

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP
NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:
NỀN TẢNG PHÁT HÀNH TRUYỆN TRANH SỐ
VỚI HỆ THỐNG BẢO VỆ NỘI DUNG VÀ TRẢI
NGHIỆM ĐỌC TỐI ƯU

Người hướng dẫn khoa học: **ThS. Nguyễn Tất Phú Cường**

Sinh viên thực hiện: **Nguyễn Hoàng Bảo**

Mã số sinh viên: **2151220280**

Lớp: **21CT6**

Đà Nẵng, 12/2025

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP
NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:
NỀN TẢNG PHÁT HÀNH TRUYỆN TRANH SỐ
VỚI HỆ THỐNG BẢO VỆ NỘI DUNG VÀ TRẢI
NGHIỆM ĐỌC TỐI ƯU

Người hướng dẫn khoa học: **ThS. Nguyễn Tất Phú Cường**

Sinh viên thực hiện: **Nguyễn Hoàng Bảo**

Mã số sinh viên: **2151220280**

Lớp: **21CT6**

Đà Nẵng, 12/2025

TÓM TẮT

Tên đề tài: Nền Tảng Phát Hành Truyện Tranh Số Với Hệ Thống Bảo Vệ Nội Dung Và Trải Nghiệm Đọc Tối Ưu.

Sinh viên thực hiện: Nguyễn Hoàng Bảo

MSSV: 2151220280 - Lớp: 21CT6

Đề án hướng đến việc xây dựng một nền tảng đọc truyện tranh trực tuyến hiện đại, mang lại trải nghiệm đọc mượt mà, tìm kiếm nhanh và tương tác cộng đồng phong phú. Hệ thống đồng thời hỗ trợ tác giả và nhóm dịch trong việc đăng tải, quản lý và biên tập chương truyện một cách thuận tiện. Nền tảng nổi bật với khả năng xử lý và mã hóa hình ảnh hiện đại: nhận diện và trích xuất chữ trong ảnh, hỗ trợ dịch tự động, tạo ảnh đã được làm sạch để người dịch dễ chỉnh sửa trước khi xuất bản, đồng thời bảo vệ nội dung tránh sao chép trái phép. Ngoài ra, hệ thống được thiết kế để hoạt động liên tục, cho phép cập nhật và triển khai phiên bản mới mà không gây gián đoạn. Khi có nâng cấp, một bản dự phòng sẽ được kích hoạt để kiểm tra ổn định trước khi chuyển đổi êm ái sang phiên bản mới, đảm bảo trải nghiệm đọc luôn liền mạch.

Nền tảng đọc truyện tranh này hứa hẹn trở thành công cụ hỗ trợ đắc lực cho cả độc giả lẫn nhóm sáng tác, giúp tối ưu hóa quy trình biên tập và phân phối nội dung. Hệ thống mang đến trải nghiệm tiện lợi, an toàn và nhất quán, góp phần xây dựng một môi trường truyện tranh trực tuyến thân thiện và chuyên nghiệp.

LỜI NÓI ĐẦU

Để hoàn thành được đồ án tốt nghiệp này, em xin gửi lời cảm ơn chân thành đến quý thầy cô Khoa Công nghệ Thông tin Trường Đại học Kiến Trúc Đà Nẵng đã tận tình giảng dạy và truyền đạt những kiến thức quý báu trong suốt thời gian học tập. Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc đến thầy **ThS. Nguyễn Tất Phú Cường**, người đã luôn hướng dẫn, chỉ bảo và hỗ trợ em nhiệt tình trong quá trình thực hiện đồ án.

Những kiến thức và kinh nghiệm mà thầy cô đã truyền dạy không chỉ là nền tảng quan trọng để em hoàn thành tốt khóa luận, mà còn là hành trang quý báu giúp em tự tin hơn khi bước vào công việc và cuộc sống sau này. Trong quá trình thực hiện đồ án, do kinh nghiệm thực tiễn còn hạn chế, em khó tránh khỏi những thiếu sót, kính mong nhận được sự góp ý của quý thầy cô để em có thể hoàn thiện bản thân hơn nữa.

Em xin trân trọng cảm ơn!

CAM ĐOAN

Em xin cam đoan rằng đồ án tốt nghiệp này được thực hiện bởi chính em dưới sự hướng dẫn của thầy ThS. Nguyễn Tất Phú Cường. Nội dung trong đồ án không sao chép của bất kỳ cá nhân hay tổ chức nào, những phần có tham khảo đều được ghi rõ nguồn và trích dẫn đúng quy định.

Toàn bộ dữ liệu, hình ảnh và tài liệu được sử dụng trong đồ án đều có xuất xứ minh bạch và được chọn lọc cẩn thận. Em xin chịu trách nhiệm hoàn toàn về tính trung thực của đồ án và chấp nhận mọi hình thức xử lý nếu có vi phạm xảy ra.

Em xin chịu trách nhiệm hoàn toàn đối với lời cam đoan này.

Sinh viên thực hiện

Nguyễn Hoàng Bảo

MỤC LỤC

Tóm tắt	
Lời nói đầu	i
Cam đoan	ii
Mục lục	iii
Danh mục bảng	v
Danh mục hình ảnh	viii
Danh mục từ viết tắt	xiii

Trang

MỞ ĐẦU.....	10
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT.....	14
1.1. Giới thiệu về JavaScript	14
1.2. Giới thiệu về NextJS.....	16
1.3. Giới thiệu về NestJS	20
1.4. Giới thiệu về PostgreSQL.....	21
1.5. Giới thiệu về Flask	23
1.6. Giới thiệu về OpenCV.....	25
1.7. Giới thiệu về PyTorch.....	28
CHƯƠNG 2: THIẾT KẾ HỆ THỐNG	31
2.1. Phân tích hệ thống.....	31
2.1.1. Tổng quan bài toán	31
2.1.2. Các thành phần chính.....	31
2.1.3. Thách thức	32
2.1.4. Luồng dữ liệu chính.....	33
2.2. Mô hình hóa yêu cầu	34
2.2.1. Chức năng xác thực	36
2.2.2. Chức năng cơ bản người dùng.....	40
2.2.3. Chức năng xem chi tiết chương	45
2.2.4. Chức năng quản lý lịch sử đọc	51
2.2.5. Chức năng báo cáo chương.....	54

2.3.6. Chức năng bình luận	57
2.2.7. Chức năng theo dõi truyện.....	61
2.2.8. Chức năng quản lí cửa hàng vật phẩm.....	66
2.3.9. Chức năng nhắn tin toàn hệ thống	70
2.2.10. Chức năng tìm kiếm truyện	73
2.2.11. Chức năng quản lí nhóm	76
2.2.12. Chức năng tạo truyện.....	81
2.2.13. Chức năng quản lí ảnh truyện	85
2.2.14. Chức năng quản lí thành viên truyện.....	89
2.2.15. Chức năng tạo chương	93
2.2.16. Chức năng dịch truyện với AI	98
2.2.17. Chức năng quản lí hình ảnh	102
2.2.18. Chức năng quản lí admin	105
2.4. Thiết kế cơ sở dữ liệu	109
2.4.1. Sơ đồ lớp.....	109
2.4.2. Sơ đồ hoạt động	111
2.5. Danh mục các bảng	112
2.6. Thiết kế giao diện	121
CHƯƠNG 3: TRIỂN KHAI HỆ THỐNG	139
3.1. Framework và các thư viện của các services	139
3.2. Xử lí tạo chương kết hợp AI dịch truyện.....	145
3.2.1. Vấn đề	145
3.2.2. Cách giải quyết	146
3.2.3. Module code	148
3.3. Xử lí bảo mật hiển thị mã hóa ảnh	152
3.3.1. Vấn đề	152
3.3.2. Cách giải quyết	152
3.3.3. Module code	154
3.4. Kiểm thử.....	155
KẾT LUẬN	160
TÀI LIỆU THAM KHẢO.....	162

DANH MỤC BẢNG

Bảng 1.1 Phân tích sự khác nhau giữa PostgreSQL và MySQL.....	22
Bảng 1.2 Phân tích tính năng của Pytorch.....	28
Bảng 1.3 Thành phần chính của PyTorch	29
Bảng 2.4 mô tả Usecase chức năng xác thực	36
Bảng 2.5 Dòng chính Usecase chức năng xác thực.....	37
Bảng 2.6 Mô tả Usecase chức năng cơ bản người dùng	40
Bảng 2.7 Dòng chính Usecase chức năng cơ bản người dùng	41
Bảng 2.8 Mô tả Usecase chức năng xem chi tiết chương.....	46
Bảng 2.9 Dòng chính Usecase chức năng xem chi tiết chương	47
Bảng 2.10 Mô tả Usecase quản lí lịch sử đọc.....	52
Bảng 2.11 Dòng chính Usecase quản lí lịch sử đọc	52
Bảng 2.12 Mô tả Usecase báo cáo chương.....	55
Bảng 2.13 Dòng chính Usecase báo cáo chương	55
Bảng 2.14 Mô tả Usecase chức năng bình luận.....	58
Bảng 2.15 Dòng chính Usecase chức năng bình luận	58
Bảng 2.16 Mô tả Usecase chức năng theo dõi truyện	62
Bảng 2.17 Dòng chính Usecase chức năng theo dõi truyện.....	62
Bảng 2.18 Mô tả Usecase chức năng quản lí cửa hàng vật phẩm	67
Bảng 2.19 Dòng chính Usecase chức năng quản lí cửa hàng vật phẩm.....	67
Bảng 2.20 Mô tả usecase chức năng nhắn tin toàn hệ thống.....	70
Bảng 2.21 Dòng chính Usecase chức năng nhắn tin toàn hệ thống	71
Bảng 2.22 Mô tả Usecase chức năng tìm kiếm truyện.....	74
Bảng 2.23 Dòng chính Usecase chức năng tìm kiếm truyện.....	74
Bảng 2.24 Mô tả Usecase chức năng quản lí nhóm	77
Bảng 2.25 Dòng chính Usecase chức năng quản lí nhóm	78
Bảng 2.26 Mô tả Usecase chức năng tạo truyện	82
Bảng 2.27 Dòng chính Usecase chức năng tạo truyện	82

Bảng 2.28 Mô tả Usecase chức năng quản lý ảnh truyện.....	86
Bảng 2.29 Dòng chính Usecase chức năng quản lý ảnh truyện	86
Bảng 2.30 Mô tả Usecase chức năng quản lý thành viên truyện	90
Bảng 2.31 Dòng chính Usecase chức năng quản lý thành viên truyện	90
Bảng 2.32 Mô tả Usecase chức năng tạo chương.....	94
Bảng 2.33 Dòng chính Usecase chức năng tạo chương	95
Bảng 2.34 Mô tả Usecase chức năng dịch truyện với AI.....	99
Bảng 2.35 Dòng chính Usecase chức năng dịch truyện với AI.....	99
Bảng 2.36 Mô tả Usecase chức năng quản lý hình ảnh.....	103
Bảng 2.37 Dòng chính Usecase chức năng quản lý hình ảnh	103
Bảng 2.38 Mô tả Usecase chức năng quản lý admin.....	106
Bảng 2.39 Dòng chính Usecase chức năng quản lý admin	106
Bảng 2.40 Các bảng trong database	112
Bảng 2.41 Bảng user.....	113
Bảng 2.42 Bảng user_item	113
Bảng 2.43 Bảng item	114
Bảng 2.44 Bảng user_item	114
Bảng 2.45 Bảng team	114
Bảng 2.46 Bảng book_contributor	115
Bảng 2.47 Bảng book	115
Bảng 2.48 Bảng chapter	116
Bảng 2.49 Bảng server	117
Bảng 2.50 Bảng protected_image.....	117
Bảng 2.51 Bảng comment	117
Bảng 2.52 Bảng user_book_follow	118
Bảng 2.53 Bảng chat.....	118
Bảng 2.54 Bảng user_view.....	118
Bảng 2.55 Bảng report_chapter.....	119
Bảng 2.56 Bảng meta	119
Bảng 2.57 Bảng image	120

Bảng 2.58 Framework và các thư viện của các services	139
--	-----

DANH MỤC HÌNH ẢNH

Hình 1.1 Ví dụ về thao tác với DOM	14
Hình 1.2 Ví dụ về thao tác với DOM	14
Hình 1.3 Xây dựng API đơn giản	15
Hình 1.4 Xây dựng một component hiển thị chi tiết sản phẩm.....	17
Hình 1.5 Kiểm tra phiên bản hiện tại của python và pip.....	24
Hình 1.6 Cài đặt flask.....	24
Hình 1.7 Khởi tạo API đơn giản in ra ‘Hello, World!’	25
Hình 1.8 Nhận diện đối tượng qua ảnh	26
Hình 1.9 Mô hình hóa đối tượng	27
Hình 1.10 Tính tensor cơ bản	29
Hình 1.11 Mạng nơ-ron đơn giản	30
Hình 2.12 Usecase tổng quát	35
Hình 2.13 Sơ đồ usecase chức năng xác thực	36
Hình 2.14 Sơ đồ tuần tự chức năng xác thực	38
Hình 2.15 Sơ đồ hoạt động chức năng xác thực.....	39
Hình 2.16 Sơ đồ usecase chức năng cơ bản người dùng.....	40
Hình 2.17 Sơ đồ tuần tự chức năng cơ bản người dùng.....	43
Hình 2.18 Sơ đồ hoạt động chức năng cơ bản người dùng	44
Hình 2.19 Sơ đồ usecase chức năng xem chi tiết chương	45
Hình 2.20 Sơ đồ tuần tự chức năng xem chi tiết chương.....	49
Hình 2.21 Sơ đồ hoạt động chức năng xem chi tiết chương	50
Hình 2.22 Sơ đồ usecase quản lý lịch sử đọc	51
Hình 2.23 Sơ đồ tuần tự quản lý lịch sử đọc	53
Hình 2.24 Sơ đồ hoạt động chức năng quản lý lịch sử đọc.....	54
Hình 2.25 Sơ đồ usecase báo cáo chương	54
Hình 2.26 Sơ đồ tuần tự báo cáo chương	56
Hình 2.27 Sơ đồ hoạt động chức năng báo cáo chương.....	57

Hình 2.28 Sơ đồ usecase chức năng bình luận	57
Hình 2.29 Sơ đồ tuần tự chức năng bình luận	60
Hình 2.30 Sơ đồ hoạt động chức năng bình luận	61
Hình 2.31 Sơ đồ usecase chức năng theo dõi truyện.....	61
Hình 2.32 Sơ đồ tuần tự chức năng theo dõi truyện.....	64
Hình 2.33 Sơ đồ hoạt động chức năng theo dõi truyện	65
Hình 2.34 Sơ đồ usecase chức năng quản lý cửa hàng vật phẩm.....	66
Hình 2.35 Sơ đồ tuần tự chức năng cửa hàng vật phẩm.....	68
Hình 2.36 Sơ đồ hoạt động chức năng cửa hàng vật phẩm	69
Hình 2.37 Sơ đồ usecase chức năng nhắn tin toàn hệ thống	70
Hình 2.38 Sơ đồ tuần tự chức năng nhắn tin toàn hệ thống	72
Hình 2.39 Sơ đồ hoạt động chức năng nhắn tin toàn hệ thống	73
Hình 2.40 Sơ đồ usecase chức năng tìm kiếm truyện	73
Hình 2.41 Sơ đồ tuần tự chức năng tìm kiếm truyện	75
Hình 2.42 Sơ đồ hoạt động chức năng tìm kiếm truyện.....	76
Hình 2.43 Sơ đồ usecase chức năng quản lý nhóm.....	76
Hình 2.44 Sơ đồ tuần tự chức năng quản lý nhóm.....	79
Hình 2.45 Sơ đồ hoạt động chức năng quản lý nhóm	80
Hình 2.46 Sơ đồ usecase chức năng tạo truyện.....	81
Hình 2.47 Sơ đồ tuần tự chức năng tạo truyện.....	83
Hình 2.48 Sơ đồ hoạt động chức năng tạo truyện	84
Hình 2.49 Sơ đồ usecase chức năng quản lý ảnh truyện.....	85
Hình 2.50 Sơ đồ tuần tự chức năng quản lý ảnh truyện.....	88
Hình 2.51 Sơ đồ hoạt động chức năng quản lý ảnh truyện	88
Hình 2.52 Sơ đồ usecase chức năng quản lý thành viên truyện.....	89
Hình 2.53 Sơ đồ tuần tự chức năng quản lý thành viên truyện.....	91
Hình 2.54 Sơ đồ hoạt động chức năng quản lý thành viên truyện	92
Hình 2.55 Sơ đồ usecase chức năng tạo chương.....	93
Hình 2.56 Sơ đồ tuần tự chức năng tạo chương	96
Hình 2.57 Sơ đồ hoạt động chức năng tạo chương	97

Hình 2.58 Sơ đồ usecase chức năng dịch truyện với AI	98
Hình 2.59 Sơ đồ tuần tự chức năng dịch truyện với AI	100
Hình 2.60 Sơ đồ hoạt động chức năng dịch truyện với AI.....	101
Hình 2.61 Sơ đồ usecase chức năng quản lý hình ảnh	102
Hình 2.62 Sơ đồ tuần tự chức năng quản lý hình ảnh	104
Hình 2.63 Sơ đồ hoạt động chức năng quản lý hình ảnh	105
Hình 2.64 Sơ đồ usecase chức năng	105
Hình 2.65 Sơ đồ tuần tự chức năng quản lý admin	107
Hình 2.66 Sơ đồ usecase chức năng quản lý admin	108
Hình 2.67 Sơ đồ lớp.....	109
Hình 2.68 Mô hình hóa thực thể liên kết.....	110
Hình 2.69 Sơ đồ hoạt động	111
Hình 2.70 Màn hình trang chủ.....	121
Hình 2.71 Màn hình đăng nhập	122
Hình 2.72 Màn hình danh sách truyện mới	122
Hình 2.73 Màn hình danh sách truyện nổi bật	123
Hình 2.74 Màn hình danh sách truyện theo dõi	123
Hình 2.75 Màn hình danh sách truyện nhiều theo dõi.....	123
Hình 2.76 Màn hình chi tiết truyện	124
Hình 2.77 Màn hình chi tiết chương.....	125
Hình 2.78 Màn hình tìm kiếm cơ bản.....	125
Hình 2.79 Màn hình tìm kiếm nâng cao	126
Hình 2.80 Màn hình chi tiết thể loại.....	126
Hình 2.81 Màn hình danh sách truyện cho SEO từ khóa	127
Hình 2.82 Màn hình thông tin người dùng.....	127
Hình 2.83 Màn hình điểm danh	128
Hình 2.84 Màn hình đổi ảnh đại diện	128
Hình 2.85 Màn hình thông tin tài khoản.....	129
Hình 2.86 Màn hình lịch sử đọc truyện.....	129
Hình 2.87 Màn hình trang bị vật phẩm.....	130

Hình 2.88 Màn hình cửa hàng phụ kiện	130
Hình 2.89 Màn hình cửa hàng viên ảnh đại diện.....	131
Hình 2.90 Thông báo mời làm thành viên nhóm dịch.....	131
Hình 2.91 Màn hình tạo nhóm dịch truyện	132
Hình 2.92 Màn hình cập nhật thông tin nhóm dịch.....	132
Hình 2.93 Màn hình danh sách nhóm dịch.....	133
Hình 2.94 Màn hình tạo truyện.....	133
Hình 2.95 Màn hình danh sách truyện.....	134
Hình 2.96 Màn hình chỉnh sửa truyện	134
Hình 2.97 Màn hình hiển thị danh sách chương.....	135
Hình 2.98 Màn hình tạo chương.....	135
Hình 2.99 Màn hình hiển thị báo cáo chương truyện lỗi.....	136
Hình 2.100 Màn hình quản lí người dùng	136
Hình 2.101 Màn hình quản lí nhóm.....	137
Hình 2.102 Quản lí truyện đã xóa	137
Hình 2.103 Quản lí chương đã xóa.....	138
Hình 2.104 Màn hình chi tiết nhóm dịch.....	138
Hình 2.105 Api bắt xử lí điều chỉnh kích ảnh	148
Hình 3.106 Điều chỉnh kích thước ảnh nếu quá lớn để tăng tốc xử lý.....	148
Hình 3.107 API bắt xử lí dịch ảnh.....	149
Hình 3.108 Xử lí dịch ảnh	149
Hình 3.109 API xử lí dịch nội dung	150
Hình 3.110 API xử lí dịch chi tiết ảnh.....	151
Hình 3.111 Mã hóa ảnh với Rust.....	154
Hình 3.112 Kịch bản kiểm thử lấy người dùng đăng nhập	155
Hình 3.113 Kịch bản kiểm thử lấy người dùng đăng nhập với token	156
Hình 3.114 Kịch bản kiểm thử đăng nhập với google	156
Hình 3.115 Kịch bản kiểm thử chuyển quyền hạn người dùng.....	157
Hình 3.116 Kịch bản kiểm thử đánh dấu truyện hay.....	158
Hình 3.117 Kịch bản kiểm thử đăng tải ảnh lên đám mây tạm.....	159

Hình 2.118 Kết quả kiểm thử trong module auth, user, book, chapter	159
Hình 2.119 Kết quả kiểm thử chịu tải 10 connection liên tục	159

DANH MỤC TỪ VIẾT TẮT

Chữ viết tắt	Tên đầy đủ	Tên đầy đủ Tiếng Việt
AI	Artificial Intelligence	Trí tuệ nhân tạo
ML	Machine Learning	Học máy
DL	Deep Learning	Học sâu
GPU	Graphics Processing Unit	Bộ xử lý đồ họa
CPU	Central Processing Unit	Bộ xử lý trung tâm
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
CV	Computer Vision	Thị giác máy tính
API	Application Programming Interface	Giao diện lập trình ứng dụng
PyTorch	(Không phải viết tắt)	Thư viện học sâu nguồn mở
CUDA	Compute Unified Device Architecture	Kiến trúc thiết bị tính toán thống nhất (NVIDIA)
DNN	Deep Neural Network	Mạng nơ-ron sâu
ReLU	Rectified Linear Unit	Hàm kích hoạt ReLU
SGD	Stochastic Gradient Descent	Thuật toán giảm dần theo ngẫu nhiên
RNN	Recurrent Neural Network	Mạng nơ-ron hồi quy
CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
LLM	Large Language Model	Mô hình ngôn ngữ lớn
FAIR	Facebook AI Research	Nhóm nghiên cứu AI của Meta

MỞ ĐẦU

1. Giới thiệu

Ở ngoài nước, nhiều nền tảng đọc truyện đã ra đời và đạt được thành công đáng kể như MangaDex, Webtoon hay Bato. Các hệ thống này không chỉ cung cấp trải nghiệm đọc mượt mà, mà còn tích hợp thêm các chức năng như bình luận, gợi ý truyện theo sở thích, hoặc cơ chế bảo vệ bản quyền bằng mã hóa hình ảnh và DRM. Một số nhóm nghiên cứu trong lĩnh vực công nghệ nội dung số cũng đã khai thác WebAssembly, OCR và dịch máy để hỗ trợ xử lý và phân phối nội dung nhanh chóng hơn. Điều này cho thấy, xu hướng quốc tế đã dịch chuyển từ nền tảng hiển thị đơn thuần sang các nền tảng đa chức năng, có tính công nghệ cao.

Ở trong nước, lĩnh vực truyện tranh trực tuyến chủ yếu phát triển ở dạng website và ứng dụng di động. Các sản phẩm này tập trung vào việc cung cấp kho truyện đa dạng, nhưng chưa có nhiều công trình nghiên cứu chuyên sâu về xử lý ảnh, bảo mật nội dung số hay tự động hóa trong dịch thuật. Một số nghiên cứu gần đây về công nghệ web, cơ sở dữ liệu và trí tuệ nhân tạo cho thấy tiềm năng ứng dụng trong lĩnh vực này, nhưng vẫn còn thiếu sự kết nối để hình thành một hệ thống đồng bộ.

Từ những kết quả trong và ngoài nước, có thể thấy rằng định hướng nghiên cứu cho đề tài này tập trung vào việc khai thác công nghệ hiện đại để xây dựng nền tảng đọc và quản lý truyện tranh. Thay vì chỉ là một trang web đơn giản, hệ thống hướng tới việc giải quyết các thách thức về bảo mật, hiệu năng và hỗ trợ cộng đồng sáng tạo nội dung. Đây cũng chính là điểm khác biệt và mang tính sáng tạo của đề tài, góp phần bổ sung cho khoảng trống nghiên cứu hiện nay trong nước.

2. Lý do chọn đề tài

Trong những năm gần đây, ngành công nghiệp truyện tranh số tại Việt Nam và trên thế giới đã có sự phát triển vượt bậc. Theo thống kê, thị trường truyện tranh kỹ thuật số toàn cầu đạt giá trị 8.5 tỷ USD năm 2023 và dự kiến tăng trưởng 12.3% hàng năm. Tại Việt Nam, lượng người dùng đọc truyện tranh trực tuyến đã tăng đáng kể, đặc biệt trong và sau đại dịch COVID-19, khi nhu cầu giải trí số ngày càng cao. Tuy nhiên, các nền tảng đọc truyện tranh hiện tại vẫn còn nhiều hạn chế về mặt kỹ thuật và trải nghiệm người dùng. Đặc biệt, vấn đề bảo vệ bản quyền và nội dung sáng tạo chưa được quan tâm đúng mức, dẫn đến tình trạng sao chép, chia sẻ trái phép tràn lan. Bên cạnh đó, quy trình quản lý và xuất bản nội dung truyện tranh còn thủ công, tốn nhiều thời gian và công sức. Xuất phát từ thực trạng trên, việc xây dựng một nền tảng đọc và quản lý truyện tranh với các giải pháp công nghệ tiên tiến là cần thiết và có ý nghĩa thực tiễn cao.

Đề tài có tiềm năng ứng dụng rộng rãi và khả năng phát triển lâu dài:

- Hệ thống có thể được triển khai cho các đơn vị xuất bản truyện tranh, các nhà sáng tạo nội dung độc lập, hoặc các cộng đồng fan dịch thuật.
- Kiến trúc hệ thống cho phép mở rộng sang các định dạng nội dung khác như light novel, webtoon, manga với những điều chỉnh phù hợp.
- Các công nghệ và giải pháp kỹ thuật được nghiên cứu có thể áp dụng cho các lĩnh vực khác như bảo vệ tài liệu số, quản lý nội dung đa phương tiện.
- Mô hình kinh doanh thông qua hệ thống điểm thưởng và marketplace tạo ra khả năng sinh lời bền vững, đảm bảo tính khả thi của dự án trong thực tế.

3. Mục tiêu

Đề tài tập trung nghiên cứu và xây dựng một nền tảng đọc và quản lý truyện tranh trực tuyến có tính ứng dụng thực tiễn, vừa phục vụ nhu cầu tiếp cận nội dung số của người dùng, vừa cung cấp công cụ hỗ trợ sáng tạo và quản lý cho cộng đồng tác giả, dịch giả. Hướng nghiên cứu không chỉ dừng ở việc phát triển một website thông thường mà còn khai thác các công nghệ tiên tiến để giải quyết những vấn đề cốt lõi như: tối ưu hiệu năng, bảo mật nội dung số, quản lý cộng đồng đa cấp và hỗ trợ xử lý dữ liệu hình ảnh.

Các mục tiêu cụ thể của đề tài bao gồm:

- Xây dựng nền tảng quản lý và đọc truyện tranh trực tuyến với các chức năng cơ bản như đăng ký, đăng nhập, đọc truyện, tìm kiếm và tương tác cộng đồng (chat, bình luận).
- Nghiên cứu và triển khai hệ thống phân quyền đa cấp (Admin, Editor, Creator, Guest) và cơ chế quản lý theo nhóm (team), nhằm đảm bảo tính minh bạch và linh hoạt trong việc phân công, kiểm soát và sáng tạo nội dung.
- Ứng dụng công nghệ xử lý ảnh hiện đại (Rust + WebAssembly) để bảo mật và tối ưu hiển thị nội dung truyện, đồng thời nghiên cứu cơ chế DRM (Digital Rights Management) nhằm hạn chế tình trạng sao chép và vi phạm bản quyền.
- Tích hợp các giải pháp trí tuệ nhân tạo (OCR, dịch tự động, xử lý ngôn ngữ tự nhiên) để hỗ trợ quá trình biên dịch truyện tranh, giúp Creator có thể dễ dàng thao tác với nội dung gốc và rút ngắn thời gian sản xuất bản dịch.
- Phát triển cơ sở hạ tầng triển khai hiện đại với đảm bảo tính ổn định, mở rộng và dễ dàng quản lý hệ thống trong môi trường thực tế.
- Xây dựng cơ chế khuyến khích và gắn kết người dùng thông qua hệ thống điểm thưởng ảo (Mirucoins), gamification và các yếu tố tương tác cộng đồng.

Mục tiêu cuối cùng của đề tài là hình thành một mô hình thực tế có thể ứng dụng rộng rãi trong lĩnh vực xuất bản và phân phối truyện tranh trực tuyến. Đồng thời, đề tài cũng góp phần khẳng định vai trò của công nghệ thông tin trong việc nâng cao trải nghiệm người dùng, hỗ trợ cộng đồng sáng tạo nội dung, và hướng tới giải quyết những thách thức về bản quyền, hiệu năng và sự phát triển bền vững của nội dung số tại Việt Nam.

4. Đối tượng, phạm vi nghiên cứu

4.1. Đối tượng nghiên cứu

Hệ thống đọc và quản lý truyện tranh trực tuyến: Nghiên cứu các thành phần cốt lõi để xây dựng một nền tảng cho phép người dùng có thể đọc truyện tranh trực tiếp trên web, với giao diện hiện đại và trải nghiệm mượt mà. Đối tượng này bao gồm các tính năng như quản lý truyện, hiển thị chương, hỗ trợ nhiều máy chủ ảnh, cũng như các chức năng tìm kiếm và phân loại.

Cơ chế phân quyền và quản lý nhóm (team): Nghiên cứu mô hình phân quyền người dùng đa cấp, bao gồm các vai trò Admin, Editor, Creator và Guest. Mỗi vai trò được xác định quyền hạn riêng, phù hợp với nhu cầu quản lý và sáng tạo nội dung. Đồng thời, nghiên cứu cách thức tổ chức nhóm (team) với các vai trò khác nhau để đảm bảo sự phối hợp hiệu quả trong việc biên tập và xuất bản truyện.

Công nghệ xử lý ảnh và bảo mật nội dung: Tìm hiểu và áp dụng Rust kết hợp WebAssembly nhằm tối ưu hóa tốc độ xử lý ảnh và đảm bảo tính bảo mật cho nội dung truyện. Đối tượng nghiên cứu còn bao gồm cơ chế DRM (Digital Rights Management), trong đó áp dụng các kỹ thuật mã hóa, chèn watermark và xáo trộn layer ảnh để ngăn chặn hành vi sao chép trái phép.

Ứng dụng trí tuệ nhân tạo hỗ trợ dịch thuật: Nghiên cứu khả năng tích hợp OCR (Optical Character Recognition) để nhận diện chữ trong ảnh truyện, kết hợp với dịch máy tự động nhằm hỗ trợ quá trình dịch thuật. Công cụ AI như PaddleOCR, OpenCV và ChatGPT được xem như đối tượng nghiên cứu quan trọng để rút ngắn thời gian biên dịch và giảm khối lượng công việc thủ công của Creator.

Cơ chế tương tác cộng đồng và gamification: Đối tượng nghiên cứu bao gồm các hình thức giao tiếp trực tuyến như chat toàn server, bình luận nhiều cấp, cùng với hệ thống Mirucoins để tạo động lực cho người dùng tham gia. Việc áp dụng gamification nhằm nâng cao trải nghiệm và tăng tính gắn kết trong cộng đồng người đọc.

4.2. Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài được xác định cụ thể như sau:

Về công nghệ: Đề tài tập trung vào việc áp dụng các công nghệ web hiện đại như Next.js cho frontend và NestJS cho backend, kết hợp cơ sở dữ liệu MySQL với Prisma để quản lý dữ liệu. Docker, Nginx phục vụ triển khai và vận hành hệ thống. Trong phạm vi đề tài, nghiên cứu không mở rộng sang các công nghệ mobile-native hoặc các framework ngoài định hướng đã chọn.

Về chức năng hệ thống: Hệ thống được xây dựng xoay quanh các chức năng chính: đăng nhập/đăng ký, đọc truyện, tìm kiếm, đăng truyện, quản lý nhóm dịch, chat và bình luận. Một số chức năng nâng cao như hệ thống Mirucoins, gamification, upload ảnh kèm watermark, dịch tự động thông qua OCR và AI cũng nằm trong phạm vi nghiên cứu. Các

chức năng thương mại hóa (thu phí, thanh toán online, quảng cáo nâng cao) không nằm trong phạm vi của đề tài.

Về bảo mật và xử lý ảnh: Đề tài tập trung vào việc nghiên cứu cơ chế DRM cơ bản cho ảnh truyện, bao gồm mã hóa và xáo trộn layer, thay vì triển khai toàn diện các chuẩn DRM thương mại phức tạp. Các thuật toán nâng cao về mã hóa dữ liệu chỉ được tham khảo ở mức độ ứng dụng thử nghiệm.

Về trí tuệ nhân tạo: Nghiên cứu dừng lại ở mức tích hợp và ứng dụng các công cụ OCR và dịch máy có sẵn (PaddleOCR, OpenCV, ChatGPT API). Đề tài không đào sâu vào việc xây dựng hoặc huấn luyện mô hình AI mới. Việc dịch thuật tự động chỉ dừng ở mức hỗ trợ Creator, chưa thể thay thế hoàn toàn công việc biên dịch thủ công.

Về hạ tầng triển khai: Phạm vi triển khai được giới hạn trong môi trường thử nghiệm trên VPS cá nhân, có tích hợp Docker, Nginx. Đề tài chưa mở rộng sang hệ thống phân tán lớn, CDN toàn cầu hay các dịch vụ cloud thương mại R2 Cloudflare.

Về phạm vi ứng dụng: Đề tài được định hướng để phục vụ cho việc nghiên cứu và minh chứng khả năng ứng dụng công nghệ thông tin trong lĩnh vực quản lý và phân phối truyện tranh trực tuyến. Ứng dụng thực tế tập trung vào truyện tranh, chưa mở rộng sang các loại nội dung số khác như sách điện tử, báo chí hoặc phim ảnh.

Về giới hạn:

- Chỉ xây dựng nền tảng web, chưa phát triển ứng dụng di động.
- Hệ thống thử nghiệm ở quy mô nhỏ, chưa đáp ứng nhu cầu hàng triệu người dùng.
- Một số tính năng mới như dịch tự động và gamification mới dừng ở mức nguyên mẫu (prototype).

Đối tượng nghiên cứu của đề tài bao gồm từ hệ thống web đọc truyện, cơ chế phân quyền, quản lý nhóm, xử lý ảnh và bảo mật nội dung, cho đến trí tuệ nhân tạo hỗ trợ dịch thuật. Phạm vi nghiên cứu tập trung vào việc xây dựng một nền tảng thử nghiệm ứng dụng công nghệ hiện đại trong lĩnh vực truyện tranh trực tuyến, với mức độ triển khai vừa đủ để chứng minh tính khả thi và định hướng phát triển. Các yếu tố thương mại hóa, mở rộng quy mô lớn hoặc triển khai đa nền tảng không nằm trong phạm vi của đề tài này.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Giới thiệu về JavaScript

JavaScript (JS) là một ngôn ngữ lập trình bậc cao, chủ yếu được sử dụng để tăng cường tính tương tác cho các website, đóng vai trò thiết yếu trong việc tạo ra những trải nghiệm hấp dẫn cho người dùng. JavaScript cho phép thực thi mã lệnh ở phía client, tức là ngay trên trình duyệt thay vì ở máy chủ từ xa (client-side).

Sau sự ra đời của Nodejs, JavaScript đã mở rộng khả năng của mình từ việc chỉ là một ngôn ngữ lập trình chạy trên trình duyệt, trở thành một công cụ mạnh mẽ để phát triển các ứng dụng server-side, giúp các Developer sử dụng một ngôn ngữ duy nhất cho toàn bộ ứng dụng (frontend + backend).

Client-side JavaScript

Client-side JavaScript là mã JavaScript được thực thi trực tiếp trên trình duyệt của người dùng (Google Chrome, Microsoft Edge, Fire Fox), thường được sử dụng để tạo ra các tính năng tương tác và động trên trang web. Dưới đây là một số tính năng của client-side JavaScript:

Thao tác với DOM (Document Object Model): JavaScript Thao tác với DOM của một trang web, làm cho nội dung thay đổi tự động dựa theo thao tác của người dùng mà không cần tải lại trang.

Ví dụ:

```
1 // Thay đổi nội dung của một thẻ <div> khi người dùng nhấp vào nút
2 document.getElementById("myButton").addEventListener("click", function() {
3     document.getElementById("myDiv").innerHTML = "Nội dung đã được thay đổi!";
4 });
```

Hình 1.1 Ví dụ về thao tác với DOM

Ví dụ:

```
1 // Lắng nghe và phản hồi các sự kiện người dùng như nhấp chuột, nhập liệu, di chuyển chuột, và nhiều sự kiện khác.
2 document.getElementById("myInput").addEventListener("input", function() {
3     alert("Bạn vừa nhập: " + this.value);
4 });
```

Hình 1.2 Ví dụ về thao tác với DOM

Server-side JavaScript

Server-side JavaScript là các đoạn mã JavaScript được thực thi trên máy chủ, giúp quản lý các yêu cầu từ client, tương tác với cơ sở dữ liệu, ... Dưới đây là một số tính năng của server-side JavaScript:

Xử lý yêu cầu HTTP: Xử lý các yêu cầu HTTP từ client, chẳng hạn như yêu cầu GET hoặc POST, và trả về phản hồi phù hợp (thường là HTML, JSON).

```
1 // Xử lý các yêu cầu GET từ client, trả về phản hồi 'Hello World'
2 const http = require("http");
3 const server = http.createServer((req, res) => {
4     res.statusCode = 200;
5     res.setHeader("Content-Type", "text/plain");
6     res.end("Hello World");
7 });
8 server.listen(3000, () => {
9     console.log("Server running at http://localhost:3000");
10 });
```

Hình 1.3 Xây dựng API đơn giản

Trong khảo sát hàng năm của Stack Overflow, JavaScript tiếp tục là ngôn ngữ lập trình phổ biến nhất trong suốt 11 năm liên tiếp. Đây là khảo sát lớn nhất trên thế giới trong lĩnh vực lập trình, với sự tham gia của 87,585 developers trên toàn cầu.

Báo cáo GitHub Octoverse năm 2023 cũng chỉ ra rằng JavaScript là ngôn ngữ lập trình phổ biến nhất trên nền tảng này trong suốt 10 năm. JavaScript luôn đứng đầu về số lượng repository và đóng góp của cộng đồng.

Sau đây là những nguyên nhân chính khiến JavaScript thống trị các bảng xếp hạng trong nhiều năm:

- Cú Pháp Đơn Giản và Dễ Học: JavaScript là một ngôn ngữ phổ biến, dễ học, chúng ta có thể kiểm tra code của mình ngay trong trình duyệt, ngôn ngữ này thân thiện với người mới bắt đầu hơn nhiều so với các ngôn ngữ lập trình khác.
- Hỗ Trợ Rộng Rãi Từ Cộng Đồng: JavaScript có một cộng đồng lập trình viên lớn mạnh và sôi động, cung cấp vô số tài nguyên học tập, thư viện mã nguồn mở, và các công cụ hỗ trợ.
- Khả năng tạo ra giao diện phong phú: JavaScript được các lập trình viên ưa chuộng nhờ khả năng tạo ra các giao diện phong phú và hấp dẫn. JavaScript tích hợp chặt chẽ với cấu trúc trang web (DOM), cho phép thay đổi nội dung, cấu trúc và kiểu dáng của trang web theo thời gian thực. Kết hợp với AJAX, JavaScript có thể giao tiếp với máy chủ và cập nhật trang web mà không cần tải lại toàn bộ. Thêm vào đó, các thư viện và framework như React, Angular, và Vue.js giúp tăng tốc quá trình phát triển.

- **Lập Trình Toàn Diện Với Nodejs:** JavaScript có thể phát triển các ứng dụng web hoàn chỉnh, quản lý cả frontend và backend chỉ với một ngôn ngữ. Với sự ra đời của Nodejs, JavaScript đã trở thành ngôn ngữ lý tưởng cho việc phát triển toàn diện (end-to-end). Trước đây, JavaScript chỉ được dùng để làm cho các website có tính tương tác hơn từ phía client (trình duyệt). Tuy nhiên, từ năm 2009, với Nodejs - một môi trường runtime cho phép JavaScript chạy trên server, các lập trình viên có thể tương tác với cơ sở dữ liệu, quản lý yêu cầu HTTP, tạo nội dung động bằng JavaScript. Điều này giúp lập trình viên sử dụng JavaScript cho cả frontend và backend, dẫn đến làn sóng "JavaScript ở khắp mọi nơi".

JavaScript đóng vai trò quan trọng trong nhiều ngành công nghiệp, giúp tạo ra các ứng dụng đa dạng, chúng ta hãy cùng tìm hiểu một số ứng dụng phổ biến được phát triển bằng JavaScript nhé.

Xây dựng trang Web Tương Tác: JavaScript giúp biến đổi những trang web tĩnh nhằm chán trở nên sống động và hấp dẫn hơn, nơi người dùng không chỉ xem mà còn có thể tương tác, kéo, thả các nội dung theo cách họ muốn.

Phát triển Ứng Dụng Di Động: Với sự hỗ trợ của các framework như React Native và Ionic, JavaScript cho phép bạn xây dựng ứng dụng di động cho cả Android và iOS chỉ với một mã nguồn, giúp tiết kiệm rất nhiều thời gian và chi phí.

Phát Triển Game: Bạn có thể dễ dàng tạo ra các trò chơi như Pong, Tic-Tac-Toe, hoặc các trò chơi câu đố đơn giản chỉ với vài dòng code JavaScript kết hợp với HTML. Game 2D hành động, phiêu lưu với đồ họa sống động và âm thanh chất lượng sử dụng Phaser, hay game 3D với Threejs hoặc Babylonjs.

Ứng dụng Trí Tuệ Nhân Tạo: Với các thư viện như TensorFlowjs, Brainjs JavaScript cho phép tạo ra các ứng dụng có khả năng nhận diện hình ảnh, hiểu ngôn ngữ tự nhiên, dự đoán và nhận diện mẫu.

Xây dựng Sơ đồ: Với JavaScript, bạn có thể dễ dàng tích hợp các biểu đồ vào bất kỳ ứng dụng web nào, cho dù đó là dashboard quản lý, trang báo cáo, hay website thương mại điện tử với các thư viện như Chartjs, D3js.

Với khả năng thích ứng và phát triển không ngừng, JavaScript đã khẳng định vị thế là một trong những ngôn ngữ lập trình được sử dụng nhiều nhất trên toàn cầu. Nếu bạn đang băn khoăn không biết học ngôn ngữ gì khi mới tiếp cận với ngành IT, mong muốn trở thành Backend/Frontend Developer trong tương lai thì JavaScript chính là lựa chọn tốt nhất vì JavaScript có thể vừa phát triển được cả backend lẫn frontend.

1.2. Giới thiệu về NextJS

NextJS không chỉ đơn giản là một framework React, nó là một môi trường phát triển mạnh mẽ, mang lại hiệu suất tuyệt vời và cho trải nghiệm người dùng vượt trội. Bằng cách tận dụng các tính năng quan tích hợp sẵn, NextJS cho phép chúng ta xây dựng các ứng dụng React một cách nhanh chóng và hiệu quả.

NextJS là một framework có mã nguồn mở được xây dựng trên nền tảng của React, cho phép chúng ta xây dựng các trang web tĩnh có tốc độ siêu nhanh và thân thiện với người dùng, cũng như xây dựng các ứng dụng web React.

NextJS được ra đời vào năm 2016, thuộc sở hữu của Vercel. NextJS bắt đầu trở nên phổ biến vào năm 2018 và tiếp tục tăng trưởng mạnh mẽ trong cộng đồng phát triển web. Sự kết hợp của các tính năng như Server-side Rendering (SSR) với Static Site Generation (SSG) đã giúp NextJS trở thành sự lựa chọn hấp dẫn cho nhiều dự án.

Các tính năng chính của NextJS:

Automatic Routing: NextJS sẽ tự động tạo các router dựa trên cấu trúc thư mục của chúng ta. Ví dụ, nếu bạn tạo một file có tên là `about.js` ở thư mục `pages`. NextJS sẽ tạo router là `/about`.

Nested Routing: Chúng ta có thể tạo các thư mục con để tạo các router lồng nhau. Ví dụ, nếu bạn tạo một folder có tên `blog` nằm trong folder `pages`, bên trong folder `blog` lại có file `post.js`, đường dẫn sẽ là `pages/blog/post.js`, thì router mà NextJS tạo ra sẽ là `/blog/post`.

Dynamic Routes: Bạn có thể tạo các router động bằng cách sử dụng cặp dấu `[]` trong tên file. Ví dụ nếu đường dẫn là `pages/blog/[slug].js` thì NextJS sẽ tạo ra các router như `/blog/blog-dau-tien` hoặc `/blog/blog-thu-hai`. Với slug là một giá trị bất kì do bạn truyền vào.

Link Component: Để tạo liên kết giữa các trang, bạn sử dụng component Link được cung cấp sẵn bởi NextJS ở thư viện `next/link`. Sử dụng link thay cho thẻ `a` giúp tránh việc tải lại trang và tối ưu hóa hiệu suất.

Query Parameters: Bạn có thể truyền dữ liệu giữa các trang sử dụng query parameters trong router bằng cách sử dụng ký tự dấu chấm hỏi `?` trong tên file. Ví dụ, `pages/product.js` có thể có các router như `/product?productId=0001`.

Ví dụ:

```
1 // pages/product.js
2 import { useRouter } from "next/router";
3 function ProductPage() {
4   const router = useRouter();
5   const { productId } = router.query;
6   return (
7     <div>
8       <h1>Product Details</h1>
9       <p>Product ID: {productId}</p>
10    </div>
11  );
12 }
13 export default ProductPage;
```

Hình 1.4 Xây dựng một component hiển thị chi tiết sản phẩm

Khi người dùng truy cập vào địa chỉ `/product?productId=123`, NextJS sẽ định tuyến bạn đến trang `/product`, và ở trang này bạn sẽ nhận được giá trị của `productId` mà bạn truyền vào.

Rendering trong NextJS

Từ những năm 2000, SSR đã được sử dụng rất phổ biến, gọi nó là SSR vì hầu hết các logic phức tạp trên trang web sẽ được xử lý ở phía server.

Cơ chế của SSR như sau:

- Khi user truy cập vào một trang web, trình duyệt sẽ gửi request tới server
- Server sẽ nhận request, truy cập vào dữ liệu trong database, render ra HTML
- Trả HTML kèm với CSS, JS về cho browser
- Browser nhận được HTML thì tiến hành tải xuống và render ra UI nhưng lúc này chưa có JS
- Browser tải xuống JS và thực thi các câu lệnh JS
- Website load xong và có thể tương tác bình thường

Ưu điểm của SSR:

- Load lần đầu nhanh vì toàn bộ dữ liệu đã được xử lý ở phía server, client chỉ cần nhận về và hiển thị ra cho user
- Tốt cho SEO vì dữ liệu được render dưới dạng HTML (bạn ấn chuột phải chọn View Page Source một trang web bất kì là sẽ thấy điều này), giúp cho bot của Google khi quét sẽ thấy được toàn bộ dữ liệu
- Lập trình viên chỉ cần code trên 1 project là đã có thể tạo ra trang web hoàn chỉnh có cả Frontend lẫn Backend, không cần tách ra làm 2 project.

Nhược điểm của SSR:

- Server sẽ phải xử lý nhiều dữ liệu dẫn đến quá tải
- Khi user chuyển trang thì cả trang sẽ phải load lại để lấy dữ liệu từ server, dẫn đến trải nghiệm không tốt

Hiện tại vẫn còn rất nhiều website đang sử dụng SSR như các website được xây dựng bằng Wordpress, các trang bán hàng lâu đời như thegioididong,...

Vì những nhược điểm nêu trên mà đến năm 2010, sự phát triển của JavaScript đã tiến thêm một bước nữa khi Client-side Rendering ra đời, nhằm khắc phục những nhược điểm của SSR.

Client-side Rendering

Sở dĩ nó được gọi là CSR vì việc render HTML sẽ được thực thi ở phía client. Hay chúng ta còn gọi là Single Page App (SPA).

Các ưu điểm của CSR:

- Chuyển việc xử lý dữ liệu sang cho client giúp server nhẹ việc hơn
- Trang chỉ cần load một lần duy nhất, khi user muốn lấy dữ liệu mới từ server chỉ cần gọi đến server thông qua AJAX
- Trang web không cần load lại nhiều khi user chuyển trang, đem đến trải nghiệm tốt hơn cho người dùng

Tuy nhiên, CSR vẫn còn một số nhược điểm như dưới đây:

- Load trang lần đầu sẽ chậm hơn SSR nếu không tối ưu. Do browser phải tải toàn bộ HTML và JS ở lần đầu load trang, chạy JS và gọi API để lấy dữ liệu từ server rồi mới render lên UI cho user.
- Lập trình viên phải chia ra làm 2 project: Backend để viết API và Frontend để hiển thị
- Website không thể chạy nếu tắt JavaScript ở trình duyệt
- Nếu user sử dụng thiết bị có cấu hình yếu thì sẽ bị load chậm
- SEO không tốt bằng SSR do dữ liệu được render bởi JS là dữ liệu động

Bạn sẽ thấy CSR thường được sử dụng trong các ứng dụng web cần sự tương tác nhiều, các ứng dụng này chủ yếu được xây dựng bằng ReactJS hay VueJS, AngularJS.

Chính vì những nhược điểm của CSR mà NextJS ra đời. NextJS là sự kết hợp cả SSR lẫn CSR để tạo nên website có trải nghiệm tuyệt vời nhất.

SSR trong NextJS

Khi user truy cập vào một trang sử dụng SSR của NextJS, browser sẽ gửi một request đến server

Trước khi gửi kết quả ra trình duyệt, NextJS chạy hàm `getServerSideProps()` để lấy dữ liệu cần thiết cho trang. Hàm này thường sẽ dùng để gọi API hoặc truy vấn vào cơ sở dữ liệu để lấy dữ liệu cần thiết.

Với dữ liệu cần thiết đã được lấy từ bước trước, NextJS tạo ra một phiên bản đã render đầy đủ cả nội dung và dữ liệu. Sau đó trả phiên bản này ra browser dưới dạng HTML

Browser nhận được code HTML từ server và hiển thị lên giao diện. Nội dung của trang sẽ được hiển thị ngay lập tức, sau đó JS được tải xuống và đổ lên để trang có thể tương tác được, ví dụ như hàm `onClick()`.

Static Site Generation trong NextJS

SSG là một phương pháp mà NextJS cung cấp sẵn cho chúng ta, cho phép bạn tạo các trang tĩnh và lưu chúng xuống dưới dạng file html tĩnh. Điều này giúp cải thiện hiệu suất tải trang và cung cấp trải nghiệm người dùng tốt hơn vì nội dung được lấy từ file html và hiển thị ngay lập tức mà không cần đợi việc tải về từ phía server.

Khi bạn chạy lệnh next build, NextJS sẽ chạy hàm getStaticProps để lấy dữ liệu cần thiết. Với dữ liệu lấy được, NextJS sẽ tạo ra các phiên bản đã được render, lưu chúng trong thư mục .next, và hiển thị lên khi user truy cập.

Với khả năng tối ưu hóa và tích hợp linh hoạt, NextJS giúp tối đa hóa trải nghiệm người dùng và giảm thời gian phát triển. Đồng thời, nó cũng cung cấp sự hỗ trợ đáng kể, tạo nền tảng vững chắc để khám phá và tiến xa hơn trong thế giới phức tạp của phát triển web.

1.3. Giới thiệu về NestJS

Nest hay NestJS là một framework để xây dựng các ứng dụng server-side mạnh mẽ dựa trên nền tảng Nodejs. NestJS được xây dựng và hỗ trợ ngôn ngữ TypeScript (vẫn cho phép lập trình viên code bằng JavaScript thuần), và kết hợp các tính năng của OOP (Object Oriented Programming), FP (Functional Programming), and FRP (Functional Reactive Programming).

Sâu bên dưới thì NestJS vẫn sử dụng các framework HTTP Server đơn giản hơn như Express và Fastify. NestJS cung cấp một level trừu tượng hơn phía bên trên các framework này, nhưng cũng cung cấp các API trực tiếp để lập trình viên có thể sử dụng các tính năng, cũng như là các module bên thứ 3 được hỗ trợ cho các nền tảng Express và Fastify.

Tại sao nên sử dụng NestJS?

- Đối với một lập trình viên Nodejs, việc lựa chọn đúng framework cho project rất quan trọng. Trong khi Expressjs được sử dụng để xây dựng ứng dụng web trong nhiều năm thì NestJS đã xuất hiện và mang lại các lợi ích và tính năng nổi trội hơn:
- Ngôn ngữ mạnh về kiểu dữ liệu: NestJS được xây dựng trên nền TypeScript, một phiên bản hỗ trợ kiểu dữ liệu của JavaScript. Sử dụng Typescript khi viết code giúp code sạch hơn và dễ xử lý khi có lỗi.
- Kiến trúc modular: NestJS được xây dựng trên kiến trúc modular giúp chúng ta dễ tổ chức và phát triển mã nguồn trên quy mô lớn hơn. Kiến trúc này bao gồm các khối tách biệt nhau như controller, provider, và module, giúp cho việc quản lý dễ dàng hơn. Mình sẽ nói kĩ hơn về các thành phần này ở phần sau của bài viết nhé.
- Dependency Injection: NestJS sử dụng Dependency Injection (DI) để quản lý luồng của các phụ thuộc giữa các module và component. Nó cho phép ứng dụng dễ kiểm thử và bảo trì hơn, do sự độc lập giữa các component với nhau. Cụ thể độc lập như thế nào thì cùng đợi đến phần sau nhé.
- Built-In Validation: NestJS được hỗ trợ sẵn tính năng validate dữ liệu đầu vào - thứ đặc biệt quan trọng khi xây dựng các API. Framework này giúp định nghĩa và thực thi các validation rule một cách dễ dàng hơn, giảm đi các sai sót và lỗi không đáng có.
- Hỗ trợ các tính năng nâng cao: NestJS hỗ trợ hàng loạt các tính năng được tích hợp sẵn, ví dụ như WebSocket, GraphQL, và Microservices. Điều này giúp đơn giản hóa việc xây dựng các ứng dụng yêu cầu tính năng realtime hay kiến trúc microservices.

- Cộng đồng hỗ trợ mạnh mẽ: NestJS có một cộng đồng các lập trình viên đóng góp cho framework thường xuyên. Điều này sẽ giúp NestJS luôn được cập nhật và sửa lỗi, trở thành 1 framework đáng tin cậy.

Nhược điểm của NestJS

- Việc hỗ trợ nhiều tính năng mạnh mẽ không có nghĩa là NestJS không có bất kỳ điểm yếu nào.
- Sự phụ thuộc vòng lặp (circular dependencies): Đây là một vấn đề phổ biến mà hầu hết các dự án NestJS sẽ gặp phải. Vấn đề này rất rắc rối và nó có thể làm chậm đi quá trình phát triển của phần mềm. May mắn là, một bài báo gần đây của Trilon đã đưa ra một công cụ tên là Madge, giúp xác định sớm circular dependencies.
- Logs bị ẩn khi ứng dụng khởi động: Đây cũng là một vấn đề khá phổ biến. Điều này có thể khiến các lập trình viên khó debug khi có lỗi xảy ra. Để giải quyết thì họ thường vô hiệu hóa việc dùng ứng dụng khi gặp lỗi và gửi lại thông báo lỗi.
- Khó khăn trong kiểm thử đơn vị (unit test): Unit test được tích hợp sâu vào framework. Việc kiểm thử ở mức đơn vị nhỏ thường gây ra nhiều boilerplate code. Hơn nữa, việc viết test có thể gây khó khăn với một số lập trình viên, vì họ phải hiểu được cơ chế hoạt động của dependency injection tree trong NestJS.

NestJS là một giải pháp mới trong lĩnh vực backend, với tập hợp các tính năng nhằm giúp lập trình viên xây dựng và triển khai các ứng dụng enterprise đáp ứng được nguyên tắc SOLID.

1.4. Giới thiệu về PostgreSQL

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ đối tượng (ORDBMS) mã nguồn mở, được biết đến với tính toàn vẹn cao, khả năng mở rộng và hỗ trợ nhiều chuẩn dữ liệu. PostgreSQL tuân thủ nghiêm ngặt các tiêu chuẩn SQL, hỗ trợ các kiểu dữ liệu nâng cao như JSON, XML, và kiểu dữ liệu địa lý (PostGIS).

Việc lựa chọn một hệ quản trị cơ sở dữ liệu phù hợp là yếu tố quan trọng để đảm bảo hiệu suất và tính ổn định cho ứng dụng của bạn. Dưới đây là những lý do tại sao PostgreSQL là lựa chọn tuyệt vời:

PostgreSQL được thiết kế với kiến trúc mạnh mẽ, cho phép xử lý các dữ liệu phức tạp cách hiệu quả. Nhờ vào trình tối ưu hóa truy vấn thông minh và hỗ trợ các tính năng như: tối ưu hóa lưu trữ, PostgreSQL có thể:

- Xử lý tốt khi dữ liệu của bạn tăng lên hàng triệu bản ghi.
- Hiệu suất cao ngay cả khi thực hiện các phép tính và kết nối nhiều bảng.
- Hỗ trợ nhiều kết nối đồng thời mà không ảnh hưởng đến hiệu suất.

Ví dụ: Hãy tưởng tượng bạn có một shop với hàng ngàn sản phẩm và hàng triệu lượt truy cập một ngày. PostgreSQL đảm bảo rằng khách hàng có thể tìm kiếm sản phẩm và thực hiện giao dịch một cách nhanh chóng mà không gặp trục trặc.

PostgreSQL nổi bật với khả năng hỗ trợ nhiều kiểu dữ liệu khác nhau, bao gồm:

- Kiểu dữ liệu cơ bản: integer, number, string, boolean
- Structured: Date/Time, Array, Range / Multirange, UUID
- JSON và JSONB: lưu trữ dữ liệu dạng JSON, cho phép truy vấn và thao tác trực tiếp trên dữ liệu JSON.
- Geometry: sử dụng với extension PostGIS để lưu trữ và xử lý dữ liệu không gian.

Cho phép bạn lưu trữ và quản lý nhiều loại dữ liệu khác nhau trong cùng một cơ sở dữ liệu, mang lại sự linh hoạt trong xây dựng ứng dụng.

Ví dụ: nếu bạn cần lưu trữ thông tin sản phẩm cùng với các thuộc tính đa dạng như màu sắc, kích thước, đánh giá của khách hàng (dưới dạng JSON), PostgreSQL hoàn toàn có thể đáp ứng.

Bảng 1.1 Phân tích sự khác nhau giữa PostgreSQL và MySQL

Tiêu chí	PostgreSQL	MySQL
Kiểu dữ liệu hỗ trợ	Đa dạng: number, string, boolean, array, JSON/JSONB, XML, hstore, geo (PostGIS),..	Cơ bản: number, string, boolean, datetime. Hỗ trợ JSON dạng chuỗi, không có JSON gốc như PostgreSQL.
Transaction & ACID	Hỗ trợ ACID hoàn chỉnh, sử dụng MVCC giúp quản lý đồng thời hiệu quả	Hỗ trợ ACID với InnoDB. MyISAM không hỗ trợ transaction và khóa ngoại
Hiệu suất	Mạnh trong xử lý giao dịch phức tạp, workload ghi cao, hệ thống lớn	Nhanh với truy vấn đơn giản, workload đọc nhiều, phù hợp web traffic lớn
Tuân thủ chuẩn SQL	Tuân thủ nghiêm chuẩn SQL	Không hoàn toàn tuân thủ chuẩn SQL
Khả năng mở rộng	Mở rộng mạnh: tạo hàm, kiểu dữ liệu, chỉ mục tùy chỉnhStored procedure nhiều ngôn ngữ (PL/pgSQL, Python,..) Extensions như PostGIS, Citus	Mở rộng qua plugin nhưng hạn chếStored procedure chủ yếu SQL/PSM
Bảo mật	Phân quyền chi tiết đến cấp hàng/cộtNhiều phương thức xác	Quản lý user và quyền ở mức bảng, ít chi tiết hơnXác thực qua plugin nhưng ít lựa chọn

	thực (md5, scram-sha-256, GSSAPI,...)	
Sao lưu và phục hồi	Hot backup: pg_dump, pg_basebackup Phục hồi theo thời gian (PITR)	Backup qua mysqldump hoặc copy file dữ liệu Không có PITR linh hoạt như PostgreSQL
Ứng dụng phù hợp	Hệ thống doanh nghiệp, ngân hàng – tài chính, ứng dụng GIS, hệ thống yêu cầu logic phức tạp	Ứng dụng web, blog, CMS, E-commerce vừa & nhỏ

1.5. Giới thiệu về Flask

Flask là một micro framework phổ biến của Python, thường được sử dụng để xây dựng các ứng dụng web và API. Với Flask, lập trình viên có thể dễ dàng tạo ra các API RESTful để phục vụ cho các ứng dụng phần mềm khác.

Flask là lựa chọn quen thuộc của nhiều lập trình viên khi xây dựng RESTful API cho các ứng dụng quy mô nhỏ đến vừa, nhờ các ưu điểm sau:

- Microframework nhẹ và tối giản: Cung cấp đúng những gì cần thiết để xây dựng ứng dụng web hoặc API, cho phép tạo API nhanh chóng với cấu hình tối thiểu, giúp đơn giản hóa quá trình phát triển.
- Có sẵn server phát triển: Giúp dễ dàng kiểm thử và debug API trong quá trình phát triển.
- Hỗ trợ Jinja2: Tích hợp engine template mạnh mẽ để xây dựng giao diện HTML hoặc phản hồi JSON một cách linh hoạt.
- Độ linh hoạt cao: Không áp đặt cấu trúc, dễ dàng tùy chỉnh theo từng dự án, độ linh hoạt cao hơn cả các framework lớn như Django.
- Kiểm soát chi tiết: Cho phép lập trình viên “làm chủ” toàn bộ hệ thống mà không bị ràng buộc bởi những quy tắc khuôn mẫu cố định.
- Dễ mở rộng: Hỗ trợ nhiều thư viện và plugin bên thứ ba như xác thực, kết nối cơ sở dữ liệu,..
- Thừa hưởng lợi thế lớn từ cộng đồng Python: Có tài nguyên mã nguồn dồi dào, hỗ trợ hiệu quả cho việc phát triển API.

Tuy nhiên, bạn cũng cần cân nhắc khi lựa chọn Flask vì framework này có một số hạn chế sau:

- Chức năng giới hạn: Vì là microframework, Flask chỉ cung cấp những thành phần cơ bản. Nếu cần tính năng phức tạp, bạn phải tích hợp thêm thư viện ngoài.

- Khó mở rộng khi dự án lớn: Dễ tiếp cận cho dự án nhỏ, nhưng khi ứng dụng mở rộng, việc quản lý phụ thuộc và cấu trúc dự án có thể trở nên phức tạp.
- Không tích hợp sẵn database: Cần dùng thêm thư viện như SQLAlchemy để làm việc với cơ sở dữ liệu.
- Tiềm ẩn rủi ro bảo mật: Giống mọi web framework khác, Flask có thể bị tấn công (SQL injection, XSS...). Lập trình viên cần nắm rõ và có biện pháp bảo mật phù hợp.

```
1 $ python --version
2
3 Python 3.10.4
4
5 $ pip --version
6
7 pip 22.0.4 from /usr/lib/python3/dist-packages/pip (python 3.10)
```

Hình 1.5 Kiểm tra phiên bản hiện tại của python và pip

Nếu một trong hai lệnh trên không hoạt động, bạn có thể truy cập python.org để tải và cài đặt Python (pip đi kèm). Sau đó, tiến hành cài đặt Flask:

```
1 $ pip install flask
2
3 Collecting flask
4
5 ...
6
7 Successfully installed Jinja2-3.1.1 MarkupSafe-2.1.1 Werkzeug-2.1.1...
```

Hình 1.6 Cài đặt flask

Viết API đầu tiên

```
1 from flask import Flask
2
3 app = Flask(__name__)
4 @app.route('/')
5
6 def hello():
7     return {'message': 'Hello, World!'}
8
9 if __name__ == '__main__':
10     app.run(debug=True)
```

Hình 1.7 Khởi tạo API đơn giản in ra ‘Hello, World!’

Flask là một lựa chọn tuyệt vời để xây dựng API nhờ vào sự đơn giản, linh hoạt và hệ sinh thái mở rộng phong phú hỗ trợ các tính năng như tạo tài liệu, xác thực và triển khai.

Có thể được sử dụng làm nền tảng backend để xây dựng API, cung cấp dữ liệu cho các ứng dụng giao diện người dùng hoặc tích hợp với dịch vụ bên thứ ba (third-party service) như ứng dụng di động (iOS, Android), hệ thống thanh toán (Paypal, Stripe),...

Thường được sử dụng trong các trường hợp như xây dựng RESTful API, phát triển microservices, tạo dashboard quản trị, thử nghiệm ý tưởng (prototyping), và tích hợp với mô hình machine learning.

1.6. Giới thiệu về OpenCV

OpenCV (Open Computer Vision) là một thư viện mã nguồn mở hàng đầu cho xử lý về thị giác máy tính, machine learning, xử lý ảnh. OpenCV được viết bằng C/C++, vì vậy có tốc độ tính toán rất nhanh, có thể sử dụng với các ứng dụng liên quan đến thời gian thực. OpenCV có các interface cho C/C++, Python Java vì vậy hỗ trợ được cho Window, Linux, MacOS lẫn Android, iOS OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần

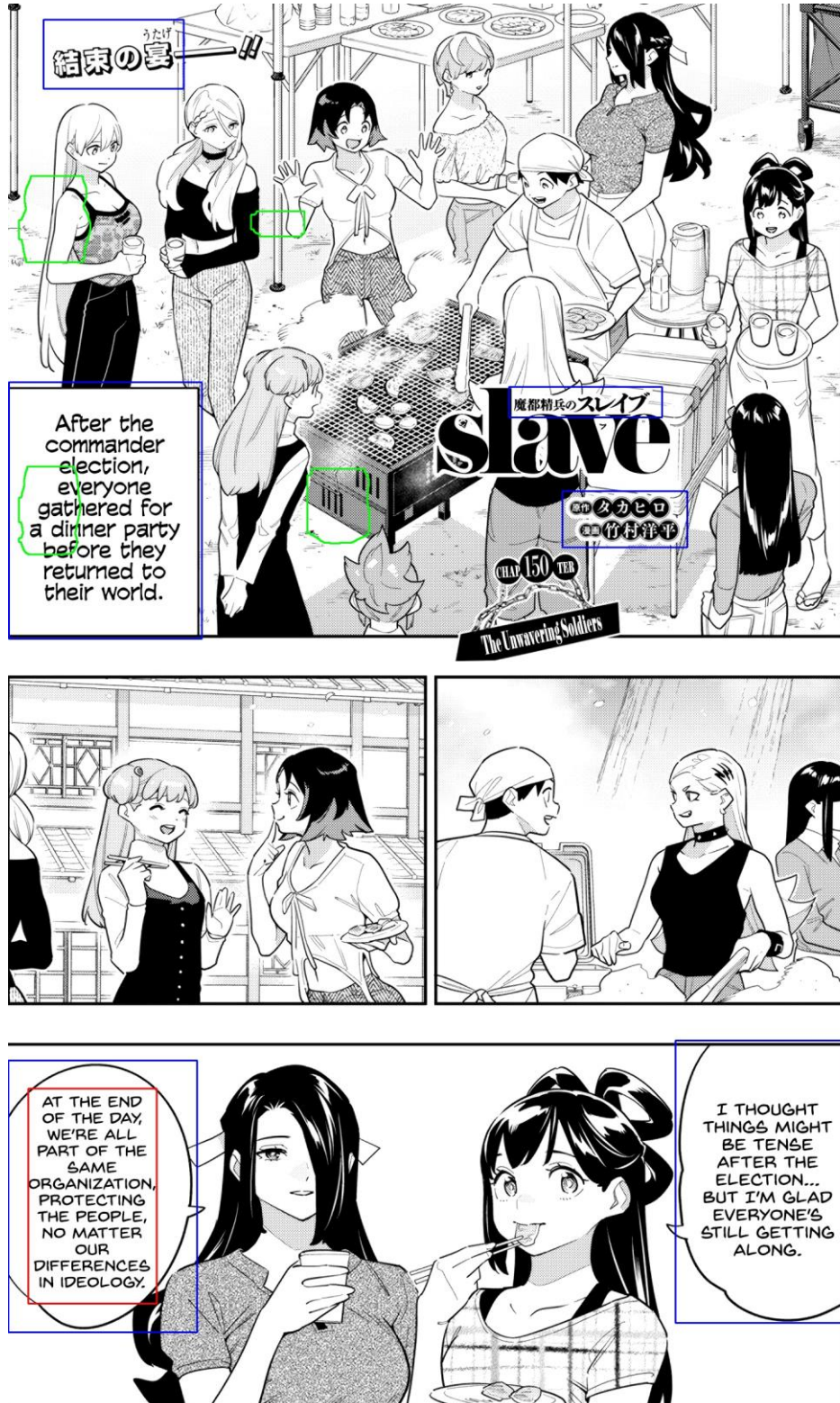
OpenCV có khả năng thực hiện nhiều tác vụ như phân loại hình ảnh, phát hiện và phân đoạn đối tượng, trích xuất đặc điểm và thậm chí nhận dạng chữ viết tay của con người,... nhờ một bộ hơn 2500 thuật toán đã được tối ưu. Trong đó các thuật toán phổ biến nhất có thể kể đến như:

Thuật toán Support Vector Machines (SVM) và K-Nearest Neighbors (KNN): Đây là hai thuật toán phổ biến trong việc phân loại và nhận dạng mẫu, được áp dụng trong các nhiệm vụ như: nhận diện chữ viết tay, phân loại khuôn mặt và các bài toán xử lý ảnh khác.

Cây Quyết Định (Decision Trees): Thuật toán này có thể kết hợp với các phương pháp học máy khác như Random Forest hoặc Gradient Boosting để tăng hiệu quả, đặc biệt trong các bài toán như phát hiện đối tượng và phân đoạn ảnh.

Deep Learning: OpenCV hỗ trợ triển khai các mô hình học sâu từ các framework như TensorFlow và PyTorch, giúp thực hiện các tác vụ như nhận diện đối tượng, phân loại và phân đoạn ảnh.

YOLO: Thuật toán phát hiện đối tượng nổi bật, cho phép nhận diện nhiều đối tượng trong một lần quét ảnh duy nhất.



Hình 1.8 Nhận diện đối tượng qua ảnh

Phân tích đặc trưng 2D và 3D

OpenCV cung cấp các công cụ phân tích các đặc trưng 2D và 3D trong hình ảnh và video như so khớp hình ảnh, theo dõi đối tượng và ứng dụng thực tế ảo (AR). Ví dụ, OpenCV có thể phát hiện và so khớp các đặc điểm chính trong hình ảnh để tạo ra ảnh toàn cảnh (panorama) từ nhiều hình ảnh khác nhau. Hoặc kết hợp với các công cụ như OpenMVG, COLMAP hoặc thư viện Structure from Motion (SfM) để có thể tạo ảnh 3D.

Nhận diện khuôn mặt và mở khóa thiết bị

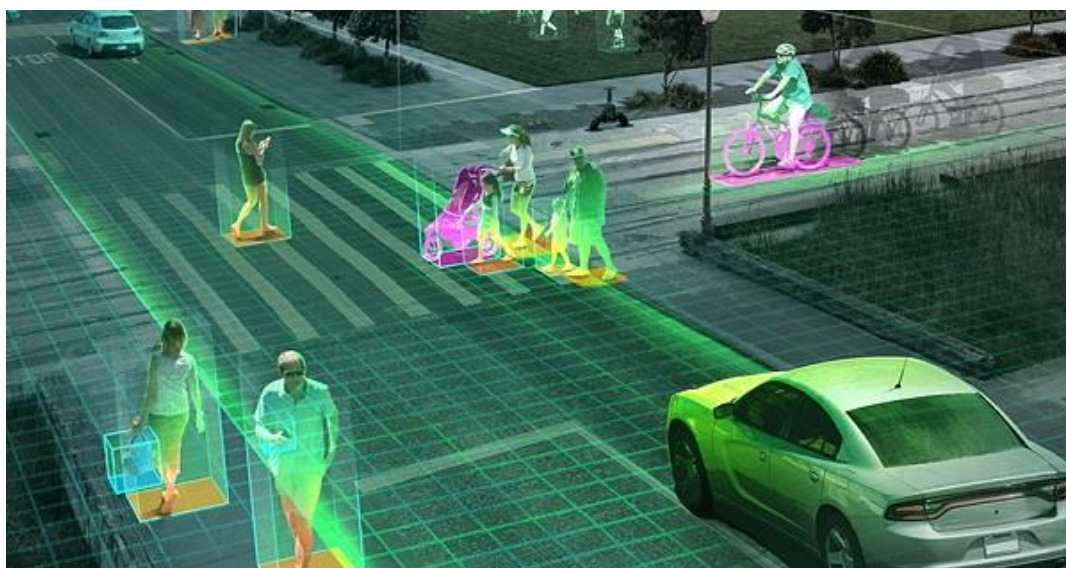
Thêm một ứng dụng phổ biến của OpenCV là nhận diện khuôn mặt đã được sử dụng trong nhiều hệ thống bảo mật. OpenCV có thể nhận diện khuôn mặt và giúp mở khóa điện thoại thông minh hoặc cấp quyền truy cập vào các khu vực bảo mật bằng cách nhận diện khuôn mặt người dùng.

Nhận diện cử chỉ và tương tác giữa người và máy tính

OpenCV giúp phát triển các hệ thống nhận diện cử chỉ, cho phép người dùng tương tác với máy tính hoặc các thiết bị thông qua các cử chỉ và chuyển động tay. Công nghệ này rất phổ biến trong các trò chơi, thực tế ảo (VR),... Ví dụ webcam có thể theo dõi chuyển động tay của người dùng, cho phép điều khiển các hành động trên màn hình như điều hướng menu hay chơi game mà không cần sử dụng bộ điều khiển vật lý.

Hiểu chuyển động và phát hiện đối tượng

Các thuật toán của OpenCV thường được tích hợp trong các hệ thống giám sát, xe tự lái (nhận diện vạch đường, vật thể) và robot nhờ khả năng hiểu chuyển động và phát hiện đối tượng. Chẳng hạn như kỹ thuật "Background Subtraction" giúp nhận diện chuyển động khi phát hiện một vật thể đã xuất hiện trong khung hình. Hoặc Optical Flow và Deep Learning cũng là hai nhóm thuật toán thường được sử dụng để theo dõi đối tượng trong thời gian thực.



Hình 1.9 Mô hình hóa đối tượng

Tạo hiệu ứng 3D và độ sâu

OpenCV hỗ trợ một số kỹ thuật xử lý ảnh và video liên quan đến hiệu ứng 3D như ước lượng độ sâu từ ảnh stereo (với các thuật toán như StereoBM và StereoSGBM) và theo dõi chuyển động (Optical Flow). Tuy nhiên, để tạo ra đồ họa 3D hoàn chỉnh hoặc các hiệu ứng thị giác phức tạp thường cần kết hợp OpenCV với các công cụ chuyên biệt như OpenGL, Unreal Engine hoặc các phần mềm dựng Hình 2D.

Bên cạnh những tính năng nổi bật, người dùng cũng cần quan tâm đến các ưu nhược điểm của OpenCV như sau:

- Độ phức tạp: Với nhiều tính năng và chức năng, OpenCV có thể khiến người mới bắt đầu cảm thấy choáng ngợp và mất thời gian để làm quen.
- Thiếu tính năng nâng cao: OpenCV phù hợp với các tác vụ cơ bản và trung cấp nhưng thiếu các tính năng tiên tiến như các thư viện chuyên dụng cho học sâu như TensorFlow hay PyTorch.
- C++ là ngôn ngữ chính: Nền tảng OpenCV chủ yếu được viết bằng C++, điều này có thể gây khó khăn cho những lập trình viên quen thuộc với các ngôn ngữ cấp cao như Python hay Java.
- Quản lý phụ thuộc: Việc tích hợp OpenCV với các thư viện khác trong dự án phức tạp có thể gặp phải vấn đề tương thích và quản lý phụ thuộc.
- Hiệu suất: OpenCV có thể không tối ưu cho các tác vụ đặc thù yêu cầu hiệu suất cao, so với việc viết mã tùy chỉnh.
- Tài liệu hạn chế: Một số tài liệu của OpenCV có thể không đầy đủ hoặc đã lỗi thời, gây khó khăn trong việc tìm kiếm thông tin.
- Hạn chế trong học máy: OpenCV không phải là thư viện chuyên dụng cho học máy nên không thể đáp ứng đầy đủ các yêu cầu phức tạp như các thư viện học sâu.

1.7. Giới thiệu về PyTorch

PyTorch là một thư viện học sâu (deep learning) mã nguồn mở được phát triển bởi Facebook AI Research (FAIR). Nó cung cấp công cụ mạnh mẽ để xử lý dữ liệu tensor và xây dựng các mô hình mạng nơ-ron phức tạp. PyTorch ngày càng trở thành tiêu chuẩn trong nghiên cứu AI nhờ tính linh hoạt và hiệu năng vượt trội.

Bảng 1.2 Phân tích tính năng của Pytorch

Tính năng	Mô tả
Dynamic Computation Graph	Biểu đồ tính toán động: giúp debug dễ dàng như code Python thông thường
Tối ưu hiệu năng	Hỗ trợ GPU/TPU nhờ CUDA xử lý nhanh gấp nhiều lần CPU
Dễ dùng & rõ ràng	API thân thiện, cú pháp giống NumPy

Hệ sinh thái phong phú	TorchVision, TorchText, TorchAudio, PyTorch Lightning, ...
Cộng đồng mạnh	Được sử dụng rộng rãi trong nghiên cứu và công nghiệp

Bảng 1.3 Thành phần chính của PyTorch

Thành phần	Vai trò
torch.Tensor	Cấu trúc dữ liệu chính, giống NumPy nhưng hỗ trợ GPU
Autograd	Tự động đạo hàm phục vụ tối ưu mô hình
Optimizers (SGD, Adam, ...)	Tối ưu trọng số trong quá trình học
nn.Module	Xây dựng mạng nơ-ron linh hoạt
DataLoader & Dataset	Quản lý dữ liệu song song và theo batch

Ứng dụng của PyTorch trong AI

- Computer Vision: phân loại hình ảnh, phát hiện đối tượng, nhận dạng khuôn mặt
- NLP: chatbot, dịch máy, tổng hợp văn bản
- Generative AI: GANs, Diffusion Models → ứng dụng tạo ảnh/video
- Reinforcement Learning: game AI, robot tự hành
- Mô hình đa dạng: Transformers, LLMs (như ChatGPT cũng học dựa trên nền tảng tương tự)

```
1 import torch
2
3 x = torch.tensor([[1, 2], [3, 4]], dtype=torch.float32)
4 y = torch.ones((2, 2))
5
6 print(x + y) # [[2., 3.],
7              # [4., 5.]]
```

Hình 1.10 Tính tensor cơ bản

```
1 import torch.nn as nn
2
3 model = nn.Sequential(
4     nn.Linear(10, 20),
5     nn.ReLU(),
6     nn.Linear(20, 2)
7 )
8
9 print(model)
```

Hình 1.11 Mạng nơ-ron đơn giản

PyTorch là một nền tảng học sâu mạnh mẽ, linh hoạt và đang giữ vai trò quan trọng trong sự phát triển của trí tuệ nhân tạo hiện đại. Đối với sinh viên Công nghệ Thông tin, nắm vững PyTorch không chỉ mở ra cơ hội trong nghiên cứu AI mà còn tạo lợi thế lớn cho phát triển sự nghiệp trong các lĩnh vực ứng dụng trí tuệ nhân tạo tương lai.

CHƯƠNG 2: THIẾT KẾ HỆ THỐNG

2.1. Phân tích hệ thống

2.1.1. Tổng quan bài toán

Bài toán đặt ra là xây dựng một nền tảng đọc và quản lý truyện tranh trực tuyến, phục vụ ba nhóm đối tượng chính: người đọc, người sáng tạo nội dung và quản trị viên. Hệ thống cần giải quyết các vấn đề then chốt bao gồm bảo vệ bản quyền nội dung số, tự động hóa quy trình biên tập và dịch thuật, quản lý quyền truy cập đa cấp, và cung cấp trải nghiệm đọc truyện ổn định, chất lượng cao.

Khác với các nền tảng đọc truyện truyền thống chỉ tập trung vào việc hiển thị nội dung, hệ thống này hướng đến việc xây dựng một hệ sinh thái hoàn chỉnh, nơi người sáng tạo có thể dễ dàng xuất bản và bảo vệ tác phẩm của mình, đồng thời người đọc có trải nghiệm tốt nhất với nội dung được bảo vệ hợp pháp.

2.1.2. Các thành phần chính

Hệ thống quản lý người dùng và phân quyền

Hệ thống cần quản lý nhiều loại người dùng với các quyền hạn khác nhau. Người dùng Guest chỉ có quyền đọc và tương tác cơ bản. Creator có khả năng tạo và quản lý nội dung của riêng mình. Editor có quyền cao hơn, có thể chỉnh sửa mọi nội dung trong hệ thống. Admin có toàn quyền quản trị, bao gồm quản lý người dùng, phân quyền và giám sát hoạt động của toàn hệ thống.

Đặc biệt, hệ thống phải hỗ trợ mô hình làm việc nhóm thông qua tính năng quản lý team. Mỗi team có cấu trúc phân quyền riêng với hai vai trò: Admin team có toàn quyền quản lý nhóm và nội dung, trong khi Member chỉ có quyền hạn chế đối với các chương truyện do chính mình đăng tải. Mô hình này phản ánh đúng thực tế hoạt động của các nhóm scanlation, nơi có sự phân công rõ ràng giữa người quản lý dự án và các thành viên thực hiện công việc cụ thể.

Hệ thống quản lý và xử lý nội dung

Quản lý nội dung truyện tranh là thành phần cốt lõi của hệ thống. Mỗi truyện tranh bao gồm nhiều thông tin metadata như tiêu đề, mô tả, thể loại, tác giả, trạng thái và ảnh bìa. Mỗi truyện chứa nhiều chương, và mỗi chương chứa một tập hợp các ảnh được sắp xếp theo thứ tự.

Vấn đề phức tạp nằm ở quy trình xử lý ảnh. Hệ thống cần hỗ trợ hai workflow khác nhau tùy theo đầu vào của Creator. Với workflow đầu tiên, khi Creator upload các chương đã dịch sẵn, hệ thống tự động thêm watermark để đánh dấu nguồn gốc, thêm banner quảng bá, merge các ảnh nếu cần thiết, và cuối cùng mã hóa ảnh bằng công nghệ DRM để bảo vệ nội dung khỏi sao chép trái phép.

Workflow thứ hai phức tạp hơn, dành cho trường hợp Creator chỉ có ảnh raw chưa được dịch. Hệ thống cần tích hợp với dịch vụ xử lý ảnh sử dụng công nghệ OCR để nhận dạng văn bản trong ảnh, sau đó sử dụng AI để dịch sang ngôn ngữ đích. Creator có thể chọn giữa hai chế độ: nhận về tọa độ và nội dung text để tự chỉnh sửa, hoặc nhận về ảnh đã được xử lý hoàn chỉnh. Điều này đòi hỏi sự tích hợp chặt chẽ giữa frontend, backend và service xử lý ảnh bên ngoài.

Hệ thống đọc truyện và bảo vệ nội dung

Trải nghiệm đọc truyện cần được tối ưu hóa về cả hiệu năng và độ tin cậy. Hệ thống phải hỗ trợ nhiều server ảnh để đảm bảo tính sẵn sàng. Khi người dùng đọc truyện, nếu một server ảnh gặp sự cố, họ có thể chuyển sang server khác ngay lập tức mà không cần reload trang hay mất vị trí đang đọc.

Vấn đề bảo mật nội dung được giải quyết thông qua việc mã hóa ảnh bằng WebAssembly. Ảnh truyện không được lưu trữ ở dạng thông thường mà được xáo trộn các layer, khiến việc tải xuống trực tiếp từ browser developer tools trở nên vô nghĩa. Khi người dùng đọc truyện, module WebAssembly sẽ giải mã và hiển thị ảnh theo thời gian thực. Phương pháp này tạo ra một lớp bảo vệ đáng kể, tuy không thể ngăn chặn hoàn toàn nhưng làm tăng đáng kể độ khó trong việc sao chép nội dung.

Hệ thống tương tác cộng đồng

Để tạo ra một cộng đồng sôi động, hệ thống cần cung cấp các kênh tương tác đa dạng. Tính năng bình luận cho phép người dùng thảo luận về từng chương truyện cụ thể, với khả năng trả lời bình luận tạo thành cấu trúc nhiều cấp. Hệ thống chat toàn server cho phép người dùng giao tiếp theo thời gian thực, tạo cảm giác kết nối trong cộng đồng.

Để khuyến khích sự tham gia và tạo nguồn thu, hệ thống triển khai cơ chế điểm danh kết hợp với quảng cáo. Người dùng xem quảng cáo để nhận Mirucoins, sau đó có thể sử dụng số coin này để mua các vật phẩm trang trí trong cửa hàng. Mô hình này vừa tạo động lực cho người dùng quay lại hàng ngày, vừa tạo nguồn thu từ quảng cáo để duy trì hoạt động của hệ thống.

Hệ thống quản trị và giám sát

Admin cần có công cụ quản trị mạnh mẽ để giám sát và điều hành hệ thống. Dashboard quản trị cung cấp các thống kê quan trọng như số lượng người dùng mới theo thời gian, top người dùng hoạt động, số lượng truyện và chương được đăng, lượt đọc theo thời gian. Các biểu đồ trực quan giúp Admin nắm bắt xu hướng và đưa ra quyết định phù hợp.

Admin cũng cần khả năng can thiệp trực tiếp vào hoạt động của hệ thống, bao gồm chỉnh sửa hoặc xóa người dùng vi phạm, quản lý các team, kiểm duyệt nội dung không phù hợp. Hệ thống ghi log chi tiết các hoạt động quan trọng để Admin có thể audit và xử lý các vấn đề khi cần thiết.

2.1.3. Thách thức

Bảo mật và bảo vệ nội dung

Thách thức lớn nhất là làm sao bảo vệ nội dung truyện tranh khỏi sao chép trái phép trong môi trường web, nơi mọi thứ đều có thể bị inspect và download. Giải pháp sử dụng WebAssembly để mã hóa ảnh là một cách tiếp cận sáng tạo, nhưng cũng đặt ra thách thức về hiệu năng. Module WASM phải đủ nhanh để giải mã và hiển thị ảnh mượt mà, đồng thời đủ phức tạp để không dễ dàng bị reverse engineer.

Watermark tự động cũng cần được thiết kế cẩn thận. Watermark phải đủ rõ ràng để truy vết nguồn gốc khi bị chia sẻ trái phép, nhưng không được làm ảnh hưởng đến trải nghiệm đọc. Vị trí, kích thước và độ trong suốt của watermark cần được cân nhắc kỹ lưỡng.

Hiệu năng và khả năng mở rộng

Với một hệ thống cung cấp nội dung đa phương tiện như truyện tranh, bandwidth và storage là hai vấn đề quan trọng. Mỗi chương truyện có thể chứa hàng chục ảnh có kích thước lớn. Khi có nhiều người dùng truy cập đồng thời, lượng dữ liệu cần truyền tải là rất lớn.

Ảnh truyện được phân tán trên nhiều server và sử dụng kỹ thuật lazy loading, chỉ tải ảnh khi người dùng cần xem. Kiến trúc microservices cho phép scale từng thành phần độc lập tùy theo nhu cầu.

Tích hợp AI và xử lý ảnh

Tính năng dịch truyện tự động đòi hỏi sự tích hợp phức tạp giữa nhiều công nghệ. OCR cần đủ chính xác để nhận dạng text trong nhiều font chữ và layout khác nhau. Xử lý ảnh bằng OpenCV cần tách text khỏi background một cách sạch sẽ. API dịch thuật cần đủ tốt để dịch đúng ngữ cảnh, đặc biệt với các thuật ngữ chuyên ngành trong manga.

Thách thức nằm ở việc xử lý các trường hợp edge case: text viết dọc, text curve theo hình dạng, text chồng lên nhau, hoặc text với font chữ đặc biệt. Hệ thống cần có cơ chế fallback, cho phép Creator can thiệp thủ công khi kết quả tự động không đạt yêu cầu.

Quản lý quyền truy cập đa cấp

Hệ thống phân quyền có nhiều cấp độ: cấp hệ thống với bốn vai trò chính, và cấp team với hai vai trò nội bộ. Mỗi request đều phải kiểm tra quyền truy cập tương ứng. Một Creator có thể là Admin của team A nhưng chỉ là Member của team B, và quyền hạn của họ đối với mỗi truyện phụ thuộc vào nhiều yếu tố.

Việc thiết kế schema database và logic kiểm tra quyền cần được cân nhắc kỹ để đảm bảo tính bảo mật mà vẫn duy trì hiệu năng tốt. Sử dụng middleware để kiểm tra quyền ở tầng backend, kết hợp với việc ẩn hiện các chức năng ở frontend giúp đảm bảo tính nhất quán.

2.1.4. Luồng dữ liệu chính

Luồng đọc truyện

Người dùng truy cập trang chủ, danh sách truyện được load từ cache hoặc database. Khi chọn một truyện, thông tin chi tiết và danh sách chương được hiển thị. Người dùng chọn

chương muốn đọc, hệ thống kiểm tra quyền truy cập, sau đó trả về danh sách URL ảnh từ các server khác nhau. Frontend load ảnh mã hóa từ server, module WebAssembly giải mã và hiển thị ảnh cho người dùng. Nếu ảnh từ server hiện tại lỗi, hệ thống tự động chuyển sang server khác và retry.

Luồng đăng truyện với ảnh đã dịch

Creator tạo truyện mới với các thông tin cơ bản. Sau đó tạo chương mới và upload ảnh thông qua giao diện kéo thả. Ảnh được gửi lên backend, hệ thống xử lý tuần tự: thêm watermark với thông tin team và uploader, thêm banner quảng bá nếu cần, merge nhiều ảnh thành một nếu được cấu hình. Ảnh sau xử lý được mã hóa bằng Rust service, sau đó upload lên các image server. Metadata của chương được lưu vào database, bao gồm URL ảnh trên từng server. Creator preview kết quả và publish khi hài lòng.

Luồng dịch truyện tự động

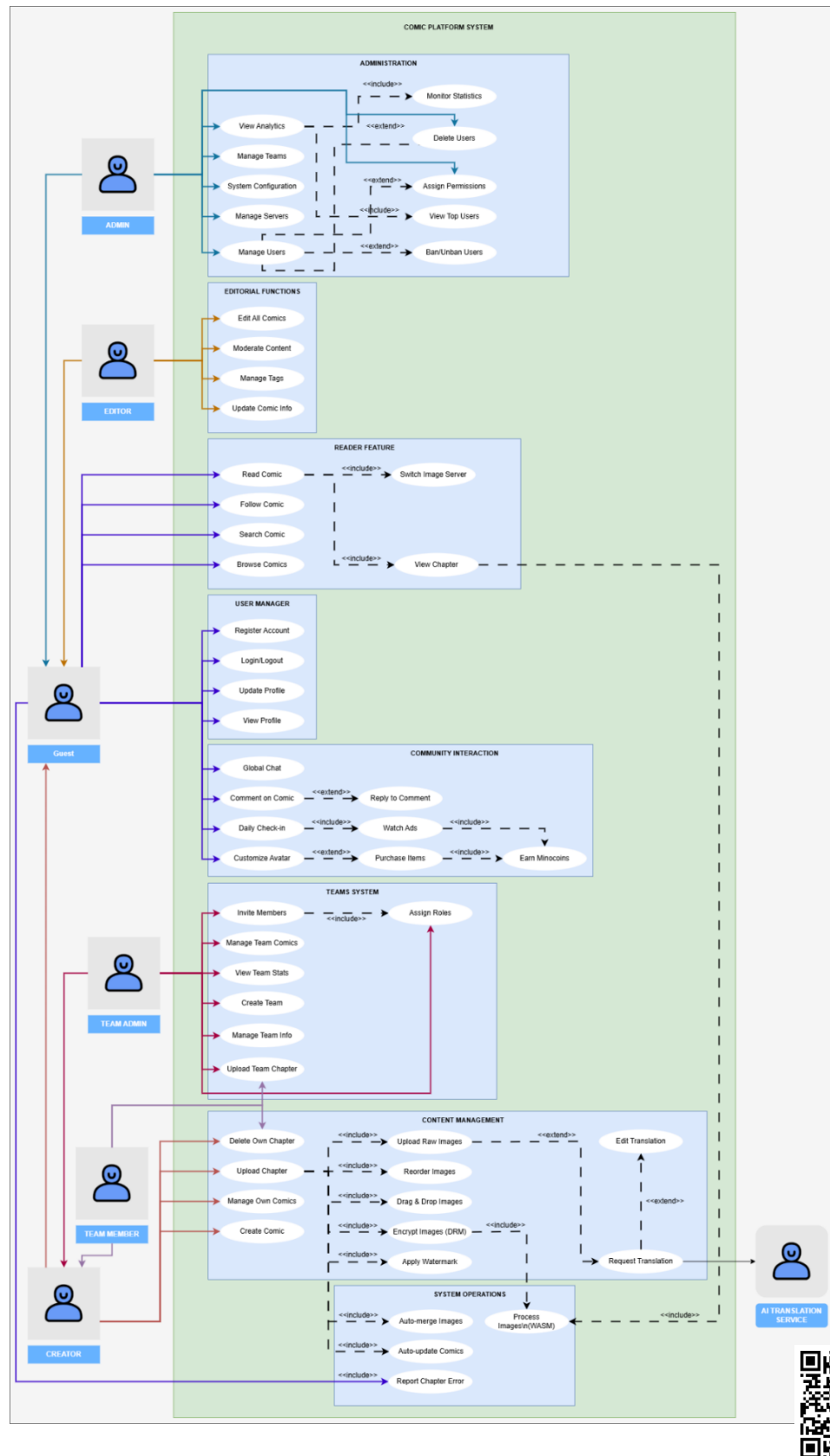
Creator upload ảnh raw và chọn ngôn ngữ nguồn và đích. Ảnh được gửi đến Python service xử lý. Service sử dụng OpenCV để tiền xử lý ảnh, tăng độ tương phản, giảm noise. PaddleOCR nhận dạng vị trí và nội dung text trong ảnh. Danh sách text được gửi đến ChatGPT API để dịch sang ngôn ngữ đích. Tùy vào workflow được chọn, service trả về danh sách tọa độ và text đã dịch, hoặc ảnh đã được xử lý hoàn chỉnh với text gốc bị xóa và text dịch được điền vào đúng vị trí. Creator kiểm tra kết quả, chỉnh sửa nếu cần, sau đó chuyển sang bước xử lý ảnh như workflow thông thường.

Luồng tương tác cộng đồng

Khi người dùng gửi bình luận, dữ liệu được validate ở frontend và backend, sau đó lưu vào database. WebSocket broadcast tin nhắn mới đến tất cả người dùng đang xem cùng chương. Đối với chat toàn server, tin nhắn được broadcast đến tất cả người dùng online. NestJS Cache được sử dụng để cache các tin nhắn gần đây, giảm số lần truy vấn database. Khi người dùng điểm danh, hệ thống kiểm tra xem họ đã điểm danh hôm nay chưa, hiển thị quảng cáo, sau khi hoàn thành cộng Mirucoins vào tài khoản và ghi log giao dịch.

2.2. Mô hình hóa yêu cầu

• Usecase tổng quát

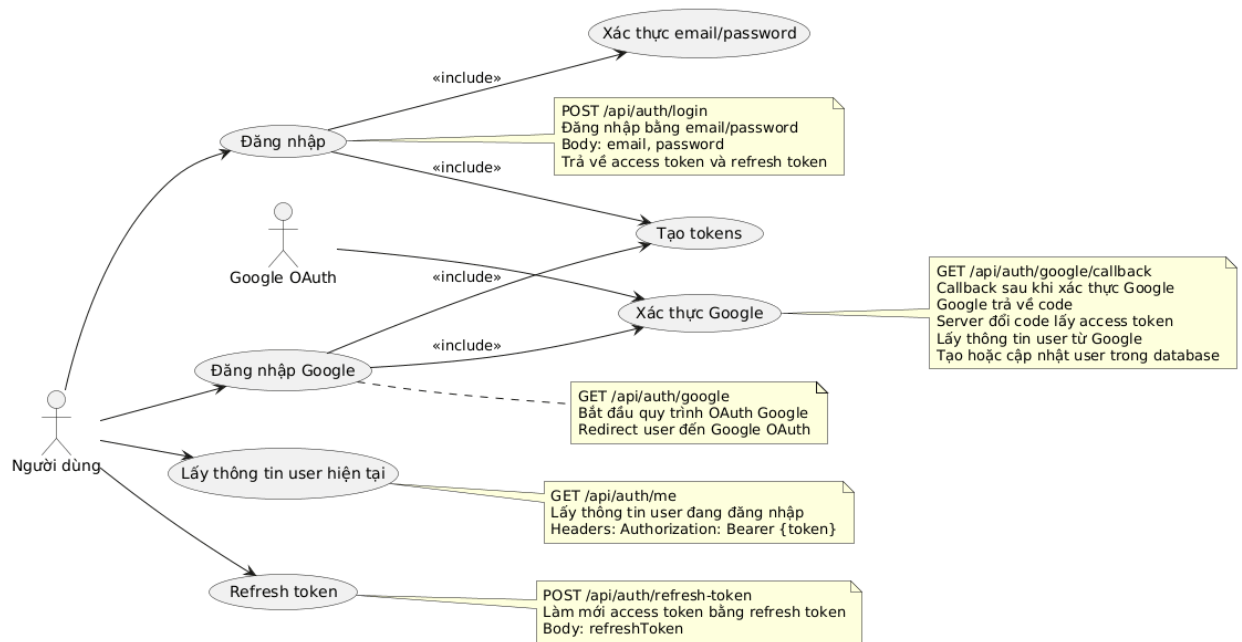


(Liên kết xem chi tiết)

Hình 2.12 Usecase tổng quát

2.2.1. Chức năng xác thực

- Sơ đồ usecase chức năng xác thực



Hình 2.13 Sơ đồ usecase chức năng xác thực

Bảng 2.4 mô tả Usecase chức năng xác thực

Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Đăng nhập	Người dùng	Đăng nhập bằng email và mật khẩu	Có tài khoản trong hệ thống	Nhận access token + refresh token
Lấy thông tin user hiện tại	Người dùng	Xem thông tin cá nhân của tài khoản đang đăng nhập	Đã đăng nhập	Thông tin chi tiết của user
Refresh token	Người dùng (tự động)	Làm mới access token khi hết hạn	Có refresh token còn hiệu lực	Access token mới
Đăng nhập bằng Google	Người dùng + Google	Đăng nhập hoặc đăng ký nhanh bằng tài khoản Google	Có tài khoản Google	Đăng nhập thành công, nhận tokens

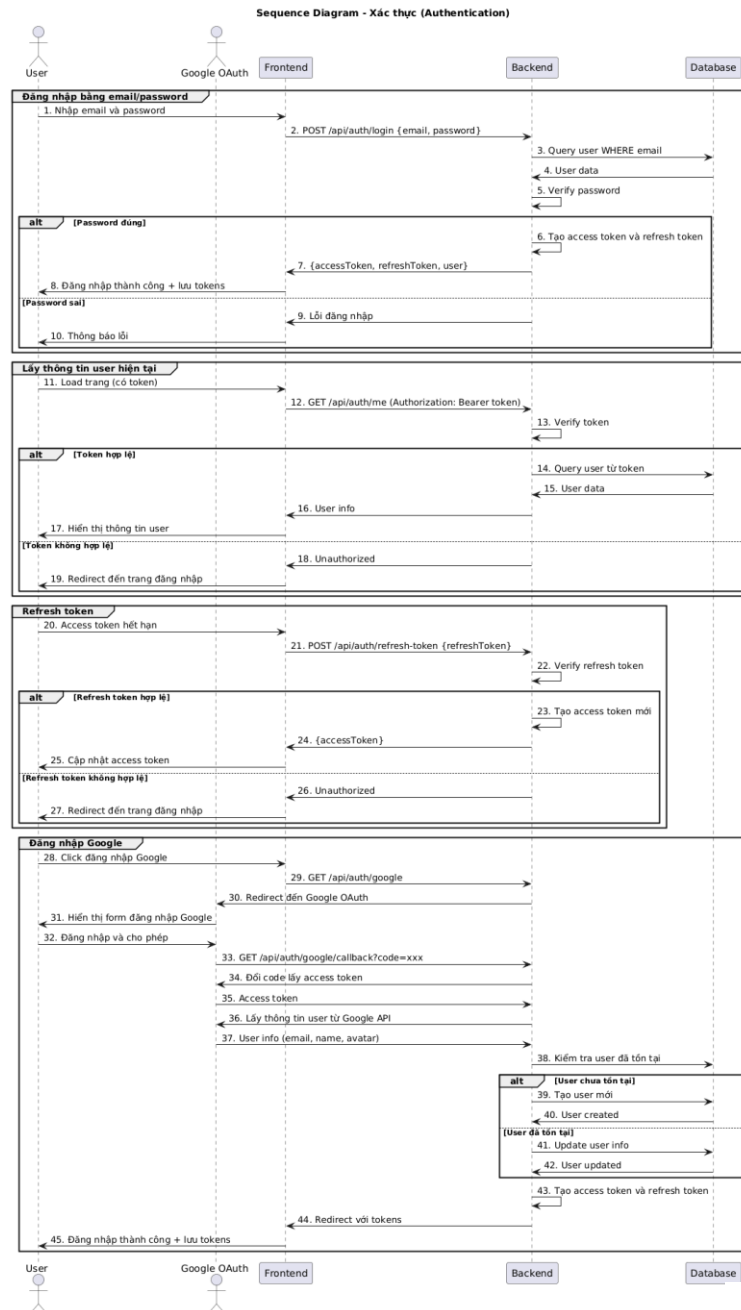
Xác thực email/password	Hệ thống	Kiểm tra email và mật khẩu hợp lệ	Nhận email + password	Xác thực thành công hoặc báo lỗi
Tạo tokens	Hệ thống	Tạo access token và refresh token cho user	Có thông tin user hợp lệ	Trả về cặp token
Xác thực Google	Hệ thống + Google	Đổi code lấy thông tin người dùng từ Google	Nhận được code từ Google	Lấy được email, tên, avatar từ Google

Bảng 2.5 Dòng chính Usecase chức năng xác thực

STT	Use Case	Thực hiện	Hành động chính
1	Đăng nhập	Người dùng	Nhập email + mật khẩu → gửi yêu cầu đăng nhập
2	Đăng nhập	Hệ thống	Kiểm tra email/password → tạo và trả về access + refresh token
3	Lấy thông tin user	Người dùng	Gửi yêu cầu kèm access token
4	Lấy thông tin user	Hệ thống	Xác thực token → trả về thông tin user hiện tại
5	Refresh token	Người dùng	Gửi refresh token khi access token hết hạn
6	Refresh token	Hệ thống	Kiểm tra refresh token → trả access token mới
7	Đăng nhập Google	Người dùng	Nhấn nút “Đăng nhập với Google”
8	Đăng nhập Google	Hệ thống	Chuyển hướng đến trang xác thực Google
9	Đăng nhập Google	Người dùng/Google	Người dùng cấp quyền → Google trả về code
10	Đăng nhập Google	Hệ thống	Đổi code lấy thông tin user → tạo/tìm tài khoản → tạo tokens

11	Tạo tokens	Hệ thống	Tạo access token (ngắn hạn) và refresh token (dài hạn)
12	Xác thực Google	Hệ thống	Dùng code gọi Google API → lấy email, tên, ảnh đại diện

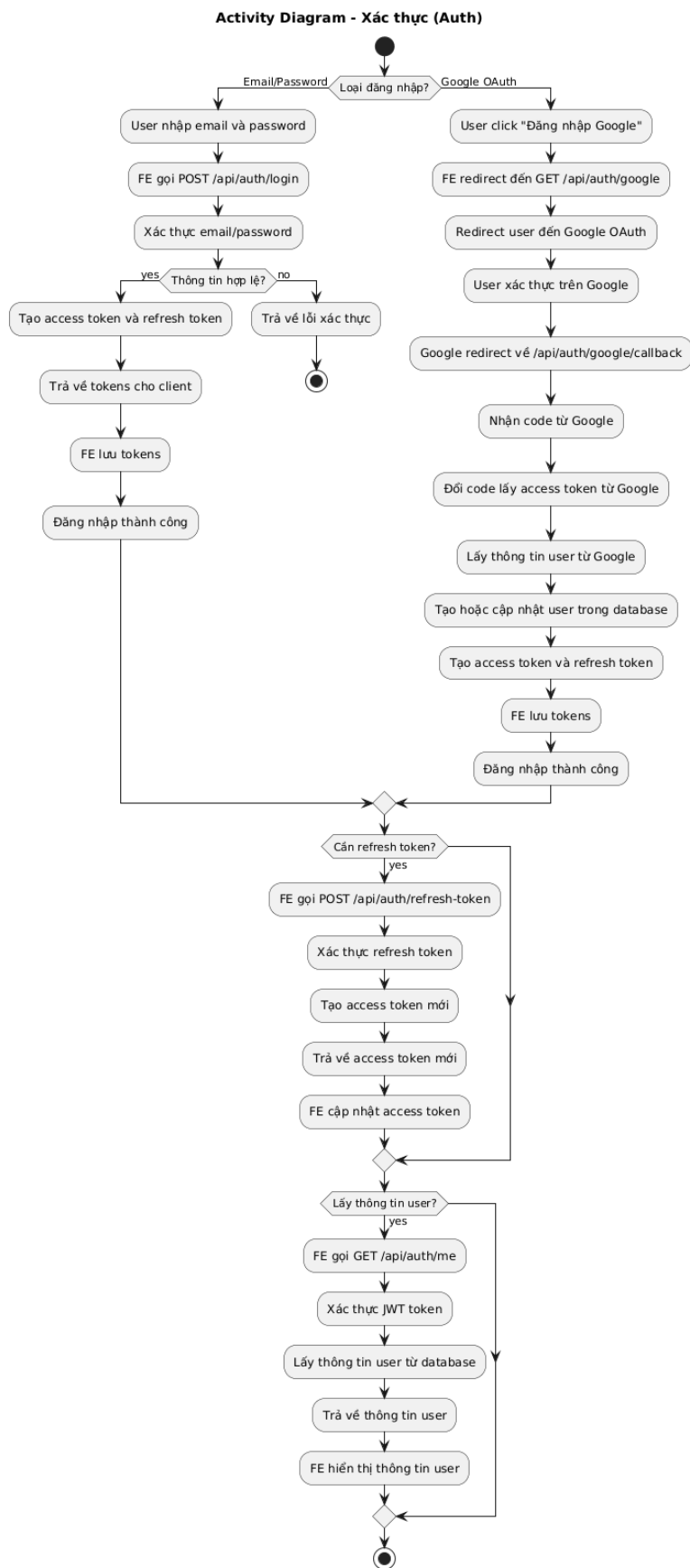
• Sơ đồ hoạt động chức năng xác thực



([Liên kết](#) xem chi tiết)

Hình 2.14 Sơ đồ tuần tự chức năng xác thực

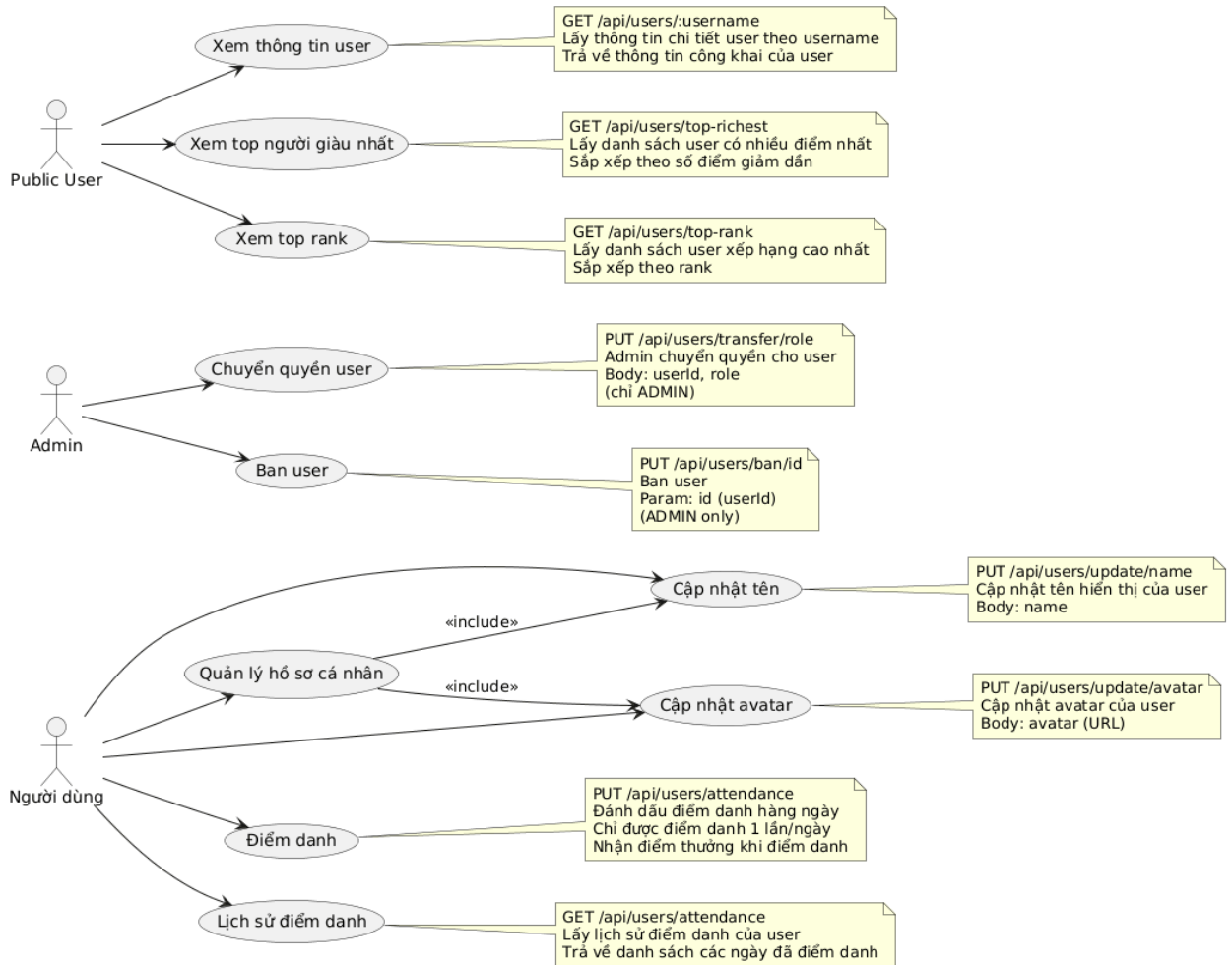
- Sơ đồ hoạt động chức năng xác thực



Hình 2.15 Sơ đồ hoạt động chức năng xác thực

2.2.2. Chức năng cơ bản người dùng

- Sơ đồ usecase chức năng cơ bản người dùng



Hình 2.16 Sơ đồ usecase chức năng cơ bản người dùng

Bảng 2.6 Mô tả Usecase chức năng cơ bản người dùng

Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý hồ sơ cá nhân	Người dùng	Chỉnh sửa thông tin cá nhân (tên, avatar)	Đã đăng nhập	Hồ sơ được cập nhật thành công
Cập nhật tên	Người dùng	Thay đổi tên hiển thị	Đã đăng nhập	Tên mới được lưu
Cập nhật avatar	Người dùng	Thay đổi ảnh đại diện	Đã đăng nhập	Avatar mới được cập nhật

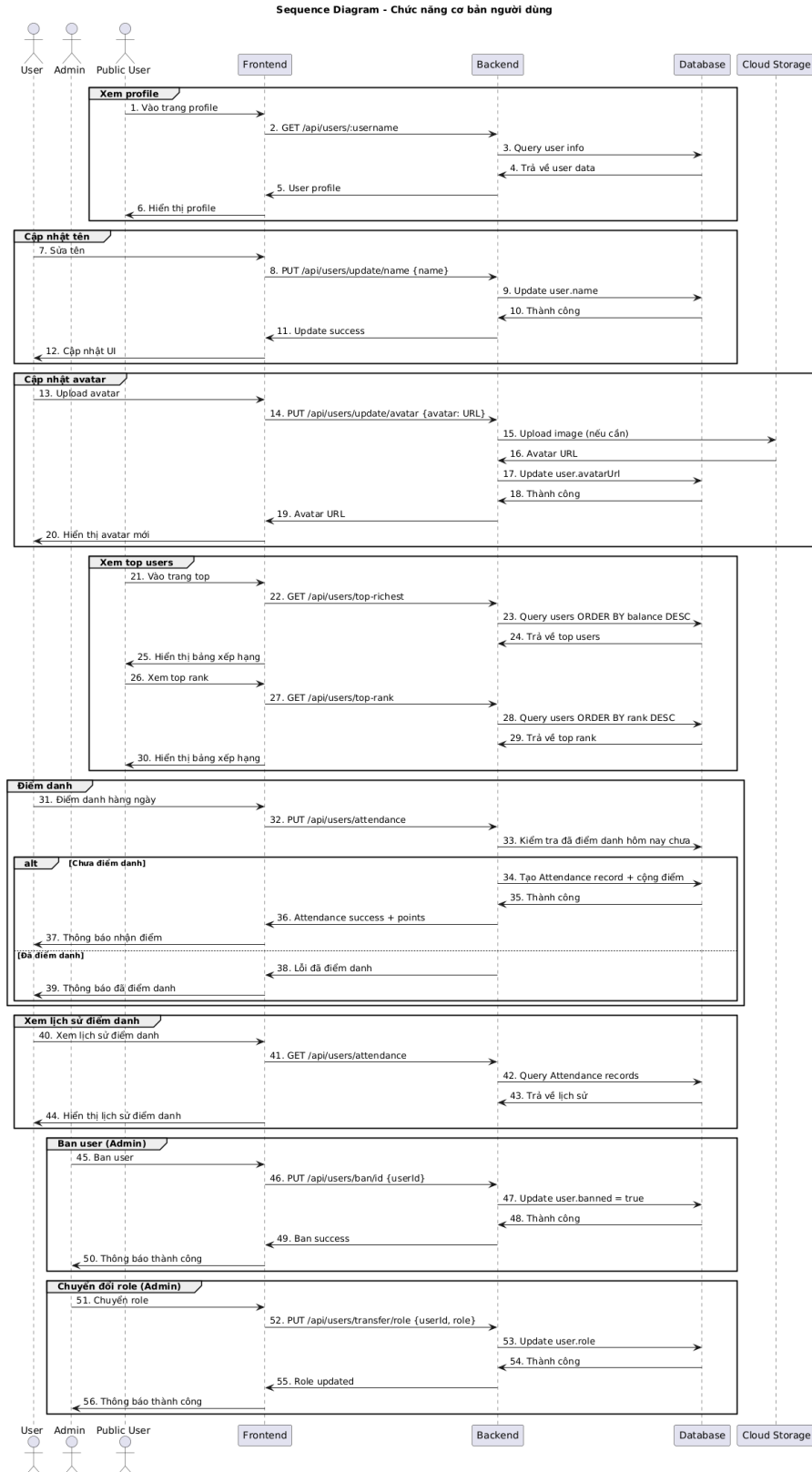
Xem thông tin user	Public User	Xem thông tin công khai của một user bất kỳ	Không cần đăng nhập	Hiển thị thông tin user (username, điểm, rank...)
Xem top người giàu nhất	Public User	Xem bảng xếp hạng người có nhiều điểm nhất	Không cần đăng nhập	Danh sách user giàu nhất
Xem top rank	Public User	Xem bảng xếp hạng theo thứ hạng	Không cần đăng nhập	Danh sách user rank cao nhất
Chuyển quyền user	Admin	Admin nâng cấp hoặc thay đổi role cho user	Là Admin	User nhận role mới
Ban user	Admin	Cấm tài khoản người dùng	Là Admin	User bị ban, không thể đăng nhập
Điểm danh	Người dùng	Điểm danh hàng ngày để nhận thưởng điểm	Đã đăng nhập, chưa điểm danh hôm nay	Nhận điểm thưởng, đánh dấu đã điểm danh
Lịch sử điểm danh	Người dùng	Xem các ngày đã điểm danh và chuỗi điểm danh liên tục	Đã đăng nhập	Hiển thị lịch sử + streak

Bảng 2.7 Dòng chính Usecase chức năng cơ bản người dùng

STT	Use Case	Thực hiện	Hành động chính
1	Quản lý hồ sơ cá nhân	Người dùng	Vào trang hồ sơ → chọn chỉnh sửa tên hoặc avatar
2	Cập nhật tên	Người dùng	Nhập tên mới → gửi yêu cầu cập nhật
3	Cập nhật avatar	Người dùng	Upload hoặc dán link ảnh → gửi yêu cầu cập nhật
4	Xem thông tin user	Public User	Truy cập /user/:username

5	Xem thông tin user	Hệ thống	Trả về thông tin công khai: username, điểm, rank, avatar, ngày tham gia...
6	Xem top người giàu nhất	Public User	Truy cập trang bảng xếp hạng giàu nhất
7	Hệ thống	Hệ thống	Lấy top user theo số điểm giảm dần → hiển thị danh sách
8	Xem top rank	Public User	Truy cập trang bảng xếp hạng
9	Hệ thống	Hệ thống	Lấy top user theo rank → hiển thị danh sách
10	Chuyển quyền user	Admin	Chọn user → chọn role mới → xác nhận
11	Ban user	Admin	Tìm user → nhấn nút Ban → xác nhận lý do (tùy chọn)
12	Điểm danh	Người dùng	Nhấn nút “Điểm danh hôm nay”
13	Điểm danh	Hệ thống	Kiểm tra chưa điểm danh hôm nay → cộng điểm thưởng → cập nhật streak
14	Lịch sử điểm danh	Người dùng	Vào trang điểm danh → xem lịch sử
15	Hệ thống	Hệ thống	Trả về danh sách ngày đã điểm danh + chuỗi hiện tại

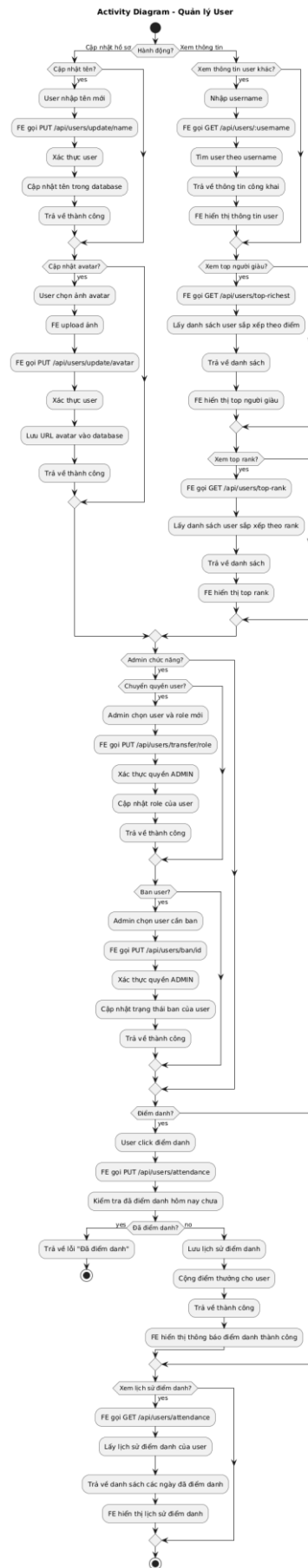
- Sơ đồ tuần tự chức năng cơ bản người dùng



([Liên kết](#) xem chi tiết)

Hình 2.17 Sơ đồ tuần tự chức năng cơ bản người dùng

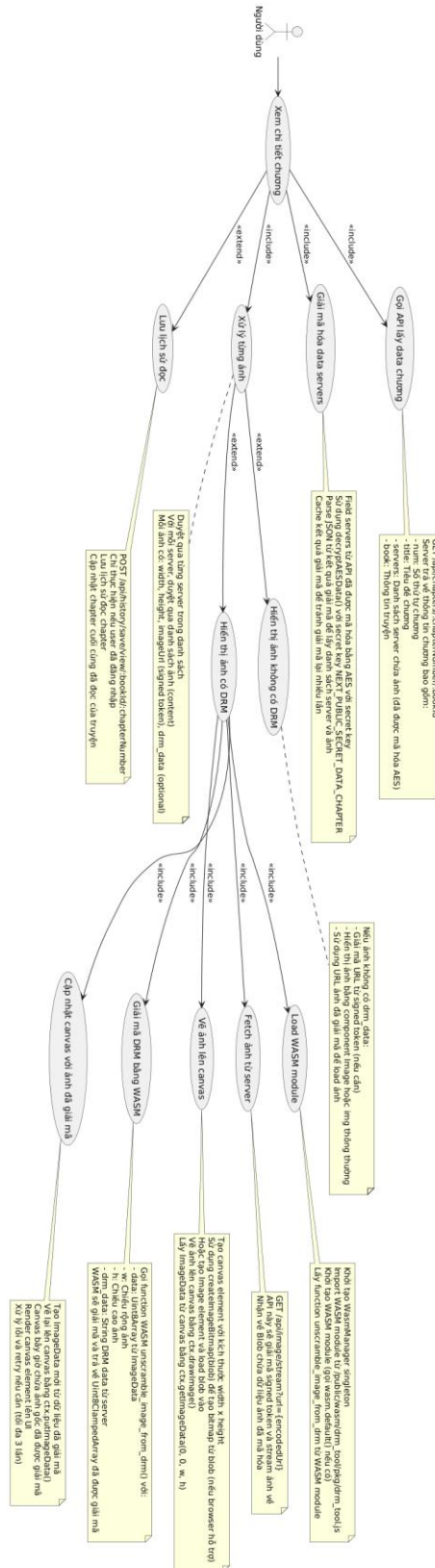
- Sơ đồ hoạt động chức năng cơ bản người dùng



Hình 2.18 Sơ đồ hoạt động chức năng cơ bản người dùng

2.2.3. Chức năng xem chi tiết chương

- Sơ đồ usecase chức năng xem chi tiết chương



(Liên kết để xem chi tiết)

Hình 2.19 Sơ đồ usecase chức năng xem chi tiết chương

Bảng 2.8 Mô tả Usecase chức năng xem chi tiết chương

Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Xem chi tiết chương	Người dùng	Đọc nội dung một chương truyện	Đã vào trang chương	Hiển thị đầy đủ các trang ảnh của chương
Gọi API lấy data chương	Hệ thống	Lấy thông tin chương từ server	Có bookId + chapterNumber	Nhận danh sách server + ảnh đã mã hóa
Giải mã hóa data servers	Hệ thống	Giải mã AES danh sách server bằng secret key	Có dữ liệu mã hóa từ API	Lấy được danh sách URL ảnh thật
Xử lý từng ảnh	Hệ thống	Duyệt và hiển thị từng ảnh trong chương	Có danh sách ảnh	Tất cả ảnh được render đúng
Hiển thị ảnh không có DRM	Hệ thống	Hiển thị ảnh bình thường (không bị scramble)	Ảnh không có drm_data	Ảnh hiển thị ngay lập tức
Hiển thị ảnh có DRM	Hệ thống	Hiển thị ảnh bị DRM bằng WASM	Ảnh có drm_data	Ảnh được giải mã và hiển thị đúng
Load WASM module	Hệ thống	Khởi tạo module WASM giải mã DRM	Lần đầu gặp ảnh DRM	WASM sẵn sàng sử dụng
Fetch ảnh từ server	Hệ thống	Tải dữ liệu ảnh đã mã hóa về	Có URL ảnh	Nhận Blob ảnh mã hóa
Vẽ ảnh lên canvas	Hệ thống	Tạo canvas và vẽ ảnh gốc lên	Có Blob ảnh	Canvas chứa ảnh gốc

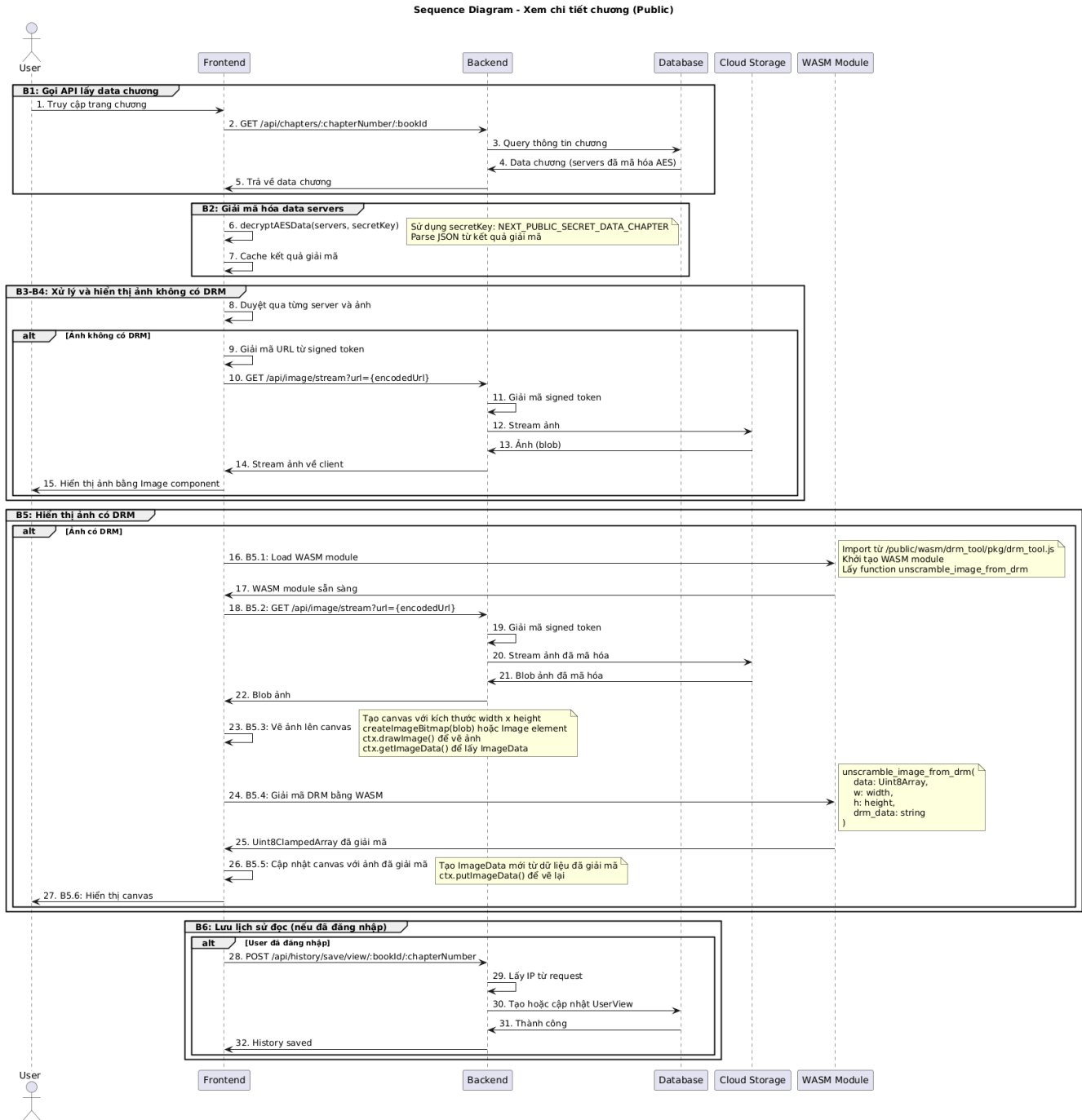
Giải mã DRM bằng WASM	Hệ thống	Gọi WASM để unscramble ảnh	Có ImageData + drm_data	Nhận dữ liệu ảnh đã giải mã
Cập nhật canvas với ảnh giải mã	Hệ thống	Vẽ lại ảnh đã giải mã lên canvas	Có dữ liệu sau khi WASM xử lý	Canvas hiển thị ảnh đẹp, không bị scramble
Lưu lịch sử đọc	Hệ thống	Ghi lại chapter đã đọc (nếu đã đăng nhập)	Người dùng đã đăng nhập	Cập nhật lịch sử đọc truyện

Bảng 2.9 Dòng chính Usecase chức năng xem chi tiết chương

STT	Use Case	Thực hiện	Hành động chính
1	Xem chi tiết chương	Người dùng	Vào trang chương cụ thể
2	Xem chi tiết chương	Hệ thống	Gọi API lấy data chương → nhận dữ liệu đã mã hóa AES
3	Giải mã hóa data servers	Hệ thống	Dùng secret key giải mã AES → lấy danh sách server + ảnh thật
4	Xử lý từng ảnh	Hệ thống	Bắt đầu duyệt từng ảnh trong danh sách
5	Hiển thị ảnh không có DRM	Hệ thống	Nếu ảnh không có drm_data → fetch ảnh bình thường → hiển thị trực tiếp
6	Hiển thị ảnh có DRM	Hệ thống	Nếu ảnh có drm_data → thực hiện luồng DRM
7	Load WASM module	Hệ thống	Khởi tạo WASM module (chỉ lần đầu)
8	Fetch ảnh từ server	Hệ thống	Tải ảnh về dưới dạng Blob (đã mã hóa/scramble)
9	Vẽ ảnh lên canvas	Hệ thống	Tạo canvas → vẽ Blob lên canvas → lấy ImageData
10	Giải mã DRM bằng WASM	Hệ thống	Gọi hàm WASM unscramble → truyền ImageData + drm_data

11	Cập nhật canvas	Hệ thống	Nhận dữ liệu đã giải mã → putImageData lên canvas lại
12	Hiển thị ảnh cuối cùng	Hệ thống	Canvas hiển thị ảnh đẹp, không bị scramble → render ra giao diện
13	Lưu lịch sử đọc	Hệ thống	Nếu user đã đăng nhập → tự động lưu lịch sử đọc chương hiện tại
14	Hoàn tất	Người dùng	Đọc truyện mượt, không bị ảnh lỗi, lịch sử được lưu

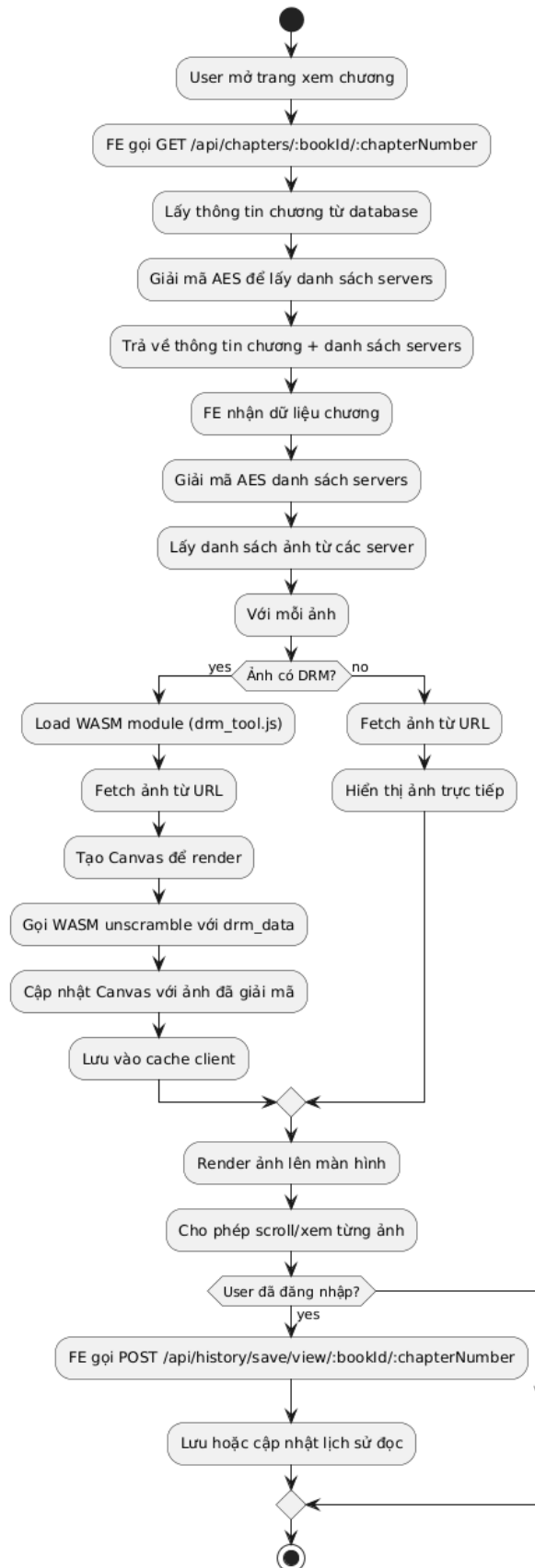
- Sơ đồ tuần tự chức năng xem chi tiết chương



Hình 2.20 Sơ đồ tuần tự chức năng xem chi tiết chương

- Sơ đồ hoạt động chức năng xem chi tiết chương

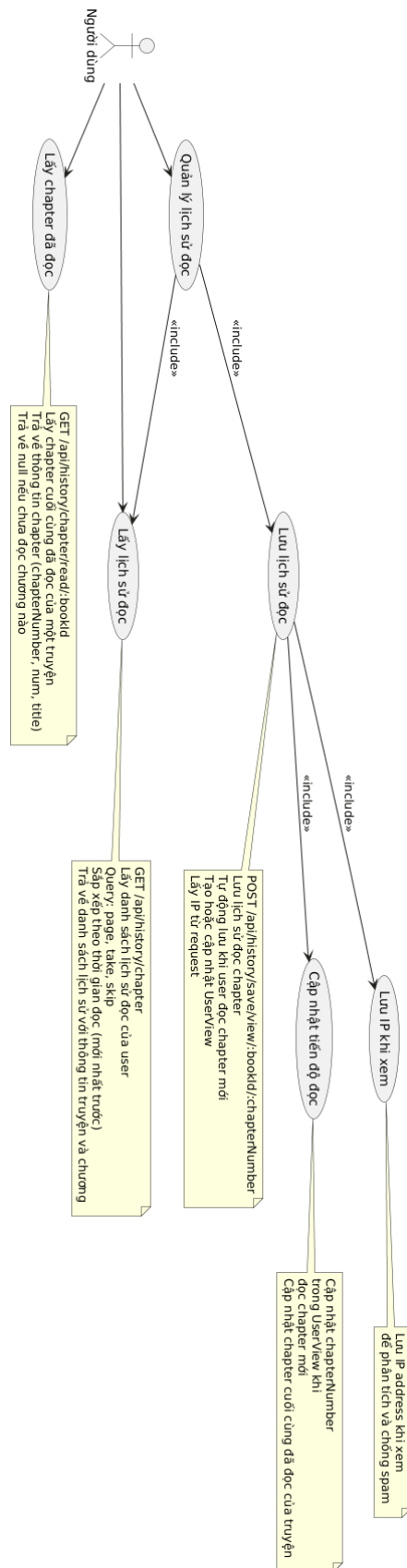
Activity Diagram - Xem chi tiết chương (Public)



Hình 2.21 Sơ đồ hoạt động chức năng xem chi tiết chương

2.2.4. Chức năng quản lý lịch sử đọc

- Sơ đồ usecase quản lý lịch sử đọc



Hình 2.22 Sơ đồ usecase quản lý lịch sử đọc

Bảng 2.10 Mô tả Usecase quản lí lịch sử đọc

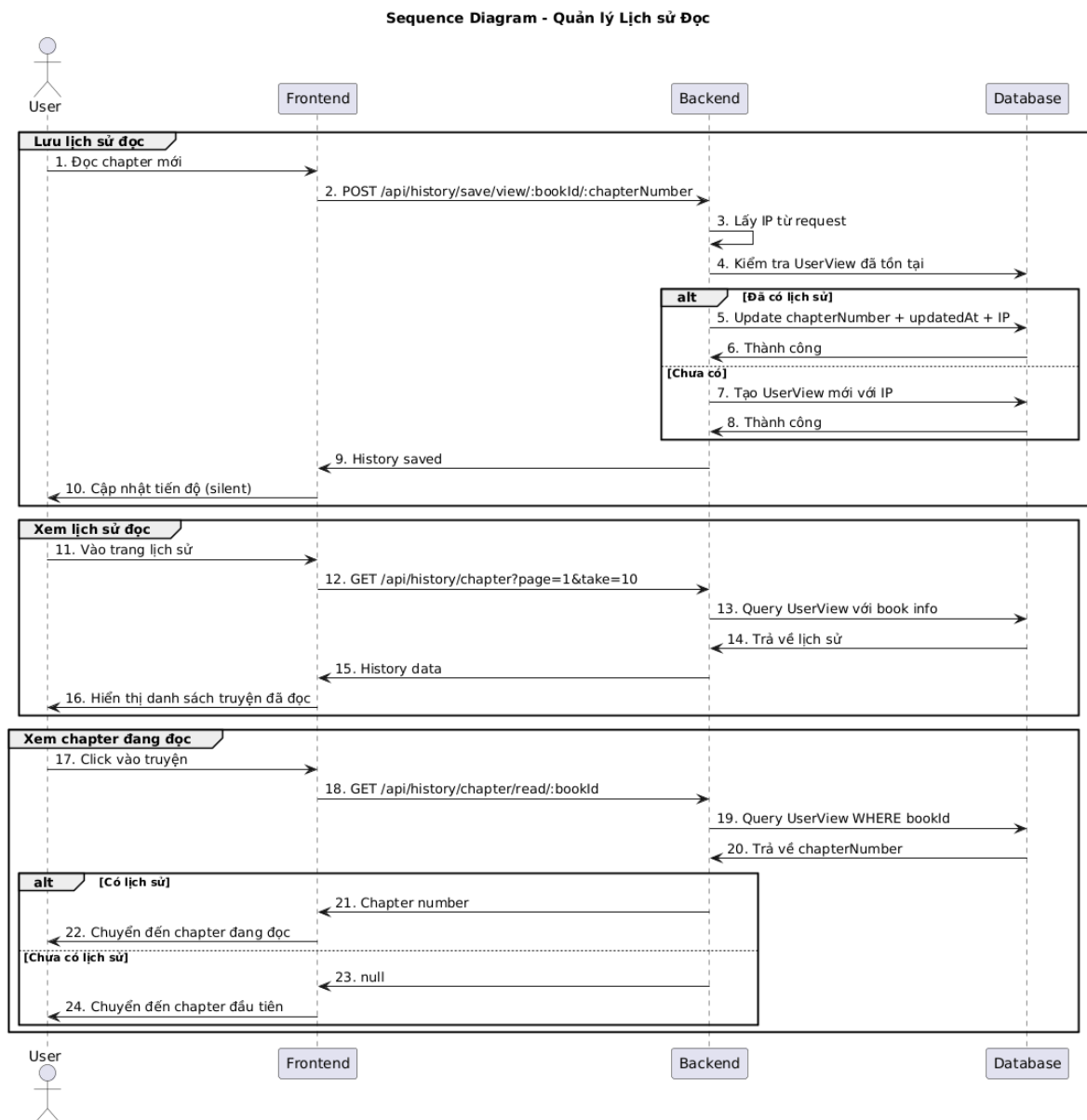
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý lịch sử đọc	Người dùng	Tự động lưu và hiển thị lịch sử đọc truyện	Đã đăng nhập	Lịch sử luôn chính xác, cập nhật realtime
Lưu lịch sử đọc	Hệ thống	Ghi lại mỗi khi user đọc một chapter	User mở chapter	Tạo/cập nhật bản ghi lịch sử đọc
Lấy lịch sử đọc	Người dùng	Xem danh sách các truyện đã đọc gần đây	Đã đăng nhập	Hiển thị danh sách lịch sử (mới nhất trước)
Lấy chapter đã đọc	Người dùng	Biết chapter cuối cùng đã đọc của một truyện	Đã từng đọc truyện đó	Trả về chương gần nhất hoặc null
Lưu IP khi xem	Hệ thống	Ghi lại IP để chống spam và phân tích hành vi	User xem chapter	IP được lưu cùng bản ghi lịch sử
Cập nhật tiến độ đọc	Hệ thống	Cập nhật chapter mới nhất đã đọc của truyện	User đọc chapter mới hơn	Tiến độ truyện luôn đúng

Bảng 2.11 Dòng chính Usecase quản lí lịch sử đọc

STT	Use Case	Thực hiện	Hành động chính
1	Quản lý lịch sử đọc	Người dùng	Mở một chapter bất kỳ để đọc
2	Lưu lịch sử đọc	Hệ thống	Tự động gọi lưu lịch sử khi chapter được load
3	Lưu IP khi xem	Hệ thống	Lấy IP từ request → lưu cùng bản ghi lịch sử
4	Cập nhật tiến độ đọc	Hệ thống	So sánh chapter hiện tại với chapter đã lưu → nếu mới hơn thì cập nhật
5	Lấy lịch sử đọc	Người dùng	Vào trang “Lịch sử đọc” hoặc “Đã đọc”

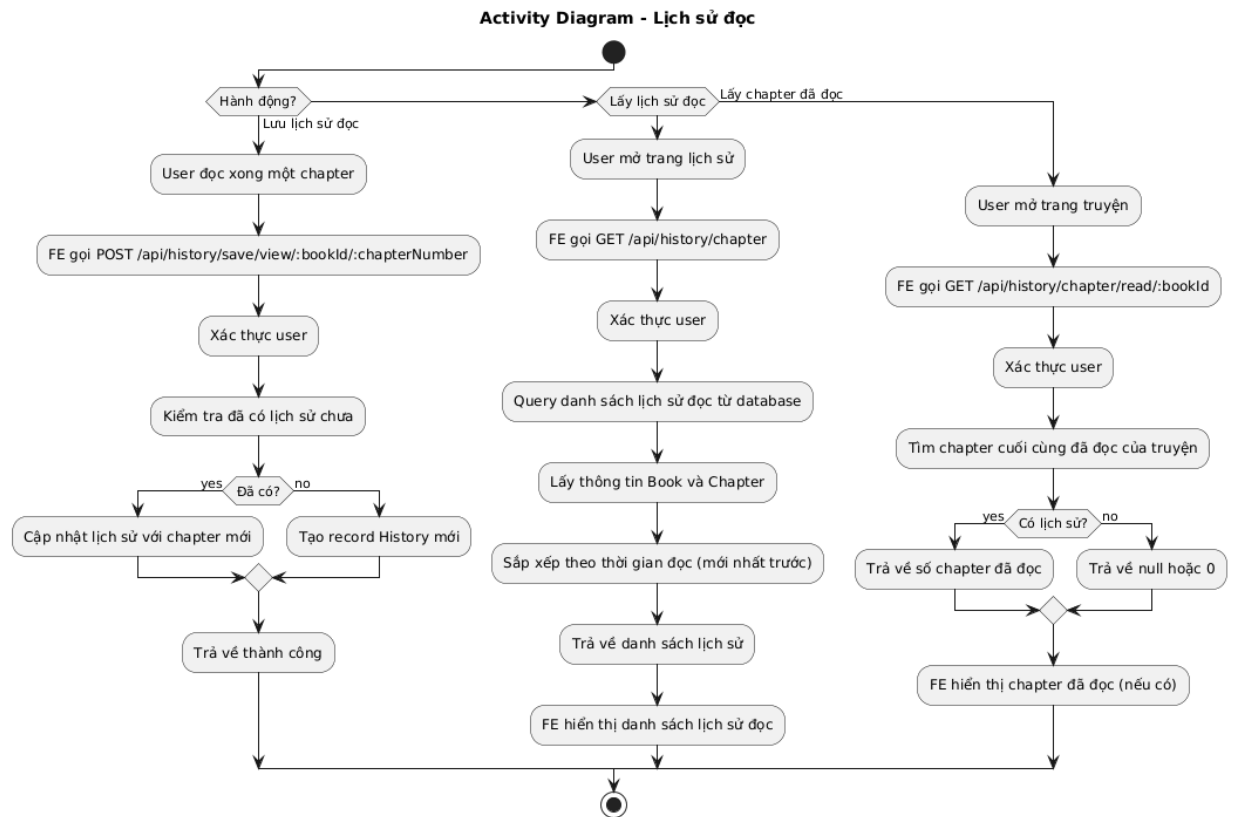
6	Lấy lịch sử đọc	Hệ thống	Trả về danh sách các truyện đã đọc → sắp xếp mới nhất trước
7	Lấy chapter đã đọc	Người dùng	Mở lại một truyện đã từng đọc
8	Lấy chapter đã đọc	Hệ thống	Kiểm tra lịch sử → trả về chapter cuối cùng đã đọc (hoặc null nếu chưa đọc)
9	Hoàn tất	Người dùng	Tiếp tục đọc đúng từ chapter đã bỏ dở → không bị mất tiến độ

- Sơ đồ tuần tự quản lí lịch sử đọc



Hình 2.23 Sơ đồ tuần tự quản lí lịch sử đọc

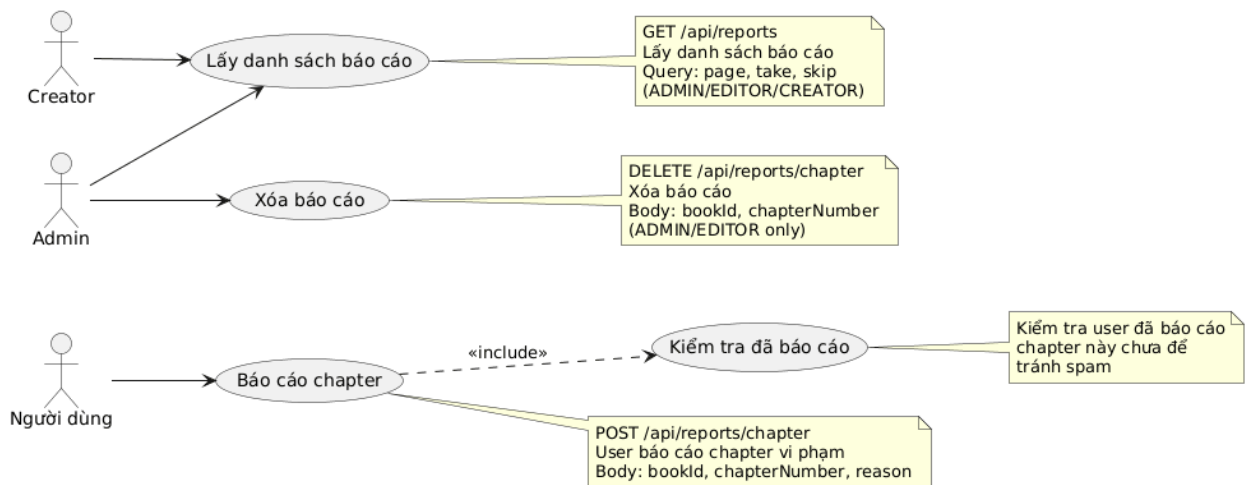
- Sơ đồ hoạt động chức năng quản lý lịch sử đọc



Hình 2.24 Sơ đồ hoạt động chức năng quản lý lịch sử đọc

2.2.5. Chức năng báo cáo chương

- Sơ đồ usecase báo cáo chương



Hình 2.25 Sơ đồ usecase báo cáo chương

Bảng 2.12 Mô tả Usecase báo cáo chương

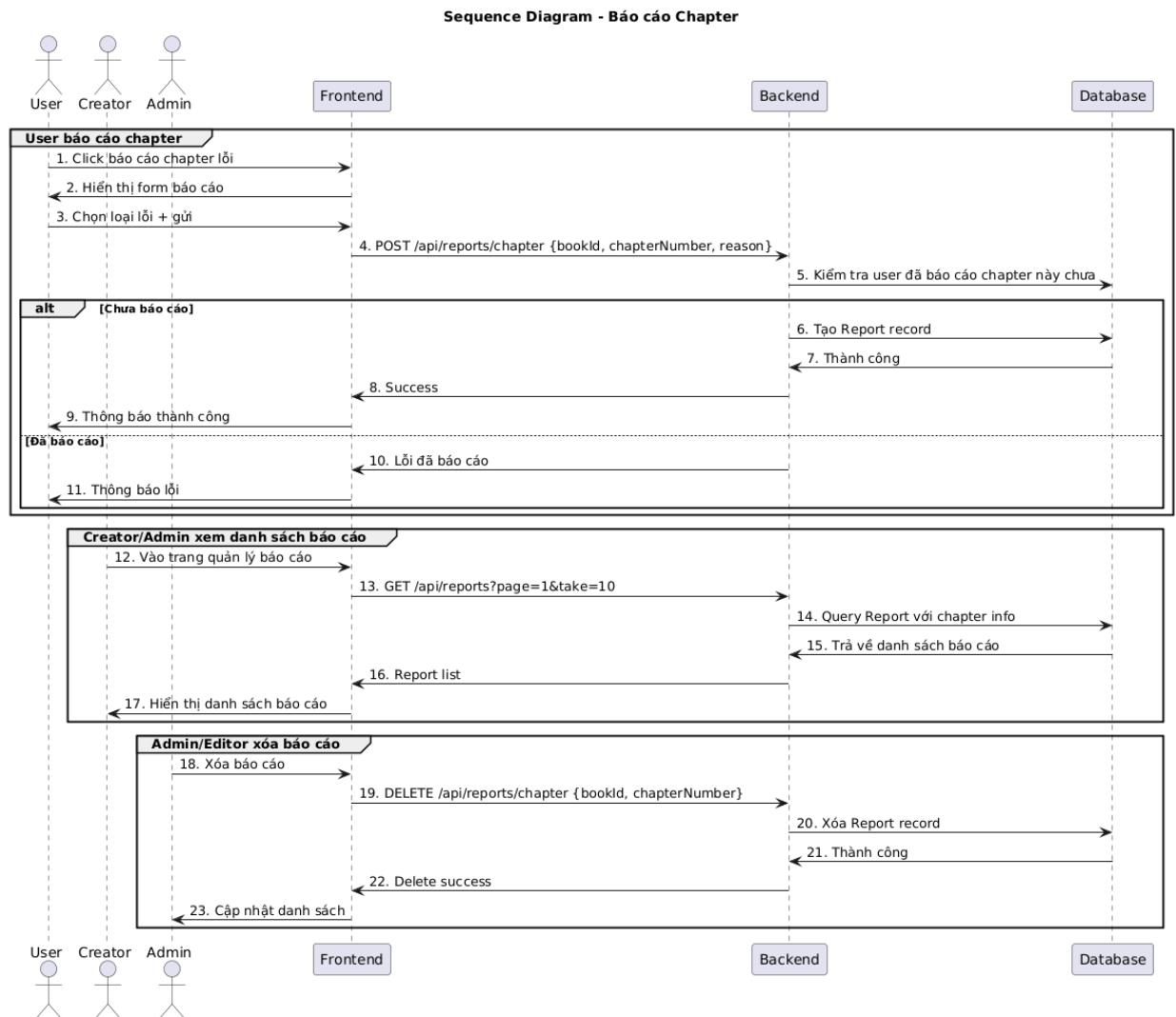
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Báo cáo chapter	Người dùng	Báo cáo chapter vi phạm nội dung	Đã đăng nhập	Báo cáo được gửi thành công
Lấy danh sách báo cáo	Creator / Admin	Xem tất cả báo cáo đang chờ xử lý	Là Creator hoặc Admin	Hiển thị danh sách báo cáo phân trang
Xóa báo cáo	Admin	Xóa hoặc đánh dấu đã xử lý một báo cáo	Là Admin	Báo cáo bị xóa khỏi danh sách
Kiểm tra đã báo cáo	Hệ thống	Ngăn user báo cáo trùng nhiều lần cùng một chapter	User từng báo cáo chapter đó	Không cho gửi báo cáo trùng (tránh spam)

Bảng 2.13 Dòng chính Usecase báo cáo chương

STT	Use Case	Thực hiện	Hành động chính
1	Báo cáo chapter	Người dùng	Vào chapter → nhấn nút “Báo cáo”
2	Báo cáo chapter	Hệ thống	Kiểm tra đã báo cáo chapter này chưa
3	Kiểm tra đã báo cáo	Hệ thống	Nếu đã báo cáo → thông báo “Bạn đã báo cáo chapter này rồi”
4	Báo cáo chapter	Hệ thống	Nếu chưa → lưu báo cáo mới + lý do + bookId + chapterNumber
5	Báo cáo chapter	Người dùng	Nhận thông báo “Báo cáo thành công”
6	Lấy danh sách báo cáo	Creator / Admin	Vào trang quản lý báo cáo
7	Lấy danh sách báo cáo	Hệ thống	Trả về danh sách báo cáo → sắp xếp mới nhất trước → phân trang
8	Xóa báo cáo	Admin	Chọn một hoặc nhiều báo cáo → nhấn “Xóa” hoặc “Đã xử lý”

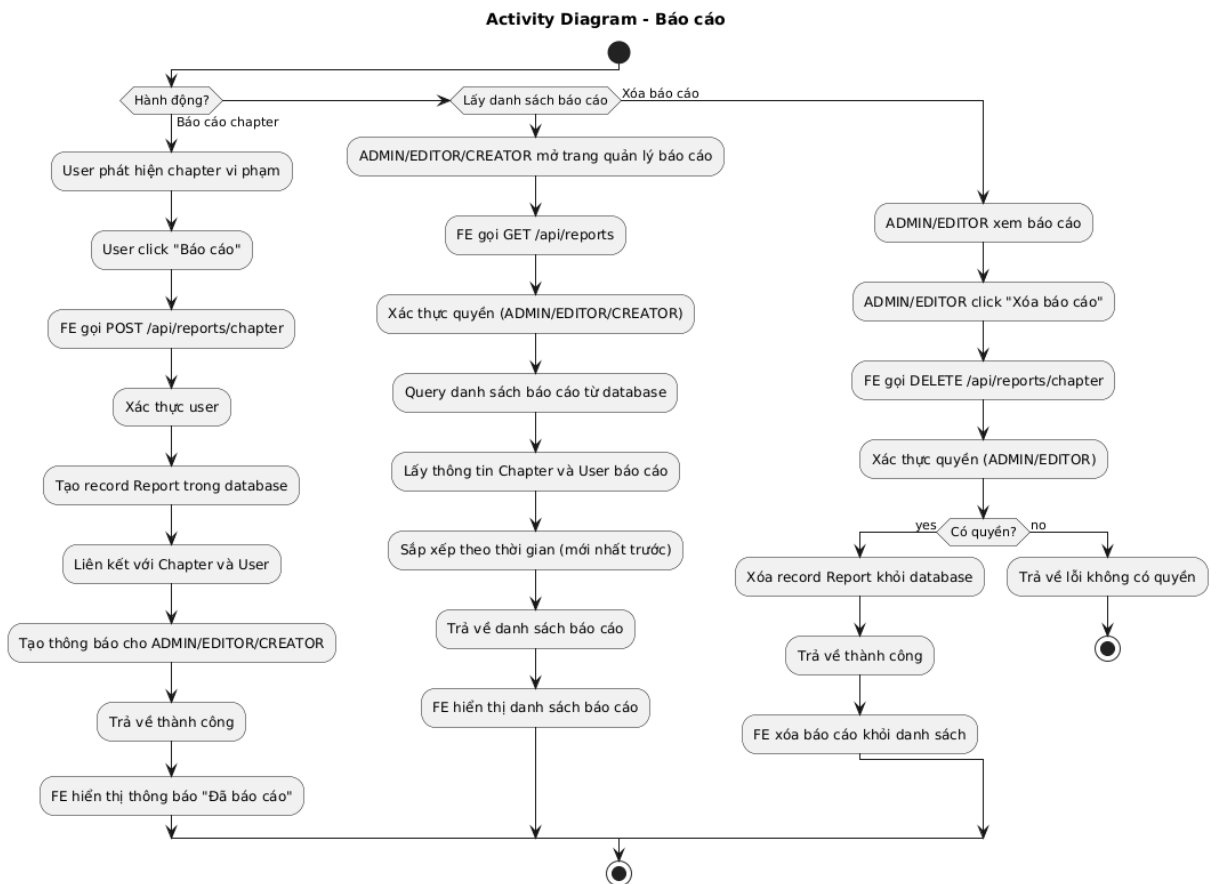
9	Xóa báo cáo	Hệ thống	Xóa báo cáo khỏi database → cập nhật lại danh sách
10	Hoàn tất	Người dùng	Nội dung xấu được báo cáo nhanh → được xử lý kịp thời

• Sơ đồ tuần tự báo cáo chương



Hình 2.26 Sơ đồ tuần tự báo cáo chương

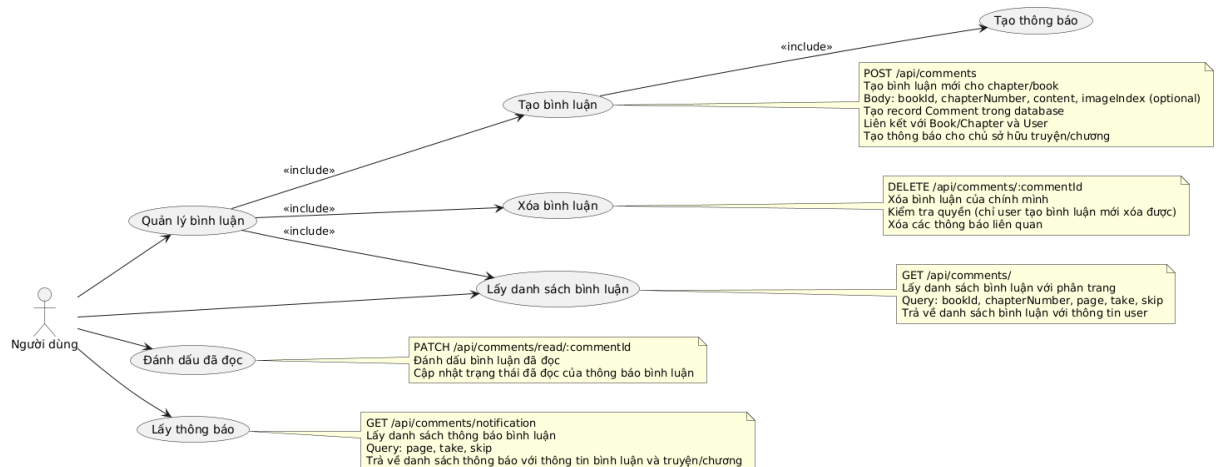
• Sơ đồ hoạt động chức năng báo cáo chương



Hình 2.27 Sơ đồ hoạt động chức năng báo cáo chương

2.3.6. Chức năng bình luận

• Sơ đồ usecase chức năng bình luận



Hình 2.28 Sơ đồ usecase chức năng bình luận

Bảng 2.14 Mô tả Usecase chức năng bình luận

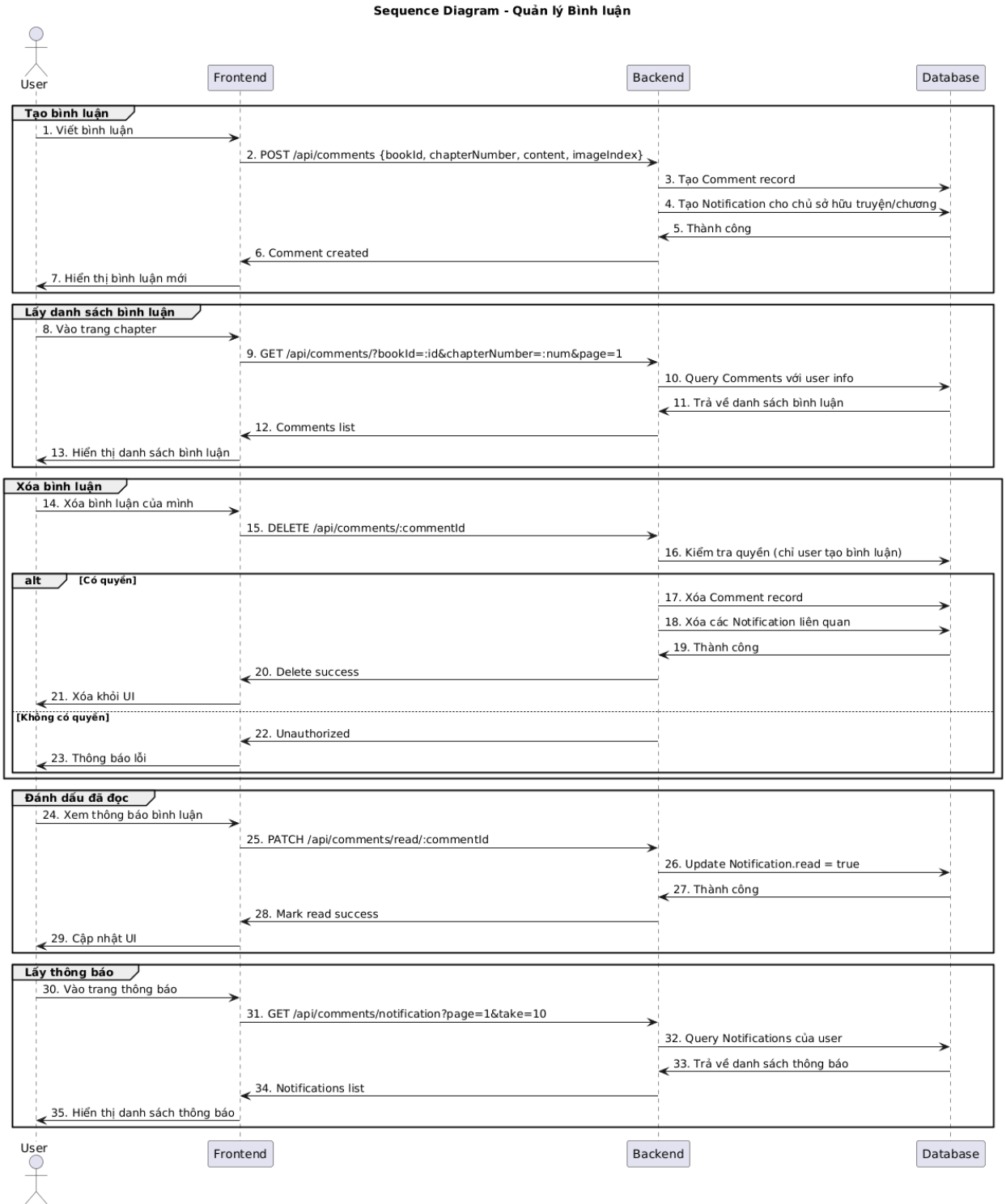
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý bình luận	Người dùng	Tạo, xem và xóa bình luận dưới truyện/chương	Đã đăng nhập	Bình luận hoạt động mượt mà, có thông báo
Tạo bình luận	Người dùng	Viết bình luận mới	Đã đăng nhập	Bình luận được lưu + gửi thông báo cho chủ truyện
Lấy danh sách bình luận	Người dùng	Xem tất cả bình luận của một chapter	Không bắt buộc đăng nhập	Hiển thị danh sách bình luận phân trang
Xóa bình luận	Người dùng	Xóa bình luận do chính mình tạo	Là chủ bình luận	Bình luận bị xóa, thông báo liên quan cũng xóa
Đánh dấu đã đọc	Người dùng	Đánh dấu thông báo bình luận đã được xem	Đã đăng nhập	Thông báo chuyển thành đã đọc
Lấy thông báo	Người dùng	Xem các thông báo bình luận mới	Đã đăng nhập	Hiển thị danh sách thông báo chưa đọc
Tạo thông báo	Hệ thống	Tự động thông báo khi có người bình luận	Có bình luận mới	Chủ truyện/creator nhận thông báo kịp thời

Bảng 2.15 Dòng chính Usecase chức năng bình luận

STT	Use Case	Thực hiện	Hành động chính
1	Quản lý bình luận	Người dùng	Vào chapter → thấy khung bình luận
2	Tạo bình luận	Người dùng	Nhập nội dung → nhấn Gửi
3	Tạo bình luận	Hệ thống	Lưu bình luận → tạo thông báo cho chủ truyện/creator

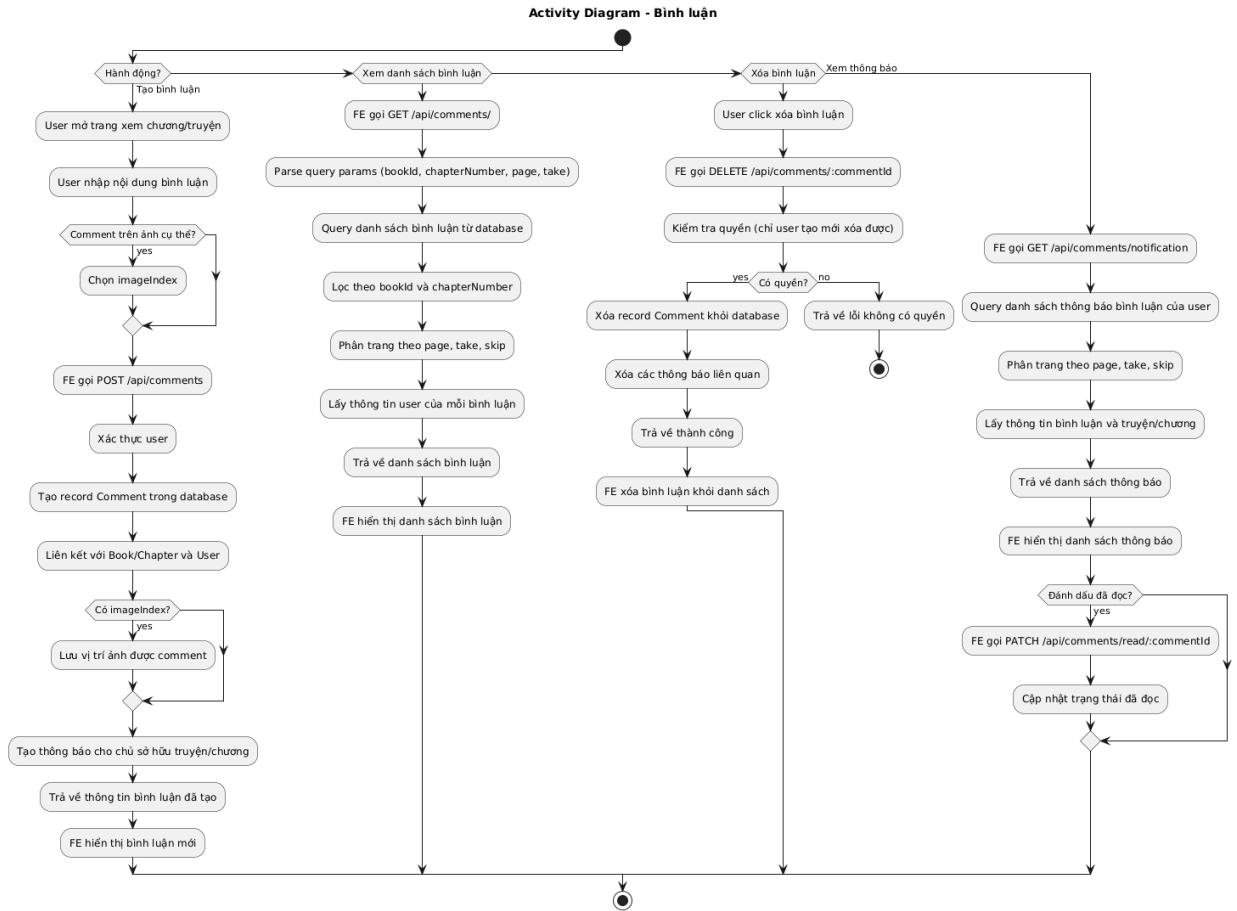
4	Tạo thông báo	Hệ thống	Gửi thông báo realtime (nếu online) hoặc lưu vào danh sách thông báo
5	Lấy danh sách bình luận	Người dùng	Scroll xuống phần bình luận hoặc load thêm
6	Lấy danh sách bình luận	Hệ thống	Trả về danh sách bình luận mới nhất → phân trang
7	Lấy thông báo	Người dùng	Nhấn chuông thông báo hoặc vào trang thông báo
8	Lấy thông báo	Hệ thống	Trả về các thông báo bình luận chưa đọc
9	Đánh dấu đã đọc	Người dùng	Nhấn vào thông báo hoặc xem bình luận
10	Đánh dấu đã đọc	Hệ thống	Cập nhật trạng thái đã đọc
11	Xóa bình luận	Người dùng	Nhấn nút xóa ở bình luận của mình
12	Xóa bình luận	Hệ thống	Kiểm tra quyền → xóa bình luận + xóa thông báo liên quan
13	Hoàn tất	Người dùng	Bình luận mượt, thông báo kịp thời, không bị spam

- Sơ đồ tuần tự chức năng bình luận



Hình 2.29 Sơ đồ tuần tự chức năng bình luận

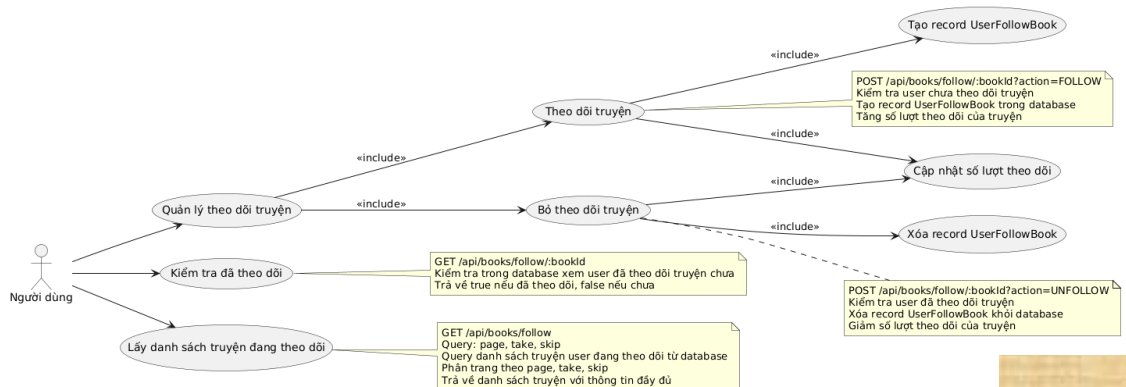
• Sơ đồ hoạt động chức năng bình luận



Hình 2.30 Sơ đồ hoạt động chức năng bình luận

2.2.7. Chức năng theo dõi truyện

• Sơ đồ usecase chức năng theo dõi truyện



(Liên kết xem chi tiết)

Hình 2.31 Sơ đồ usecase chức năng theo dõi truyện

Bảng 2.16 Mô tả Usecase chức năng theo dõi truyện

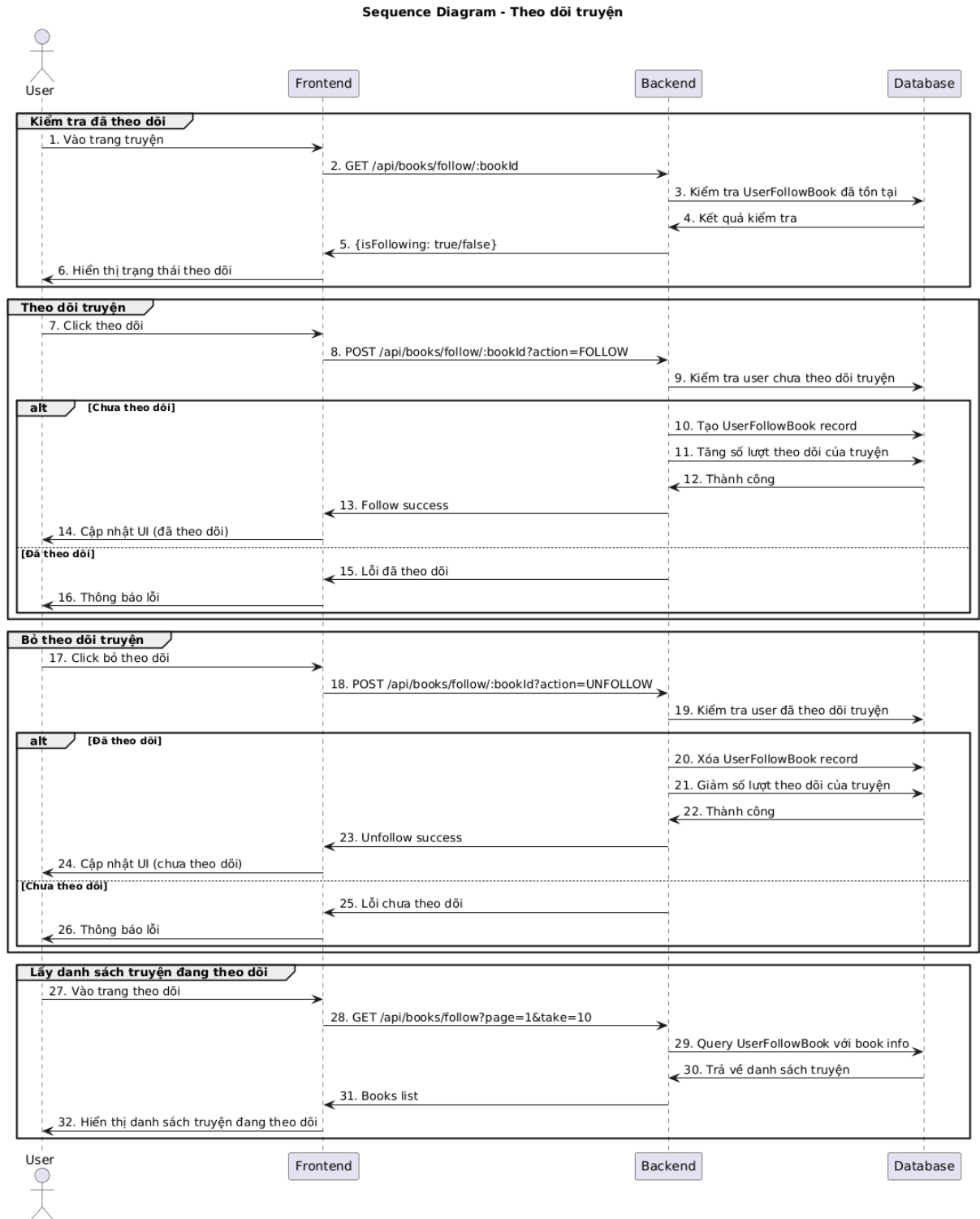
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý theo dõi truyện	Người dùng	Theo dõi / bỏ theo dõi truyện yêu thích	Đã đăng nhập	Trạng thái theo dõi luôn chính xác
Kiểm tra đã theo dõi	Hệ thống	Kiểm tra user đã theo dõi truyện đó chưa	Đã đăng nhập	Trả về true/false
Theo dõi truyện	Người dùng	Bắt đầu theo dõi một truyện	Chưa theo dõi	Nút chuyển thành “Đang theo dõi” + tăng lượt
Bỏ theo dõi truyện	Người dùng	Ngừng theo dõi truyện	Đang theo dõi	Nút chuyển thành “Theo dõi” + giảm lượt
Lấy danh sách truyện đang theo dõi	Người dùng	Xem tất cả truyện đang theo dõi	Đã đăng nhập	Hiển thị danh sách phân trang
Tạo record UserFollowBook	Hệ thống	Lưu quan hệ user - truyện trong DB	Gọi từ UC3	Record được tạo thành công
Xóa record UserFollowBook	Hệ thống	Xóa quan hệ user - truyện khỏi DB	Gọi từ UC4	Record bị xóa
Cập nhật số lượt theo dõi	Hệ thống	Tăng/giảm trường followers của truyện	Gọi từ UC3 và UC4	Số lượt theo dõi luôn đúng realtime

Bảng 2.17 Dòng chính Usecase chức năng theo dõi truyện

STT	Use Case	Thực hiện	Hành động chính
1	Quản lý theo dõi truyện	Người dùng	Vào trang chi tiết truyện

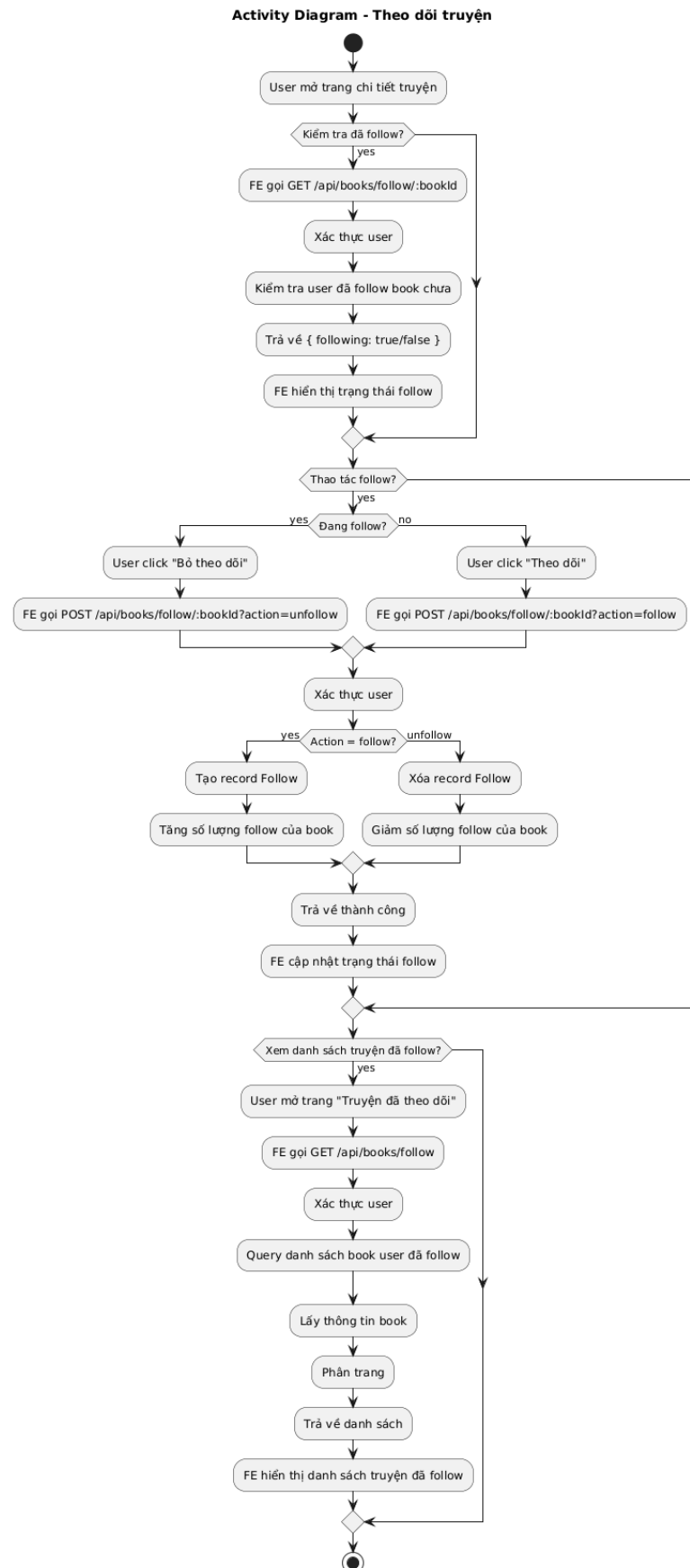
2	Kiểm tra đã theo dõi	Hệ thống	Kiểm tra DB → trả về trạng thái hiện tại
3	Theo dõi truyện	Người dùng	Nhấn nút “Theo dõi”
4	Theo dõi truyện	Hệ thống	Tạo record UserFollowBook → tăng followers của truyện → đổi nút thành “Bỏ theo dõi”
5	Bỏ theo dõi truyện	Người dùng	Nhấn nút “Bỏ theo dõi”
6	Bỏ theo dõi truyện	Hệ thống	Xóa record UserFollowBook → giảm followers → đổi nút thành “Theo dõi”
7	Lấy danh sách truyện đang theo dõi	Người dùng	Vào trang “Truyện theo dõi” hoặc tab cá nhân
8	Lấy danh sách truyện đang theo dõi	Hệ thống	Query DB theo userId → trả danh sách phân trang + thông tin truyện + chapter mới nhất
9	Hoàn tất	Người dùng	Để đảm bảo quản lý truyện yêu thích → số lượt theo dõi luôn chính xác

- Sơ đồ tuần tự chức năng theo dõi truyện



Hình 2.32 Sơ đồ tuần tự chức năng theo dõi truyện

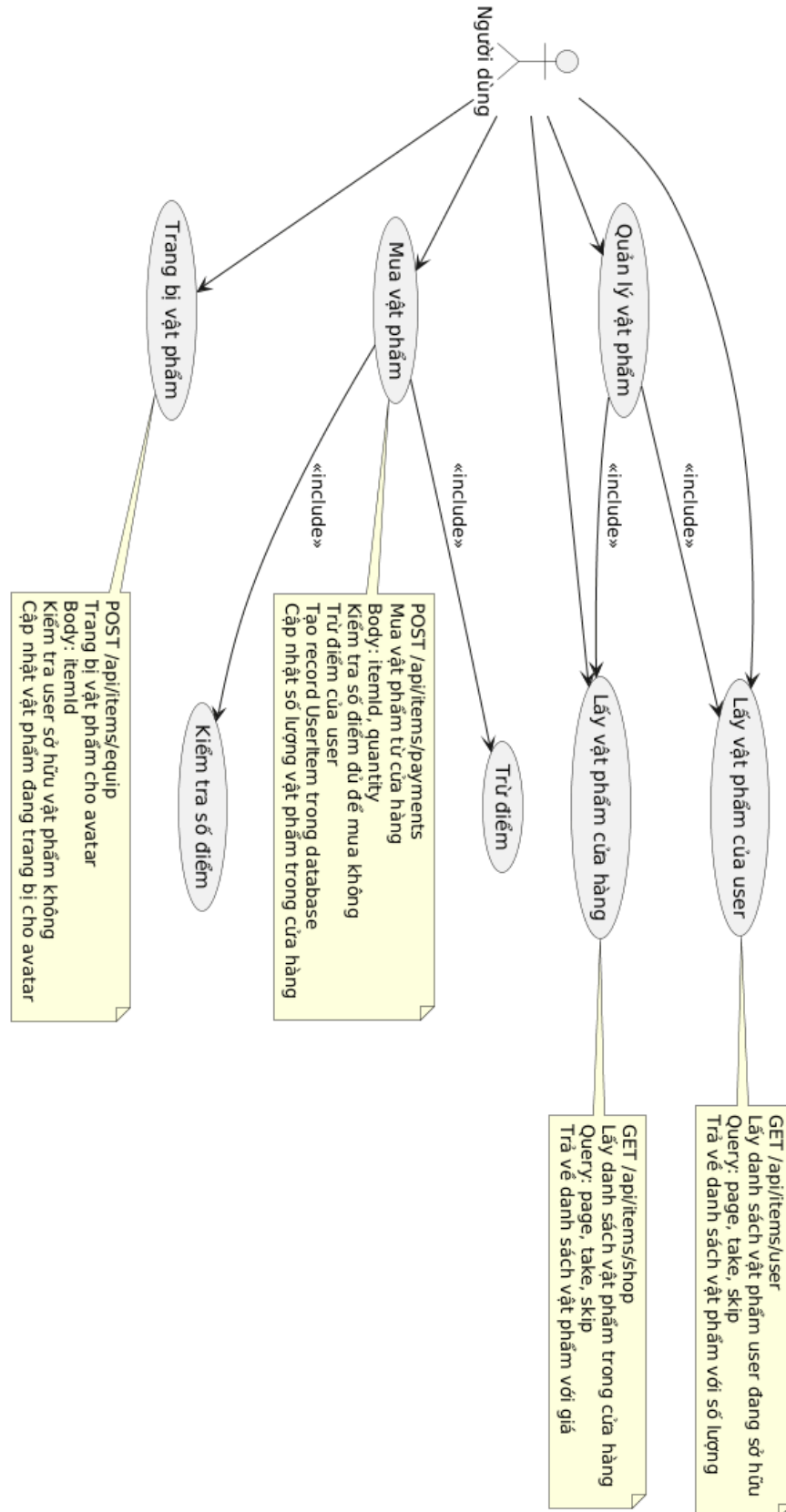
- Sơ đồ hoạt động chức năng theo dõi truyện



Hình 2.33 Sơ đồ hoạt động chức năng theo dõi truyện

2.2.8. Chức năng quản lý cửa hàng vật phẩm

- Sơ đồ usecase chức năng quản lý cửa hàng vật phẩm



Hình 2.34 Sơ đồ usecase chức năng quản lý cửa hàng vật phẩm

Bảng 2.18 Mô tả Usecase chức năng quản lí cửa hàng vật phẩm

Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý vật phẩm	Người dùng	Xem, mua và trang bị vật phẩm	Đã đăng nhập	Toàn bộ thao tác vật phẩm mượt mà
Lấy vật phẩm của user	Người dùng	Xem kho vật phẩm cá nhân	Đã đăng nhập	Hiển thị danh sách vật phẩm đang sở hữu
Lấy vật phẩm của hàng	Người dùng	Xem các vật phẩm đang bán trong shop	Không cần đăng nhập	Hiển thị danh sách vật phẩm + giá
Mua vật phẩm	Người dùng	Mua vật phẩm bằng điểm	Đã đăng nhập, đủ điểm	Sở hữu vật phẩm mới, điểm bị trừ
Trang bị vật phẩm	Người dùng	Gắn vật phẩm lên avatar	Đã đăng nhập, sở hữu vật phẩm	Avatar hiển thị vật phẩm mới
Kiểm tra số điểm	Hệ thống	Kiểm tra user có đủ điểm để mua không		Trả về true/false
Trừ điểm	Hệ thống	Trừ điểm của user sau khi mua thành công		Số điểm user giảm đúng số lượng

Bảng 2.19 Dòng chính Usecase chức năng quản lí cửa hàng vật phẩm

STT	Use Case	Thực hiện	Hành động chính
1	Quản lý vật phẩm	Người dùng	Vào trang “Kho đồ” hoặc “Cửa hàng”
2	Lấy vật phẩm của user	Hệ thống	Trả về danh sách vật phẩm user đang sở hữu + số lượng

3	Lấy vật phẩm cửa hàng	Hệ thống	Trả về toàn bộ vật phẩm đang bán + giá điểm
4	Mua vật phẩm	Người dùng	Chọn vật phẩm → nhấn “Mua”
5	Kiểm tra số điểm	Hệ thống	Kiểm tra điểm hiện tại → nếu không đủ → báo lỗi
6	Trừ điểm	Hệ thống	Nếu đủ → trừ điểm → thêm vật phẩm vào kho user
7	Mua vật phẩm	Người dùng	Nhận thông báo “Mua thành công” → kho đồ cập nhật realtime
8	Trang bị vật phẩm	Người dùng	Vào kho đồ → chọn vật phẩm → nhấn “Trang bị”
9	Trang bị vật phẩm	Hệ thống	Kiểm tra sở hữu → cập nhật trường equipped cho user/avatar
10	Hoàn tất	Người dùng	Avatar hiển thị vật phẩm mới ngay lập tức → điểm và kho luôn chính xác

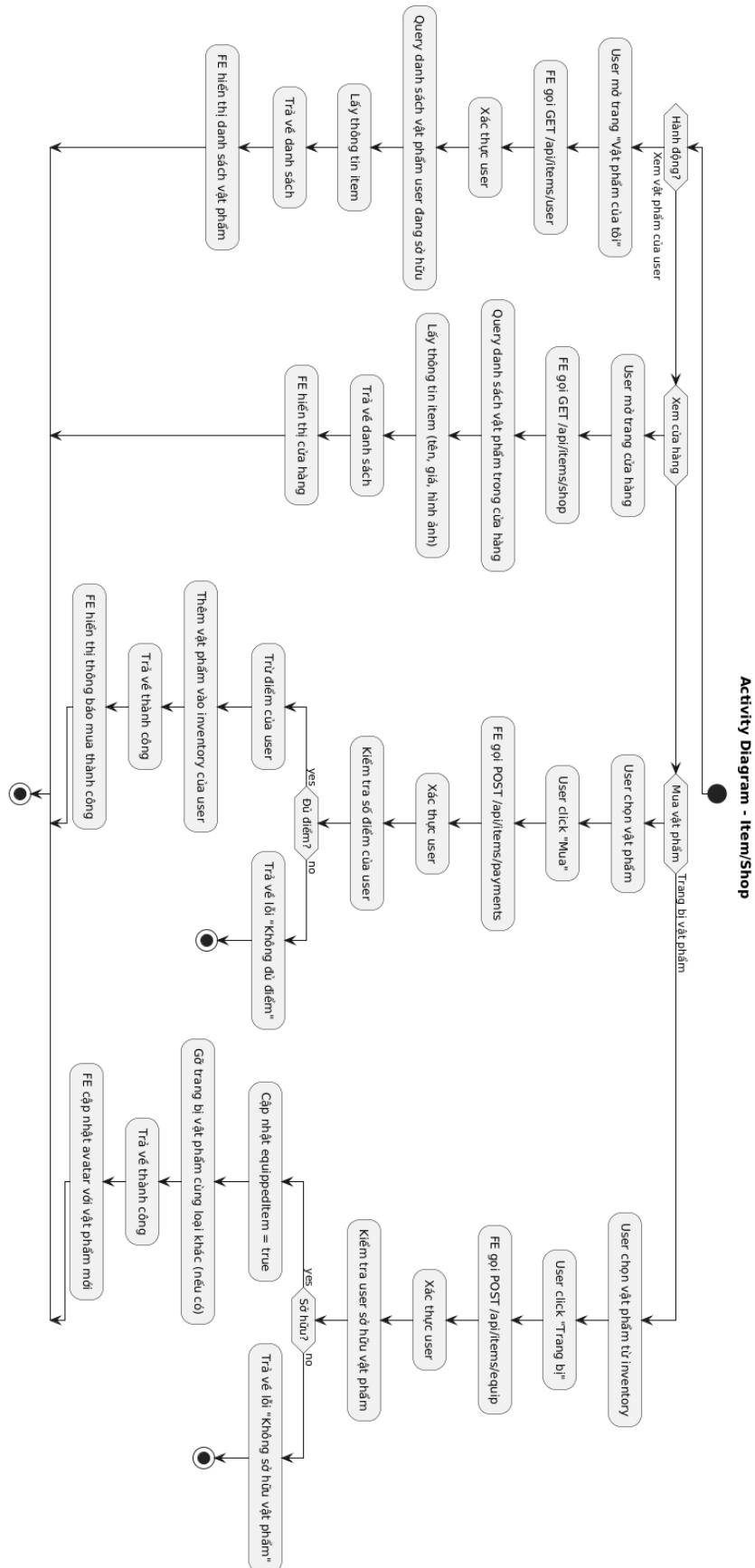
- Sơ đồ tuần tự chức năng quản lý cửa hàng vật phẩm



([Liên kết](#) xem chi tiết)

Hình 2.35 Sơ đồ tuần tự chức năng cửa hàng vật phẩm

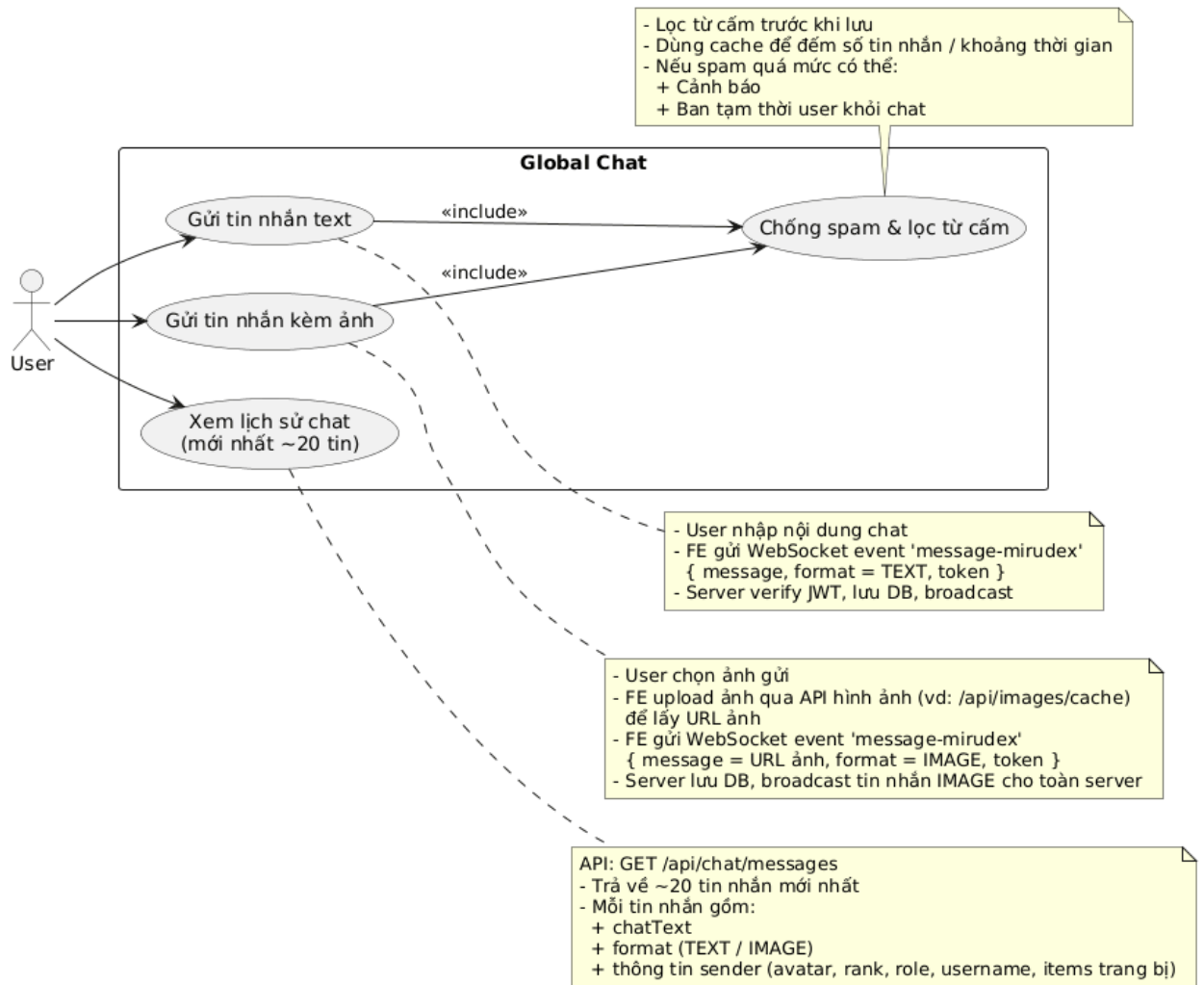
- Sơ đồ hoạt động chức năng cửa hàng vật phẩm



Hình 2.36 Sơ đồ hoạt động chức năng cửa hàng vật phẩm

2.3.9. Chức năng nhắn tin toàn hệ thống

- Sơ đồ usecase chức năng nhắn tin toàn hệ thống



Hình 2.37 Sơ đồ usecase chức năng nhắn tin toàn hệ thống

Bảng 2.20 Mô tả usecase chức năng nhắn tin toàn hệ thống

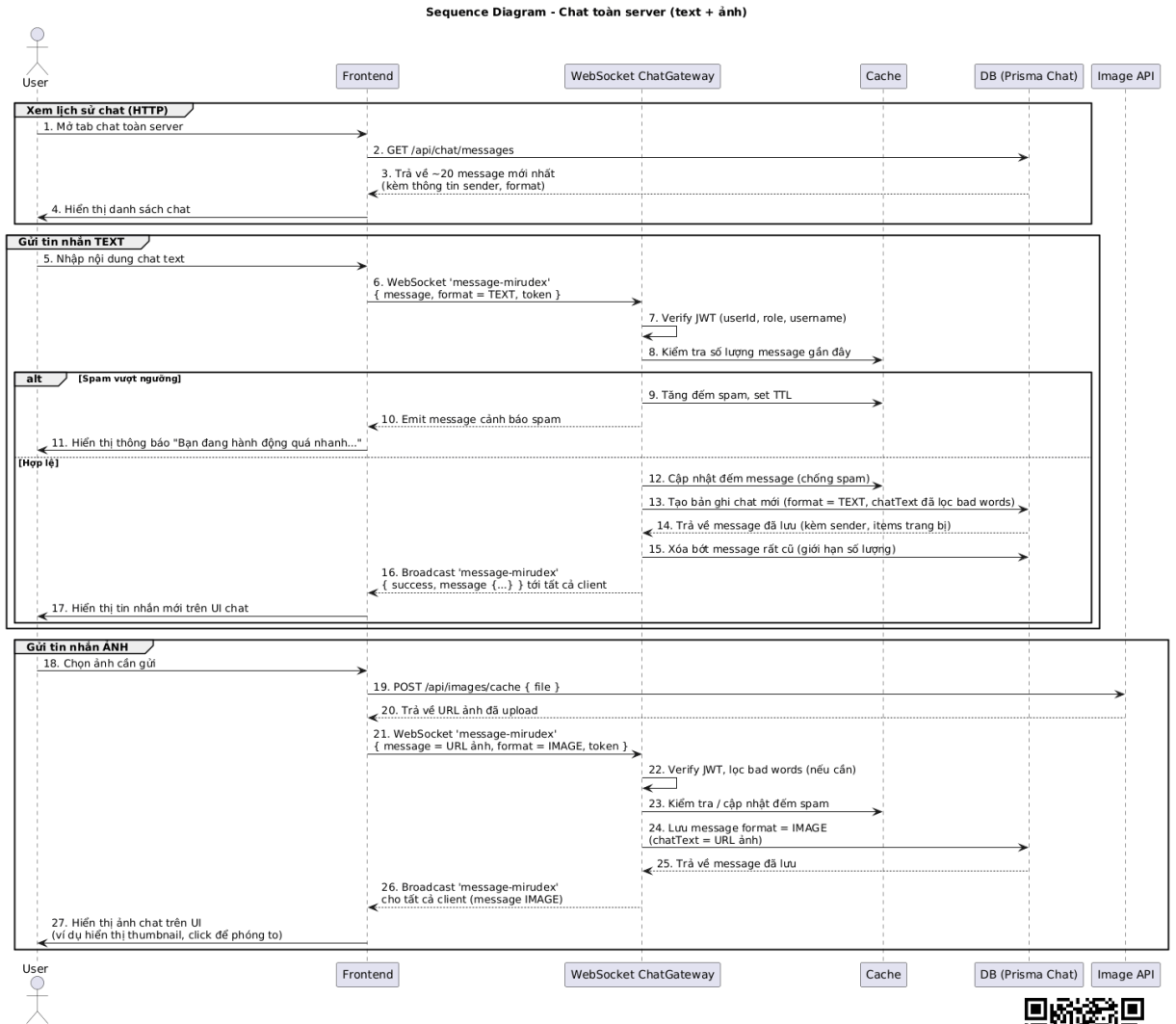
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Xem lịch sử chat (mới nhất ~20 tin)	User	Xem các tin nhắn gần nhất khi vào chat	Đã vào phòng chat toàn server	Hiển thị ~20 tin nhắn mới nhất + avatar, rank
Gửi tin nhắn text	User	Gửi tin nhắn dạng chữ	Đã đăng nhập	Tin nhắn hiện ngay cho mọi người + đã lọc từ cấm

Gửi tin nhắn kèm ảnh	User	Gửi tin nhắn có hình ảnh	Đã đăng nhập	Ảnh được upload → hiển thị dưới dạng ảnh chat
Chống spam & lọc từ cấm	Hệ thống	Ngăn spam, từ cấm, hành vi xấu trong chat	Mọi tin nhắn gửi lên	Tin nhắn sạch, user spam bị cảnh báo/ban chat

Bảng 2.21 Dòng chính Usecase chức năng nhắn tin toàn hệ thống

STT	Use Case	Thực hiện	Hành động chính
1	Vào Global Chat	User	Mở tab chat toàn server
2	Xem lịch sử chat	Hệ thống	Tự động gọi API → trả về ~20 tin nhắn mới nhất kèm avatar, rank, role, vật phẩm trang bị
3	Gửi tin nhắn text	User	Gõ nội dung → nhấn Enter hoặc nút Gửi
4	Chống spam & lọc từ cấm	Hệ thống	Kiểm tra từ cấm → kiểm tra tần suất gửi (cache) → nếu OK thì lưu DB và broadcast qua WS
5	Gửi tin nhắn kèm ảnh	User	Nhấn icon ảnh → chọn ảnh → tự động upload lấy URL
6	Gửi tin nhắn kèm ảnh	Hệ thống	Sau khi có URL ảnh → gửi qua WebSocket với format = IMAGE
7	Chống spam & lọc từ cấm	Hệ thống	Kiểm tra URL ảnh hợp lệ + chống spam như tin text → lưu DB và broadcast
8	Nhận tin nhắn realtime	Tất cả user	WebSocket push tin nhắn mới → frontend append ngay vào khung chat
9	Spam quá mức	Hệ thống	Nếu vượt ngưỡng → tự động cảnh báo hoặc mute/ban chat tạm thời cho user đó
10	Hoàn tất	User	Chat toàn server hoạt động mượt, sạch sẽ, realtime, có ảnh và chống spam hiệu quả

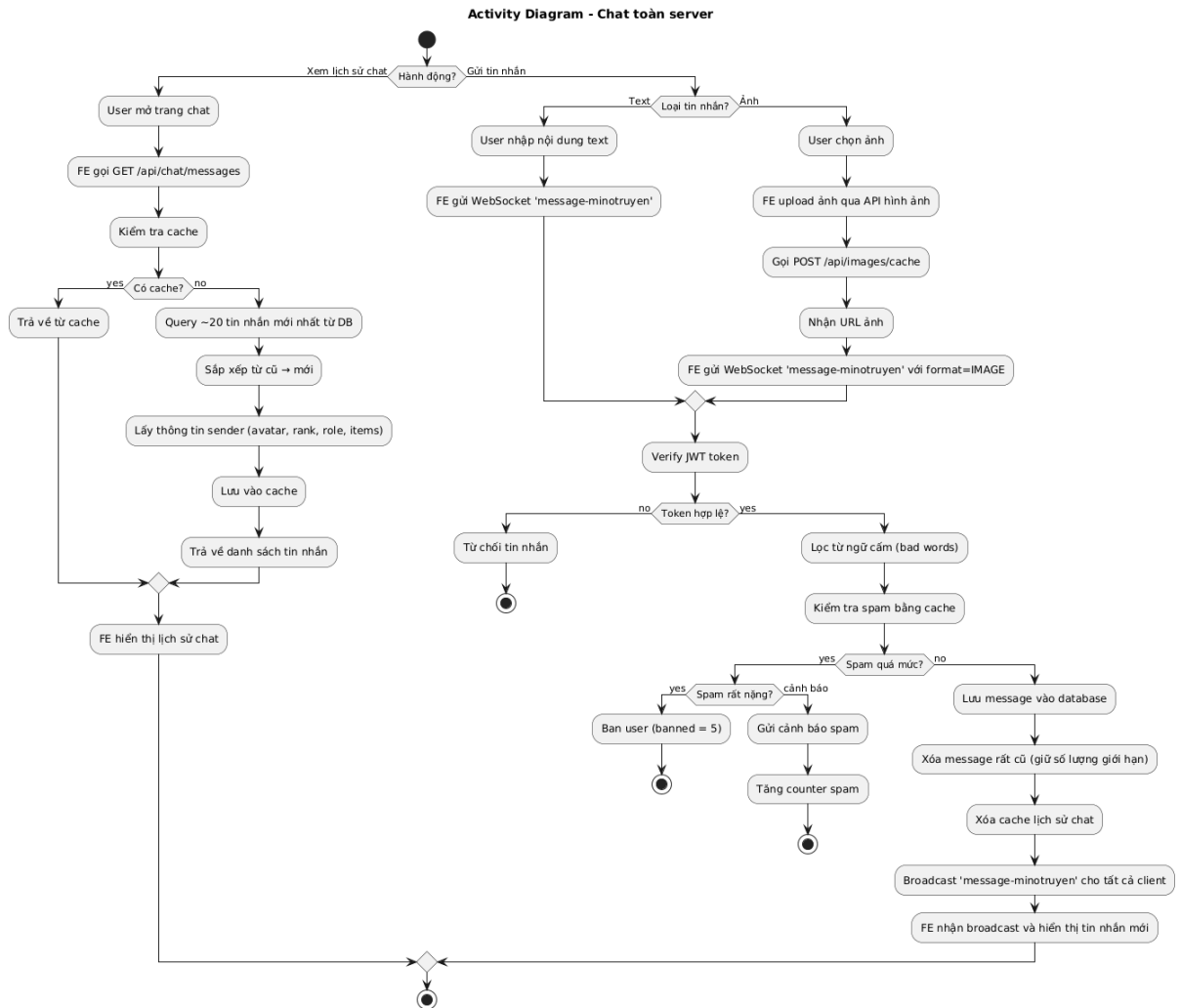
- Sơ đồ tuần tự chức năng nhắn tin toàn hệ thống



([Liên kết](#) xem chi tiết)

Hình 2.38 Sơ đồ tuần tự chức năng nhắn tin toàn hệ thống

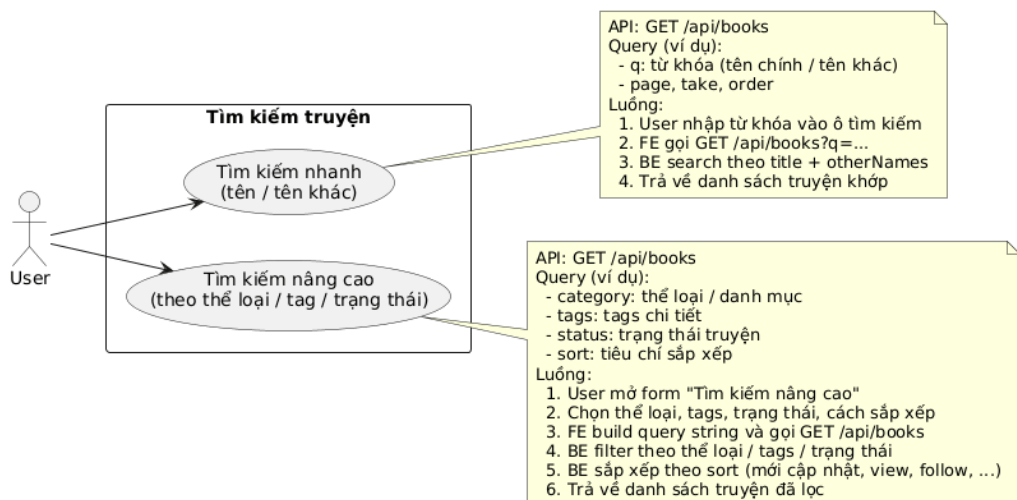
• Sơ đồ hoạt động chức năng nhắn tin toàn hệ thống



Hình 2.39 Sơ đồ hoạt động chức năng nhắn tin toàn hệ thống

2.2.10. Chức năng tìm kiếm truyện

• Sơ đồ usecase chức năng tìm kiếm truyện



Hình 2.40 Sơ đồ usecase chức năng tìm kiếm truyện

Bảng 2.22 Mô tả Usecase chức năng tìm kiếm truyện

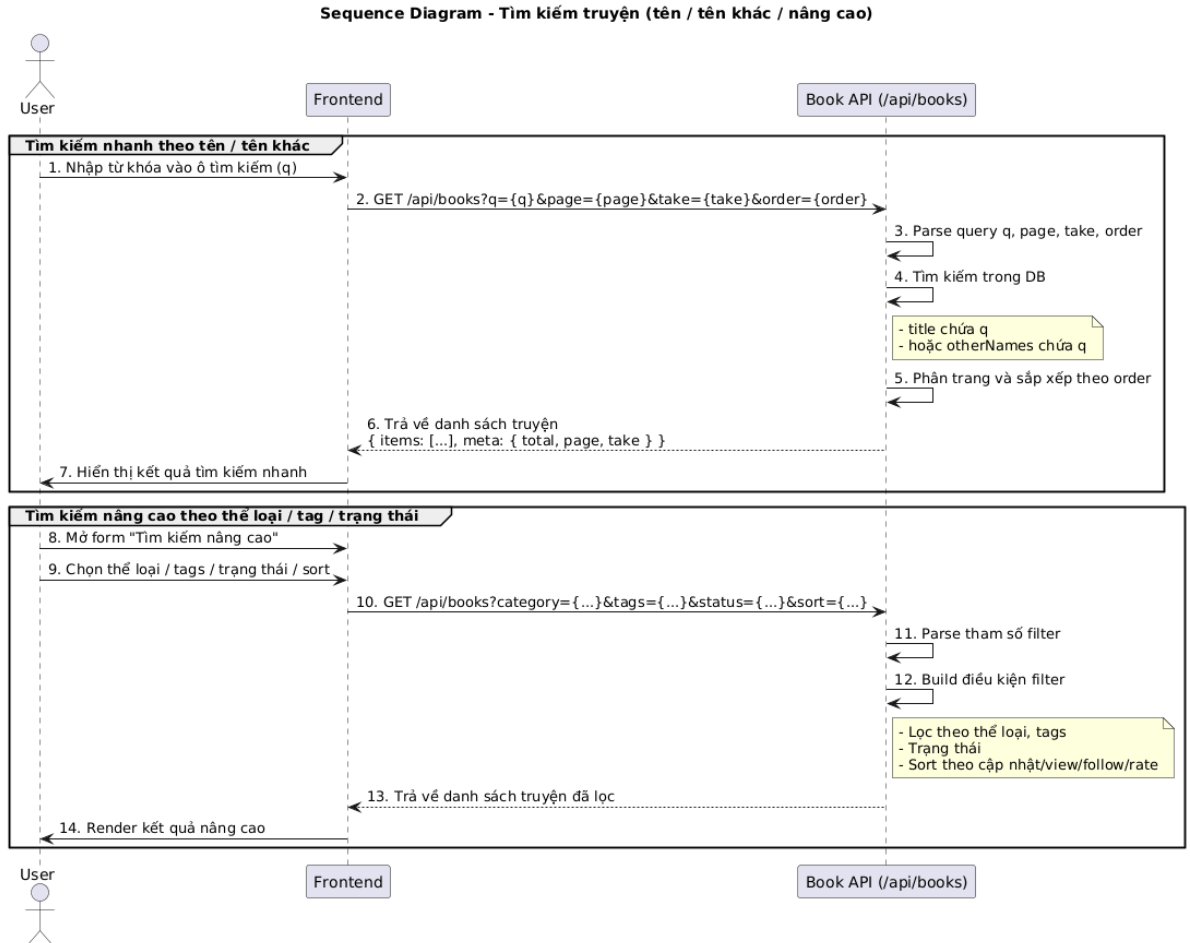
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Tìm kiếm nhanh (tên / tên khác)	User	Tìm truyện bằng từ khóa tên chính hoặc tên khác	Không cần đăng nhập	Hiển thị nhanh danh sách truyện khớp
Tìm kiếm nâng cao	User	Lọc truyện theo thể loại, tag, trạng thái, sắp xếp	Không cần đăng nhập	Danh sách truyện chính xác theo bộ lọc

Bảng 2.23 Dòng chính Usecase chức năng tìm kiếm truyện

STT	Use Case	Thực hiện	Hành động chính
1	Tìm kiếm nhanh	User	Gõ từ khóa vào ô tìm kiếm toàn site hoặc trang chủ
2	Tìm kiếm nhanh	Hệ thống	Tự động gọi API tìm kiếm → tìm trong title + otherNames
3	Tìm kiếm nhanh	Hệ thống	Trả về kết quả ngay khi đang gõ (debounce) hoặc khi nhấn Enter
4	Tìm kiếm nâng cao	User	Nhấn nút “Tìm kiếm nâng cao” hoặc biểu tượng filter
5	Tìm kiếm nâng cao	User	Chọn thể loại → tag → trạng thái (đang ra/hoàn thành) → cách sắp xếp
6	Tìm kiếm nâng cao	Hệ thống	Build query string → gọi API với các param đã chọn
7	Tìm kiếm nâng cao	Hệ thống	Lọc chính xác theo thể loại + tag + trạng thái → sắp xếp theo tiêu chí → phân trang
8	Cả hai loại tìm kiếm	User	Nhận danh sách truyện dưới dạng grid/list → có thumbnail, tên, chapter mới nhất

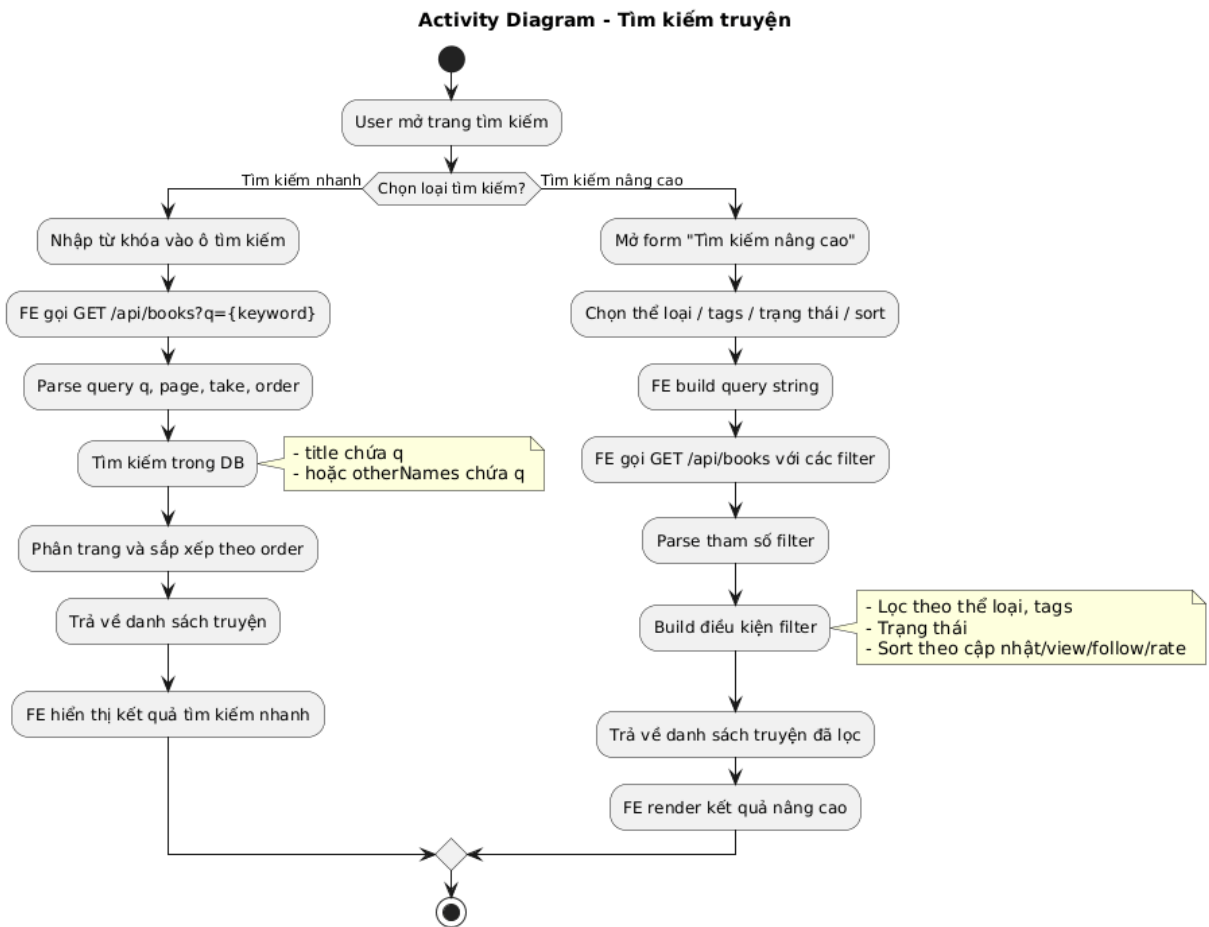
9	Hoàn tất	User	Tìm được truyện mong muốn cực nhanh và chính xác dù chỉ nhớ tên hoặc muốn lọc chi tiết
---	----------	------	--

- Sơ đồ tuần tự chức năng tìm kiếm truyện



Hình 2.41 Sơ đồ tuần tự chức năng tìm kiếm truyện

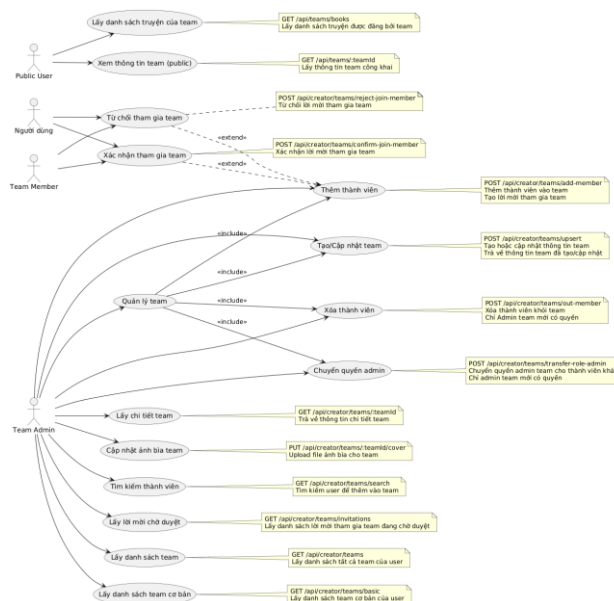
- Sơ đồ hoạt động chức năng tìm kiếm truyện



Hình 2.42 Sơ đồ hoạt động chức năng tìm kiếm truyện

2.2.11. Chức năng quản lý nhóm

- Sơ đồ usecase chức năng quản lý nhóm



Hình 2.43 Sơ đồ usecase chức năng quản lý nhóm

Bảng 2.24 Mô tả Usecase chức năng quản lý nhóm

Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý team	Team Admin	Tạo, chỉnh sửa, quản lý thành viên và quyền team	Là Admin team	Team hoạt động trơn tru, quyền rõ ràng
Tạo/Cập nhật team	Team Admin	Tạo mới hoặc chỉnh sửa thông tin team	Đã đăng nhập	Team được tạo/cập nhật thành công
Lấy danh sách team	Team Admin	Xem tất cả team mình đang sở hữu/quản lý	Đã đăng nhập	Hiển thị đầy đủ danh sách team
Lấy danh sách team cơ bản	Team Admin	Lấy thông tin rút gọn các team	Đã đăng nhập	Dùng cho dropdown, chọn team nhanh
Lấy chi tiết team	Team Admin	Xem thông tin chi tiết một team	Là thành viên team	Hiển thị đầy đủ info + thành viên
Cập nhật ảnh bìa team	Team Admin	Đổi cover cho team	Là Admin team	Ảnh bìa được cập nhật
Tìm kiếm thành viên	Team Admin	Tìm user để mời vào team	Là Admin team	Gợi ý user phù hợp
Thêm thành viên	Team Admin	Mời user vào team	Là Admin team	Lời mời được gửi
Xóa thành viên	Team Admin	Kick thành viên khỏi team	Là Admin team	Thành viên bị loại bỏ
Chuyển quyền admin	Team Admin	Chuyển vai trò Admin cho thành viên khác	Là Admin hiện tại	Quyền Admin được chuyển giao
Lấy lời mời chờ duyệt	Team Admin	Xem các lời mời đang chờ thành viên xác nhận	Là Admin team	Hiển thị danh sách lời mời

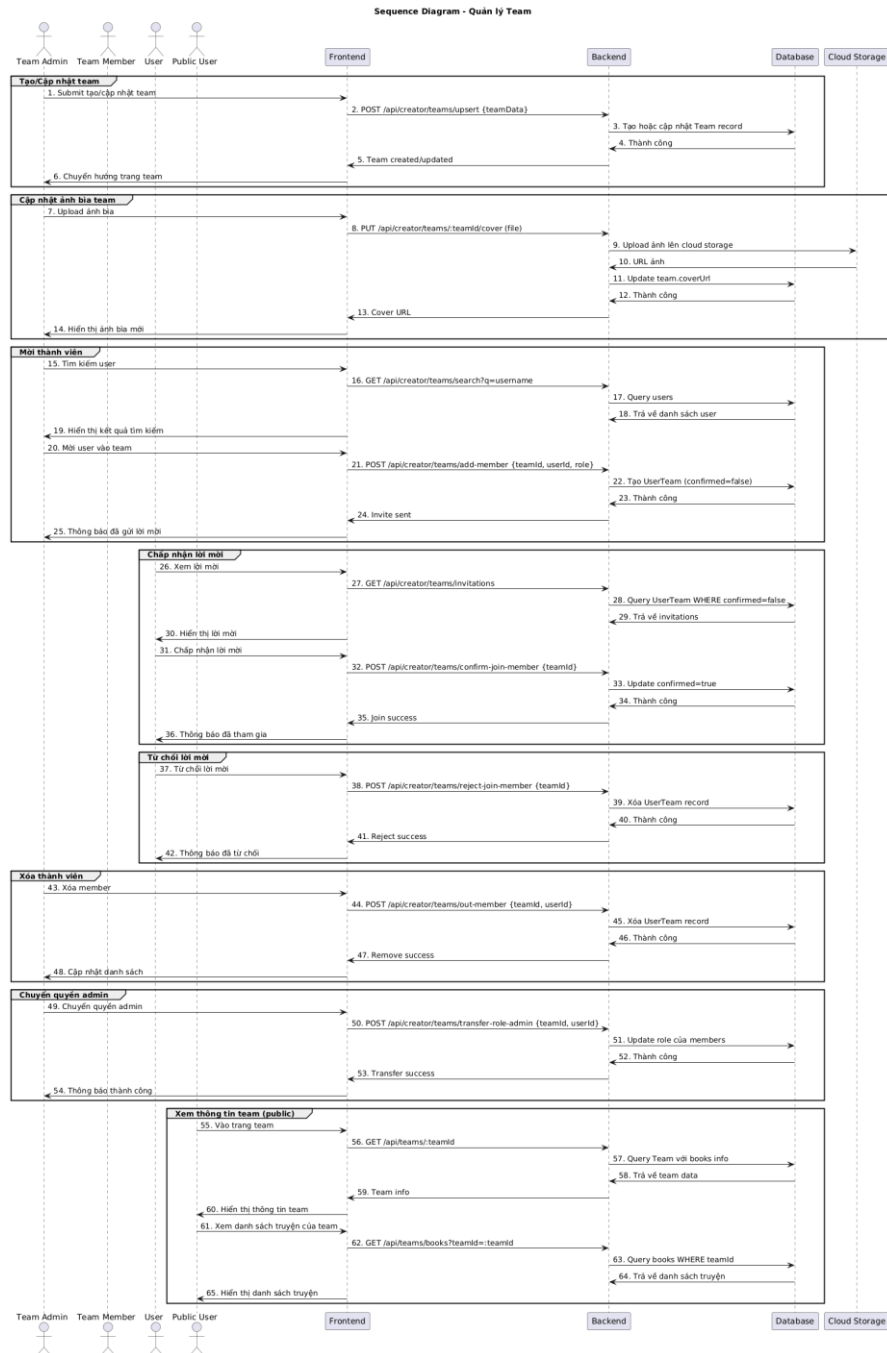
Xác nhận tham gia team	User / Member	Đồng ý tham gia team sau khi nhận lời mời	Nhận được lời mời	Trở thành thành viên team
Từ chối tham gia team	User / Member	Từ chối lời mời tham gia team	Nhận được lời mời	Lời mời bị hủy
Lấy danh sách truyện của team	Public User	Xem các truyện được đăng bởi team	Không cần đăng nhập	Hiển thị truyện công khai của team
Xem thông tin team (public)	Public User	Xem trang team công khai	Không cần đăng nhập	Hiển thị info team + thành viên + truyện

Bảng 2.25 Dòng chính Usecase chức năng quản lý nhóm

STT	Use Case	Thực hiện	Hành động chính
1	Tạo/Cập nhật team	Team Admin	Vào trang Creator → Team → Tạo mới hoặc chỉnh sửa team
2	Cập nhật ảnh bìa team	Team Admin	Upload ảnh bìa mới → lưu lại
3	Thêm thành viên	Team Admin	Tìm kiếm user → gửi lời mời tham gia team
4	Xác nhận tham gia team	User được mời	Vào phần Thông báo / Lời mời → nhấn “Đồng ý”
5	Từ chối tham gia team	User được mời	Nhấn “Từ chối” → lời mời bị hủy
6	Xóa thành viên	Team Admin	Vào danh sách thành viên → chọn user → nhấn “Xóa”
7	Chuyển quyền admin	Team Admin	Chọn thành viên đáng tin → nhấn “Chuyển quyền Admin”
8	Lấy danh sách truyện của team	Public User	Vào trang team công khai → xem tab “Truyện”
9	Xem thông tin team (public)	Public User	Truy cập /team/:teamId hoặc click từ truyện → xem info + thành viên + ảnh bìa

10	Hoàn tất	Tất cả	Team hoạt động chuyên nghiệp như nhóm dịch/nhóm sáng tác → quyền rõ ràng, public đẹp mắt
----	----------	--------	--

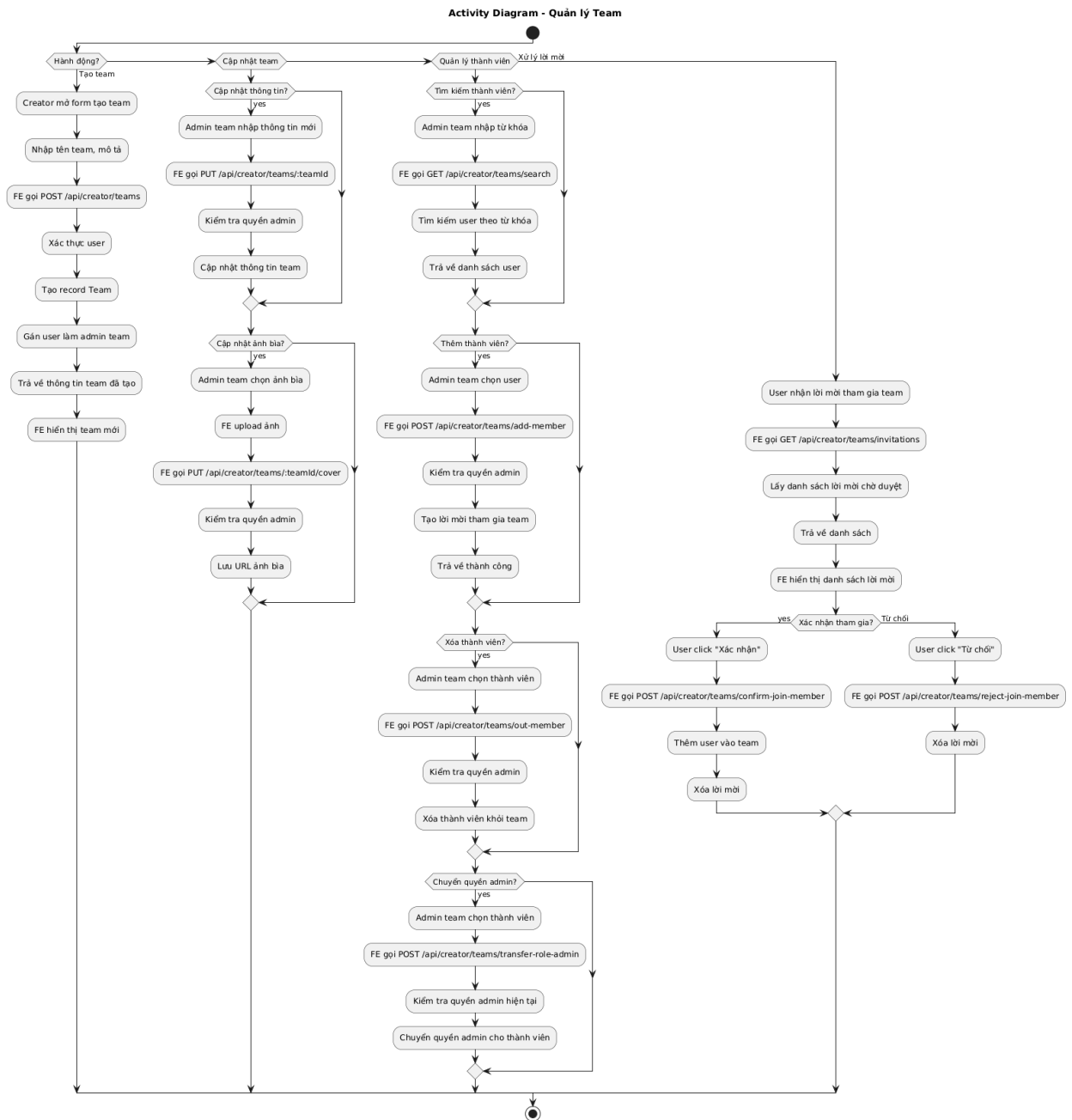
• Sơ đồ tuần tự chức năng quản lý nhóm



([Liên kết](#) xem chi tiết)

Hình 2.44 Sơ đồ tuần tự chức năng quản lý nhóm

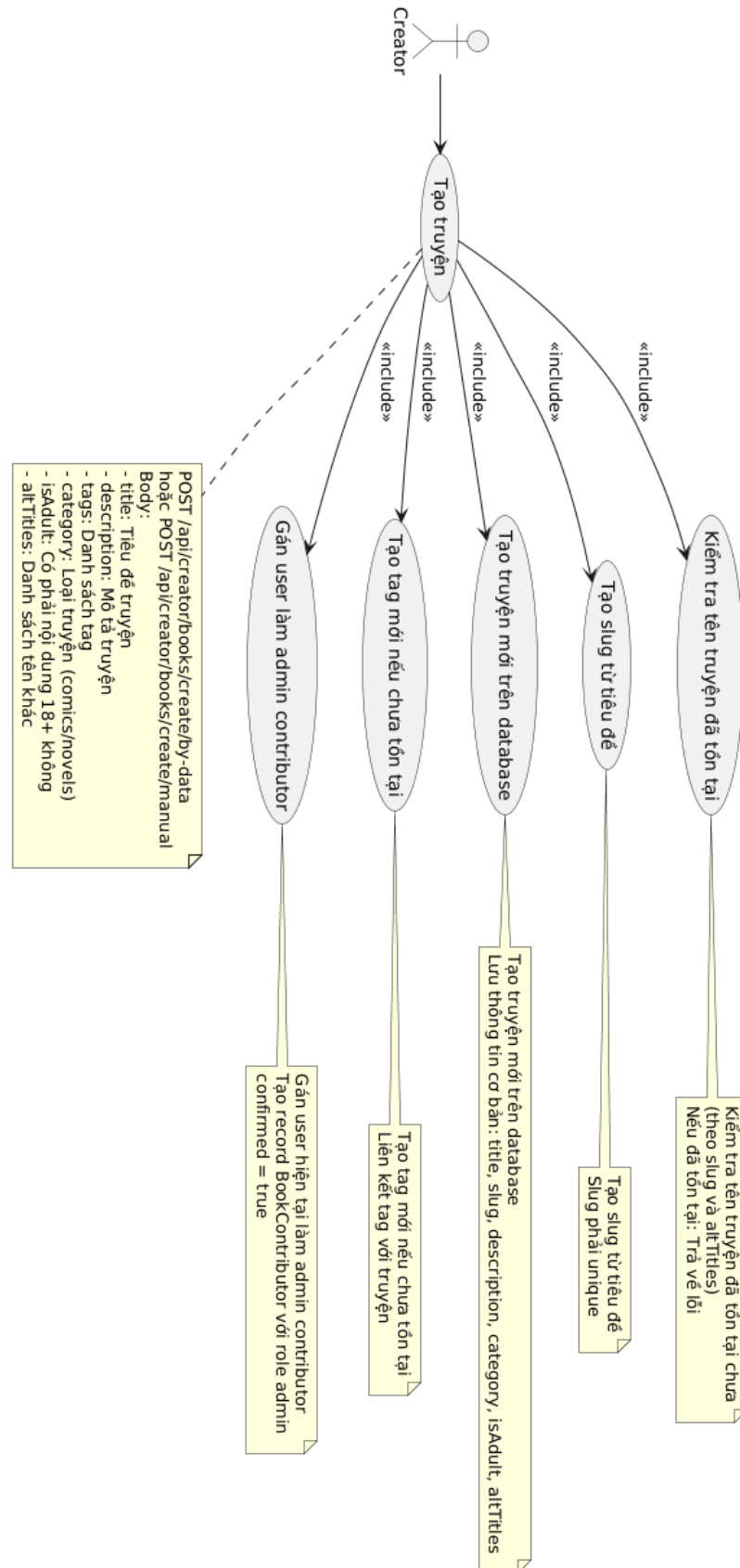
• Sơ đồ hoạt động chức năng quản lý nhóm



Hình 2.45 Sơ đồ hoạt động chức năng quản lý nhóm

2.2.12. Chức năng tạo truyện

- Sơ đồ usecase chức năng tạo truyện



Hình 2.46 Sơ đồ usecase chức năng tạo truyện

Bảng 2.26 Mô tả Usecase chức năng tạo truyện

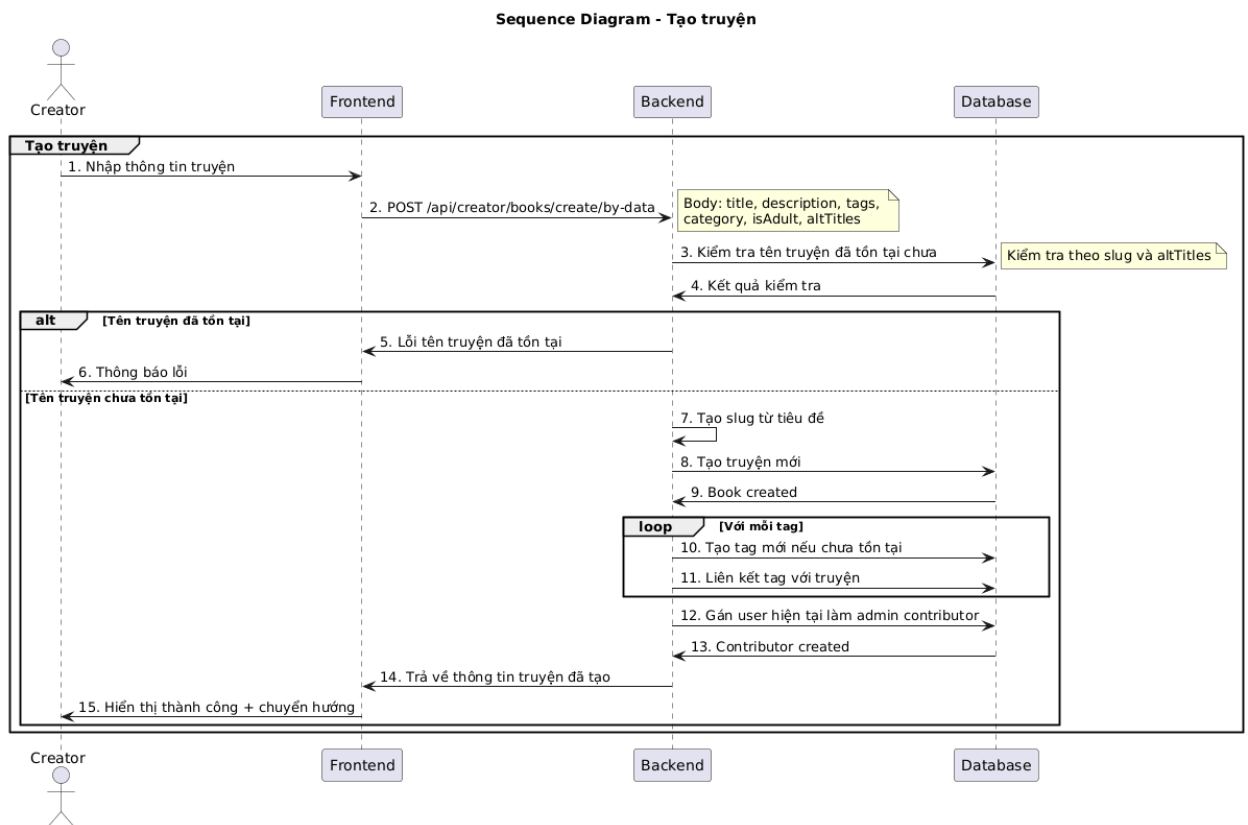
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Tạo truyện	Creator	Tạo một truyện mới (comic/novel)	Đã đăng nhập, có quyền Creator	Truyện được tạo thành công, sẵn sàng đăng chapter
Kiểm tra tên truyện đã tồn tại	Hệ thống	Ngăn tạo truyện trùng tên hoặc tên khác	Nhập tiêu đề hoặc altTitles	Báo lỗi nếu trùng, cho phép nếu chưa tồn tại
Tạo slug từ tiêu đề	Hệ thống	Tạo đường dẫn SEO thân thiện và duy nhất	Có tiêu đề	Slug đẹp, unique, không trùng
Tạo truyện mới trên database	Hệ thống	Lưu toàn bộ thông tin truyện vào DB	Đã qua kiểm tra trùng	Record Book được tạo thành công
Tạo tag mới nếu chưa tồn tại	Hệ thống	Đảm bảo tag hợp lệ và liên kết với truyện	Có danh sách tag	Tag được tạo hoặc tái sử dụng → gắn vào truyện
Gán user làm admin contributor	Hệ thống	Tự động gán người tạo làm chủ truyện	Truyện vừa được tạo	Creator có full quyền quản lý truyện

Bảng 2.27 Dòng chính Usecase chức năng tạo truyện

STT	Use Case	Thực hiện	Hành động chính
1	Tạo truyện	Creator	Vào trang Creator → nhấn “Tạo truyện mới”
2	Tạo truyện	Creator	Nhập tiêu đề → mô tả → chọn thể loại → tag → tên khác
3	Kiểm tra tên truyện đã tồn tại	Hệ thống	Kiểm tra title + altTitles → nếu trùng → báo lỗi “Tên truyện đã tồn tại”
4	Tạo slug từ tiêu đề	Hệ thống	Tự động sinh slug từ title → đảm bảo unique (thêm số nếu cần)

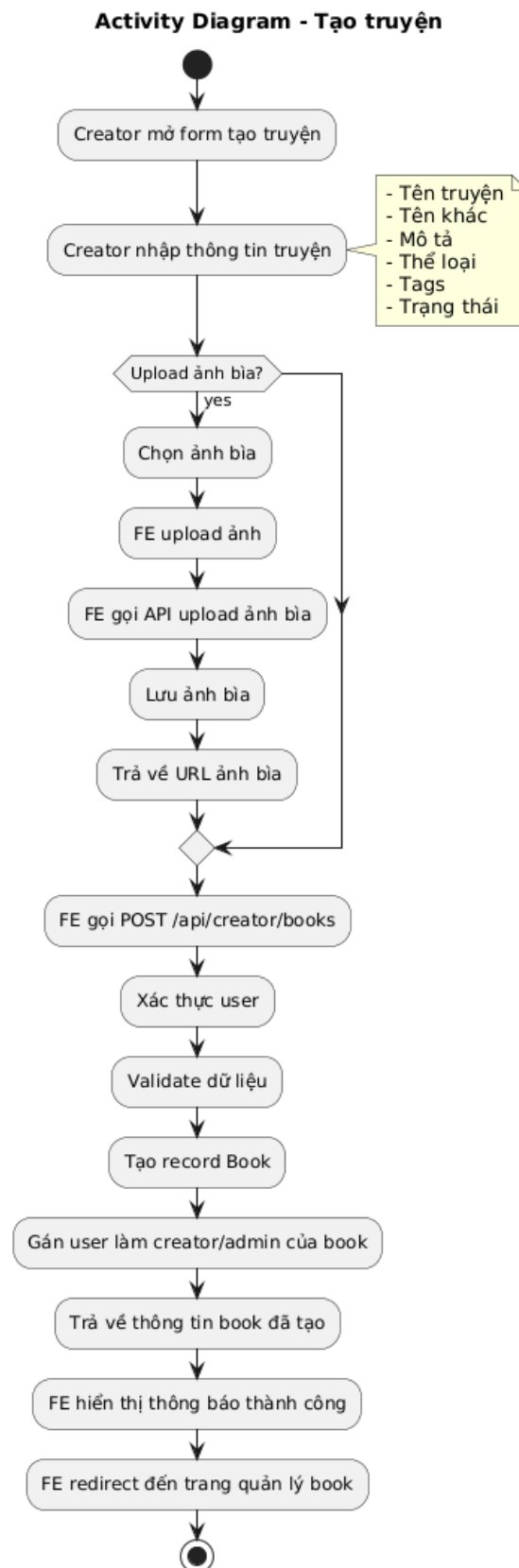
5	Tạo truyện mới trên database	Hệ thống	Lưu toàn bộ thông tin truyện: title, slug, description, category, isAdult, altTitles...
6	Tạo tag mới nếu chưa tồn tại	Hệ thống	Với mỗi tag → kiểm tra tồn tại → nếu chưa có thì tạo mới → liên kết với truyện
7	Gán user làm admin contributor	Hệ thống	Tạo record BookContributor: userId = creator, role = ADMIN, confirmed = true
8	Hoàn tất	Creator	Nhận thông báo “Tạo truyện thành công” → được chuyển đến trang quản lý truyện vừa tạo
9	Hoàn tất	Creator	Có thể bắt đầu upload chapter đầu tiên ngay lập tức

- Sơ đồ tuần tự chức năng tạo truyện



Hình 2.47 Sơ đồ tuần tự chức năng tạo truyện

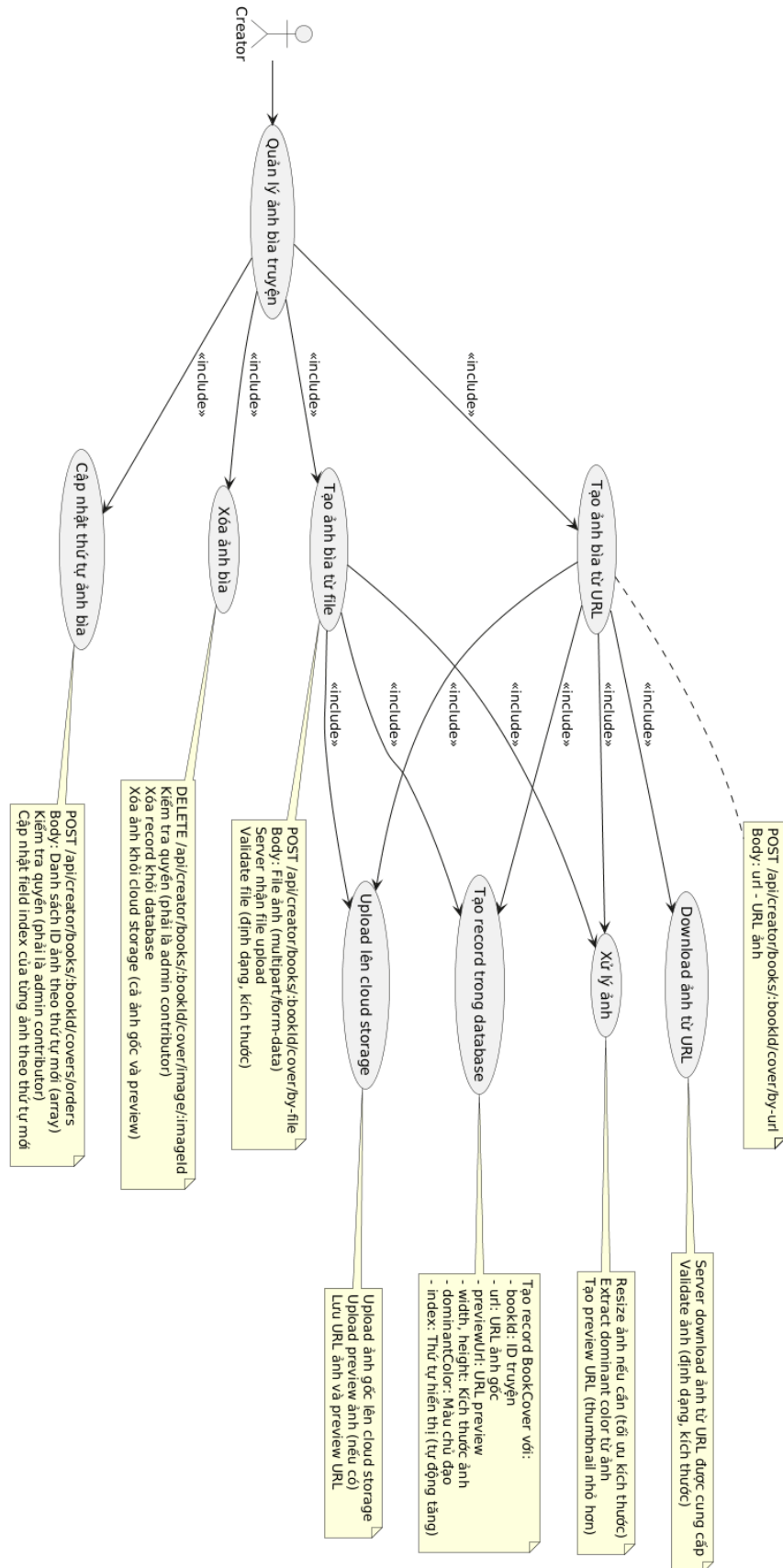
- Sơ đồ hoạt động chức năng tạo truyện



Hình 2.48 Sơ đồ hoạt động chức năng tạo truyện

2.2.13. Chức năng quản lý ảnh truyện

- Sơ đồ usecase chức năng quản lý ảnh truyện



Hình 2.49 Sơ đồ usecase chức năng quản lý ảnh truyện

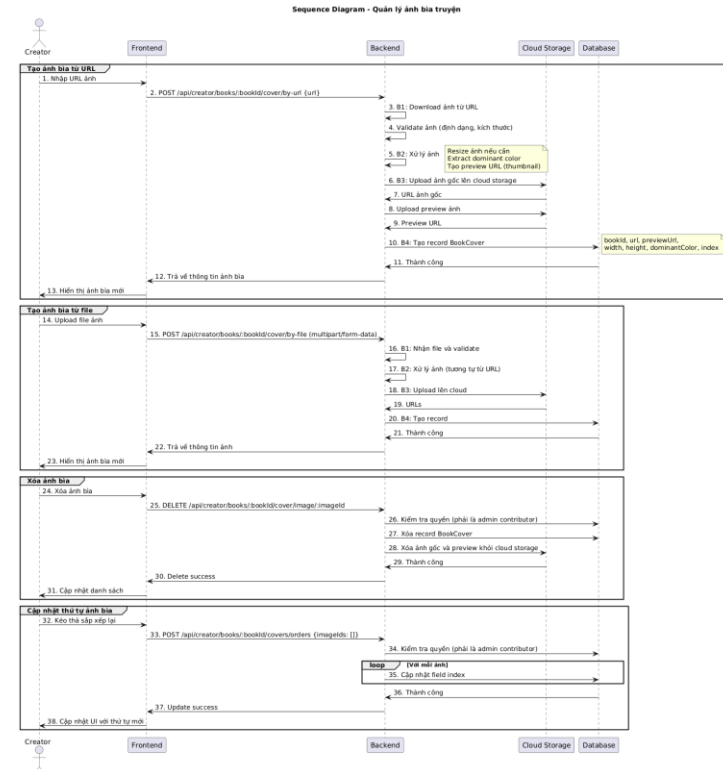
Bảng 2.28 Mô tả Usecase chức năng quản lý ảnh truyện

Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý ảnh bìa truyện	Creator	Thêm, xóa, sắp xếp lại các ảnh bìa của truyện	Là admin contributor của truyện	Ảnh bìa đẹp, nhiều ảnh, hiển thị đúng thứ tự
Tạo ảnh bìa từ URL	Creator	Thêm ảnh bìa bằng cách dán link	Có URL ảnh hợp lệ	Ảnh được tải về, xử lý và lưu thành công
Tạo ảnh bìa từ file	Creator	Upload ảnh bìa trực tiếp từ máy	File ảnh hợp lệ (jpg/png/webp...)	Ảnh được upload, xử lý và lưu thành công
Xóa ảnh bìa	Creator	Xóa một ảnh bìa không muốn dùng nữa	Là admin contributor	Ảnh bị xóa hoàn toàn khỏi DB + cloud storage
Cập nhật thứ tự ảnh bìa	Creator	Sắp xếp lại thứ tự hiển thị các ảnh bìa	Có ít nhất 2 ảnh bìa	Thứ tự hiển thị đúng như mong muốn
Download ảnh từ URL	Hệ thống	Tải ảnh từ link người dùng cung cấp	URL hợp lệ	Nhận được file ảnh gốc
Xử lý ảnh	Hệ thống	Resize, tạo thumbnail, lấy màu chủ đạo	Có file ảnh	Ảnh tối ưu + có preview + dominant color
Upload lên cloud storage	Hệ thống	Lưu ảnh gốc + preview lên CDN/cloud	Đã xử lý ảnh	Có URL ảnh ổn định, nhanh
Tạo record trong database	Hệ thống	Lưu thông tin ảnh bìa vào bảng BookCover	Đã upload thành công	Record đầy đủ: url, preview, size, color, index

Bảng 2.29 Dòng chính Usecase chức năng quản lý ảnh truyện

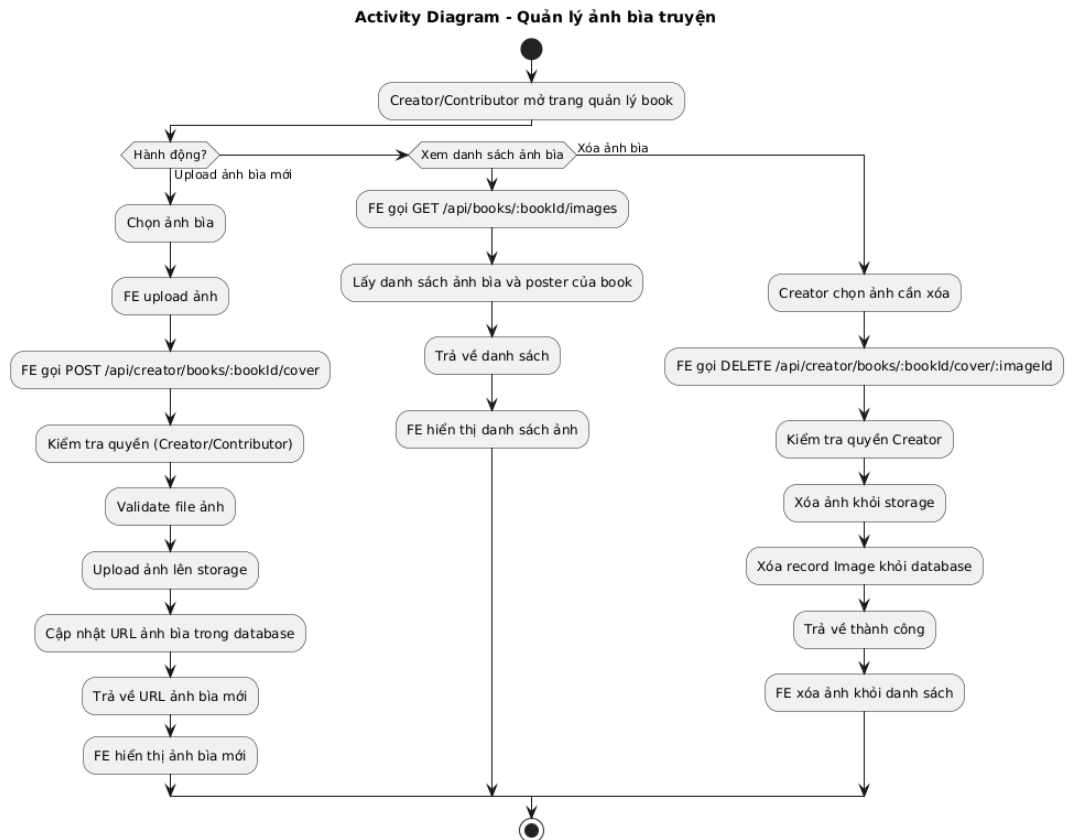
STT	Use Case	Thực hiện	Hành động chính
1	Quản lý ảnh bìa truyện	Creator	Vào trang chi tiết truyện → tab “Ảnh bìa”
2	Tạo ảnh bìa từ URL	Creator	Dán link ảnh → nhấn “Thêm từ URL”
3	Download ảnh từ URL	Hệ thống	Server tự động tải ảnh → kiểm tra định dạng + kích thước
4	Tạo ảnh bìa từ file	Creator	Kéo thả hoặc chọn file → upload trực tiếp
5	Xử lý ảnh	Hệ thống	Resize (nếu quá lớn) → tạo thumbnail → lấy dominant color
6	Upload lên cloud storage	Hệ thống	Upload cả ảnh gốc + preview lên cloud → lấy URL ổn định
7	Tạo record trong database	Hệ thống	Lưu vào bảng BookCover: url, previewUrl, width, height, dominantColor, index tự tăng
8	Cập nhật thứ tự ảnh bìa	Creator	Kéo thả sắp xếp lại hoặc nhấn nút lên/xuống → lưu thứ tự mới
9	Xóa ảnh bìa	Creator	Nhấn icon thùng rác trên ảnh → xác nhận → xóa record + xóa file trên cloud
10	Hoàn tất	Creator	Truyện có nhiều ảnh bìa đẹp → tự động đổi ngẫu nhiên hoặc theo thứ tự → giao diện cực chất

- Sơ đồ tuần tự chức năng quản lý ảnh truyện



Hình 2.50 Sơ đồ tuần tự chức năng quản lý ảnh truyện

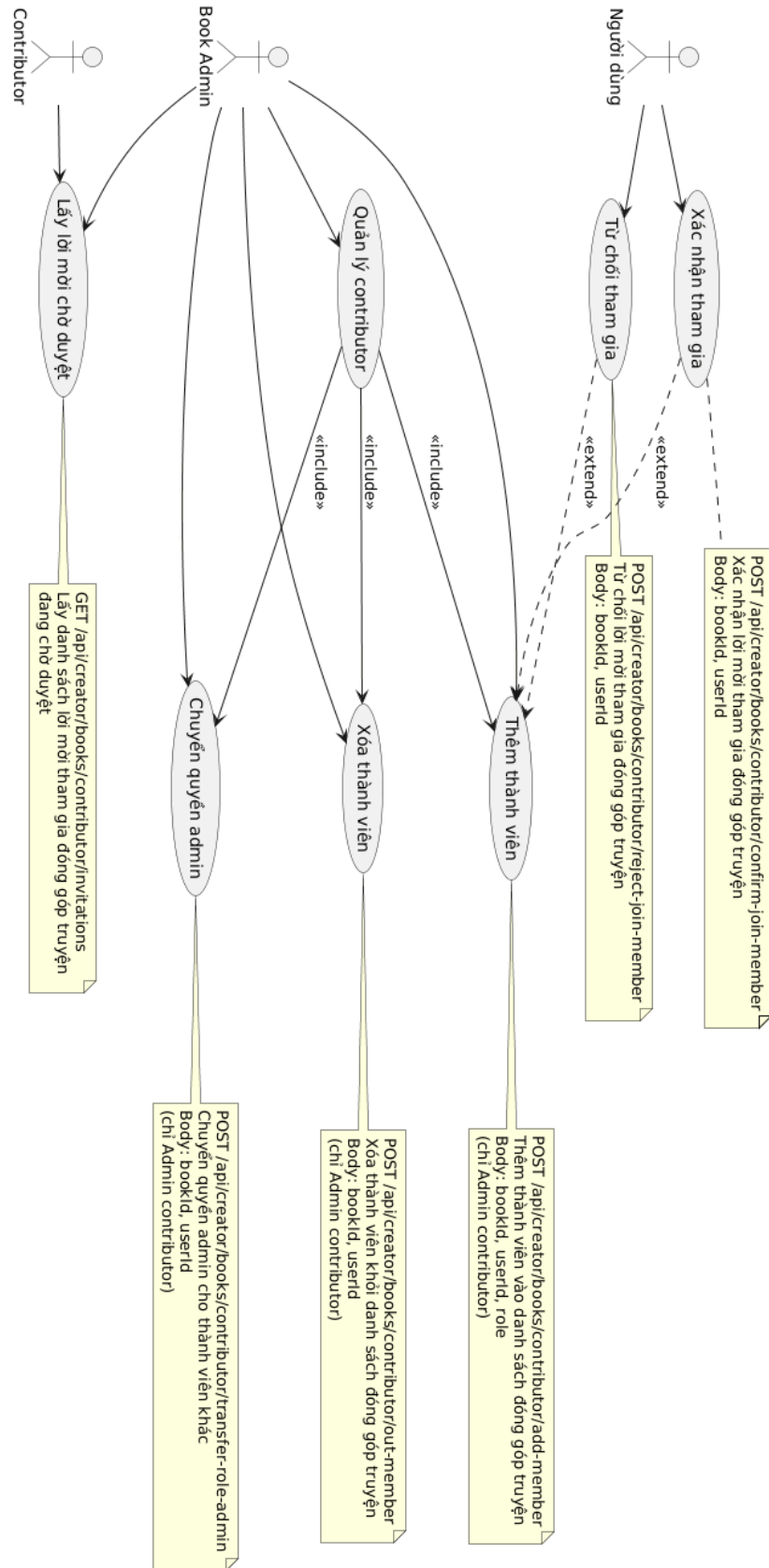
- Sơ đồ hoạt động chức năng quản lý ảnh truyện



Hình 2.51 Sơ đồ hoạt động chức năng quản lý ảnh truyện

2.2.14. Chức năng quản lý thành viên truyện

- Sơ đồ usecase chức năng quản lý thành viên truyện



Hình 2.52 Sơ đồ usecase chức năng quản lý thành viên truyện

Bảng 2.30 Mô tả Usecase chức năng quản lý thành viên truyện

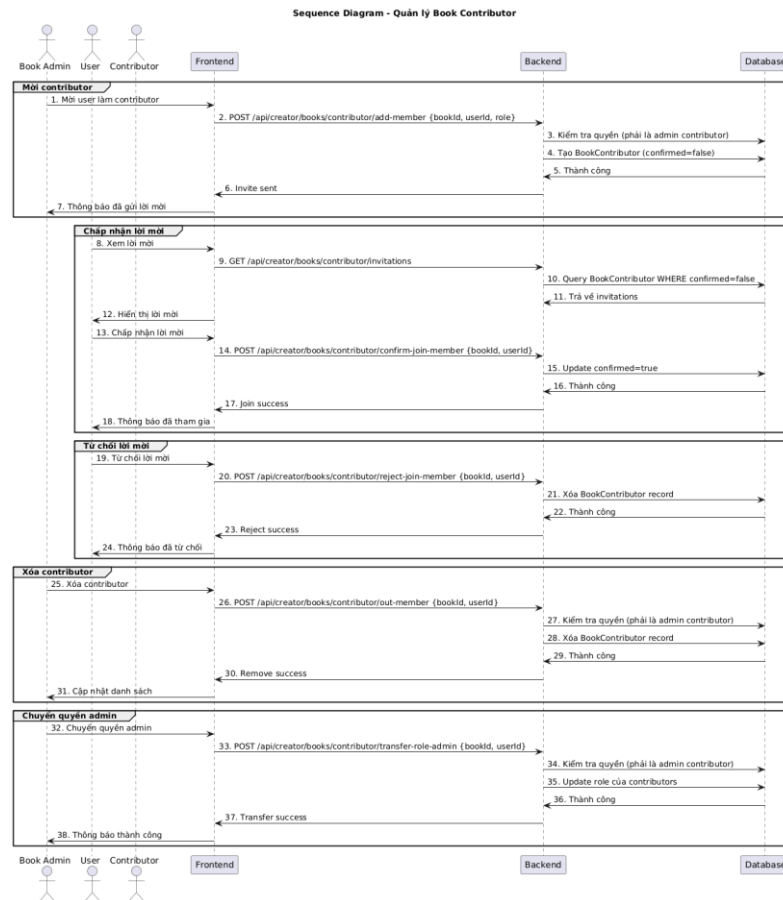
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý contributor	Book Admin	Thêm, xóa, chuyển quyền contributor của truyện	Là Admin contributor của truyện	Quyền quản lý truyện rõ ràng, an toàn
Lấy lời mời chờ duyệt	Book Admin / Contributor	Xem danh sách lời mời đang chờ xác nhận	Có lời mời đang pending	Hiển thị danh sách lời mời chi tiết
Xác nhận tham gia	Người dùng	Đồng ý trở thành contributor của truyện	Đã nhận được lời mời	Trở thành contributor chính thức
Từ chối tham gia	Người dùng	Từ chối lời mời đóng góp truyện	Đã nhận được lời mời	Lời mời bị hủy, không tham gia
Thêm thành viên	Book Admin	Mời user làm contributor	Là Admin contributor	Lời mời được gửi đi
Xóa thành viên	Book Admin	Kick contributor ra khỏi truyện	Là Admin contributor	Thành viên bị xóa khỏi danh sách
Chuyển quyền admin	Book Admin	Chuyển quyền Admin contributor cho người khác	Là Admin hiện tại	Quyền Admin được chuyển giao thành công

Bảng 2.31 Dòng chính Usecase chức năng quản lý thành viên truyện

STT	Use Case	Thực hiện	Hành động chính
1	Quản lý contributor	Book Admin	Vào trang truyện → tab “Đội ngũ” / “Contributor”
2	Thêm thành viên	Book Admin	Tìm user → chọn vai trò (editor, uploader...) → gửi lời mời

3	Xác nhận tham gia	Người được mời	Vào phần thông báo / lời mời → nhấn “Đồng ý”
4	Từ chối tham gia	Người được mời	Nhấn “Từ chối” → lời mời bị hủy
5	Lấy lời mời chờ duyệt	Book Admin	Vào tab contributor → xem danh sách “Đang chờ duyệt”
6	Xóa thành viên	Book Admin	Chọn contributor → nhấn “Xóa khỏi truyện” → xác nhận
7	Chuyển quyền admin	Book Admin	Chọn contributor đáng tin cậy → nhấn “Chuyển quyền Admin” → xác nhận
8	Hoàn tất	Tất cả	Truyện có thể được quản lý bởi nhiều người → quyền rõ ràng, chuyển giao dễ dàng → an toàn tuyệt đối

• Sơ đồ tuần tự chức năng quản lý thành viên truyện



([Liên kết](#) xem chi tiết)

Hình 2.53 Sơ đồ tuần tự chức năng quản lý thành viên truyện

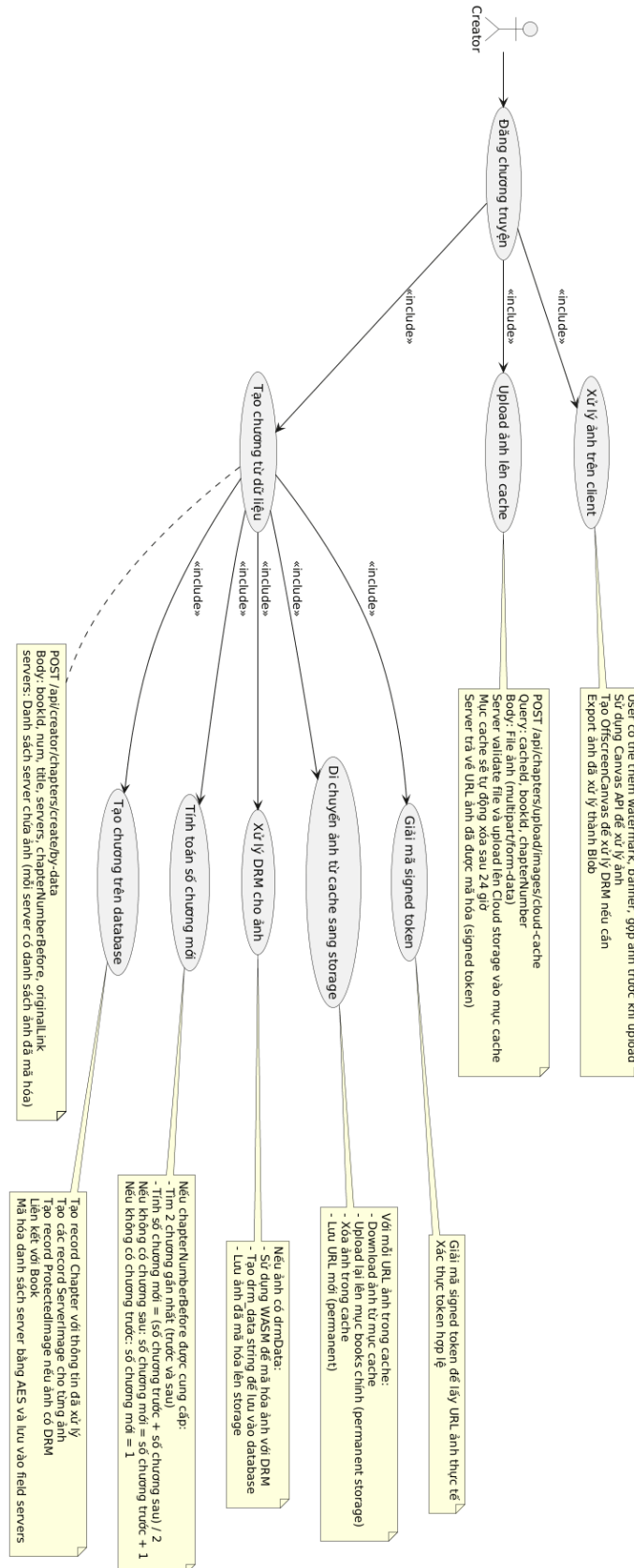
• Sơ đồ hoạt động chức năng quản lý thành viên truyện



Hình 2.54 Sơ đồ hoạt động chức năng quản lý thành viên truyện

2.2.15. Chức năng tạo chương

- Sơ đồ usecase chức năng tạo chương



Hình 2.55 Sơ đồ usecase chức năng tạo chương

Bảng 2.32 Mô tả Usecase chức năng tạo chương

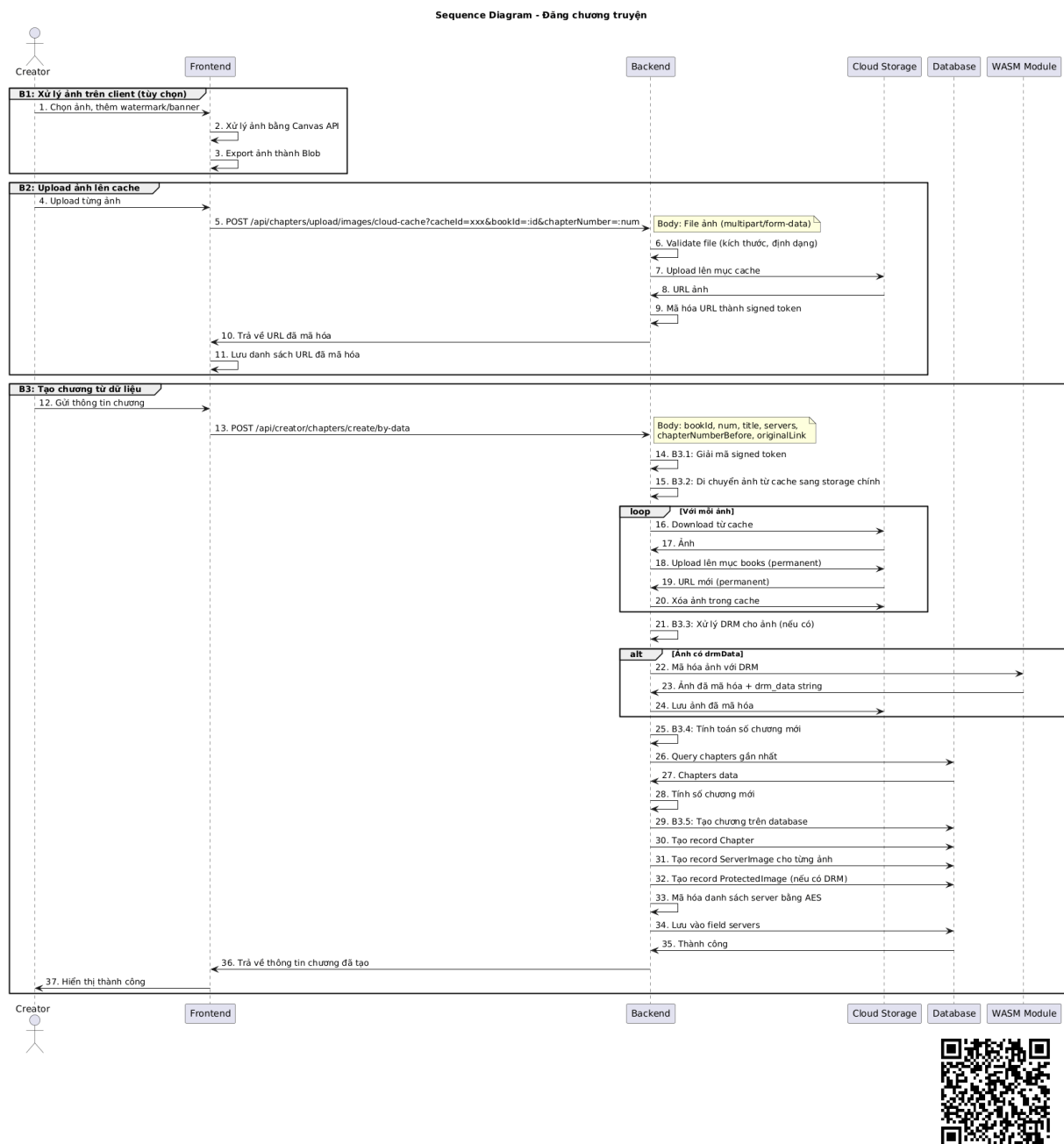
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Đăng chương truyện	Creator	Upload và đăng một chương mới cho truyện	Là contributor truyện, có quyền đăng	Chương được xuất bản thành công, hiển thị ngay
Xử lý ảnh trên client	Creator	Thêm watermark, banner, gộp ảnh, chuẩn bị DRM	Có ảnh gốc	Ảnh đã được chỉnh sửa theo ý Creator
Upload ảnh lên cache	Creator	Tạm lưu ảnh lên cloud cache để lấy signed token	Đã xử lý ảnh xong	Nhận danh sách signed token an toàn
Tạo chương từ dữ liệu	Creator	Gửi toàn bộ dữ liệu chương lên server để xử lý	Có signed token + thông tin chương	Server tiếp nhận và xử lý hoàn tất
Giải mã signed token	Hệ thống	Xác thực và lấy URL thật của ảnh trong cache	Nhận signed token	Lấy được ảnh từ cache
Di chuyển ảnh từ cache > storage	Hệ thống	Chuyển ảnh từ thư mục tạm sang thư mục chính thức	Ảnh đã được xác thực	Ảnh được lưu vĩnh viễn, cache tự xóa sau 24h
Xử lý DRM cho ảnh	Hệ thống	Mã hóa ảnh bằng WASM + tạo drm_data	User bật tùy chọn DRM	Ảnh được bảo vệ chống chụp màn hình/copy
Tính toán số chương mới	Hệ thống	Tự động tính số thứ tự chương (hỗ trợ chèn chương)	Có/không có chapterNumberBefore	Số chương chính xác, không bị trùng

Tạo chương trên database	Hệ thống	Lưu chapter + ảnh + server + DRM vào DB	Đã xử lý toàn bộ ảnh	Chapter chính thức xuất hiện trên hệ thống
--------------------------	----------	---	----------------------	--

Bảng 2.33 Dòng chính Usecase chức năng tạo chương

STT	Use Case	Thực hiện	Hành động chính
1	Đăng chương truyện	Creator	Vào trang truyện → nhấn “Đăng chương mới”
2	Xử lý ảnh trên client	Creator	Kéo thả ảnh → thêm watermark/banner → bật DRM nếu muốn → xuất Blob
3	Upload ảnh lên cache	Creator	Upload từng ảnh (hoặc hàng loạt) lên cache → nhận danh sách signed token
4	Tạo chương từ dữ liệu	Creator	Nhập số chương (hoặc để trống để chèn) → tiêu đề → original link → nhấn “Xuất bản”
5	Giải mã signed token	Hệ thống	Server giải mã token → lấy ảnh thật từ cache
6	Di chuyển ảnh từ cache > storage	Hệ thống	Tải ảnh từ cache → upload lại vào thư mục chính thức → xóa cache
7	Xử lý DRM cho ảnh	Hệ thống	Nếu bật DRM → dùng WASM mã hóa ảnh → tạo drm_data
8	Tính toán số chương mới	Hệ thống	Tự động tính số chương chính xác (hỗ trợ chèn giữa 2 chương)
9	Tạo chương trên database	Hệ thống	Tạo record Chapter → ServerImage → ProtectedImage (nếu có) → mã hóa AES danh sách server
10	Hoàn tất	Creator	Nhận thông báo “Đăng chương thành công” → chương hiển thị ngay → người đọc nhận thông báo mới

• Sơ đồ tuần tự chức năng tạo chương

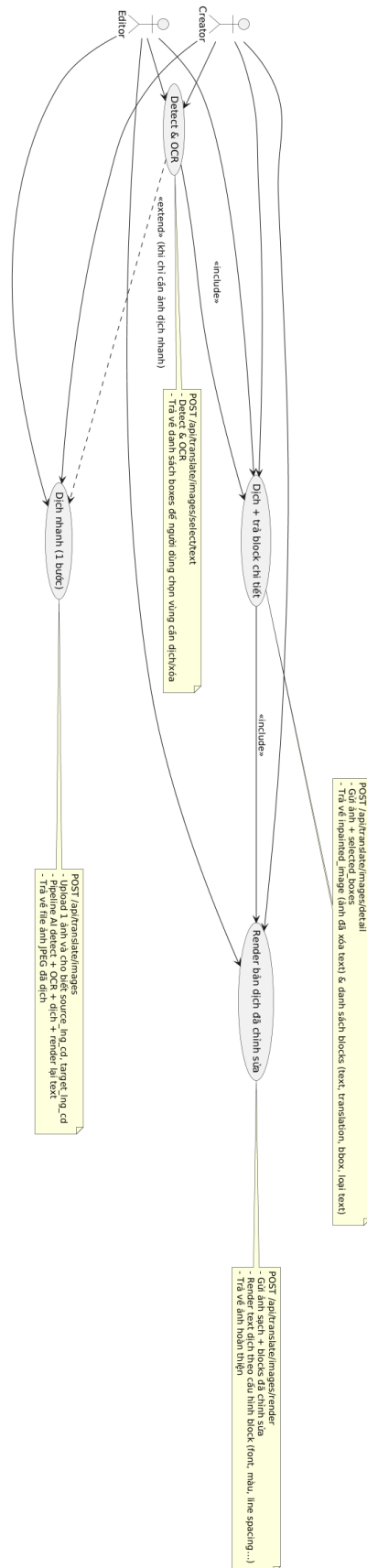


([Liên kết](#) xem chi tiết)

Hình 2.56 Sơ đồ tuần tự chức năng tạo chương

2.2.16. Chức năng dịch truyện với AI

- Sơ đồ usecase chức năng dịch truyện với AI



([Liên kết](#) xem chi tiết)

Hình 2.58 Sơ đồ usecase chức năng dịch truyện với AI

Bảng 2.34 Mô tả Usecase chức năng dịch truyền với AI

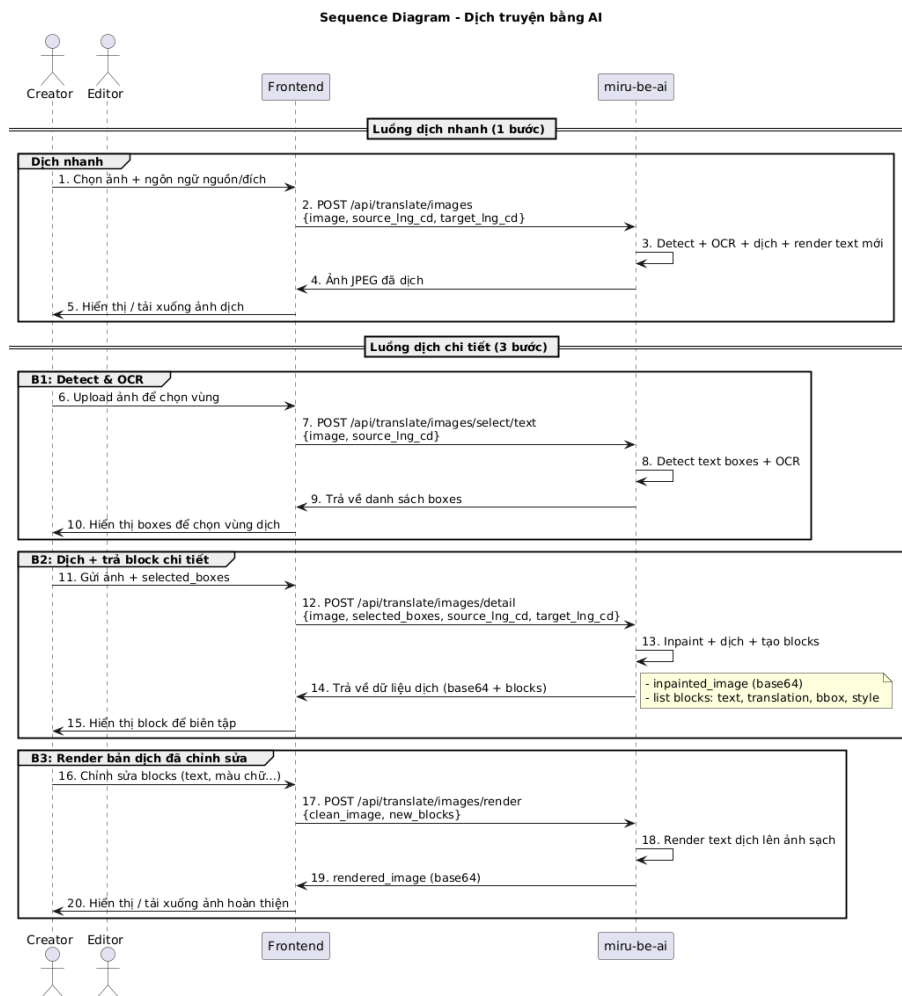
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Dịch nhanh (1 bước)	Creator / Editor	Dịch toàn bộ ảnh chỉ trong 1 click	Có ảnh gốc + chọn ngôn ngữ nguồn → đích	Nhận ngay ảnh đã dịch hoàn chỉnh (JPEG)
Detect & OCR	Creator / Editor	Phát hiện và trích xuất text từ ảnh	Có ảnh gốc	Trả về các box text + nội dung gốc để chọn vùng
Dịch + trả block chi tiết	Creator / Editor	Xóa text gốc → dịch → trả về ảnh sạch + block chi tiết	Đã chọn vùng text cần dịch	Ảnh đã xóa text + danh sách block đầy đủ (text, bản dịch, bbox...)
Render bản dịch đã chỉnh sửa	Creator / Editor	Vẽ lại bản dịch đã chỉnh sửa lên ảnh sạch	Có ảnh sạch + block đã chỉnh	Ảnh hoàn thiện đẹp, đúng font/màu/line spacing

Bảng 2.35 Dòng chính Usecase chức năng dịch truyền với AI

STT	Use Case	Thực hiện	Hành động chính
1	Dịch nhanh (1 bước)	Creator / Editor	Kéo ảnh vào tool → chọn ngôn ngữ nguồn → đích → nhấn “Dịch nhanh”
2	Dịch nhanh (1 bước)	Hệ thống	AI tự động Detect → OCR → Dịch → Render → trả ngay ảnh JPEG đã dịch hoàn chỉnh
3	Detect & OCR	Creator / Editor	Upload ảnh → nhấn “Detect text”
4	Detect & OCR	Hệ thống	Trả về các box vùng text + nội dung gốc → người dùng chọn vùng cần dịch / xóa
5	Dịch + trả block chi tiết	Creator / Editor	Chọn xong vùng → nhấn “Dịch chi tiết”

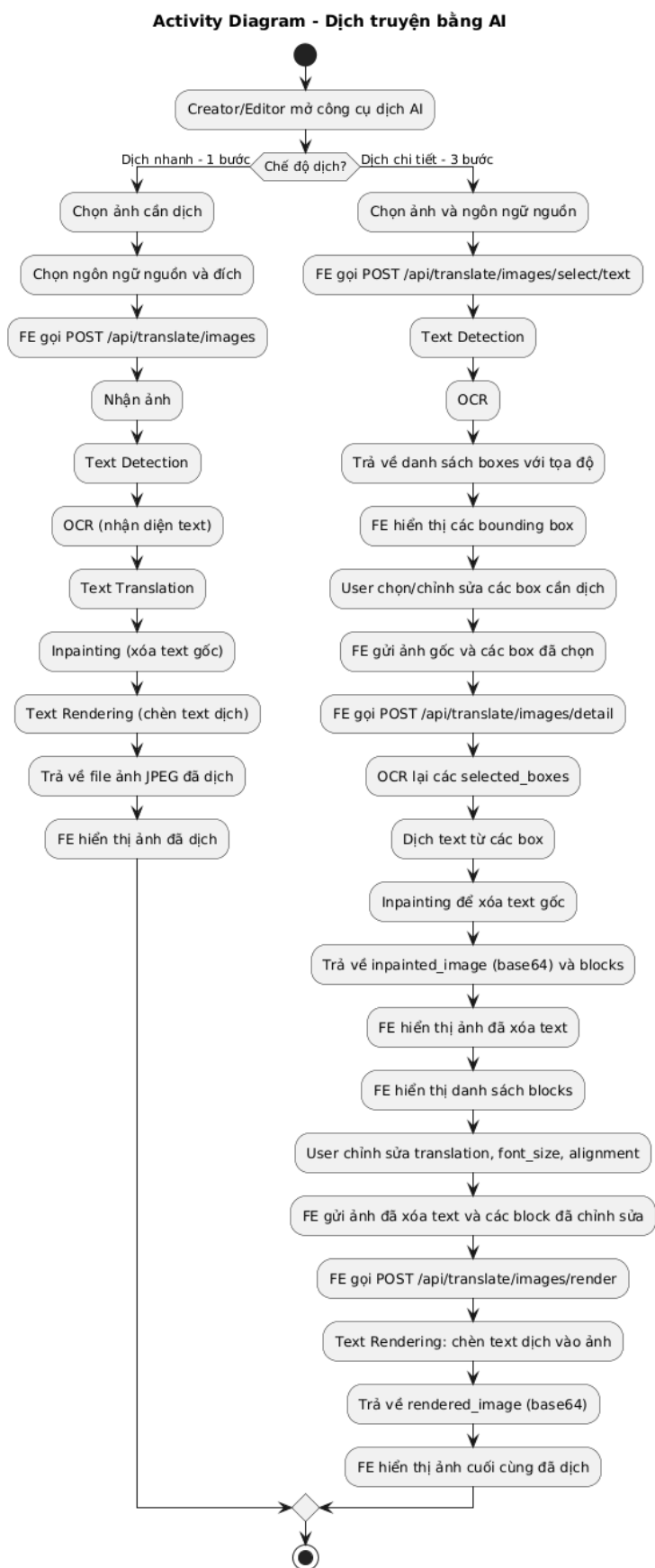
6	Dịch + trả block chi tiết	Hệ thống	AI xóa sạch text gốc (inpainting) → dịch từng block → trả ảnh sạch + danh sách block chi tiết
7	Render bản dịch đã chỉnh sửa	Creator / Editor	Chỉnh sửa bản dịch, font, màu, cỡ chữ, line spacing... trong editor
8	Render bản dịch đã chỉnh sửa	Creator / Editor	Nhấn “Render cuối cùng”
9	Render bản dịch đã chỉnh sửa	Hệ thống	Vẽ lại toàn bộ text đã chỉnh lên ảnh sạch → trả ảnh hoàn thiện chất lượng cao
10	Hoàn tất	Creator / Editor	Có 2 cách: dịch siêu nhanh 1-click hoặc dịch chi tiết chỉnh tay đẹp lung linh → dùng đăng chapter ngay

• Sơ đồ tuần tự chức năng dịch truyện với AI



Hình 2.59 Sơ đồ tuần tự chức năng dịch truyện với AI

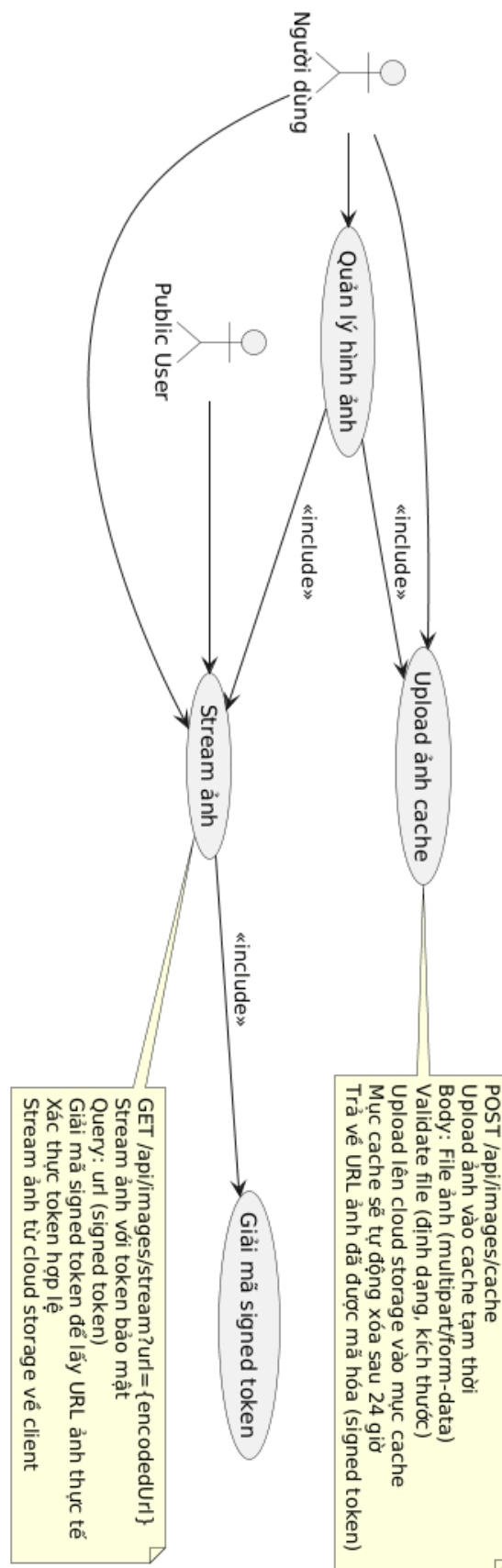
- Sơ đồ hoạt động chức năng dịch truyện với AI



Hình 2.60 Sơ đồ hoạt động chức năng dịch truyện với AI

2.2.17. Chức năng quản lí hình ảnh

- Sơ đồ usecase chức năng quản lí hình ảnh



Hình 2.61 Sơ đồ usecase chức năng quản lí hình ảnh

Bảng 2.36 Mô tả Usecase chức năng quản lí hình ảnh

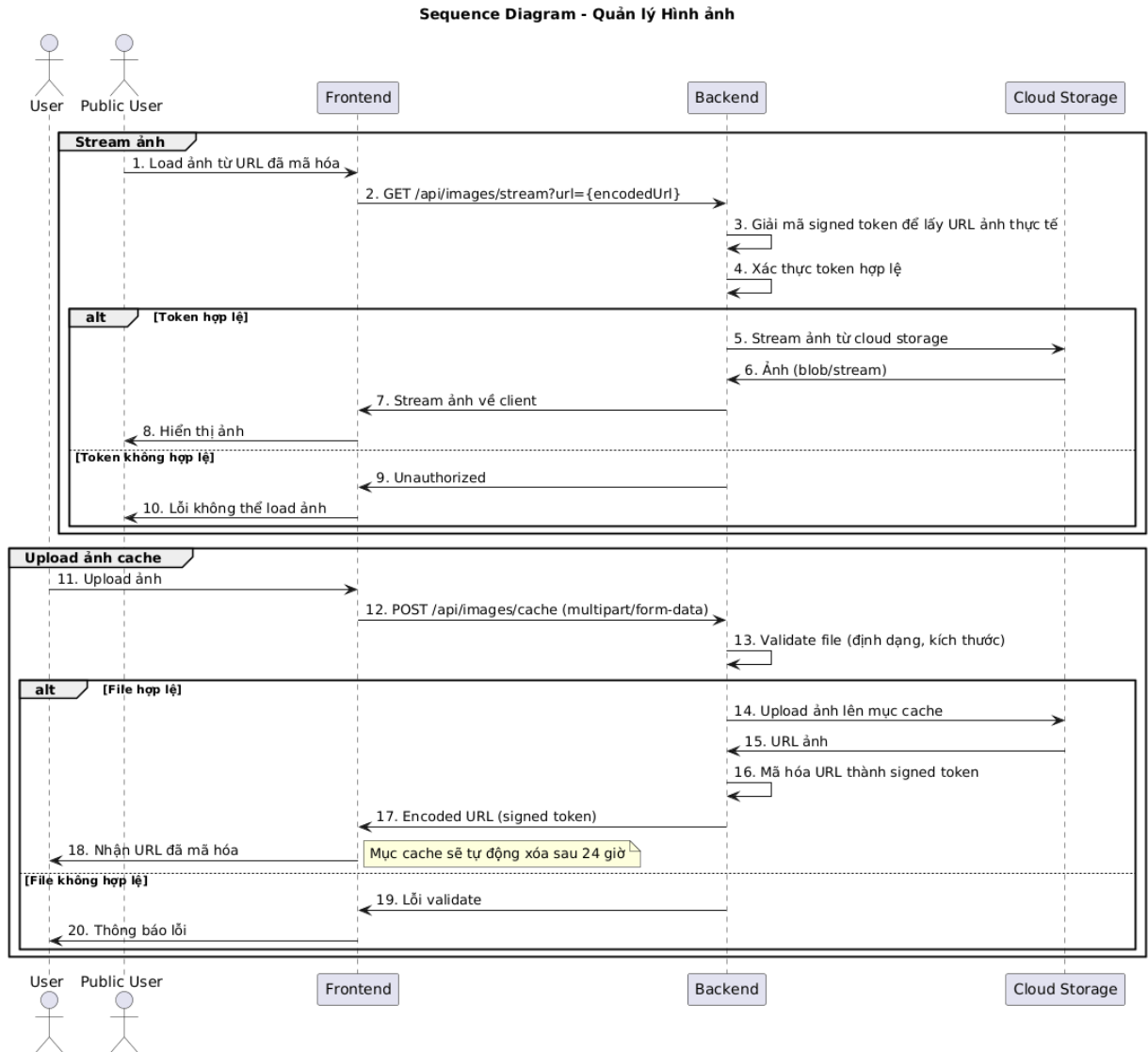
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý hình ảnh	Người dùng	Upload tạm + hiển thị ảnh an toàn	Đã đăng nhập	Ảnh được upload nhanh và stream bảo mật
Stream ảnh	Public User / User	Xem ảnh (bình luận, avatar, ảnh bìa, chat...)	Có signed token hợp lệ	Ảnh hiển thị nhanh, không bị hotlink
Upload ảnh cache	Người dùng	Upload ảnh tạm vào cache để lấy token bảo mật	Đã đăng nhập	Nhận signed token dùng 1 lần, hết hạn sau 24h
Giải mã signed token	Hệ thống	Xác thực và lấy URL ảnh thật từ token	Nhận request stream	Stream đúng ảnh + chống truy cập trực tiếp

Bảng 2.37 Dòng chính Usecase chức năng quản lí hình ảnh

STT	Use Case	Thực hiện	Hành động chính
1	Quản lý hình ảnh	Người dùng	Cần đăng ảnh (bình luận, đổi avatar, gửi chat, thêm ảnh bìa...)
2	Upload ảnh cache	Người dùng	Chọn ảnh → upload lên endpoint cache
3	Upload ảnh cache	Hệ thống	Validate file → upload vào thư mục cache (tự xóa sau 24h) → tạo signed token → trả về token
4	Stream ảnh	Người dùng/Public	FE nhận token → gắn vào
5	Giải mã signed token	Hệ thống	Server nhận request → giải mã token → kiểm tra hợp lệ & chưa hết hạn
6	Stream ảnh	Hệ thống	Lấy ảnh thật từ cloud storage (cache hoặc permanent) → stream trực tiếp về browser

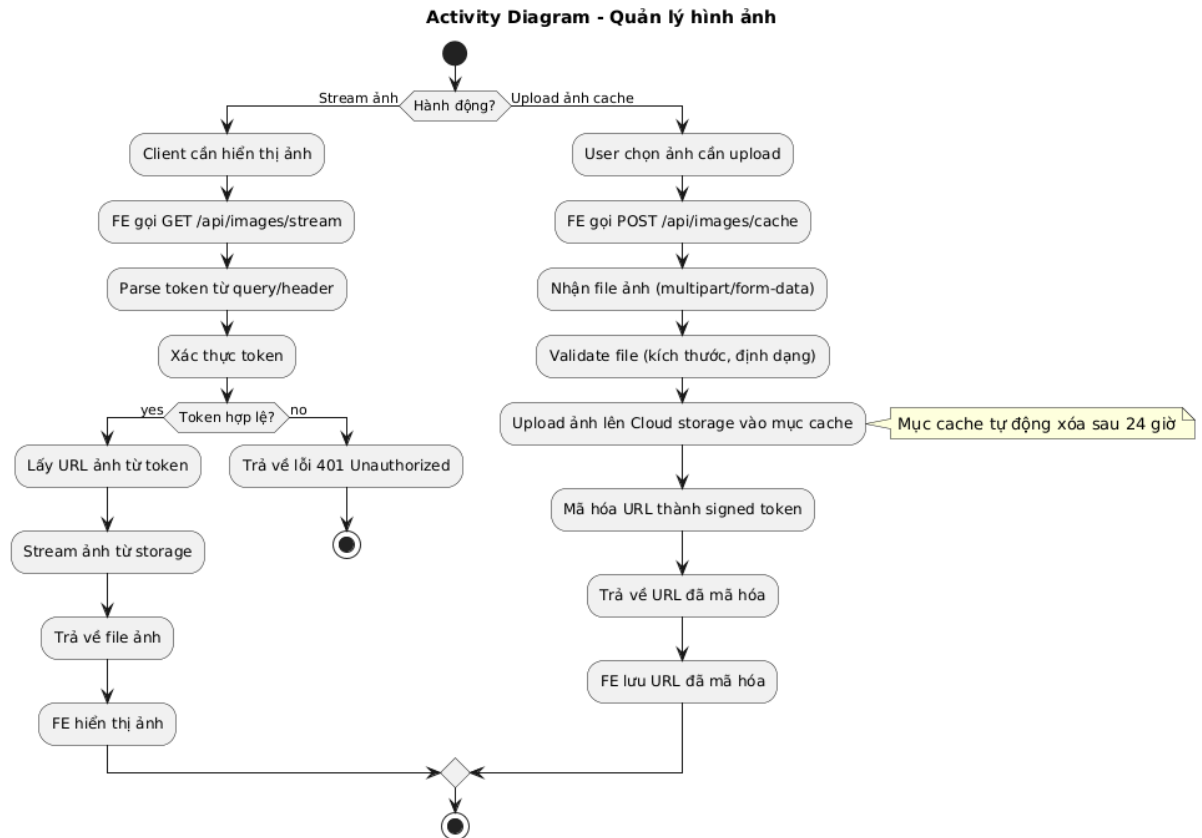
7	Hoàn tất	Người dùng/Public	Ảnh hiển thị tức thì, cực mượt → không ai truy cập trực tiếp link gốc → bảo mật 100%
---	----------	-------------------	--

- Sơ đồ tuần tự chức năng quản lý hình ảnh



Hình 2.62 Sơ đồ tuần tự chức năng quản lý hình ảnh

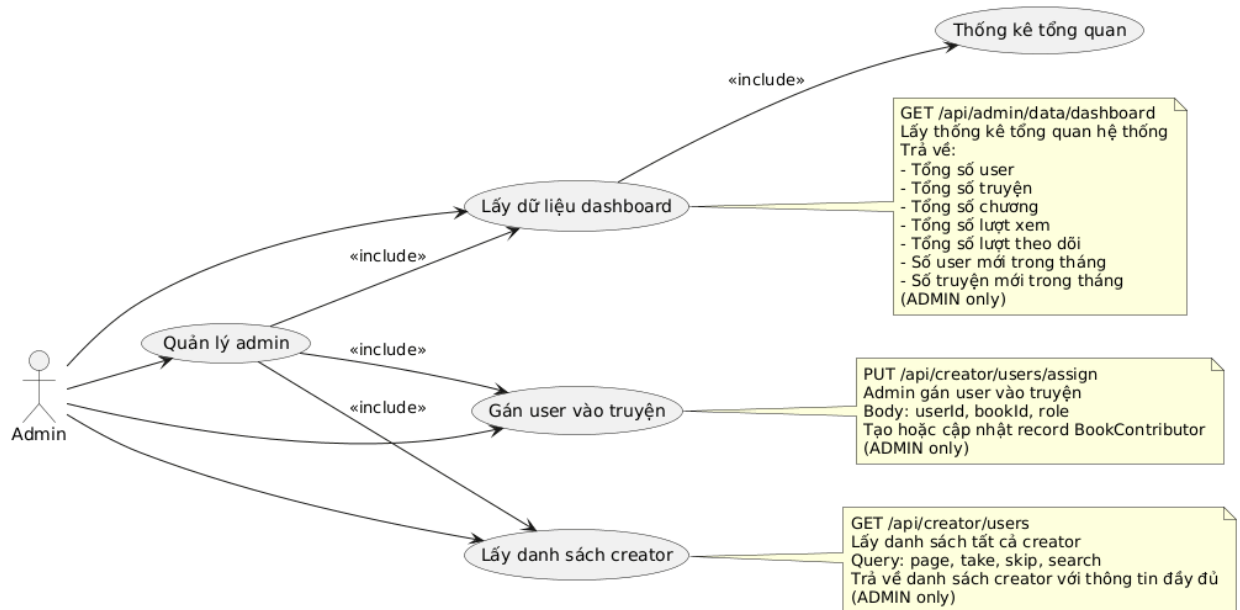
- Sơ đồ hoạt động chức năng quản lý hình ảnh



Hình 2.63 Sơ đồ hoạt động chức năng quản lý hình ảnh

2.2.18. Chức năng quản lý admin

- Sơ đồ usecase chức năng quản lý admin



Hình 2.64 Sơ đồ usecase chức năng

Bảng 2.38 Mô tả Usecase chức năng quản lý admin

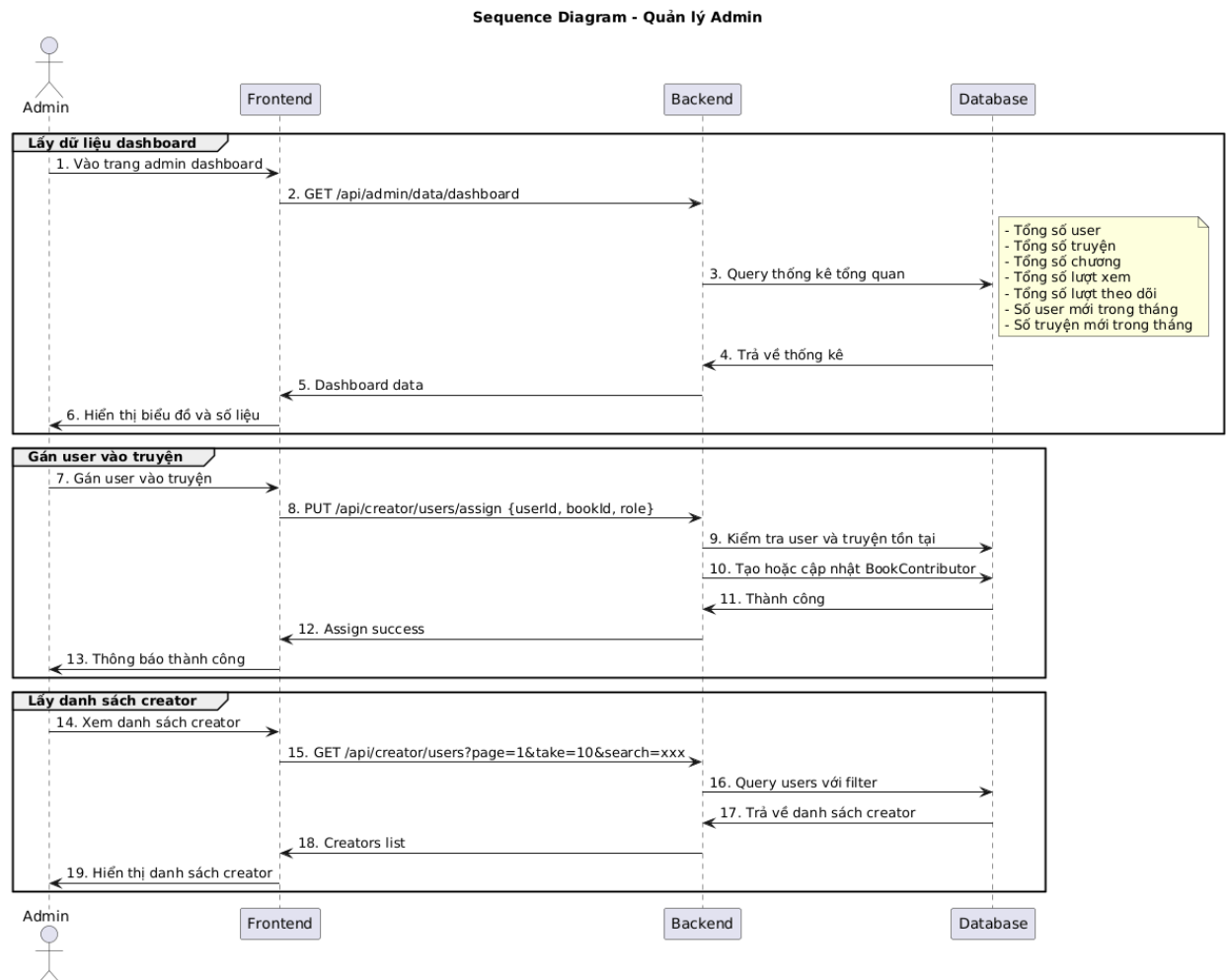
Tên Use Case	Tác nhân	Mục đích chính	Điều kiện tiên quyết	Kết quả mong đợi
Quản lý admin	Admin	Toàn quyền quản trị hệ thống	Là Super Admin	Thực hiện mọi thao tác admin nhanh chóng, chính xác
Lấy dữ liệu dashboard	Admin	Xem tổng quan số liệu toàn hệ thống	Là Super Admin	Hiển thị realtime: user, truyện, chương, view...
Gán user vào truyện	Admin	Gán thủ công user làm contributor cho truyện	Là Super Admin	Tạo/cập nhật BookContributor ngay lập tức
Lấy danh sách creator	Admin	Xem và tìm kiếm toàn bộ creator trên hệ thống	Là Super Admin	Danh sách phân trang + tìm kiếm chính xác
Thống kê tổng quan	Hệ thống	Cung cấp số liệu realtime cho dashboard	Gọi từ UC2	Dữ liệu luôn mới nhất, không cần refresh

Bảng 2.39 Dòng chính Usecase chức năng quản lý admin

STT	Use Case	Thực hiện	Hành động chính
1	Quản lý admin	Admin	Đăng nhập tài khoản Super Admin → vào Admin Panel
2	Lấy dữ liệu dashboard	Admin	Mở trang Dashboard
3	Thống kê tổng quan	Hệ thống	Tự động query → trả realtime: tổng user, truyện, chương, view, follow, user mới, truyện mới...
4	Lấy danh sách creator	Admin	Vào tab “Creator Management” → tìm kiếm theo tên/email

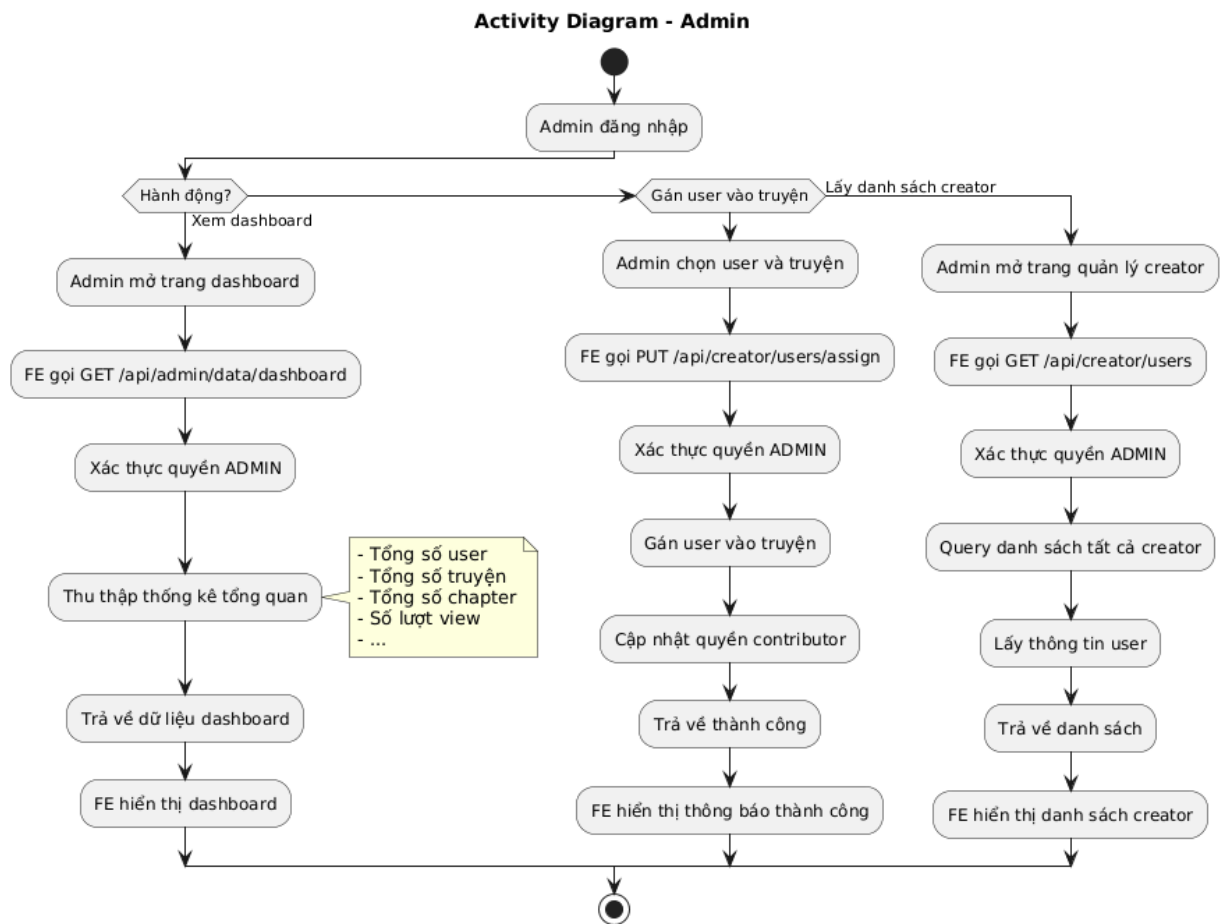
5	Lấy danh sách creator	Hệ thống	Trả danh sách creator phân trang + thông tin chi tiết
6	Gán user vào truyện	Admin	Chọn user → chọn truyện → chọn role (admin/editor/uploader...) → nhấn “Gán quyền”
7	Gán user vào truyện	Hệ thống	Tạo/cập nhật record BookContributor ngay lập tức → thông báo thành công
8	Hoàn tất	Admin	Quản trị toàn hệ thống cực nhanh → fix lỗi quyền, xem số liệu realtime, gán contributor chỉ 1 click

- Sơ đồ tuần tự chức năng quản lý admin



Hình 2.65 Sơ đồ tuần tự chức năng quản lý admin

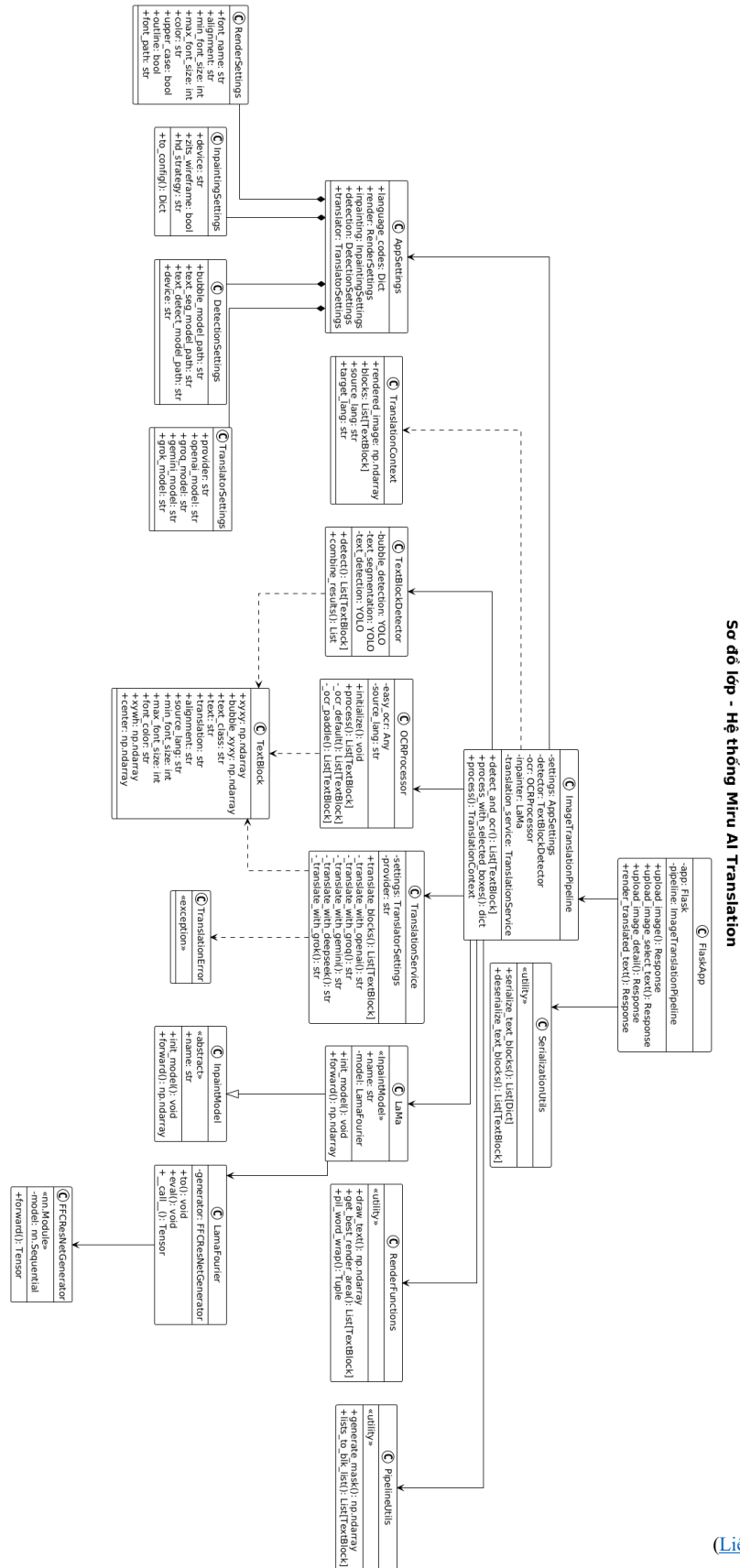
- Sơ đồ usecase chức năng quản lí admin



Hình 2.66 Sơ đồ usecase chức năng quản lí admin

2.4. Thiết kế cơ sở dữ liệu

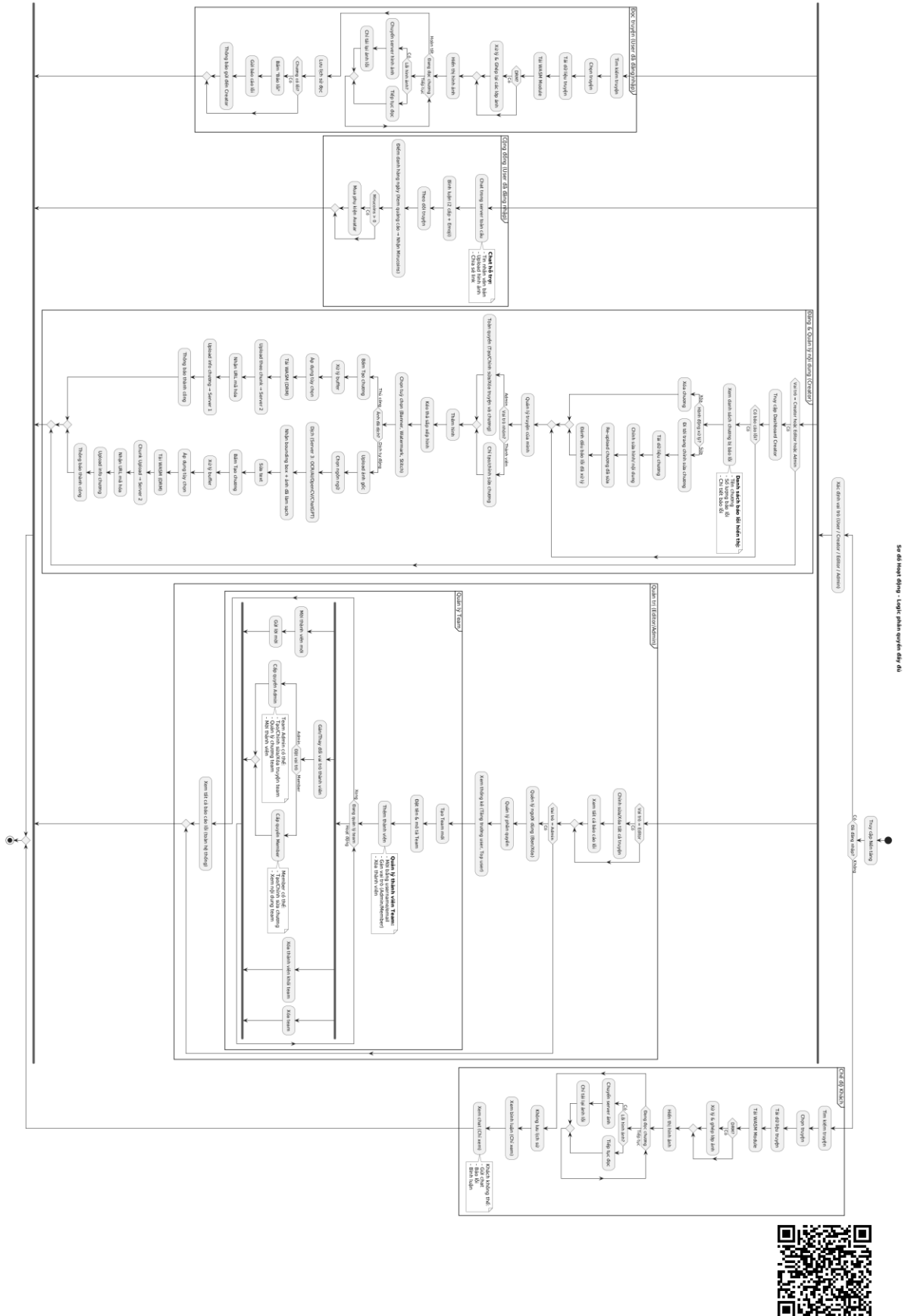
2.4.1. Sơ đồ lớp



([Liên kết](#) xem chi tiết)

Hình 2.67 Sơ đồ lớp

2.4.2. Sơ đồ hoạt động



Hình 2.69 Sơ đồ hoạt động

([Liên kết](#) xem chi tiết)

2.5. Danh mục các bảng

Bảng 2.40 Các bảng trong database

TT	Tên bảng	Mô tả
1	user	Lưu thông tin người dùng
2	user_item	Liên kết người dùng và vật phẩm họ sở hữu
3	item	Thông tin vật phẩm
4	user_team	Liên kết người dùng với đội nhóm
5	team	Thông tin đội nhóm
6	book_contributor	Liên kết người dùng đóng góp cho sách
7	book	Thông tin sách
8	chapter	Thông tin chương sách
9	server	Máy chủ chứa nội dung chương
10	protected_image	Hình ảnh được bảo vệ DRM
11	comment	Bình luận của người dùng
12	user_book_follow	Người dùng theo dõi sách
13	chat	Tin nhắn chat giữa người dùng
14	user_view	Lượt xem sách/chương của người dùng
15	banner_ads	Quảng cáo banner
16	report_chapter	Thống kê báo cáo chương
17	meta	Thông tin meta (thể loại, tag, tác giả...)
18	image	Hình ảnh liên kết với sách, meta, đội nhóm

Bảng 2.41 Bảng user

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
user_id	Int	Không	Primary	Mã người dùng
name	String	Không		Tên người dùng
username	String	Không	Unique	Tên đăng nhập
email	String	Không	Unique	Email người dùng
password	String	Có		Mật khẩu
description	Text	Có		Mô tả người dùng
is_bot	Boolean	Không		Là bot hay không
rank	Float	Không		Điểm xếp hạng
avatar_url	String	Có		URL avatar
attendance	Text	Có		Điểm danh
attendance_history	JSON	Có		Lịch sử điểm danh
banned	Int	Không		Trạng thái bị cấm
ban_unlock_at	DateTime	Có		Thời gian mở cấm
ban_type	Enum	Có		Loại cấm
ban_count	Int	Không		Số lần bị cấm
facebook_url	String	Có		URL Facebook
web_url	String	Có		URL website
balance	Int	Không		Số dư tài khoản
strength_mapping	Enum	Không		Mức sức mạnh
created_at	DateTime	Không		Ngày tạo
updated_at	DateTime	Không		Ngày cập nhật
role_id	Int	Không	Foreign	Liên kết với bảng role

Bảng 2.42 Bảng user_item

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
item_id	Int	Không	Primary (cùng user_id)	Mã vật phẩm

user_id	Int	Không	Primary (cùng item_id)	Mã người dùng
equipped_item	Boolean	Không		Trang bị hay không
created_at	DateTime	Không		Ngày tạo

Bảng 2.43 Bảng item

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
item_id	Int	Không	Primary	Mã vật phẩm
name	String	Có		Tên vật phẩm
spacing	Int	Có		Khoảng cách
image_original_url	String	Không		URL ảnh gốc
image_preview	String	Không		URL ảnh preview
image_small	String	Không		URL ảnh nhỏ
price	Float	Không		Giá
category	Enum	Không		Loại vật phẩm

Bảng 2.44 Bảng user_item

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
user_team_id	Int	Không	Primary	Mã user_team
role	String	Có		Vai trò trong team
joined_at	DateTime	Không		Ngày tham gia
confirmed	Boolean	Không		Xác nhận tham gia
user_id	Int	Không	Unique, Foreign	Mã người dùng
team_id	Int	Không	Foreign	Mã team

Bảng 2.45 Bảng team

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
team_id	Int	Không	Primary	Mã team
slug	String	Không	Unique	Đường dẫn đội nhóm
name	String	Không		Tên đội nhóm

description	Text	Có		Mô tả
is_ads	Boolean	Không		Có quảng cáo hay không
facebook	String	Có		URL Facebook
created_at	DateTime	Không		Ngày tạo
updated_at	DateTime	Không		Ngày cập nhật

Bảng 2.46 Bảng book_contributor

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
user_id	Int	Không	Primary (cùng book_id)	Mã người dùng
book_id	Int	Không	Primary (cùng user_id)	Mã sách
role	String	Có		Vai trò đóng góp
joined_at	DateTime	Không		Ngày tham gia
confirmed	Boolean	Không		Xác nhận

Bảng 2.47 Bảng book

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
book_id	Int	Không	Primary	Mã sách
category	String	Không		Thẻ loại
original_link	String(500)	Có		Link gốc
next	String	Có		Chương tiếp theo
type	String	Có		Loại sách
is_adult	Boolean	Không		Sách 18+
is_featured	Boolean	Không		Nổi bật
is_delete	Boolean	Không		Đã xóa
auto_generate	Boolean	Không		Tự động tạo
storage	Enum	Không		Trạng thái lưu trữ
created_at	DateTime	Không		Ngày tạo
updated_at	DateTime	Không		Ngày cập nhật

new_chapter_at	DateTime	Không		Ngày có chương mới
posted_by_id	Int	Có	Foreign	Người đăng
team_id	Int	Có	Foreign	Liên kết team

Bảng 2.48 Bảng chapter

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
book_id	Int	Không	Primary (cùng chapter_number)	Mã sách
chapter_number	Float	Không	Primary (cùng book_id)	Số chương
num	String	Không		Số thứ tự chương
next	String(500)	Có		Chương tiếp theo
title	String	Có		Tiêu đề chương
thumbnail	String	Có		Ảnh đại diện
original_link	String(500)	Có		Link gốc
views_count	Int	Không		Lượt xem
is_protected	Boolean	Không		Bảo vệ nội dung
source	String	Không		Nguồn
created_at	DateTime	Không		Ngày tạo
updated_at	DateTime	Không		Ngày cập nhật
status	Enum	Không		Trạng thái
storage	Enum	Không		Lưu trữ

Bảng 2.49 Bảng server

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
server_id	Int	Không	Primary	Mã server
order	Int	Có		Thứ tự
cloud	String	Không		Loại cloud
content	Text	Không		Nội dung
created_at	DateTime	Không		Ngày tạo
updated_at	DateTime	Không		Ngày cập nhật
book_id	Int	Không	Foreign	Mã sách
chapter_number	Float	Không	Foreign	Số chương
book_server_domain_id	Int	Có		Liên kết domain

Bảng 2.50 Bảng protected_image

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
order	Int	Không	Primary (cùng server_id)	Thứ tự ảnh
server_id	Int	Không	Primary (cùng order), Foreign	Mã server
width	Int	Không		Chiều rộng
height	Int	Không		Chiều cao
image_url	String	Không		URL ảnh
key_secure	Int	Có		Khóa bảo mật
secure_image_url	String	Có		URL ảnh bảo mật
drm_data	String	Có		Dữ liệu DRM

Bảng 2.51 Bảng comment

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
comment_id	Int	Không	Primary	Mã bình luận

comment_text	Text	Không		Nội dung bình luận
is_read	Boolean	Không		Đã đọc
created_at	DateTime	Không		Ngày tạo
updated_at	DateTime	Không		Ngày cập nhật
image_index	Int	Có		Index ảnh
book_id	Int	Không	Foreign	Mã sách
chapter_number	Float	Có	Foreign	Số chương
sender_id	Int	Không	Foreign	Người gửi
receiver_id	Int	Có	Foreign	Người nhận
parent_id	Int	Có	Foreign	Bình luận cha

Bảng 2.52 Bảng user_book_follow

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
user_id	Int	Không	Primary (cùng book_id), Foreign	Mã người dùng
book_id	Int	Không	Primary (cùng user_id), Foreign	Mã sách

Bảng 2.53 Bảng chat

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
chat_id	Int	Không	Primary	Mã chat
socket_id	String	Không		Socket ID
chat_text	Text	Không		Nội dung chat
format	Enum	Không		Loại chat
created_at	DateTime	Không		Ngày tạo
sender_id	Int	Có	Foreign	Người gửi

Bảng 2.54 Bảng user_view

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
user_id	Int	Không	Primary (cùng book_id), Foreign	Mã người dùng
book_id	Int	Không	Primary (cùng user_id), Foreign	Mã sách
chapter_number	Float	Không	Foreign	Số chương
ip	String	Có		IP người dùng
updated_at	DateTime	Không		Ngày cập nhật

Bảng 2.55 Bảng report_chapter

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
book_id	Int	Không	Primary (cùng chapter_number), Foreign	Mã sách
chapter_number	Float	Không	Primary (cùng book_id), Foreign	Số chương
count	Int	Không		Số lượt báo cáo
created_at	DateTime	Không		Ngày tạo
updated_at	DateTime	Không		Ngày cập nhật

Bảng 2.56 Bảng meta

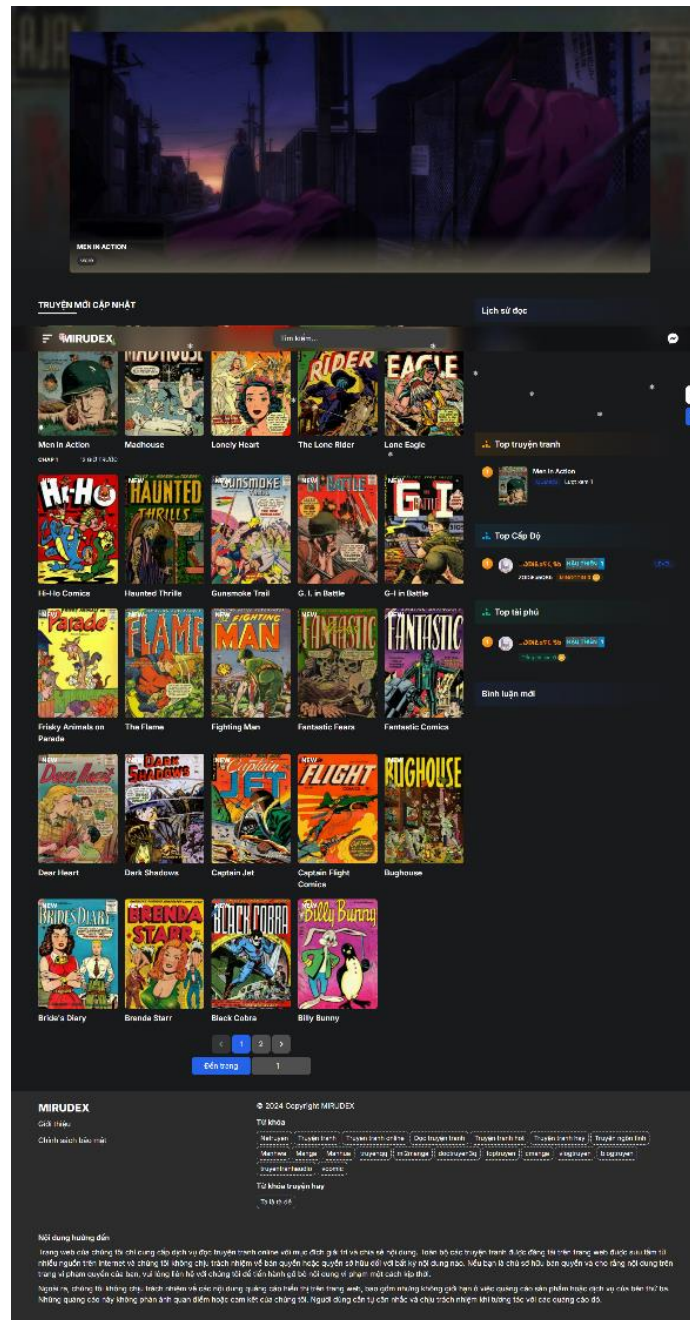
Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
meta_id	Int	Không	Primary	Mã meta
name	String	Không		Tên meta (tag, genre, author,...)
slug	String	Không	Unique	Đường dẫn định danh
description	Text	Có		Mô tả

type	Enum	Không		Loại meta (TAG, GENRE, SEASON,...)
created_at	DateTime	Không		Ngày tạo
updated_at	DateTime	Không		Ngày cập nhật

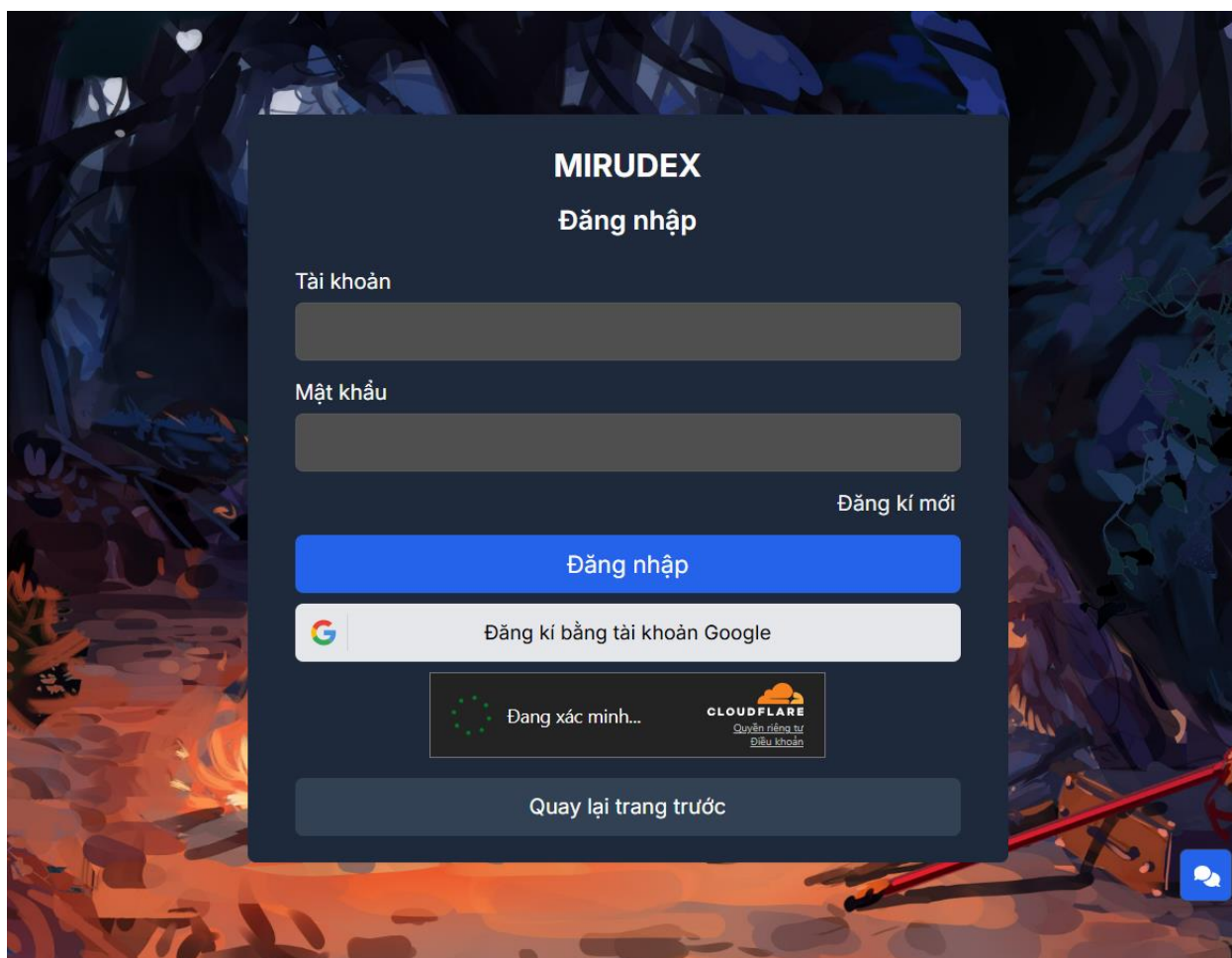
Bảng 2.57 Bảng image

Tên trường	Kiểu dữ liệu	Null	Key	Ghi chú
image_id	Int	Không	Primary	Mã ảnh
image_type	String	Có		Loại ảnh
width	Int	Có		Chiều rộng ảnh
height	Int	Có		Chiều cao ảnh
index	Int	Không		Vị trí ảnh trong danh sách
url	String	Không		URL ảnh
source	String	Có		Nguồn ảnh
dominant_color	String	Có		Màu chủ đạo
meta_id	Int	Có	Foreign	Liên kết Meta
preview_book_id	Int	Có	Foreign	Ảnh preview sách
cover_book_id	Int	Có	Foreign	Ảnh cover sách
poster_book_id	Int	Có	Foreign	Ảnh poster sách
preview_team_id	Int	Có	Foreign	Ảnh preview đội nhóm
cover_team_id	Int	Có	Foreign	Ảnh cover đội nhóm
poster_team_id	Int	Có	Foreign	Ảnh poster đội nhóm

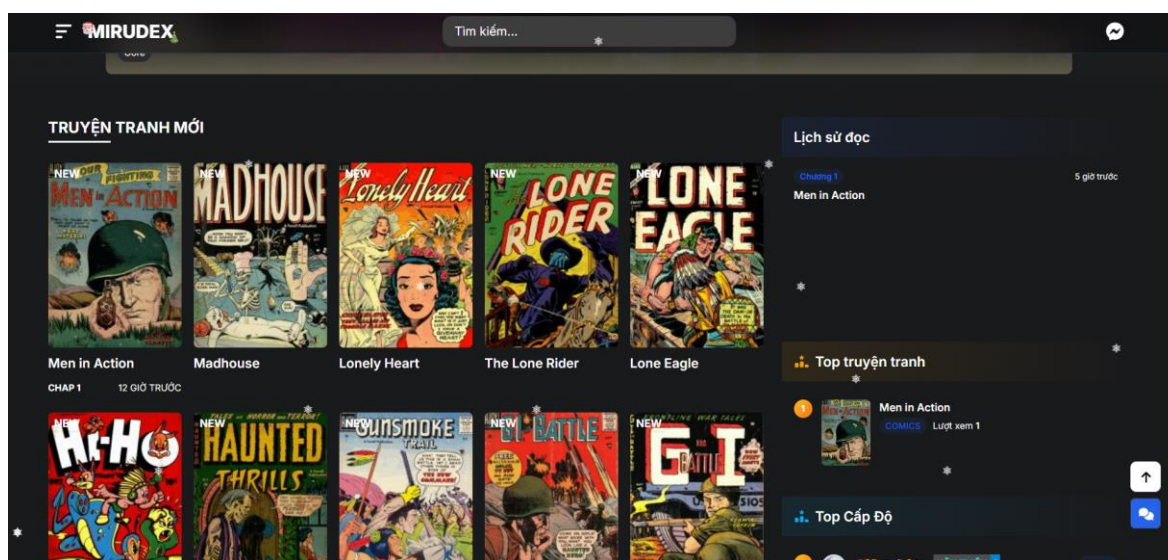
2.6. Thiết kế giao diện



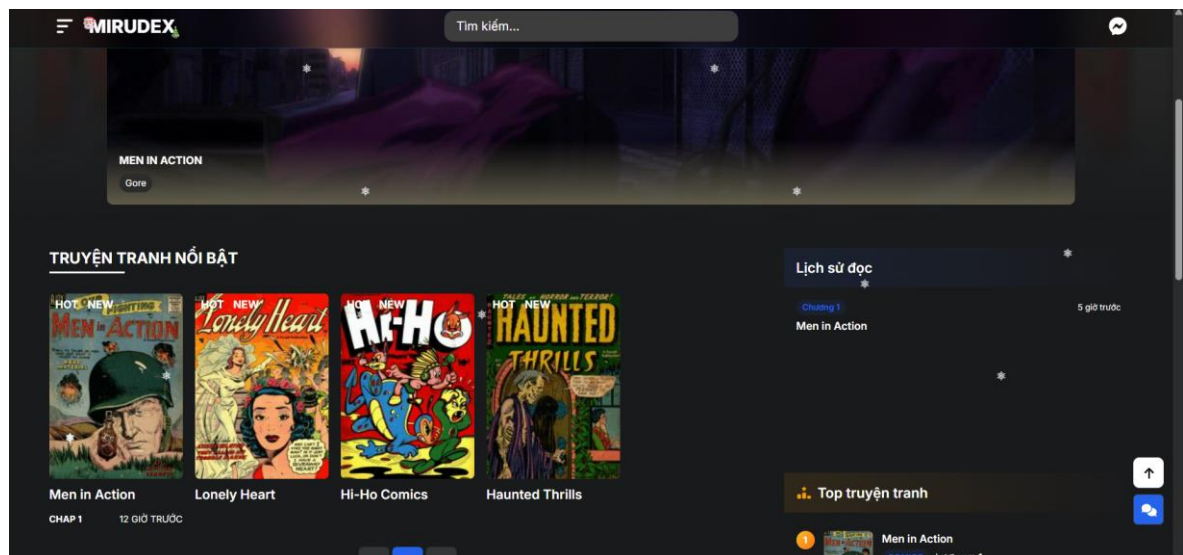
Hình 2.70 Màn hình trang chủ



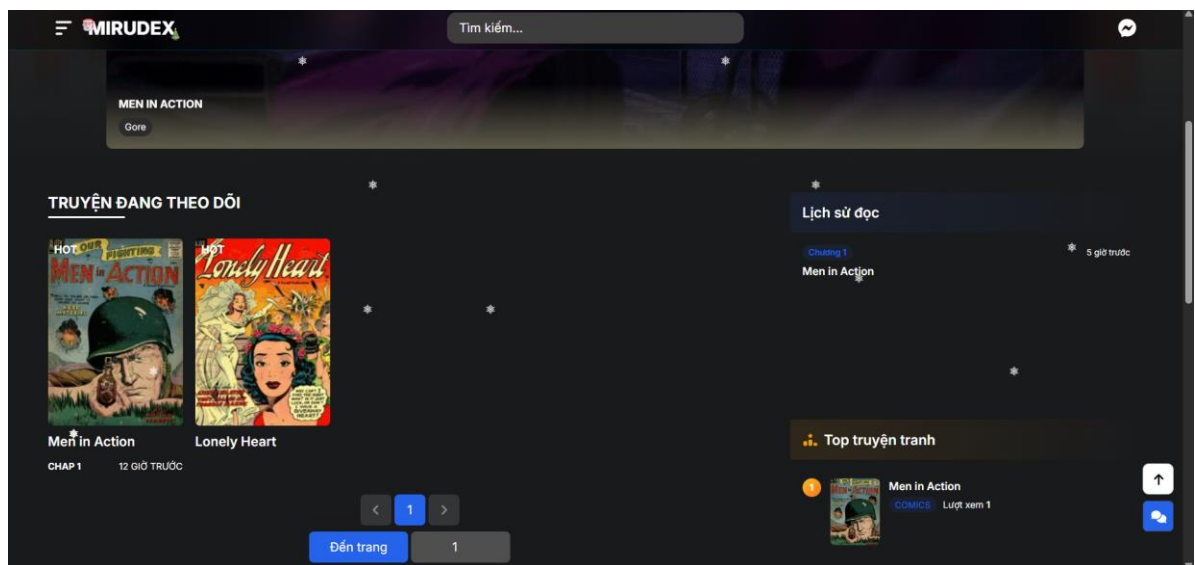
Hình 2.71 Màn hình đăng nhập



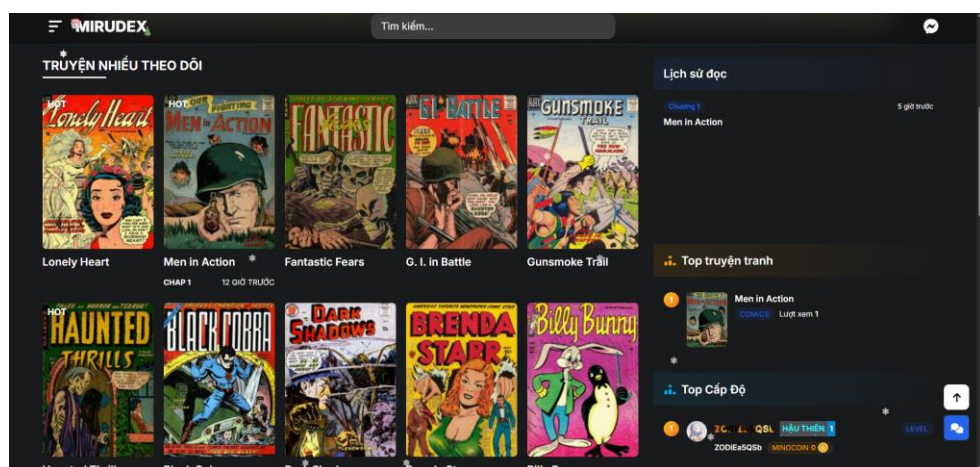
Hình 2.72 Màn hình danh sách truyện mới



Hình 2.73 Màn hình danh sách truyện nổi bật



Hình 2.74 Màn hình danh sách truyện theo dõi



Hình 2.75 Màn hình danh sách truyện nhiều theo dõi

MIRUDEX > Men in Action

MEN IN ACTION

THEO DÕI

ĐỌC TỪ ĐẦU

ĐỌC MỚI NHẤT

GORE

NỘI DUNG

Men in Action được cập nhật nhanh và đầy đủ nhất tại MIRUDEX. Bạn đọc đừng quên để lại bình luận và chia sẻ, ủng hộ MIRUDEX ra các chương mới nhất của truyện

Xem thêm

CHIA SẺ

0 Theo dõi

1 Lượt xem

Tiêu đề thay thế

Men in Action

Chính sách bình luận

- Cấm spam, chia sẻ liên kết không liên quan hoặc quảng cáo.
- Cấm sử dụng ngôn từ tục tĩu, đối trudy, từ ngữ thô tục.
- Cấm sử dụng ngôn ngữ xúc phạm, phân biệt hoặc gây thù hằn giữa các cá nhân, nhóm người.

Mọi hành vi được cho là vi phạm sẽ được xem xét và khóa tài khoản vĩnh viễn!

DANH SÁCH CHƯƠNG

DS.ẢNH

ĐỌC TIẾP

#1

Lượt xem: 1

12 giờ trước

Đổi tên ngay

MIRU ADMIN

Gửi

<

>

Đến trang

1

TRUYỆN CÙNG THỂ LOẠI

TỪ KHÓA

MEN IN ACTION RAW

MEN IN ACTION MANGA

MEN IN ACTION COMICS

MEN IN ACTION ANIME

MEN IN ACTION DONGHUA

MIRUDEX

Giới thiệu

Chính sách bảo mật

© 2024 Copyright MIRUDEX

Từ khóa

Netruyen

Truyện tranh

Truyen tranh online

Đọc truyện tranh

Truyện tranh hot

Truyện tranh hay

Truyện ngôn tình

Manhwa

Manga

Manhua

truyenqq

mi2manga

doctruyen3q

toptruyen

cmanga

vlogtruyen

blogtruyen

truyentranhaudio

vcomic

Từ khóa truyện hay

Ta là tà đế

Nội dung hướng đến

Trang web của chúng tôi chỉ cung cấp dịch vụ đọc truyện tranh online với mục đích giải trí và chia sẻ nội dung. Toàn bộ các truyện tranh được đăng tải trên trang web được sưu tầm từ nhiều nguồn trên internet và chúng tôi không chịu trách nhiệm về bản quyền hoặc quyền sở hữu đối với bất kỳ nội dung nào. Nếu bạn là chủ sở hữu bản quyền và cho rằng nội dung trên trang vi phạm quyền của bạn, vui lòng liên hệ với chúng tôi để tiến hành gỡ bỏ nội dung vi phạm một cách kịp thời.

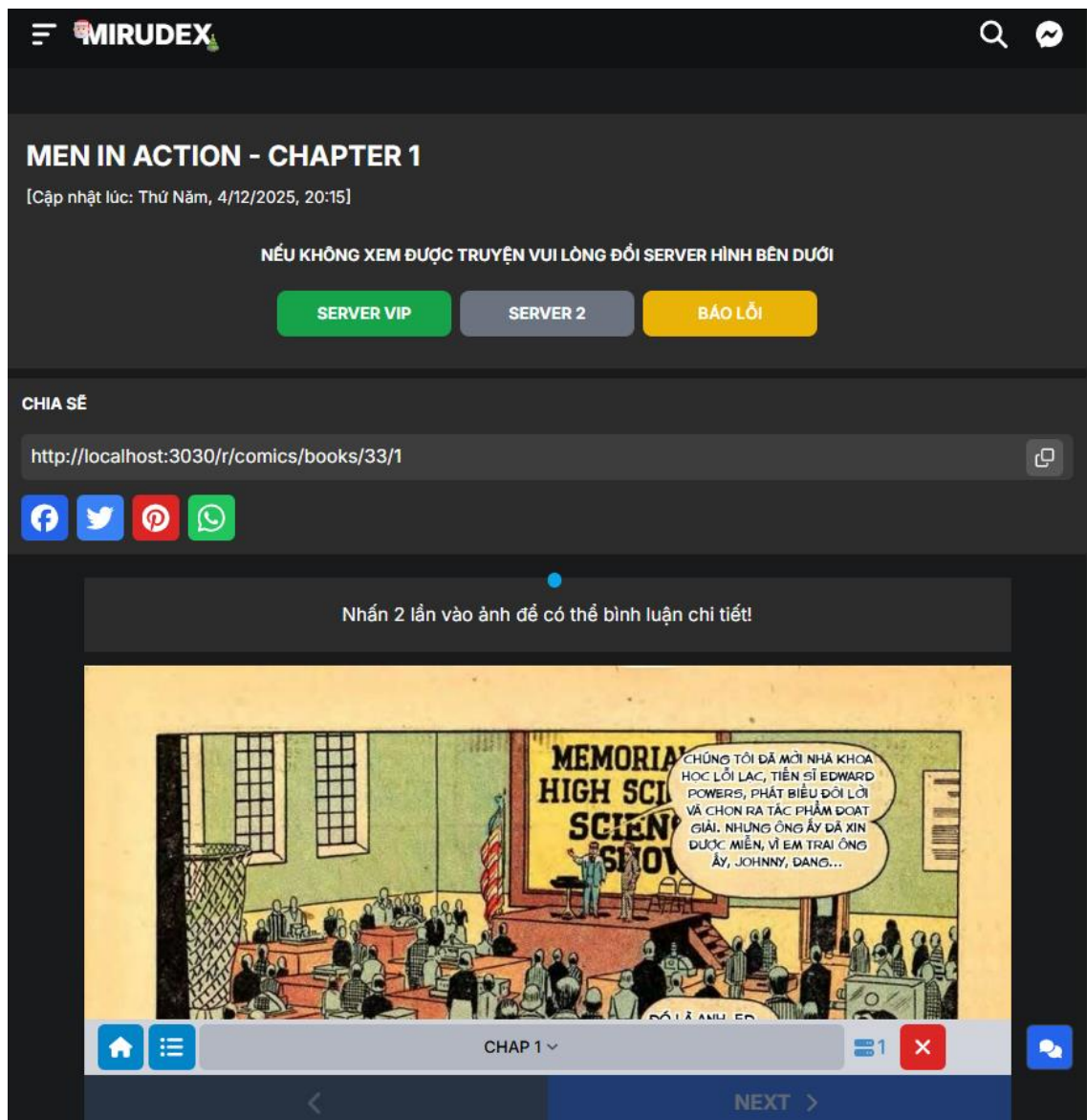
Ngoài ra, chúng tôi không chịu trách nhiệm về các nội dung quảng cáo hiển thị trên trang web, bao gồm nhưng không giới hạn ở việc quảng cáo sản phẩm hoặc dịch vụ của bên thứ ba. Những quảng cáo này không phản ánh quan điểm hoặc cam kết của chúng tôi. Người dùng cần tự cân nhắc và chịu trách nhiệm khi tương tác với các quảng cáo đó.

Hình 2.76 Màn hình chi tiết truyện

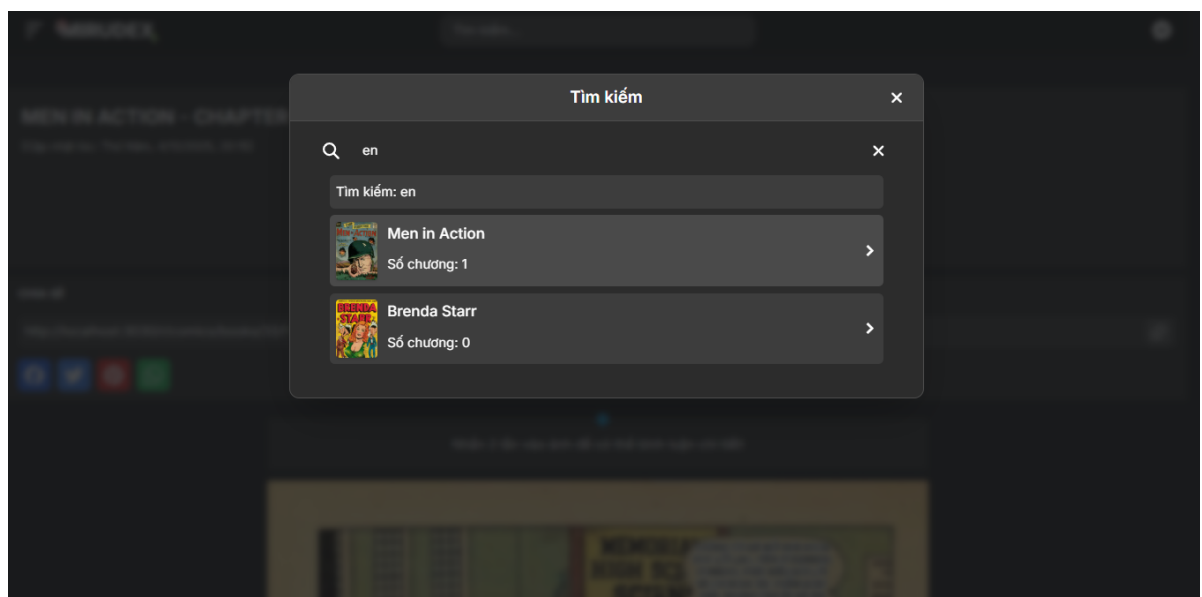
Sinh viên thực hiện: Nguyễn Hoàng Bảo

Hướng dẫn: ThS. Nguyễn Tất Phú Cường

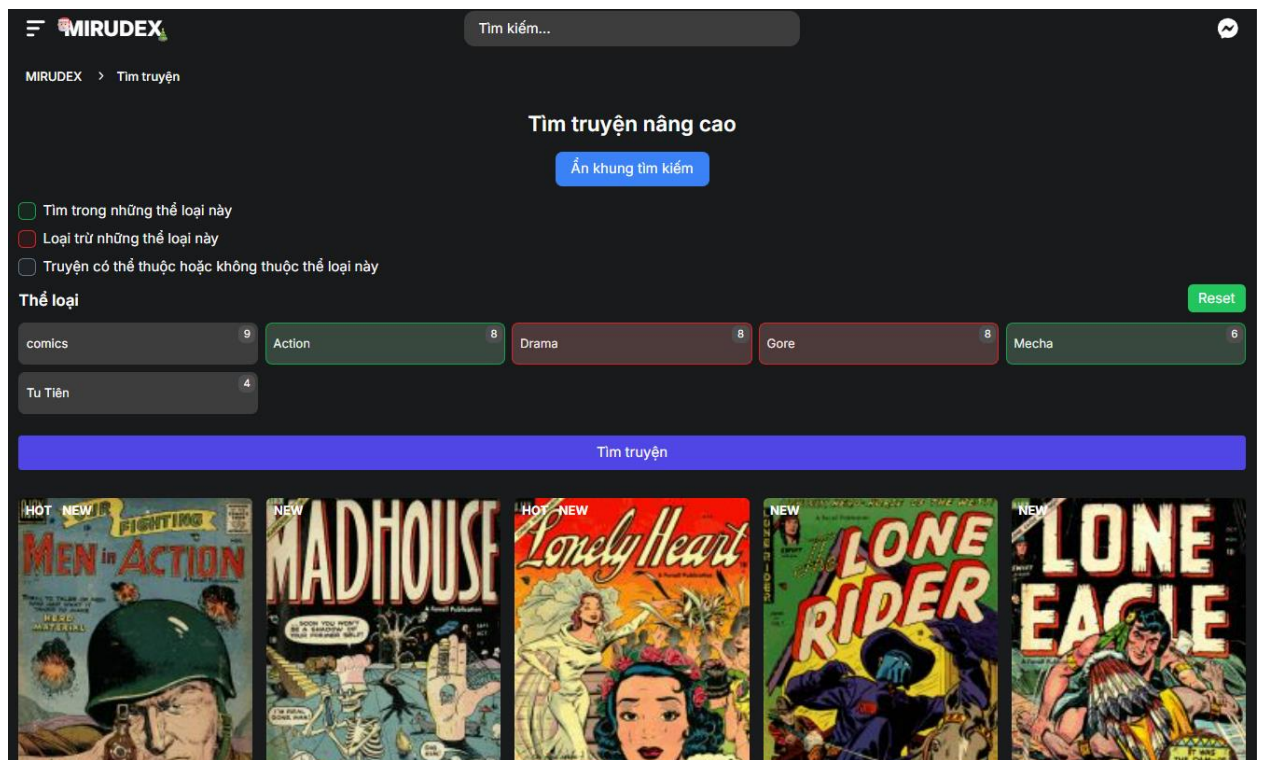
124



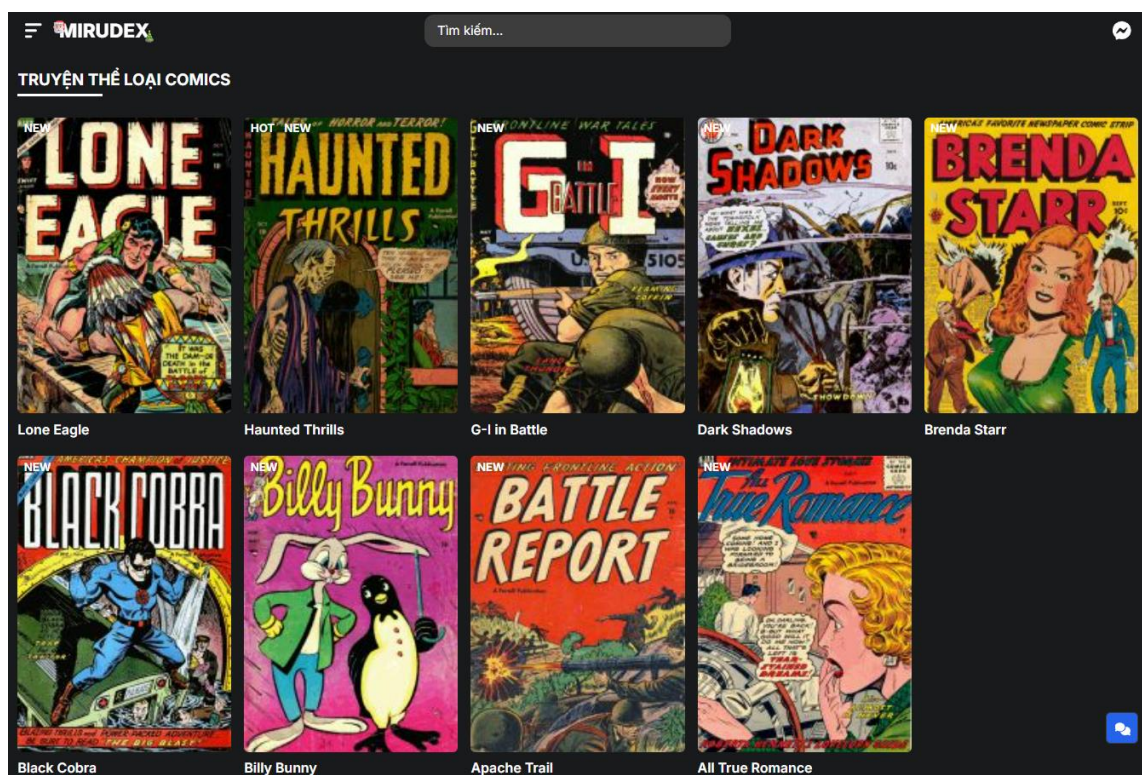
Hình 2.77 Màn hình chi tiết chương



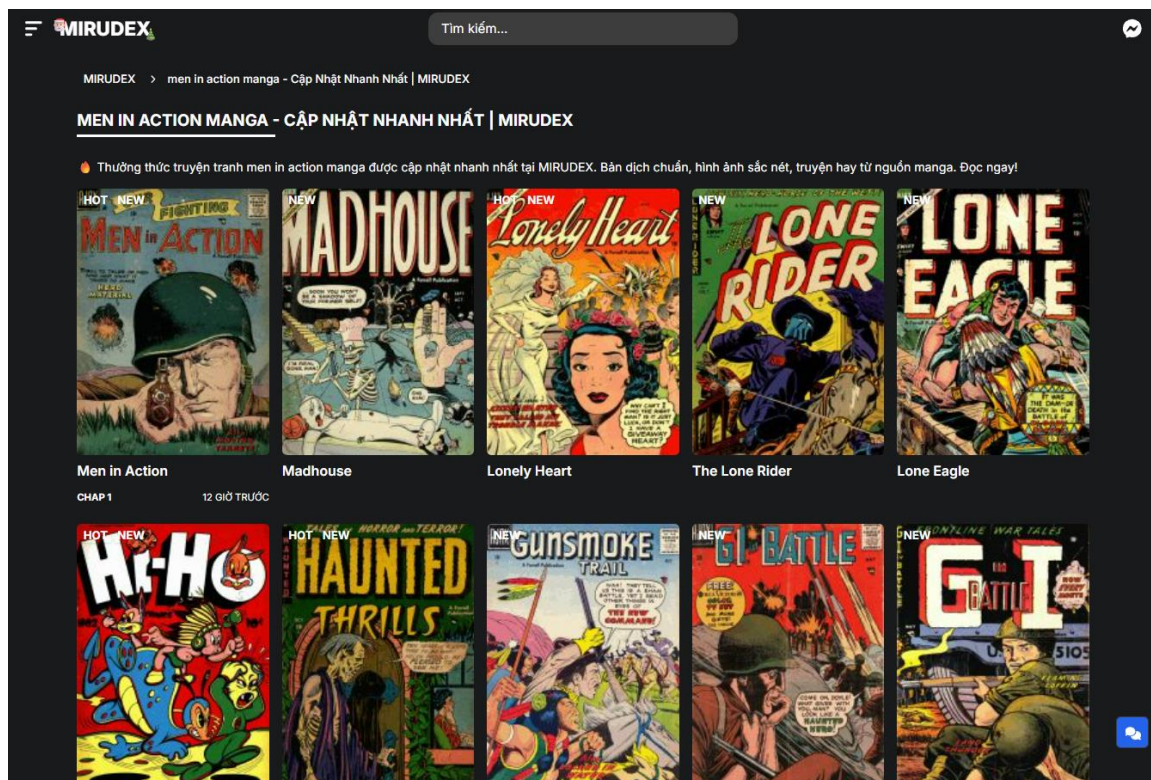
Hình 2.78 Màn hình tìm kiếm cơ bản



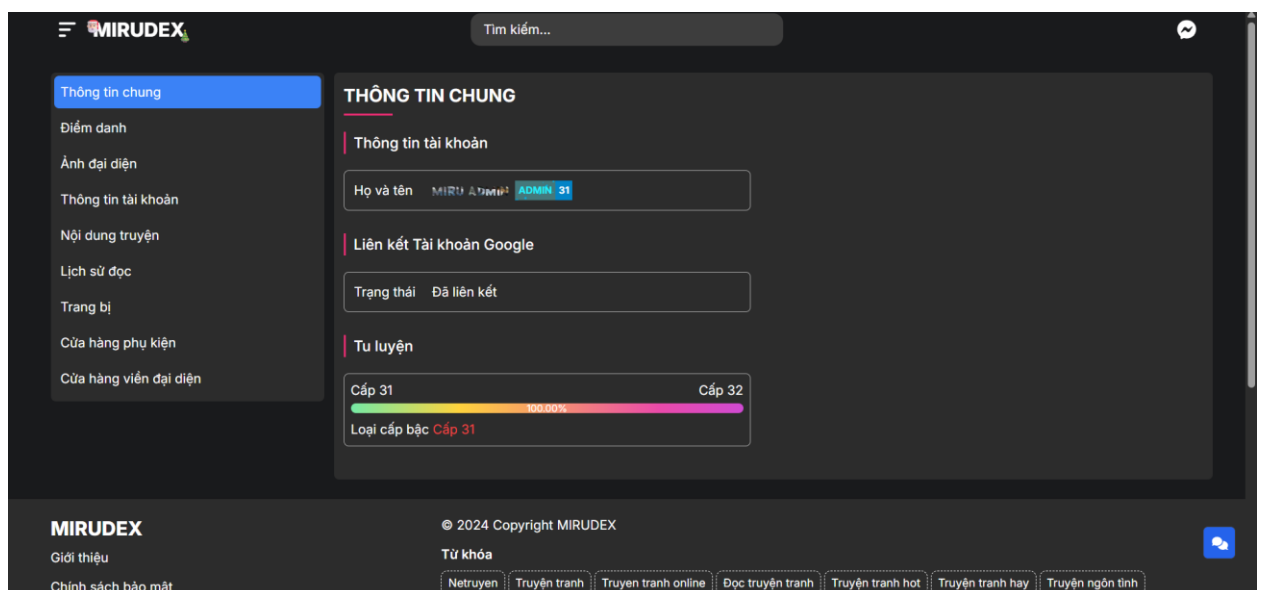
Hình 2.79 Màn hình tìm kiếm nâng cao



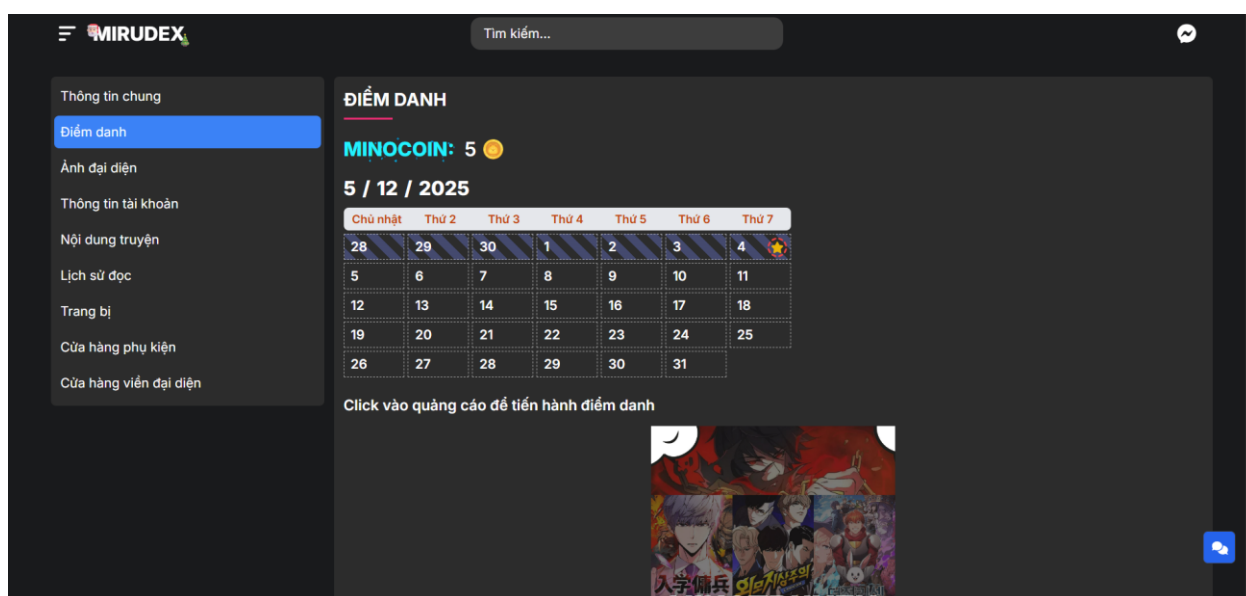
Hình 2.80 Màn hình chi tiết thể loại



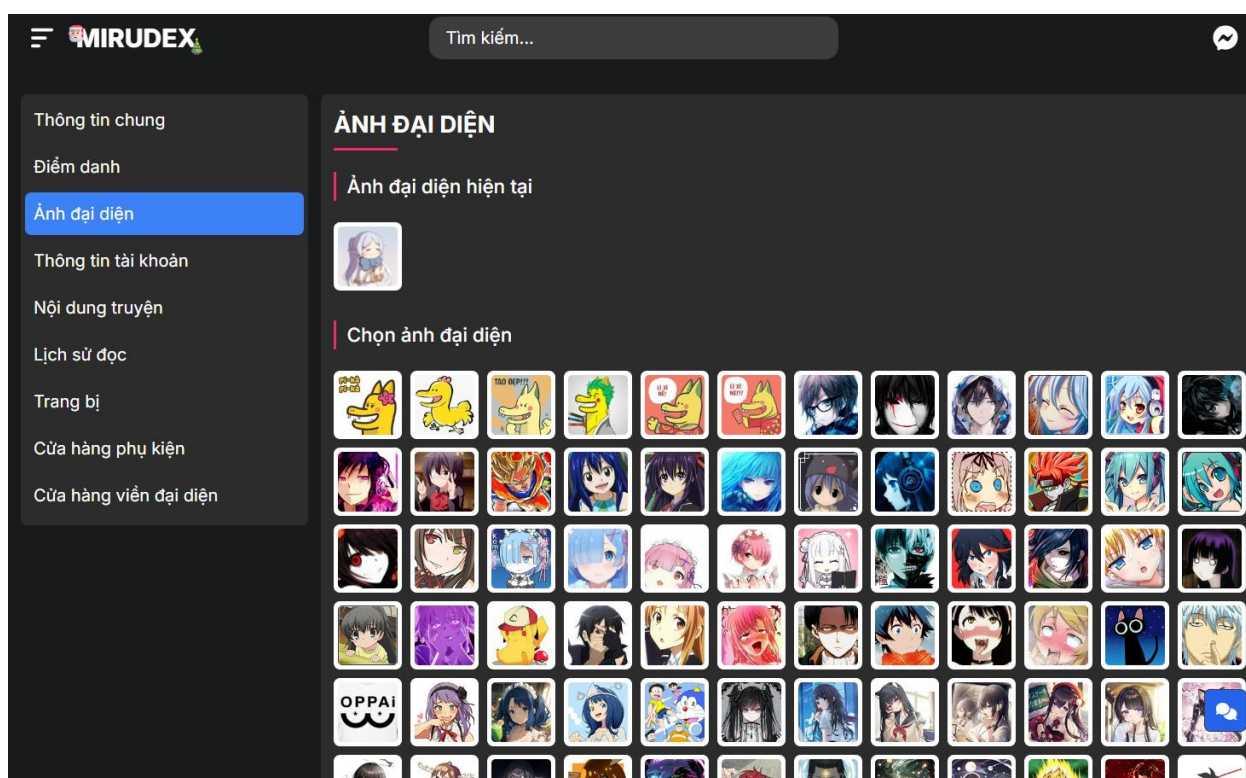
Hình 2.81 Màn hình danh sách truyện cho SEO từ khóa



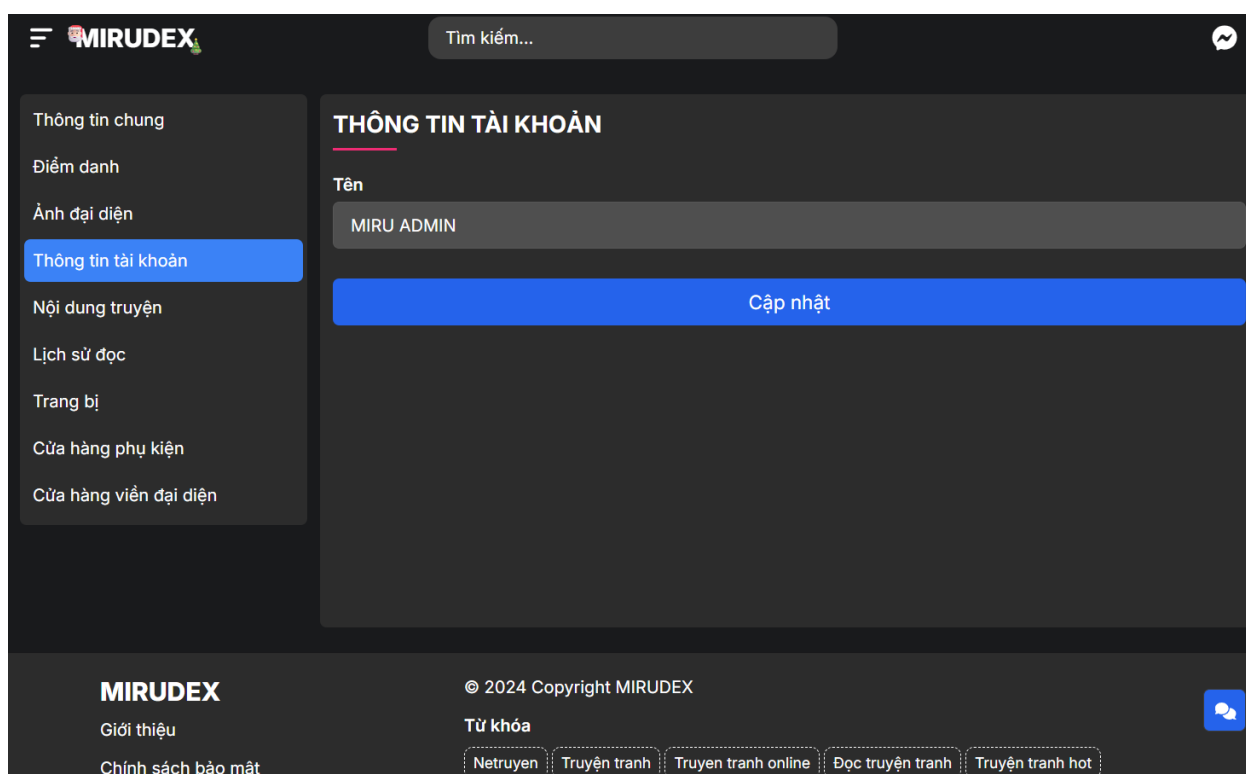
Hình 2.82 Màn hình thông tin người dùng



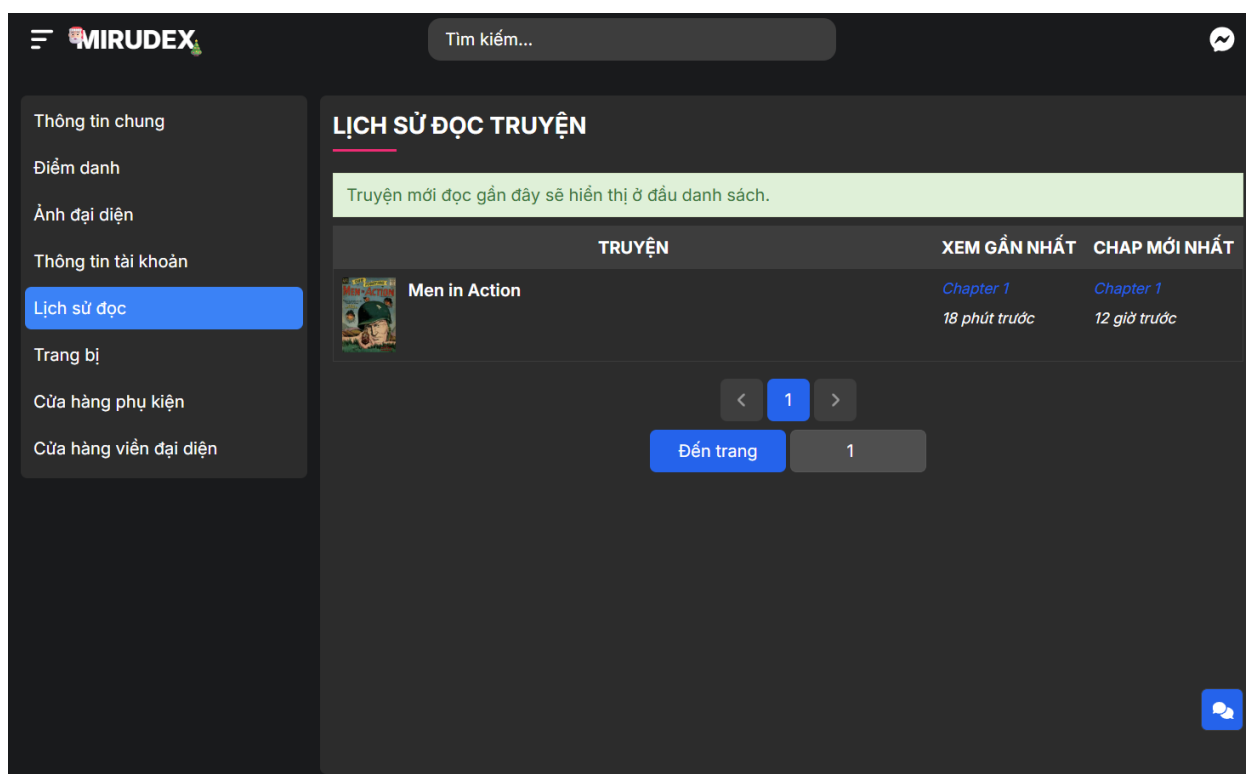
Hình 2.83 Màn hình điểm danh



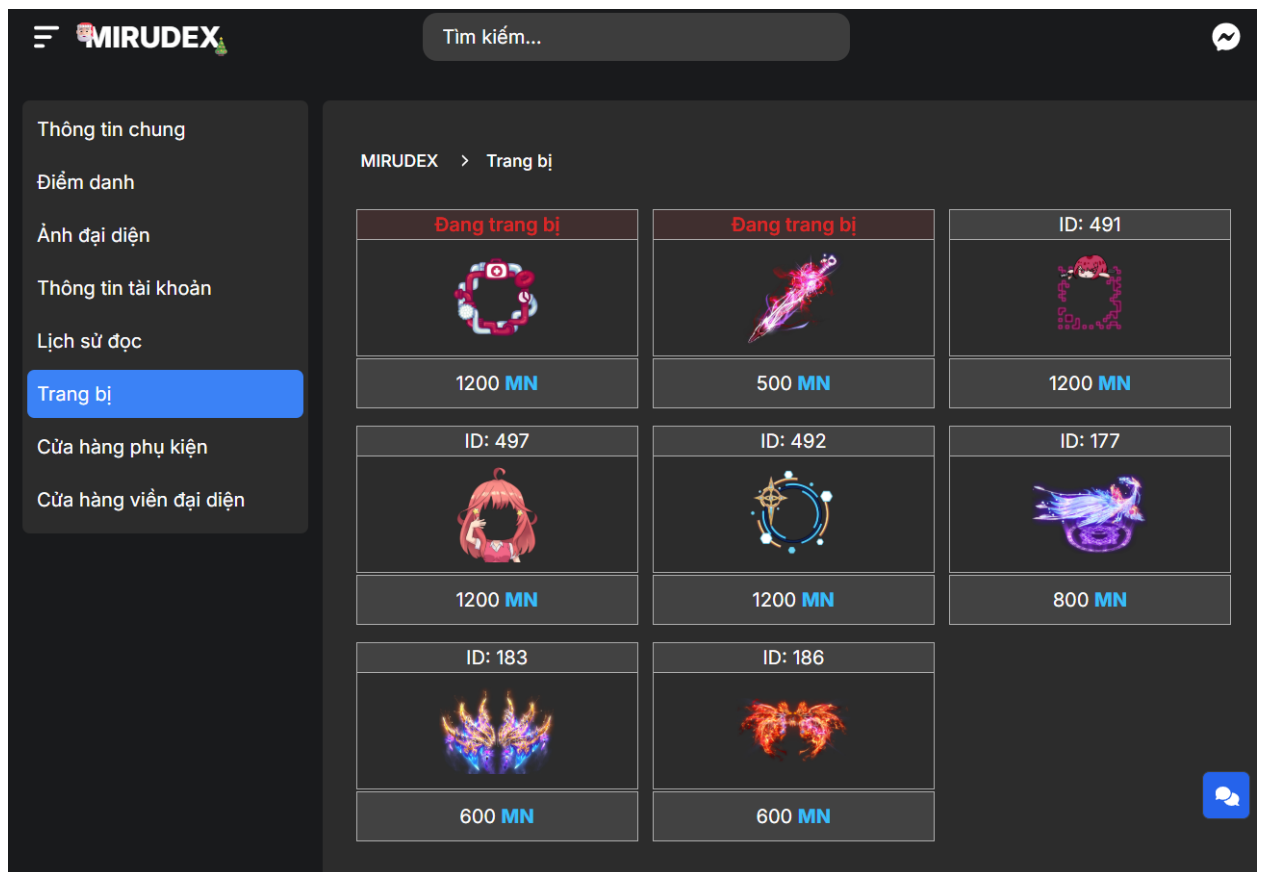
Hình 2.84 Màn hình đổi ảnh đại diện



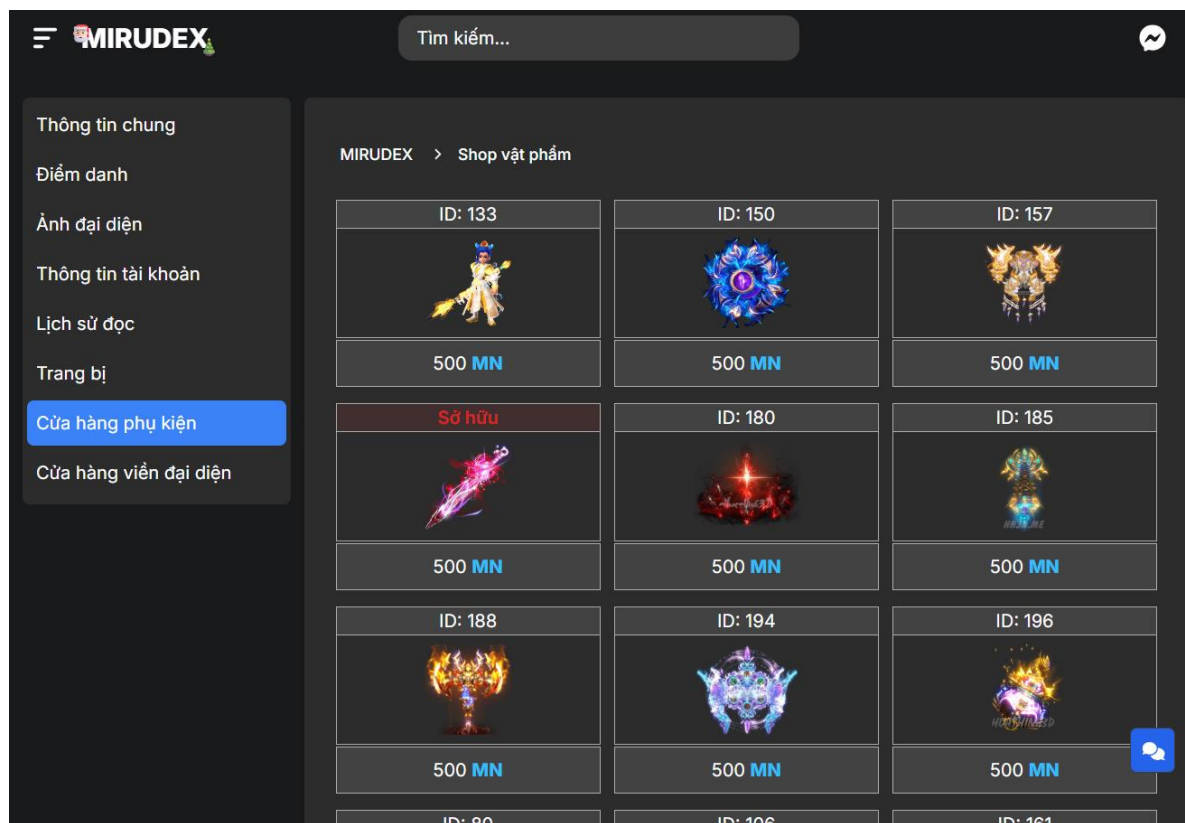
Hình 2.85 Màn hình thông tin tài khoản



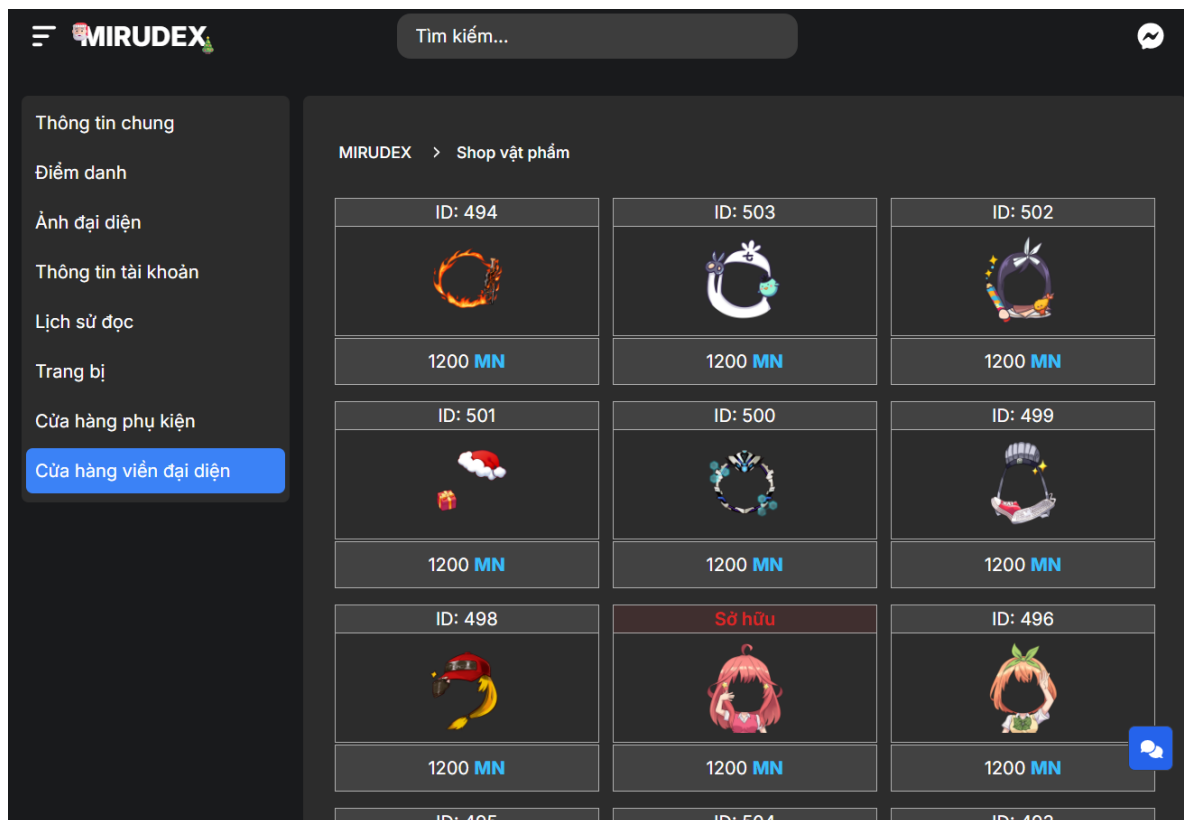
Hình 2.86 Màn hình lịch sử đọc truyện



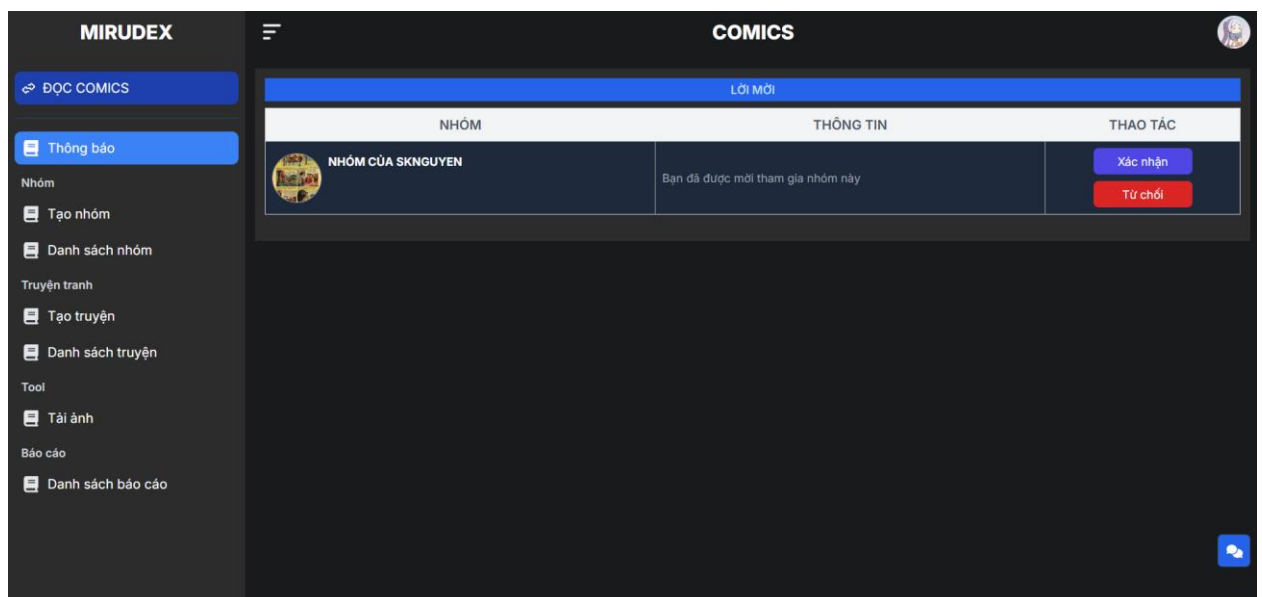
Hình 2.87 Màn hình trang bị vật phẩm



Hình 2.88 Màn hình cửa hàng phụ kiện



Hình 2.89 Màn hình cửa hàng viên ảnh đại diện



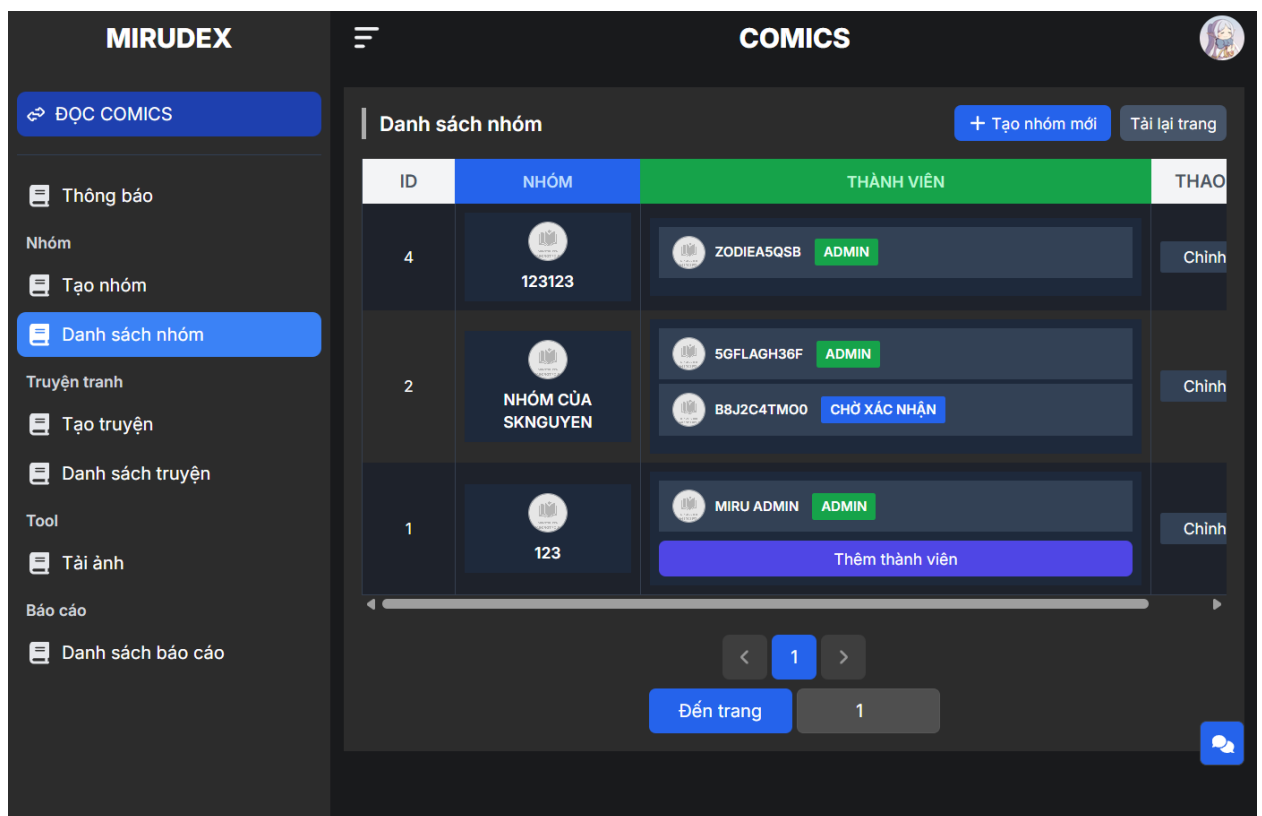
Hình 2.90 Thông báo mời làm thành viên nhóm dịch

The screenshot shows the 'COMICS' section of the 'MIRUDEX' application. The left sidebar contains a menu with options: 'ĐỌC COMICS', 'Thông báo', 'Nhóm' (with sub-options 'Tạo nhóm', 'Danh sách nhóm'), 'Truyện tranh' (with sub-options 'Tạo truyện', 'Danh sách truyện'), 'Tool' (with sub-options 'Tải ảnh'), 'Báo cáo' (with sub-option 'Danh sách báo cáo'), and a chat icon. The main content area is titled 'COMICS' and features a 'Tạo nhóm' (Create Group) form. The form includes fields for 'Tiêu đề' (Title) with a placeholder 'Nhập tên nhóm', 'Facebook của nhóm' (Group Facebook) with a placeholder 'https://facebook.com/' and a 'Đường dẫn Facebook' (Facebook Link) field, and a 'Mô tả' (Description) field. A blue 'Tạo nhóm' button is at the bottom of the form.

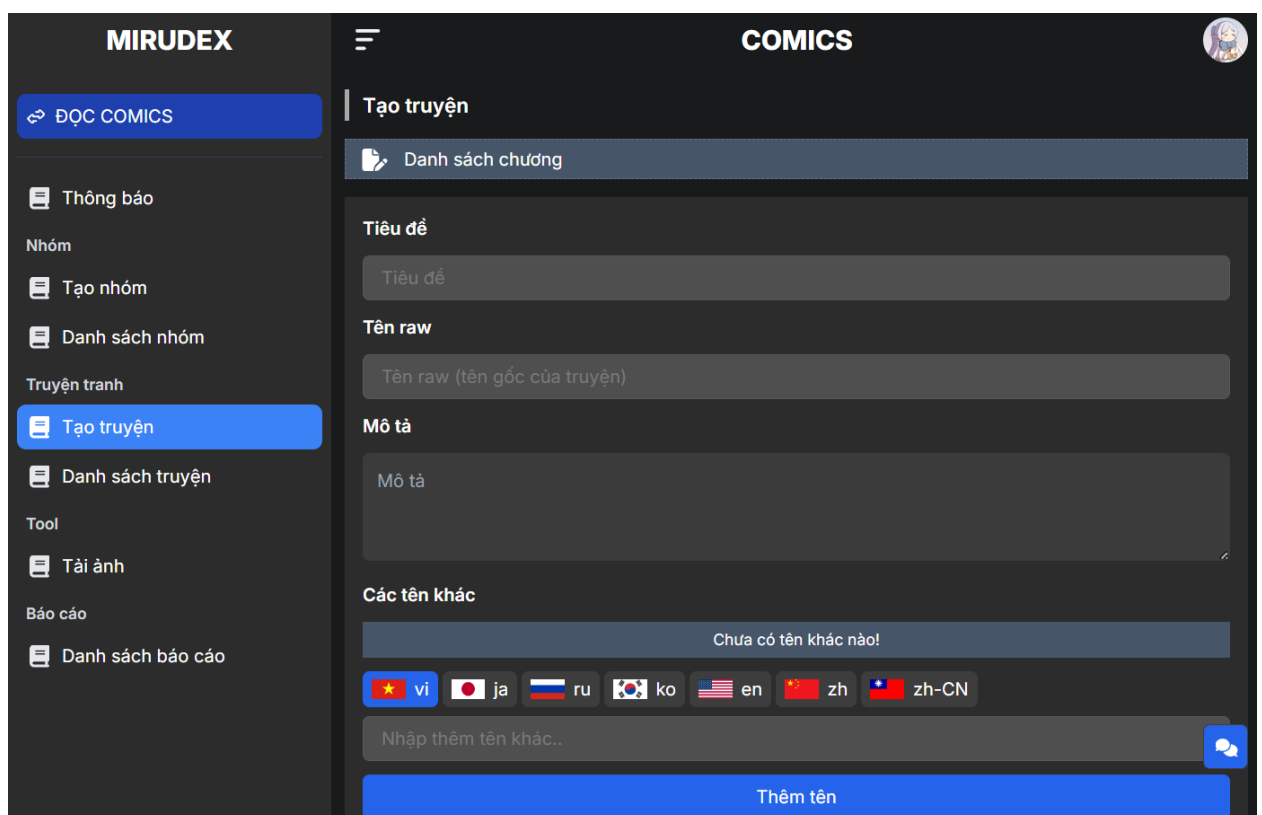
Hình 2.91 Màn hình tạo nhóm dịch truyện

The screenshot shows the 'COMICS' section of the 'MIRUDEX' application, specifically the 'Cập nhật nhóm' (Update Group) form. The left sidebar is identical to the previous screenshot. The main content area is titled 'COMICS' and features a 'Cập nhật nhóm' form. The form includes a 'Cập nhật ảnh đại diện' (Update Profile Picture) section with a circular image placeholder and a 'Cập nhật ảnh đại diện' button. Below this are fields for 'Tiêu đề' (Title) with the value '123123', 'Facebook của nhóm' (Group Facebook) with the value 'https://facebook.com/ 123123', and a 'Mô tả' (Description) field with the value '435345345'. A blue chat icon is visible in the bottom right corner.

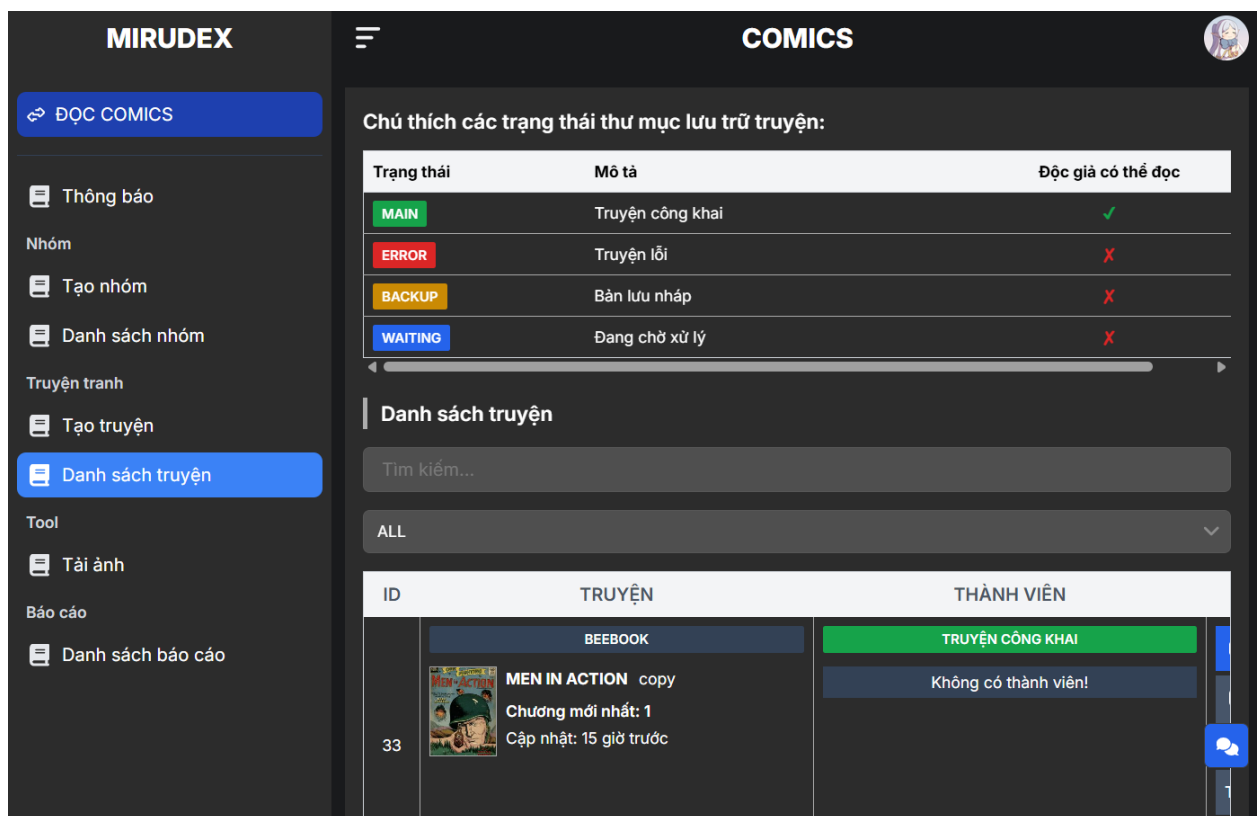
Hình 2.92 Màn hình cập nhật thông tin nhóm dịch



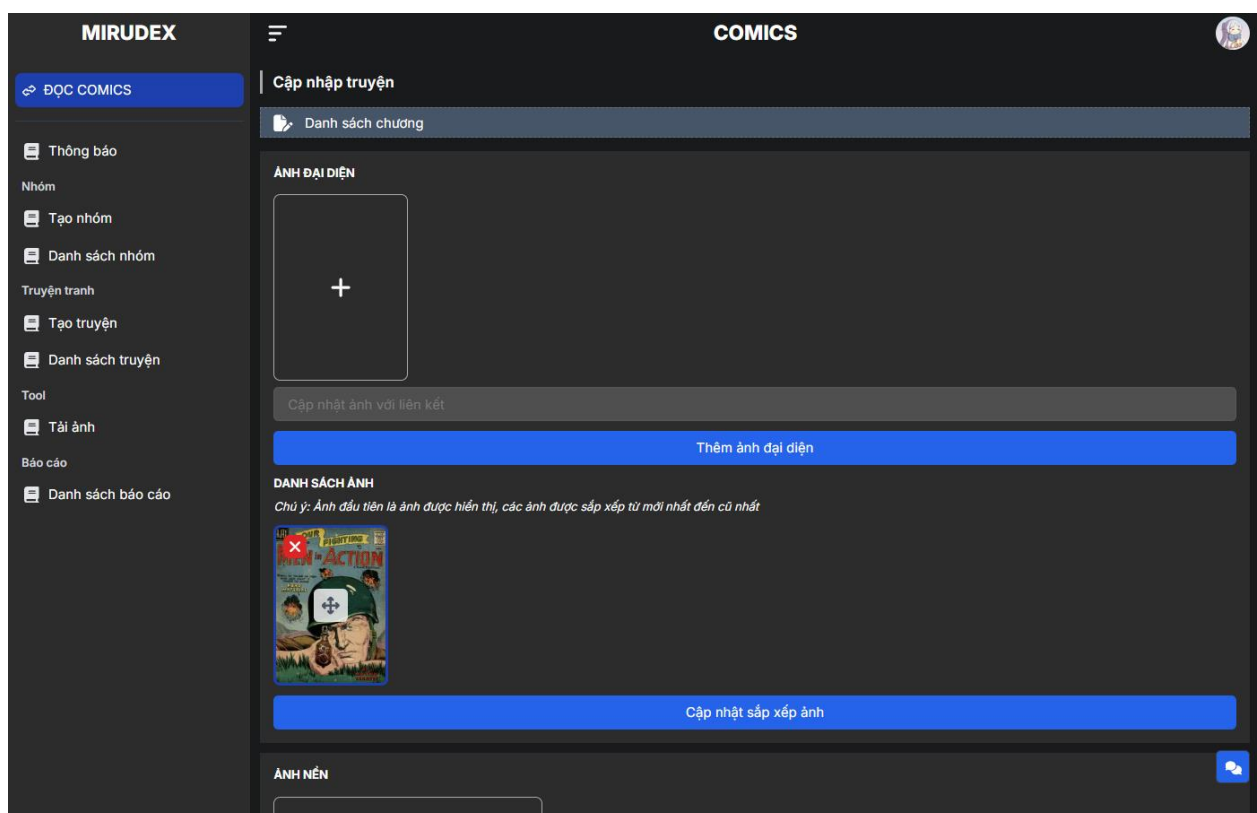
Hình 2.93 Màn hình danh sách nhóm dịch



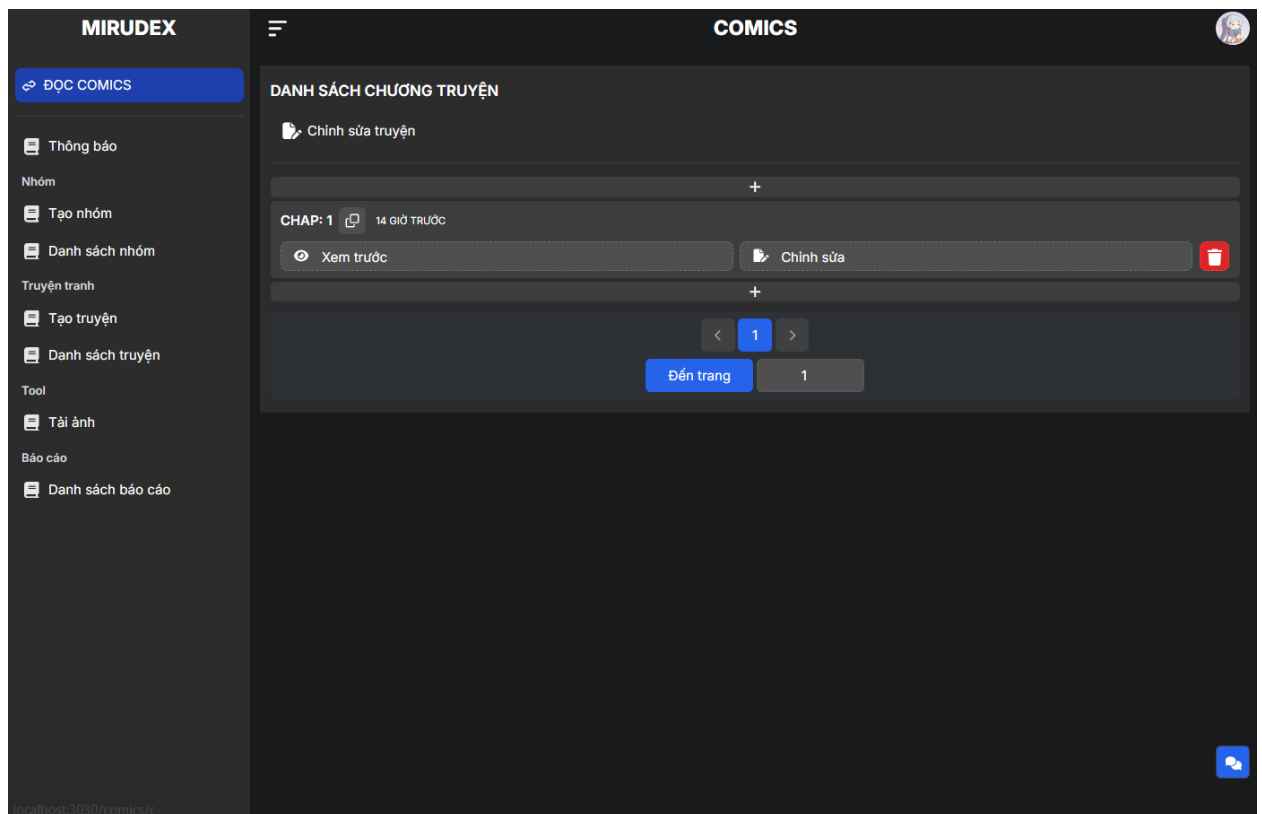
Hình 2.94 Màn hình tạo truyện



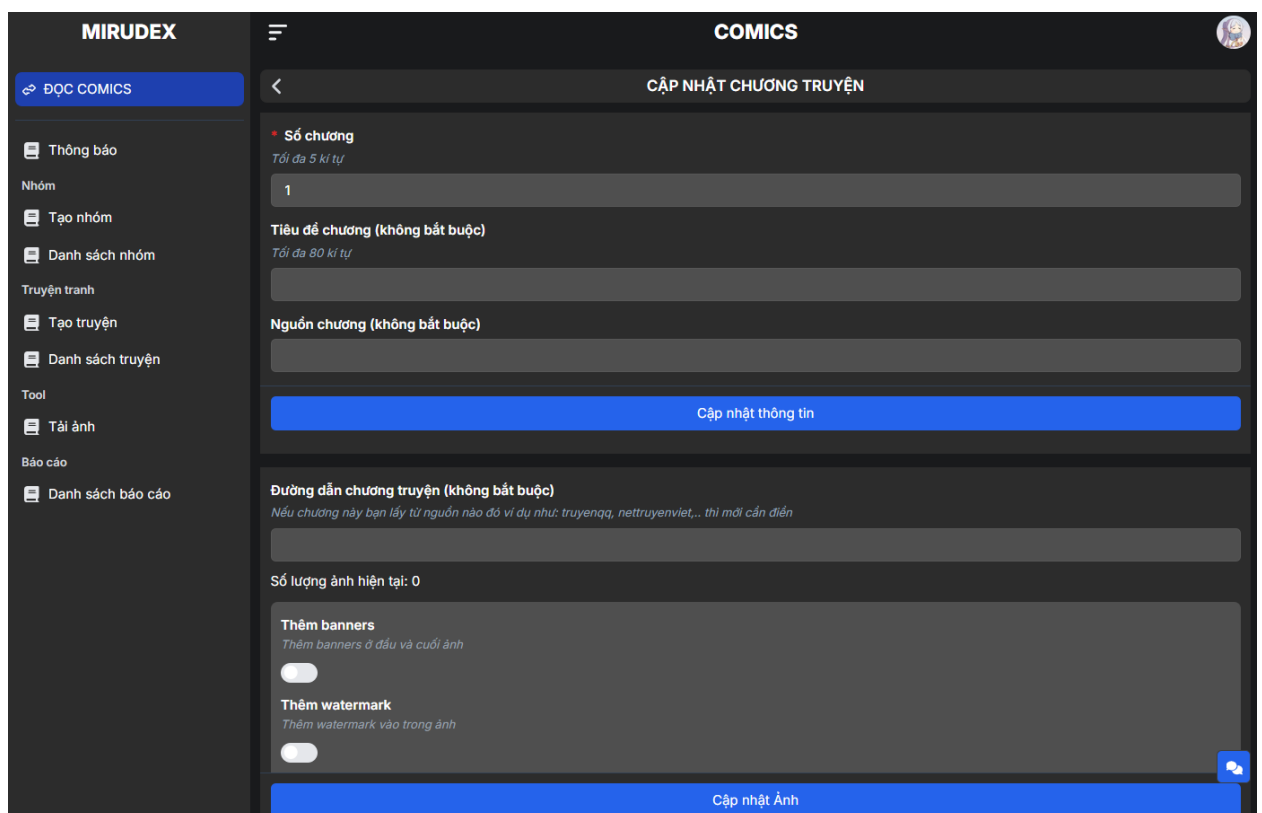
Hình 2.95 Màn hình danh sách truyện



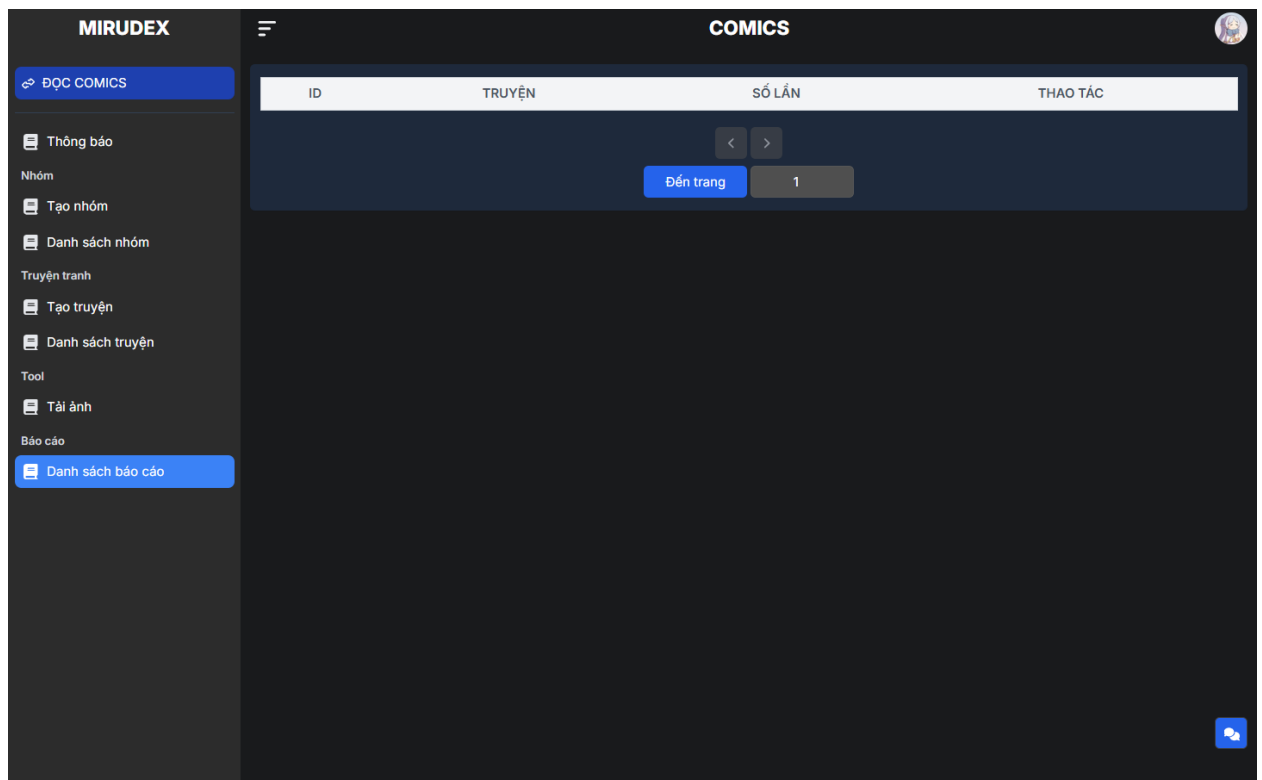
Hình 2.96 Màn hình chỉnh sửa truyện



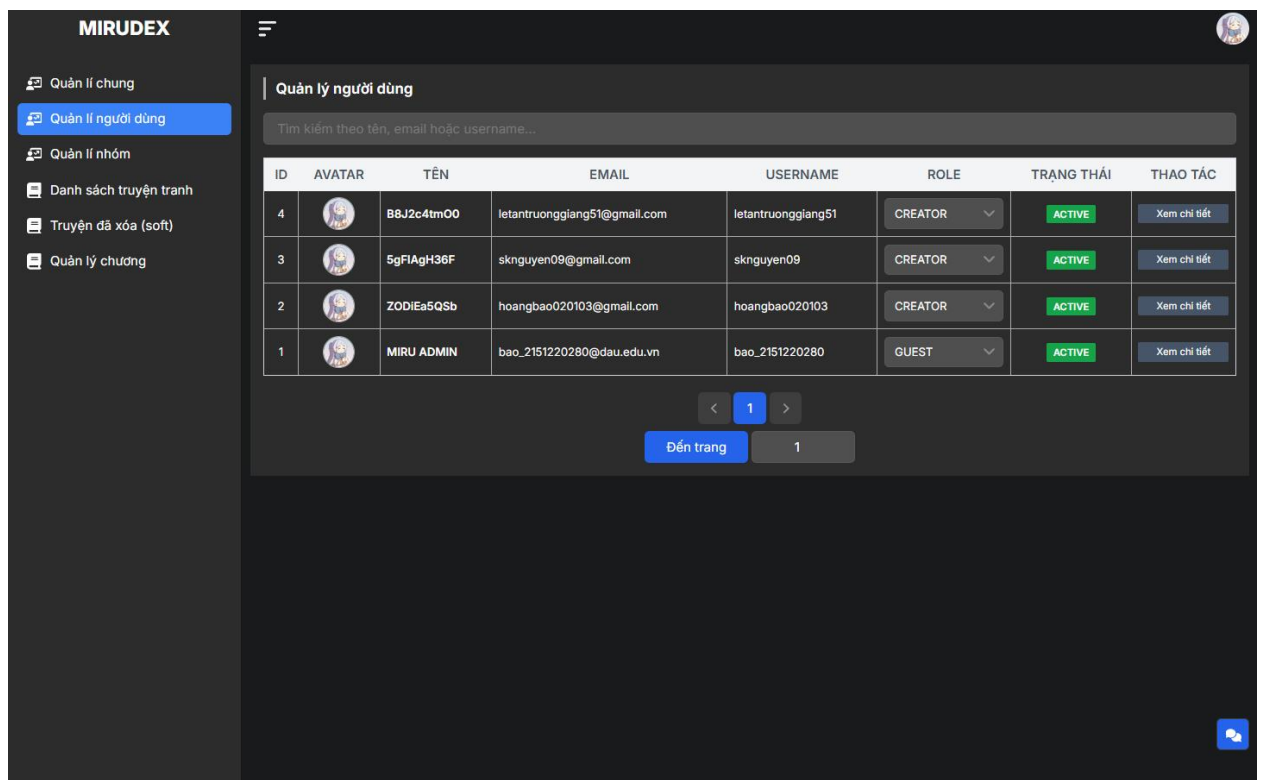
Hình 2.97 Màn hình hiển thị danh sách chương



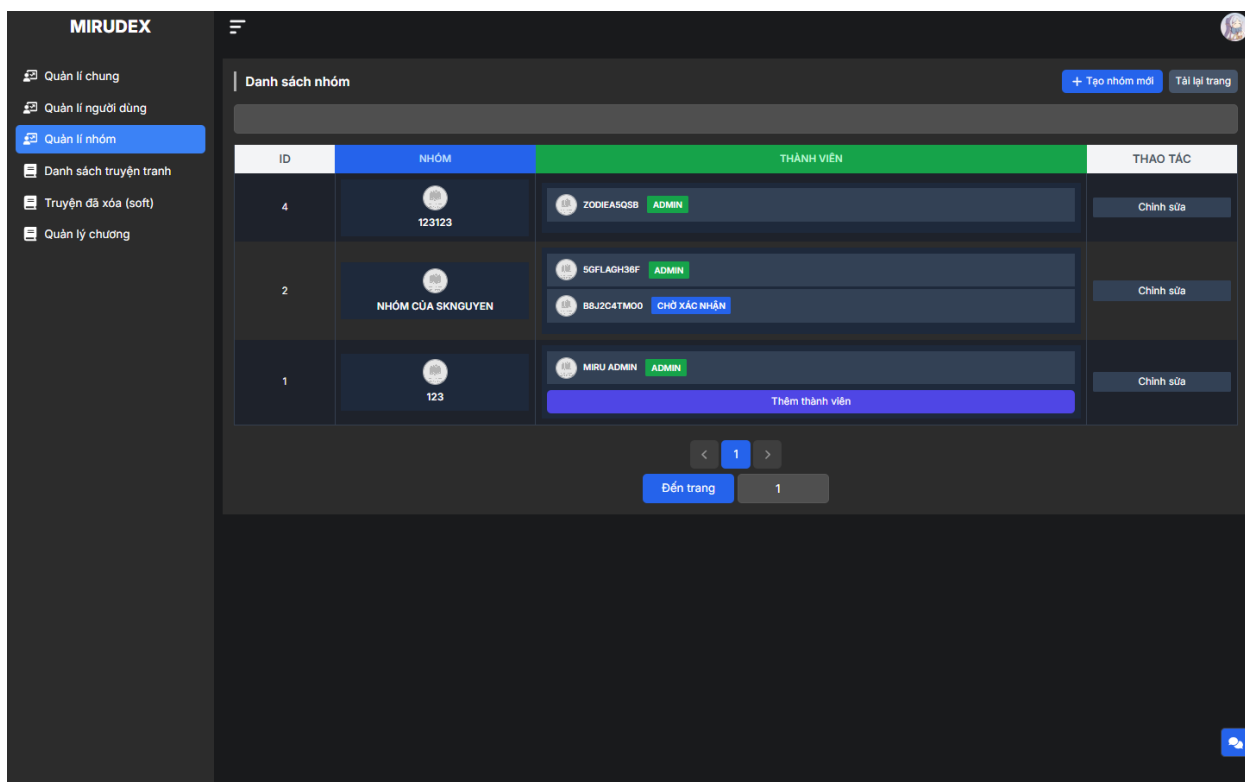
Hình 2.98 Màn hình tạo chương



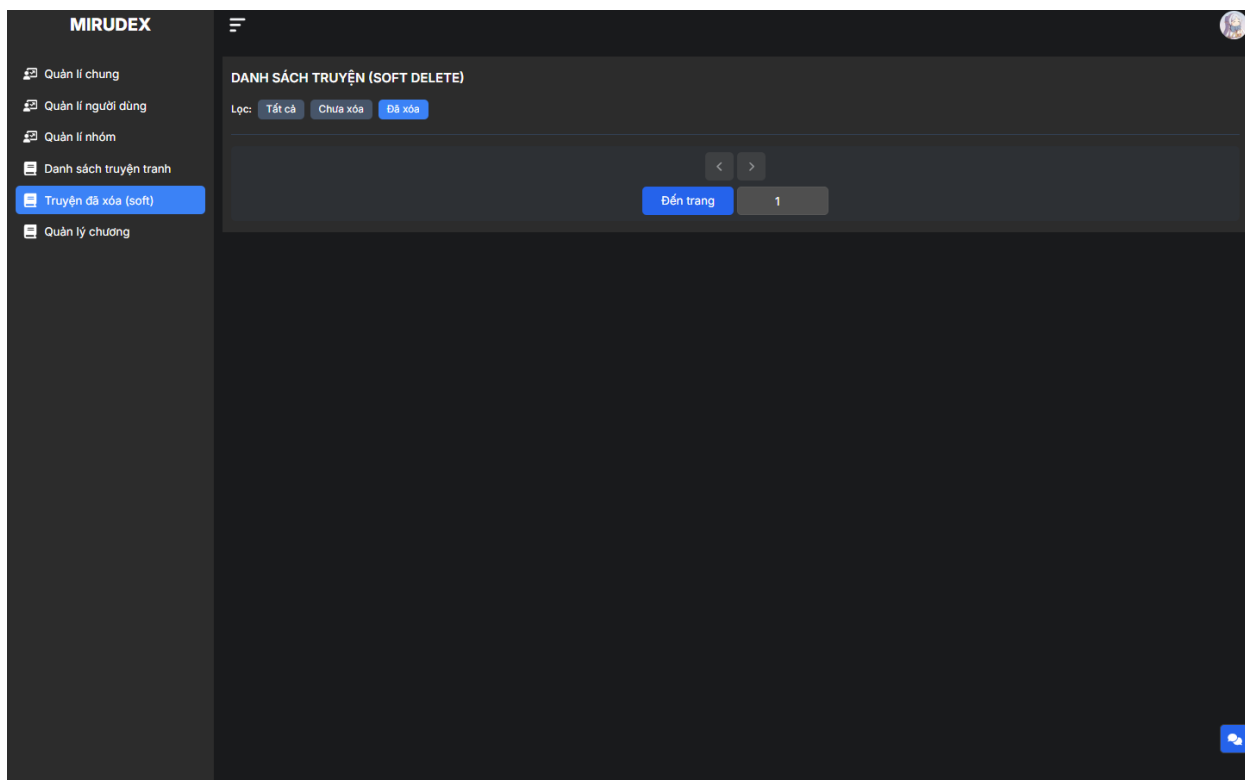
Hình 2.99 Màn hình hiển thị báo cáo chương truyện lỗi



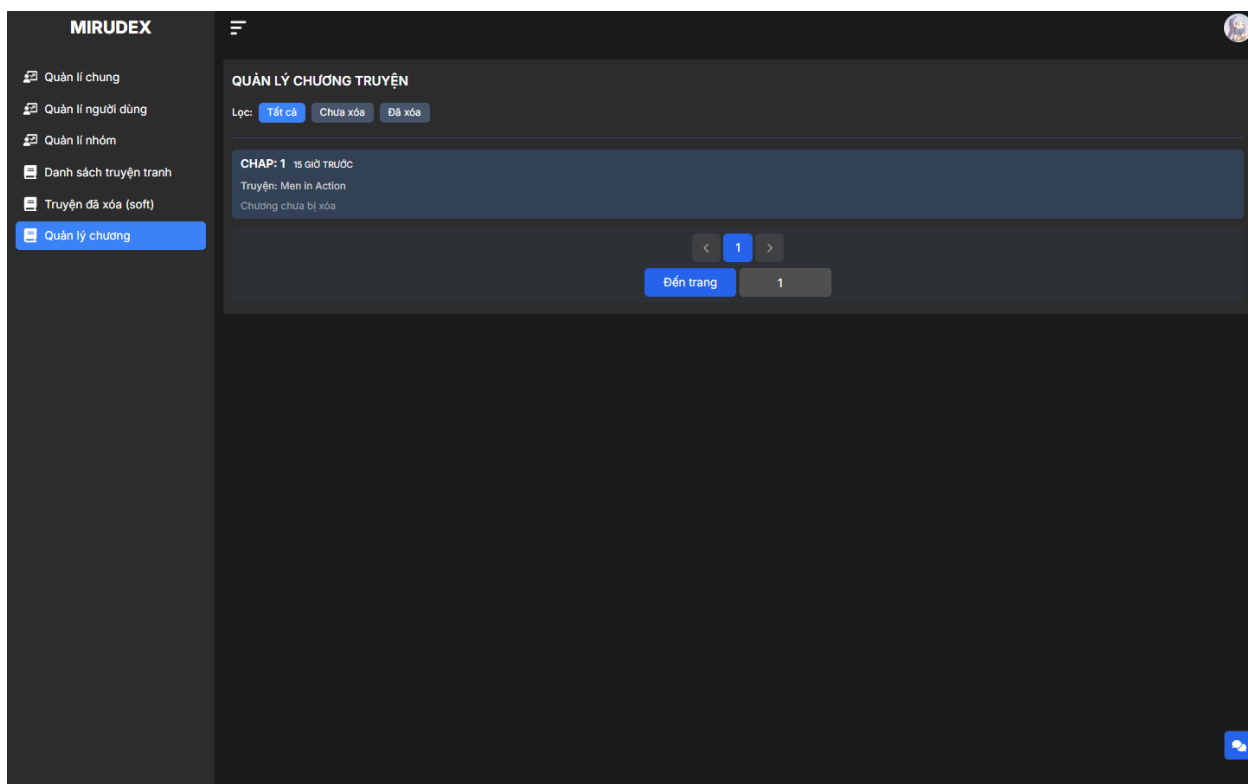
Hình 2.100 Màn hình quản lý người dùng



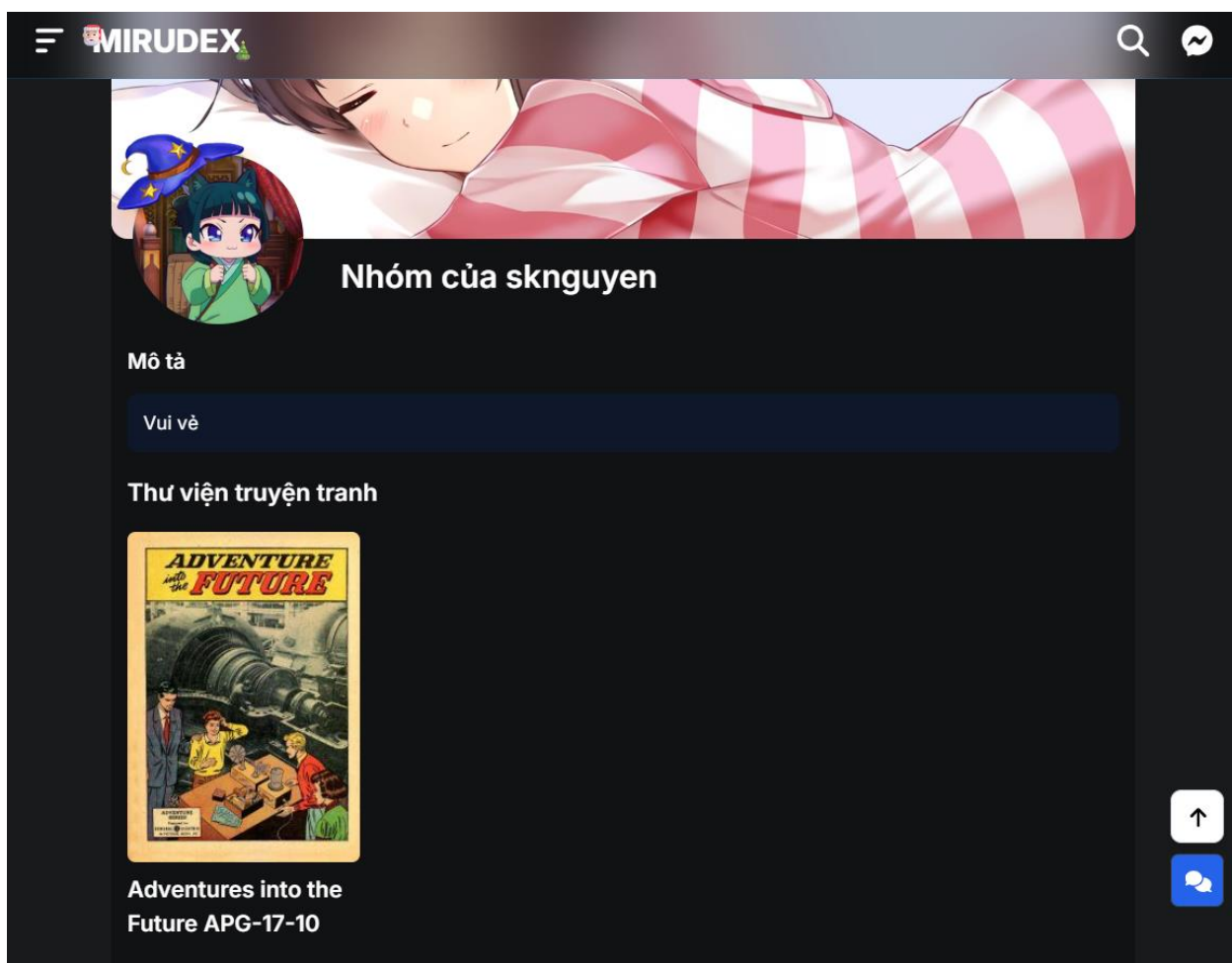
Hình 2.101 Màn hình quản lý nhóm



Hình 2.102 Quản lý truyện đã xóa



Hình 2.103 Quản lý chương đã xóa



Hình 2.104 Màn hình chi tiết nhóm dịch

CHƯƠNG 3: TRIỂN KHAI HỆ THỐNG

3.1. Framework và các thư viện của các services

Bảng 2.58 Framework và các thư viện của các services

Service	Loại	Thư viện / Công nghệ sử dụng
Client Service	FrontEnd	React.js, Next.js, Webpack, Babel, Chart.js, Ant Design, Bootstrap, SyncFusion, React Beautiful DnD, HTML React Parser, Redux Toolkit, React Redux, Tailwind CSS, Framer Motion, React Quill, Socket.io Client, React Chart.js 2, React SortableJS, React Virtualized, Swiper, Next Auth, Axios, Sass, Styled Components
Auth Service	BackEnd	NestJS, Express.js, CORS, NATS, JSON Web Token, Cookie Session, Bcrypt / Argon2, Express Validation / Class Validator, Mongoose / Prisma, NodeMailer, Jest, MongoDB Memory Server, Nodemon, Supertest, Passport, Passport JWT, Passport Google OAuth
Comment Service	BackEnd	NestJS, Express.js, CORS, NATS, JSON Web Token, Cookie Session, Prisma
Project Management Service	BackEnd	NestJS, Express.js, CORS, NATS, JSON Web Token, Cookie Session, Prisma, Nodemon
Issue-Process Service	BackEnd	NestJS, Express.js, CORS, NATS, JSON Web Token, Cookie Session, Prisma, Nodemon, Random Color
Category Service	BackEnd	NestJS, Express.js, CORS, NATS, JSON Web Token, Cookie Session, Prisma
Issue Service	BackEnd	NestJS, Express.js, CORS, NATS, JSON Web Token, Cookie Session, Prisma
Sprint Service	BackEnd	NestJS, Express.js, CORS, NATS, JSON Web Token, Cookie Session, Prisma

File Service	BackEnd	NestJS, Express.js, CORS, JSON Web Token, Cookie Session, Prisma, AWS SDK S3, Sharp, Canvas
Issue-History Service	BackEnd	NestJS, Express.js, CORS, NATS, JSON Web Token, Cookie Session, Prisma
Notification Service	BackEnd	NestJS, Express.js, CORS, JSON Web Token, Cookie Session, Prisma, Socket.io
AI Translation Service	BackEnd	Flask, Flask-CORS, OpenCV, PyTorch, Torchvision, EasyOCR, PaddleOCR, Ultralytics (YOLO), Transformers, OpenAI, Google Generative AI, Anthropic, Deep Translator, Azure AI Vision, Stanza, Requests, NumPy, Pillow

4.2. Cấu trúc thư mục code

4.2.1. Cấu trúc thư mục code Front End

miru-fe/

— node_modules/	# Thư viện bên thứ ba được cài qua npm
— public/	# Tài nguyên tĩnh (hình ảnh, fonts, icons)
— static/	
— images/	# Hình ảnh tĩnh
— fonts/	# Font chữ tùy chỉnh
— icons/	# Icon SVG
— favicon.ico	
— src/	
— app/	# App Router (Next.js 13+)
— (auth)/	# Group layout cho auth pages
— (admin)/	# Group layout cho admin pages
— creator/	# Creator pages
— layout.tsx	# Root layout
— page.tsx	# Home page
— components/	# React components tái sử dụng
— Layouts/	# Layout components (CreatorLayout, AdminLayout)

		— Modules/	# UI modules (Icons, Buttons, Cards)
		— Partial/	# Partial components (Header, Footer, Nav)
		— Common/	# Reusable common components
		— config/	# Tập cấu hình
		— Env.ts	# Environment variables
		— api.config.ts	# API endpoints config
		— constants.ts	# App constants
		— common/	# Dữ liệu chung, types
		— data.types.ts	# Type definitions
		— constants.ts	# Common constants
		— hooks/	# Custom React hooks
		— useAuth.ts	
		— useFetch.ts	
		— services/	# API services, business logic
		— api/	# API calls
		— auth.service.ts	
		— book.service.ts	
		— helpers/	# Helper functions
		— store/	# State management (Redux/Zustand)
		— slices/	
		— store.ts	
		— styles/	# Global styles
		— globals.css	
		— variables.css	
		— utils/	# Utility functions
		— classNames.ts	
		— formatters.ts	
		— validators.ts	
		— middleware.ts	# Next.js middleware
		— .env.local	# Biến môi trường (không commit)
		— .env.example	# Mẫu file .env

- ├── .gitignore
- ├── .eslintrc.json # ESLint config
- ├── .prettierrc # Prettier config
- ├── next.config.js # Next.js config
- ├── package.json
- ├── package-lock.json
- ├── tsconfig.json # TypeScript config
- └── README.md

4.1.2. Cấu trúc thư mục code Back End

miro-be/

- ├── src/
 - | ├── common/ # Các DTO và type dùng chung toàn ứng dụng
 - | | ├── dto/
 - | | | └── page-options.dto.ts
 - | | | └── data.types.ts
 - | |
 - | ├── config/ # Cấu hình ứng dụng
 - | | ├── database.ts
 - | | ├── environment.ts
 - | | └── constants.ts
 - | |
 - | ├── constants/ # Enum, Interface và dữ liệu hằng số
 - | | ├── cloud.enum.ts
 - | | ├── data.ts
 - | | ├── error-create-chapter.enum.ts
 - | | ├── strength-mapping.enum.ts
 - | | ├── token-payload.interface.ts
 - | | ├── user-book-contributor.enum.ts
 - | | ├── user-role.enum.ts
 - | | └── user-team.enum.ts

```
| |  
| |— decorators/                # Custom Decorators cho NestJS  
| | |— auth-user.decorator.ts  
| | |— book-contributor.decorator.ts  
| | |— http.decorator.ts  
| |  
| |— modules/                  # Feature modules (theo domain)  
| | |— creator/  
| | | |— dto/  
| | | | |— book.dto.ts  
| | | | |— book-contributor.dto.ts  
| | | | |— chapter.dto.ts  
| | | | |— cloud.dto.ts  
| | | | |— comment.dto.ts  
| | | | |— create-eposide.dto.ts  
| | | | |— create-video.dto.ts  
| | | | |— task.dto.ts  
| | | | |— team.dto.ts  
| | | | |— user.dto.ts  
| | | |— creator.controller.ts  
| | | |— creator.service.ts  
| | | |— creator.module.ts  
| | |  
| | |— book/                  # Module quản lý sách  
| | | |— book.service.ts  
| | | |— book.module.ts  
| | |  
| | |— chapter/              # Module quản lý chapter  
| | | |— chapter.service.ts  
| | | |— chapter.module.ts  
| | |
```

```
| | | └─ cloud/ # Module xử lý Cloud Storage
| | |   └─ cloud.service.ts
| | |   └─ cloud.module.ts
| | |
| | | └─ cache-data/ # Module quản lý cache
| | |   └─ cache-data.service.ts
| | |   └─ cache-data.module.ts
| | |
| | | └─ video/ # Module quản lý video/anime
| | |   └─ video.service.ts
| | |   └─ video.types.ts
| | |   └─ video.module.ts
| | |
| | | └─ auth/ # Module xác thực & phân quyền
| | |   └─ auth.service.ts
| | |   └─ auth.module.ts
| | |
| | | └─ shared/ # Dịch vụ & middleware dùng chung
| | |   └─ services/
| | |     └─ api-config.service.ts
| | |     └─ telegram.service.ts
| | |
| | |   └─ middleware/
| | |     └─ auth.middleware.ts
| | |     └─ error-handler.middleware.ts
| | |
| | | └─ utils/ # Hàm tiện ích & helpers
| | |   └─ mylog.ts
| | |   └─ chunkArray.ts
| | |   └─ signedUrl.ts
| | |   └─ dataEncryptionUtil.ts
```

```
| | └─ getBaseUrlCloudImage.ts
| |
| └─ prisma/                                # Prisma ORM
|   └─ schema.prisma
|   └─ migrations/
|     └─ [migration files]
|
| └─ server.ts
└─ public/                                # Tập tĩnh phục vụ public
    └─ ads/
        └─ chapter/
            └─ bottom/
                └─ index.html
            └─ file-640-200-.blpha.js
└─ .babelrc
└─ .env
└─ .env.example
└─ .gitignore
└─ .sequelizerc
└─ package.json
└─ package-lock.json
└─ docker-compose.yml
└─ Dockerfile
└─ docs.md
└─ README.md
```

3.2. Xử lý tạo chương kết hợp AI dịch truyện

3.2.1. Vấn đề

Trong quá trình phát triển hệ thống dịch truyện tranh tự động, việc nhận dạng và trích xuất văn bản từ ảnh truyện tranh gặp nhiều thách thức đặc thù mà các giải pháp OCR thương mại hoặc open-source phổ biến không đáp ứng đầy đủ.

Đặc thù của truyện tranh: Truyện tranh có đặc điểm khác biệt so với tài liệu văn bản thông thường. Văn bản trong truyện tranh thường được đặt trong các speech bubble với hình dạng đa dạng, có thể là hình tròn, elip, hoặc các hình dạng tự do. Văn bản cũng có thể xuất hiện dưới dạng free text, không nằm trong bubble, với các góc nghiêng, độ cong, và kích thước font thay đổi. Ngoài ra, văn bản có thể được đặt trên nền phức tạp với nhiều chi tiết hình ảnh, làm cho việc phân tách văn bản và nền trở nên khó khăn. Các giải pháp OCR thông thường được thiết kế cho tài liệu scan hoặc ảnh văn bản có cấu trúc rõ ràng, không phù hợp với đặc thù này.

Hạn chế của OCR services có sẵn: Các dịch vụ OCR thương mại như Google Cloud Vision API, Azure Computer Vision, hoặc AWS Textract được tối ưu cho tài liệu văn bản, không chuyên biệt cho truyện tranh. Các dịch vụ này thường không phân biệt được speech bubble và free text, dẫn đến việc nhận dạng sai hoặc bỏ sót văn bản. Chi phí sử dụng các dịch vụ này cũng cao khi xử lý số lượng lớn ảnh truyện tranh. Các giải pháp OCR open-source như Tesseract OCR cũng gặp khó khăn tương tự, không được train trên dữ liệu truyện tranh nên độ chính xác thấp.

Yêu cầu về độ chính xác và hiệu năng: Để dịch truyện tranh tự động, hệ thống cần phát hiện chính xác vị trí của tất cả văn bản, phân biệt được speech bubble và free text, và nhận dạng chính xác nội dung văn bản. Bất kỳ lỗi nào trong quá trình này sẽ dẫn đến dịch sai hoặc thiếu sót nội dung. Ngoài ra, hệ thống cần xử lý nhanh để đáp ứng nhu cầu dịch số lượng lớn ảnh trong thời gian ngắn. Các giải pháp có sẵn không đáp ứng được cả hai yêu cầu về độ chính xác và hiệu năng.

Yêu cầu về tính tùy biến và mở rộng: Mỗi loại truyện tranh có phong cách vẽ khác nhau, từ manga Nhật Bản với speech bubble đặc trưng, đến comic phương Tây với phong cách khác biệt. Hệ thống cần có khả năng tùy biến để xử lý các loại truyện tranh khác nhau. Ngoài ra, hệ thống cần có khả năng mở rộng để hỗ trợ thêm các ngôn ngữ mới hoặc cải thiện độ chính xác thông qua việc train lại model. Các dịch vụ OCR có sẵn không cho phép tùy biến sâu hoặc train lại model theo nhu cầu cụ thể.

3.2.2. Cách giải quyết

Để giải quyết các vấn đề trên, hệ thống đã quyết định tự xây dựng một pipeline AI OCR chuyên biệt cho truyện tranh, kết hợp các công nghệ deep learning hiện đại với các kỹ thuật xử lý ảnh tùy biến.

Kiến trúc pipeline tự xây dựng: Hệ thống sử dụng kiến trúc pipeline gồm nhiều bước xử lý tuần tự. Đầu tiên, hệ thống sử dụng YOLO models tự train để phát hiện speech bubble và vùng văn bản. Cụ thể, hệ thống sử dụng ba YOLO models riêng biệt: một model để phát hiện speech bubble, một model để segmentation văn bản, và một model để detection văn bản. Việc sử dụng nhiều models chuyên biệt giúp tăng độ chính xác so với việc sử dụng một model duy nhất. Sau đó, hệ thống kết hợp kết quả từ ba models này để tạo ra danh sách các text blocks với thông tin về vị trí, loại text (text bubble hoặc free text), và vùng cần xóa để inpainting.

Tích hợp OCR engines chuyên biệt: Thay vì chỉ dựa vào một OCR engine, hệ thống tích hợp nhiều OCR engines khác nhau và chọn engine phù hợp dựa trên ngôn ngữ nguồn. Đối với tiếng Anh, hệ thống sử dụng EasyOCR, một công cụ OCR mã nguồn mở được tối ưu cho nhiều ngôn ngữ. Đối với tiếng Trung, hệ thống sử dụng PaddleOCR, một công cụ OCR chuyên biệt cho tiếng Trung với độ chính xác cao. Việc sử dụng nhiều OCR engines giúp hệ thống tận dụng điểm mạnh của từng engine cho từng ngôn ngữ cụ thể.

Xử lý ảnh tùy biến cho truyện tranh: Hệ thống áp dụng các kỹ thuật xử lý ảnh tùy biến để cải thiện độ chính xác OCR. Khi phát hiện văn bản trong speech bubble, hệ thống tự động điều chỉnh tọa độ để mở rộng vùng OCR, đảm bảo không bỏ sót văn bản ở viền bubble. Hệ thống cũng sử dụng adaptive thresholding để xử lý ảnh với độ sáng thay đổi, giúp tách văn bản khỏi nền phức tạp. Ngoài ra, hệ thống sử dụng connected component analysis để phát hiện các vùng văn bản nhỏ và loại bỏ noise, đảm bảo chỉ xử lý các vùng văn bản thực sự.

Khả năng train và cải thiện model: Một trong những lợi ích lớn nhất của việc tự xây dựng hệ thống là khả năng train lại các YOLO models trên dữ liệu truyện tranh cụ thể. Hệ thống có thể thu thập dữ liệu từ các truyện tranh thực tế, label các speech bubble và vùng văn bản, sau đó train lại models để cải thiện độ chính xác. Quá trình này cho phép hệ thống thích ứng với các phong cách truyện tranh khác nhau và cải thiện hiệu suất theo thời gian. Các dịch vụ OCR có sẵn không cho phép train lại model, khiến hệ thống không thể cải thiện độ chính xác cho các trường hợp cụ thể.

Tối ưu hiệu năng và chi phí: Việc không phụ thuộc vào các dịch vụ bên ngoài giúp giảm chi phí vận hành, đặc biệt khi xử lý số lượng lớn ảnh. Ngoài ra, hệ thống có thể tối ưu các bước xử lý để giảm thời gian xử lý, ví dụ như resize ảnh lớn trước khi xử lý, hoặc sử dụng batch processing để xử lý nhiều ảnh cùng lúc.

3.2.3. Module code

```
def _decode_image_from_request() -> tuple[np.ndarray, str, str]:
    """
    Helper function để decode ảnh và lấy tham số từ request.
    Tối ưu: chỉ đọc file một lần.

    Returns:
    | Tuple (image_bgr, source_lng_cd, target_lng_cd)
    """
    if "image" not in request.files:
        raise ValueError("No image file uploaded")

    image_file = request.files["image"]
    source_lng_cd = request.form.get("source_lng_cd", "en")
    target_lng_cd = request.form.get("target_lng_cd", "vi")

    # Tối ưu: đọc file một lần
    file_bytes = image_file.read()
    file_array = np.frombuffer(file_bytes, dtype=np.uint8)
    image_bgr = cv2.imdecode(file_array, cv2.IMREAD_COLOR)

    if image_bgr is None:
        raise ValueError("Invalid image file")

    # Resize nếu ảnh quá lớn để tăng tốc xử lý
    image_bgr, _ = _resize_image_if_large(image_bgr, max_dimension=2048)

    return image_bgr, source_lng_cd, target_lng_cd
```

Hình 2.105 Api bắt xử lý điều chỉnh kích ảnh

```
def _resize_image_if_large(image_bgr: np.ndarray, max_dimension: int = 2048) -> tuple[np.ndarray,
float]:
    """
    Resize ảnh nếu quá lớn để tăng tốc xử lý.

    Args:
    | image_bgr: Ảnh BGR
    | max_dimension: Kích thước tối đa (chiều dài hoặc rộng)

    Returns:
    | Tuple (resized_image, scale_factor)
    """
    h, w = image_bgr.shape[:2]
    max_size = max(h, w)

    if max_size <= max_dimension:
        return image_bgr, 1.0

    scale = max_dimension / max_size
    new_w = int(w * scale)
    new_h = int(h * scale)

    resized = cv2.resize(image_bgr, (new_w, new_h), interpolation=cv2.INTER_AREA)
    return resized, scale
```

Hình 3.106 Điều chỉnh kích thước ảnh nếu quá lớn để tăng tốc xử lý

```
@app.route("/api/translate/images", methods=["POST"])
def upload_image():
    """
    API 1: Gửi ảnh, trả về ảnh đã dịch.

    Request:
    - image: File ảnh
    - source_lng_cd: Mã ngôn ngữ nguồn (mặc định: "en")
    - target_lng_cd: Mã ngôn ngữ đích (mặc định: "vi")

    Returns:
    File ảnh JPEG đã dịch
    """
    try:
        image_bgr, source_lng_cd, target_lng_cd = _decode_image_from_request()
        context = pipeline.process(image_bgr, source_lng_cd, target_lng_cd)
        rendered_bgr = cv2.cvtColor(context.rendered_image, cv2.COLOR_RGB2BGR)

        # Tối ưu JPEG encoding: cân bằng quality và tốc độ
        encode_params = [cv2.IMWRITE_JPEG_QUALITY, 85] # Giảm từ 95 xuống 85 để nhanh hơn
        img_encoded = cv2.imencode(".jpg", rendered_bgr, encode_params)
        return send_file(
            BytesIO(img_encoded.tobytes()),
            mimetype="image/jpeg",
            as_attachment=True,
            download_name="processed_image.jpg",
        )
    except Exception as err:
        return jsonify({"error": str(err)}), 500
```

Hình 3.107 API bắt xử lý dịch ảnh

```
def process(
    self, image_bgr: np.ndarray, source_lng_cd: str, target_lng_cd: str
) -> TranslationContext:
    """Execute the full translation pipeline on a BGR image."""
    source_lang = self.language_codes.get(source_lng_cd, "English")
    target_lang = self.language_codes.get(target_lng_cd, "Vietnamese")

    block_list = self.detector.detect(image_bgr)
    self.ocr.initialize(source_lang, source_lng_cd)
    block_list = self.ocr.process(image_bgr, block_list)

    inpaint_config = self.settings.inpainting.to_config()
    mask = generate_mask(image_bgr, block_list)
    image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
    inpainted = self.inpainter(image_rgb, mask, inpaint_config)
    inpainted = cv2.convertScaleAbs(inpainted)

    block_list = self.translation_service.translate_blocks(
        source_lang, target_lang, block_list
    )

    render_settings = self.settings.render
    format_translations(
        block_list, target_lng_cd, upper_case=render_settings.upper_case
    )
    get_best_render_area(block_list, image_bgr, inpainted)

    for blk in block_list:
        blk.alignment = render_settings.alignment

    rendered = draw_text(
        inpainted,
        block_list,
        render_settings.font_path,
        colour=render_settings.color,
        init_font_size=render_settings.max_font_size,
        min_font_size=render_settings.min_font_size,
        outline=render_settings.outline,
    )

    return TranslationContext(
        rendered_image=rendered,
        blocks=block_list,
        source_lang=source_lang,
        target_lang=target_lang,
        render_font_path=render_settings.font_path,
    )
```

Hình 3.108 Xử lý dịch ảnh

```
@app.route("/api/translate/images/select/text", methods=["POST"])
def upload_image_select_text():
    """
    API: Detect và OCR, trả về tọa độ các text boxes.

    Request:
    - image: File ảnh
    - source_lng_cd: Mã ngôn ngữ nguồn (mặc định: "en")

    Returns:
    JSON chứa:
    - boxes: Danh sách các text box với text_bbox [x1, y1, x2, y2]
    """
    try:
        if "image" not in request.files:
            return jsonify({"error": "No image file uploaded"}), 400

        image_file = request.files["image"]
        source_lng_cd = request.form.get("source_lng_cd", "en")

        # Tối ưu: đọc file một lần
        file_bytes = image_file.read()
        file_array = np.frombuffer(file_bytes, dtype=np.uint8)
        image_bgr = cv2.imdecode(file_array, cv2.IMREAD_COLOR)

        if image_bgr is None:
            return jsonify({"error": "Invalid image file"}), 400

        # Resize nếu ảnh quá lớn
        image_bgr, _ = _resize_image_if_large(image_bgr, max_dimension=2048)

        # Chỉ detect và OCR
        block_list = pipeline.detect_and_ocr(image_bgr, source_lng_cd)

        # Serialize chỉ text_bbox - sử dụng hàm serialize_text_blocks để đảm bảo format giống nhau
        from modules.utils.serialization import _bbox_to_list

        boxes_data = []
        for idx, blk in enumerate(block_list):
            if blk.xyxy is not None:
                # Sử dụng _bbox_to_list để đảm bảo format giống với API detail
                bbox = _bbox_to_list(blk.xyxy)
                if bbox is not None:
                    boxes_data.append({
                        "id": idx,
                        "text_bbox": bbox,
                    })

        return jsonify({"boxes": boxes_data})
    except Exception as err:
        return jsonify({"error": str(err)}), 500
```

Hình 3.109 API xử lý dịch nội dung

```
@app.route("/api/translate/images/detail", methods=["POST"])
def upload_image_detail():
    """
    API 2: Gửi ảnh với selected_boxes, trả về ảnh đã xóa text và thông tin chi tiết.

    Request:
    - image: File ảnh
    - source_lng_cd: Mã ngôn ngữ nguồn (mặc định: "en")
    - target_lng_cd: Mã ngôn ngữ đích (mặc định: "vi")
    - selected_boxes: JSON string danh sách boxes [[x1, y1, x2, y2], ...]

    Returns:
    JSON chứa:
    - inpainted_image: Base64 encoded ảnh đã xóa text
    - blocks: Danh sách các text block với:
        - text: Text nhận dạng được (OCR)
        - translation: Text đã dịch
        - text_bbox: Vị trí text nhận dạng [x1, y1, x2, y2]
        - bubble_bbox: Vị trí bubble (nếu có) [x1, y1, x2, y2]
        - inpaint_bboxes: Danh sách vị trí xóa text [[x1, y1, x2, y2], ...]
        - text_class: Loại text (text_bubble, text-free)
    - source_lang: Tên ngôn ngữ nguồn
    - target_lang: Tên ngôn ngữ đích
    - source_lng_cd: Mã ngôn ngữ nguồn
    - target_lng_cd: Mã ngôn ngữ đích
    """
    import json
    import base64

    try:
        if "image" not in request.files:
            return jsonify({"error": "No image file uploaded"}), 400

        image_file = request.files["image"]
        source_lng_cd = request.form.get("source_lng_cd", "en")
        target_lng_cd = request.form.get("target_lng_cd", "vi")
        selected_boxes_json = request.form.get("selected_boxes", "")

        # Parse selected_boxes
        try:
            selected_boxes = json.loads(selected_boxes_json)
        except json.JSONDecodeError:
            return jsonify({"error": "Invalid selected_boxes format"}), 400

        # Tối ưu: đọc file một lần
        file_bytes = image_file.read()
        file_array = np.frombuffer(file_bytes, dtype=np.uint8)
        image_bgr = cv2.imdecode(file_array, cv2.IMREAD_COLOR)

        if image_bgr is None:
            return jsonify({"error": "Invalid image file"}), 400

        # Resize nếu ảnh quá lớn
        image_bgr, _ = _resize_image_if_large(image_bgr, max_dimension=2048)

        # Xử lý với selected_boxes
        result = pipeline.process_with_selected_boxes(
            image_bgr, source_lng_cd, target_lng_cd, selected_boxes
        )

        # Encode ảnh đã xóa text thành base64 (ảnh đang ở định dạng BGR từ OpenCV)
        _, img_encoded = cv2.imencode('.jpg', result["inpainted_image"])
        img_base64 = base64.b64encode(img_encoded.tobytes()).decode('utf-8')

        blocks_data = serialize_text_blocks(result["blocks"])

        return jsonify({
            "inpainted_image": f"data:image/jpeg;base64,{img_base64}",
            "blocks": blocks_data,
            "source_lang": result["source_lang"],
            "target_lang": result["target_lang"],
            "source_lng_cd": source_lng_cd,
            "target_lng_cd": target_lng_cd,
        })
    except Exception as err:
        return jsonify({"error": str(err)}), 500
```

Hình 3.110 API xử lý dịch chi tiết ảnh

3.3. Xử lý bảo mật hiển thị mã hóa ảnh

3.3.1. Vấn đề

Trong hệ thống Miru, việc bảo vệ nội dung ảnh truyện tranh khỏi sao chép và phân phối trái phép là yêu cầu quan trọng. Tuy nhiên, triển khai mã hóa ảnh gặp các thách thức sau:

- **Yêu cầu giải mã phía client:** Ảnh cần được giải mã trực tiếp trên trình duyệt để hiển thị, không thể giải mã hoàn toàn trên server vì sẽ tạo gánh nặng băng thông và độ trễ lớn khi phải stream toàn bộ ảnh đã giải mã. Hệ quả là logic giải mã phải chạy ở client-side, khiến code dễ bị reverse engineering và bẻ khóa.
- **Hiệu năng xử lý ảnh lớn:** Ảnh truyện tranh thường có độ phân giải cao (có thể lên đến hàng nghìn pixel). Việc xử lý pixel-level (như xáo trộn các phần của ảnh, giải mã DRM) yêu cầu xử lý hàng triệu pixel nhanh chóng. JavaScript thuần không đủ hiệu năng để xử lý mượt mà, đặc biệt trên thiết bị di động có tài nguyên hạn chế.
- **Bảo mật và obfuscation:** Code JavaScript dễ bị đọc, phân tích và reverse engineering qua DevTools. Bất kỳ thuật toán mã hóa nào viết bằng JavaScript đều có thể bị người dùng xem và hiểu logic. Hệ quả là nguy cơ cao bị bẻ khóa, dẫn đến việc người dùng có thể tự giải mã và lưu trữ ảnh gốc mà không cần quyền truy cập hợp pháp.

Cần một giải pháp chạy trên mọi trình duyệt hiện đại (Chrome, Firefox, Safari, Edge) và các thiết bị khác nhau (desktop, mobile, tablet). Hệ quả là không thể dùng các giải pháp native như C++ compiled code hoặc extension trình duyệt vì không đảm bảo tính tương thích.

3.3.2. Cách giải quyết

Để giải quyết các vấn đề trên, hệ thống đã lựa chọn sử dụng Rust compiled to WebAssembly (WASM) cho việc mã hóa và giải mã ảnh. Đây là giải pháp tối ưu với các lợi ích sau:

Hiệu năng vượt trội: Rust là ngôn ngữ hệ thống, được thiết kế để đạt hiệu năng gần như C/C++ nhưng an toàn hơn về memory. Khi compile sang WASM, code Rust có thể đạt hiệu năng gần như native code. Rust hỗ trợ tốt xử lý song song, cho phép xử lý các pixel operations trên ảnh lớn hiệu quả. Rust cho phép viết code ở mức abstraction cao nhưng vẫn đạt hiệu năng tối ưu khi compile nhờ zero-cost abstractions. Kết quả là việc giải mã ảnh DRM có thể được thực hiện trong vài trăm milliseconds thay vì vài giây như JavaScript thuần, đảm bảo trải nghiệm người dùng mượt mà.

Bảo mật và khó reverse engineering: WASM được compile thành binary format, khó đọc và phân tích hơn nhiều so với JavaScript source code. Mặc dù vẫn có thể reverse engineer, nhưng độ khó tăng đáng kể. Quá trình compile từ Rust sang WASM tự động tạo ra code đã được obfuscate một phần, làm cho việc hiểu logic thuật toán trở nên khó khăn hơn. Rust đảm bảo memory safety tại compile-time, giảm thiểu lỗ hổng bảo mật so với C/C++. Kết quả là thuật toán mã hóa được bảo vệ tốt hơn, giảm nguy cơ bị bẻ khóa và sao chép trái phép.

Tương thích đa nền tảng: WASM là một standard được hỗ trợ bởi tất cả các trình duyệt hiện đại (Chrome, Firefox, Safari, Edge) và các runtime JavaScript (Node.js, Deno). Code WASM có thể chạy trên bất kỳ platform nào có hỗ trợ WASM runtime, bao gồm desktop, mobile, và thậm chí là server-side. Kết quả là một codebase duy nhất có thể chạy trên mọi môi trường mà không cần viết lại code cho từng platform.

Tích hợp dễ dàng với JavaScript: WASM có thể giao tiếp trực tiếp với JavaScript thông qua các function exports và imports, cho phép tích hợp mượt mà với codebase frontend hiện tại. Các tool như wasm-bindgen cho phép generate TypeScript definitions từ Rust code, đảm bảo type safety khi gọi WASM functions từ TypeScript. Kết quả là việc tích hợp WASM module vào ứng dụng Next.js/React hiện tại trở nên đơn giản và an toàn về mặt type.

Kích thước file nhỏ gọn: WASM binary được optimize tốt, thường nhỏ hơn so với JavaScript bundle tương đương cho các thao tác xử lý phức tạp. WASM module có thể được load một cách lazy, chỉ khi cần thiết (khi có ảnh DRM cần giải mã), giảm thiểu ảnh hưởng đến thời gian load trang ban đầu. Kết quả là trải nghiệm người dùng tốt hơn với thời gian load nhanh và hiệu năng cao.

Khả năng mở rộng và bảo trì: Logic mã hóa/giải mã có thể được viết một lần bằng Rust và sử dụng lại ở cả client-side (WASM) và server-side (native Rust hoặc qua FFI). Rust có hệ sinh thái testing mạnh mẽ, cho phép viết unit tests và integration tests một cách dễ dàng. Kết quả là code dễ bảo trì, dễ test, và có thể tái sử dụng ở nhiều nơi trong hệ thống.

Triển khai thực tế trong hệ thống: Trong hệ thống Miru, WASM module được sử dụng như sau. Module được đặt tại đường dẫn /public/wasm/drm_tool/pkg/drm_tool.js và drm_tool_bg.wasm. Hàm chính được sử dụng là unscramble_image_from_drm với các tham số data (Uint8Array), width (number), height (number), và drm_data (string). Component PrivateImage.tsx sử dụng WasmManager singleton để load và sử dụng WASM module khi cần giải mã ảnh DRM. Quy trình hoạt động bao gồm các bước sau: đầu tiên, load ảnh đã mã hóa từ server, tiếp theo vẽ ảnh lên Canvas để lấy ImageData, sau đó gọi WASM function để giải mã, tiếp theo cập nhật Canvas với ảnh đã giải mã; cuối cùng, hiển thị ảnh cho người dùng.

3.3.3. Module code

```
export function decrypt_data(drm_data_encrypt) {
    let deferred3_0;
    let deferred3_1;
    try {
        const ptr0 = passStringToWasm0(drm_data_encrypt, wasm.__wbindgen_malloc, wasm
            __wbindgen_realloc);
        const len0 = WASM_VECTOR_LEN;
        const ret = wasm.decrypt_data(ptr0, len0);
        var ptr2 = ret[0];
        var len2 = ret[1];
        if (ret[3]) {
            ptr2 = 0; len2 = 0;
            throw takeFromExternrefTable0(ret[2]);
        }
        deferred3_0 = ptr2;
        deferred3_1 = len2;
        return getStringFromWasm0(ptr2, len2);
    } finally {
        wasm.__wbindgen_free(deferred3_0, deferred3_1, 1);
    }
}

/**
 * @param {Uint8Array} scrambled_data
 * @param {number} width
 * @param {number} height
 * @param {string} drm_data_encrypt
 * @returns {Uint8Array}
 */
export function unscramble_image_from_drm(scrambled_data, width, height, drm_data_encrypt) {
    const ptr0 = passArray8ToWasm0(scrambled_data, wasm.__wbindgen_malloc);
    const len0 = WASM_VECTOR_LEN;
    const ptr1 = passStringToWasm0(drm_data_encrypt, wasm.__wbindgen_malloc, wasm.__wbindgen_realloc);
    const len1 = WASM_VECTOR_LEN;
    const ret = wasm.unscramble_image_from_drm(ptr0, len0, width, height, ptr1, len1);
    if (ret[3]) {
        throw takeFromExternrefTable0(ret[2]);
    }
    var v3 = getArrayU8FromWasm0(ret[0], ret[1]).slice();
    wasm.__wbindgen_free(ret[0], ret[1] * 1, 1);
    return v3;
}
```

Hình 3.111 Mã hóa ảnh với Rust

3.4. Kiểm thử

```
describe('POST /api/auth/login', () => {
  it('TC-AUTH-001: should return 400 for missing required fields', () => {
    return request(app.getHttpServer())
      .post('/api/auth/login')
      .send({})
      .expect((res) => {
        expect([400, 422]).toContain(res.status);
      });
  });

  it('TC-AUTH-002: should return error for invalid credentials', () => {
    return request(app.getHttpServer())
      .post('/api/auth/login')
      .send({
        accout: 'invaliduser',
        password: 'wrongpassword',
        tokenVerify: 'test-token',
      })
      .expect((res) => {
        expect([400, 401, 422]).toContain(res.status);
      });
  });

  it('TC-AUTH-003: should return error for invalid captcha token', () => {
    return request(app.getHttpServer())
      .post('/api/auth/login')
      .send({
        accout: 'testuser',
        password: 'testpassword',
        tokenVerify: 'invalid-captcha-token',
      })
      .expect((res) => {
        expect([400, 401, 422]).toContain(res.status);
      });
  });
});
```

Hình 3.112 Kịch bản kiểm thử lấy người dùng đăng nhập

```
describe('POST /api/auth/login/token', () => {
  it('TC-AUTH-004: should return 401 without authentication token', () => {
    return request(app.getHttpServer())
      .post('/api/auth/login/token')
      .expect((res) => {
        expect([401, 403]).toContain(res.status);
      });
  });

  it('TC-AUTH-005: should return 401 with invalid token', () => {
    return request(app.getHttpServer())
      .post('/api/auth/login/token')
      .set('Authorization', 'Bearer invalid-token')
      .expect((res) => {
        expect([401, 403]).toContain(res.status);
      });
  });
});
```

Hình 3.113 Kịch bản kiểm thử lấy người dùng đăng nhập với token

```
describe('GET /api/auth/google', () => {
  it('TC-AUTH-013: should redirect to Google OAuth', () => {
    return request(app.getHttpServer())
      .get('/api/auth/google')
      .expect((res) => {
        // OAuth endpoints typically redirect (302) or return 200
        expect([200, 302, 307]).toContain(res.status);
      });
  });
});

describe('GET /api/auth/google/callback', () => {
  it('TC-AUTH-014: should handle Google OAuth callback', () => {
    return request(app.getHttpServer())
      .get('/api/auth/google/callback')
      .expect((res) => {
        // OAuth callback may return various status codes depending on state
        expect([200, 302, 307, 400, 401]).toContain(res.status);
      });
  });

  it('TC-AUTH-015: should handle Google OAuth callback with state parameter', () => {
    return request(app.getHttpServer())
      .get('/api/auth/google/callback?state=test-state')
      .expect((res) => {
        expect([200, 302, 307, 400, 401]).toContain(res.status);
      });
  });
});
```

Hình 3.114 Kịch bản kiểm thử đăng nhập với google

```
describe('PUT /api/users/transfer/role', () => {
  it('TC-USER-018: should return 401 without authentication token', () => {
    return request(app.getHttpServer())
      .put('/api/users/transfer/role')
      .send({ userId: 1, role: RoleUserEnum.ADMIN })
      .expect((res) => {
        expect([401, 403]).toContain(res.status);
      });
  });

  it('TC-USER-019: should return 403 for non-admin user', async () => {
    const nonAdminToken = await createTestAccessToken({
      userId: 1,
      username: 'testuser',
      role: RoleUserEnum.GUEST,
      banned: 0,
    });

    return request(app.getHttpServer())
      .put('/api/users/transfer/role')
      .set('Authorization', `Bearer ${nonAdminToken}`)
      .send({ userId: 1, role: RoleUserEnum.ADMIN })
      .expect((res) => {
        expect([401, 403]).toContain(res.status);
      });
  });

  it('TC-USER-020: should transfer role with admin token', async () => {
    const adminToken = await createTestAccessToken({
      userId: 1,
      username: 'admin',
      role: RoleUserEnum.ADMIN,
      banned: 0,
    });

    return request(app.getHttpServer())
      .put('/api/users/transfer/role')
      .set('Authorization', `Bearer ${adminToken}`)
      .send({ userId: 1, role: RoleUserEnum.ADMIN })
      .expect((res) => {
        expect([200, 404, 400]).toContain(res.status);
      });
  });
});
```

Hình 3.115 Kịch bản kiểm thử chuyển quyền hạn người dùng

```
describe('PATCH /api/books/toggle/featured', () => {
  it('TC-BOOK-028: should return 401 without authentication token', () => {
    return request(app.getHttpServer())
      .patch('/api/books/toggle/featured')
      .send({ bookId: 1 })
      .expect((res) => {
        expect([401, 403]).toContain(res.status);
      });
  });

  it('TC-BOOK-029: should return 403 for non-admin user', async () => {
    const nonAdminToken = await createTestAccessToken({
      userId: 1,
      username: 'testuser',
      role: RoleUserEnum.GUEST,
      banned: 0,
    });

    return request(app.getHttpServer())
      .patch('/api/books/toggle/featured')
      .set('Authorization', `Bearer ${nonAdminToken}`)
      .send({ bookId: 1 })
      .expect((res) => {
        expect([401, 403]).toContain(res.status);
      });
  });

  it('TC-BOOK-030: should toggle featured with admin token', async () => {
    const adminToken = await createTestAccessToken({
      userId: 1,
      username: 'admin',
      role: RoleUserEnum.ADMIN,
      banned: 0,
    });

    return request(app.getHttpServer())
      .patch('/api/books/toggle/featured')
      .set('Authorization', `Bearer ${adminToken}`)
      .send({ bookId: 1 })
      .expect((res) => {
        expect([200, 404, 400]).toContain(res.status);
      });
  });
});
```

Hình 3.116 Kịch bản kiểm thử đánh dấu truyện hay

```
describe('POST /api/chapters/upload/images/cloud-cache', () => {
  it('TC-CHAPTER-011: should return 401 without authentication token', () => {
    return request(app.getHttpServer())
      .post('/api/chapters/upload/images/cloud-cache?cacheId=test&bookId=1&chapterNumber=1')
      .expect((res) => {
        expect([401, 403, 400]).toContain(res.status);
      });
  });

  it('TC-CHAPTER-012: should upload images with valid token', async () => {
    const testToken = await createTestAccessToken({
      userId: 1,
      username: 'testuser',
      role: RoleUserEnum.GUEST,
      banned: 0,
    });

    return request(app.getHttpServer())
      .post('/api/chapters/upload/images/cloud-cache?cacheId=test&bookId=1&chapterNumber=1')
      .set('Authorization', `Bearer ${testToken}`)
      .expect((res) => {
        // May require file upload, so expect various status codes
        expect([200, 400, 404, 422]).toContain(res.status);
      });
  });
});
```

Hình 3.117 Kịch bản kiểm thử đăng tải ảnh lên đám mây tạm

Kết quả kiểm thử

```
PS D:\WORKSPACE\WEB\MY\miru\miru-be> npm run test:e2e
> miru-be@0.0.1 test:e2e
> jest --config ./test/jest-e2e.json

PASS test/app.e2e-spec.ts (5.682 s)
PASS test/auth.e2e-spec.ts (6.519 s)
PASS test/user.e2e-spec.ts (7.075 s)
PASS test/chapter.e2e-spec.ts (7.951 s)
PASS test/book.e2e-spec.ts (8.173 s)
A worker process has failed to exit gracefully and has been force exited. This is likely caused by test
s leaking due to improper teardown. Try running with --detectOpenHandles to find leaks. Active timers c
an also cause this, ensure that .unref() was called on them.

Test Suites: 5 passed, 5 total
Tests:       36 skipped, 40 passed, 76 total
Snapshots:   0 total
Time:        9.357 s
Ran all test suites.
```

Hình 2.118 Kết quả kiểm thử trong module auth, user, book, chapter

10 connections

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	256 ms	316 ms	642 ms	675 ms	327.51 ms	71.21 ms	718 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	16	16	32	35	30,1	5,03	16
Bytes/Sec	221 kB	221 kB	441 kB	483 kB	415 kB	69.3 kB	221 kB

Hình 2.119 Kết quả kiểm thử chịu tải 10 connection liên tục

KẾT LUẬN

Trên đây là toàn bộ báo cáo đồ án tốt nghiệp. Như vậy, sau một thời gian tìm hiểu, nghiên cứu thì em đã hoàn thành đồ án tốt nghiệp "Nền Tảng Phát Hành Truyện Tranh Số Với Hệ Thống Bảo Vệ Nội Dung Và Trải Nghiệm Đọc Tối Ưu".

1. Kết quả đạt được

Về công nghệ

- Thành thạo sử dụng các công nghệ hiện đại: Next.js cho frontend, NestJS cho backend, MySQL với Prisma ORM, và xử lý hình ảnh bằng Rust + WebAssembly.
- Hiểu rõ hơn về kiến trúc hệ thống microservices với các server chuyên biệt (Server chính, Server cache hình ảnh, Server AI).
- Nắm vững kỹ năng triển khai và quản lý hạ tầng: sử dụng Docker, Nginx, PM2, GitHub Actions, và áp dụng chiến lược Blue-Green Deployment.
- Biết cách phân tích, thiết kế, vận hành một nền tảng web hoàn chỉnh với hệ thống phân quyền phức tạp.
- Áp dụng được các kiến thức về cache management, xử lý hình ảnh, bảo mật DRM vào thực tế.

Về chương trình

- Đã hoàn thành được các mục tiêu chức năng chính: hệ thống đọc truyện, quản lý người dùng, cộng đồng tương tác, và công cụ đăng truyện.
- Xây dựng thành công hệ thống phân quyền đa cấp (Admin, Editor, Creator, Guest) đáp ứng nhu cầu quản lý phức tạp.
- Tích hợp AI để hỗ trợ dịch tự động OCR giúp creator tiết kiệm thời gian.
- Giao diện thân thiện, dễ sử dụng, hỗ trợ tương tác cộng đồng giàu có (chat, bình luận, hệ thống điểm danh).
- Xử lý lỗi linh hoạt: hỗ trợ nhiều server hình ảnh, tự động chuyển đổi khi ảnh bị lỗi.
- Đáp ứng được nhu cầu của một nền tảng truyện tranh hiện đại với tính năng xác thực, tìm kiếm, lịch sử đọc và kiếm tiền ảo.

2. Hạn chế

Mặc dù đã đạt được những kết quả tích cực, em nhận thức rõ rằng đồ án còn tồn tại một số hạn chế cần được cải thiện trong tương lai. Dưới đây là một số điểm hạn chế mà em đã nhận diện:

- Hiệu suất: Khi có số lượng người dùng lớn truy cập đồng thời, cần tối ưu hóa thêm cache strategy, database queries, và xử lý concurrent connections để đảm bảo response time tối ưu.
- Bảo mật nâng cao: Mặc dù đã có hệ thống DRM cơ bản, cần tăng cường thêm các biện pháp như rate limiting, DDOS protection, xác thực hai yếu tố (2FA), và mã hóa end-to-end cho dữ liệu nhạy cảm.
- Xử lý AI: Độ chính xác của hệ thống dịch tự động còn cần cải thiện để giảm khối lượng chỉnh sửa thủ công của creator.
- Quản lý tệp lớn: Hiện tại còn một số vấn đề khi xử lý các chapter với số lượng hình ảnh rất lớn hoặc kích thước tệp lớn.
- Mở rộng tính năng: Chưa có các tính năng tiên tiến như donate tác giả, subscription model, hay tích hợp với các platform khác.

3. Hướng phát triển

Dựa vào những hạn chế đã đề cập, dưới đây là những hướng phát triển mà tôi đề xuất để nâng cao chất lượng và khả năng sử dụng của đồ án:

- Tối ưu hóa hiệu suất: Áp dụng các kỹ thuật như database sharding, caching multilayer (Redis, CDN), query optimization, lazy loading, code splitting, và monitoring performance bằng các công cụ như New Relic hoặc DataDog.
- Bảo mật cao cấp: Tăng cường bảo mật bằng cách triển khai rate limiting, DDOS protection, JWT token refresh strategy, xác thực hai yếu tố (2FA), audit logging, và thực hiện security audit định kỳ.
- Cải thiện hệ thống AI: Đầu tư vào mô hình AI hiện đại hơn, fine-tuning cho từng loại thể loại truyện tranh, tích hợp feedback loop để học từ các chỉnh sửa của creator.
- Hỗ trợ đa nền tảng: Phát triển ứng dụng mobile (React Native hoặc Flutter) để mở rộng đối tượng người dùng và cung cấp trải nghiệm đọc tốt hơn trên thiết bị di động.
- Mở rộng mô hình kinh doanh: Thêm các tính năng như hệ thống subscription, donate cho creator, premium chapters, và tích hợp các dịch vụ thanh toán để tạo doanh thu bền vững.
- Tính năng cộng đồng nâng cao: Phát triển hệ thống xếp hạng creator, forum thảo luận, tổ chức các cuộc thi vẽ tranh, và hệ thống recommendation nâng cao dựa trên machine learning.

TÀI LIỆU THAM KHẢO

1. Next.js. (2025). Next.js Documentation.
URL: <https://nextjs.org/docs>
2. React. (2025). React Documentation.
URL: <https://react.dev>
3. TypeScript. (2025). TypeScript Documentation.
URL: <https://www.typescriptlang.org/docs>
4. NestJS. (2025). NestJS Documentation.
URL: <https://docs.nestjs.com>
5. Prisma. (2025). Prisma Documentation.
URL: <https://www.prisma.io/docs>
6. Flask. (2025). Flask Documentation.
URL: <https://flask.palletsprojects.com>
7. PyTorch. (2025). PyTorch Documentation.
URL: <https://pytorch.org/docs/stable/index.html>
8. Redux Toolkit. (2025). Redux Toolkit Documentation.
URL: <https://redux-toolkit.js.org>
9. Tailwind CSS. (2025). Tailwind CSS Documentation.
URL: <https://tailwindcss.com/docs>
10. NextAuth.js. (2025). NextAuth.js Documentation.
URL: <https://next-auth.js.org>
11. Socket.io. (2025). Socket.io Documentation.
URL: <https://socket.io/docs/v4>
12. Chart.js. (2025). Chart.js Documentation.
URL: <https://www.chartjs.org/docs/latest>
13. Framer Motion. (2025). Framer Motion Documentation.
URL: <https://www.framer.com/motion>
14. Axios. (2025). Axios Documentation.
URL: <https://axios-http.com/docs/intro>

15. Passport.js. (2025). Passport.js Documentation.
URL: <http://www.passportjs.org/docs>
16. OpenAI. (2025). OpenAI API Documentation.
URL: <https://platform.openai.com/docs>
17. EasyOCR. (2025). EasyOCR Documentation.
URL: <https://github.com/JaidedAI/EasyOCR>
18. PaddleOCR. (2025). PaddleOCR Documentation.
URL: <https://github.com/PaddlePaddle/PaddleOCR>
19. Transformers. (2025). Hugging Face Transformers Documentation.
URL: <https://huggingface.co/docs/transformers>
20. OpenCV. (2025). OpenCV Documentation.
URL: <https://docs.opencv.org>
21. React Quill. (2025). React Quill Documentation.
URL: <https://github.com/zenoamaro/react-quill>
22. Zod. (2025). Zod Documentation.
URL: <https://zod.dev>
23. AWS SDK. (2025). AWS SDK for JavaScript Documentation.
URL: <https://docs.aws.amazon.com/sdk-for-javascript>
24. Sharp. (2025). Sharp Documentation.
URL: <https://sharp.pixelplumbing.com>
25. Canvas. (2025). node-canvas Documentation.
URL: <https://github.com/Automattic/node-canvas>
26. PostgreSQL. (2025). PostgreSQL Documentation.
URL: <https://www.postgresql.org/docs>
27. JWT. (2025). JSON Web Token Documentation.
URL: <https://jwt.io/introduction>