



## **ĐỒ ÁN CƠ SỞ**

**ĐỀ TÀI**

### **XÂY DỰNG WEBSITE MẠNG XÃ HỘI HONEY**

Giảng viên hướng dẫn: **Trần Văn Hùng**

Sinh viên thực hiện: **2280602828 - Trần Tân Tài**

**2280618597 - Trần Đình Ty**

**2280603283 - Đặng Doanh Toại**

**2280603036 - Phan Thanh Thiên**

# Mục lục

<b>LỜI MỞ ĐẦU</b>	<b>1</b>
<b>LỜI CAM ĐOAN</b>	<b>2</b>
<b>Chương 1. TỔNG QUAN VỀ HONEY SOCIAL</b>	<b>3</b>
1.1 Giới thiệu . . . . .	3
1.2 Tổng quan hệ thống . . . . .	3
1.2.1 Mục đích hệ thống . . . . .	3
1.2.2 Khảo sát các sản phẩm tương tự . . . . .	4
1.2.3 Yêu cầu hoạt động của ứng dụng . . . . .	4
1.2.3.1 Phần dành cho người dùng cuối . . . . .	4
1.2.3.2 Phần dành cho người quản trị . . . . .	5
1.3 Thiết kế tương tác . . . . .	5
1.4 Phương pháp tiếp cận và giải quyết vấn đề . . . . .	6
1.4.1 Mô hình tổng quát hệ thống . . . . .	6
1.4.2 Phương pháp xây dựng phần mềm . . . . .	6
1.4.3 Kiến trúc phần mềm . . . . .	6
1.4.4 Công nghệ triển khai hệ thống . . . . .	7
1.4.4.1 Server-Side . . . . .	7
1.4.4.2 Client-Side . . . . .	7
1.4.4.3 Công nghệ hỗ trợ . . . . .	7
1.5 Tổng kết chương . . . . .	8
<b>Chương 2. CƠ SỞ LÝ THUYẾT</b>	<b>9</b>
2.1 Giới thiệu . . . . .	9
2.2 Mô hình MVC (Model-View-Controller) . . . . .	9
2.3 Công nghệ nền tảng . . . . .	10
2.3.1 MongoDB . . . . .	10
2.3.2 Express.js . . . . .	11
2.3.3 React.js . . . . .	12
2.3.4 Node.js . . . . .	13
2.4 Công nghệ hỗ trợ . . . . .	14
2.4.1 JWT (JSON Web Token) . . . . .	14
2.4.2 WebSocket và Socket.IO . . . . .	15
2.4.3 Cloudinary . . . . .	16
2.4.4 Elasticsearch và Vector Search . . . . .	16

2.4.5 Caching Strategy . . . . .	17
2.4.6 Lazy Loading . . . . .	18
2.5 Architecture Patterns . . . . .	18
2.5.1 Monolithic vs Microservices . . . . .	18
2.5.2 Event-Driven Architecture . . . . .	19
2.6 Tổng kết chương . . . . .	19
<b>Chương 3. Phân tích thiết kế hệ thống</b>	<b>21</b>
3.1 Phân Tích Hệ Thống . . . . .	21
3.1.1 Biểu đồ Usecase . . . . .	21
3.1.2 Kịch bản của các UseCase . . . . .	31
3.2 Thiết Kế Hệ Thống . . . . .	49
<b>Chương 4. THIẾT KẾ CƠ SỞ DỮ LIỆU</b>	<b>57</b>
<b>Chương 5. KẾT QUẢ THỰC NGHIỆM</b>	<b>66</b>
<b>Chương 6. KẾT LUẬN VÀ KIẾN NGHỊ</b>	<b>75</b>
6.1 Kết Luận . . . . .	75
6.1.1 Tổng kết quá trình thực hiện đồ án . . . . .	75
6.1.2 Kết quả đạt được . . . . .	75
6.1.3 Hạn chế và thách thức . . . . .	76
6.2 Kiến nghị và hướng phát triển . . . . .	76
6.2.1 Phát triển ứng dụng di động . . . . .	76
6.2.2 Nâng cấp hệ thống gợi ý nội dung . . . . .	76
6.2.3 Triển khai AI Chatbot học tự động . . . . .	76
6.2.4 Tối ưu cơ chế làm mới Redis Cache . . . . .	76
6.2.5 Cải tiến hệ thống Post Recommendation . . . . .	77
6.2.6 Phát triển ChatboxAI nội bộ . . . . .	77
<b>TÀI LIỆU THAM KHẢO</b>	<b>78</b>

# Danh sách hình vẽ

1	Hình ảnh Use Case Tổng quát . . . . .	21
2	Hình ảnh Đăng bài viết . . . . .	22
3	Hình ảnh Tìm kiếm nâng cao . . . . .	23
4	Hình ảnh Thông báo . . . . .	24
5	Hình ảnh Xem Hồ sơ người dùng . . . . .	25
6	Hình ảnh Báo cáo bài viết . . . . .	26
7	Hình ảnh Trò chuyện AI . . . . .	26
8	Hình ảnh Xem bảng tin . . . . .	27
9	Hình ảnh Xem hồ sơ người khác . . . . .	28
10	Hình ảnh Đăng ký . . . . .	28
11	Hình ảnh Quản lý nội dung báo cáo . . . . .	29
12	Hình ảnh Tương tác bài viết . . . . .	30
13	Hình ảnh Chat AI . . . . .	49
14	Hình ảnh Chat AI 2 . . . . .	50
15	Hình ảnh Đăng bài viết . . . . .	51
16	Hình ảnh Gợi ý bài viết . . . . .	52
17	Hình ảnh Gợi ý bạn bè . . . . .	53
18	Hình ảnh Hệ thống tin nhắn trên Honey Social . . . . .	54
19	Hình ảnh Hệ thống tin nhắn trên Honey Social 2 . . . . .	55
20	Hình ảnh Tìm kiếm nâng cao . . . . .	56
21	Hình ảnh Quan hệ giữa các bảng . . . . .	65
22	Hình ảnh Bảng kiểm duyệt nội dung vi phạm . . . . .	66
23	Hình ảnh Ngữ nghĩa . . . . .	67
24	Hình ảnh Nhắn tin . . . . .	68
25	Hình ảnh Gợi ý . . . . .	69
26	Hình ảnh Trí tuệ nhân tạo . . . . .	70
27	Hình ảnh Xác nhận mail . . . . .	71
28	Hình ảnh Deployment . . . . .	73

# Danh sách bảng

1	Bảng thông tin hoạt động của chức năng đăng bài viết . . . . .	31
2	Bảng thông tin hoạt động của chức năng tìm kiếm nâng cao . . . . .	32
3	Bảng thông tin hoạt động của chức năng thông báo . . . . .	33
4	Bảng thông tin hoạt động của chức năng xem hồ sơ người dùng . .	34
5	Bảng thông tin hoạt động của chức năng báo cáo bài viết . . . . .	35
6	Bảng thông tin hoạt động của chức năng trò chuyện AI . . . . .	37
7	Bảng thông tin hoạt động của chức năng xem bảng tin . . . . .	39
8	Bảng thông tin hoạt động của chức năng xem hồ sơ người khác .	41
9	Bảng thông tin hoạt động của chức năng đăng ký . . . . .	43
10	Bảng thông tin hoạt động của chức năng quản lý nội dung báo cáo .	45
11	Bảng thông tin hoạt động của chức năng tương tác bài viết . . . . .	47
12	Cấu trúc của bảng posts . . . . .	57
13	Cấu trúc của bảng chatmessages . . . . .	57
14	Cấu trúc của bảng threads . . . . .	58
15	Cấu trúc của bảng reports . . . . .	59
16	Cấu trúc của bảng verifications . . . . .	60
17	Cấu trúc của bảng messages . . . . .	61
18	Cấu trúc của bảng users . . . . .	62
19	Cấu trúc của bảng notifications . . . . .	63

## LỜI MỞ ĐẦU

Trong thời đại công nghệ số phát triển mạnh mẽ, mạng xã hội đã trở thành một phần không thể thiếu trong đời sống con người. Các nền tảng như Facebook, Twitter, Instagram hay Threads của Meta giúp kết nối hàng triệu người dùng trên toàn thế giới, hỗ trợ giao tiếp, chia sẻ thông tin và mở rộng cơ hội kinh doanh.

Threads, một nền tảng mới của Meta, được thiết kế để tối ưu hóa khả năng tương tác nhanh chóng, đơn giản hóa trải nghiệm chia sẻ nội dung dưới dạng văn bản, hình ảnh và video. Với sự thành công ban đầu của Threads, ý tưởng xây dựng một nền tảng mạng xã hội tương tự giúp sinh viên hiểu rõ hơn về cách phát triển một hệ thống mạng xã hội từ đầu, ứng dụng công nghệ hiện đại và tối ưu hóa hiệu suất.

Chính vì vậy, nhóm chúng tôi quyết định thực hiện đồ án “Xây dựng website mạng xã hội Honey” với mục tiêu xây dựng một hệ thống mạng xã hội có đầy đủ các tính năng quan trọng như đăng bài, bình luận, theo dõi người dùng, trò chuyện trực tuyến và thông báo thời gian thực.

## **LỜI CAM ĐOAN**

Nhóm chúng em xin cam đoan rằng đồ án “**Xây dựng website mạng xã hội Honey**” là kết quả nghiên cứu và thực hiện của tất cả thành viên, dưới sự hướng dẫn của giảng viên **Trần Văn Hùng**.

Toàn bộ nội dung trong đồ án này được chúng tôi tìm hiểu, phân tích và triển khai dựa trên kiến thức đã học, các tài liệu tham khảo có trích dẫn đầy đủ. Chúng tôi cam kết không sao chép hoặc sử dụng trái phép nội dung từ bất kỳ nguồn nào mà không ghi rõ nguồn gốc.

Nếu phát hiện có bất kỳ hành vi đạo văn hoặc gian lận nào trong đồ án này, tôi xin chịu hoàn toàn trách nhiệm trước nhà trường và hội đồng đánh giá.

Tôi xin chân thành cảm ơn!

# **Chương 1. TỔNG QUAN VỀ HONEY SOCIAL**

## **1.1. Giới thiệu**

Trong thời đại công nghệ số phát triển mạnh mẽ, mạng xã hội đã trở thành một phần không thể thiếu trong đời sống con người, hỗ trợ kết nối, chia sẻ thông tin và mở rộng cơ hội kinh doanh. Các nền tảng như Facebook, Twitter, Instagram hay Threads của Meta đã đạt được thành công lớn trong việc kết nối hàng triệu người dùng trên toàn cầu. Tuy nhiên, các nền tảng này vẫn đối mặt với các thách thức như nội dung độc hại, thiếu cá nhân hóa thông minh, và trải nghiệm người dùng chưa tối ưu.

Đồ án “Xây dựng website mạng xã hội Honey” được thực hiện với mục tiêu phát triển một nền tảng mạng xã hội hiện đại, tích hợp trí tuệ nhân tạo (AI) để mang lại trải nghiệm an toàn, thông minh và cá nhân hóa cho người dùng. Hệ thống sử dụng các công nghệ tiên tiến như MERN Stack, OpenAI Moderation API, Elasticsearch, và RabbitMQ để đảm bảo hiệu suất cao, khả năng mở rộng, và quản lý nội dung hiệu quả. Chương này sẽ trình bày tổng quan về hệ thống, bao gồm mục đích, yêu cầu chức năng, thiết kế tương tác, phương pháp tiếp cận, và các công nghệ triển khai.

## **1.2. Tổng quan hệ thống**

### **1.2.1. Mục đích hệ thống**

Mạng xã hội Honey được thiết kế để cung cấp một nền tảng kết nối cộng đồng, cho phép người dùng chia sẻ bài viết, tương tác (thích, bình luận), theo dõi lẫn nhau, và trò chuyện theo thời gian thực. Hệ thống tích hợp AI để cá nhân hóa nội dung, kiểm duyệt tự động, và hỗ trợ người dùng thông qua chatbot thông minh. Mục tiêu chính bao gồm:

- Xây dựng một nền tảng mạng xã hội với giao diện thân thiện, responsive, hỗ trợ chế độ sáng/tối.
- Tăng cường trải nghiệm người dùng thông qua gợi ý bài viết cá nhân hóa và tìm kiếm thông minh.
- Đảm bảo an toàn nội dung bằng cách tích hợp OpenAI Moderation API để kiểm duyệt văn bản và hình ảnh.
- Tối ưu hóa hiệu suất hệ thống với Redis caching, RabbitMQ, và Elasticsearch KNN.

Hệ thống không nhằm cạnh tranh với các nền tảng lớn như Facebook hay Instagram, mà tập trung vào việc cung cấp một giải pháp mạng xã hội quy mô vừa phải, ứng dụng công nghệ hiện đại để giải quyết các vấn đề về nội dung, bảo mật, và trải nghiệm người dùng.

### **1.2.2. Khảo sát các sản phẩm tương tự**

Các nền tảng mạng xã hội hiện tại như Threads, Instagram, và Twitter đã đạt được nhiều thành công, nhưng vẫn tồn tại một số hạn chế:

- **Threads (Meta)**: Tối ưu hóa tương tác nhanh với nội dung dạng văn bản, hình ảnh, và video, nhưng thiếu các tính năng cá nhân hóa thông minh và kiểm duyệt tự động hiệu quả.
- **Instagram**: Tập trung vào hình ảnh và video, nhưng các gợi ý nội dung đôi khi thiếu chính xác và không hỗ trợ chatbot thông minh.
- **Twitter**: Hỗ trợ chia sẻ nhanh chóng, nhưng gặp vấn đề về nội dung độc hại và kiểm duyệt thủ công tốn kém.

Dựa trên khảo sát, Honey Social đề xuất các cải tiến:

- Tích hợp OpenAI Moderation API để tự động kiểm duyệt nội dung.
- Sử dụng vector embeddings và Elasticsearch KNN để gợi ý bài viết cá nhân hóa.
- Triển khai chatbot AI hỗ trợ người dùng tương tác và báo cáo nội dung vi phạm.

### **1.2.3. Yêu cầu hoạt động của ứng dụng**

#### **1.2.3.1 Phản hồi cho người dùng cuối**

Người dùng cuối (end-user) có thể thực hiện các chức năng sau:

- **Đăng ký và đăng nhập**: Tạo tài khoản với xác thực email qua Resend API, đăng nhập an toàn bằng JWT.
- **Quản lý hồ sơ**: Cập nhật thông tin cá nhân, ảnh đại diện (lưu trữ trên Cloudinary), và xem hồ sơ của người dùng khác.
- **Đăng bài viết**: Tạo bài viết với văn bản và hình ảnh, chia sẻ liên kết bài viết.

- **Tương tác bài viết:** Thích, bình luận, và phản hồi bình luận.
- **Xem bảng tin:** Hiển thị bài viết từ người dùng đang theo dõi và gợi ý bài viết dựa trên sở thích.
- **Tìm kiếm nâng cao:** Sử dụng Elasticsearch để tìm kiếm bài viết với fuzzy matching và gợi ý vector.
- **Trò chuyện AI:** Tương tác với chatbot AI để nhận hỗ trợ hoặc gợi ý nội dung.
- **Báo cáo bài viết:** Gửi báo cáo về nội dung vi phạm với lý do cụ thể.
- **Thông báo:** Nhận thông báo thời gian thực về tương tác (thích, bình luận) qua Socket.IO.

#### 1.2.3.2 Phản dành cho người quản trị

Quản trị viên (admin) có các chức năng:

- **Quản lý người dùng:** Xem, chỉnh sửa, hoặc khóa tài khoản người dùng.
- **Quản lý nội dung báo cáo:** Xem xét và xử lý báo cáo vi phạm (xóa bài viết, khóa tài khoản, hoặc bỏ qua).
- **Phân loại vi phạm:** Đánh giá mức độ vi phạm (nhẹ, vừa, nặng) để đưa ra hành động phù hợp.
- **Giao diện quản trị:** Sử dụng admin dashboard để quản lý báo cáo, tìm kiếm và lọc theo thời gian hoặc mức độ vi phạm.

### 1.3. Thiết kế tương tác

Hệ thống Honey Social được thiết kế với giao diện thân thiện, responsive, hoạt động tốt trên cả desktop và mobile. Giao diện hỗ trợ chế độ sáng/tối để tối ưu trải nghiệm người dùng. Các thành phần chính bao gồm:

- **Trang chủ/Bảng tin:** Hiển thị bài viết từ người dùng đang theo dõi và bài viết gợi ý, sử dụng lazy loading để tối ưu tốc độ tải.
- **Hồ sơ người dùng:** Hiển thị thông tin cá nhân, bài viết, số lượng người theo dõi/đang theo dõi, và các nút tương tác (theo dõi, nhắn tin).

- **Giao diện chat:** Hỗ trợ trò chuyện thời gian thực với người dùng khác và chatbot AI, tích hợp Socket.IO.
- **Admin dashboard:** Cung cấp bảng điều khiển để quản lý báo cáo và người dùng, với khả năng tìm kiếm và lọc dữ liệu.

## 1.4. Phương pháp tiếp cận và giải quyết vấn đề

### 1.4.1. Mô hình tổng quát hệ thống

Hệ thống Honey Social được xây dựng dựa trên kiến trúc MERN Stack, kết hợp với các công nghệ hỗ trợ như OpenAI API, Cloudinary, Redis, và RabbitMQ. Mô hình tổng quát bao gồm:

- **Client-side:** React.js quản lý giao diện, Socket.IO xử lý tương tác thời gian thực.
- **Server-side:** Node.js và Express.js xử lý API, tích hợp RabbitMQ cho tác vụ bất đồng bộ và Redis cho caching.
- **Cơ sở dữ liệu:** MongoDB lưu trữ dữ liệu người dùng, bài viết, tin nhắn, và báo cáo.
- **Bên thứ ba:** OpenAI API (kiểm duyệt và chatbot), Cloudinary (lưu trữ media), Resend API (gửi email xác thực).

### 1.4.2. Phương pháp xây dựng phần mềm

Dự án áp dụng phương pháp phát triển phần mềm Agile, với các giai đoạn:

- Phân tích yêu cầu: Xác định các chức năng chính và công nghệ sử dụng.
- Thiết kế hệ thống: Xây dựng kiến trúc MERN Stack, thiết kế cơ sở dữ liệu, và tích hợp AI.
- Phát triển: Triển khai từng module (quản lý người dùng, bài viết, chat, kiểm duyệt).
- Triển khai: Đưa hệ thống lên AWS EC2.

### 1.4.3. Kiến trúc phần mềm

Hệ thống sử dụng kiến trúc RESTful API với MERN Stack:

- **Frontend:** React.js với component-based architecture, sử dụng Virtual DOM để render nhanh.

- **Backend:** Express.js quản lý routing và middleware, tích hợp JWT cho xác thực.
- **Database:** MongoDB với schema linh hoạt, hỗ trợ index trên userId để tối ưu truy vấn.

#### 1.4.4. Công nghệ triển khai hệ thống

##### 1.4.4.1 Server-Side

- **MongoDB:** Cơ sở dữ liệu NoSQL lưu trữ dữ liệu dạng JSON/BSON, quản lý người dùng, bài viết, tin nhắn, và báo cáo. Sử dụng MongoDB Atlas trên AWS để đảm bảo khả năng mở rộng.
- **Node.js:** Môi trường chạy JavaScript server-side, xử lý đồng thời nhiều kết nối với mô hình bất đồng bộ.
- **Express.js:** Framework nhẹ, hỗ trợ xây dựng API RESTful, quản lý routing và middleware.
- **RabbitMQ:** Xử lý tác vụ bất đồng bộ, như xếp hàng kiểm duyệt nội dung.
- **Redis:** Caching dữ liệu (Cache-Aside) để giảm tải cơ sở dữ liệu, tối ưu API GetFeedPosts.

##### 1.4.4.2 Client-Side

- **React.js:** Thư viện frontend xây dựng giao diện responsive, sử dụng component tái sử dụng và Virtual DOM.
- **Socket.IO:** Hỗ trợ chat và thông báo thời gian thực, sử dụng WebSocket để giảm độ trễ.
- **Cloudinary:** Lưu trữ và tối ưu hóa hình ảnh (avatar, bài viết), tự động nén sang WebP và điều chỉnh kích thước.

##### 1.4.4.3 Công nghệ hỗ trợ

- **JWT:** Xác thực người dùng với token mã hóa, đảm bảo an toàn phiên đăng nhập.
- **Elasticsearch:** Hỗ trợ tìm kiếm nâng cao và gợi ý bài viết bằng vector search (KNN).

- **OpenAI API:** Kiểm duyệt nội dung (Moderation API) và cung cấp chatbot thông minh.
- **Resend API:** Gửi email xác thực tài khoản.

## 1.5. Tổng kết chương

Chương này đã trình bày tổng quan về hệ thống mạng xã hội Honey, bao gồm mục đích, khảo sát các sản phẩm tương tự, yêu cầu chức năng, thiết kế tương tác, và phương pháp triển khai. Hệ thống sử dụng MERN Stack kết hợp với các công nghệ AI hiện đại để mang lại trải nghiệm an toàn, thông minh, và hiệu quả. Chương tiếp theo sẽ đi sâu vào cơ sở lý thuyết của các công nghệ được sử dụng, bao gồm MERN Stack, JWT, Socket.IO, và OpenAI API.

## **Chương 2. CƠ SỞ LÝ THUYẾT**

### **2.1. Giới thiệu**

Chương này trình bày cơ sở lý thuyết của các công nghệ được sử dụng trong đồ án "Xây dựng website mạng xã hội Honey". Các công nghệ chính bao gồm MERN Stack (MongoDB, Express.js, React.js, Node.js), cùng với các công nghệ hỗ trợ như JWT, Socket.IO, Cloudinary, Elasticsearch, và OpenAI API. Mỗi công nghệ sẽ được phân tích về khái niệm, kiến trúc, ứng dụng trong hệ thống, lợi ích, và cách tối ưu hóa.

### **2.2. Mô hình MVC (Model-View-Controller)**

**Khái niệm:** MVC là mô hình kiến trúc phần mềm chia ứng dụng thành ba thành phần chính: Model (Mô hình), View (Giao diện), và Controller (Điều khiển). Mô hình này tách biệt logic nghiệp vụ, giao diện người dùng và xử lý yêu cầu, tạo ra code có tính bảo trì cao và dễ mở rộng.

**Các thành phần của MVC:**

#### **1. Model (Mô hình):**

- Quản lý dữ liệu và logic nghiệp vụ của ứng dụng
- Tương tác với cơ sở dữ liệu, validation dữ liệu
- Không phụ thuộc vào View và Controller
- Thông báo cho View khi dữ liệu thay đổi

#### **2. View (Giao diện):**

- Hiển thị dữ liệu cho người dùng và nhận input
- Render UI components, xử lý tương tác người dùng
- Không chứa business logic, chỉ hiển thị dữ liệu
- Lắng nghe thay đổi từ Model để cập nhật giao diện

#### **3. Controller (Điều khiển):**

- Xử lý input từ user, điều phối giữa Model và View
- Nhận request từ View, gọi Model xử lý
- Cập nhật View với dữ liệu từ Model

- Không chứa business logic hay presentation logic

### **Luồng hoạt động:**

1. User tương tác với View (click, submit form)
2. View gửi request đến Controller
3. Controller xử lý request và gọi Model
4. Model thực hiện business logic và truy cập database
5. Model trả kết quả về Controller
6. Controller cập nhật View với dữ liệu mới
7. View hiển thị kết quả cho User

### **Lợi ích của MVC:**

- **Separation of Concerns:** Tách biệt rõ ràng các chức năng
- **Maintainability:** Dễ sửa đổi và bảo trì từng thành phần
- **Testability:** Có thể test riêng từng layer
- **Reusability:** Tái sử dụng Model và View cho nhiều mục đích
- **Scalability:** Hỗ trợ phát triển team và mở rộng hệ thống

### **Ứng dụng trong Honey Social:**

- **Model:** Mongoose schemas (User, Post, Comment), business logic
- **View:** React components hiển thị UI, JSON responses từ API
- **Controller:** Express.js route handlers xử lý HTTP requests

## **2.3. Công nghệ nền tảng**

### **2.3.1. MongoDB**

**Khái niệm và kiến trúc:** MongoDB là hệ quản trị cơ sở dữ liệu NoSQL thuộc loại document-based, lưu trữ dữ liệu dưới dạng BSON (Binary JSON). Khác với cơ sở dữ liệu quan hệ truyền thống, MongoDB không yêu cầu schema cố định, cho phép lưu trữ các document có cấu trúc khác nhau trong cùng một collection.

#### **Đặc điểm kỹ thuật:**

- **Document Structure:** Hỗ trợ nested objects và arrays, phù hợp với dữ liệu phức tạp
- **Horizontal Scaling:** Hỗ trợ sharding để phân tán dữ liệu trên nhiều server
- **ACID Transactions:** Đảm bảo tính nhất quán dữ liệu trong các thao tác phức tạp
- **Aggregation Pipeline:** Xử lý và phân tích dữ liệu mạnh mẽ

### **Ứng dụng trong Honey Social:**

- **Collections chính:** Users (người dùng), Posts (bài viết), Comments (bình luận), Messages (tin nhắn), Reports (báo cáo), Notifications (thông báo)
- **Schema linh hoạt:** Trường postVector trong schema Post lưu vector nhúng từ OpenAI API để gợi ý bài viết thông minh
- **Quan hệ dữ liệu:** Sử dụng ObjectId để liên kết giữa các collections, populate để join dữ liệu

### **Tối ưu hóa hiệu suất:**

- **Indexing Strategy:**
  - + Compound index trên {userId: 1, createdAt: -1} cho API GetFeedPosts
  - + Text index trên content cho tính năng tìm kiếm bài viết
  - + Sparse index trên postVector cho vector search
- **Connection Pooling:** Sử dụng mongoose với maxPoolSize = 10 để tối ưu kết nối
- **MongoDB Atlas:** Triển khai trên AWS với auto-scaling và backup tự động

#### **2.3.2. Express.js**

**Khái niệm và kiến trúc:** Express.js là web framework tối giản và linh hoạt cho Node.js, xây dựng dựa trên middleware pattern. Framework này cung cấp các tính năng mạnh mẽ để phát triển ứng dụng web và API một cách nhanh chóng.

#### **Middleware Architecture:**

- **Built-in Middleware:** express.json(), express.static(), express.urlencoded()
- **Third-party Middleware:**

- + cors: Cross-Origin Resource Sharing
- + helmet: Security headers
- + morgan: HTTP request logging
- + express-rate-limit: Rate limiting
- **Custom Middleware:** Authentication, error handling, request validation

### **RESTful API Design:**

- **Resource-based URLs:** /api/users/:id, /api/posts/:postId/comments
- **HTTP Methods:** GET (lấy dữ liệu), POST (tạo mới), PUT (cập nhật), DELETE (xóa)
- **Status Codes:** 200 (thành công), 201 (tạo mới), 400 (lỗi client), 401 (chưa xác thực), 500 (lỗi server)

### **Ứng dụng trong Honey Social:**

- **Route Handlers:** Xử lý đăng ký/đăng nhập, đăng bài viết, tương tác (like, comment), chat thời gian thực
- **Middleware JWT:** Xác thực người dùng và phân quyền truy cập API
- **Error Handling:** Centralized error handling với custom error classes
- **Validation:** Sử dụng joi hoặc express-validator để validate input

#### **2.3.3. React.js**

**Khái niệm và kiến trúc:** React.js là thư viện JavaScript để xây dựng giao diện người dùng, sử dụng component-based architecture và Virtual DOM để tối ưu hiệu suất rendering.

#### **Core Concepts:**

- **Virtual DOM:** Representation của DOM trong memory, cho phép efficient updates
- **Component Lifecycle:** Mounting, updating, unmounting phases
- **Unidirectional Data Flow:** Dữ liệu chảy từ parent xuống child components
- **React Hooks:** useState, useEffect, useContext, useReducer

### **Component Architecture:**

- **Functional Components:** Sử dụng hooks thay vì class components
- **Higher-Order Components (HOC):** Tái sử dụng logic giữa các components
- **Custom Hooks:** Tách logic phức tạp thành hooks có thể tái sử dụng

### **Ứng dụng trong Honey Social:**

- **UI Components:** Header, Sidebar, PostCard, CommentSection, ChatBox
- **Pages:** Home Feed, Profile, Messages, Admin Dashboard
- **State Management:** Context API cho global state, local state cho component-specific data
- **Routing:** React Router cho SPA navigation
- **Responsive Design:** CSS Modules và Material-UI cho giao diện adaptive

### **Performance Optimization:**

- **React.memo:** Ngăn re-render không cần thiết
- **useMemo & useCallback:** Memoization cho expensive calculations
- **Code Splitting:** Lazy loading với React.lazy() và Suspense
- **Bundle Optimization:** Webpack optimization cho production build

#### **2.3.4. Node.js**

**Khái niệm và kiến trúc:** Node.js là runtime environment cho phép chạy JavaScript trên server-side, sử dụng V8 JavaScript engine của Google Chrome và mô hình event-driven, non-blocking I/O.

### **Event Loop Architecture:**

- **Single-threaded:** Main thread xử lý JavaScript code
- **Event Loop:** Quản lý callbacks và async operations
- **Thread Pool:** Libuv thread pool cho I/O operations
- **Callback Queue:** Queue các callback functions để execute

### **Core Modules:**

- **HTTP/HTTPS**: Tạo web servers
- **File System (fs)**: Thao tác với files
- **Path**: Xử lý đường dẫn files
- **Crypto**: Mã hóa và hash
- **Events**: Event emitter pattern

### **Ứng dụng trong Honey Social:**

- **Backend Server**: Vận hành Express.js server
- **Database Integration**: Kết nối và thao tác với MongoDB
- **Third-party APIs**: Tích hợp OpenAI API, Cloudinary, Elasticsearch
- **Real-time Features**: Socket.IO cho chat và notifications
- **Background Jobs**: Xử lý email, image processing, data analytics

## **2.4. Công nghệ hỗ trợ**

### **2.4.1. JWT (JSON Web Token)**

**Khái niệm và cấu trúc:** JWT là chuẩn mở (RFC 7519) để truyền thông tin an toàn giữa các bên dưới dạng JSON object. Token bao gồm ba phần được mã hóa base64 và ngăn cách bởi dấu chấm.

#### **Cấu trúc Token:**

- **Header**: Chứa loại token (JWT) và thuật toán mã hóa (HS256, RS256)
- **Payload**: Claims (thông tin user, permissions, expiry time)
- **Signature**: Đảm bảo token không bị thay đổi, tạo từ header + payload + secret

#### **Security Considerations:**

- **Token Expiry**: Access token (1 giờ), Refresh token (7 ngày)
- **Secret Management**: Sử dụng environment variables, rotation định kỳ
- **Token Storage**: HttpOnly cookies vs localStorage trade-offs
- **CSRF Protection**: SameSite cookie attribute

### **Ứng dụng trong Honey Social:**

- **Authentication Flow:** Login → Generate JWT → Store in httpOnly cookie
- **Authorization:** Middleware kiểm tra JWT trong header Authorization
- **User Context:** Decode JWT để lấy userId cho API calls
- **Session Management:** Refresh token mechanism cho long-term sessions

#### **2.4.2. WebSocket và Socket.IO**

**WebSocket Protocol:** WebSocket là giao thức giao tiếp full-duplex qua single TCP connection, cho phép real-time communication giữa client và server.

##### **Protocol Features:**

- **Handshake:** HTTP upgrade request để chuyển sang WebSocket
- **Frame Structure:** Text frames, binary frames, control frames
- **Connection States:** Connecting, open, closing, closed
- **Low Latency:** Giảm overhead so với HTTP polling

##### **Socket.IO Enhancements:**

- **Fallback Mechanisms:** Long-polling khi WebSocket không khả dụng
- **Room Management:** Grouping connections cho targeted messaging
- **Event-based Communication:** Custom events với structured data
- **Auto-reconnection:** Tự động kết nối lại khi mất connection

### **Ứng dụng trong Honey Social:**

- **Real-time Chat:** Tin nhắn tức thời giữa users
- **Live Notifications:** Thông báo comment, like, follow ngay lập tức
- **Online Status:** Hiển thị trạng thái online/offline của users
- **Namespace Organization:** /chat, /notifications cho separation of concerns

### 2.4.3. Cloudinary

**Khái niệm và kiến trúc:** Cloudinary là cloud-based service chuyên về quản lý, tối ưu hóa và phân phối media assets (images, videos) qua global CDN network.

#### Image Processing Pipeline:

- **Upload:** Direct upload từ client hoặc server-side với signed URLs
- **Transformation:** Real-time resize, crop, format conversion, quality optimization
- **Delivery:** CDN distribution với edge caching và geo-optimization
- **Storage:** Backup và version control cho media assets

#### Optimization Techniques:

- **Responsive Images:** Dynamic resizing dựa trên device và viewport
- **Format Optimization:** Auto-convert sang WebP/AVIF cho browsers hỗ trợ
- **Progressive Loading:** Progressive JPEG và lazy loading
- **Compression:** Intelligent quality adjustment để giảm file size

#### Ứng dụng trong Honey Social:

- **Media Storage:** Ảnh bài viết, avatar, cover photos, chat images
- **Auto-optimization:** Tự động nén và convert format
- **Transformation Presets:** Avatar cropping (200x200), post images (max 1200px width)
- **CDN Delivery:** Fast loading từ nearest edge server

### 2.4.4. Elasticsearch và Vector Search

**Elasticsearch Architecture:** Elasticsearch là distributed search engine xây dựng trên Apache Lucene, hỗ trợ full-text search và analytics.

#### Core Concepts:

- **Inverted Index:** Mapping từ terms đến documents chứa terms đó
- **Sharding:** Phân tán data trên multiple nodes
- **Replication:** Backup data cho high availability

- **Cluster:** Tập hợp các nodes working together

### **Vector Search (KNN):**

- **Dense Vectors:** High-dimensional representations từ machine learning models
- **Similarity Metrics:** Cosine similarity, dot product, euclidean distance
- **HNSW Algorithm:** Hierarchical Navigable Small World graphs cho fast approximate search
- **Vector Indexing:** Optimized index structures cho vector search

### **Ứng dụng trong Honey Social:**

- **Content Recommendation:** Gợi ý bài viết dựa trên semantic similarity
- **Vector Storage:** Lưu OpenAI embeddings trong postVector field
- **KNN Search:** Tìm bài viết tương đồng với cosine similarity
- **Performance:** Inference Processor giảm query time xuống 50ms

#### **2.4.5. Caching Strategy**

**Cache-Aside Pattern:** Application quản lý cache manually, đọc từ cache trước, nếu cache miss thì query database và update cache.

### **Redis Data Structures:**

- **Strings:** Simple key-value pairs cho session data
- **Hashes:** Object-like structures cho user profiles
- **Lists:** Ordered collections cho recent activities
- **Sets:** Unordered unique collections cho followers/following
- **Sorted Sets:** Leaderboards và ranking features

### **Cache Optimization:**

- **TTL Management:** Expire keys tự động (feed cache: 5 phút)
- **Cache Invalidation:** Xóa cache khi có data updates
- **Cache Warming:** Pre-load popular content vào cache

- **Memory Management:** Eviction policies (LRU, LFU)

#### **Ứng dụng trong Honey Social:**

- **Feed Caching:** Cache user feeds để giảm database queries
- **Session Storage:** Store user sessions trong Redis
- **Rate Limiting:** Track API calls per user/IP
- **Popular Content:** Cache trending posts và popular hashtags

#### **2.4.6. Lazy Loading**

**Khái niệm và implementation:** Lazy loading là kỹ thuật defer loading của resources cho đến khi thực sự cần thiết, giảm initial load time và bandwidth usage.

##### **Implementation Techniques:**

- **Intersection Observer API:** Detect khi elements enter viewport
- **Pagination:** Load content theo chunks thay vì load all
- **Virtual Scrolling:** Render chỉ visible items trong long lists
- **Image Lazy Loading:** Load images khi scroll đến

#### **Ứng dụng trong Honey Social:**

- **Feed Posts:** Load 10 posts mỗi lần, load thêm khi scroll
- **Comment Loading:** Load comments on-demand khi expand
- **Image Loading:** Progressive image loading với placeholders
- **User Lists:** Lazy load followers/following lists

### **2.5. Architecture Patterns**

#### **2.5.1. Monolithic vs Microservices**

##### **Current Monolithic Approach:**

- **Single Deployment Unit:** Toàn bộ application trong một codebase
- **Shared Database:** Tất cả modules sử dụng chung MongoDB instance
- **Internal Communication:** Function calls và shared memory

- **Pros:** Đơn giản deploy, testing, debugging
- **Cons:** Scaling limitations, technology lock-in

### Potential Microservices Evolution:

- **User Service:** Authentication, profiles, relationships
- **Content Service:** Posts, comments, media handling
- **Notification Service:** Real-time notifications, email, push
- **Search Service:** Elasticsearch integration, recommendations
- **Analytics Service:** User behavior tracking, metrics

#### 2.5.2. Event-Driven Architecture

**Message Queue Integration:** Sử dụng RabbitMQ cho asynchronous communication và event processing.

##### Event Types:

- **Domain Events:** UserRegistered, PostCreated, CommentAdded
- **Integration Events:** Cross-service communication
- **System Events:** ErrorOccurred, PerformanceMetric

##### Benefits:

- **Loose Coupling:** Services không phụ thuộc trực tiếp
- **Scalability:** Scale individual event processors
- **Reliability:** Message persistence và retry mechanisms
- **Auditability:** Event sourcing cho complete audit trail

#### 2.6. Tổng kết chương

Chương này đã trình bày cơ sở lý thuyết chi tiết của các công nghệ sử dụng trong hệ thống Honey Social. MERN Stack cung cấp foundation mạnh mẽ với MongoDB cho flexible data storage, Express.js cho robust API development, React.js cho interactive UI, và Node.js cho high-performance backend. Các công nghệ hỗ trợ như JWT, Socket.IO, Cloudinary, Elasticsearch, và Redis giải quyết các yêu cầu chuyên biệt về security, real-time communication, media

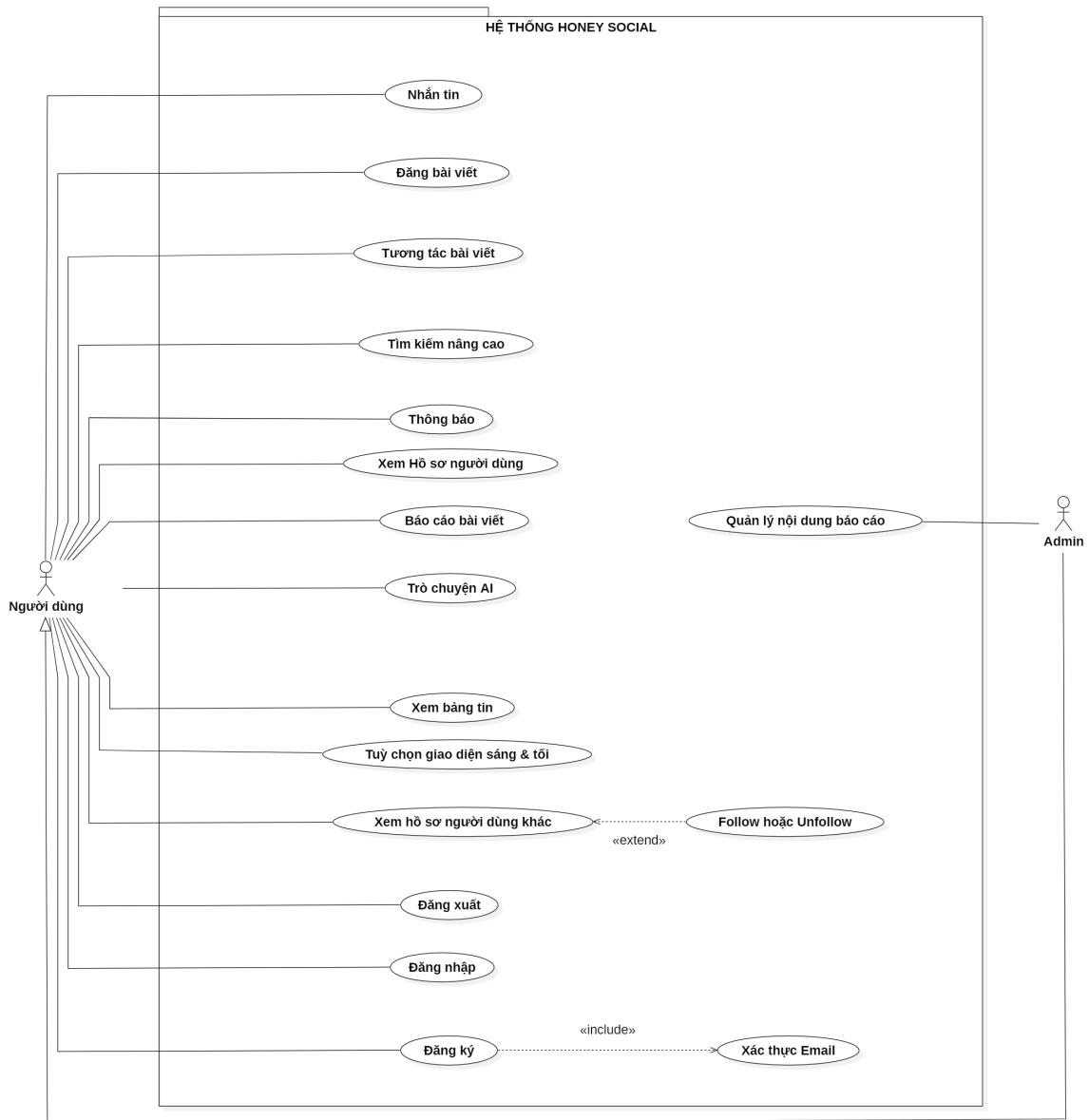
management, intelligent search, và performance optimization. Architecture patterns được áp dụng đảm bảo hệ thống có thể mở rộng và maintainable. Những kiến thức lý thuyết này tạo nền tảng cho việc triển khai thực tế được mô tả trong chương tiếp theo.

## Chương 3. Phân tích thiết kế hệ thống

### 3.1. Phân Tích Hệ Thống

#### 3.1.1. Biểu đồ Usecase

##### Use Case Tổng quát



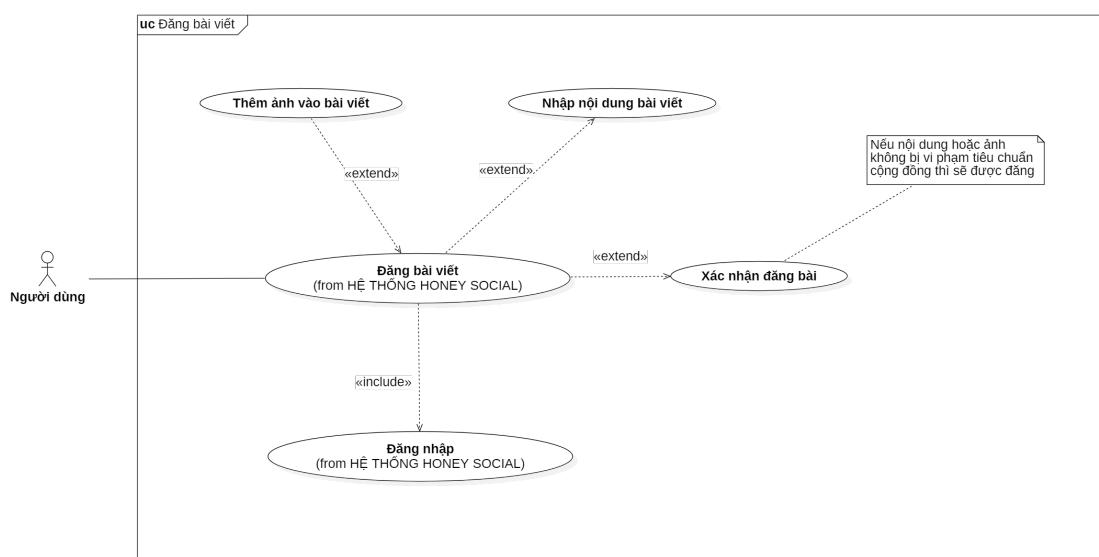
Hình 1. Hình ảnh Use Case Tổng quát

Các chức năng được bao bì bởi hệ thống **HỆ THỐNG HONEY SOCIAL** và là những hành động mà người dùng có thể thực hiện như sau:

- Nhắn tin
- Đăng bài viết

- Tương tác bài viết: Thích, bình luận, chia sẻ,...
- Tìm kiếm nâng cao
- Thông báo
- Xem hồ sơ người dùng
- Báo cáo bài viết
- Trò chuyện AI
- Xem bảng tin
- Tùy chọn giao diện sáng & tối
- Đăng nhập
- Đăng xuất

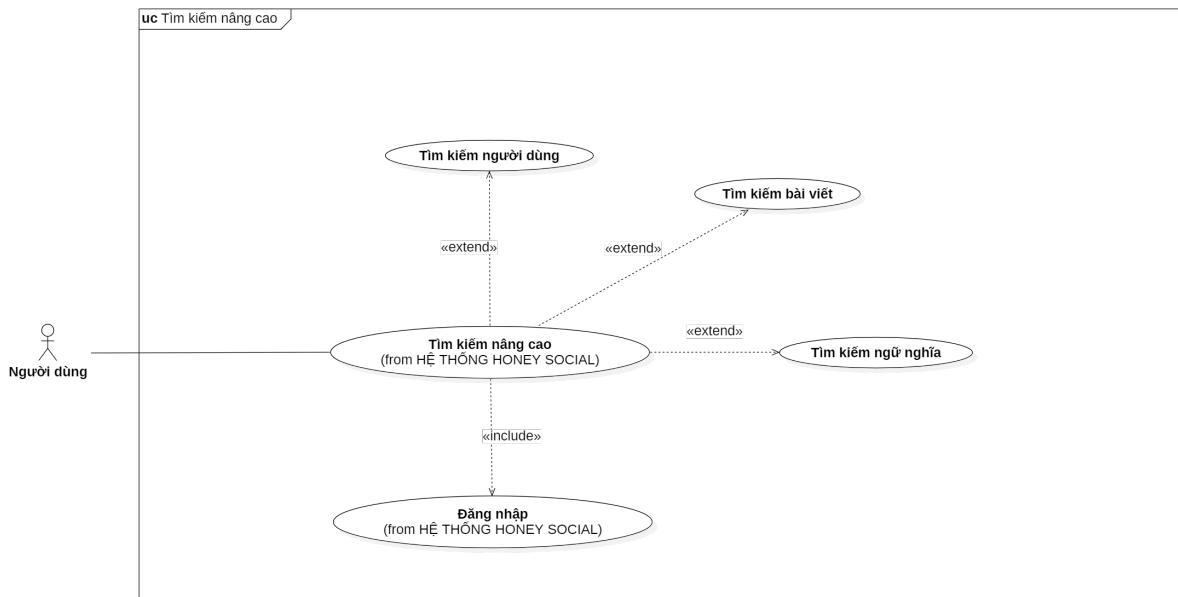
## Đăng bài viết



Hình 2. Hình ảnh Đăng bài viết

Mô tả chi tiết quy trình đăng bài, bắt đầu từ việc người dùng nhập nội dung văn bản, tải lên ảnh hoặc video, chỉnh sửa trước khi đăng (bao gồm thêm hashtag hoặc thẻ người), trải qua kiểm duyệt tự động bằng AI, và cuối cùng là xác nhận để bài viết được công khai trên bảng tin.

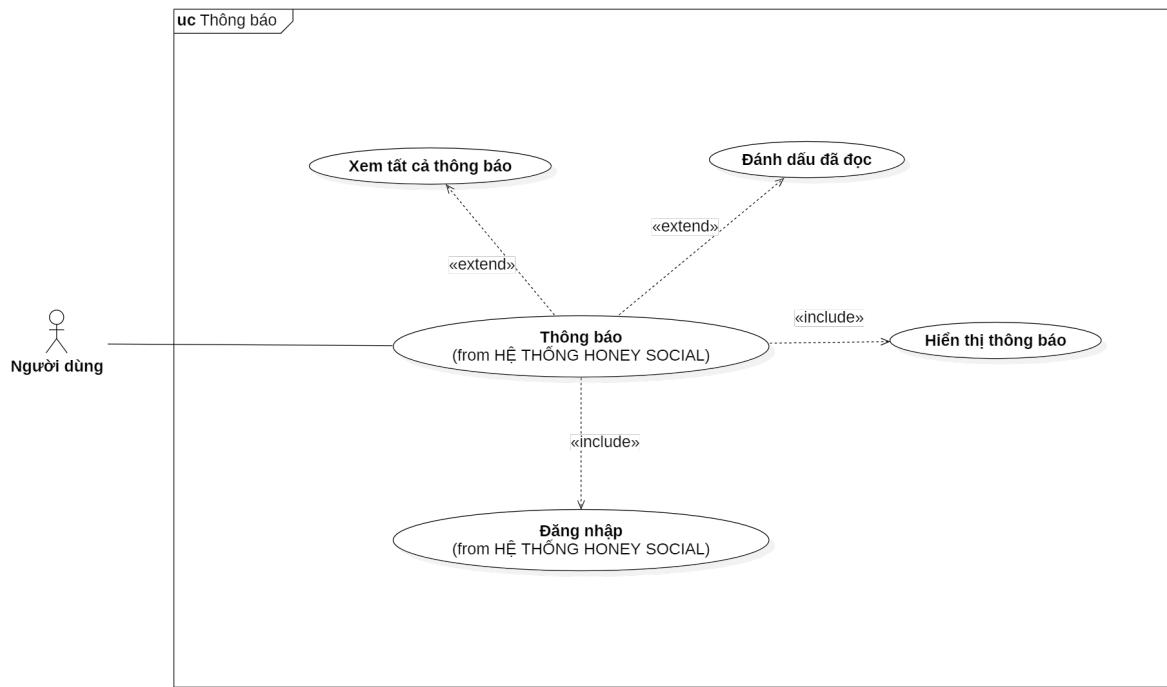
## Tìm kiếm nâng cao



Hình 3. Hình ảnh Tìm kiếm nâng cao

Hiển thị quy trình tìm kiếm thông minh, cho phép người dùng sử dụng bộ lọc theo từ khóa, thời gian, hoặc sở thích cá nhân, tích hợp gợi ý dựa trên vector similarity từ Elasticsearch, và sắp xếp kết quả theo mức độ liên quan hoặc thời gian đăng tải để tối ưu hóa trải nghiệm tìm kiếm.

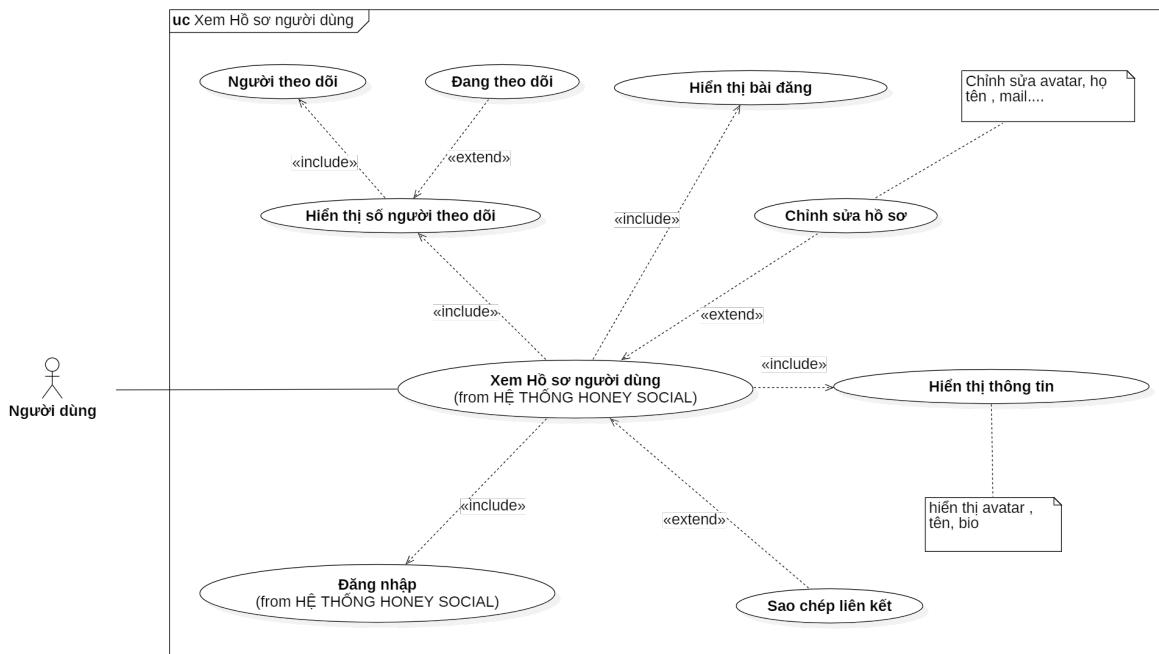
## Thông báo



Hình 4. Hình ảnh Thông báo

Chi tiết hóa cách hệ thống gửi thông báo thời gian thực, bao gồm thông báo khi có tương tác (thích, bình luận), cập nhật từ người theo dõi, hoặc thông báo hệ thống (như xác nhận email), với tùy chọn tắt/mở thông báo để cá nhân hóa trải nghiệm.

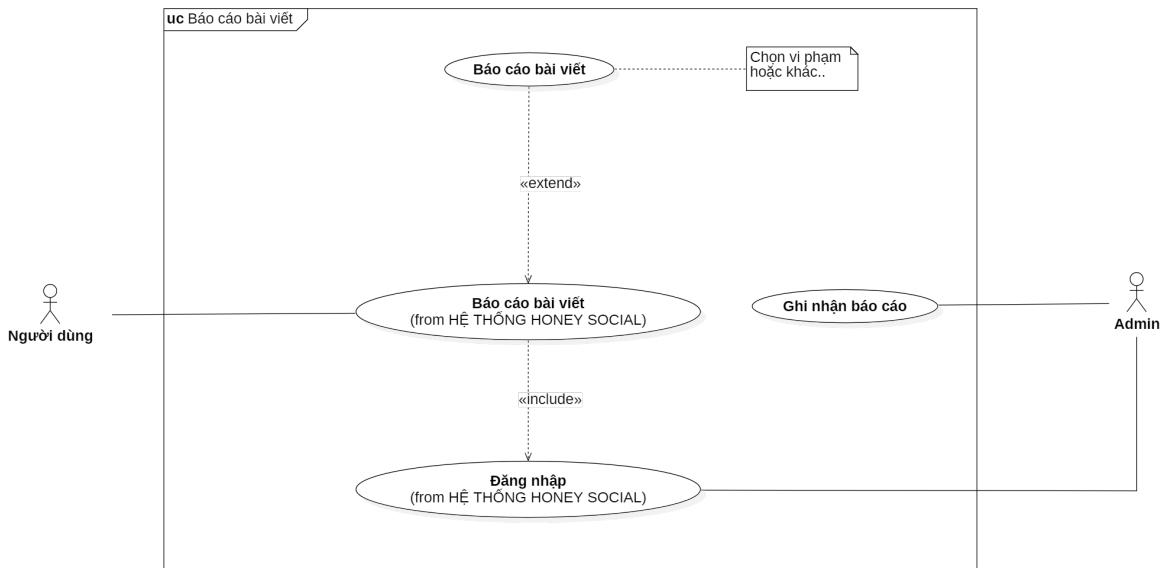
## Xem hồ sơ người dùng



Hình 5. Hình ảnh Xem Hồ sơ người dùng

Hướng dẫn chi tiết cách người dùng xem và chỉnh sửa hồ sơ cá nhân, bao gồm cập nhật avatar thông qua Cloudinary, chỉnh sửa thông tin như tên, bio, và danh sách theo dõi/người theo dõi, cùng với các thống kê cơ bản như số bài đăng.

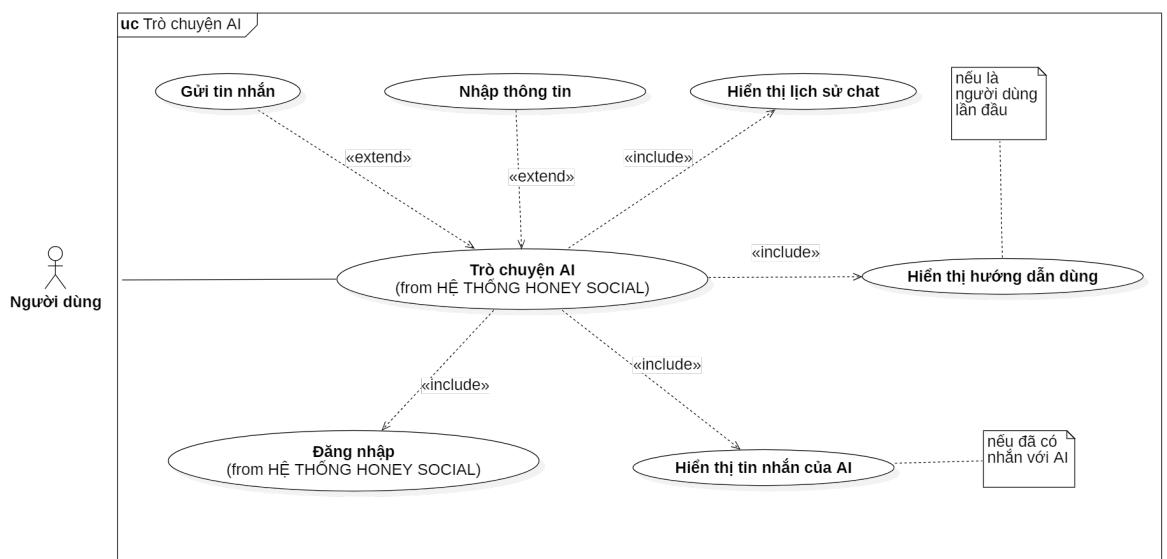
## Báo cáo bài viết



Hình 6. Hình ảnh Báo cáo bài viết

Mô tả quy trình người dùng báo cáo vi phạm, bao gồm chọn lý do cụ thể (nội dung không phù hợp, bạo lực, spam...), gửi thông tin kèm bằng chứng qua API, và chuyển đến quản trị viên để xử lý theo các cấp độ (nhẹ, vừa, nặng).

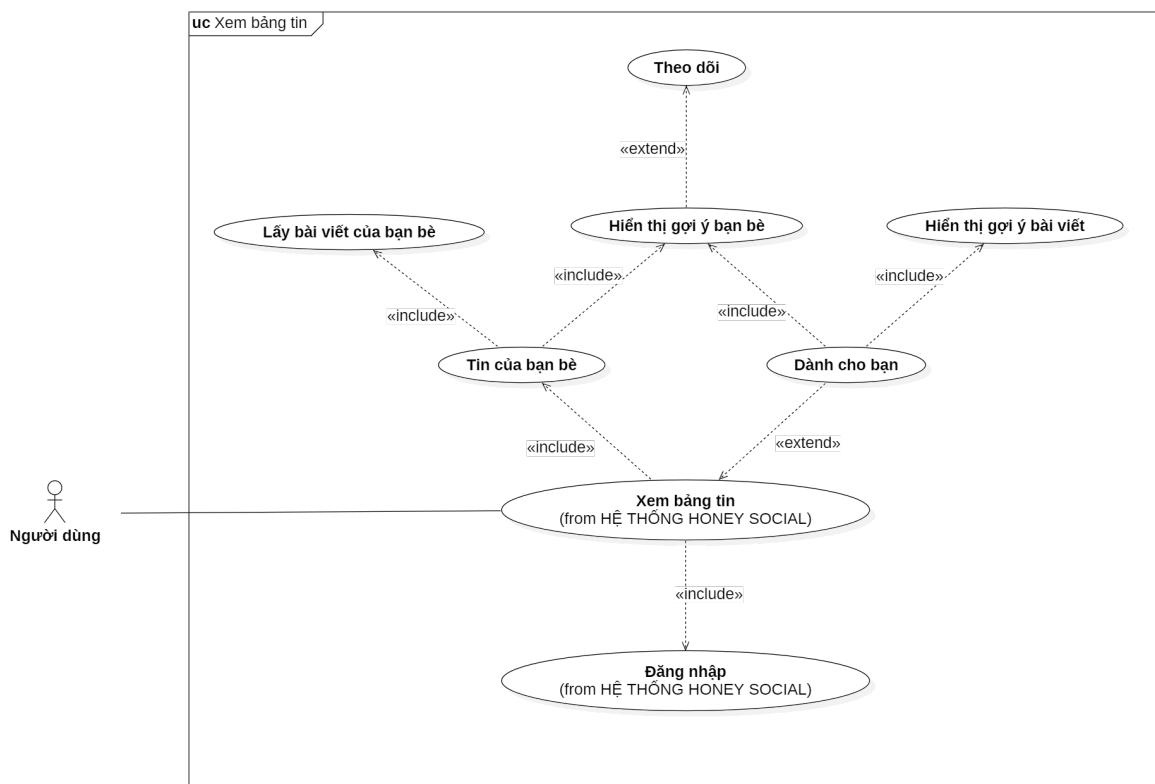
### Trò chuyện AI



Hình 7. Hình ảnh Trò chuyện AI

Phác thảo chức năng chat AI, hỗ trợ trả lời câu hỏi liên quan đến nội dung bài viết, gợi ý bài viết dựa trên sở thích, và tự động phát hiện/phân loại báo cáo vi phạm, với giao diện thân thiện và phản hồi thời gian thực.

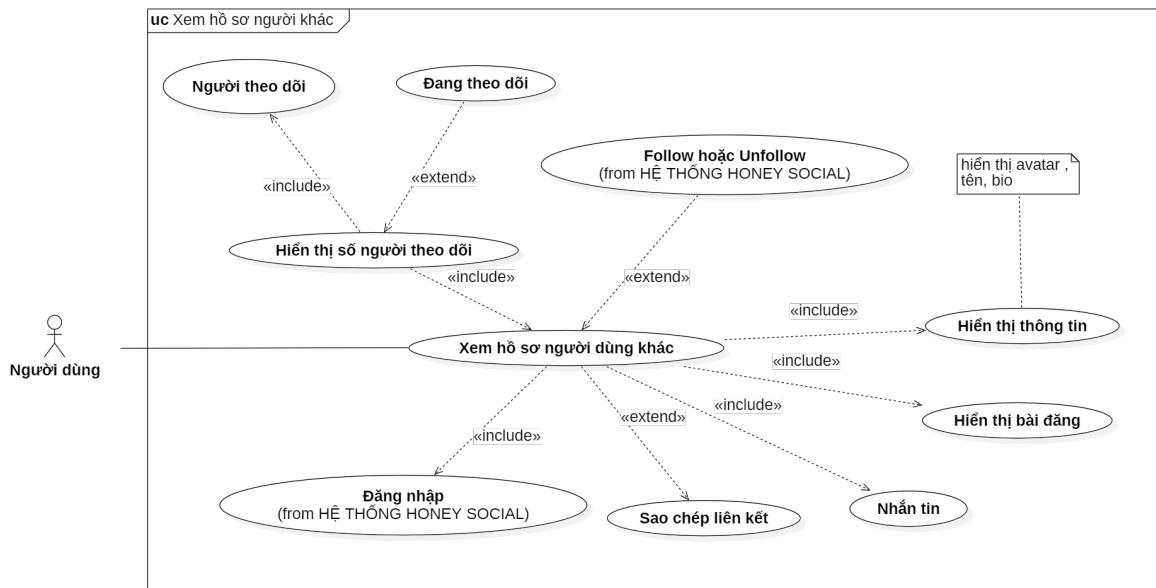
## Xem bảng tin



Hình 8. Hình ảnh Xem bảng tin

Thể hiện cách hệ thống hiển thị feed bài viết cá nhân hóa, lấy dữ liệu từ người dùng theo dõi, tích hợp gợi ý từ AI dựa trên vector nhúng, và cho phép làm mới feed hoặc lọc theo chủ đề để tăng tính tương tác.

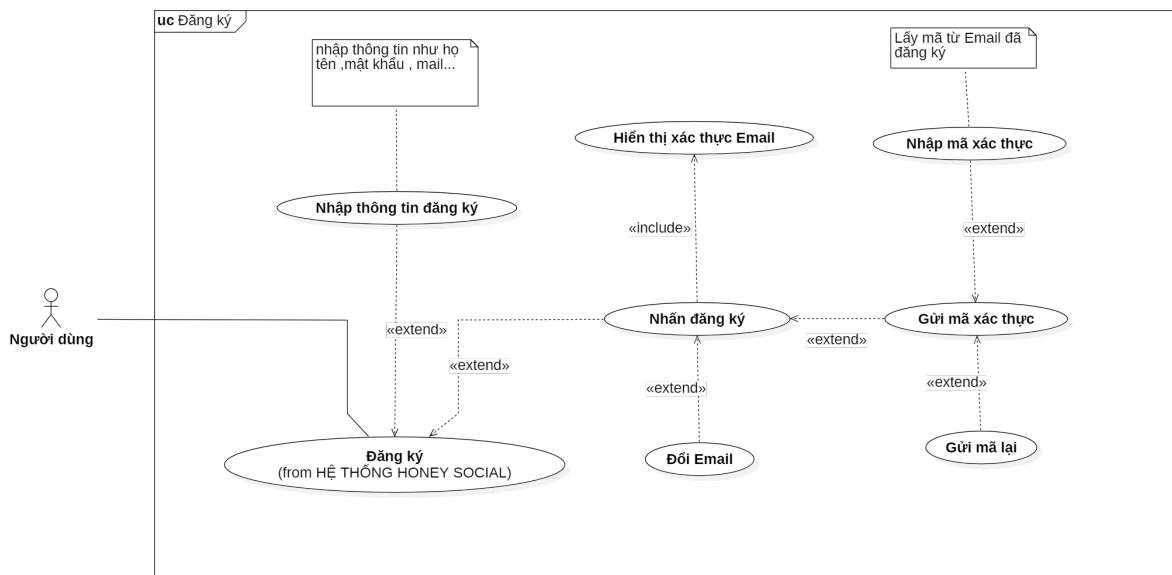
## Xem hồ sơ người khác



Hình 9. Hình ảnh Xem hồ sơ người khác

Chi tiết quy trình xem hồ sơ người dùng khác, bao gồm thông tin cơ bản (tên, avatar, bio), danh sách bài đăng công khai, và các tùy chọn tương tác như theo dõi hoặc gửi tin nhắn.

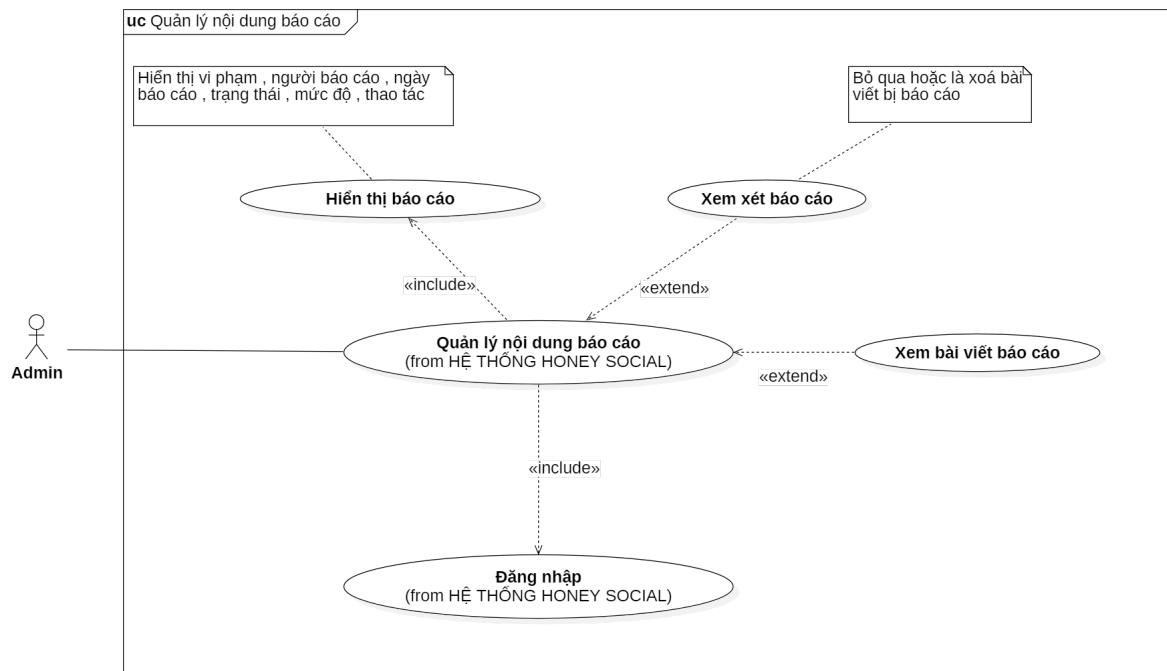
## Đăng ký



Hình 10. Hình ảnh Đăng ký

Mô tả quy trình đăng ký tài khoản, yêu cầu nhập thông tin (email, mật khẩu), gửi mã OTP qua Resend API để xác thực, và hoàn tất với tùy chọn cá nhân hóa hồ sơ ngay sau đăng ký.

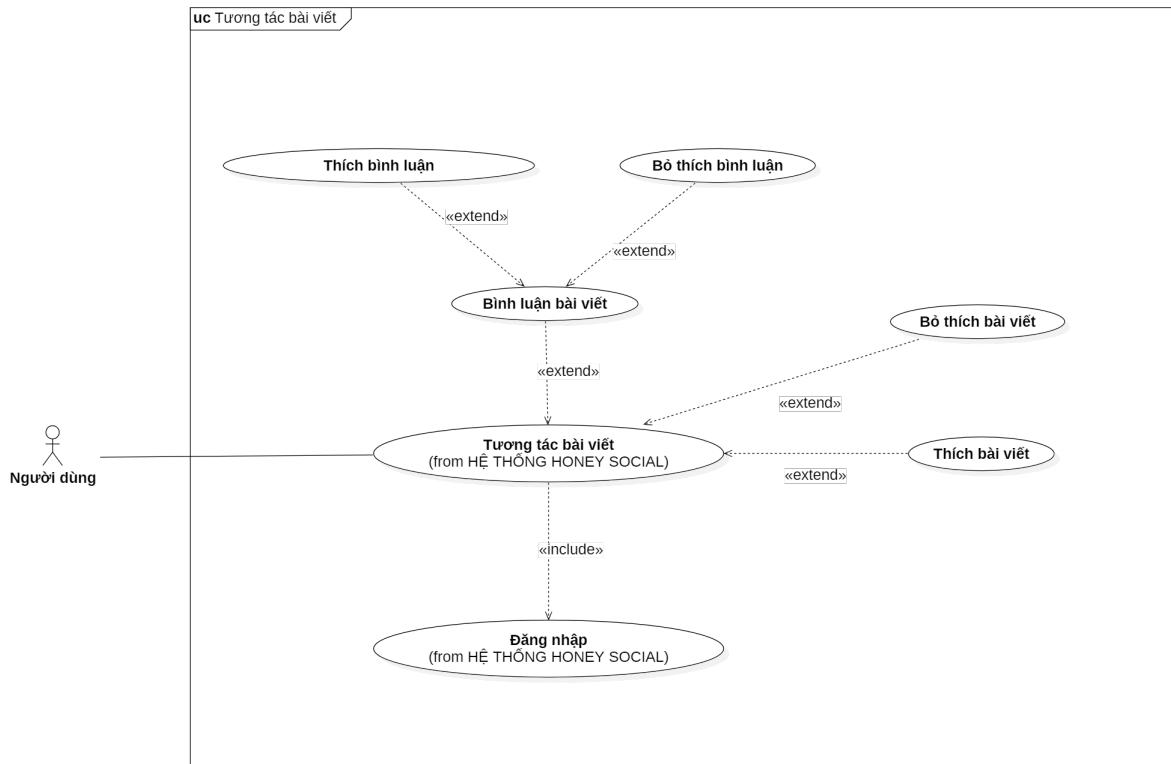
## Quản lý nội dung báo cáo



Hình 11. Hình ảnh Quản lý nội dung báo cáo

Hiển thị cách quản trị viên tiếp nhận báo cáo từ người dùng, phân loại theo mức độ vi phạm (nhẹ: cảnh báo; vừa: ẩn bài; nặng: xóa và khóa tài khoản), và xử lý qua dashboard admin với các công cụ lọc theo thời gian hoặc lý do.

## Tương tác bài viết



Hình 12. Hình ảnh Tương tác bài viết

Phác thảo các hành động tương tác như thích (like), bình luận (với khả năng phản hồi), và chia sẻ liên kết bài viết ra ngoài, tất cả đều được xử lý qua Socket.IO để đảm bảo cập nhật thời gian thực.

### 3.1.2. Kịch bản của các UseCase

Bảng 1. Bảng thông tin hoạt động của chức năng đăng bài viết

<b>Use-case name</b>	Đăng bài viết
<b>Description</b>	Người dùng đăng một bài viết mới lên hệ thống, có thể kèm hình ảnh.
<b>Actor</b>	Người dùng đã đăng nhập
<b>Pre-Conditions</b>	-Người dùng đã đăng nhập vào hệ thống. -Email của người dùng đã được xác thực (nếu hệ thống yêu cầu).
<b>Post-Condition</b>	-Bài viết mới được lưu vào hệ thống và hiển thị trên bảng tin. -Nếu có lỗi, thông báo lỗi được hiển thị cho người dùng.
<b>Trigger</b>	Người dùng nhấn nút "Tạo bài viết mới" hoặc biểu tượng "+".
<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng nhấn nút "Tạo bài viết mới".</li> <li>2. Hệ thống hiển thị modal nhập nội dung bài viết.</li> <li>3. Người dùng nhập nội dung, chọn hình ảnh (nếu muốn).</li> <li>4. Người dùng nhấn nút "Đăng".</li> <li>5. Hệ thống kiểm tra điều kiện (đăng nhập, xác thực email, nội dung hợp lệ).</li> <li>6. Nếu hợp lệ, bài viết được lưu và hiển thị; thông báo thành công cho người dùng.</li> </ol>

<b>Alternative Flow</b>	<p>Nếu người dùng chưa đăng nhập: Hệ thống thông báo và chuyển hướng đến trang đăng nhập.</p> <p>Nếu email chưa xác thực: Hệ thống thông báo yêu cầu xác thực email.</p> <p>Nếu nội dung vi phạm hoặc lỗi khác: Hệ thống hiển thị thông báo lỗi chi tiết.</p> <p>Nếu người dùng nhấn "Hủy": Modal đóng, không lưu bài viết.</p>
-------------------------	---

Bảng 2. Bảng thông tin hoạt động của chức năng tìm kiếm nâng cao

<b>Use-case name</b>	Tìm kiếm nâng cao
<b>Description</b>	Người dùng thực hiện tìm kiếm nâng cao để tìm kiếm người dùng, bài viết hoặc theo ngữ nghĩa trên hệ thống.
<b>Actor</b>	Người dùng đã đăng nhập
<b>Pre-Conditions</b>	Người dùng đã đăng nhập vào hệ thống. Hệ thống có dữ liệu người dùng/bài viết.
<b>Post-Condition</b>	Kết quả tìm kiếm được hiển thị cho người dùng theo tiêu chí đã chọn. Người dùng có thể chuyển trang, sắp xếp hoặc chọn kết quả để xem chi tiết.
<b>Trigger</b>	Người dùng truy cập trang tìm kiếm nâng cao hoặc nhập từ khóa vào ô tìm kiếm.

<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng truy cập chức năng tìm kiếm nâng cao.</li> <li>2. Người dùng nhập từ khóa tìm kiếm.</li> <li>3. Hệ thống gửi yêu cầu tìm kiếm đến server với từ khóa và các tham số (phân trang, sắp xếp).</li> <li>4. Server xử lý và trả về danh sách kết quả phù hợp.</li> <li>5. Hệ thống hiển thị kết quả tìm kiếm cho người dùng, kèm số lượng và các tùy chọn sắp xếp.</li> <li>6. Người dùng có thể chuyển trang, thay đổi tiêu chí sắp xếp hoặc nhấn vào kết quả để xem chi tiết.</li> </ol>
<b>Alternative Flow</b>	<p>Nếu người dùng chưa đăng nhập: Hệ thống yêu cầu đăng nhập trước khi sử dụng tìm kiếm nâng cao.</p> <p>Nếu không có kết quả phù hợp: Hệ thống hiển thị thông báo "Không tìm thấy người dùng phù hợp".</p> <p>Nếu xảy ra lỗi server hoặc kết nối: Hệ thống hiển thị thông báo lỗi cho người dùng.</p>

Bảng 3. Bảng thông tin hoạt động của chức năng thông báo

<b>Use-case name</b>	Thông báo
<b>Description</b>	Người dùng xem danh sách thông báo, đánh dấu thông báo đã đọc hoặc xem chi tiết thông báo.
<b>Actor</b>	Người dùng đã đăng nhập
<b>Pre-Conditions</b>	Người dùng đã đăng nhập vào hệ thống. Hệ thống có thông báo liên quan đến người dùng.
<b>Post-Condition</b>	Thông báo được hiển thị cho người dùng. Thông báo được đánh dấu là đã đọc (nếu người dùng thực hiện hành động này).

<b>Trigger</b>	Người dùng nhấn vào biểu tượng thông báo trên giao diện.
<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng nhấn vào biểu tượng thông báo.</li> <li>2. Hệ thống hiển thị danh sách thông báo.</li> <li>3. Người dùng chọn một thông báo để xem chi tiết.</li> <li>4. Hệ thống đánh dấu thông báo đã đọc và hiển thị nội dung chi tiết.</li> <li>5. Người dùng quay lại danh sách thông báo hoặc tiếp tục sử dụng hệ thống.</li> </ol>
<b>Alternative Flow</b>	<p>Nếu không có thông báo: Hệ thống hiển thị thông báo "Không có thông báo mới".</p> <p>Nếu người dùng chưa đăng nhập: Hệ thống yêu cầu người dùng đăng nhập để xem thông báo.</p>

Bảng 4. Bảng thông tin hoạt động của chức năng xem hồ sơ người dùng

<b>Use-case name</b>	Xem hồ sơ người dùng
<b>Description</b>	Người dùng có thể xem hồ sơ của bản thân hoặc người dùng khác, bao gồm thông tin cá nhân, danh sách bài đăng, số lượng người theo dõi và đang theo dõi.
<b>Actor</b>	Người dùng đã đăng nhập
<b>Pre-Conditions</b>	<ul style="list-style-type: none"> <li>– Người dùng đã đăng nhập vào hệ thống.</li> <li>– Hồ sơ của người dùng hoặc người khác tồn tại trong hệ thống.</li> </ul>
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>– Hồ sơ người dùng được hiển thị đầy đủ thông tin.</li> <li>– Người dùng có thể thực hiện các hành động như theo dõi, nhắn tin, hoặc chỉnh sửa hồ sơ (nếu là hồ sơ của chính họ).</li> </ul>

<b>Trigger</b>	Người dùng nhấn vào tên hoặc ảnh đại diện của một người dùng khác hoặc chọn mục "Hồ sơ" từ menu cá nhân.
<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng nhấn vào tên hoặc ảnh đại diện của một người dùng khác hoặc chọn mục "Hồ sơ".</li> <li>2. Hệ thống kiểm tra quyền truy cập và tải thông tin hồ sơ từ cơ sở dữ liệu.</li> <li>3. Hệ thống hiển thị thông tin cá nhân (tên, ảnh đại diện, tiểu sử, v.v.).</li> <li>4. Hệ thống hiển thị danh sách bài đăng của người dùng.</li> <li>5. Hệ thống hiển thị số lượng người theo dõi và đang theo dõi.</li> <li>6. Người dùng có thể thực hiện các hành động như theo dõi, nhắn tin, hoặc chỉnh sửa hồ sơ (nếu là hồ sơ của chính họ).</li> </ol>
<b>Alternative Flow</b>	<ul style="list-style-type: none"> <li>– Nếu người dùng chưa đăng nhập: Hệ thống yêu cầu đăng nhập trước khi xem hồ sơ.</li> <li>– Nếu hồ sơ không tồn tại: Hệ thống hiển thị thông báo lỗi "Hồ sơ không tồn tại".</li> <li>– Nếu xảy ra lỗi kết nối: Hệ thống hiển thị thông báo lỗi và yêu cầu thử lại.</li> </ul>

Bảng 5. Bảng thông tin hoạt động của chức năng báo cáo bài viết

<b>Use-case name</b>	Báo cáo bài viết
<b>Description</b>	Người dùng báo cáo một bài viết vi phạm các quy định của cộng đồng.
<b>Actor</b>	Người dùng đã đăng nhập

<b>Pre-Conditions</b>	<ul style="list-style-type: none"> <li>– Người dùng đã đăng nhập.</li> <li>– Bài viết tồn tại và được hiển thị trên giao diện.</li> </ul>
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>– Báo cáo được ghi nhận và gửi đến bộ phận kiểm duyệt.</li> <li>– Bài viết có thể bị ẩn khỏi người dùng báo cáo (tùy thuộc vào cài đặt).</li> </ul>
<b>Trigger</b>	Người dùng chọn tùy chọn "Báo cáo bài viết" từ menu tương tác của bài viết.
<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng nhấn vào menu “More options” của bài viết.</li> <li>2. Hệ thống hiển thị danh sách các tùy chọn tương tác.</li> <li>3. Người dùng chọn "Báo cáo bài viết".</li> <li>4. Hệ thống hiển thị hộp thoại báo cáo, cho phép chọn lý do báo cáo (ví dụ: nội dung khiêu dâm, bạo lực, ngôn từ gây thù ghét, v.v.).</li> <li>5. Người dùng có thể nhập thêm thông tin chi tiết về lý do báo cáo (nếu cần).</li> <li>6. Người dùng nhấn nút "Gửi báo cáo".</li> <li>7. Hệ thống gửi thông tin báo cáo đến server.</li> <li>8. Hệ thống nhận phản hồi từ server và hiển thị thông báo kết quả cho người dùng (thành công hoặc lỗi).</li> </ol>
<b>Alternative Flow</b>	<ul style="list-style-type: none"> <li>– Nếu người dùng chưa chọn lý do báo cáo: Hệ thống yêu cầu chọn một lý do trước khi gửi.</li> <li>– Nếu có lỗi kết nối hoặc server: Hệ thống hiển thị thông báo lỗi và yêu cầu thử lại.</li> </ul>

Bảng 6. Bảng thông tin hoạt động của chức năng trò chuyện AI

<b>Use-case name</b>	Trò chuyện AI
<b>Description</b>	Người dùng tương tác với trợ lý AI thông qua giao diện chat, có thể gửi tin nhắn, nhận phản hồi và xem lịch sử trò chuyện.
<b>Actor</b>	Người dùng đã đăng nhập
<b>Pre-Conditions</b>	<ul style="list-style-type: none"> <li>– Người dùng đã đăng nhập vào hệ thống.</li> <li>– Kết nối internet hoạt động bình thường.</li> <li>– Hệ thống AI đang hoạt động.</li> </ul>
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>– Tin nhắn của người dùng được gửi và lưu trữ.</li> <li>– AI phản hồi và hiển thị tin nhắn cho người dùng.</li> <li>– Lịch sử trò chuyện được cập nhật.</li> </ul>
<b>Trigger</b>	Người dùng truy cập tính năng Chat AI thông qua menu điều hướng hoặc biểu tượng AI Chat.

<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng nhấn vào biểu tượng Chat AI trên thanh điều hướng (desktop) hoặc chọn "Chat với AI" từ menu chat (mobile).</li> <li>2. Hệ thống chuyển hướng đến trang chat AI (/chat).</li> <li>3. Hệ thống hiển thị giao diện chat và hướng dẫn sử dụng (nếu là lần đầu truy cập).</li> <li>4. Người dùng nhập nội dung tin nhắn vào ô văn bản.</li> <li>5. Người dùng gửi tin nhắn bằng cách nhấn nút gửi hoặc phím Enter.</li> <li>6. Hệ thống hiển thị tin nhắn của người dùng trong cửa sổ chat và gửi yêu cầu đến API AI.</li> <li>7. AI xử lý yêu cầu và gửi phản hồi.</li> <li>8. Hệ thống hiển thị phản hồi của AI trong cửa sổ chat.</li> </ol>
<b>Alternative Flow</b>	<ul style="list-style-type: none"> <li>– Nếu người dùng chưa đăng nhập: Hệ thống yêu cầu người dùng đăng nhập trước khi sử dụng tính năng.</li> <li>– Nếu kết nối bị gián đoạn: Hệ thống hiển thị thông báo lỗi và lưu tin nhắn để gửi lại sau.</li> <li>– Nếu AI không thể xử lý yêu cầu: Hệ thống hiển thị thông báo lỗi phù hợp và gợi ý người dùng thử lại.</li> <li>– Nếu người dùng tải lại trang: Hệ thống tải lại lịch sử trò chuyện từ cơ sở dữ liệu.</li> </ul>

Bảng 7. Bảng thông tin hoạt động của chức năng xem bảng tin

<b>Use-case name</b>	Xem bảng tin
<b>Description</b>	Người dùng xem bảng tin chứa các bài viết từ bạn bè hoặc bài viết được gợi ý dựa trên sở thích và tương tác của họ.
<b>Actor</b>	Người dùng đã đăng nhập
<b>Pre-Conditions</b>	<ul style="list-style-type: none"> <li>– Người dùng đã đăng nhập vào hệ thống.</li> <li>– Kết nối internet hoạt động bình thường.</li> </ul>
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>– Người dùng xem được các bài viết trên bảng tin.</li> <li>– Hệ thống ghi nhận các tương tác của người dùng với bảng tin.</li> </ul>
<b>Trigger</b>	Người dùng truy cập trang chủ hoặc tính năng "Bảng tin".

<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng truy cập trang chủ của hệ thống.</li> <li>2. Hệ thống hiển thị giao diện bảng tin với hai tab: "Tin của bạn bè" và "Dành cho bạn".</li> <li>3. Mặc định, hệ thống hiển thị tab "Tin của bạn bè" với: <ul style="list-style-type: none"> <li>– Các gợi ý bạn bè (ngang) ở phần trên</li> <li>– Các bài viết từ những người mà người dùng đang theo dõi, sắp xếp theo thứ tự thời gian mới nhất</li> </ul> </li> <li>4. Người dùng cuộn xuống để xem thêm bài viết (lazy loading).</li> <li>5. Người dùng có thể chuyển sang tab "Dành cho bạn" để xem các bài viết được gợi ý dựa trên sở thích.</li> <li>6. Người dùng có thể tương tác với bài viết (like, comment, share) trực tiếp từ bảng tin.</li> </ol>
<b>Alternative Flow</b>	<ul style="list-style-type: none"> <li>– Nếu chưa đăng nhập: Hệ thống chuyển hướng người dùng đến trang đăng nhập.</li> <li>– Nếu không có bài viết nào từ bạn bè: Hiển thị gợi ý theo dõi thêm người dùng.</li> <li>– Nếu có lỗi kết nối: Hiển thị thông báo lỗi và tùy chọn tải lại.</li> <li>– Nếu người dùng theo dõi một người dùng mới từ phần gợi ý: Cập nhật bảng tin với bài viết mới từ người dùng đó.</li> </ul>

Bảng 8. Bảng thông tin hoạt động của chức năng xem hồ sơ người khác

<b>Use-case name</b>	Xem hồ sơ người dùng khác
<b>Description</b>	Người dùng xem thông tin chi tiết về hồ sơ của người dùng khác trên mạng xã hội, bao gồm thông tin cá nhân, bài viết và thực hiện các tương tác như theo dõi, nhắn tin.
<b>Actor</b>	Người dùng đã đăng nhập
<b>Pre-Conditions</b>	<ul style="list-style-type: none"> <li>– Người dùng đã đăng nhập vào hệ thống.</li> <li>– Hồ sơ người dùng cần xem tồn tại trong hệ thống.</li> </ul>
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>– Người dùng xem được thông tin chi tiết về người dùng khác.</li> <li>– Thực hiện được các tương tác với người dùng khác (theo dõi/bỏ theo dõi, nhắn tin).</li> </ul>
<b>Trigger</b>	Người dùng nhấp vào tên người dùng, ảnh đại diện hoặc đường dẫn đến hồ sơ người dùng khác.

<p><b>Normal Flow</b></p>	<ol style="list-style-type: none"> <li>1. Người dùng nhấn vào tên hoặc ảnh đại diện của người dùng khác.</li> <li>2. Hệ thống tải và hiển thị thông tin cá nhân của người dùng đó (tên, tên người dùng, ảnh đại diện, tiểu sử, trạng thái xác thực).</li> <li>3. Hệ thống hiển thị số lượng người theo dõi.</li> <li>4. Hệ thống hiển thị các nút tương tác: <ul style="list-style-type: none"> <li>– Nút "Theo dõi/Bỏ theo dõi"</li> <li>– Nút "Nhắn tin"</li> </ul> </li> <li>5. Hệ thống hiển thị bài viết của người dùng được xem.</li> <li>6. Người dùng có thể tương tác với hồ sơ bằng cách: <ul style="list-style-type: none"> <li>– Theo dõi hoặc bỏ theo dõi</li> <li>– Nhấn vào số người theo dõi để xem danh sách người theo dõi</li> <li>– Sao chép liên kết đến hồ sơ</li> <li>– Gửi tin nhắn</li> <li>– Xem và tương tác với bài viết</li> </ul> </li> </ol>
---------------------------	---

<b>Alternative Flow</b>	<ul style="list-style-type: none"> <li>– Nếu người dùng chưa đăng nhập: Vẫn có thể xem thông tin cơ bản, nhưng không thể thực hiện các hành động như theo dõi hoặc nhắn tin.</li> <li>– Nếu đang xem hồ sơ cá nhân của mình: Hiển thị nút "Cập nhật thông tin cá nhân" thay vì các nút theo dõi và nhắn tin.</li> <li>– Nếu người dùng đã bị chặn: Hiển thị thông báo không thể xem hồ sơ này.</li> <li>– Nếu có lỗi kết nối: Hiển thị thông báo lỗi và nút thử lại.</li> </ul>
-------------------------	---

Bảng 9. Bảng thông tin hoạt động của chức năng đăng ký

<b>Use-case name</b>	Đăng ký tài khoản
<b>Description</b>	Người dùng tạo tài khoản mới trên hệ thống Honey Social với xác thực email.
<b>Actor</b>	Người dùng chưa đăng nhập
<b>Pre-Conditions</b>	<ul style="list-style-type: none"> <li>– Người dùng chưa có tài khoản trên hệ thống.</li> <li>– Người dùng có quyền truy cập internet và trang đăng ký.</li> <li>– Người dùng có email hợp lệ để nhận mã xác thực.</li> </ul>
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>– Tài khoản mới được tạo và lưu trong cơ sở dữ liệu.</li> <li>– Email của người dùng được xác thực.</li> <li>– Người dùng có thể đăng nhập vào hệ thống với thông tin đăng nhập mới.</li> </ul>
<b>Trigger</b>	Người dùng truy cập trang đăng ký hoặc nhấn nút "Đăng ký" trên trang đăng nhập.

<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng nhập thông tin đăng ký (tên, tên người dùng, mật khẩu, email).</li> <li>2. Người dùng đồng ý với điều khoản sử dụng (nếu có) và nhấn nút "Đăng ký".</li> <li>3. Hệ thống kiểm tra tính hợp lệ của thông tin (định dạng email, mật khẩu đủ mạnh, tên người dùng chưa tồn tại).</li> <li>4. Hệ thống tạo tài khoản tạm thời và gửi email chứa mã xác thực đến địa chỉ email đã đăng ký.</li> <li>5. Hệ thống hiển thị giao diện nhập mã xác thực email.</li> <li>6. Người dùng nhận mã xác thực từ email và nhập mã vào hệ thống.</li> <li>7. Hệ thống xác minh mã và kích hoạt tài khoản.</li> <li>8. Hệ thống hiển thị thông báo đăng ký thành công và chuyển hướng người dùng đến trang đăng nhập hoặc trang chủ.</li> </ol>
--------------------	--

<b>Alternative Flow</b>	<ul style="list-style-type: none"> <li>– Nếu thông tin đăng ký không hợp lệ: Hệ thống hiển thị thông báo lỗi tương ứng và yêu cầu người dùng nhập lại.</li> <li>– Nếu email hoặc tên người dùng đã tồn tại: Hệ thống thông báo và gợi ý sử dụng thông tin khác.</li> <li>– Nếu người dùng không nhận được mã xác thực: Người dùng có thể nhấn "Gửi lại mã" để hệ thống gửi mã mới.</li> <li>– Nếu người dùng nhập sai mã xác thực: Hệ thống thông báo lỗi và cho phép nhập lại.</li> <li>– Nếu người dùng muốn thay đổi email: Người dùng có thể chọn "Đổi email" và nhập email mới.</li> </ul>
-------------------------	---

Bảng 10. Bảng thông tin hoạt động của chức năng quản lý nội dung báo cáo

<b>Use-case name</b>	Quản lý nội dung báo cáo
<b>Description</b>	Quản trị viên xem xét và xử lý các báo cáo về nội dung vi phạm từ người dùng.
<b>Actor</b>	Quản trị viên hệ thống
<b>Pre-Conditions</b>	<ul style="list-style-type: none"> <li>– Quản trị viên đã đăng nhập vào hệ thống với quyền quản lý nội dung.</li> <li>– Có ít nhất một báo cáo trong hệ thống cần được xử lý.</li> </ul>
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>– Báo cáo được xử lý (chấp nhận hoặc từ chối).</li> <li>– Nội dung vi phạm được xóa hoặc giữ lại tùy theo quyết định.</li> <li>– Trạng thái báo cáo được cập nhật.</li> </ul>

<b>Trigger</b>	Quản trị viên truy cập vào trang quản lý báo cáo hoặc nhận thông báo về báo cáo mới.
<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Quản trị viên truy cập vào phần quản lý báo cáo trong giao diện quản trị.</li> <li>2. Hệ thống hiển thị danh sách các báo cáo với thông tin: người báo cáo, nội dung vi phạm, ngày báo cáo, trạng thái, mức độ nghiêm trọng và các thao tác có thể thực hiện.</li> <li>3. Quản trị viên chọn một báo cáo để xem xét chi tiết.</li> <li>4. Hệ thống hiển thị thông tin đầy đủ về báo cáo và bài viết bị báo cáo.</li> <li>5. Quản trị viên xem nội dung bài viết bị báo cáo để đánh giá mức độ vi phạm.</li> <li>6. Quản trị viên đưa ra quyết định: <ul style="list-style-type: none"> <li>– Bỏ qua báo cáo (báo cáo không có cơ sở)</li> <li>– Xóa bài viết vi phạm (báo cáo hợp lệ)</li> </ul> </li> <li>7. Hệ thống cập nhật trạng thái báo cáo và thực hiện hành động tương ứng (giữ nguyên hoặc xóa bài viết).</li> </ol>

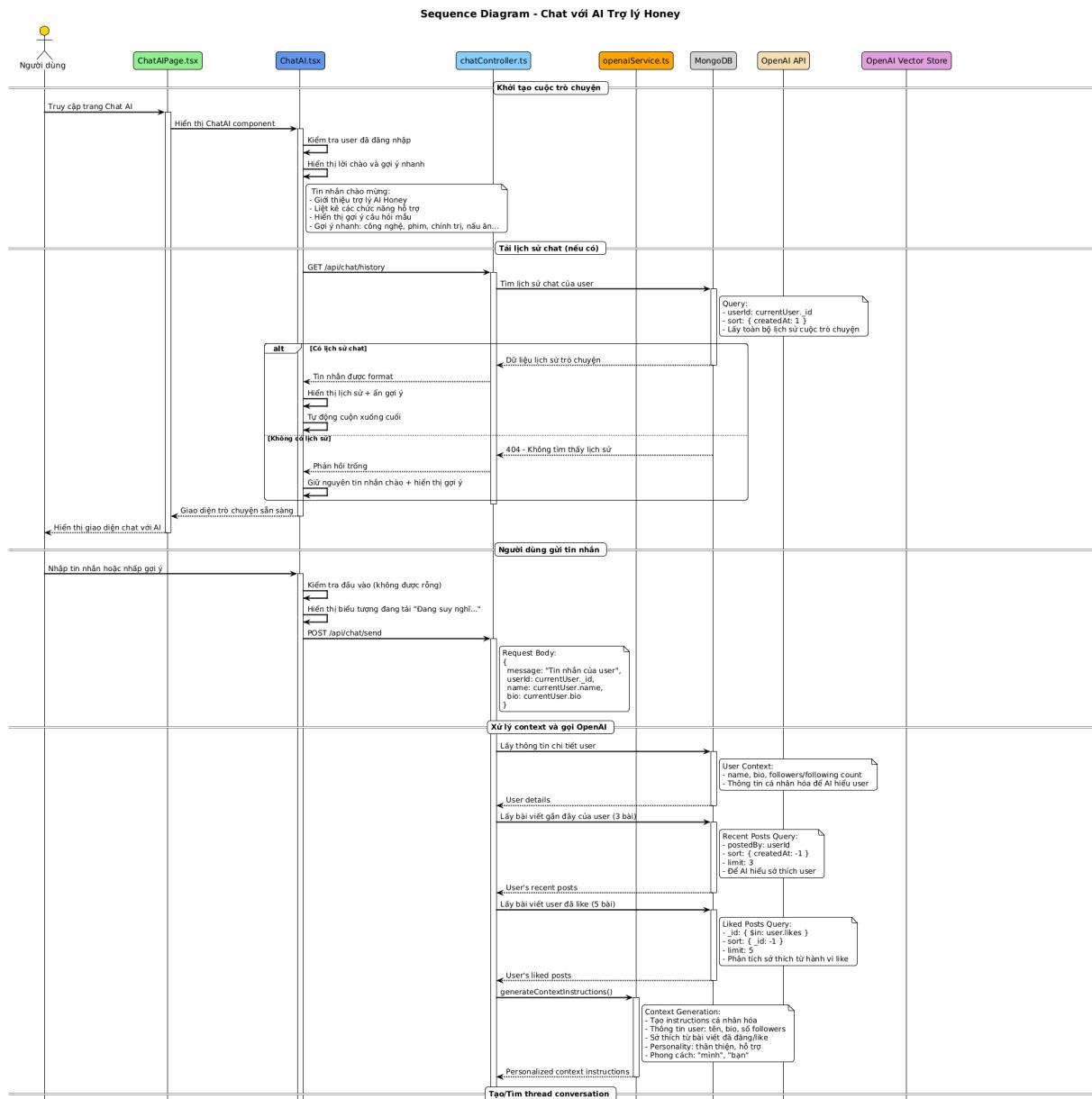
<b>Alternative Flow</b>	<ul style="list-style-type: none"> <li>– Nếu cần thêm thông tin: Quản trị viên có thể yêu cầu thêm chi tiết từ người báo cáo.</li> <li>– Nếu vi phạm nghiêm trọng: Quản trị viên có thể đình chỉ tài khoản người vi phạm.</li> <li>– Nếu có nhiều báo cáo cho cùng một nội dung: Hệ thống gom nhóm các báo cáo để quản trị viên xử lý một lần.</li> <li>– Trong trường hợp nhầm lẫn: Quản trị viên có thể khôi phục nội dung đã xóa và cập nhật trạng thái báo cáo.</li> </ul>
-------------------------	--

Bảng 11. Bảng thông tin hoạt động của chức năng tương tác bài viết

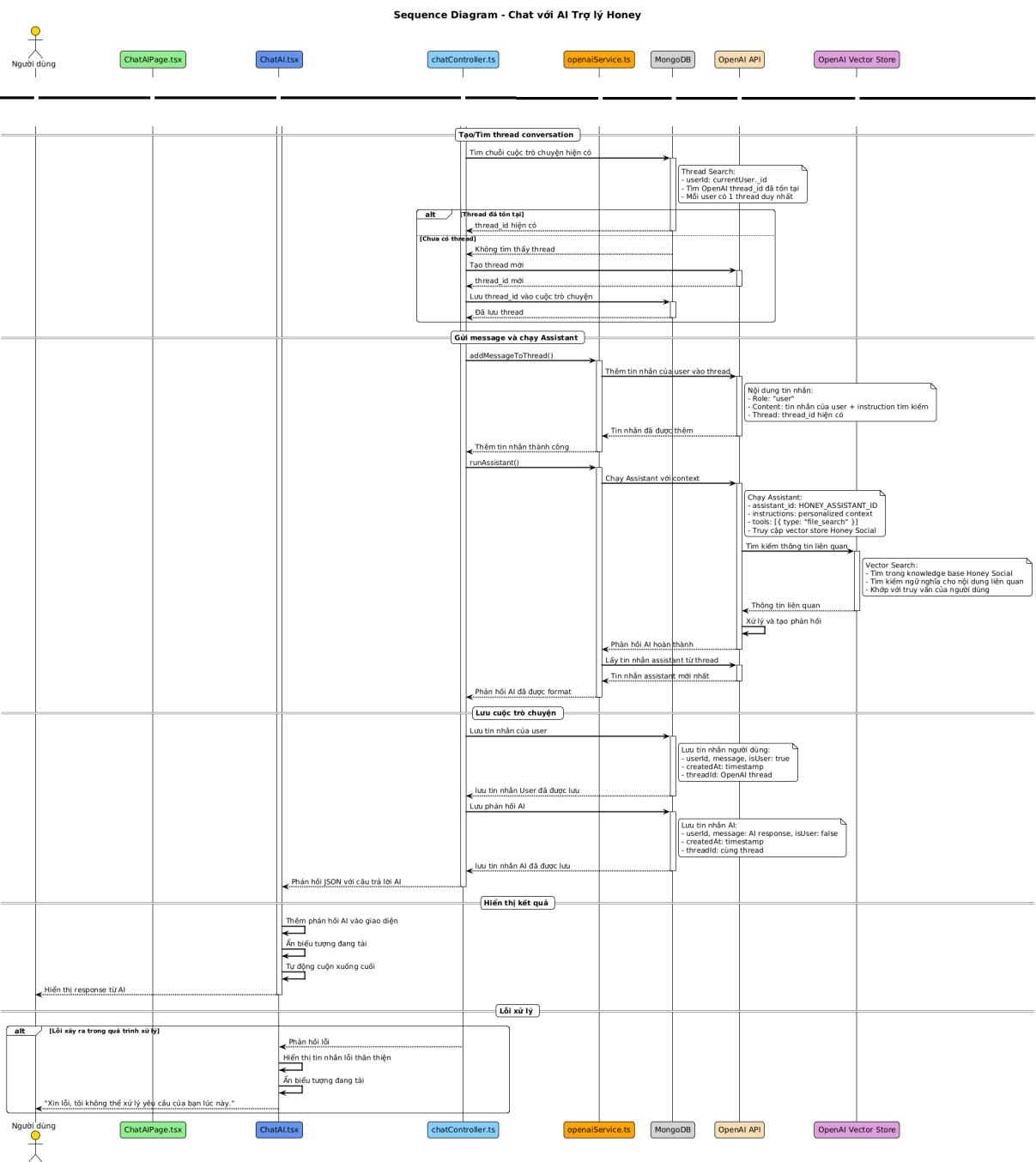
<b>Use-case name</b>	Tương tác bài viết
<b>Description</b>	Người dùng thực hiện các tương tác với bài viết như thích, bỏ thích, bình luận và tương tác với bình luận.
<b>Actor</b>	Người dùng đã đăng nhập
<b>Pre-Conditions</b>	<ul style="list-style-type: none"> <li>– Người dùng đã đăng nhập vào hệ thống.</li> <li>– Bài viết tồn tại và hiển thị trên giao diện.</li> </ul>
<b>Post-Condition</b>	<ul style="list-style-type: none"> <li>– Tương tác của người dùng được lưu và hiển thị (thích, bình luận).</li> <li>– Số lượt thích và/hoặc bình luận của bài viết được cập nhật.</li> <li>– Người dùng đăng bài viết và người dùng khác có liên quan nhận được thông báo (tùy loại tương tác).</li> </ul>
<b>Trigger</b>	Người dùng nhấn vào nút thích, ô nhập bình luận hoặc các biểu tượng tương tác khác trên bài viết.

<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. Người dùng xem bài viết trên bảng tin hoặc trang chi tiết.</li> <li>2. Người dùng thực hiện một trong các hành động sau: <ul style="list-style-type: none"> <li>– Nhấn nút "Thích" để thích bài viết (hoặc nhấn lại để bỏ thích)</li> <li>– Nhập nội dung trong ô bình luận và gửi bình luận</li> <li>– Nhấn nút thích/bỎ thích trên một bình luận cụ thể</li> </ul> </li> <li>3. Hệ thống ghi nhận hành động và cập nhật trạng thái: <ul style="list-style-type: none"> <li>– Đổi với thích: Cập nhật số lượt thích, thay đổi biểu tượng thích</li> <li>– Đổi với bình luận: Hiển thị bình luận mới, cập nhật số lượng bình luận</li> <li>– Đổi với thích bình luận: Cập nhật trạng thái và số lượt thích của bình luận đó</li> </ul> </li> <li>4. Hệ thống gửi thông báo đến chủ bài viết hoặc người bình luận (nếu cần).</li> </ol>
<b>Alternative Flow</b>	<ul style="list-style-type: none"> <li>– Nếu người dùng chưa đăng nhập: Yêu cầu đăng nhập khi thực hiện tương tác.</li> <li>– Nếu bình luận trống hoặc chỉ có khoảng trắng: Hệ thống hiển thị cảnh báo và không cho phép gửi bình luận.</li> <li>– Nếu xảy ra lỗi kết nối: Hiển thị thông báo lỗi và cho phép thử lại tương tác.</li> <li>– Nếu bài viết đã bị xóa: Hiển thị thông báo phù hợp và không cho phép tương tác thêm.</li> </ul>

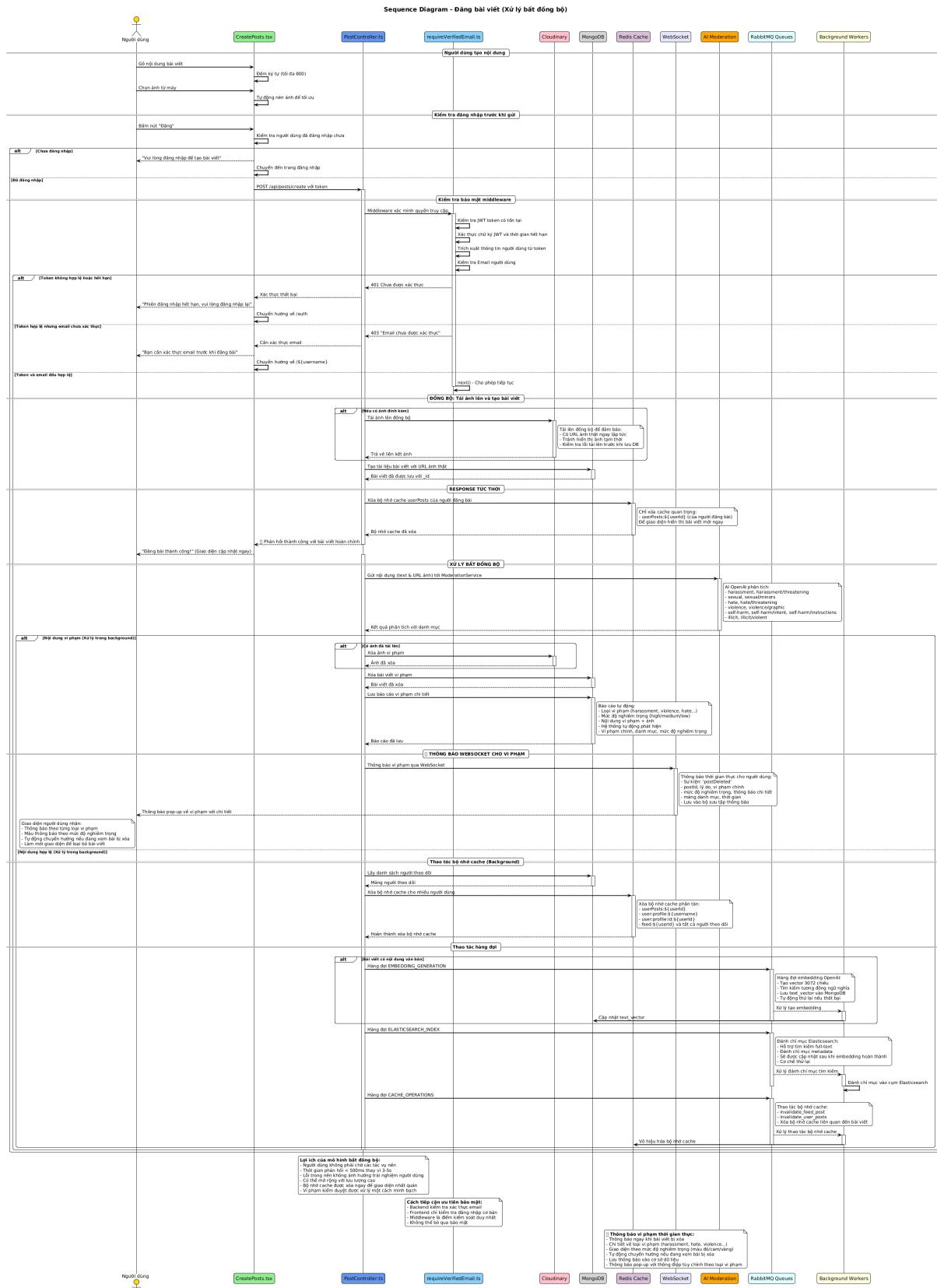
## 3.2. Thiết Kế Hệ Thống



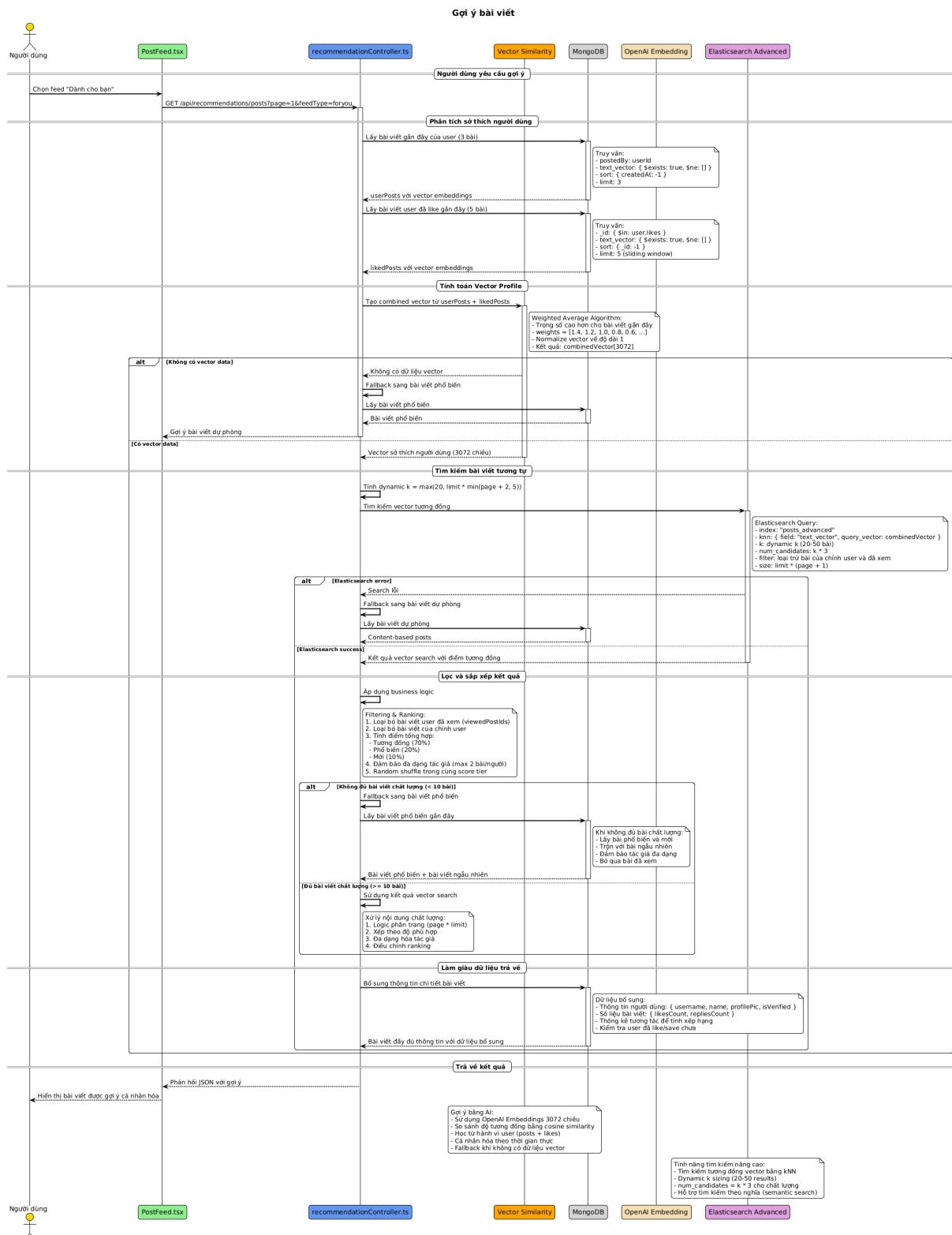
Hình 13. Hình ảnh Chat AI



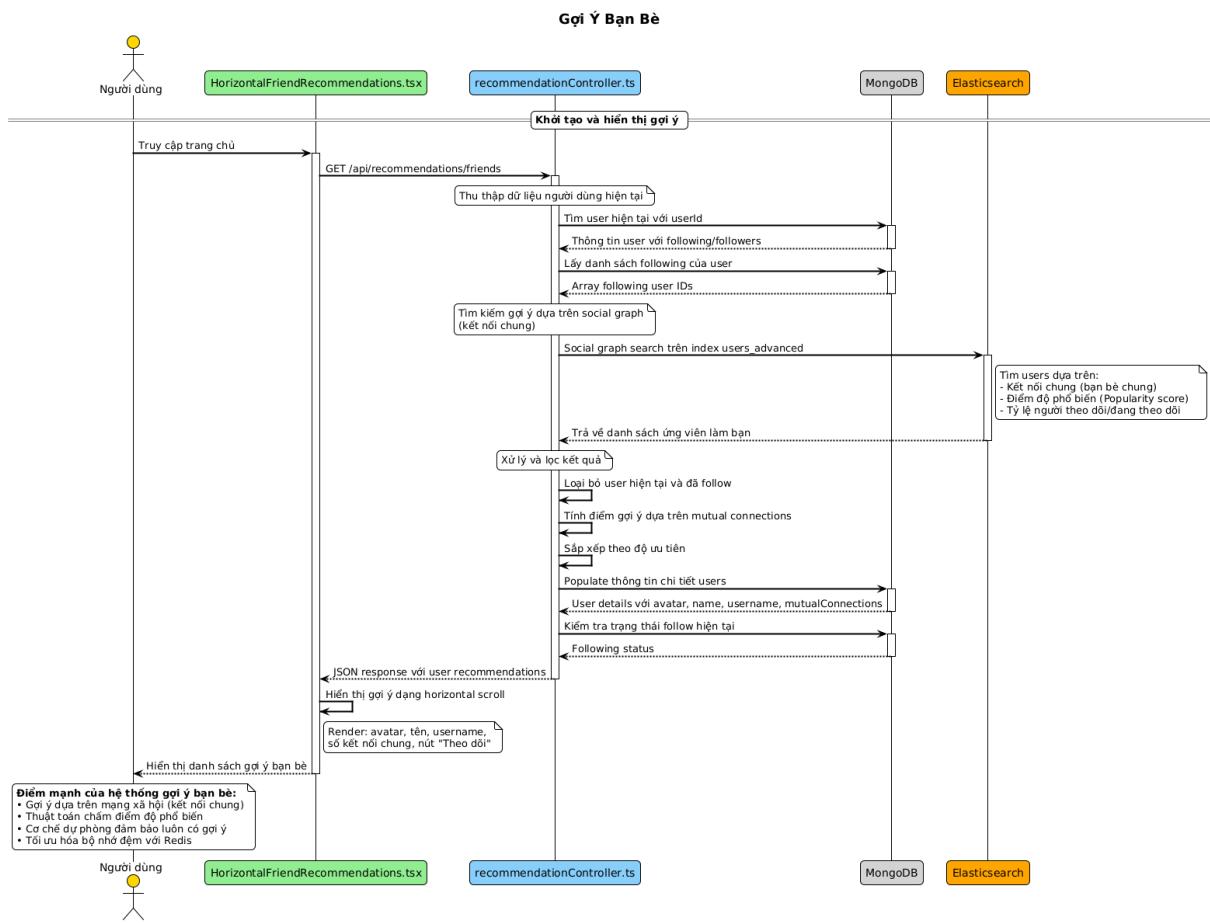
Hình 14. Hình ảnh Chat AI 2



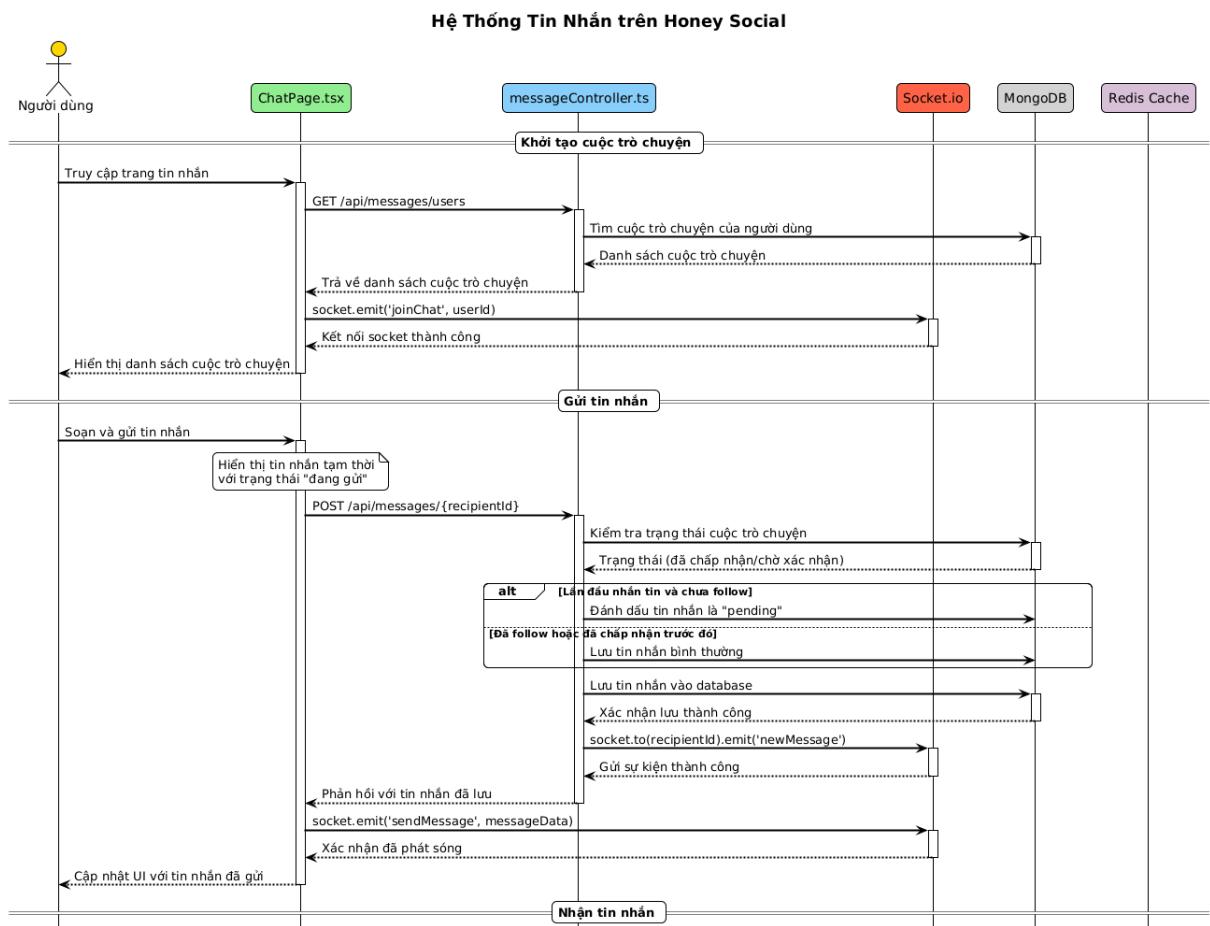
Hình 15. Hình ảnh Đăng bài viết



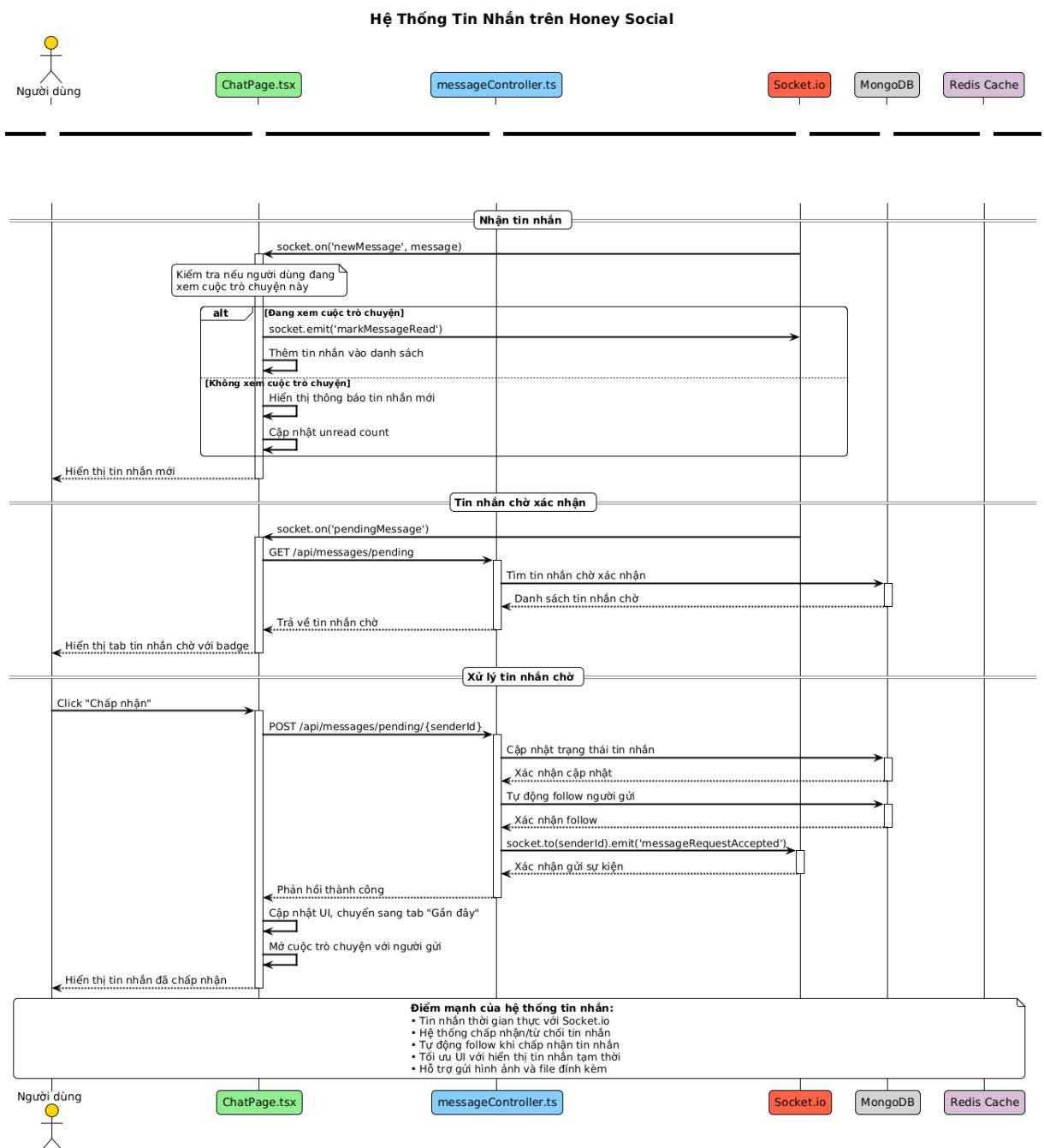
Hình 16. Hình ảnh Gợi ý bài viết



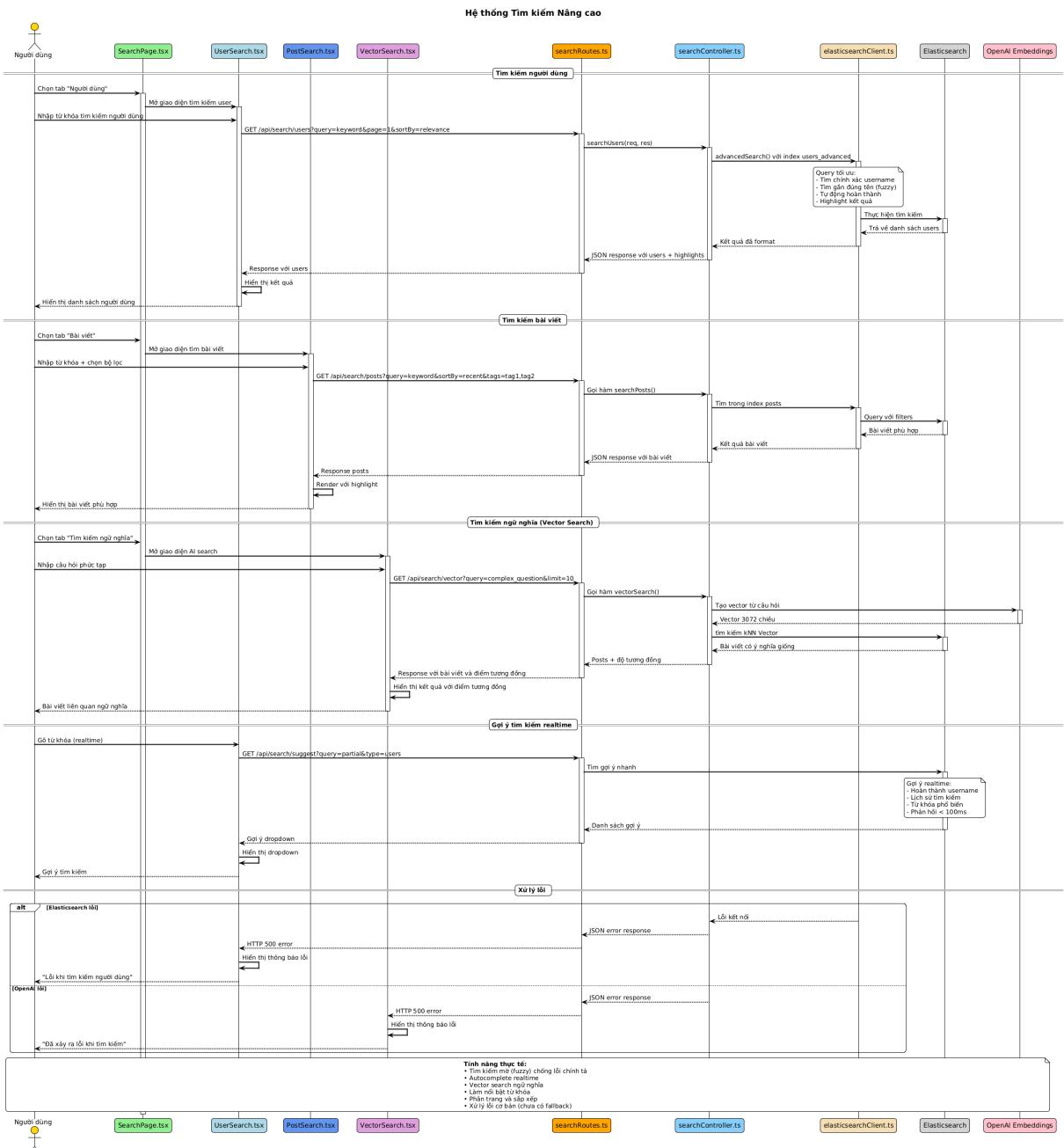
Hình 17. Hình ảnh Gợi ý bạn bè



Hình 18. Hình ảnh Hệ thống tin nhắn trên Honey Social



Hình 19. Hình ảnh Hệ thống tin nhắn trên Honey Social 2



Hình 20. Hình ảnh Tìm kiếm nâng cao

## Chương 4. THIẾT KẾ CƠ SỞ DỮ LIỆU

### 1. posts

Name	Data Type	Array	Key	Required
_id	objectId		Yes	Yes
postedBy	objectId			Yes
text	string			Yes
text_vector	double	Yes		Yes
img	string			Yes
likes	objectId	Yes		Yes
likesCount	double			Yes
replies	string	Yes		Yes
createdAt	date			Yes
updatedAt	date			Yes

Bảng 12. Cấu trúc của bảng posts

### 2. chatmessages

Name	Data Type	Array	Key	Required
_id	objectId		Yes	Yes
userId	objectId			Yes
isUser	bool			Yes
message	string			Yes
createdAt	date			Yes
updatedAt	date			Yes

Bảng 13. Cấu trúc của bảng chatmessages

### 3. threads

Name	Data Type	Array	Key	Required
_id	objectId		Yes	Yes
userId	objectId			Yes
createdAt	date			Yes
lastUsed	date			Yes
threadId	string			Yes

Bảng 14. Cấu trúc của bảng threads

#### 4. reports

Name	Data Type	Array	Key	Required
_id	objectId		Yes	Yes
reportedBy	any			Yes
reportedContent	objectId			Yes
contentType	string			Yes
reason	string			Yes
details	string			Yes
status	string			Yes
createdAt	date			Yes
updatedAt	date			Yes
resolvedBy	objectId			Yes
postId	objectId			Yes
postContent	string			Yes
moderationResult	string			Yes
resolution	string			Yes
severity	string			Yes
reportedByUsername	string			Yes
postImage	any			Yes

Bảng 15. Cấu trúc của bảng reports

## 5. verifications

Name	Data Type	Array	Key	Required
_id	objectId		Yes	Yes
userId	objectId			Yes
email	string			Yes
code	string			Yes
createdAt	date			Yes

Bảng 16. Cấu trúc của bảng verifications

## 6. messages

Name	Data Type	Array	Key	Required
_id	objectId		Yes	Yes
sender	objectId			Yes
recipient	objectId			Yes
messageType	string			Yes
text	string			Yes
media	null			
file	string			
fileName	null			
fileSize	null			
fileType	null			
thumbnailUrl	null			
duration	null			
read	bool			Yes
createdAt	date			Yes
updatedAt	date			Yes
cloudinary	string			Yes
fileInfo	string			Yes
mediaDetails	string			Yes
status	string			Yes
img	string			Yes
pending	bool			Yes

Bảng 17. Cấu trúc của bảng messages

## 7. users

Name	Data Type	Array	Key	Required
_id	objectId		Yes	Yes
name	string			Yes
username	string			Yes
email	string			Yes
password	string			Yes
profilePic	string			Yes
followers	string	Yes		Yes
following	string	Yes		Yes
bio	string			Yes
isFrozen	bool			Yes
createdAt	date			Yes
updatedAt	date			Yes
isEmailVerified	any			Yes
viewedPosts	objectId	Yes		Yes
lastActive	date			Yes
totalSessionTime	double			Yes
likes	objectId	Yes		Yes
isAdmin	bool			Yes

Bảng 18. Cấu trúc của bảng users

## 8. notifications

Name	Data Type	Array	Key	Required
_id	objectId		Yes	Yes
recipient	objectId			Yes
sender	objectId			Yes
type	string			Yes
post	objectId			Yes
read	bool			Yes
createdAt	date			Yes
updatedAt	date			Yes

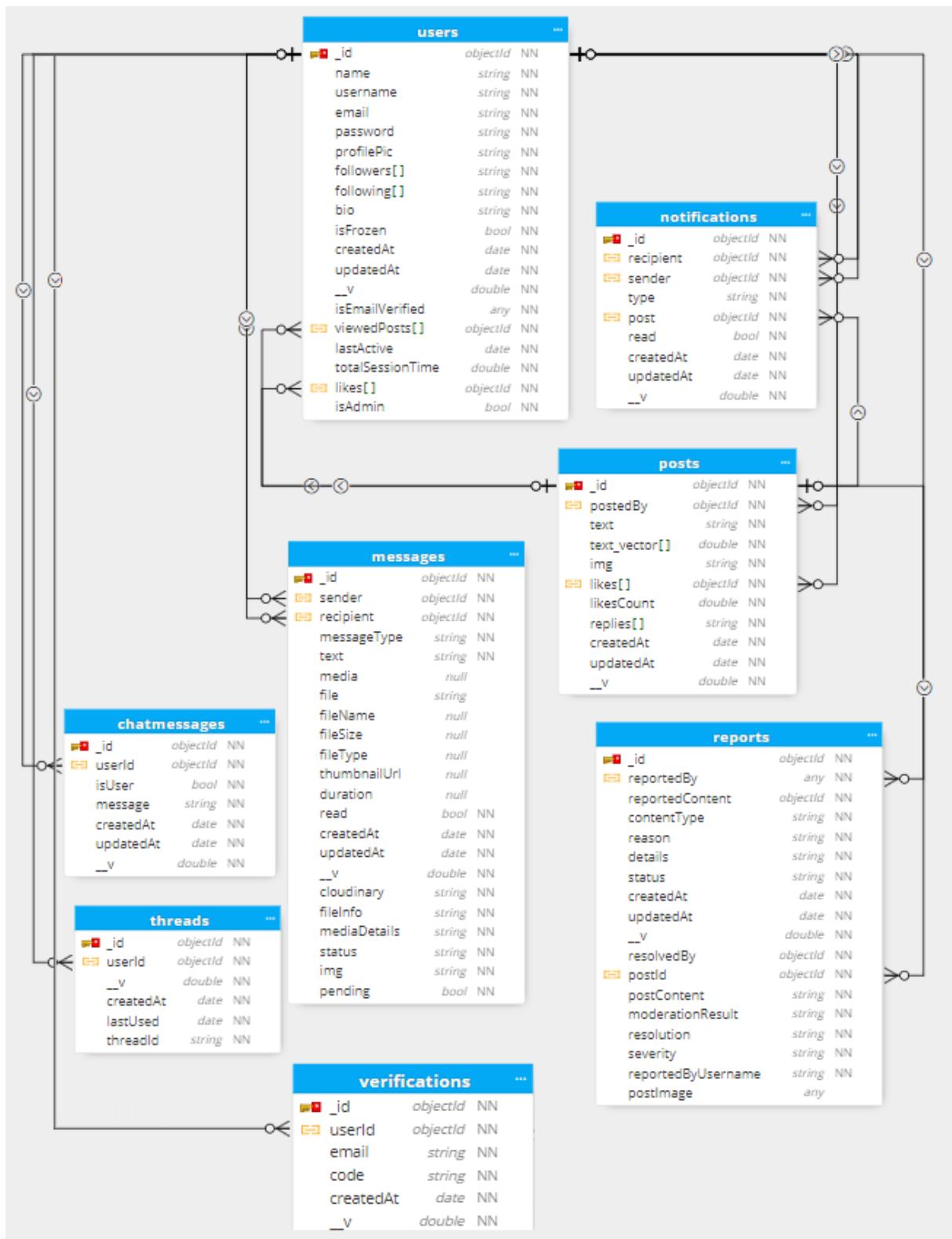
Bảng 19. Cấu trúc của bảng notifications

## 9. Quan hệ giữa các bảng

- 1 User có thể tạo nhiều Posts (qua trường postedBy)
- 1 User có thể thích nhiều Posts (qua mảng likes)
- 1 User có thể xem nhiều Posts (qua mảng viewedPosts)
- 1 User có thể gửi nhiều Messages (qua trường sender)
- 1 User có thể nhận nhiều Messages (qua trường recipient)
- 1 User có thể nhận nhiều Notifications (qua trường recipient)
- 1 User có thể tạo nhiều Notifications (qua trường sender)
- 1 User có thể có nhiều Chatmessages (qua trường userId)
- 1 User có thể có nhiều Threads (qua trường userId)
- 1 Post có thể tạo nhiều Notifications (qua trường post)
- 1 Post có thể có nhiều Reports (qua trường postId)
- 1 User có thể tạo nhiều Reports (qua trường reportedBy)
- 1 User có thể xử lý nhiều Reports (qua trường resolvedBy)

- 1 Post có thể có nhiều Replies (qua mảng replies)
- 1 User có thể theo dõi nhiều Users khác (qua mảng following)
- 1 User có thể được nhiều Users khác theo dõi (qua mảng followers)
- 1 User có thể có nhiều Verifications (qua trường userId)

Từ đó xây dựng biểu đồ cơ sở dữ liệu như hình vẽ dưới:



Hình 21. Hình ảnh Quan hệ giữa các bảng

## Chương 5. KẾT QUẢ THỰC NGHIỆM

### 1. Hệ thống kiểm duyệt nội dung đa phương thức

The screenshot shows a moderation dashboard titled "Bảng Kiểm Duyệt Nội Dung Vi Phạm". It includes a sidebar with icons for user management and reporting. The main area displays a table with 7 rows of data, each representing a moderation report. The columns are: VI PHẠM (Spam), NGƯỜI BÁO CÁO (Reporter), NGÀY BÁO CÁO (Report Date), TRẠNG THÁI (Status), MỨC ĐỘ (Severity), and THAO TÁC (Actions). All reports show "SPAM" in the first column, "RECOMMEND" in the second, and "30/5/2025" in the third. The status is "CHỜ XỬ LÝ" (Pending) and severity is "NHẸ" (Light). Each row has two buttons: "Xem xét" (Review) and "Xem bài viết" (View post).

VI PHẠM	NGƯỜI BÁO CÁO	NGÀY BÁO CÁO	TRẠNG THÁI	MỨC ĐỘ	THAO TÁC
SPAM	RECOMMEND	30/5/2025	CHỜ XỬ LÝ	NHẸ	Xem xét Xem bài viết
SPAM	RECOMMEND	30/5/2025	CHỜ XỬ LÝ	NHẸ	Xem xét Xem bài viết
SPAM	RECOMMEND	30/5/2025	CHỜ XỬ LÝ	NHẸ	Xem xét Xem bài viết
SPAM	RECOMMEND	30/5/2025	CHỜ XỬ LÝ	NHẸ	Xem xét Xem bài viết
SPAM	RECOMMEND	30/5/2025	CHỜ XỬ LÝ	NHẸ	Xem xét Xem bài viết
SPAM	RECOMMEND	30/5/2025	CHỜ XỬ LÝ	NHẸ	Xem xét Xem bài viết

Hình 22. Hình ảnh Bảng kiểm duyệt nội dung vi phạm

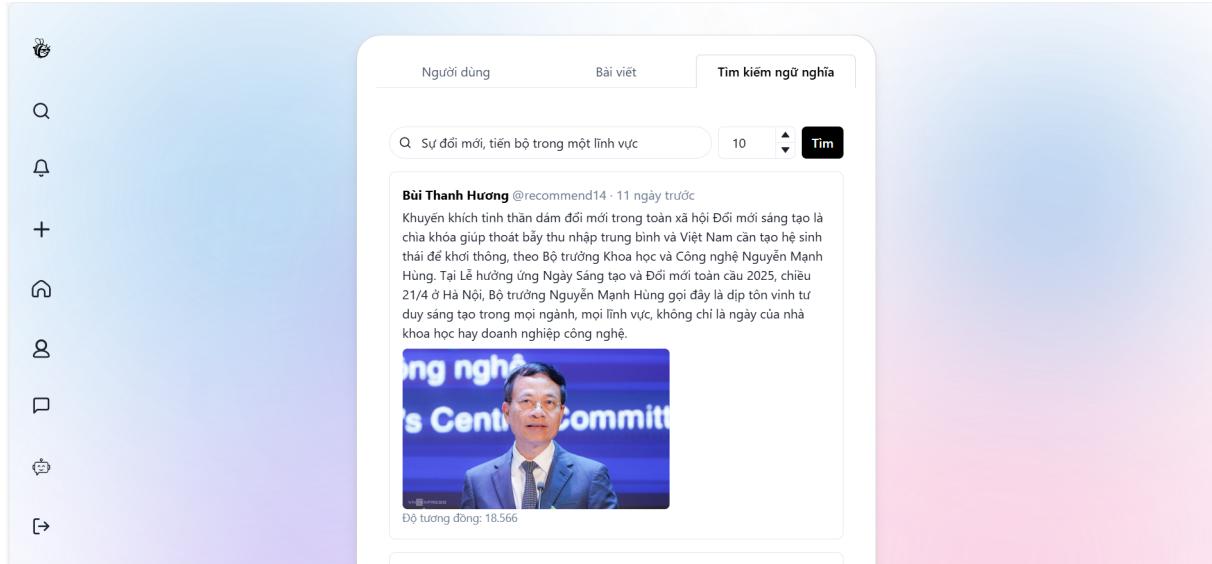
#### Phân tích thông minh với OpenAI Moderation API

- **Kiểm duyệt đa phương thức:** Phân tích cả văn bản và hình ảnh trong moderationService.ts
- **Phân loại mức độ nghiêm trọng:** Tự động xác định low, medium, high dựa trên điểm số vi phạm
- **Xử lý thời gian thực:** Kiểm tra ngay khi đăng bài, tạo báo cáo tự động trong moderationController.ts

#### Dashboard quản trị viên

- Giao diện quản lý báo cáo vi phạm hoàn chỉnh trong AdminPage.tsx
- Workflow phê duyệt/tù chối với lý do chi tiết
- Thống kê vi phạm theo thời gian và mức độ

## 2. Tìm kiếm ngữ nghĩa vector-based



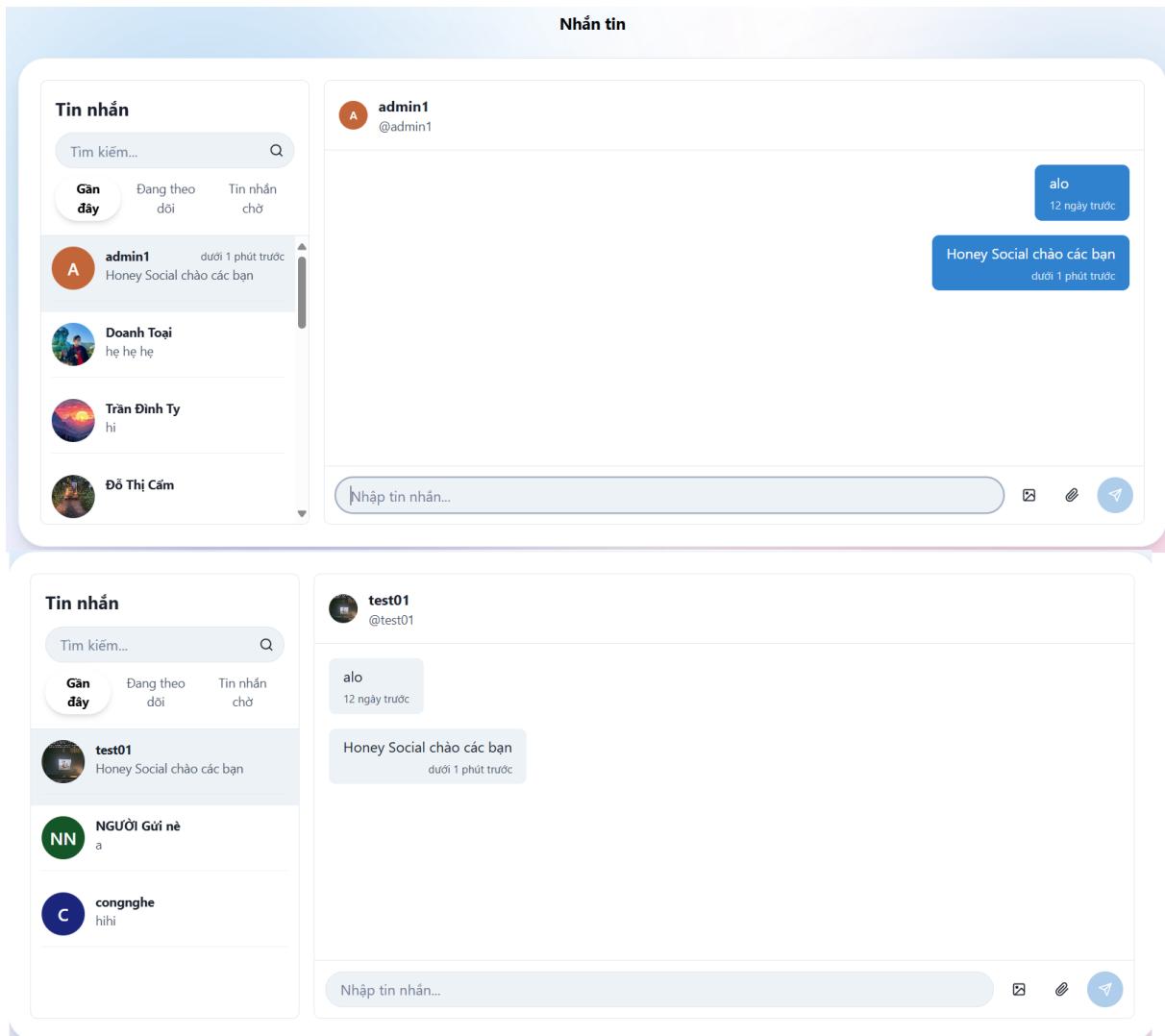
Hình 23. Hình ảnh Ngữ nghĩa

- **Vector embeddings:** Sử dụng OpenAI text-embedding-3-large trong embeddingService.ts
- **Elasticsearch kNN:** Tìm kiếm vector với độ chính xác cao
- **Tìm kiếm đa trường:** Kết hợp users, posts, comments trong một truy vấn

### Indexing thông minh

- **Auto-indexing:** Tự động tạo embeddings khi tạo bài viết mới
- **Batch processing:** Script đánh index hàng loạt trong indexAdvancedData.ts

### 3. Hệ thống nhắn tin thời gian thực



Hình 24. Hình ảnh Nhắn tin

#### Tính năng nâng cao

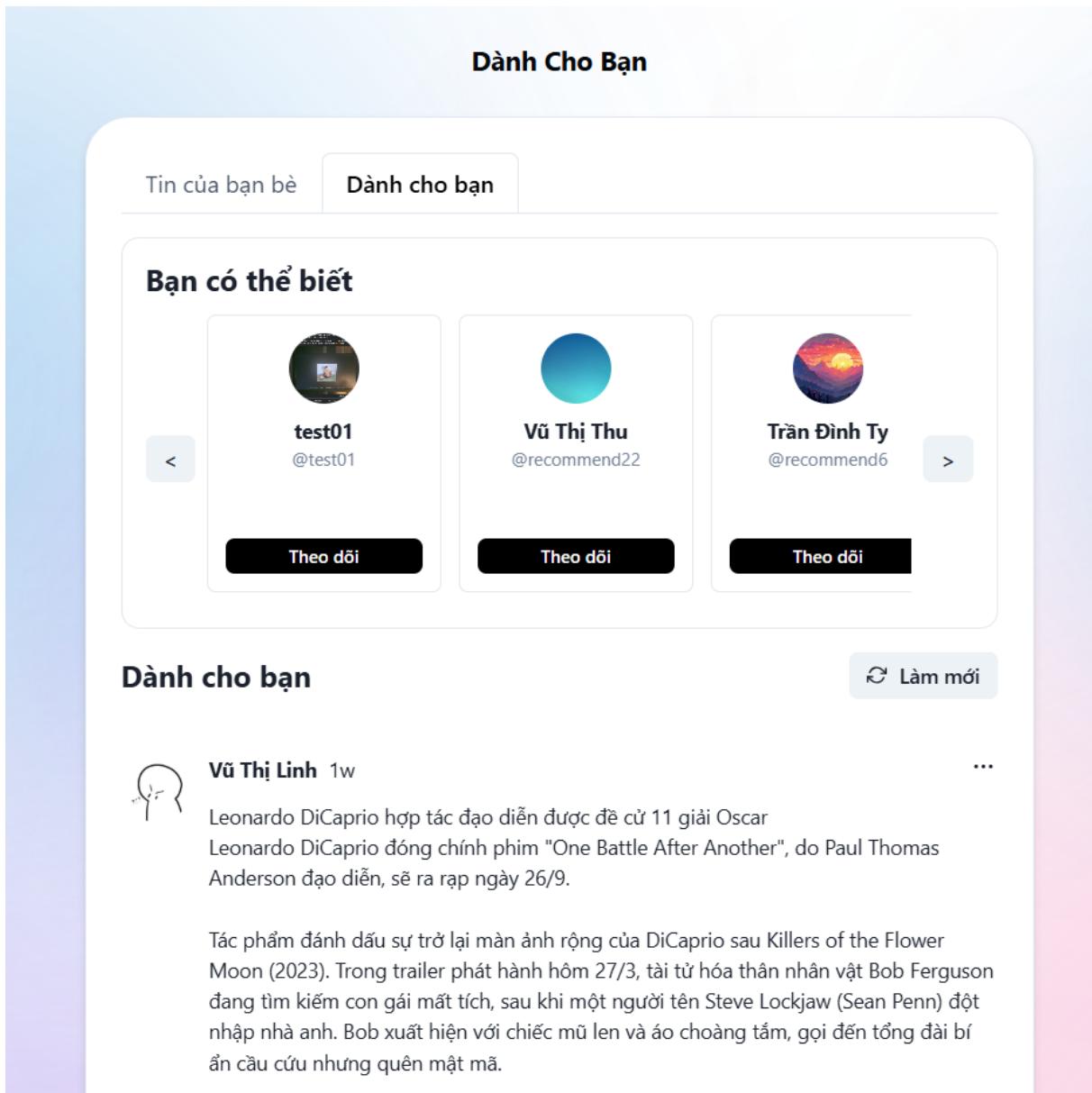
- **Socket.io real-time:** Tin nhắn tức thời với read receipts trong socket.ts
- **Message pending system:** Bảo vệ quyền riêng tư với workflow chấp nhận tin nhắn
- **Media sharing:** Upload ảnh/file với Cloudinary integration

#### Quản lý cuộc trò chuyện

- **Conversation management:** Giao diện chat với tabs Recent, Following, Pending
- **Auto-follow on accept:** Tự động follow khi chấp nhận tin nhắn

- **File upload:** Hỗ trợ nhiều định dạng với preview

#### 4. Hệ thống gợi ý thông minh



Hình 25. Hình ảnh Gợi ý

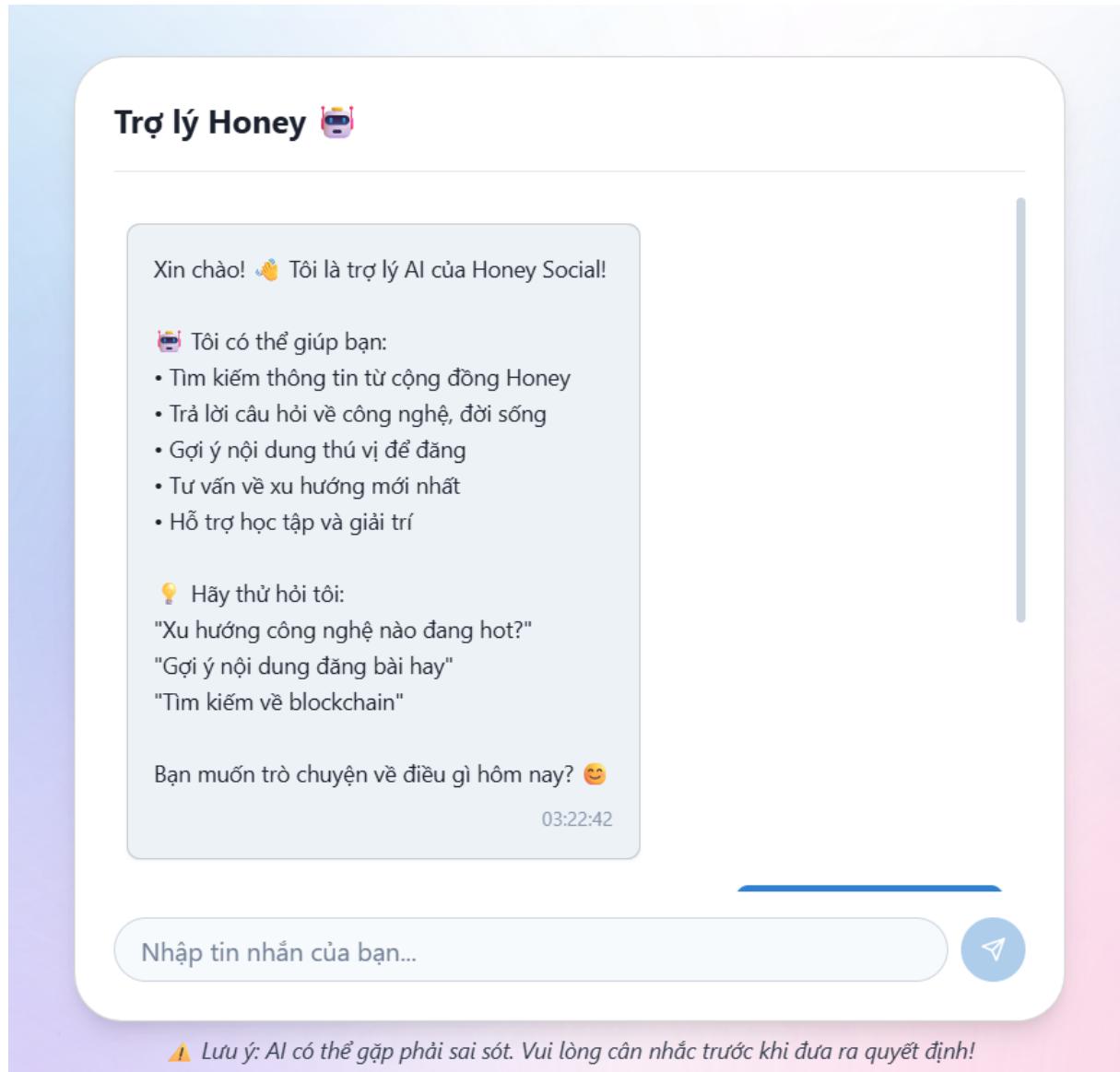
#### Gợi ý kết bạn dựa trên social graph

- **Mutual connections:** Phân tích kết nối chung trong recommendationController.ts
- **Popularity scoring:** Tính điểm uy tín dựa trên followers/following ratio
- **Elasticsearch aggregation:** Query phức tạp để tìm gợi ý tối ưu

#### Gợi ý nội dung cá nhân hóa

- **Vector similarity:** So sánh embedding của bài viết đã thích
- **Weighted combination:** Kết hợp nhiều vector với trọng số
- **Fallback mechanism:** Đảm bảo luôn có gợi ý dù thiếu dữ liệu

## 5. Trợ lý AI tích hợp



Hình 26. Hình ảnh Trí tuệ nhân tạo

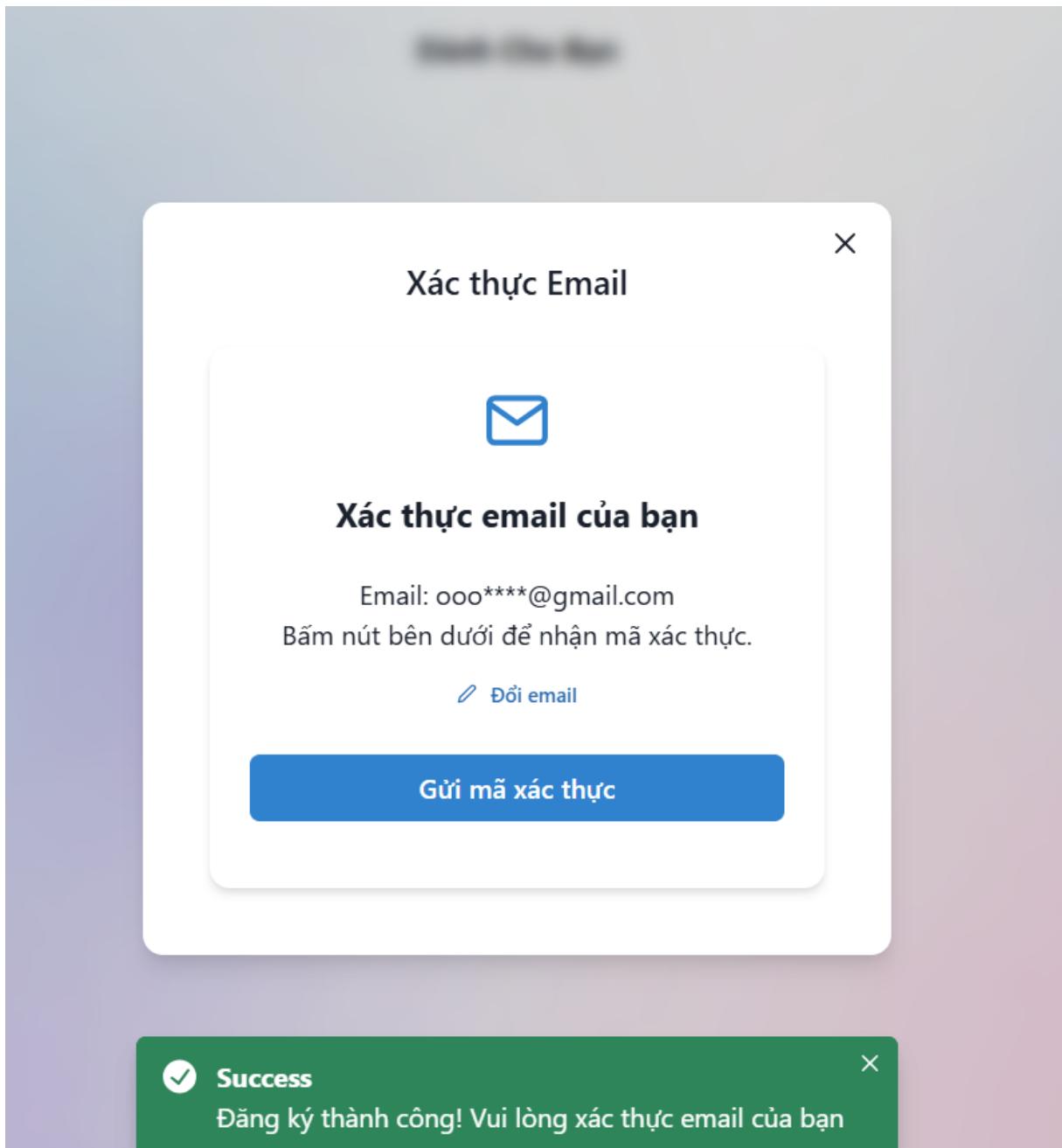
## OpenAI Assistant với context

- **Thread-based conversation:** Lưu trữ lịch sử trò chuyện trong openaiService.ts
- **Context awareness:** AI hiểu thông tin user như bio, posts, likes
- **File search integration:** Tìm kiếm thông tin từ database

## Giao diện chat thông minh

- **Quick suggestions:** Gợi ý câu hỏi nhanh trong ChatAI.tsx
- **Real-time typing:** Hiệu ứng typing indicator
- **Message history:** Lưu trữ và khôi phục cuộc trò chuyện

## 6. Hệ thống xác minh email



Hình 27. Hình ảnh Xác nhận mail

## Security với UX tối ưu

- **Rate limiting:** Chống spam với cooldown 45 giây trong verificationController.ts
- **Email masking:** Bảo mật thông tin email
- **Protected routes:** Component bảo vệ yêu cầu xác minh trong EmailProtectedRoute.tsx

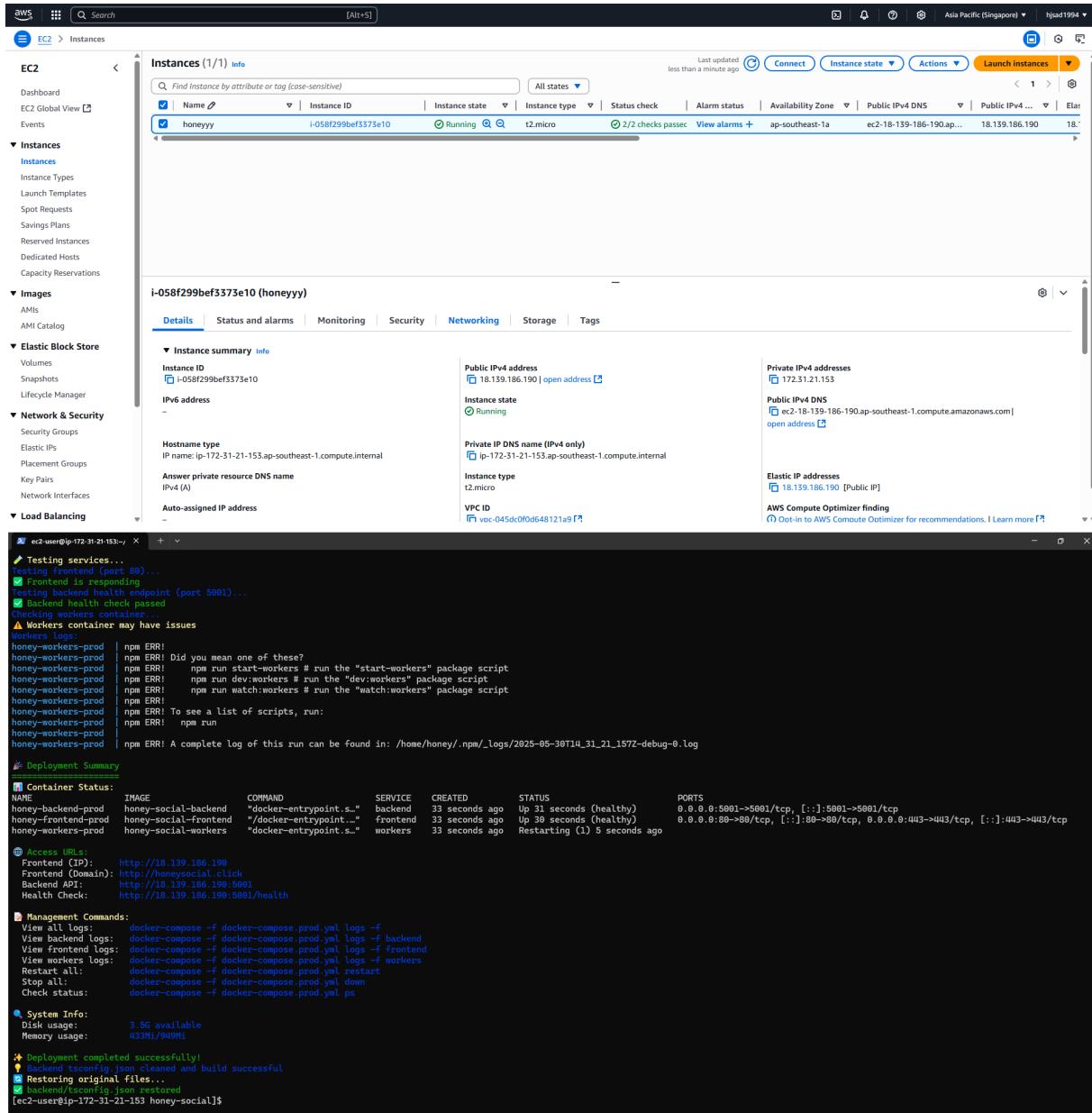
### **Workflow hoàn chỉnh**

- **Email change flow:** Quy trình đổi email an toàn
- **Queue processing:** Xử lý email bất đồng bộ
- **Modal integration:** Giao diện xác minh liền mạch

### **Background processing**

- **Social actions worker:** Xử lý like/follow bất đồng bộ trong socialActionsWorker.ts
- **Queue-based architecture:** Tách biệt logic nặng khỏi response time
- **Error handling:** Retry mechanism và fallback

## 7. Bảo mật và triển khai



Hình 28. Hình ảnh Deployment

## SSL Certificate tự động

- **Let's Encrypt SSL:** Certificate miễn phí cho domain honeysocial.click
- **Auto-renewal:** Script tự động gia hạn certificate
- **HTTPS redirect:** Tự động chuyển hướng từ HTTP sang HTTPS

## Deployment hoàn chỉnh

- **Docker containerization:** Triển khai với docker-compose

- **Nginx reverse proxy:** Load balancing và SSL termination
- **Production monitoring:** Health checks và logging

## **Chương 6. KẾT LUẬN VÀ KIẾN NGHỊ**

### **6.1. Kết Luận**

#### **6.1.1. Tổng kết quá trình thực hiện đồ án**

Đồ án đã thực hiện xây dựng thành công mạng xã hội Honey Social với kiến trúc hiện đại, tách biệt Frontend và Backend rõ ràng. Quá trình phát triển tuân theo phương pháp Agile, cho phép linh hoạt điều chỉnh khi gặp thách thức. Việc áp dụng các mô hình thiết kế như MVC, đã giúp mã nguồn có cấu trúc tốt, dễ bảo trì và mở rộng.

#### **6.1.2. Kết quả đạt được**

##### **1. Hệ thống đa chức năng - Mạng xã hội hoàn chỉnh với các tính năng chính:**

- Đăng bài viết với hình ảnh và văn bản
- Hệ thống chat realtime giữa người dùng
- Trợ lý AI thông minh tích hợp
- Hệ thống gợi ý nội dung dựa trên vector search
- Hệ thống thông báo realtime
- Tìm kiếm nâng cao với Elasticsearch

##### **2. Kiến trúc kỹ thuật hiện đại:**

- Frontend: React.js với state management thông qua Redux
- Backend: Node.js, Express.js với TypeScript
- Cơ sở dữ liệu: MongoDB kết hợp với Redis cache
- WebSocket cho giao tiếp realtime
- RabbitMQ cho hàng đợi xử lý bất đồng bộ
- Docker cho việc phát triển và triển khai

##### **3. Hiệu năng cao:**

- Hệ thống cache đa tầng giúp giảm tải database
- Xử lý bất đồng bộ các tác vụ nặng như phân tích nội dung, moderation
- Vector search cho khả năng tìm kiếm và gợi ý thông minh

### **6.1.3. Hạn chế và thách thức**

#### **1. Độ phức tạp của hệ thống:**

- Cache invalidation khó xử lý đúng trong mọi tình huống
- Phân tán microservice làm tăng độ phức tạp vận hành

#### **2. Hiệu năng của AI:**

- Thời gian phản hồi của ChatAI đôi khi còn chậm
- Vector embedding tồn kém tài nguyên khi hệ thống lớn

#### **3. Giới hạn tích hợp:**

- Chưa có ứng dụng di động native
- Hệ thống gợi ý cần thêm dữ liệu để tăng độ chính xác

## **6.2. Kiến nghị và hướng phát triển**

### **6.2.1. Phát triển ứng dụng di động**

Xây dựng ứng dụng di động native cho iOS và Android sử dụng React Native hoặc Flutter để tận dụng codebase hiện có, đồng thời cải thiện trải nghiệm người dùng trên thiết bị di động với các tính năng như thông báo đẩy, camera tích hợp và trải nghiệm offline.

### **6.2.2. Nâng cấp hệ thống gợi ý nội dung**

Cải tiến hệ thống gợi ý bằng cách kết hợp các yếu tố hành vi người dùng (thời gian xem, tương tác), ngữ cảnh (thời gian, vị trí) và mạng xã hội (kết nối giữa người dùng) để tạo ra gợi ý cá nhân hóa chính xác hơn.

### **6.2.3. Triển khai AI Chatbot học tự động**

Phát triển khả năng học tự động cho ChatAI để liên tục cải thiện chất lượng phản hồi dựa trên tương tác của người dùng. Triển khai fine-tuning mô hình để chatbot hiểu tốt hơn ngữ cảnh và văn hóa cụ thể của cộng đồng Honey Social.

### **6.2.4. Tối ưu cơ chế làm mới Redis Cache**

Cải tiến cơ chế cache với chiến lược write-through và read-through thông minh hơn, đồng thời sử dụng Redis Streams để quản lý việc làm mới cache phân tán, giảm thiểu lỗi cache inconsistency trong hệ thống nhiều node.

### **6.2.5. Cải tiến hệ thống Post Recommendation**

Nâng cấp hệ thống gợi ý bài đăng bằng cách sử dụng kỹ thuật collaborative filtering kết hợp với vector search, đồng thời tối ưu hóa thuật toán lọc bọt (filter bubble) để đảm bảo người dùng tiếp cận được nhiều nội dung đa dạng.

### **6.2.6. Phát triển ChatboxAI nội bộ**

Xây dựng công cụ chatbot nội bộ được huấn luyện trên dữ liệu hệ thống để hỗ trợ người quản trị trong việc phân tích xu hướng, phát hiện nội dung vi phạm, và tổng hợp báo cáo hoạt động của cộng đồng.

## TÀI LIỆU THAM KHẢO

### Tài liệu

- [1] Amazon Web Services, "Deploy a React-based single-page application to Amazon S3 and CloudFront." [Online].  
Available: <https://short.com.vn/P0gh>
- [2] OpenAI, "Moderation Guide." [Online].  
Available: <https://platform.openai.com/docs/guides/moderation>
- [3] OpenAI, "Assistants API Overview." [Online].  
Available: <https://platform.openai.com/docs/assistants/overview>
- [4] OpenAI, "Embeddings Guide." [Online].  
Available: <https://platform.openai.com/docs/guides/embeddings>
- [5] GeeksforGeeks, "Cache-Aside Pattern." [Online].  
Available: <https://www.geeksforgeeks.org/cache-aside-pattern/>
- [6] Knowi, "What is Elasticsearch?" [Online].  
Available: <https://www.knowi.com/blog/what-is-elasticsearch/>