    - Ảnh bìa (Cover Image)
  6. Người dùng chỉnh sửa thông tin mong muốn.
  7. Nếu Người dùng muốn thay đổi ảnh, thực hiện Luồng I1: Cắt và tải ảnh lên.
  8. Người dùng nhấn nút "Lưu".
  9. Hệ thống (Server Action) xác thực dữ liệu bằng Zod.

10. Nếu hợp lệ, thực hiện Luồng I2: Cập nhật thành công.  
 11. Nếu không hợp lệ, thực hiện Luồng I3: Dữ liệu không hợp lệ.
- + Luồng con: I1. Cắt và tải ảnh lên
- Người dùng chọn file ảnh từ thiết bị.
  - Hệ thống hiển thị CropImageDialog cho phép người dùng cắt/chỉnh ảnh.
  - Người dùng điều chỉnh và xác nhận.
  - Hệ thống gọi UploadThing để tải ảnh đã cắt lên server.
  - UploadThing trả về URL của ảnh.
  - Luồng quay lại bước 7 của Luồng chính.
- + Luồng con: I2. Cập nhật thành công
- Hệ thống lưu thông tin mới vào CSDL (Postgres).
  - Hệ thống revalidate cache của trang hồ sơ.
  - Hệ thống đóng Dialog và hiển thị thông tin đã cập nhật.
  - Use case kết thúc.
- + Luồng lỗi: I3. Dữ liệu không hợp lệ
- Hệ thống hiển thị thông báo lỗi (ví dụ: "Bio quá dài",  
 2. "Định dạng ảnh không hỗ trợ").
  - Use case quay lại Bước 5.



Hình 3.17. Giao diện sửa hồ sơ bản thân

Bảng 3.10. Quản lý hồ sơ cá nhân

### 3.2.9. UC-2.9 - Quản lý bài viết cá nhân

UC-2.9	QUẢN LÝ BÀI VIẾT CÁ NHÂN
Mô tả	Chức năng cho phép Người dùng đã đăng nhập thực hiện chỉnh sửa hoặc xóa các bài viết do chính họ tạo ra.
Tác nhân	Người dùng (User)
Tiền điều kiện	Người dùng đã đăng nhập thành công. Người dùng phải

		là chính chủ của bài viết đang thao tác.
Hậu kiện	điều Lỗi	Bài viết được cập nhật nội dung hoặc bị xóa khỏi CSDL. Bài viết không được thay đổi hoặc xóa, người dùng nhận thông báo lỗi (ví dụ: không có quyền).
<b>ĐẶC TẢ CHỨC NĂNG</b>		
+ Luồng sự kiện chính/Kịch bản chính		
<ol style="list-style-type: none"> <li>1. Chức năng bắt đầu khi Người dùng đang xem bài viết của chính mình.</li> <li>2. Hệ thống hiển thị bài viết</li> <li>3. Người dùng nhấn vào menu tùy chọn (ví dụ: dấu ba chấm) trên bài viết.</li> <li>4. Người dùng chọn chức năng "Xóa bài viết"</li> <li>5. Hệ thống hiển thị hộp thoại xác nhận (DeletePostDialog) cảnh báo đây là hành động không thể hoàn tác.</li> <li>6. Người dùng nhấn "Xác nhận xóa".</li> <li>7. Hệ thống (Client) gọi yêu cầu Server Action (deletePost(postId)) lên máy chủ.</li> <li>8. Hệ thống (Server Action) Xác minh quyền (đảm bảo UserID khớp với AuthorID của bài viết).</li> <li>9. Nếu xác minh quyền thành công, thực hiện Luồng C1: Xóa Bài viết thành công.</li> <li>10. Nếu xác minh quyền thất bại, thực hiện Luồng C3: Thao tác thất bại.</li> </ol>		
+ Luồng con: C1. Xóa Bài viết thành công		
<ol style="list-style-type: none"> <li>1. Hệ thống (Server) thực hiện lệnh DELETE lên bảng posts và các bảng liên quan (như comments, likes, bookmarks).</li> <li>2. Hệ thống kiểm tra và gọi dịch vụ UploadThing để xóa tệp vật lý (nếu có)</li> <li>3. Hệ thống (Next.js) revalidate cache cho trang feed và trang chi tiết bài viết.</li> <li>4. Hệ thống trả về trạng thái xóa thành công.</li> <li>5. Client nhận phản hồi à Cập nhật giao diện (xóa bài viết khỏi màn hình).</li> <li>6. Use case kết thúc</li> </ol>		
+ Luồng lỗi: C3. Thao tác thất bại		
<ol style="list-style-type: none"> <li>1. Server trả về lỗi (ví dụ: "Không có quyền thực hiện thao tác này" nếu UserID không khớp).</li> <li>2. Client hiển thị thông báo lỗi rõ ràng cho Người dùng.</li> <li>3. Người dùng có thể thử lại hoặc hủy bỏ. Use case kết thúc.</li> </ol>		
+ Yêu cầu phi chức năng		
<ul style="list-style-type: none"> <li>• Bảo mật: Chỉ người tạo bài viết hoặc Quản trị viên mới có quyền xóa bài viết</li> <li>• Khả năng sử dụng: Thao tác xóa phải có hộp thoại xác nhận cảnh báo.</li> <li>• Toàn vẹn: Thao tác xóa phải xóa tất cả các dữ liệu liên quan (lượt thích, bình luận, tệp đính kèm).</li> </ul>		



Hình 3.18. Giao diện xóa bài viết của người dùng

Bảng 3.11. Đặc tả Use case Quản lý bài viết cá nhân

### 3.2.10. UC- 2.10 - Xem thông báo

UC-2.10		XEM THÔNG BÁO
Mô tả		Hệ thống tự động ghi nhận và tạo một bản ghi thông báo (Notification) cho người nhận khi một người dùng khác thực hiện tương tác (Thích, Bình luận, Theo dõi) với nội dung/tài khoản của họ.
Tác nhân		Hệ thống (System-triggered)
Tiền điều kiện		Hành động tương tác (Like, Comment, Follow) đã được thực hiện thành công và lưu vào CSDL.
Hậu diều kiện	Thành công	Bản ghi thông báo mới được lưu vào bảng notifications.
	Lỗi	Thông báo không được tạo, nhưng hành động tương tác chính vẫn được giữ nguyên.

#### ĐẶC TẢ CHỨC NĂNG

+ Luồng sự kiện chính/Kịch bản chính

- Khởi đầu khi một hành động tương tác được thực hiện thành công (ví dụ: Người dùng A thích bài viết của Người dùng B, và bản ghi được lưu vào bảng likes). Hệ thống hiển thị bài viết
- Hệ thống (Server/API) xác định: Loại tương tác (LIKE, COMMENT, hoặc FOLLOW), Người thực hiện (issuerId), Người nhận (recipientId), và Bài viết liên quan (postId, nếu có).
- Hệ thống thực hiện lệnh INSERT vào bảng notifications với các dữ liệu đã xác định.
- Hệ thống đặt trạng thái read của thông báo là FALSE (chưa đọc).
- Nếu lưu thành công, thực hiện Luồng C1: Cập nhật giao diện Client.
- Nếu lưu thất bại, thực hiện Luồng C2: Tạo thông báo thất bại.

+ Luồng con: C1. Cập nhật giao diện Client

- Server trả về response thành công.
- Client (Browser) sử dụng Polling hoặc TanStack Query để làm mới số lượng thông báo chưa đọc.
- Giao diện hiển thị số lượng thông báo chưa đọc mới nhất (ví dụ: tăng từ 3 lên 4).
- Use Case kết thúc.

- + Luồng thay thế: A1. Xem và Đánh dấu đã đọc
  1. Người dùng B nhấp vào biểu tượng chuông thông báo.
  2. Client gọi API GET /api/notifications để lấy danh sách thông báo.
  3. Hệ thống hiển thị danh sách (phân biệt giữa đã đọc và chưa đọc).
  4. Người dùng chọn "Đánh dấu tất cả là đã đọc".
  5. Client gọi API PATCH /api/notifications/mark-as-read.
  6. Hệ thống (Server) thực hiện lệnh UPDATE trên bảng notifications, đặt read = TRUE cho tất cả thông báo của User B.
  7. Giao diện cập nhật, số lượng thông báo chưa đọc trở về 0.
- + Luồng lỗi: C2. Tạo thông báo thất bại
  1. Lệnh INSERT vào bảng notifications thất bại do lỗi DB, lỗi ràng buộc, v.v.
  2. Hệ thống ghi lại lỗi vào log (Log Error), nhưng không ảnh hưởng đến hành động tương tác chính đã thành công (bản ghi Like/Comment/Follow đã được lưu).
  3. Use Case kết thúc mà không có thông báo được gửi.
- + Yêu cầu phi chức năng
  - Hiệu năng: Việc tạo thông báo phải có độ trễ thấp (< 500ms) để không làm chậm hành động tương tác chính.
  - Khả năng mở rộng: Hệ thống phải có khả năng mở rộng để xử lý hàng triệu thông báo.
  - Tương lai: Cần nâng cấp lên Real-time Notifications (sử dụng WebSocket) thay vì chỉ sử dụng Polling/Revalidation.

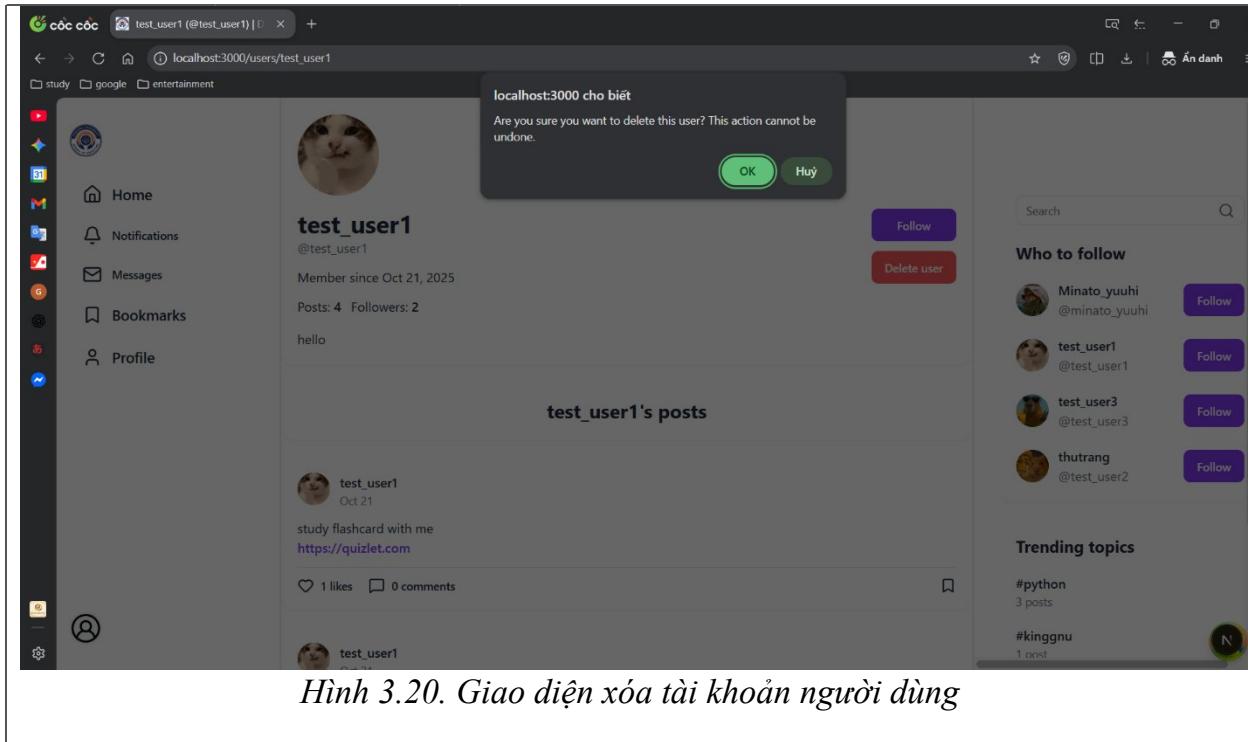
Hình 3.19. Giao diện thông báo

Bảng 3.12. Đặc tả usecase Xem thông báo

### 3.3. Nhóm chức năng Quản trị Hệ thống (Tác nhân: Quản trị viên)

#### 3.3.1. UC-3.1 – Quản lý tài khoản người dùng

UC-3.1		QUẢN LÝ TÀI KHOẢN NGƯỜI DÙNG
Mô tả		Chức năng cho phép Quản trị viên (Admin) thực hiện các hành động quản lý như Khóa, Mở khóa, hoặc Xóa vĩnh viễn tài khoản của một Người dùng khác.
Tác nhân		Quản trị viên (Admin)
Tiền điều kiện		Quản trị viên đã đăng nhập thành công.
Hậu điều kiện	Thành công	Trạng thái hoặc sự tồn tại của tài khoản Người dùng bị thay đổi trong CSDL.
	Lỗi	Hành động không được thực hiện.
<b>ĐẶC TẢ CHỨC NĂNG</b>		
+ Luồng sự kiện chính/Kịch bản chính <ol style="list-style-type: none"> <li>Chức năng bắt đầu khi Quản trị viên truy cập vào trang hồ sơ (/users/[username]) của một Người dùng bất kỳ.</li> <li>Hệ thống (Client) kiểm tra (ví dụ: qua useSession) và xác nhận người xem là Admin.</li> <li>Hệ thống hiển thị các nút điều khiển đặc biệt (AdminUserControls.tsx) (ví dụ: ", "Xóa tài khoản").</li> <li>Nếu Admin muốn xóa tài khoản, thực hiện Luồng «extend»: G. Xóa tài khoản người dùng.</li> </ol>		
+ Luồng «extend»: G1. Xóa tài khoản người dùng <ol style="list-style-type: none"> <li>Admin nhấn nút "Xóa tài khoản".</li> <li>Hệ thống hiển thị Dialog xác nhận (cảnh báo hành động nguy hiểm).</li> <li>Admin xác nhận.</li> <li>Hệ thống (API Route) gọi đến api/admin/delete-user/route.ts.</li> <li>Hệ thống (Server) xóa vĩnh viễn User và các nội dung liên quan (hoặc thực hiện "soft delete") khỏi CSDL.</li> <li>Hệ thống thông báo hành động thành công.</li> <li>Use case kết thúc.</li> </ol>		



Hình 3.20. Giao diện xóa tài khoản người dùng

Bảng 3.13. Đặc tả chức năng Quản lý tài khoản người dùng

### 3.3.2. UC-3.2 – Quản lý nội dung vi phạm

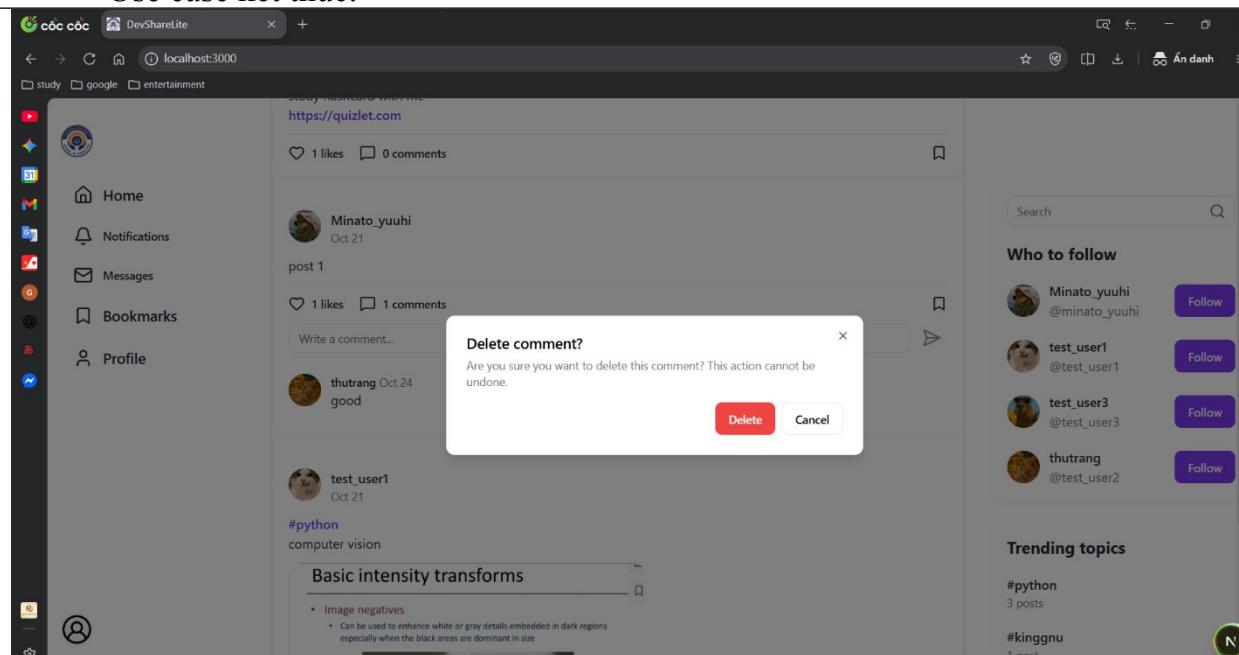
UC-3.2		QUẢN LÝ NỘI DUNG VI PHẠM
Mô tả		Chức năng cho phép Quản trị viên (Admin) xem xét các vi phạm, hoặc chủ động xử lý (xóa) các bài viết/bình luận vi phạm quy định.
Tác nhân		Quản trị viên (Admin)
Tiền điều kiện		Quản trị viên đã đăng nhập.
Hậu kiện	Thành công	Thành công: Nội dung vi phạm được xóa khỏi hệ thống.
	Lỗi	Nội dung không bị xóa.
<b>ĐẶC TẢ CHỨC NĂNG</b>		
+ Luồng sự kiện chính/Kịch bản chính (Xử lý trực tiếp)		
1. Chức năng bắt đầu khi Quản trị viên đăng nhập và duyệt các bài viết trên hệ thống. 2. Quản trị viên phát hiện một bài viết hoặc bình luận vi phạm. 3. Quản trị viên nhấp vào menu (dấu ba chấm) của bài viết/bình luận đó. 4. Vì là Admin, Quản trị viên sẽ thấy các tùy chọn "Xóa" mà người dùng thường không thấy. 5. Nếu muốn xóa bài viết, thực hiện Luồng «extend»: H1. Xóa bài viết (của người khác). 6. Nếu muốn xóa bình luận, thực hiện Luồng «extend»: H2. Xóa bình luận (của người khác).		

+ Luồng «extend»: H1. Xóa bài viết (của người khác)

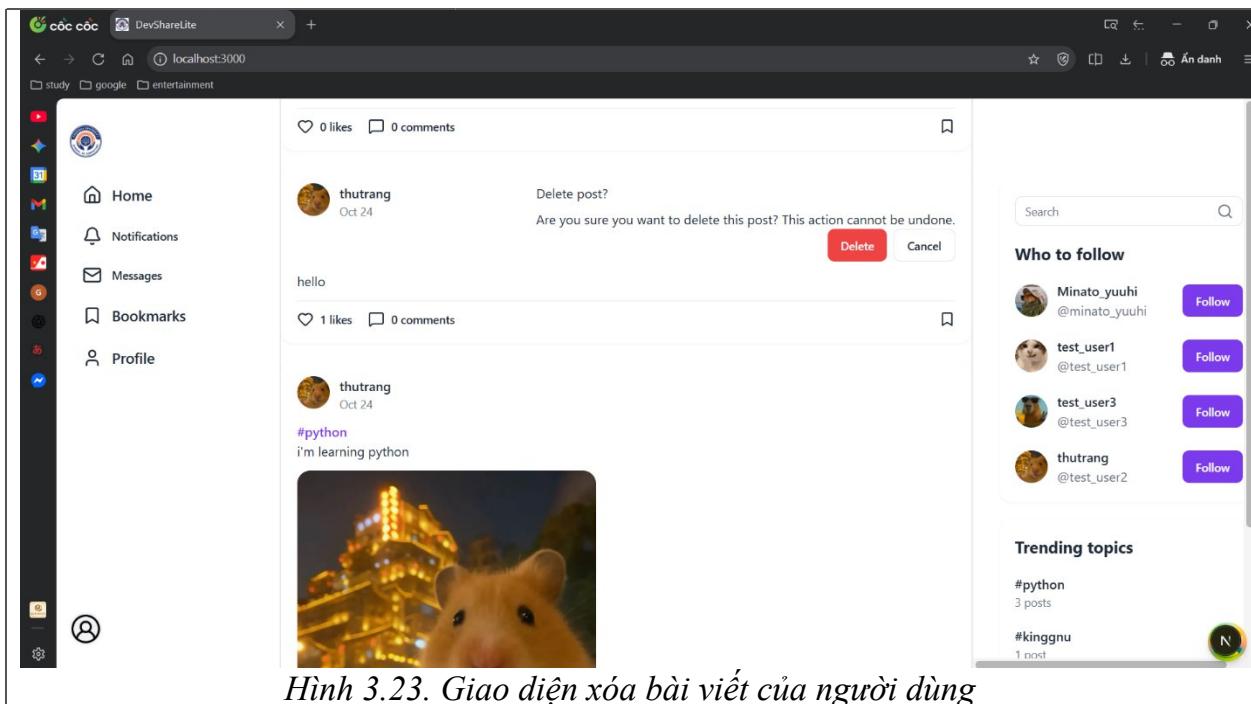
1. Quản trị viên chọn "Xóa bài viết".
2. Hệ thống hiển thị DeletePostDialog (Dialog xác nhận).
3. Quản trị viên nhấn "Xác nhận xóa".
4. Hệ thống (Server Action) xác thực quyền Admin, sau đó xóa bài viết khỏi CSDL (Postgres).
5. Bài viết biến mất khỏi giao diện.
6. Use case kết thúc.

+ Luồng «extend»: H2. Xóa bình luận (của người khác)

- Quản trị viên chọn "Xóa bình luận".
- Hệ thống hiển thị DeleteCommentDialog (Dialog xác nhận).
- Quản trị viên nhấn "Xác nhận xóa".
- Hệ thống (Server Action) xác thực quyền Admin, sau đó xóa bình luận khỏi CSDL.
- Bình luận biến mất khỏi giao diện.
- Use case kết thúc.



Hình 3.22. Giao diện xóa bình luận của người dùng



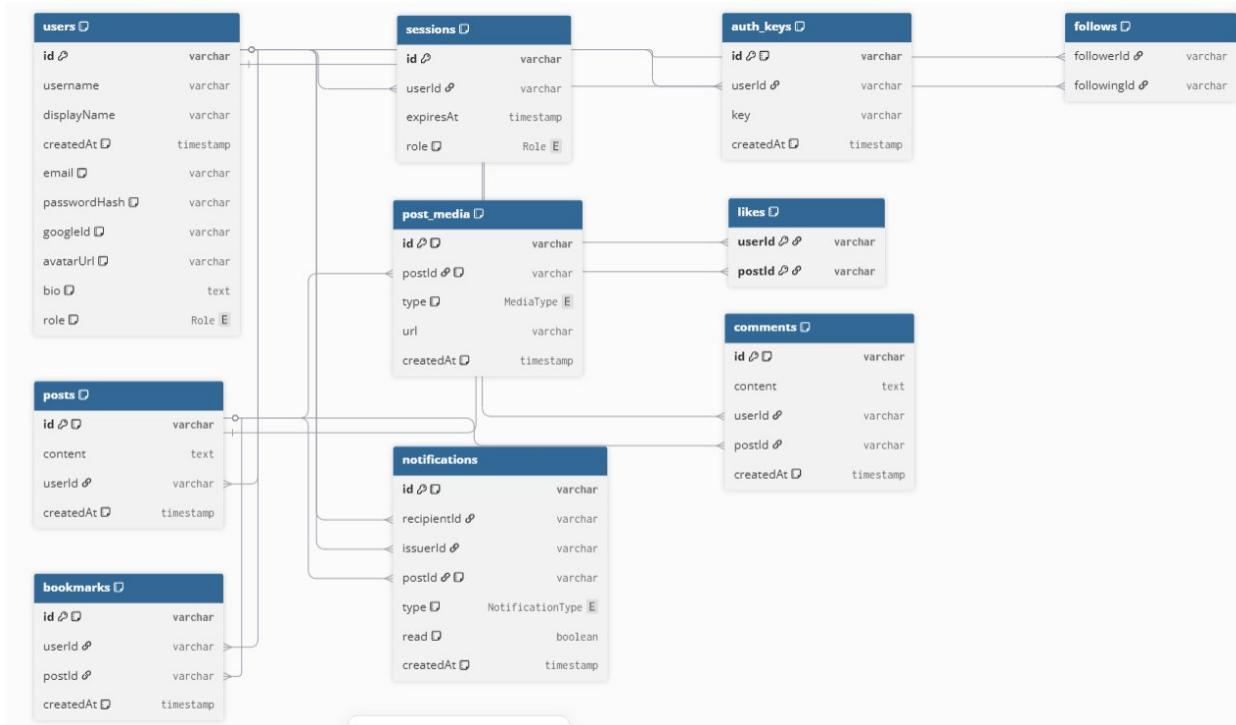
Hình 3.23. Giao diện xóa bài viết của người dùng

Bảng 3.14. Đặc tả chức năng Quản lý nội dung vi phạm

## CHƯƠNG 4. THIẾT KẾ PHẦN MỀM

### 4.1 Thiết kế cơ sở dữ liệu

#### 4.1.1. Lược đồ cơ sở dữ liệu cho các nhóm chức năng



Hình 4.1. Lược đồ cơ sở dữ liệu

#### 4.1.2. Danh sách các bảng: Số thứ tự, tên bảng và ý nghĩa bảng

STT	Tên bảng	Ý nghĩa
1.	User	Lưu trữ thông tin người dùng: tên đăng nhập, email, mật khẩu (mã hóa), ảnh đại diện, tiểu sử, vai trò (USER hoặc ADMIN). Đây là bảng trung tâm kết nối với các bảng khác như bài viết, bình luận, lượt thích, thông báo.
2.	Session	Quản lý phiên đăng nhập của người dùng. Mỗi khi người dùng đăng nhập, một bản ghi mới được tạo để theo dõi trạng thái và thời gian hết hạn phiên.
3.	Auth_Key	Lưu trữ khóa xác thực (key) cho cơ chế đăng nhập và xác thực người dùng, đặc biệt dùng khi triển khai đăng nhập Google hoặc token bảo mật.
4.	Follow	Quản lý mối quan hệ “theo dõi” giữa người dùng (ai theo dõi ai). Mỗi bản ghi thể hiện một kết nối follower-following.
5.	Post	Chứa dữ liệu bài viết của người dùng, bao gồm nội dung, thời gian tạo, liên kết tới người tạo, danh sách bình luận,

		lượt thích và tệp đính kèm. Đây là bảng trọng tâm trong hệ thống mạng xã hội.
6.	Media	Lưu trữ các tệp đa phương tiện (ảnh, video) được gắn kèm trong bài viết. Mỗi bản ghi chứa đường dẫn URL, loại file (IMAGE hoặc VIDEO) và liên kết đến bài viết tương ứng.
7.	Comments	Chứa các bình luận của người dùng trên bài viết. Mỗi bình luận liên kết với một người dùng và một bài viết cụ thể.
8.	Likes	Lưu các lượt “thích” bài viết. Mỗi bản ghi thể hiện mối quan hệ giữa một người dùng và một bài viết được thích.
9.	Bookmarks	Quản lý danh sách bài viết được người dùng lưu lại để xem sau. Mỗi bản ghi gồm người lưu, bài viết, và thời gian lưu.
10.	Notifications	Lưu thông tin các thông báo (like, comment, follow). Mỗi bản ghi ghi nhận người gửi, người nhận, bài viết liên quan, trạng thái đọc/chưa đọc.

Bảng 4.1. Danh sách các bảng sử dụng trong CSDL

#### 4.1.3. Chi tiết bảng

- User – Thông tin người dùng

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	id	String	Mã định danh duy nhất của người dùng
2.	username	String	Tên đăng nhập của người dùng, duy nhất trong hệ thống
3.	displayName	String	Tên hiển thị công khai trên hồ sơ
4.	createdAt	DateTime	Ngày tạo tài khoản
5.	email	String	Địa chỉ email (có thể rỗng)
6.	passwordHash	String	Mật khẩu đã được mã hóa
7.	googleId	String	Mã định danh tài khoản Google (nếu đăng nhập qua Google)
8.	avatarUrl	String	Đường dẫn ảnh đại diện
9.	bio	String	Phân mô tả ngắn của người dùng
10.	role	Enum(role)	Vai trò người dùng (USER hoặc ADMIN)

Bảng 4.2. Chi tiết bảng User

- Session – Phiên làm việc của người dùng

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	id	String	Mã định danh phiên đăng nhập
2.	userId	String	Khóa ngoại liên kết đến bảng users

3.	expiresAt	DateTime	Thời điểm hết hạn phiên
4.	role	Enum(Role)	Vai trò của người dùng trong phiên hiện tại

Bảng 4.3. Chi tiết bảng Session

- Auth\_key – Khóa xác thực

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	id	String	Mã định danh khóa xác thực
2.	userId	String	Khóa ngoại đến bảng users
3.	key	String	Chuỗi khóa xác thực (unique)
4.	createdAt	DateTime	Ngày tạo khóa xác thực

Bảng 4.4. Chi tiết bảng Auth\_key

- Follow – Quan hệ theo dõi người dùng

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	followerId	String	Người theo dõi (follower)
2.	followingId	String	Người được theo dõi (following)

Bảng 4.5. Chi tiết bảng Follow

- Post – Bài viết

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	id	String	Mã định danh bài viết
2.	content	String	Nội dung văn bản của bài viết
3.	userId	String	Khóa ngoại đến người đăng (users.id)
4.	createdAt	DateTime	Ngày giờ bài viết được tạo

Bảng 4.6. Chi tiết bảng Post

- Media – Phương tiện trong bài viết

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	id	String	Mã định danh phương tiện
2.	postId	String	Khóa ngoại liên kết bài viết chứa phương tiện
3.	type	Enum(MediaType)	Loại phương tiện: IMAGE hoặc VIDEO
4.	url	String	Đường dẫn file phương tiện
5.	createdAt	DateTime	Thời điểm tải lên phương tiện

Bảng 4.7. Chi tiết bảng Media

- Comments – Bình luận

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	id	String	Mã định danh bình luận
2.	content	String	Nội dung bình luận
3.	userId	String	Người đăng bình luận
4.	postId	String	Bài viết được bình luận
5.	createdAt	DateTime	Ngày tạo bình luận

Bảng 4.8. Chi tiết bảng Comments

- Likes – Lượt thích

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	userId	String	Người thực hiện hành động thích
2.	postId	String	Bài viết được thích

Bảng 4.9. Chi tiết bảng Likes

- Bookmarks – Bài viết đã lưu

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	id	String	Mã định danh bản ghi lưu
2.	userId	String	Người dùng thực hiện lưu
3.	postId	String	Bài viết được lưu
4.	createdAt	DateTime	Thời gian lưu bài viết

Bảng 4.10. Chi tiết bảng Bookmarks

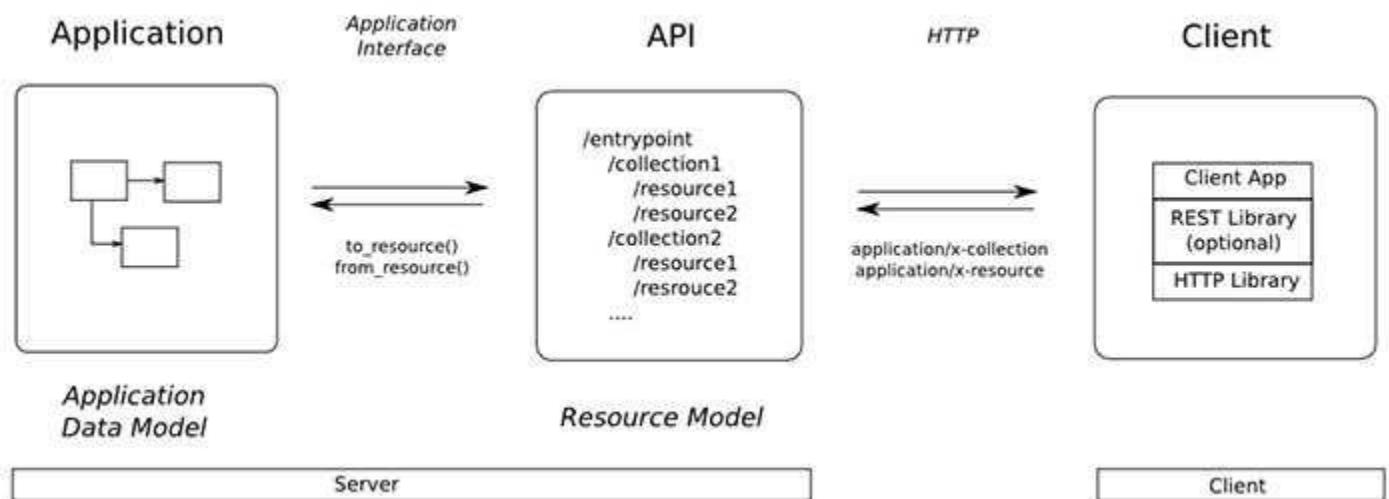
- Notifications – Thông báo hệ thống

STT	Tên field	Kiểu dữ liệu	Miêu tả
1.	id	String	Mã định danh thông báo
2.	recipientId	String	Người nhận thông báo
3.	issuerId	String	Người tạo thông báo (tác nhân)
4.	postId	String	Bài viết liên quan đến thông báo (nếu có)
5.	type	Enum(NotificationType)	Loại thông báo: LIKE, COMMENT, FOLLOW
6.	read	Boolean	Trạng thái đã đọc/chưa đọc
7.	createdAt	DateTime	Ngày tạo thông báo

Bảng 4.11. Chi tiết bảng Notifications

## 4.2. Thiết kế API

### 4.2.1. Cấu trúc thư mục API



REST (Representational State Transfer) là phong cách kiến trúc dùng để thiết kế các dịch vụ web. REST hoạt động chủ yếu dựa trên giao thức HTTP — giao thức nền tảng của web, giúp các hệ thống giao tiếp và trao đổi dữ liệu với nhau một cách đơn giản, linh hoạt và thống nhất.

Các hoạt động cơ bản trong REST tương ứng với các phương thức HTTP sau:

Phương thức HTTP	Chức năng (CRUD)	Ý nghĩa
GET	Read (SELECT)	Truy xuất hoặc đọc dữ liệu từ máy chủ. Trả về một Resource hoặc danh sách Resource.
POST	Create (INSERT)	Gửi dữ liệu mới lên máy chủ để tạo mới một Resource.
PUT	Update (UPDATE)	Cập nhật hoặc thay thế dữ liệu của một Resource hiện có.
DELETE	Delete (DELETE)	Xóa một Resource khỏi hệ thống.

Bảng 4.12. Các hoạt động cơ bản trong REST

### 4.2.2. Danh sách các API

STT	URL	Phương thức

Auth		
1.	/api/auth/callback/google	POST
2.	/api/get-token	GET
3.	/api/clear-uploads	DELETE
User		
4.	/api/users/[userId]	GET, PUT, DELETE
5.	/api/users/username/[username]	GET
6.	/api/users/[userId]/followers	GET
7.	/api/users/[userId]/posts	GET
Post		
8.	/api/posts	GET, POST
9.	/api/posts/for-you	GET
10.	/api/posts/following	GET
11.	/api/posts/bookmarked	GET
12.	/api/posts/[postId]	GET, DELETE
13.	/api/posts/[postId]/likes	POST, DELETE
14.	/api/posts/[postId]/bookmark	POST, DELETE
15.	/api/posts/[postId]/comments	GET, POST
Comment		
16.	/api/posts/[postId]/comments	POST, DELETE
Notification		
17.	/api/notifications	GET
18.	/api/notifications/mark-as-read	PATCH
19.	/api/notifications/unread-count	GET
Message		
20.	/api/messages/unread-count	GET
Search		
21.	/api/search	GET
Upload		
22.	/api/uploadthing	POST
Admin		
23.	/api/admin/delete-user	DELETE
24.	/api/admin/promote	PATCH

Bảng 4.13. Bảng danh sách các API

#### 4.2.3. Thiết kế chi tiết API

a) Auth – Quản lý đăng ký, đăng nhập và xác thực người dùng

URL	Mô tả chi tiết
/api/auth/callback/google	Phương thức: POST - Mục đích: Xác thực và đăng nhập người dùng bằng

	tài khoản Google. - Tham số: + token: String – Mã xác thực trả về từ Google.
/api/get-token	- Phương thức: GET - Mục đích: Lấy token truy cập hiện tại của người dùng đã đăng nhập.
/api/clear-uploads	- Phương thức: DELETE - Mục đích: Xóa các tệp tạm thời đã tải lên (chỉ dành cho Admin).

Bảng 4.14. Bảng mô tả API Quản lý Xác thực người dùng

b) User – Quản lý thông tin tài khoản người dùng

URL	Mô tả chi tiết
api/users/[userId]	- Phương thức: GET - Mục đích: Lấy thông tin hồ sơ của người dùng. - Tham số: + userId: String – Mã người dùng.
/api/users/[userId]	- Phương thức: PUT - Mục đích: Cập nhật thông tin hồ sơ cá nhân. - Tham số: + displayName: String + bio: String + avatarUrl: String
/api/users/username/[username]	- Phương thức: GET - Mục đích: Tìm người dùng theo tên tài khoản. - Tham số: + username: String
/api/users/[userId]/followers	- Phương thức: GET - Mục đích: Lấy danh sách người theo dõi tài khoản.
/api/users/[userId]/posts	- Phương thức: GET - Mục đích: Lấy danh sách bài viết của người dùng.

Bảng 4.15. Bảng mô tả API Quản lý thông tin tài khoản người dùng

c) Post – Quản lý bài viết

URL	Mô tả chi tiết
/api/posts	- Phương thức: POST - Mục đích: Tạo mới một bài viết. - Tham số: + content: String + attachments: [File]
/api/posts/[postId]	- Phương thức: GET

	<ul style="list-style-type: none"> <li>- Mục đích: Lấy thông tin chi tiết của một bài viết.</li> <li>- Tham số: <ul style="list-style-type: none"> <li>+ postId: String</li> </ul> </li> </ul>
/api/posts/[postId]	<ul style="list-style-type: none"> <li>- Phương thức: DELETE</li> <li>- Mục đích: Xóa bài viết (chỉ Admin hoặc người tạo).</li> </ul>
/api/posts/following	<ul style="list-style-type: none"> <li>- Phương thức: GET</li> <li>- Mục đích: Lấy danh sách bài viết từ những người đang theo dõi.</li> </ul>
/api/posts/for-you	<ul style="list-style-type: none"> <li>- Phương thức: GET</li> <li>- Mục đích: Lấy danh sách bài viết được đề xuất cho người dùng.</li> </ul>

Bảng 4.16. Bảng mô tả API Quản lý bài viết

#### d) Comment – Quản lý bình luận

URL	Mô tả chi tiết
/api/posts/[postId]/comments	<ul style="list-style-type: none"> <li>- Phương thức: POST</li> <li>- Mục đích: Thêm bình luận cho một bài viết.</li> <li>- Tham số: <ul style="list-style-type: none"> <li>+ postId: String</li> <li>+ content: String</li> </ul> </li> </ul>
/api/posts/[postId]/comments	<ul style="list-style-type: none"> <li>- Phương thức: DELETE</li> <li>- Mục đích: Xóa bình luận (chỉ người tạo hoặc quản trị viên).</li> </ul>

Bảng 4.17. Bảng mô tả API Quản lý bình luận

#### e) Notification – Quản lý thông báo

URL	Mô tả chi tiết
/api/notifications	<ul style="list-style-type: none"> <li>- Phương thức: GET</li> <li>- Mục đích: Lấy danh sách thông báo của người dùng</li> </ul>
/api/notifications/mark-as-read	<ul style="list-style-type: none"> <li>- Phương thức: PATCH</li> <li>- Mục đích: Đánh dấu tất cả thông báo là “đã đọc”.</li> </ul>
/api/notifications/unread-count	<ul style="list-style-type: none"> <li>- Phương thức: GET</li> <li>- Mục đích: Lấy số lượng thông báo chưa đọc.</li> </ul>

Bảng 4.18. Bảng mô tả API Quản lý thông báo

#### f) Message – Quản lý tin nhắn

URL	Mô tả chi tiết
/api/messages/unread-count	<ul style="list-style-type: none"> <li>- Phương thức: GET</li> <li>- Mục đích: Lấy số lượng tin nhắn chưa đọc của người dùng</li> </ul>

Bảng 4.19. Bảng mô tả API Quản lý tin nhắn

g) Admin – Quản trị viên hệ thống

URL	Mô tả chi tiết
/api/admin/delete-user	<ul style="list-style-type: none"> <li>- Phương thức: DELETE</li> <li>- Mục đích: Xóa tài khoản của người dùng bất kỳ (chỉ Admin).</li> <li>- Tham số: <ul style="list-style-type: none"> <li>+ userId: String</li> </ul> </li> </ul>
/api/admin/promote	<ul style="list-style-type: none"> <li>- Phương thức: PATCH</li> <li>- Mục đích: Nâng quyền người dùng thành Admin.</li> <li>- Tham số: <ul style="list-style-type: none"> <li>+ userId: String</li> </ul> </li> </ul>

Bảng 4.20. Bảng mô tả API dành riêng cho quản trị viên

h) Upload & Search – Các API phụ trợ

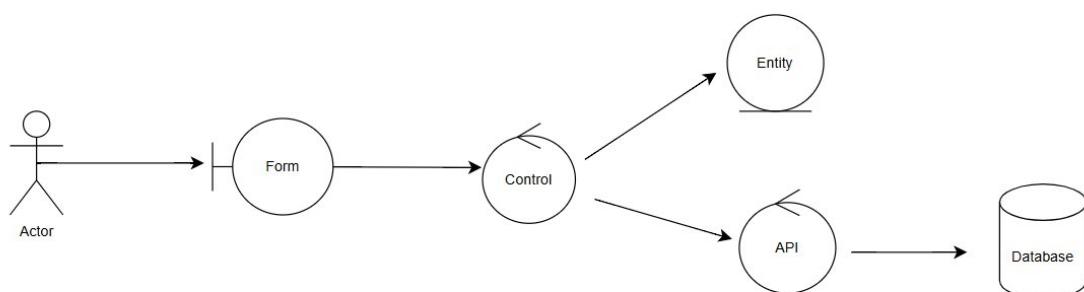
URL	Mô tả chi tiết
/api/uploadthing	<ul style="list-style-type: none"> <li>- Phương thức: POST</li> <li>- Mục đích: Tải lên tệp đa phương tiện (anh, video).</li> </ul>
/api/search	<ul style="list-style-type: none"> <li>- Phương thức: GET</li> <li>- Mục đích: Tìm kiếm bài viết hoặc người dùng theo từ khóa.</li> <li>- Tham số: <ul style="list-style-type: none"> <li>+ query: String</li> </ul> </li> </ul>

Bảng 4.21. Bảng mô tả các API phụ trợ

### 4.3. Thiết kế chức năng

#### 4.3.1. Cấu trúc chung

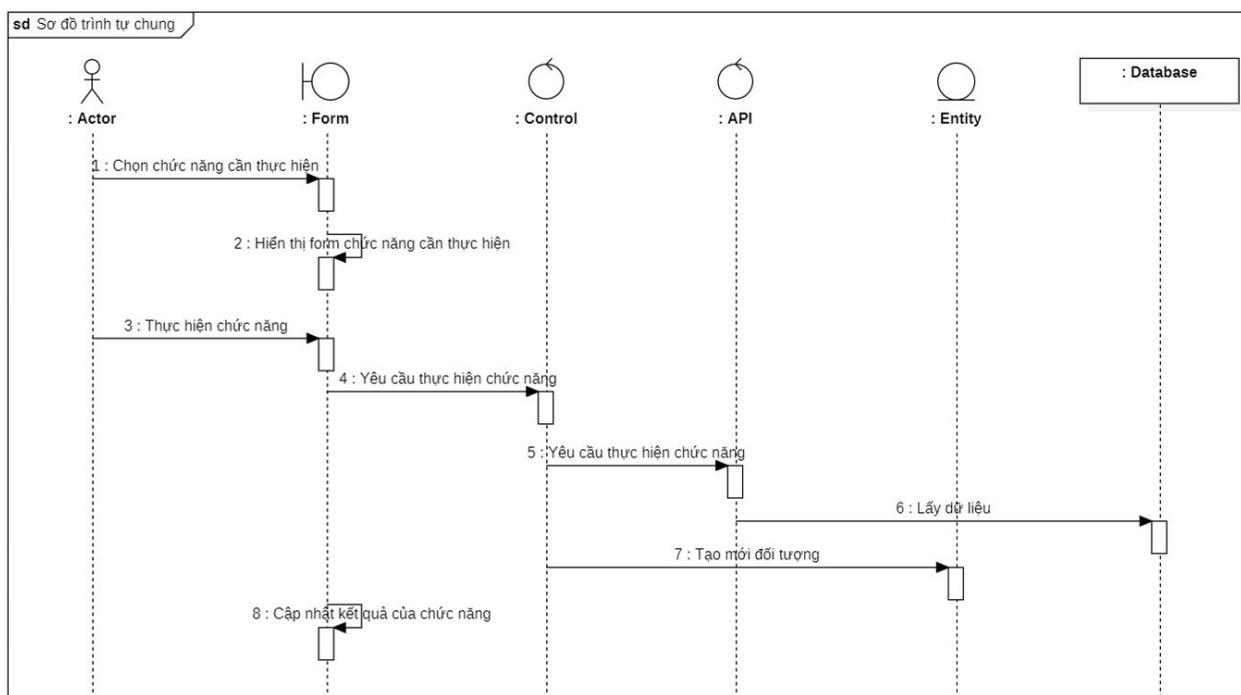
- ❖ Sơ đồ lớp phân tích:



*Hình 4.2. Sơ đồ lớp phân tích chung*

- **Actor:** Tác nhân tham gia sử dụng hệ thống.
- **Form:** Form là lớp trung gian thể hiện sự tương tác giữa hệ thống và những gì bên ngoài hệ thống. Ở đây là giao diện giữa người dùng và hệ thống.
- **Control:** Lớp điều khiển thực hiện chức năng chính của UC.
- **Entity:** Lớp thực thể chứa những thông tin tồn tại và được lưu trữ lâu dài trong hệ thống.
- **API:** Lớp điều khiển thực hiện việc nhận/trả các API.
- **Database:** Cơ sở dữ liệu.

❖ Sơ đồ trình tự:



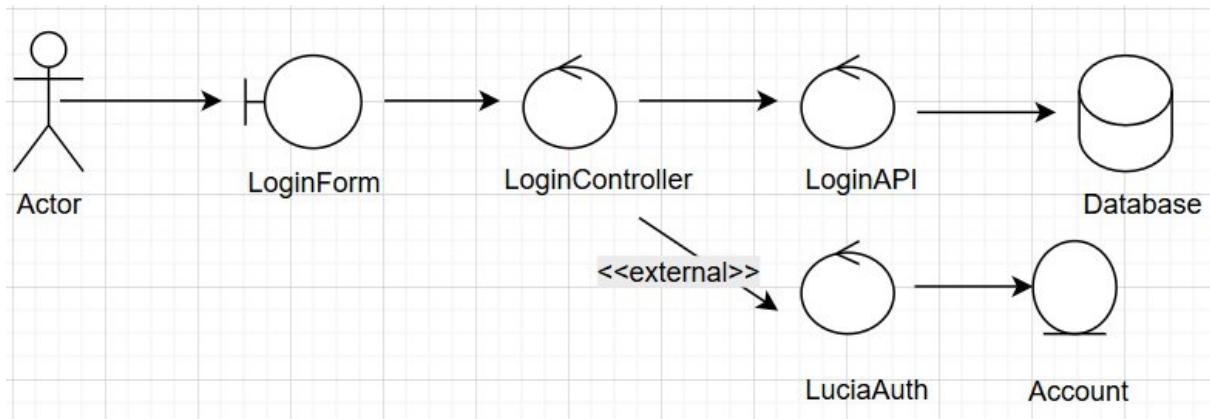
*Hình 4.3. Sơ đồ trình tự chung*

### 4.3.2. Thiết kế một số chức năng

#### 4.3.2.1. Đăng nhập

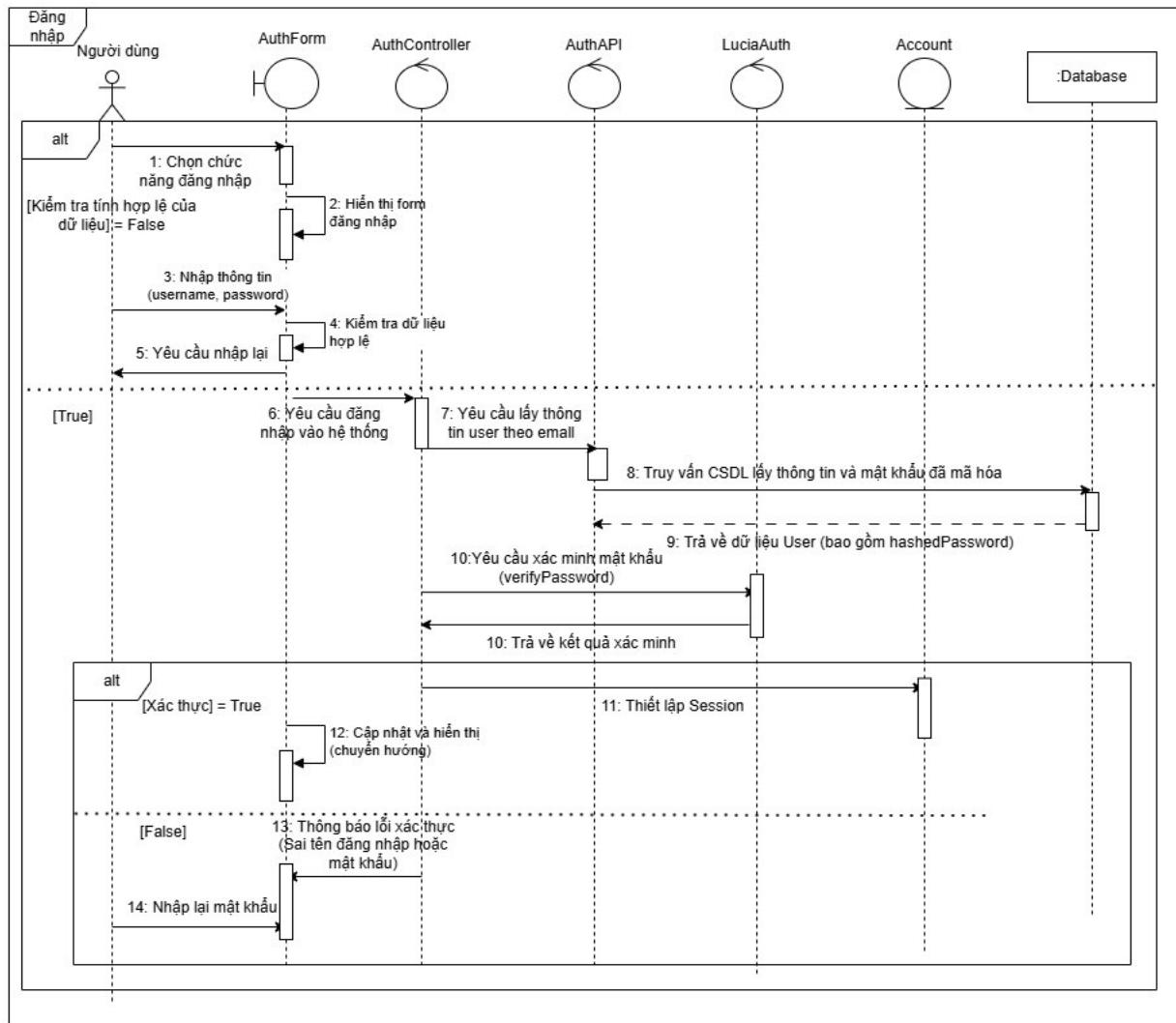
- Sơ đồ lớp phân tích:

Sơ đồ lớp phân tích của chức năng đăng nhập



Hình 4.4. Sơ đồ lớp phân tích của chức năng đăng nhập

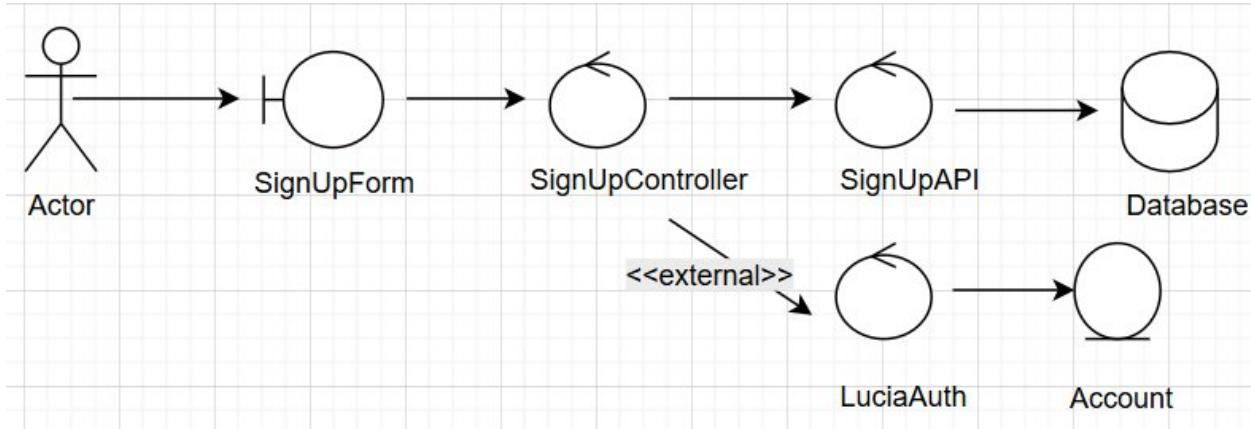
-Sơ đồ trình tự:



*Hình 4.5. Sơ đồ trình tự của chức năng đăng nhập*

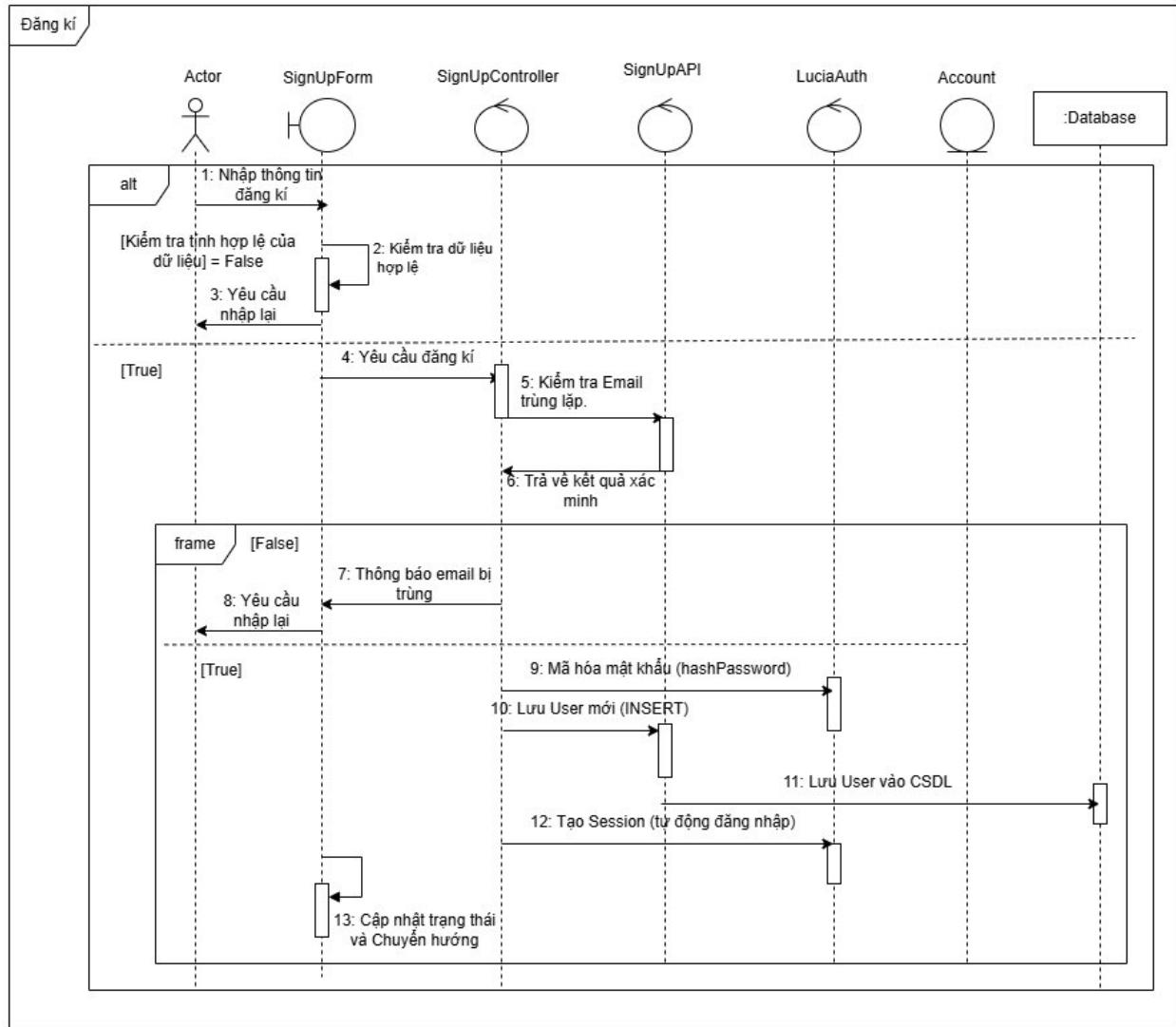
#### 4.3.2.2. Đăng ký

- Sơ đồ lớp phân tích:



*Hình 4.6. Sơ đồ lớp phân tích của chức năng đăng ký*

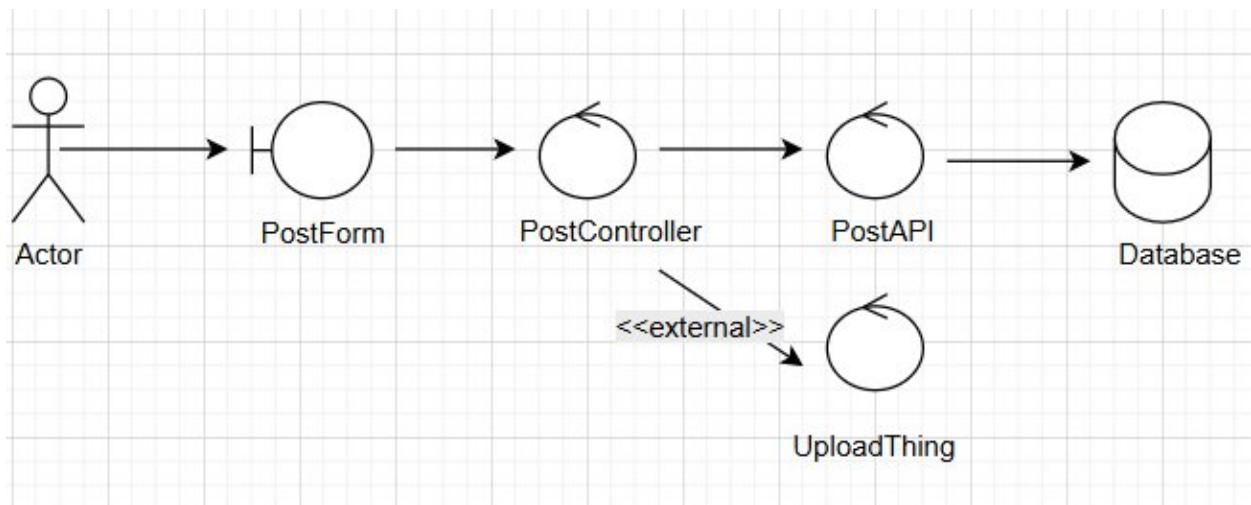
- Sơ đồ trình tự:



Hình 4.7. Sơ đồ trình tự của chức năng đăng kí

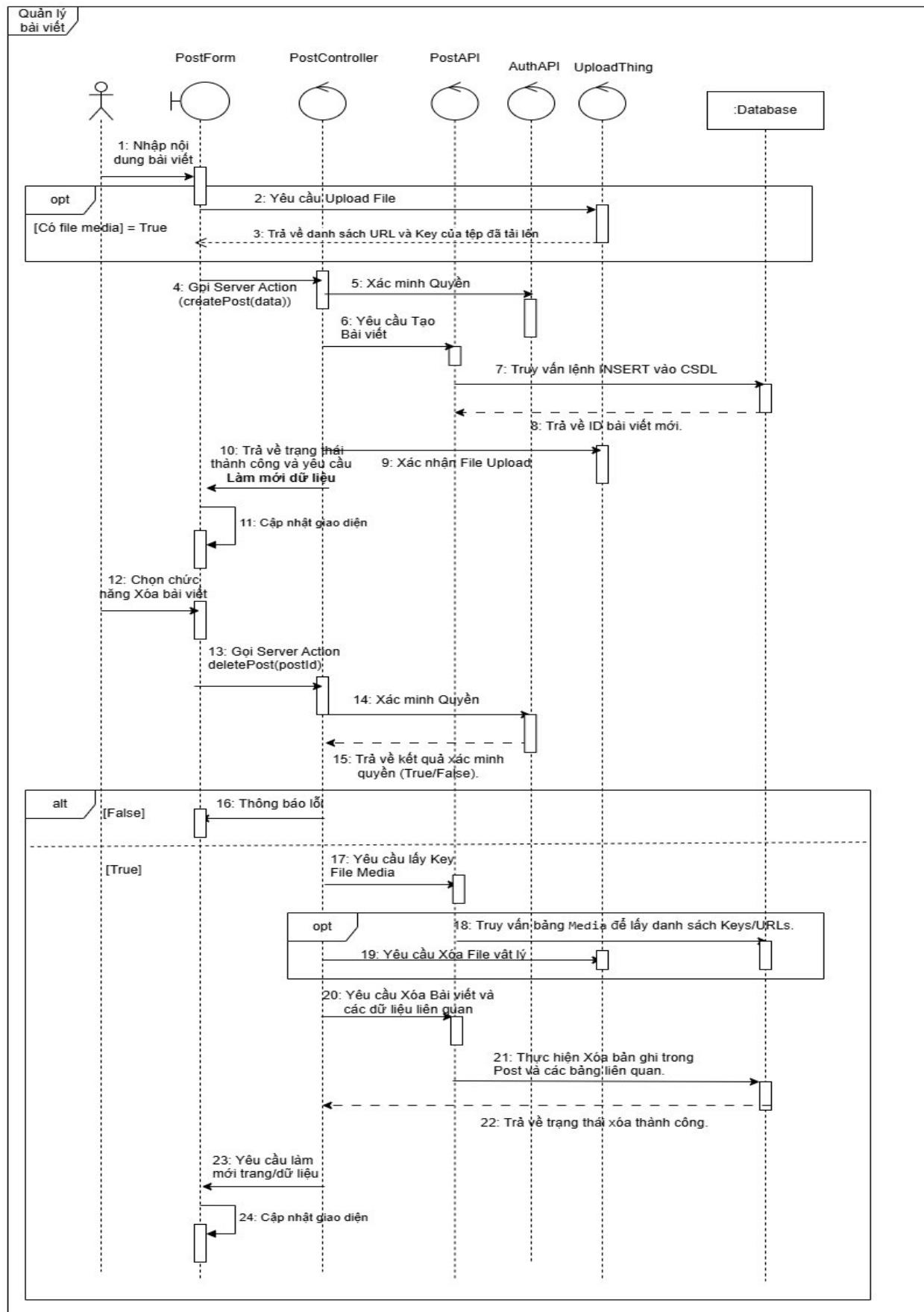
#### 4.3.2.3. Quản lý bài viết

- Sơ đồ lớp phân tích:



*Hình 4.8. Sơ đồ lớp phân tích của chức năng quản lý bài đăng*

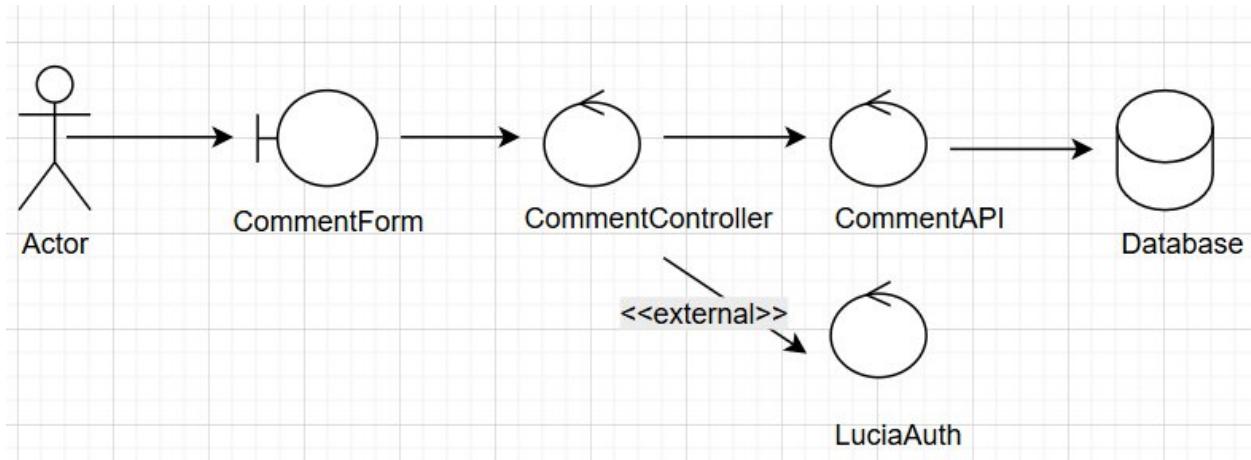
- Sơ đồ trình tự:



Hình 4.9. Sơ đồ trình tự của chức năng quản lý bài đăng

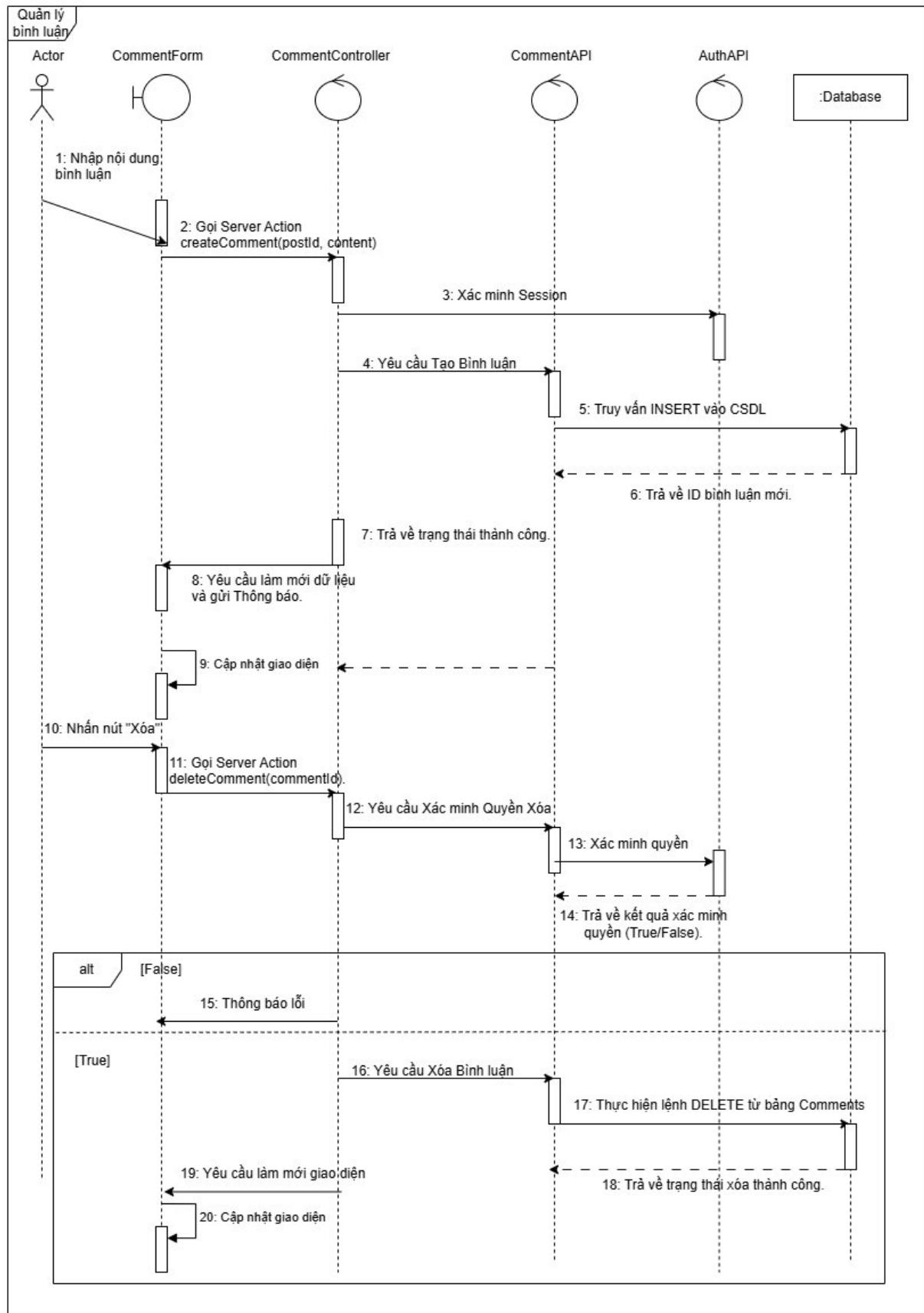
#### 4.3.2.4. Bình luận bài viết

- Sơ đồ lớp phân tích:



Hình 4.10. Sơ đồ lớp phân tích của chức năng bình luận bài viết

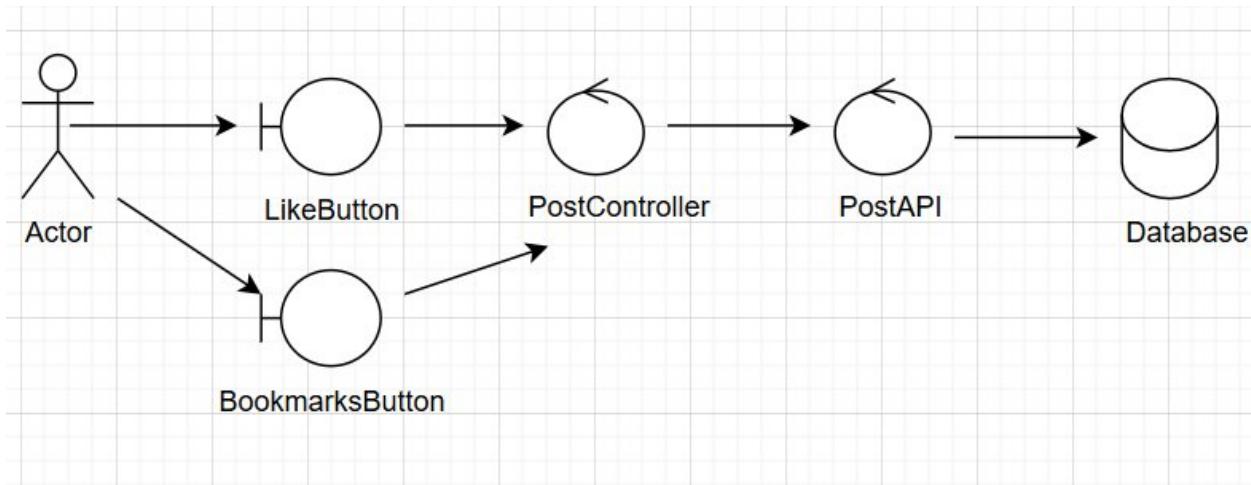
- Sơ đồ trình tự:



Hình 4.11. Sơ đồ trình tự của chức năng bình luận bài viết

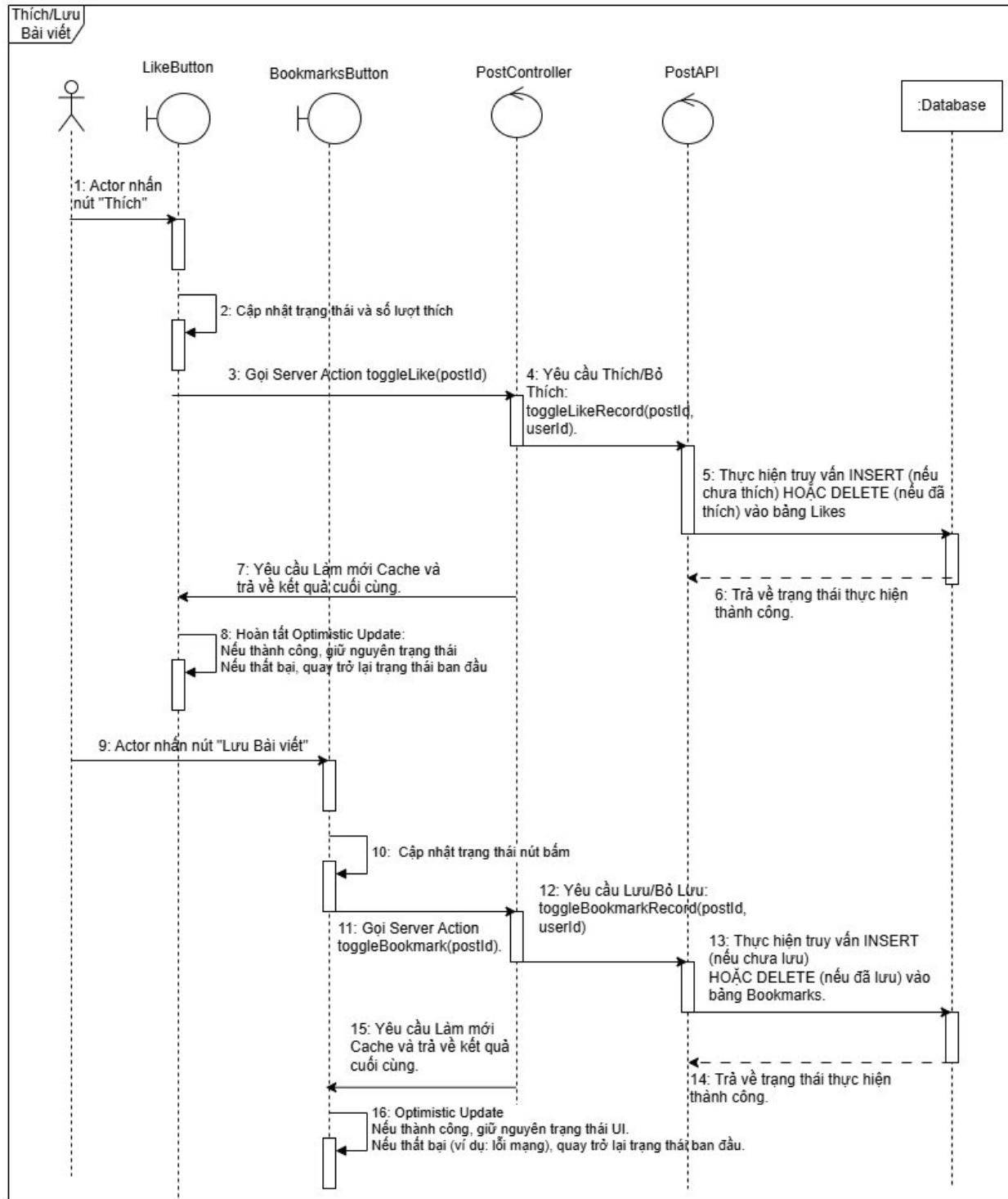
#### 4.3.2.4. Thích/ Lưu bài viết

- Sơ đồ lớp phân tích:



Hình 4.12. Sơ đồ lớp phân tích của chức năng thích/ lưu bài viết

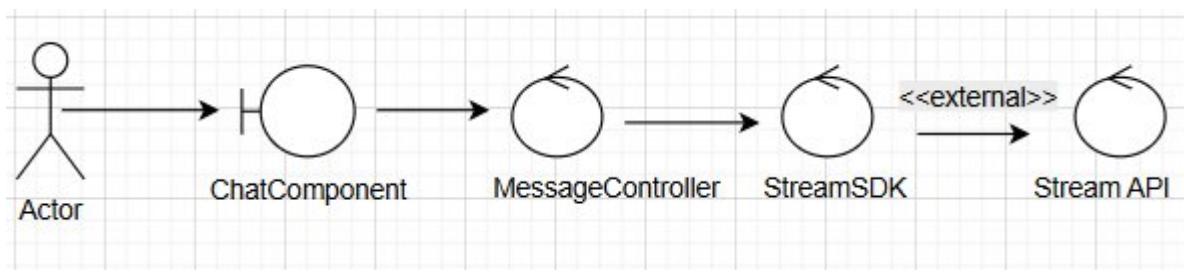
- Sơ đồ trình tự:



Hình 4.13. Sơ đồ trình tự của chức năng thích/lưu bài viết

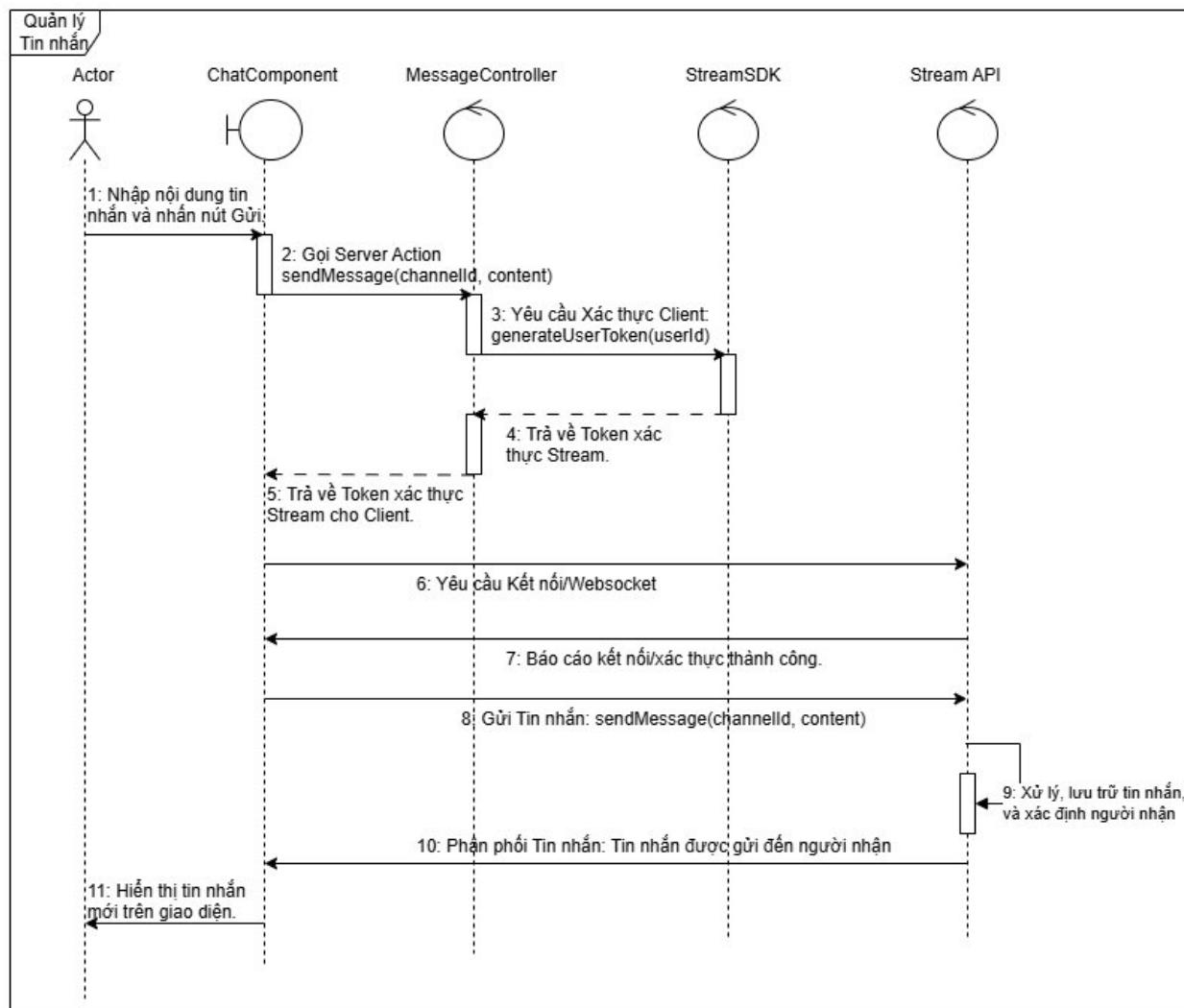
#### 4.3.2.5. Gửi tin nhắn (DM)

- Sơ đồ lớp phân tích:



Hình 4.14. Sơ đồ lớp phân tích của chức năng gửi tin nhắn (DM)

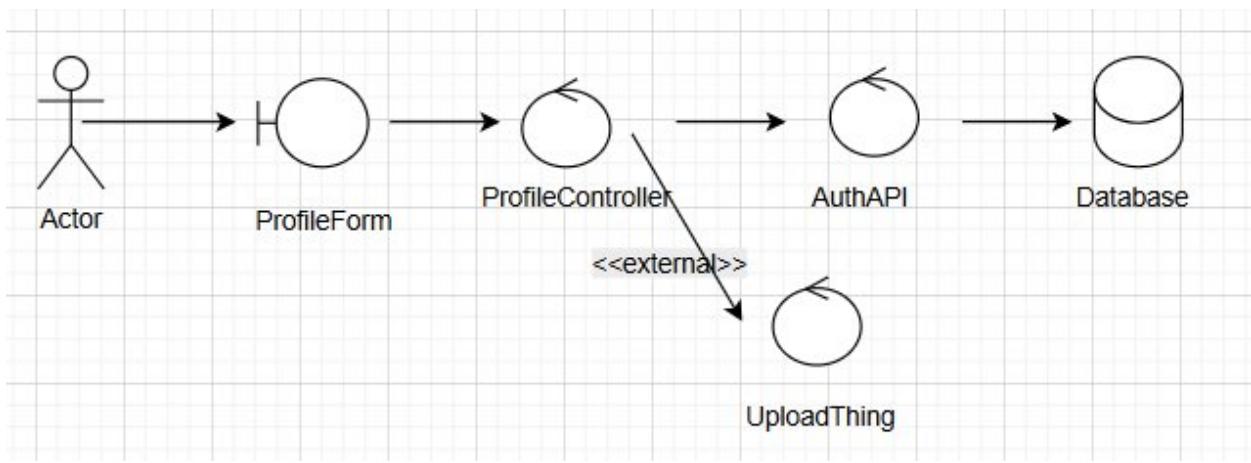
- Sơ đồ trình tự:



Hình 4.15. Sơ đồ trình tự của chức năng gửi tin nhắn (DM)

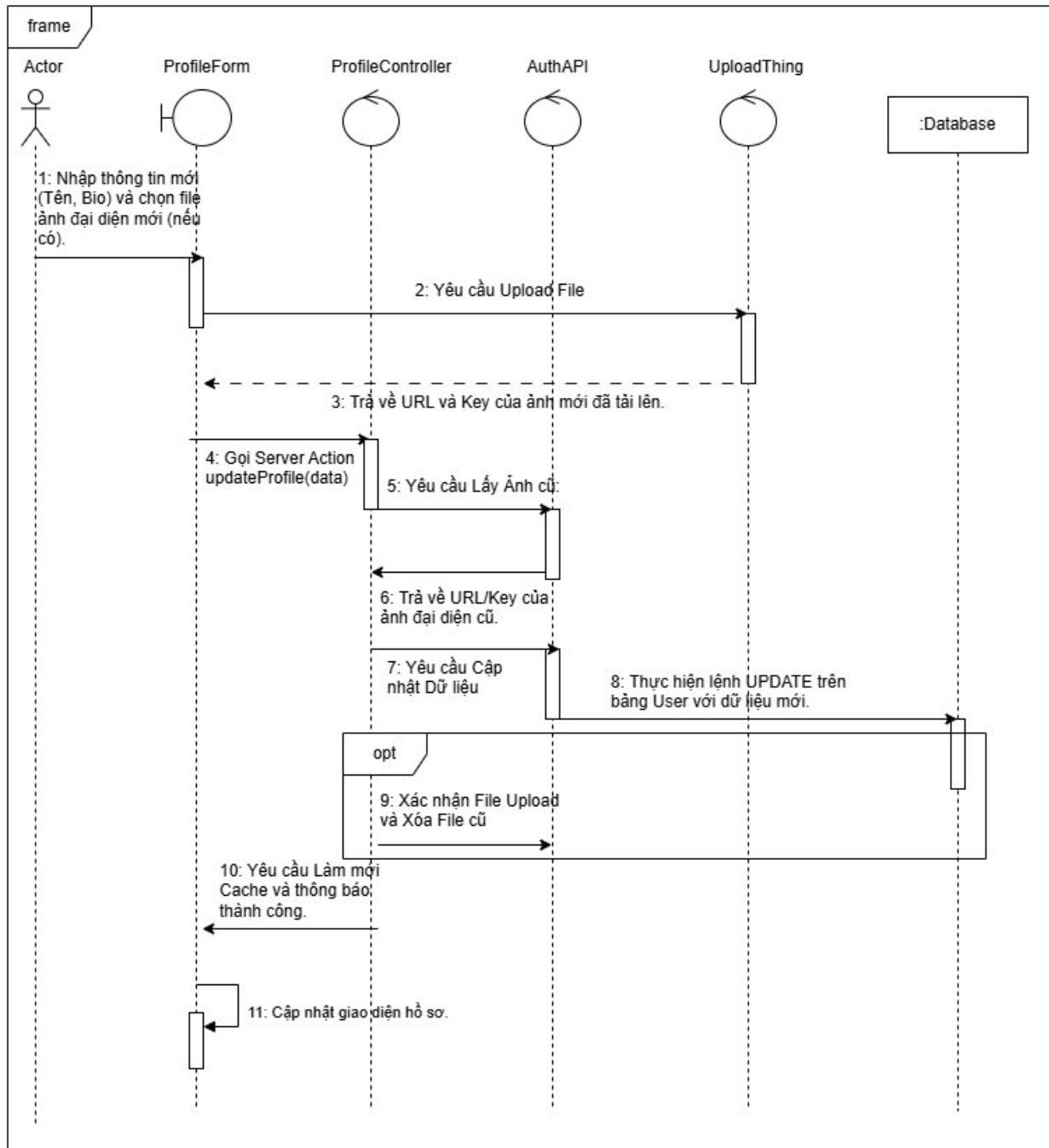
#### 4.3.2.6. Cập nhật hồ sơ cá nhân

- Sơ đồ lớp phân tích:



*Hình 4.16. Sơ đồ lớp phân tích của chức năng cập nhật hồ sơ cá nhân*

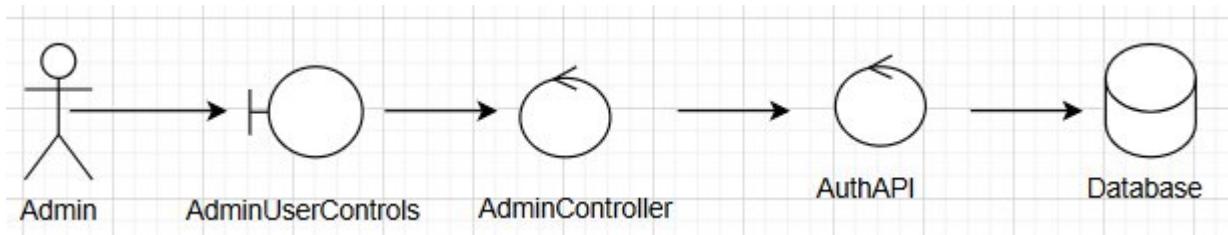
- Sơ đồ trình tự:



Hình 4.17. Sơ đồ trình tự của chức năng cập nhật hồ sơ cá nhân

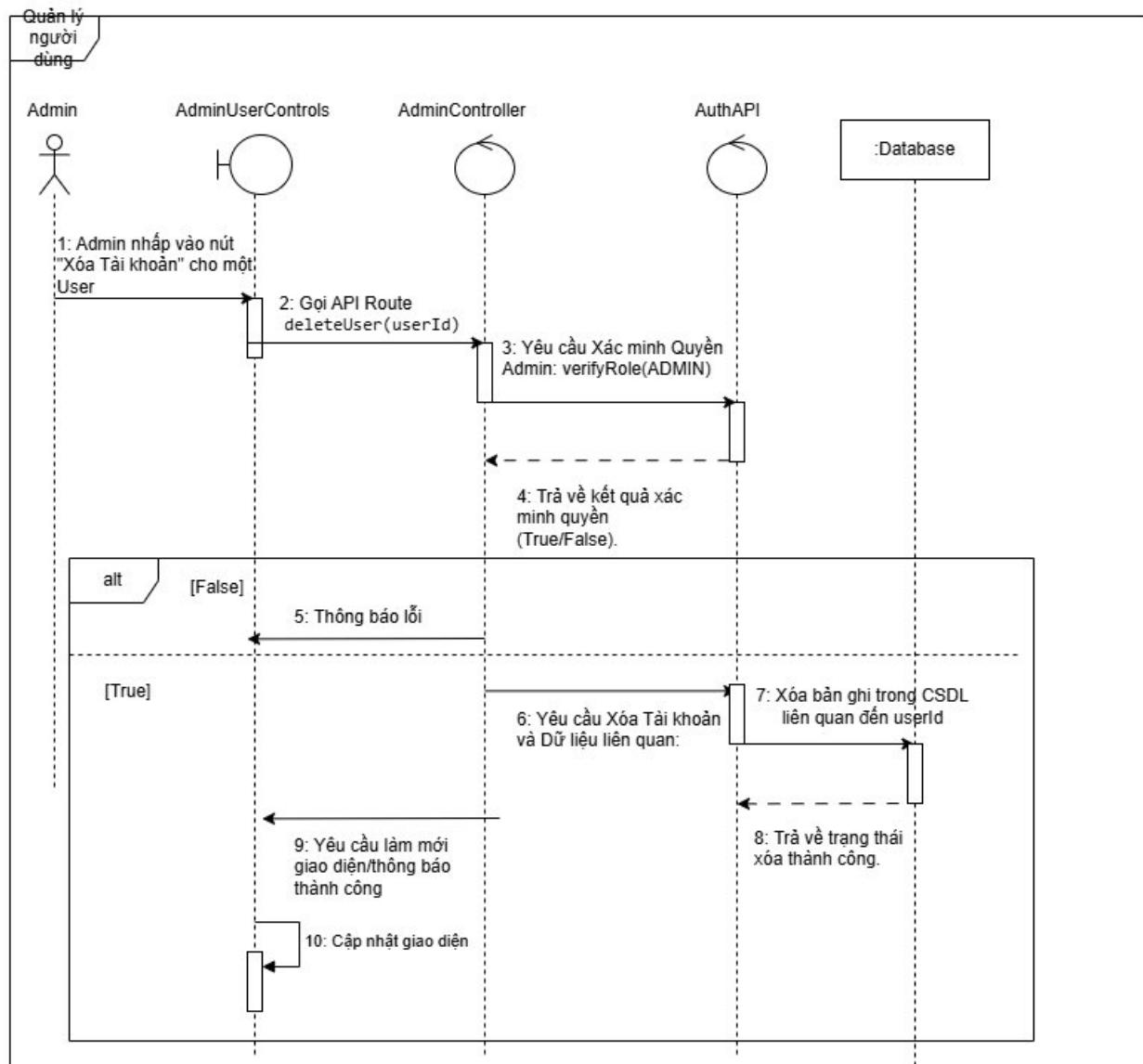
#### 4.3.2.7. Quản lý người dùng

- Sơ đồ lớp phân tích:



Hình 4.18. Sơ đồ lớp phân tích của chức năng quản lý người dùng

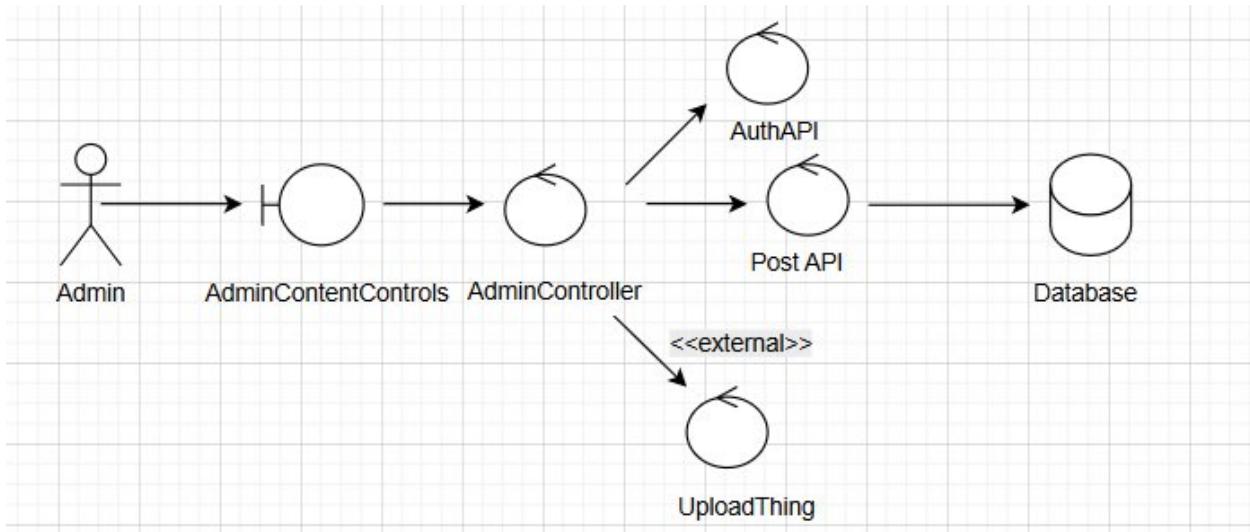
- Sơ đồ trình tự:



Hình 4.19. Sơ đồ trình tự của chức năng quản lý người dùng

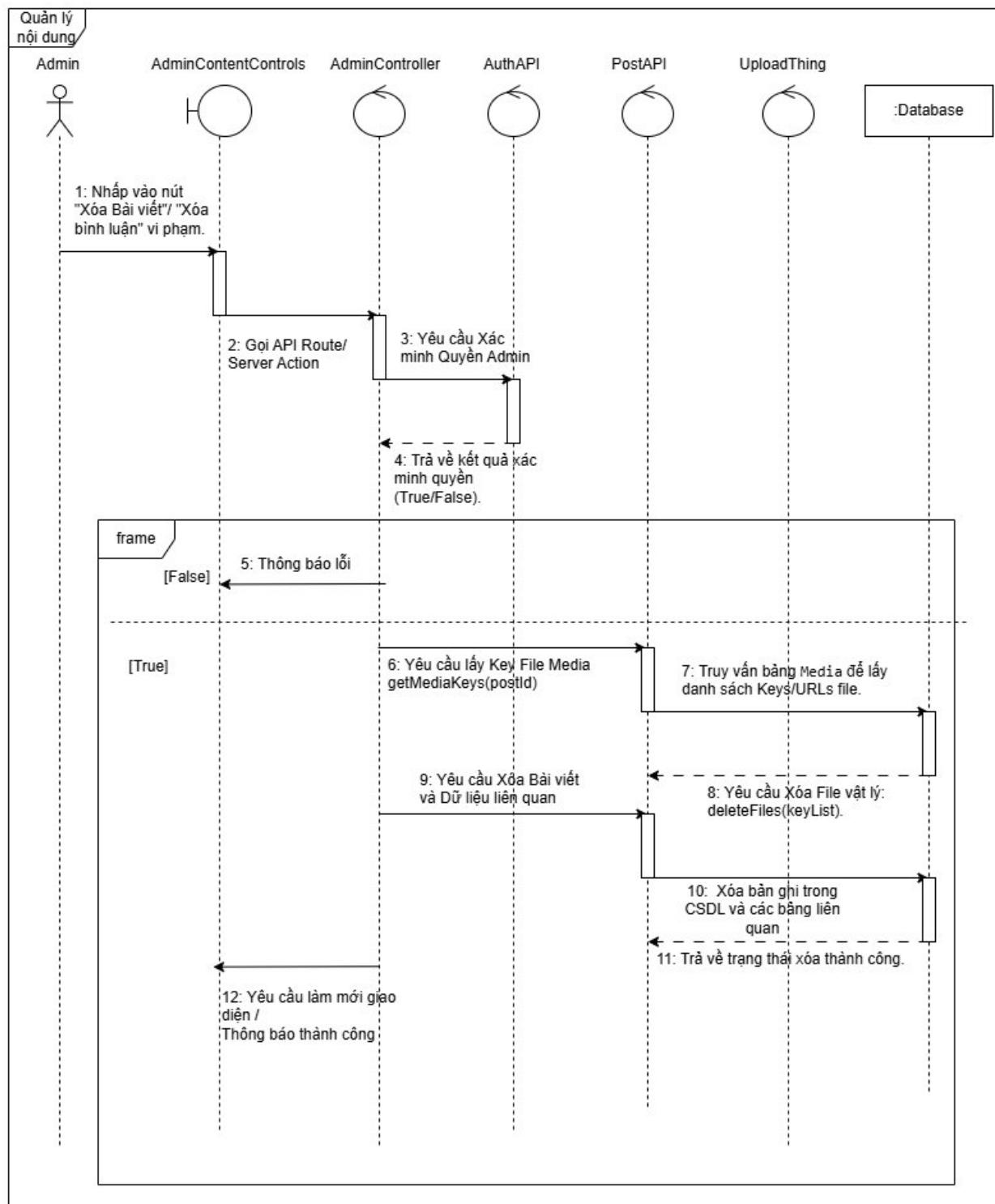
#### 4.3.2.8. Quản lý nội dung vi phạm

- Sơ đồ lớp phân tích:



Hình 4.20. Sơ đồ lớp phân tích của chức năng quản lý nội dung vi phạm

- Sơ đồ trình tự:



Hình 4.21. Sơ đồ trình tự của chức năng quản lý nội dung vi phạm

## CHƯƠNG 5. CÀI ĐẶT VÀ KIỂM THỬ

### 5.1. Cài đặt

#### 5.1.1. Cấu trúc thư mục mã nguồn

Thư mục / File	Chức năng & Vai trò (Server hay Client)
app/(auth)	(Nhóm Route) Chứa các trang không yêu cầu đăng nhập (Login, Signup). Chạy chủ yếu ở Client.
app/(main)	(Nhóm Route) Chứa các tính năng chính (Home, Messages, Profile). Chứa cả Server Components (render dữ liệu) và Client Components (tương tác).
(main)/page.tsx	Server Component. Trang chủ chính, fetch và hiển thị ForYouFeed.
(main)/users/[username]/page.tsx	(main)/users/[username]/page.tsx Server Component. Trang hồ sơ, fetch dữ liệu user và các bài post của họ.
(main)/messages/Chat.tsx	Client Component (đánh dấu "use client"). Dùng để tương tác real-time với Stream API.
app/api	Server. Chứa các API routes truyền thống, dùng cho webhook (UploadThing) hoặc các tác vụ Admin (/api/admin/delete-user).
(auth)/login/actions.ts	Server. Chứa các hàm Server Actions để xử lý logic đăng nhập phía máy chủ.

Bảng 5.1. Chi tiết thư mục logic nghiệp vụ và routing (src/app)

Thư mục / File	Chức năng & Vai trò (Chủ yếu là Client Components)
components/ui	Các component "nguyên tử" từ Shadcn UI (Button, Dialog, Input...). Đều là "use client".
components/posts/PostEditor.tsx	Client Component. Trình soạn thảo TipTap để người dùng tạo bài viết mới.
components/posts/LikeButton.tsx	Client Component. Xử lý logic nhán "Like", gọi Server Action và thực hiện Optimistic Update.
components/posts/BookmarkButton.tsx	Client Component. Tương tự LikeButton, dùng cho chức năng Bookmark.
components/AdminUserControls.tsx	Client Component. Hiển thị các nút (Xóa User) chỉ dành cho Admin.

Bảng 5.2. Chi tiết thư mục component và UI (src/components)

<b>Thư mục / File</b>	<b>Chức năng &amp; Vai trò (Chỉ chạy ở Server)</b>
prisma/schema.prisma	Định nghĩa mô hình Cơ sở dữ liệu Postgres (User, Post, Comment...).
lib/prisma.ts	Khởi tạo đối tượng PrismaClient để giao tiếp với CSDL Postgres.
lib/auth.ts	Cấu hình chính của Lucia Auth (adapter, session...).
lib/stream.ts	Khởi tạo SDK và kết nối đến Stream API (dịch vụ DM).
lib/uploadthing.ts	Cấu hình và đăng ký file router cho UploadThing (dịch vụ Upload).
lib/validation.ts	Chứa các schema của Zod để kiểm tra (validate) dữ liệu.

Bảng 5.3. Chi tiết thư mục lõi, dịch vụ và CSDL (src/lib, prisma, auth.ts)

### 5.1.2. Yêu cầu môi trường & Hướng dẫn cài đặt

Yêu cầu môi trường:

- Node.js v18.x trở lên
- NPM (hoặc Pnpm/Yarn)
- Một server PostgreSQL đang chạy
- VS Code

Đường link github dự án: [https://github.com/thutrangym/social\\_media\\_web.git](https://github.com/thutrangym/social_media_web.git)

Hướng dẫn cài đặt:

- Clone repository: git clone ...
- Cài đặt thư viện: npm install
- Tạo file .env.local (dựa trên file .env.example) và điền các khóa DATABASE\_URL, STREAM\_API\_KEY, UPLOADTHING\_SECRET, v.v.
- Chạy migration để tạo bảng CSDL: npx prisma migrate dev
- Chạy dự án: npm run dev

### 5.1.3. Ánh xạ chức năng và file cài đặt

Use Case (UC)	Tác nhân	Các file/thư mục chính thực hiện
UC2.1: Đăng ký	Guest	app/(auth)/signup/ (UI & Form), lib/validation.ts (Zod), (auth)/actions.ts (Server Action)
UC2.2: Đăng nhập	Guest	app/(auth)/login/ (UI & Form), (auth)/login/google/ (Google OAuth)
UC3.1: Đăng	User	components/posts/editor/PostEditor.tsx (UI),

bài		components/posts/editor/action.ts (Server Action)
UC3.5: Gửi tin nhắn	User	app/(main)/messages/ (Layout), Chat.tsx (UI), lib/stream.ts (Cấu hình Stream API)
UC2.5: Xóa User	Admin	components/AdminUserControls.tsx (UI), app/api/admin/delete-user/route.ts (API Route xử lý)

Bảng 5.4. Ánh xạ Use Case và file mã nguồn chính

## 5.2. Kiểm thử

### 5.2.1. Giới thiệu

Trong dự án này, chúng tôi áp dụng phương pháp kiểm thử chức năng thủ công (Manual Functional Testing) để đảm bảo các tính năng cốt lõi của hệ thống hoạt động chính xác theo đặc tả kỹ thuật.

Quy trình kiểm thử tập trung kiểm thử các chức năng quan trọng nhất bao gồm các chức năng sau:

- Quản lý tài khoản cá nhân: Người dùng có thể đăng nhập, đăng ký, cập nhật thông tin với tài khoản của mình, theo dõi người dùng khác. Quản trị viên có thêm quyền xóa người dùng khỏi hệ thống.
- Quản lý dòng thời gian: Cho phép người dùng đăng và tương tác với bài viết, bình luận, theo dõi và nhận thông báo về bài viết, lưu bài viết và xóa bài viết của mình. Đối với quản trị viên có thể xóa bài của mình và người dùng khác.
- Hệ thống nhắn tin: Cho phép người dùng có thể nhắn tin với nhau theo thời gian thực, tạo nhóm để trò chuyện, gửi tài liệu, ảnh, tạo bình chọn (vote).

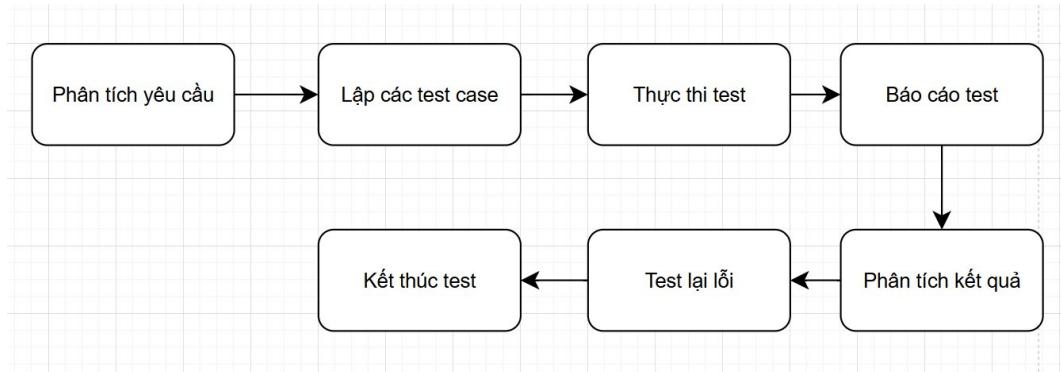
Ngoài ra, dự án cũng tiến hành một số kiểm thử phi chức năng:

- Hiệu năng: Đảm bảo thời gian tải trang nhanh, giao diện mượt mà
- Bảo mật: Chỉ người tạo và quản trị viên mới được xóa bài đăng, bình luận.

Mục tiêu:

- Đảm bảo tất cả các chức năng (đã định nghĩa trong Use Case) hoạt động chính xác.
- Xác minh các luồng nghiệp vụ (đăng ký, đăng bài, quản trị) chạy đúng logic.
- Kiểm tra tính hợp lệ của dữ liệu (validation) tại các biểu mẫu.
- Đảm bảo hệ thống tích hợp đúng với các dịch vụ bên thứ ba (Stream, UploadThing, Google Auth).

### 5.2.2. Quy trình kiểm thử



Hình 5.1. Quy trình kiểm thử

1. Phân tích yêu cầu:
  - Thu thập xem xét các yêu cầu chức năng (UC001-UC010) và phi chức năng trong tài liệu đặc tả.
  - Xác định các chức năng chính cần kiểm thử.
  - Xác định điều kiện vào, kết quả mong đợi, quyền truy cập của người dùng và quản trị viên.
  - Mục tiêu là đảm bảo rõ hành vi của hệ thống trước khi thiết kế kịch bản kiểm thử.
2. Lập các test case: Dựa trên từng Usecase, nhóm tiến hành xây dựng danh sách test case cụ thể.
3. Thực thi test:
  - Tiến hành kiểm thử thủ công (Manual Testing) trên trình duyệt Chrome.
  - Dữ liệu giả lập: 5 tài khoản Người dùng, 3 Quản trị viên, 20 bài viết, 10 bình luận.
  - Ghi nhận trạng thái từng test case.
4. Báo cáo test:

Tổng hợp kết quả kiểm thử trong bảng tổng hợp.  
Ghi chú rõ những lỗi hoặc độ trễ phát sinh, đặc biệt: Gửi tin nhắn hơi lag, các test case còn lại hoạt động ổn định.
5. Phân tích kết quả test:

Kết quả cho thấy hệ thống đáp ứng hầu hết yêu cầu chức năng và phi chức năng, ngoại trừ chức năng truyền tải real-time cần cải thiện.
6. Test lại lỗi:

Lỗi về độ trễ tin nhắn được ghi nhận và chuyển cho nhóm phát triển để tối ưu WebSocket.  
Sau khi fix, nhóm kiểm thử lại trên mạng ổn định, thời gian phản hồi giảm, hệ thống hoạt động tốt hơn.

7. Kết thúc test:

Toàn bộ các module chính đã được kiểm thử đầy đủ

Tỉ lệ test pass cao sau khi sửa lỗi

Kết luận: Hệ thống đạt yêu cầu kiểm thử.

### **5.2.3. Thực hiện kiểm thử**

#### **5.2.3.1. Kiểm thử cho Người dùng**

Bảng 5.5. Bảng tổng hợp ca kiểm thử cho người dùng (Test Case Summary)

STT	Mục tiêu kiểm thử	Ý nghĩa	Trạng thái
1.	Kiểm tra chức năng Đăng ký tài khoản	Kiểm tra các trường thông tin không được bỏ trống. Kiểm tra tính duy nhất và định dạng email. Đăng ký và hệ thống tạo tài khoản mới, chuyển đến “Trang chủ”. Đăng ký thất bại. Hiển thị thông báo “Email đã được đăng ký” hoặc báo lỗi.	Pass
2.	Kiểm tra chức năng Đăng nhập bằng tài khoản	Kiểm tra các trường thông tin không được bỏ trống. Đăng nhập thành công vào hệ thống. Hiển thị “Trang chủ” Đăng nhập thất bại. Hiển thị thông báo lỗi: “Sai tên đăng nhập hoặc mật khẩu” nếu tên đăng nhập không tồn tại trong hệ thống hoặc sai mật khẩu	Pass
3.	Kiểm thử chức năng Đăng nhập bằng Google (OAuth 2.0)	Xác minh chức năng xác thực bên thứ ba hoạt động đúng. Hiển thị “Trang chủ” Đăng nhập thất bại. Hiển thị thông báo lỗi: “Email không tồn tại” hoặc báo lỗi	
4.	Kiểm tra đăng xuất	Kiểm tra Đăng xuất khỏi hệ thống thành công, người dùng thoát ra khỏi hệ thống, kết thúc phiên đăng nhập. Hiển thị trang “Đăng nhập”	Pass
5.	Kiểm thử chức năng Quản lý thông tin cá nhân	Kiểm tra giới hạn kích thước file ảnh. Đảm bảo các trường thông tin quan trọng như tên người dùng không được để trống. Hiển thị thông báo “Thông tin tài khoản đã được cập nhật thành công”	Pass
6.	Kiểm thử Phân quyền người dùng	Kiểm tra chức năng hiển thị cho người dùng bình thường và quản trị viên	Pass
7.	Kiểm thử chức năng Quản lý bài	Kiểm tra người dùng có thể thêm bài viết vào hệ thống.	Pass

	viết	Kiểm tra quyền thao tác xóa bài cá nhân. Kiểm tra chức năng đảm bảo lượt yêu thích được ghi nhận. Kiểm tra chức năng thêm bình luận. Kiểm tra chức năng lưu bài viết, đảm bảo bài viết được lưu trong danh sách Bookmarks.	
8.	Kiểm thử chức năng thông báo	Kiểm tra người dùng có nhận được thông báo khi được một người dùng khác theo dõi hoặc yêu thích hoặc bình luận vào bài của mình.	Pass
9.	Kiểm thử chức năng Nhắn tin	Kiểm tra khả năng gửi – nhận tin nhắn. Kiểm tra giới hạn kích thước file ảnh. Kiểm tra chức năng tạo bình chọn trong tin nhắn. Kiểm tra chức năng tạo nhóm nhắn tin. Đảm bảo hiển thị đúng danh sách hội thoại.	Pass
10.	Kiểm thử chức năng hiển thị Bảng tin	Kiểm tra luồng hiển thị và lọc dữ liệu danh sách bài viết ở 2 chế độ (ForYou/ Following).	Pass
11.	Kiểm thử chức năng Tìm kiếm	Kiểm tra bộ lọc và tìm kiếm theo từ khóa.	Pass

#### 5.2.3.2. Kiểm thử chức năng cho Quản trị viên (Admin)

Bảng 5.6. Bảng tổng hợp ca kiểm thử cho Quản trị viên

STT	Mục tiêu kiểm thử	Ý nghĩa	Trạng thái
1.	Kiểm thử chức năng xóa bài viết của bất kì người dùng nào	Đảm bảo quyền hạn của quản trị viên được thực thi đúng	Pass
2.	Kiểm thử chức năng xóa bình luận của bất kì người dùng nào	Kiểm tra khả năng kiểm duyệt nội dung vi phạm	Pass
3.	Kiểm thử xóa tài khoản của người dùng bất kì	Kiểm tra chức năng quản lý tài khoản và quản lý người dùng	Pass

#### 5.2.3.3. Kiểm thử phi chức năng (Non-functional Testing)

Bảng 5.7. Bảng tổng hợp ca kiểm thử phi chức năng

STT	Mục tiêu kiểm thử	Ý nghĩa	Trạng thái
1.	Kiểm thử chức	Đảm bảo trang bảng tin hiển thị danh sách bài đăng	Pass

	năng tải trang chủ (Feed)	nhanh (<5s) khi có dưới 100 bài đăng	
2.	Kiểm thử độ mượt khi cuộn (scroll) và tải thêm bài viết	Đảm bảo hiệu ứng cuộn mượt, không giật lag khi tải thêm nội dung (lazy load)	Pass
3.	Kiểm thử hiệu năng chức năng gửi tin nhắn	Đánh giá thời gian phản hồi và khả năng truyền tải tin nhắn theo thời gian thực giữa hai hay nhóm người dùng	Not pass (Chưa tối ưu)
4.	Kiểm thử chức năng bảo mật quyền xóa bài đăng	Đảm bảo chỉ người tạo hoặc quản trị viên có quyền xóa bài viết	Pass
5.	Kiểm thử chức năng xóa bình luận	Đảm bảo chỉ người bình luận hoặc quản trị viên có thể xóa bình luận	Pass
6.	Kiểm thử an toàn dữ liệu đăng nhập	Kiểm tra thông tin mật khẩu không hiển thị khi chưa nhấn xem rõ và sử dụng hàm băm khi lưu mật khẩu người dùng vào cơ sở dữ liệu	Pass
7.	Kiểm thử độ ổn định của hệ thống	Đảm bảo hệ thống vẫn ổn định khi có $\geq 10$ người dùng truy cập đồng thời	Pass
8.	Kiểm thử tính thân thiện giao diện	Xác minh bố cục, font chữ, màu sắc hiển thị đồng nhất và dễ sử dụng	Pass

## KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 1. Kết luận

Báo cáo này đã trình bày toàn bộ quá trình phân tích, thiết kế, và cài đặt Hệ thống Mạng xã hội DevShareLite, từ các yêu cầu nghiệp vụ ban đầu đến việc triển khai một sản phẩm phần mềm hoàn chỉnh.

Dự án đã đạt được các mục tiêu cốt lõi đề ra:

1. Về chức năng: Hệ thống đã đáp ứng đầy đủ các yêu cầu nghiệp vụ của một mạng xã hội, bao gồm các chức năng cho User (đăng ký, đăng nhập, đăng bài, bình luận, theo dõi, nhắn tin DM, lưu bài viết) và các chức năng cho Admin (quản lý, xóa tài khoản vi phạm, xóa nội dung).
2. Về kỹ thuật: Dự án đã áp dụng thành công kiến trúc full-stack Next.js 15 (App Router). Việc sử dụng các công nghệ hiện đại như Server Components và Server Actions đã giúp tối ưu hóa hiệu suất, giảm tải JavaScript cho client và mang lại trải nghiệm người dùng mượt mà thông qua các kỹ thuật như Optimistic Updates.
3. Về tích hợp: Hệ thống đã tích hợp thành công với cơ sở dữ liệu PostgreSQL (qua Prisma ORM) để lưu trữ dữ liệu chính, Lucia Auth cho việc xác thực an toàn (bao gồm Google OAuth), và các dịch vụ bên thứ ba quan trọng là Stream API (cho tin nhắn real-time) và UploadThing (cho upload file).

Qua quá trình cài đặt và kiểm thử (Chương 5), hệ thống đã chứng minh được sự ổn định, các chức năng hoạt động chính xác theo kịch bản.

Tuy nhiên, do hạn chế về thời gian, dự án vẫn còn một số điểm có thể cải thiện, ví dụ như chức năng tìm kiếm còn ở mức cơ bản (sử dụng "full-text search" của Postgres) và hệ thống chưa có quy trình kiểm thử tự động (Unit Test) cho các logic nghiệp vụ.

Nhìn chung, dự án là một minh chứng thực tế cho khả năng và tính ưu việt của việc sử dụng Next.js 15 làm một framework full-stack để xây dựng các ứng dụng web phức tạp, hiệu suất cao.

## **2. Hướng phát triển**

Để hệ thống DevShareLite trở nên hoàn thiện, mạnh mẽ và có khả năng mở rộng hơn trong tương lai, nhóm đề xuất các hướng phát triển sau:

### 1.Nâng cấp chức năng Tìm kiếm (Search):

- Thay vì sử dụng "full-text search" cơ bản, sẽ tích hợp một dịch vụ tìm kiếm chuyên biệt như Elasticsearch hoặc Algolia. Điều này sẽ cải thiện tốc độ, độ chính xác của kết quả tìm kiếm và hỗ trợ các truy vấn phức tạp hơn.

### 2. Hoàn thiện Kiểm thử Tự động (Automated Testing):

- Viết Unit Test (sử dụng Jest hoặc Vitest) cho các Server Actions và các hàm logic quan trọng trong src/lib để đảm bảo code luôn chính xác khi có thay đổi.
- Xây dựng End-to-End (E2E) Test (sử dụng Cypress hoặc Playwright) để tự động hóa việc kiểm tra các luồng nghiệp vụ hoàn chỉnh (như đăng ký, đăng bài).

### 3.Phát triển Ứng dụng Di động (Mobile App):

- Xây dựng ứng dụng di động cho cả iOS và Android sử dụng React Native (hoặc Flutter) để cung cấp trải nghiệm tối ưu cho người dùng điện thoại, đồng thời có thể tái sử dụng một phần logic nghiệp vụ từ API (nếu cần).

### 4.Cải thiện Hệ thống Thông báo (Notification System):

- Nâng cấp hệ thống thông báo (notifications) hiện tại thành real-time (thời gian thực) bằng cách sử dụng WebSockets (ví dụ: qua Pusher hoặc một dịch vụ tương tự), thay vì chỉ cập nhật khi tải lại trang.

### 5.Tối ưu hóa CI/CD và Triển khai:

- Xây dựng một quy trình CI/CD (Continuous Integration/Continuous Deployment) hoàn chỉnh (ví dụ: sử dụng GitHub Actions) để tự động hóa việc kiểm thử và triển khai (deploy) dự án lên Vercel hoặc các nền tảng đám mây khác

# TÀI LIỆU THAM KHẢO

## I. Tài liệu Học thuật và Sách tham khảo

Martin, Robert C. (2018). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill Education.

## II. Tài liệu Công nghệ (Framework, CSDL và ORM)

Vercel. (2024). Next.js 15 Documentation. Truy cập tại <https://nextjs.org/docs>

Meta. (2024). React Documentation. Truy cập tại <https://react.dev/>

The PostgreSQL Global Development Group. (2024). PostgreSQL Documentation. Truy cập tại <https://www.postgresql.org/docs/>

Prisma. (2024). Prisma ORM Documentation. Truy cập tại <https://www.prisma.io/docs/>

## III. Tài liệu Thư viện và Dịch vụ API

Lucia. (2024). Lucia Auth Documentation. Truy cập tại <https://lucia-auth.com/>

Stream. (2024). Stream Chat API & SDK Documentation. Truy cập tại <https://getstream.io/chat/docs/>

UploadThing.(2024). UploadThing Documentation. Truy cập tại <https://docs.uploadthing.com/>

TanStack.(2024). TanStack Query Documentation. Truy cập tại <https://tanstack.com/query/latest/docs/>

Tailwind Labs.(2024). Tailwind CSS Documentation. Truy cập tại <https://tailwindcss.com/docs/>

Shadcn. (2024). Shadcn UI Documentation. Truy cập tại <https://ui.shadcn.com/docs>

Zod. (2024). Zod: TypeScript-first schema validation. Truy cập tại <https://zod.dev/>

Hook Form. (2024). React Hook Form Documentation. Truy cập tại <https://react-hook-form.com/>