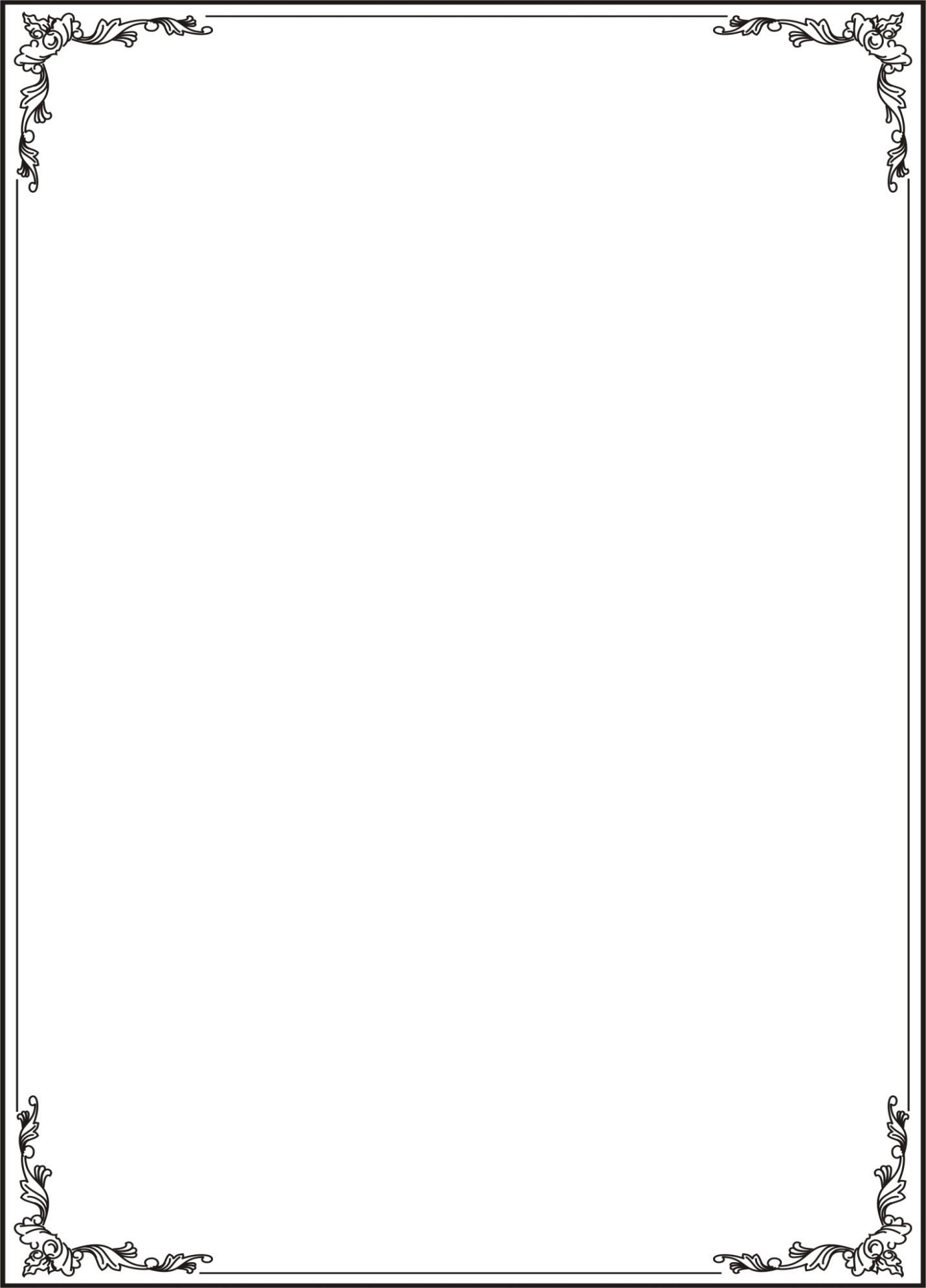
**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

KHOA ĐÀO TẠO SAU ĐẠI HỌC

-----🙞🙜🕮🙞🙜-----

A logo with a star and a book

AI-generated content may be incorrect.

BÁO CÁO

**CÁC HỆ THỐNG PHÂN TÁN**

**ĐỀ TÀI: XÂY DỰNG HỆ THỐNG MẠNG XÃ HỘI BENTO SOCIAL**

**ÁP DỤNG KIẾN TRÚC MICROSERVICES**

|  |  |  |
| --- | --- | --- |
| *Giảng viên:* | **TS. Kim Ngọc Bách** |  |
| *Lớp:* | **M25CQHT01-B** |  |
| *Các học viên thực hiện:* | **Nguyễn Hoàng Anh** | **B25CHHT003** |
|  | **Hoàng Thị Thanh Bình** | **B25CHHT004** |
|  | **Trần Thị Lan Anh** | **B25CHHT002** |

**Hà Nội - 2025**

MỤC LỤC

[LỜI MỞ ĐẦU 4](#_Toc217851496)

[I. TỔNG QUAN VỀ ĐỀ TÀI VÀ KIẾN TRÚC MICROSERVICES 5](#_Toc217851497)

[1.1. Đặt vấn đề 5](#_Toc217851498)

[1.2. Tổng quan về bài toán và mục tiêu đề tài 6](#_Toc217851499)

[1.2.1. Đặc điểm bài toán hệ thống mạng xã hội 6](#_Toc217851500)

[1.2.2. Mục tiêu đề tài 6](#_Toc217851501)

[1.3. Tổng quan kiến trúc Microservices 7](#_Toc217851502)

[1.3.1. Khái niệm kiến trúc Microservices 7](#_Toc217851503)

[1.3.2. Đặc trưng của Microservices 8](#_Toc217851504)

[1.3.3. Đánh giá ưu và nhược điểm của Microservices 10](#_Toc217851505)

[1.4. So sánh kiến trúc Monolithic và Microservices 12](#_Toc217851506)

[1.4.1. Kiến trúc Monolithic 12](#_Toc217851507)

[1.4.2. Kiến trúc Microservices 12](#_Toc217851508)

[1.4.3. Bảng so sánh kiến trúc Monolithic và Microservices 13](#_Toc217851509)

[1.5. Định hướng áp dụng Microservices cho hệ thống Bento Social 13](#_Toc217851510)

[1.6. Tổng quan công nghệ sử dụng 14](#_Toc217851511)

[II. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 15](#_Toc217851512)

[2.1. Kiến trúc hệ thống tổng quát 15](#_Toc217851513)

[2.1.1. Mô hình tổng thể Client–Server 15](#_Toc217851514)

[2.1.2. Định hướng Microservices-ready và Event-driven 16](#_Toc217851515)

[2.2. Phân tích và thiết kế chi tiết hệ thống 16](#_Toc217851516)

[2.2.1. Phân tích tác nhân và chức năng hệ thống 16](#_Toc217851517)

[2.2.2. Biểu đồ Use Case 19](#_Toc217851518)

[2.2.3. Biểu đồ Sequence 30](#_Toc217851519)

[2.2.4. Thiết kế cơ sở dữ liệu (Database Design) 34](#_Toc217851520)

[III. XÂY DỰNG ỨNG DỤNG 35](#_Toc217851521)

[3.1. Môi trường phát triển 35](#_Toc217851522)

[3.2. Cài đặt và triển khai hệ thống 36](#_Toc217851523)

[3.2.1. Thiết lập Backend 36](#_Toc217851524)

[3.2.2. Thiết lập Frontend 37](#_Toc217851525)

[3.3. Giao diện người dùng 37](#_Toc217851526)

[3.3.1. Giao diện Đăng nhập (Login) 37](#_Toc217851527)

[3.3.2. Giao diện Đăng ký (Register) 39](#_Toc217851528)

[3.3.3. Giao diện Trang chủ (Home) 40](#_Toc217851529)

[3.3.4. Giao diện Khám phá (Explore) 41](#_Toc217851530)

[3.3.5. Giao diện Thông báo (Notifications) 41](#_Toc217851531)

[3.3.6. Giao diện Trang cá nhân (Profile) 42](#_Toc217851532)

[3.3.7. Giao diện Chi tiết bài viết (Post Detail) 43](#_Toc217851533)

[KẾT LUẬN 44](#_Toc217851534)

[TÀI LIỆU THAM KHẢO 46](#_Toc217851535)

**MỤC LỤC HÌNH ẢNH**

[Hình 1.1 Mô hình kiến trúc Microservice 8](#_Toc217851066)

[Hình 1.2 Đặc trưng của Microservice 9](#_Toc217851067)

[Hình 1.3 Kiến trúc bảo mật trong hệ thống Microservices sử dụng JWT 11](#_Toc217851068)

[Hình 1.4 So sánh kiến trúc Monolithic và kiến trúc Microservices 13](#_Toc217851069)

[Hình 2.1 Biểu đồ Use Case Tổng quát 21](#_Toc217851070)

[Hình 2.2 Biểu đồ Use Case Đăng nhập 22](#_Toc217851071)

[Hình 2.3 Biểu đồ Use Case Đăng bài viết 24](#_Toc217851072)

[Hình 2.4 Biểu đồ Use Case Bình luận bài viết 25](#_Toc217851073)

[Hình 2.5 Biểu đồ Use Case Xem thông báo 27](#_Toc217851074)

[Hình 2.6 Biểu đồ Use Case Quản lý chủ đề 28](#_Toc217851075)

[Hình 2.7 Biểu đồ Use Case Theo dõi người dùng 30](#_Toc217851076)

[Hình 2.8 Biểu đồ Sequence Đăng nhập 32](#_Toc217851077)

[Hình 2.9 Biểu đồ Sequence Đăng bài viết 32](#_Toc217851078)

[Hình 2.10 Biểu đồ Sequence Bình luận 33](#_Toc217851079)

[Hình 2.11 Biểu đồ Sequence Thông báo 33](#_Toc217851080)

[Hình 2.12 Biểu đồ Sequence Quản lý chủ đề 34](#_Toc217851081)

[Hình 2.13 Biểu đồ Sequence Theo dõi người dùng 34](#_Toc217851082)

[Hình 2.14 Sơ đồ thiết kế cơ sở dữ liệu của hệ thống Bento Social 35](#_Toc217851083)

[Hình 3.1 Giao diện Đăng nhập 39](#_Toc217851084)

[Hình 3.2 Giao diện Đăng ký 40](#_Toc217851085)

[Hình 3.3 Giao diện Trang chủ 41](#_Toc217851086)

[Hình 3.4 Giao diện Khám phú 42](#_Toc217851087)

[Hình 3.5 Giao diện Thông báo 42](#_Toc217851088)

[Hình 3.6 Giao diện Trang cá nhân 43](#_Toc217851089)

[Hình 3.7 Giao diện Chi tiết bài viết 44](#_Toc217851090)

# LỜI MỞ ĐẦU

Sự phát triển của các hệ thống phần mềm hiện đại ngày càng gắn liền với kiến trúc phân tán nhằm đáp ứng yêu cầu về khả năng mở rộng, tính sẵn sàng và hiệu quả xử lý. Trong đó, các hệ thống mạng xã hội là ví dụ điển hình với lượng người dùng lớn và mức độ tương tác cao, đòi hỏi kiến trúc hệ thống linh hoạt và dễ mở rộng.

Kiến trúc Microservices là một trong những hướng tiếp cận phổ biến hiện nay để giải quyết các bài toán trên trong môi trường hệ thống phân tán. Việc chia hệ thống thành các dịch vụ độc lập giúp giảm sự phụ thuộc giữa các thành phần, tăng tính linh hoạt trong phát triển và triển khai, đồng thời tạo điều kiện thuận lợi cho việc mở rộng hệ thống khi cần thiết.

Trong khuôn khổ môn học *Các hệ thống phân tán* thuộc chương trình đào tạo Thạc sĩ chuyên ngành Hệ thống thông tin, báo cáo này lựa chọn đề tài **“Xây dựng hệ thống mạng xã hội Bento Social áp dụng kiến trúc Microservices”** nhằm nghiên cứu và áp dụng các nguyên lý của hệ thống phân tán vào một bài toán thực tế. Đề tài tập trung vào việc phân tích, thiết kế và triển khai một hệ thống mạng xã hội ở mức cơ bản, trong đó nhấn mạnh kiến trúc phân tách Client–Server, cơ chế xử lý bất đồng bộ và định hướng Microservices-ready.

Nội dung báo cáo không chỉ trình bày kết quả triển khai hệ thống mà còn làm rõ các quyết định thiết kế kiến trúc và tổ chức hệ thống. Thông qua việc xây dựng và cài đặt Bento Social, báo cáo góp phần củng cố kiến thức lý thuyết về hệ thống phân tán và giúp người học tiếp cận gần hơn với các mô hình kiến trúc đang được áp dụng trong thực tế.

Do giới hạn về thời gian và phạm vi môn học, báo cáo khó tránh khỏi những thiếu sót nhất định. Nhóm thực hiện rất mong nhận được những ý kiến đóng góp quý báu từ giảng viên để có thể hoàn thiện và phát triển đề tài tốt hơn trong tương lai.

# I. TỔNG QUAN VỀ ĐỀ TÀI VÀ KIẾN TRÚC MICROSERVICES

## 1.1. Đặt vấn đề

Trong những năm gần đây, sự phát triển mạnh mẽ của Internet, điện toán đám mây và các thiết bị di động đã thúc đẩy sự bùng nổ của các nền tảng mạng xã hội. Các hệ thống này không chỉ phục vụ nhu cầu giao tiếp, chia sẻ thông tin mà còn đóng vai trò quan trọng trong việc hình thành cộng đồng, lan tỏa tri thức và kết nối con người trên quy mô lớn.

Tuy nhiên, cùng với sự gia tăng nhanh chóng về số lượng người dùng và mức độ tương tác, các hệ thống mạng xã hội hiện đại phải đối mặt với nhiều thách thức kỹ thuật, đặc biệt là:

* Khả năng **mở rộng hệ thống (scalability)** khi lưu lượng truy cập tăng cao;
* Xử lý **tương tác thời gian thực** như like, bình luận, theo dõi và thông báo;
* Đảm bảo **tính sẵn sàng cao và khả năng chịu lỗi** trong môi trường phân tán;
* Giảm sự phụ thuộc chặt chẽ giữa các thành phần để hệ thống dễ bảo trì và phát triển lâu dài.

Trong bối cảnh đó, các kiến trúc phần mềm truyền thống theo mô hình nguyên khối (Monolithic Architecture) dần bộc lộ nhiều hạn chế, đặc biệt đối với các hệ thống có quy mô lớn và yêu cầu tính linh hoạt cao. Thay vào đó, kiến trúc **Microservices** đã và đang trở thành một xu hướng chủ đạo trong việc xây dựng các hệ thống phân tán hiện đại.

Xuất phát từ các yêu cầu thực tiễn trên, đề tài **“Xây dựng hệ thống mạng xã hội Bento Social áp dụng kiến trúc Microservices”** được lựa chọn nhằm nghiên cứu và áp dụng mô hình kiến trúc Microservices vào một bài toán cụ thể – hệ thống mạng xã hội – từ đó làm rõ các đặc trưng của hệ thống phân tán trong thiết kế và triển khai phần mềm.

## 1.2. Tổng quan về bài toán và mục tiêu đề tài

### 1.2.1. Đặc điểm bài toán hệ thống mạng xã hội

Hệ thống mạng xã hội là một dạng hệ thống thông tin có mức độ tương tác cao, trong đó người dùng có thể:

* Tạo và chia sẻ nội dung (bài viết, hình ảnh);
* Tương tác với nội dung thông qua các hành động như like, bình luận;
* Xây dựng mối quan hệ xã hội thông qua cơ chế theo dõi (follow);
* Nhận thông báo khi có các hoạt động liên quan.

Đặc điểm nổi bật của bài toán mạng xã hội là:

* Lượng truy cập lớn và không ổn định;
* Dữ liệu phát sinh liên tục;
* Nhiều thao tác cần được xử lý gần như thời gian thực;
* Yêu cầu cao về khả năng mở rộng và tính sẵn sàng của hệ thống.

Những đặc điểm này cho thấy bài toán mạng xã hội là một **bài toán điển hình để áp dụng kiến trúc hệ thống phân tán**, đặc biệt là kiến trúc Microservices.

### 1.2.2. Mục tiêu đề tài

Mục tiêu của đề tài là:

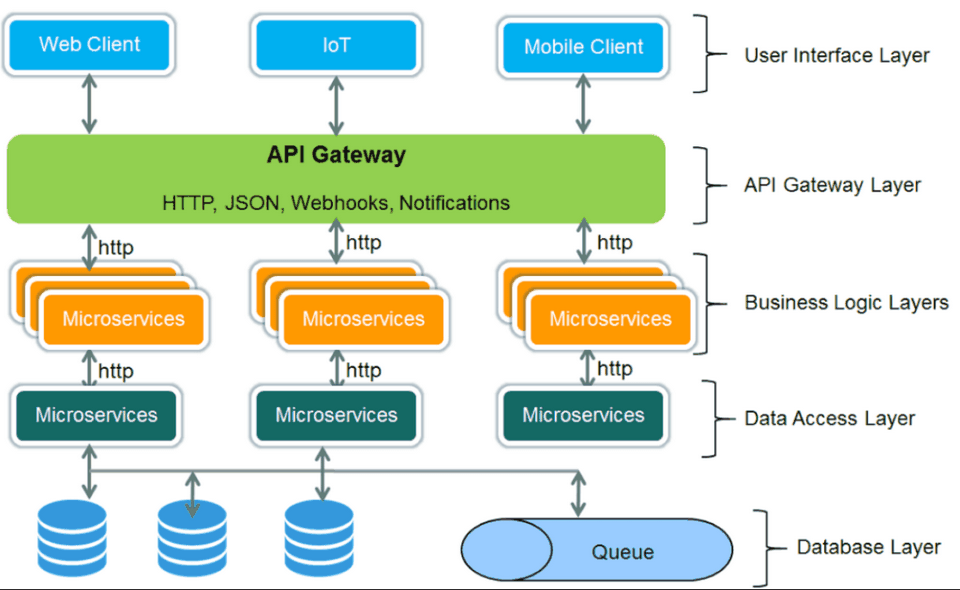
* Thiết kế và xây dựng một hệ thống mạng xã hội cơ bản mang tên **Bento Social**;
* Áp dụng kiến trúc **Microservices** và tư duy hệ thống phân tán trong thiết kế hệ thống;
* Minh họa cách tổ chức các thành phần hệ thống theo hướng **tách biệt, độc lập và sẵn sàng mở rộng**;
* Phân tích các luồng xử lý bất đồng bộ và event-driven trong hệ thống thông qua các chức năng tương tác xã hội.

Đề tài không đặt nặng việc xây dựng một sản phẩm hoàn chỉnh như các mạng xã hội thương mại, mà tập trung vào **việc áp dụng và làm rõ các nguyên lý của kiến trúc Microservices trong bối cảnh hệ thống phân tán**.

## 1.3. Tổng quan kiến trúc Microservices

### 1.3.1. Khái niệm kiến trúc Microservices

Microservice là một mô hình kiến trúc phần mềm phân tán được ứng dụng với mục đích phân chia thành các dịch vụ nhỏ. Nền tảng hoạt động hoàn toàn độc lập dựa trên các bước chạy và quản ly. Mỗi dịch vụ do Microservice cung cấp đều tập trung vào một chức năng cụ thể.



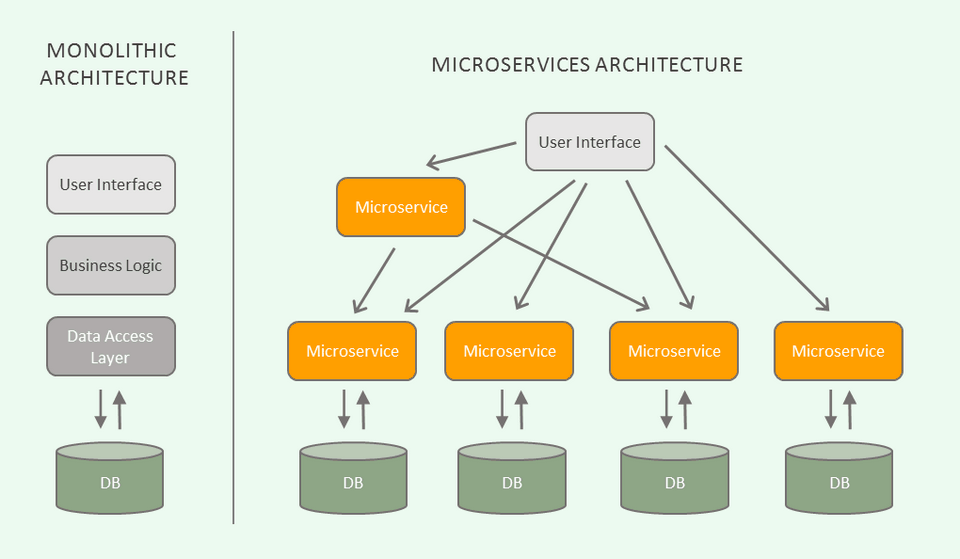
##### Hình 1.1 Mô hình kiến trúc Microservice

Công cụ có thể giao tiếp với các dịch vụ khác thông qua nhiều giao diện chuẩn như HTTP hoặc Message queues. Kiến trúc Microservice giúp cho việc phát triển và mở rộng ứng dụng trở nên linh hoạt hơn. Đồng thời, nền tảng còn tạo ra hệ thống có khả năng chịu lỗi tốt hơn.

### 1.3.2. Đặc trưng của Microservices

Kiến trúc Microservice có nhiều đặc điểm nổi bật, giúp các doanh nghiệp tăng cường khả năng mở rộng và giảm thiểu rủi ro khi vận hành:

* **Tính độc lập**:



##### Hình 1.2 Đặc trưng của Microservice

Sự độc lập của Microservice thể hiện ở một số khía cạnh quan trọng, điển hình như

* **Độc lập trong phát triển:** Mỗi dịch vụ Microservice được phát triển bởi một nhóm phát triển độc lập. Điều này giúp tăng tốc độ phát triển và giảm sự phụ thuộc giữa các phần của hệ thống.
* **Độc lập trong triển khai:** Các dịch vụ Microservice sẽ được triển khai một cách độc lập. Người dùng có thể cập nhật hoặc mở rộng một dịch vụ cơ bản mà không ảnh hưởng đến các dịch vụ khác trong hệ thống.
* **Độc lập trong quản lý:** Mỗi dịch vụ Microservice được ứng dụng công nghệ quản lý và mở rộng một cách độc lập. Đây là cách tối ưu hóa và thực hiện các thay đổi mà không ảnh hưởng đến các dịch vụ khác.
* **Độc lập trong ngôn ngữ và công nghệ:** Mỗi dịch vụ Microservice được xây dựng bằng ngôn ngữ lập trình và công nghệ khác nhau, tùy thuộc vào yêu cầu cụ thể của dịch vụ đó.
* **Tính chuyên biệt**

Tính chuyên biệt của Microservice nằm ở cách dịch vụ tập trung vào việc thực hiện một chức năng cụ thể hoặc phục vụ một mục tiêu kinh doanh rõ ràng. Người dùng có thể phân chia công việc cụ thể trong hệ thống để dễ dàng quản lý hơn.

Mỗi Microservice được thiết kế để hoàn thành một nhiệm vụ nhất định. Điển hình như: quản lý thành phần cụ thể của dữ liệu, xử lý giao dịch cơ bản, cung cấp dịch vụ cho người dùng và các chức năng khác.

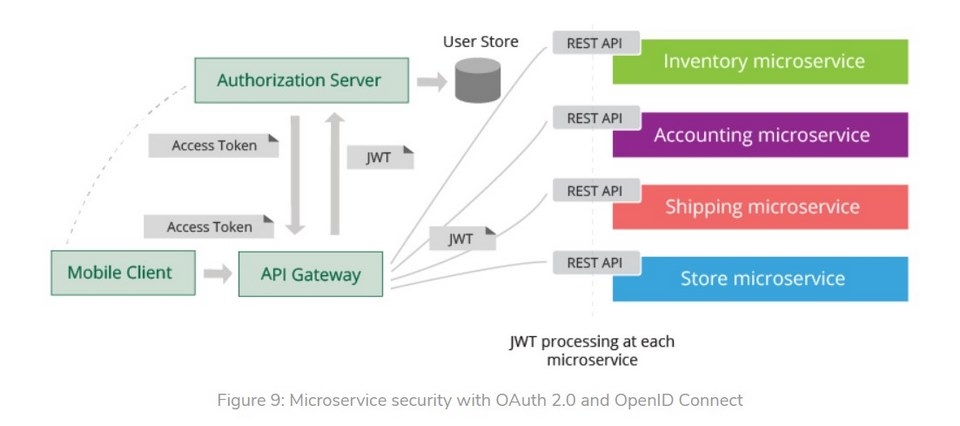
Tính chuyên biệt của nền tảng tạo ra sự linh hoạt khi người dùng phát triển và triển khai hệ thống. Ngoài ra, nó còn tạo điều kiện thuận lợi cho việc quản lý và bảo trì. Mặt khác, khi một dịch vụ cần cải thiện hoặc thay đổi thì người dùng chỉ cần tập trung vào một Microservice cụ thể. Hoạt động cải tiến không ảnh hưởng đến các phần khác của hệ thống

* **Khả năng phòng lỗi**

Kiến trúc microservice sinh ra là để dành cho các hệ thống từ lớn đến vô cùng lớn. Nó áp dụng phương pháp chia để trị, phương pháp này giúp việc áp dụng các công cụ, kỹ thuật cho việc giám sát,phòng chống lỗi phần mềm, lỗi hệ thống hiệu quả.

Khi một thành phần trong hệ thống bị lỗi, nó có thể được thay thế bằng các thành phần dự phòng một cách dễ dàng, trong quá trình thay thế thành phần bị lỗi, các thành phần khác vẫn hoạt động bình thường, do vậy hoạt động của toàn bộ hệ thống sẽ không hoặc ít bị gián đoạn.

### 1.3.3. Đánh giá ưu và nhược điểm của Microservices



##### Hình 1.3 Kiến trúc bảo mật trong hệ thống Microservices sử dụng JWT

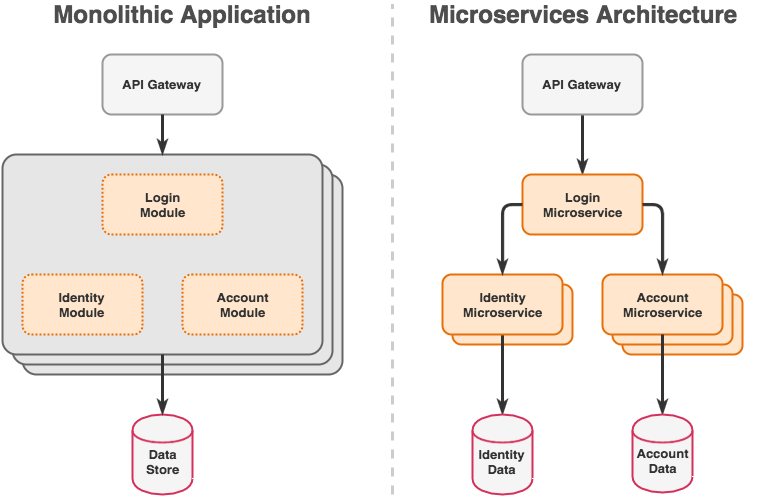
**Ưu điểm**

* **Tính linh hoạt và mở rộng:** Mô hình Microservice cho phép mỗi dịch vụ được phát triển, triển khai và mở rộng một cách độc lập. Từ đó tạo ra nhiều điều kiện thuận lợi cho việc tối ưu hóa tài nguyên và hiệu suất của hệ thống.
* **Khả năng chịu lỗi:** Thiết kế phân tán của Microservice làm giảm tác động của sự cố và nâng cao khả năng phục hồi của hệ thống.
* **Sự tập trung:** Mỗi dịch vụ Microservice chịu trách nhiệm cho một chức năng chính của phần mềm. Nguyên tắc này tạo sự rõ ràng và về trách nhiệm, cách sử dụng, quản lý.
* **Công nghệ đa dạng:** Mô hình Microservice tích hợp nhiều công nghệ khác nhau cho từng dịch vụ, tối ưu hóa khả năng sử dụng công nghệ phù hợp cho mỗi trường hợp cụ thể.

**Nhược điểm**

* **Phức tạp trong quản lý:** Mô hình Microservice đòi hỏi việc quản lý nhiều dịch vụ độc lập. Người dùng cần chú ý đến cách xử lý các vấn đề liên quan đến hoạt động giao tiếp giữa các dịch vụ.
* **Đồng nhất hóa và chuẩn hóa:** Việc đảm bảo sự đồng nhất hóa và chuẩn hóa giữa các dịch vụ đòi hỏi khả năng quản lý cẩn thận để tránh sự phân tán.
* **Chi phí phát triển ban đầu:** Việc phân chia ứng dụng thành các dịch vụ nhỏ có thể tạo ra chi phí phát triển ban đầu cao hơn so với việc phát triển một ứng dụng lớn đơn nhất.
* **Giao tiếp và mạng lưới:** Cách thức hoạt động của hệ thống Microservice yêu cầu tiêu chuẩn sử dụng mạng lưới tin cậy, hiệu quả để đảm bảo tiện ích giao tiếp giữa các dịch vụ diễn ra mượt mà.

## 1.4. So sánh kiến trúc Monolithic và Microservices



##### Hình 1.4 So sánh kiến trúc Monolithic và kiến trúc Microservices

### 1.4.1. Kiến trúc Monolithic

Kiến trúc Monolithic tổ chức toàn bộ hệ thống trong một khối thống nhất, trong đó các thành phần nghiệp vụ, xử lý dữ liệu và giao diện thường được triển khai chung trong một ứng dụng duy nhất. Mô hình này có ưu điểm là dễ phát triển ban đầu và đơn giản trong triển khai.

Tuy nhiên, khi hệ thống phát triển lớn, kiến trúc Monolithic bộc lộ nhiều hạn chế như:

* Khó mở rộng theo từng chức năng;
* Khó bảo trì khi hệ thống trở nên phức tạp;
* Một lỗi nhỏ có thể ảnh hưởng đến toàn bộ hệ thống.

### 1.4.2. Kiến trúc Microservices

Ngược lại, Microservices cho phép chia hệ thống thành các dịch vụ nhỏ, độc lập, mỗi dịch vụ tập trung vào một nghiệp vụ cụ thể. Việc thay đổi hoặc mở rộng một dịch vụ không ảnh hưởng trực tiếp đến các dịch vụ khác.

So với Monolithic, Microservices có ưu thế rõ rệt trong các hệ thống:

* Có quy mô lớn;
* Có yêu cầu cao về mở rộng và độ sẵn sàng;
* Có nhiều nhóm phát triển song song.

### ****1.4.3. Bảng so sánh kiến trúc Monolithic và Microservices****

Để làm rõ hơn sự khác biệt giữa hai mô hình kiến trúc, bảng so sánh dưới đây tổng hợp các tiêu chí quan trọng trong thiết kế hệ thống phân tán:

| **Tiêu chí** | **Monolithic** | **Microservices** |
| --- | --- | --- |
| Cấu trúc hệ thống | Một khối thống nhất | Nhiều dịch vụ độc lập |
| Khả năng mở rộng | Mở rộng toàn hệ thống | Mở rộng theo từng dịch vụ |
| Mức độ phụ thuộc | Phụ thuộc chặt chẽ | Liên kết lỏng lẻo |
| Triển khai | Triển khai một lần | Triển khai độc lập |
| Chịu lỗi | Lỗi dễ lan rộng | Lỗi được cô lập |
| Phù hợp hệ thống phân tán | Thấp | Cao |

## 1.5. Định hướng áp dụng Microservices cho hệ thống Bento Social

Trong đề tài này, kiến trúc Microservices được áp dụng cho hệ thống Bento Social theo định hướng **Microservices-ready**, cụ thể:

* Hệ thống được tách biệt rõ ràng giữa Frontend và Backend;
* Các chức năng tương tác xã hội (like, comment, follow, notification) được thiết kế theo hướng **event-driven**;
* Các tác vụ thời gian thực được xử lý thông qua cơ chế truyền thông bất đồng bộ;
* Hệ thống sẵn sàng cho việc tách các chức năng thành các dịch vụ độc lập khi mở rộng trong tương lai.

Việc áp dụng Microservices không chỉ giúp hệ thống Bento Social dễ mở rộng mà còn minh họa rõ cách tiếp cận của một **hệ thống phân tán hiện đại**.

## 1.6. Tổng quan công nghệ sử dụng

Để hiện thực hóa kiến trúc Microservices trong đề tài, hệ thống Bento Social sử dụng các công nghệ hiện đại, phù hợp với môi trường hệ thống phân tán:

* **NestJS**: Nền tảng Backend hỗ trợ tổ chức module rõ ràng, phù hợp với kiến trúc Microservices-ready;
* **Next.js**: Framework Frontend hỗ trợ render phía server và cải thiện trải nghiệm người dùng;
* **PostgreSQL**: Hệ quản trị cơ sở dữ liệu quan hệ, đảm bảo tính toàn vẹn và nhất quán dữ liệu;
* **Redis**: Được sử dụng cho cache và cơ chế Pub/Sub, hỗ trợ xử lý sự kiện và thông báo thời gian thực;
* **Docker**: Công cụ container hóa, hỗ trợ triển khai và mở rộng hệ thống trong môi trường phân tán.

Chi tiết về cách cài đặt và triển khai các công nghệ này sẽ được trình bày trong các chương sau.

# II. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 2.1. Kiến trúc hệ thống tổng quát

Hệ thống Bento Social được thiết kế theo mô hình **Client–Server hiện đại**, trong đó kiến trúc được tổ chức theo hướng **tách biệt hoàn toàn giữa Frontend và Backend (Decoupled Architecture)**. Cách tiếp cận này giúp hệ thống giảm sự phụ thuộc giữa các thành phần, đồng thời tạo điều kiện thuận lợi cho việc mở rộng và bảo trì trong môi trường hệ thống phân tán.

### 2.1.1. Mô hình tổng thể Client–Server

Về mặt tổng thể, hệ thống Bento Social bao gồm các thành phần chính sau:

* **Client (Frontend – Next.js):**

Chịu trách nhiệm hiển thị giao diện và xử lý tương tác người dùng. Người dùng thao tác thông qua trình duyệt web, các hành động như đăng nhập, đăng bài, bình luận hay theo dõi sẽ được gửi về phía máy chủ dưới dạng các yêu cầu HTTP thông qua REST API.

* **API Server (Backend – NestJS):**

Đóng vai trò trung tâm xử lý nghiệp vụ của hệ thống. API Server tiếp nhận yêu cầu từ Client, thực hiện xác thực người dùng, xử lý logic nghiệp vụ và tương tác với các thành phần lưu trữ dữ liệu.

* **Data Layer:**
  + **PostgreSQL:** Hệ quản trị cơ sở dữ liệu quan hệ, đảm nhiệm lưu trữ dữ liệu bền vững như thông tin người dùng, bài viết, bình luận và các mối quan hệ theo dõi.
  + **Redis:** Được sử dụng cho các tác vụ yêu cầu tốc độ cao và xử lý bất đồng bộ, đặc biệt là cơ chế Pub/Sub cho các chức năng thời gian thực như thông báo.

Mô hình này đảm bảo tính linh hoạt trong phát triển, cho phép các thành phần giao diện và xử lý nghiệp vụ được phát triển độc lập, đồng thời sẵn sàng cho việc mở rộng theo hướng Microservices trong tương lai.

### 2.1.2. Định hướng Microservices-ready và Event-driven

Mặc dù trong phạm vi đồ án môn học, hệ thống Bento Social chưa được tách thành nhiều microservice độc lập hoàn toàn, nhưng kiến trúc được thiết kế theo định hướng **Microservices-ready**, tức là sẵn sàng cho việc phân tách các chức năng thành các dịch vụ độc lập khi hệ thống mở rộng.

Một số đặc điểm thể hiện định hướng này bao gồm:

* Tách biệt rõ ràng giữa các nhóm chức năng nghiệp vụ;
* Hạn chế sự phụ thuộc chặt chẽ giữa các thành phần xử lý;
* Áp dụng cơ chế xử lý **bất đồng bộ và hướng sự kiện (event-driven)** cho các thao tác có tần suất cao.

Việc sử dụng Redis với cơ chế Pub/Sub cho phép hệ thống xử lý các sự kiện như like, comment, follow một cách độc lập với luồng xử lý chính, từ đó nâng cao khả năng mở rộng và phản ánh đúng đặc trưng của hệ thống phân tán.

## 2.2. Phân tích và thiết kế chi tiết hệ thống

Phần này tập trung vào việc phân tích chi tiết các chức năng nghiệp vụ và thiết kế cấu trúc dữ liệu của hệ thống Bento Social. Nội dung được trình bày dựa trên các tác nhân tham gia hệ thống, các ca sử dụng (Use Case), luồng xử lý (Sequence Diagram) và thiết kế cơ sở dữ liệu.

### 2.2.1. Phân tích tác nhân và chức năng hệ thống

**Mô tả tổng quan về hệ thống**

Hệ thống **Bento Social** là một nền tảng mạng xã hội (Social Network Service) được xây dựng nhằm tạo ra môi trường kết nối, chia sẻ thông tin đa phương tiện giữa người dùng trên internet. Đối với người quản trị, hệ thống cung cấp các công cụ để quản lý người dùng, kiểm duyệt nội dung và theo dõi sự phát triển của cộng đồng. Đối với người dùng cuối (User), website/ứng dụng là nơi để họ tự do sáng tạo nội dung (bài viết, hình ảnh), thể hiện quan điểm cá nhân, tương tác với cộng đồng thông qua các hành động like, bình luận, và xây dựng mạng lưới kết nối cá nhân thông qua tính năng theo dõi (follow).

Hệ thống được thiết kế theo kiến trúc hiện đại, tách biệt giữa Web Client (Frontend) và API Server (Backend), đảm bảo tính mở rộng và trải nghiệm người dùng mượt mà.

**Xác định người dùng**

Đối với hệ thống Bento Social, có 3 nhóm người dùng chính tương tác với hệ thống:

Hệ thống Bento Social được phân quyền thành ba nhóm đối tượng chính, bao gồm: **Khách (Guest)**, **Thành viên (Authenticated User)** và **Quản trị viên (Admin)**. Việc phân chia này giúp hệ thống kiểm soát quyền truy cập, đảm bảo an toàn dữ liệu và tổ chức các chức năng nghiệp vụ một cách rõ ràng.

* **Nhóm khách (Guest - Người dùng chưa đăng nhập):** Khi truy cập vào hệ thống, người dùng này chưa được xác thực danh tính. Họ sẽ bị giới hạn quyền truy cập, chủ yếu nhìn thấy:
* Trang Landing Page giới thiệu về nền tảng.
* Các chức năng Đăng ký (Register) và Đăng nhập (Login) để tham gia vào hệ thống.
* (Tùy chỉnh) Có thể xem một số bài viết công khai hoặc trang profile công khai mà không cần đăng nhập (tùy thuộc vào chính sách bảo mật của hệ thống).
* **Nhóm thành viên (Authenticated User - Người dùng đã đăng nhập):** Đây là đối tượng sử dụng chính của hệ thống. Sau khi đăng nhập thành công, họ có toàn quyền tham gia các hoạt động mạng xã hội:
* **Tạo nội dung:** Đăng bài viết (text, hình ảnh), chọn chủ đề (Topic) cho bài viết.
* **Tương tác:** Thả tim (Like), Bình luận (Comment), Trả lời bình luận (Reply), Lưu bài viết (Save).
* **Kết nối:** Theo dõi (Follow) người dùng khác để cập nhật newsfeed, xem thông tin profile của người khác.
* **Cá nhân:** Quản lý profile, cập nhật ảnh đại diện/cover, xem thông báo cá nhân.
* **Nhóm quản trị viên (Administrator):** Là người vận hành hệ thống, có quyền hạn cao nhất. Họ đăng nhập vào khu vực quản trị riêng (nếu có) hoặc có quyền đặc biệt trên giao diện chính để:
* Quản lý danh sách người dùng (User Management): Xem danh sách, khóa tài khoản (Ban), kích hoạt lại tài khoản.
* Quản lý nội dung (Content Moderation): Duyệt các bình luận (nếu cấu hình duyệt trước), xử lý các nội dung vi phạm, quản lý các Chủ đề (Topics).

**Yêu cầu về chức năng**

Qua việc phân tích nghiệp vụ mạng xã hội và cơ sở dữ liệu hiện tại, các chức năng chính của hệ thống được phân chia như sau:

|  |  |
| --- | --- |
| **User (Thành viên)** | **Admin (Quản trị viên)** |
| **Đăng ký / Đăng nhập** | **Đăng nhập quản trị viên** |
| **Quản lý Profile**: Cập nhật Avatar, Cover, Bio, Website | **Quản lý Dashboard**: Xem thống kê tổng quan (số user, bài viết) |
| **Đăng bài (Create Post)**: Viết text, upload ảnh, chọn Topic | **Quản lý Người dùng (Users)**: Xem danh sách, Khóa (Ban)/Mở khóa user |
| **Tương tác**: Like bài viết, Lưu bài viết (Bookmark) | **Quản lý Chủ đề (Topics)**: Thêm/Sửa/Xóa các Topic thảo luận |
| **Bình luận**: Viết bình luận, Reply bình luận | **Quản lý Bình luận**: Duyệt (Approve) hoặc Xóa/Spam bình luận |
| **Social Graph**: Follow / Unfollow người dùng khác | **Cấu hình hệ thống**: Các cài đặt chung |
| **Newsfeed**: Xem bài viết từ người mình follow hoặc bài nổi bật |  |
| **Thông báo (Notifications)**: Nhận thông báo khi có tương tác mới |  |

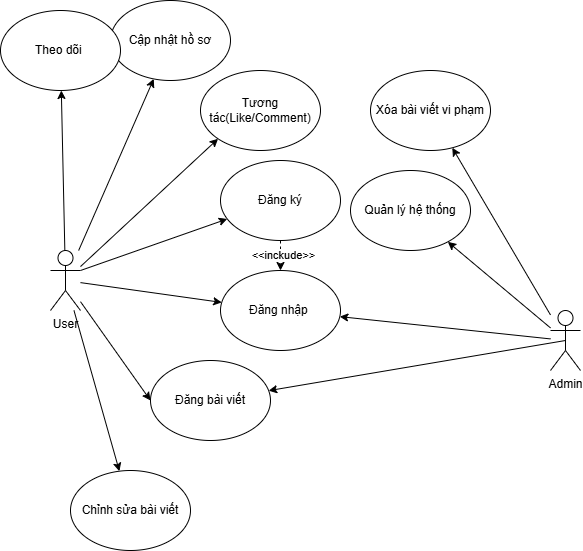
**Yêu cầu về phi chức năng**

* Về giao diện:
* Giao diện dễ sử dụng, thân thiện với người dùng
* Thanh menu đơn giản
* Font chữ dễ nhìn
* Về tính tương thích:
* Tương thích với các trình duyệt phổ biến
* Tương thích trên màn hình của nhiều thiết bị : pc, laptop, tablet, mobile
* Về tính bảo mật:
* Có thông tin liên hệ
* An toàn bảo mật dữ liệu

### 2.2.2. Biểu đồ Use Case

Biểu đồ Use Case được sử dụng nhằm mô tả các chức năng mà từng nhóm tác nhân có thể thực hiện cũng như mối quan hệ giữa người dùng và hệ thống. Các biểu đồ này giúp làm rõ phạm vi chức năng và là cơ sở cho việc thiết kế chi tiết các luồng xử lý sau này.

#### 2.2.2.1. Use Case Tổng quát



##### Hình 2.1 Biểu đồ Use Case Tổng quát

**Các thành phần trong Usecase tổng quát gồm:**

**Nhóm chức năng dành cho User:**

Người dùng có thể thực hiện các hành động sau:

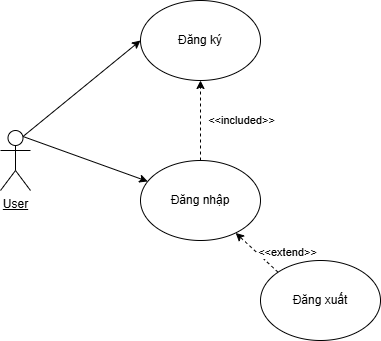
* **Theo dõi:** Theo dõi người dùng khác hoặc các chuyên mục.
* **Cập nhật hồ sơ:** Thay đổi thông tin cá nhân.
* **Tương tác (Like/Comment):** Bày tỏ cảm xúc hoặc để lại bình luận trên bài viết.
* **Đăng ký:** Tạo tài khoản mới.
* **Đăng nhập:** Truy cập vào hệ thống bằng tài khoản đã có.
* **Đăng bài viết:** Tạo nội dung mới trên nền tảng.
* **Chỉnh sửa bài viết:** Sửa đổi nội dung bài viết đã đăng trước đó.

**Nhóm chức năng dành cho Admin:**

Quản trị viên có các quyền hạn đặc biệt:

* **Quản lý hệ thống:** Giám sát hoạt động chung của toàn bộ nền tảng.
* **Xóa bài viết vi phạm:** Gỡ bỏ các nội dung không phù hợp hoặc vi phạm quy định.
* **Đăng nhập:** Truy cập vào quyền quản trị.
* **Đăng bài viết:** Admin cũng có thể đăng nội dung như một người dùng thông thường.

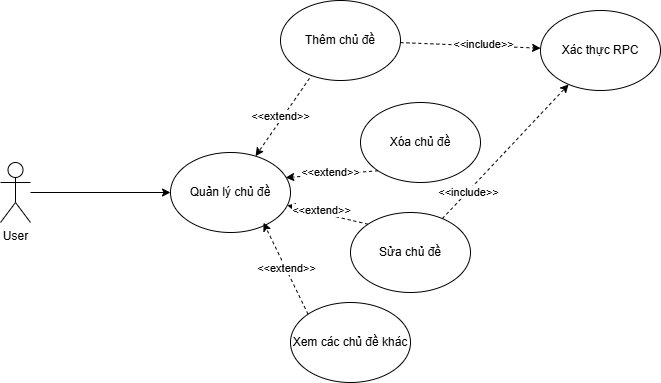
#### 2.2.2.2. Usecase Đăng nhập



##### Hình 2.2 Biểu đồ Use Case Đăng nhập

|  |  |
| --- | --- |
| **Tên** | **Đăng nhập** |
| **Tác nhân** | Cá nhân người dùng |
| **Mô tả mục tiêu** | Người dùng đăng nhập vào hệ thống thành công |
| **Yêu cầu liên quan** | Usecase Đăng ký |
| **Tiền điều kiện** | Người dùng đã có tài khoản  Người dùng phải chọn chức năng đăng nhập |
| **Mô tả** | 1. Người dùng nhập Email, password 2. Người dùng chọn button Log In 3. Hệ thống kiểm tra thông tin đầu vào |
| **Hậu điều kiện** | Người dùng đăng nhập thành công, hệ thống hiển thị giao diện trang chủ |
| **Biến thể** | Người dùng muốn tạo tài khoản mới -> Usecase Đăng ký |
| **Ngoại lệ** | TH1: Người dùng nhập thiếu email hoặc password   1. Hệ thống báo nhập thiếu email hoặc password 2. Quay lại bước 1   TH2: Người dùng nhập sai email hoặc password   1. Hệ thống báo nhập sai email hoặc password 2. Quay lại bước 1 |

#### 2.2.2.3. Use Case Đăng bài viết

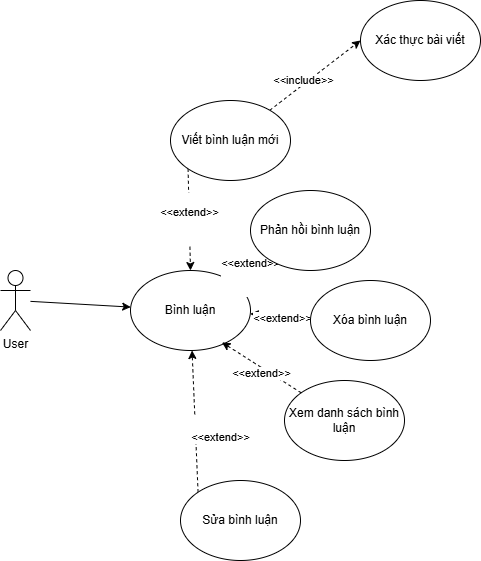


##### Hình 2.3 Biểu đồ Use Case Đăng bài viết

Chức năng cốt lõi giúp người dùng chia sẻ nội dung.

|  |  |
| --- | --- |
| **Tên** | **Đăng bài viết mới** |
| **Tác nhân** | Thành viên (Authenticated User) |
| **Mô tả mục tiêu** | Người dùng đăng tải một nội dung (văn bản hoặc hình ảnh) lên hệ thống |
| **Yêu cầu liên quan** | Xem Newsfeed, Upload ảnh |
| **Tiền điều kiện** | Người dùng đã đăng nhập thành công |
| **Mô tả** | 1. Trên giao diện chính, người dùng chọn chức năng "Create Post". 2. Hệ thống hiển thị popup/khung soạn thảo. 3. Người dùng nhập nội dung văn bản (Content) 4. (Tùy chọn) Người dùng upload hình ảnh đính kèm. 5. Người dùng chọn chủ đề (Topic) cho bài viết. 6. Người dùng nhấn button **Post**. 7. Hệ thống lưu bài viết và liên kết với tài khoản người đăng. |
| **Hậu điều kiện** | Bài viết hiển thị trên trang cá nhân của người dùng và Newsfeed của những người theo dõi họ. |
| **Biến thể** | **Đăng bài không có ảnh**: Người dùng chỉ nhập text và bỏ qua bước chọn ảnh. |
| **Ngoại lệ** | **TH1: Upload ảnh thất bại (File quá lớn/sai định dạng)**   1. Hệ thống thông báo lỗi upload. 2. Người dùng chọn lại ảnh khác.   **TH2: Nội dung rỗng**  Hệ thống disable nút Post hoặc báo lỗi yêu cầu nhập nội dung. |

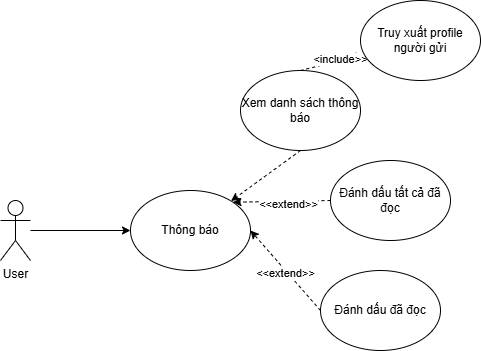
#### 2.2.2.4. Use Case Bình luận bài viết



##### Hình 2.4 Biểu đồ Use Case Bình luận bài viết

|  |  |
| --- | --- |
| **Tên** | **Bình luận bài viết** |
| **Tác nhân** | Thành viên (Authenticated User) |
| **Mô tả mục tiêu** | Người dùng để lại ý kiến dưới bài viết của mình hoặc người khác |
| **Yêu cầu liên quan** | Xem bài viết, Reply bình luận |
| **Tiền điều kiện** | Người dùng đã đăng nhập và đang xem chi tiết một bài viết |
| **Mô tả** | 1. Người dùng cuộn đến phần bình luận của bài viết.  2. Người dùng nhập nội dung vào ô "Write a comment...".  3. Người dùng nhấn Enter hoặc nút Gửi.  4. Hệ thống kiểm tra nội dung (không được rỗng).  5. Hệ thống lưu bình luận.  6. Hệ thống gửi thông báo (Notification) đến chủ bài viết. |
| **Hậu điều kiện** | Bình luận hiển thị ngay lập tức dưới bài viết. Số lượng bình luận của bài viết tăng lên 1. |
| **Biến thể** | **Trả lời bình luận (Reply)**: Người dùng bấm "Reply" vào một bình luận có sẵn -> Hệ thống ghi nhận đây là comment con (có  parentId). |
| **Ngoại lệ** | **TH1: Bài viết đã bị xóa**   1. Hệ thống báo lỗi "Post not found". 2. Làm mới trang để cập nhật trạng thái. |

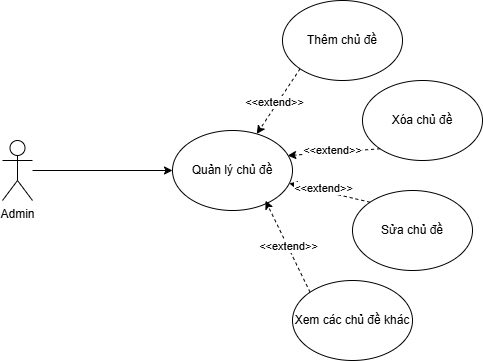
#### 2.2.2.5. Use Case Xem thông báo



##### Hình 2.5 Biểu đồ Use Case Xem thông báo

|  |  |
| --- | --- |
| **Tên** | **Xem thông báo** |
| **Tác nhân** | Thành viên (Authenticated User) |
| **Mô tả mục tiêu** | Người dùng xem danh sách các thông báo mới nhất về các hoạt động liên quan đến mình (có người like, comment, follow). |
| **Yêu cầu liên quan** | Đăng nhập |
| **Tiền điều kiện** | Người dùng đã đăng nhập vào hệ thống. |
| **Mô tả** | **Luồng hiển thị:**  1. Người dùng nhấn vào biểu tượng chuông báo (Notification Icon).  2. Hệ thống tải danh sách thông báo, sắp xếp theo thời gian mới nhất.  3. Hệ thống hiển thị danh sách gồm: Avatar người tương tác, nội dung hành động (liked, replied, followed), thời gian.  **Luồng xử lý sự kiện (Backend Automatic):**  1. Khi User A thực hiện hành động (Like Post, Comment, Follow User B).  2. Hệ thống bắn sự kiện (Event) qua Redis Pub/Sub.  3. Notification Service nhận sự kiện, tạo bản ghi Notification trong database cho User B. |
| **Hậu điều kiện** | Người dùng nhìn thấy danh sách thông báo. Các thông báo chưa đọc được đánh dấu rõ ràng. |
| **Biến thể** | **Đọc thông báo:** Người dùng click vào một thông báo -> Hệ thống đánh dấu là "Đã đọc" (isRead = true) và điều hướng đến bài viết/profile tương ứng.  **Đọc tất cả:** Người dùng chọn "Mark all as read" -> Hệ thống cập nhật tất cả thông báo thành đã đọc. |
| **Ngoại lệ** | **Không có thông báo nào:** Hệ thống hiển thị trạng thái trống (Empty State). |

#### 2.2.2.6. Use Case Quản lý Chủ đề



##### Hình 2.6 Biểu đồ Use Case Quản lý chủ đề

|  |  |
| --- | --- |
| **Tên** | **Quản lý Chủ đề (Topics)** |
| **Tác nhân** | Quản trị viên (Admin) |
| **Mô tả mục tiêu** | Admin thêm, sửa, hoặc xóa các chủ đề (Topic) để người dùng có thể gắn thẻ (tag) vào bài viết. |
| **Yêu cầu liên quan** | Đăng nhập Admin |
| **Tiền điều kiện** | Người dùng đăng nhập với quyền Admin (UserRole.ADMIN). |
| **Mô tả** | **1. Xem danh sách:**  - Admin truy cập trang quản lý Topic.  - Hệ thống hiển thị danh sách Topic kèm màu sắc (Color) và số lượng bài viết (Post Count).  **2. Tạo mới (Create):**  - Admin nhập Tên Topic (Name) và Mã màu (Hex Color).  - Hệ thống kiểm tra trùng tên.  - Hệ thống lưu Topic mới.  **3. Cập nhật (Update):**  - Admin sửa tên hoặc màu sắc của Topic.  **4. Xóa (Delete):**  - Admin chọn xóa Topic không còn phù hợp. |
| **Hậu điều kiện** | Danh sách Topic được cập nhật. Người dùng cuối có thể chọn Topic mới khi đăng bài. |
| **Biến thể** | N/A |
| **Ngoại lệ** | **TH1: Trùng tên Topic**  1. Hệ thống báo lỗi  ErrTopicNameAlreadyExists  .2. Yêu cầu nhập tên khác.  **TH2: Mã màu không hợp lệ**  1. Hệ thống báo lỗi  ErrTopicColorInvalid (Màu phải là mã Hex hợp lệ, VD: #008000). |

#### 2.2.2.7. Use Case Theo dõi người dùng



##### Hình 2.7 Biểu đồ Use Case Theo dõi người dùng

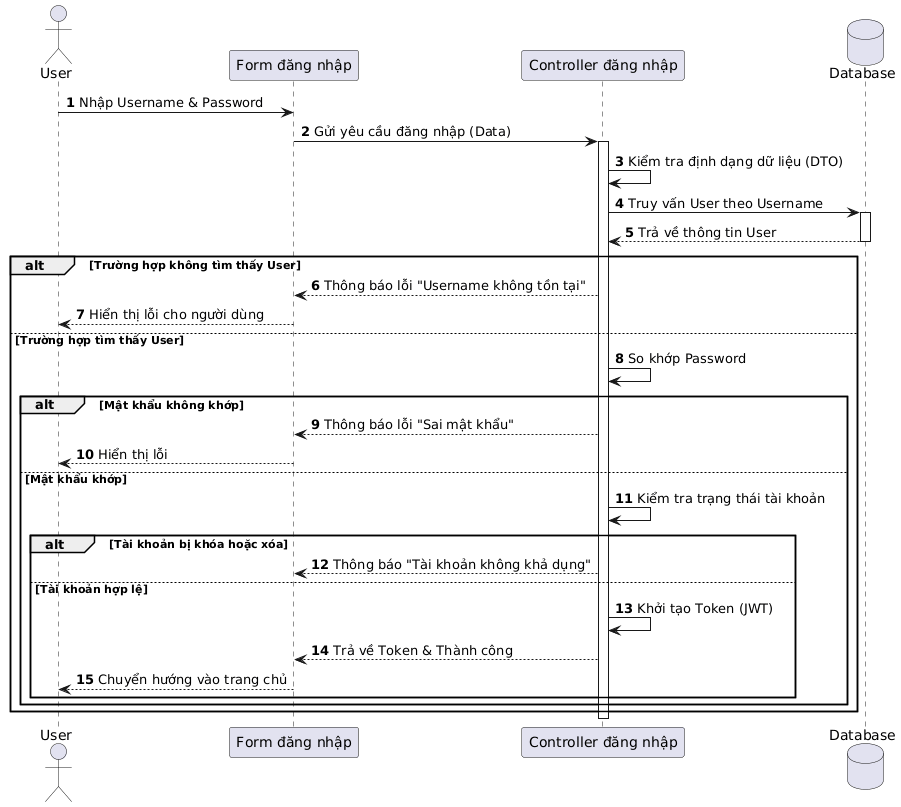
Chức năng cho phép người dùng kết nối với những người khác để cập nhật hoạt động của họ.

|  |  |
| --- | --- |
| **Tên** | **Theo dõi người dùng** |
| **Tác nhân** | Thành viên (Authenticated User) |
| **Mô tả mục tiêu** | Người dùng thực hiện hành động "Theo dõi" một người dùng khác để nhận các cập nhật bài viết của họ trên Newsfeed. |
| **Yêu cầu liên quan** | Đăng nhập, Xem Profile người khác, Unfollow |
| **Tiền điều kiện** | 1. Người dùng đã đăng nhập.2. Người dùng đang xem Profile hoặc danh sách gợi ý của một người dùng khác (User B). |
| **Mô tả** | 1. Người dùng nhấn nút **Follow** trên giao diện.  2. Hệ thống gửi yêu cầu  FollowDTO chứa followerId (người nhấn) và  followingId (người được follow) lên server.  3. Server thực hiện các kiểm tra:  - followerId và  followingId phải khác nhau (Không thể tự follow chính mình).  - Kiểm tra xem User B có tồn tại trong hệ thống không.  - Kiểm tra xem User A đã follow User B trước đó chưa.  4. Nếu hợp lệ, Server lưu bản ghi vào bảng  followers.  5. Server publish sự kiện  FollowedEvent.  6. Server Notification nhận sự kiện và tạo thông báo cho User B. |
| **Hậu điều kiện** | - User A chính thức theo dõi User B.  - Số lượng follower của User B tăng lên 1.  - Số lượng following của User A tăng lên 1.  Nút trạng thái chuyểntừ **Follow** sang **Following** hoặc **Unfollow**. |
| **Biến thể** | **Bỏ theo dõi (Unfollow):**  - Người dùng nhấn lại vào nút trạng thái (đang là Following/Unfollow).  - Hệ thống xóa bản ghi trong bảng  followers  .- Các chỉ số đếm giảm đi 1. |
| **Ngoại lệ** | **TH1: Tự follow chính mình**  Hệ thống báo lỗi  ErrFollowYourself  **TH2: Người dùng đích không tồn tại**  Hệ thống báo lỗi ErrNotFound  **TH3: Đã follow rồi nhưng lại gửi yêu cầu follow tiếp**  Hệ thống báo lỗi  ErrAlreadyFollowed |

### 2.2.3. Biểu đồ Sequence

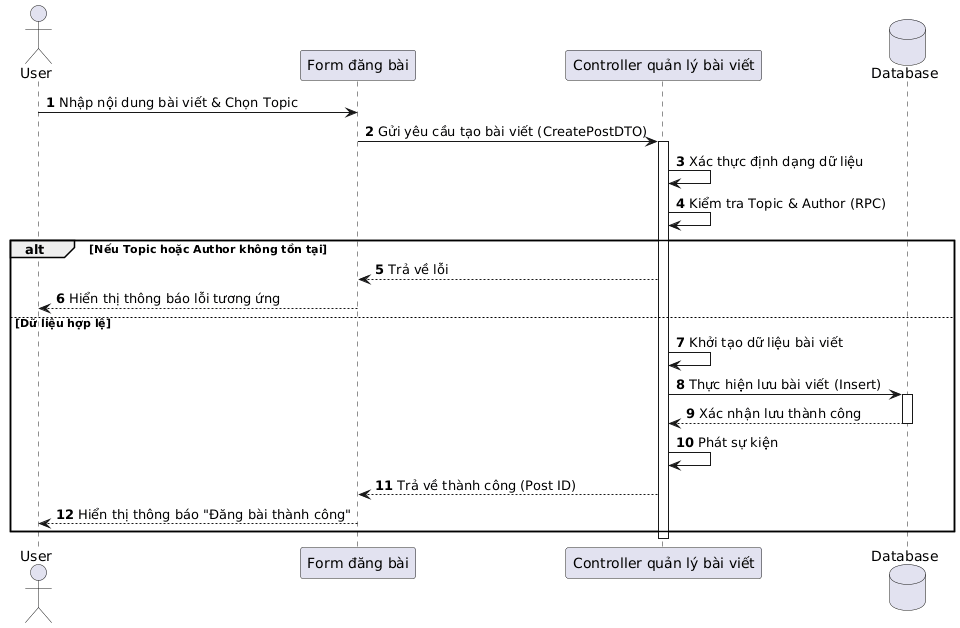
Biểu đồ tuần tự (Sequence Diagram) được sử dụng để mô tả chi tiết luồng trao đổi dữ liệu giữa các thành phần trong hệ thống theo trục thời gian. Thông qua các biểu đồ này, quá trình xử lý nghiệp vụ của hệ thống được thể hiện một cách trực quan và rõ ràng.

#### 2.2.3.1. Sequence Đăng nhập



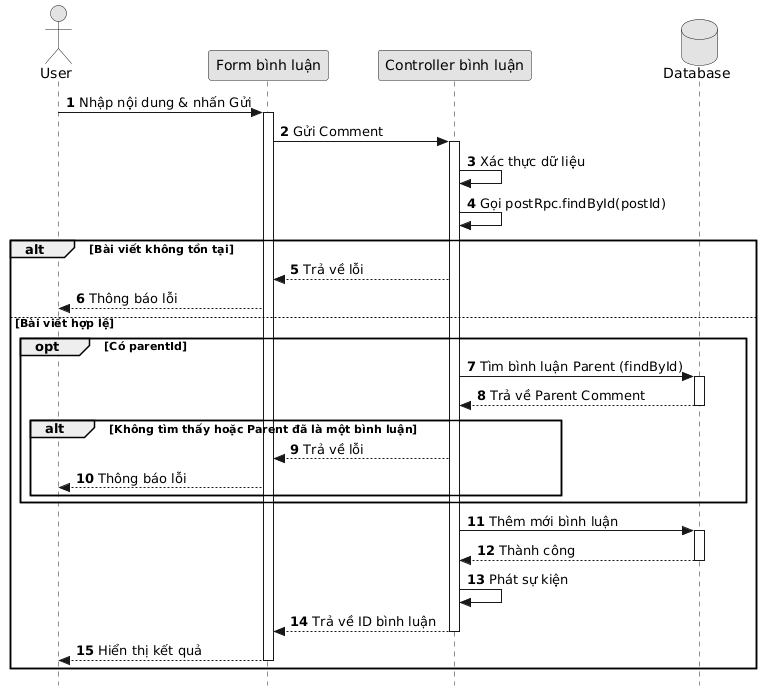
##### Hình 2.8 Biểu đồ Sequence Đăng nhập

#### 2.2.3.2. Sequence Đăng bài viết



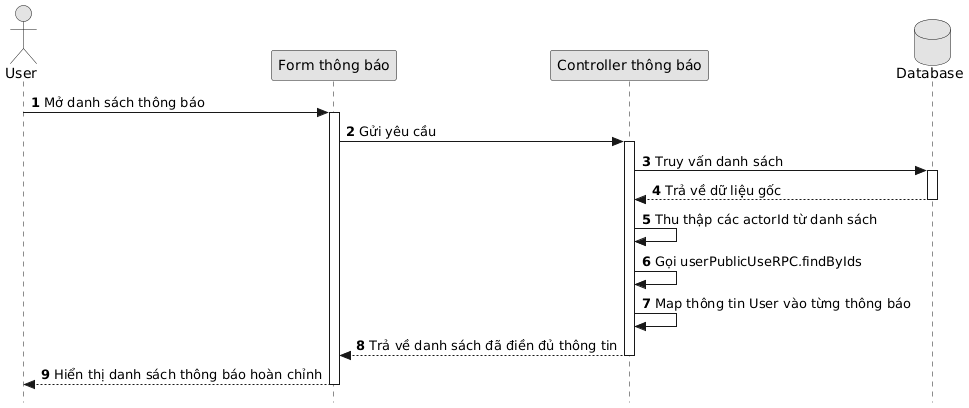
##### Hình 2.9 Biểu đồ Sequence Đăng bài viết

#### 2.2.3.3. Sequence Bình luận



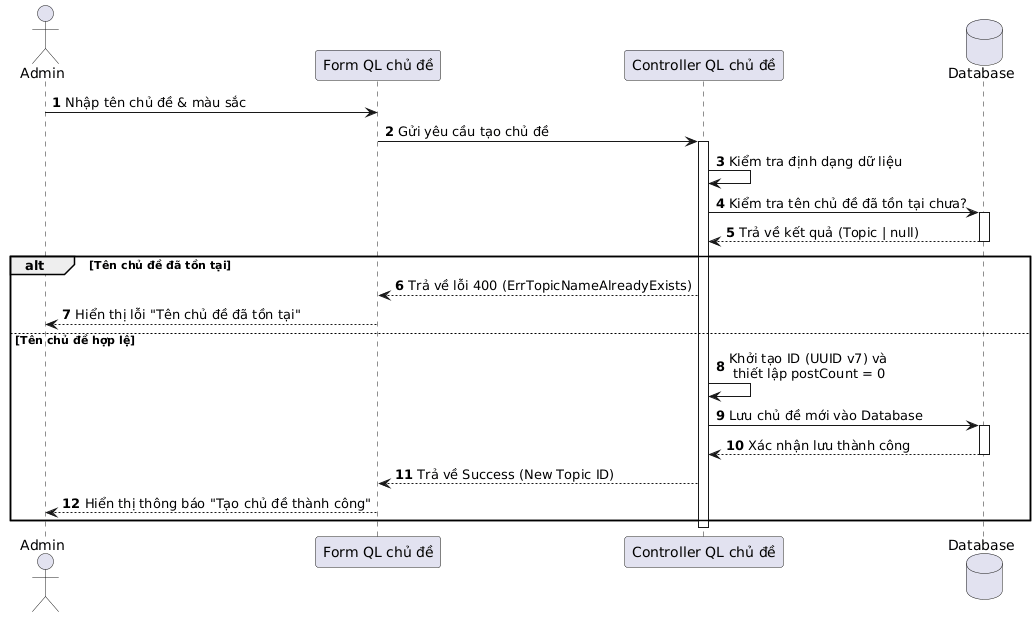
##### Hình 2.10 Biểu đồ Sequence Bình luận

#### 2.2.3.4. Sequence Thông báo



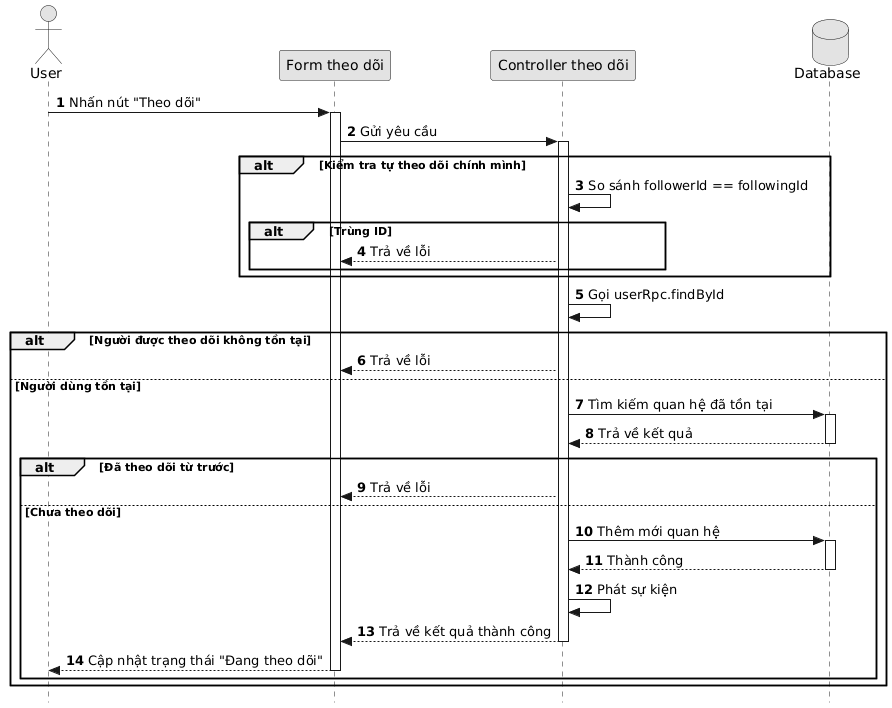
##### Hình 2.11 Biểu đồ Sequence Thông báo

#### 2.2.3.5. Sequence Quản lý chủ đề



##### Hình 2.12 Biểu đồ Sequence Quản lý chủ đề

#### 2.2.3.6. Sequence Theo dõi người dùng

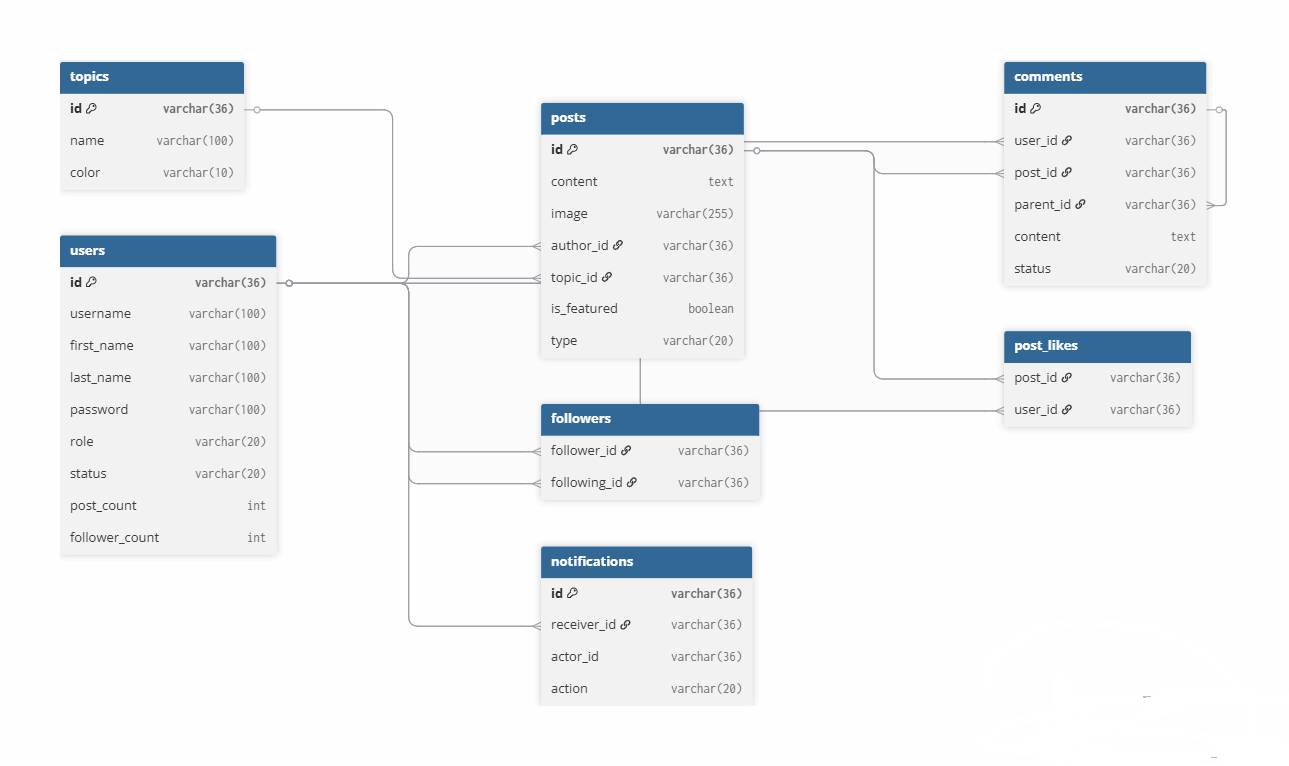


##### Hình 2.13 Biểu đồ Sequence Theo dõi người dùng

### 2.2.4. Thiết kế cơ sở dữ liệu (Database Design)

Cơ sở dữ liệu của hệ thống Bento Social được thiết kế theo mô hình quan hệ, đảm bảo tính nhất quán và toàn vẹn dữ liệu. Các bảng dữ liệu chính bao gồm: users, posts, comments, topics, followers và notifications.

Các bảng được liên kết với nhau thông qua khóa ngoại nhằm phản ánh đúng mối quan hệ nghiệp vụ trong hệ thống, đồng thời hỗ trợ hiệu quả cho các thao tác truy vấn và xử lý dữ liệu.



##### Hình 2.14 Sơ đồ thiết kế cơ sở dữ liệu của hệ thống Bento Social

# III. XÂY DỰNG ỨNG DỤNG

## 3.1. Môi trường phát triển

Kho mã nguồn của hệ thống Bento Social được tổ chức theo mô hình tách biệt giữa Backend và Frontend, bao gồm hai thành phần chính như sau:

* **bento-social-be:**  
  Dịch vụ API Backend của hệ thống, được xây dựng dựa trên nền tảng NestJS, sử dụng Prisma ORM để tương tác với cơ sở dữ liệu PostgreSQL, đồng thời tích hợp Redis nhằm hỗ trợ bộ nhớ đệm và xử lý các tác vụ thời gian thực.
* **bento-social-fe:**  
  Ứng dụng Frontend chịu trách nhiệm hiển thị giao diện và tương tác với người dùng, được phát triển bằng framework Next.js kết hợp với TailwindCSS để xây dựng giao diện linh hoạt và nhất quán.

**Điều kiện tiên quyết**

Trước khi tiến hành triển khai và khởi chạy hệ thống, môi trường phát triển cần được cài đặt đầy đủ các công cụ sau:

* **Node.js:** Môi trường chạy JavaScript phía máy chủ, khuyến nghị sử dụng phiên bản v18 trở lên để đảm bảo khả năng tương thích với các thư viện hiện đại.
* **pnpm:** Trình quản lý gói thư viện được sử dụng để cài đặt và quản lý các phụ thuộc của dự án.
* **Docker và Docker Compose:** Công cụ dùng để khởi tạo và quản lý các container dịch vụ, phục vụ cho việc triển khai cơ sở dữ liệu PostgreSQL và bộ nhớ đệm Redis một cách tự động và nhất quán.

## 3.2. Cài đặt và triển khai hệ thống

### 3.2.1. Thiết lập Backend

Backend yêu cầu cơ sở dữ liệu PostgreSQL và một instance Redis, cả hai đều được điều phối thông qua Docker Compose.

**Bước 1:** Di chuyển vào thư mục backend

|  |
| --- |
| cd bento-social-be |

**Bước 2:** Cài đặt các gói phụ thuộc

|  |
| --- |
| pnpm install |

**Bước 3:** Khởi động các dịch vụ hạ tầng (Postgres & Redis)

|  |
| --- |
| docker-compose up -d |

**Bước 4** Thiết lập sơ đồ (schema) cơ sở dữ liệu

|  |
| --- |
| # Chạy migrations  pnpx prisma migrate dev  # Tạo Prisma client  pnpx prisma generate |

**Bước 5:** Khởi động máy chủ backend:

|  |
| --- |
| pnpm start |

API hiện đã được chạy (cổng mặc định thường là 3000 hoặc 4000, hãy kiểm tra tệp bento-social-be/.env hoặc nhật ký hệ thống).

### 3.2.2. Thiết lập Frontend

Bước 1: Mở một cửa sổ dòng lệnh mới và di chuyển vào thư mục frontend:

|  |
| --- |
| cd bento-social-fe |

Bước 2: Cài đặt các gói phụ thuộc:

|  |
| --- |
| pnpm install |

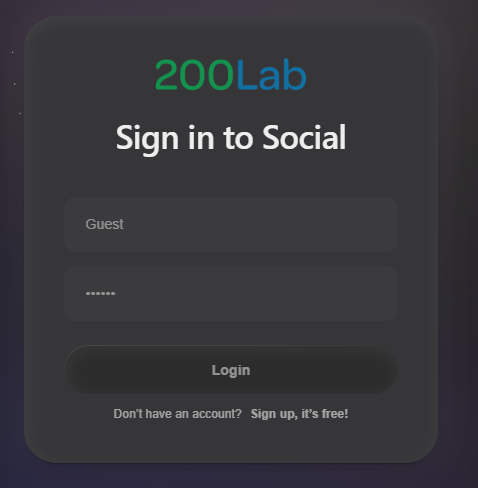
Bước 3: Khởi động máy chủ phát triển (development server):

|  |
| --- |
| pnpm dev |

Bước 4: Mở trình duyệt và truy cập vào: [http://localhost:3000](https://www.google.com/search?q=http://localhost:3000).

## 3.3. Giao diện người dùng

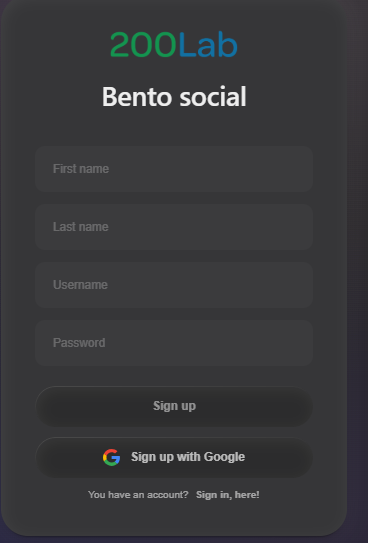
### 3.3.1. Giao diện Đăng nhập (Login)

****

##### Hình 3.1 Giao diện Đăng nhập

* **Kích hoạt**: Đây là trang mặc định khi người dùng chưa có phiên làm việc (token) truy cập vào hệ thống, hoặc bấm vào nút "Sign in". Đường dẫn: /login.
* **Chức năng**:
  + Cho phép người dùng nhập **Username** và **Password** để truy cập hệ thống.
  + Hệ thống kiểm tra tính hợp lệ của thông tin ngay khi nhập (validate).
  + Button "Login" bị vô hiệu hóa (disable) hoặc hiển thị trạng thái loading khi đang xử lý.
  + Có liên kết chuyển hướng sang trang Đăng ký nếu chưa có tài khoản.

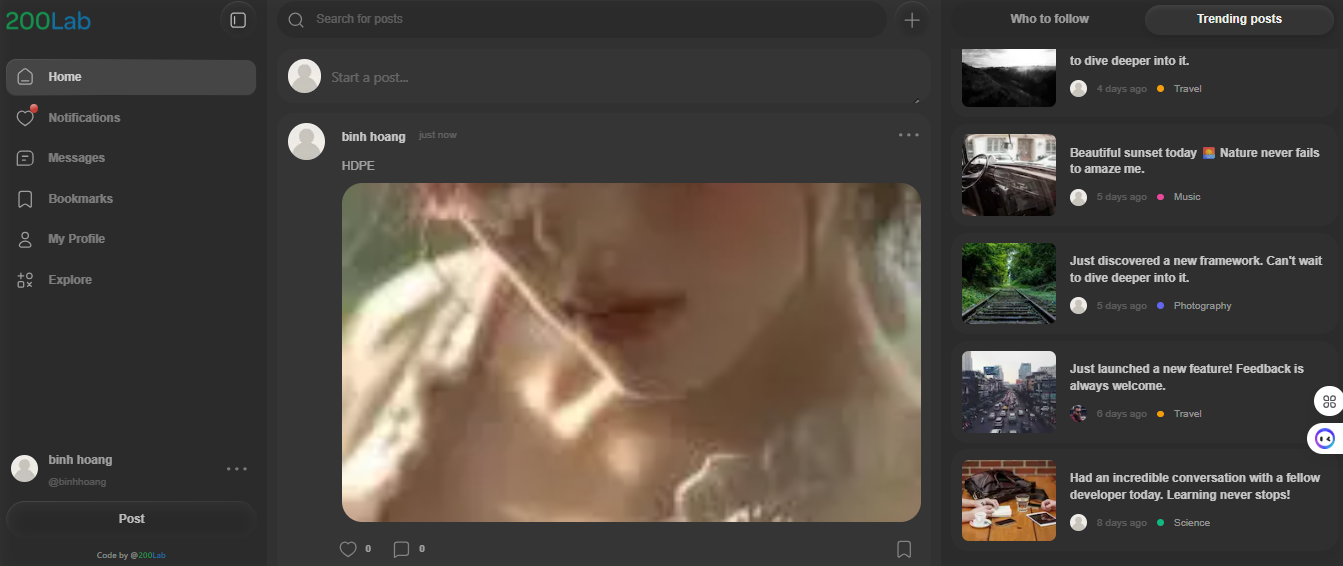
### 3.3.2. Giao diện Đăng ký (Register)

****

##### Hình 3.2 Giao diện Đăng ký

* **Kích hoạt**: Khi người dùng chọn "Sign up" từ trang Login..
* **Chức năng**:
  + Cho phép tạo tài khoản mới với các thông tin: **First name**, **Last name**, **Username**, **Password**.
  + Hiển thị thông báo lỗi cụ thể ngay dưới từng trường nếu nhập sai định dạng hoặc thiếu.
  + Nút "Sign up with Google" (hiện tại hiển thị thông báo "chưa hỗ trợ" khi click).

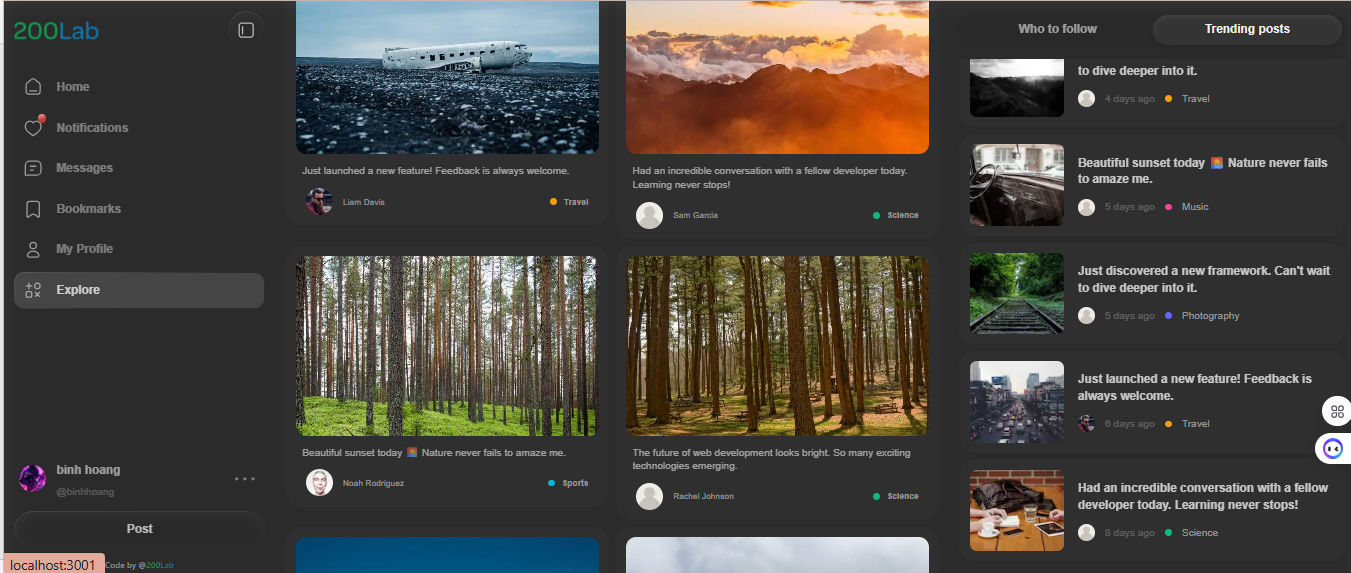
### 3.3.3. Giao diện Trang chủ (Home)

****

##### Hình 3.3 Giao diện Trang chủ

* **Kích hoạt**: Hiển thị ngay sau khi người dùng đăng nhập thành công.
* **Chức năng**:
  + Hiển thị danh sách bài viết (Newsfeed) từ cộng đồng hoặc người mình theo dõi.
  + Thanh điều hướng bên trái (Navigation Bar) để truy cập nhanh các mục khác.
  + Cột bên phải hiển thị Gợi ý theo dõi (Suggested Users) hoặc Xu hướng.
  + Cho phép tương tác nhanh (Like, Comment) trực tiếp trên bài viết.

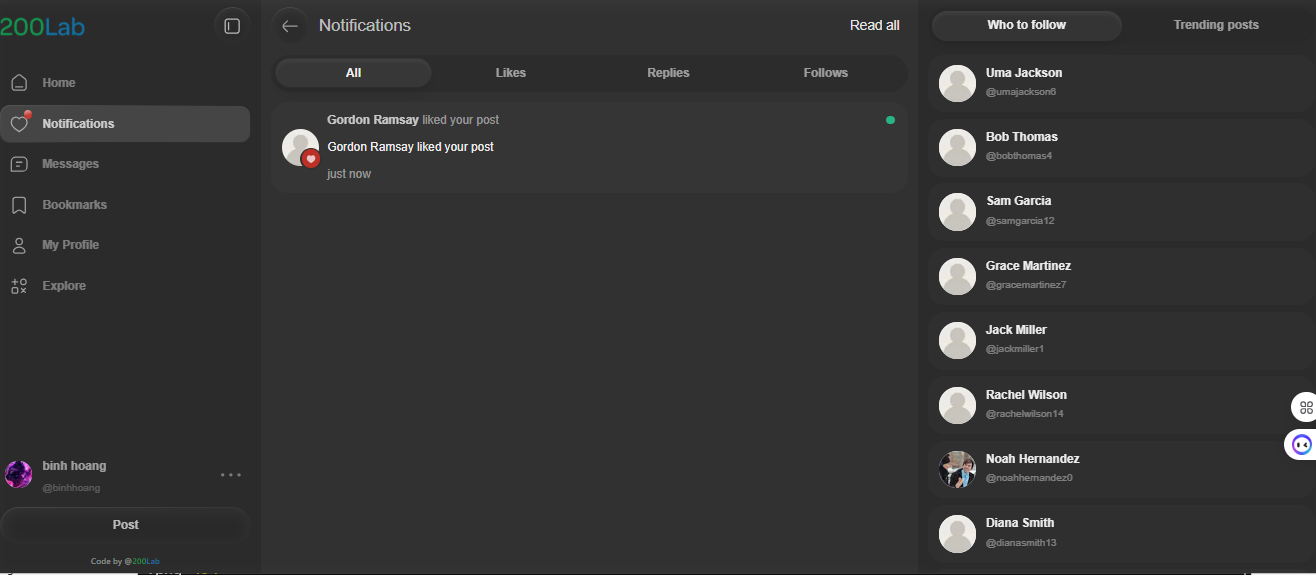
### 3.3.4. Giao diện Khám phá (Explore)

****

##### Hình 3.4 Giao diện Khám phú

* **Kích hoạt**: Khi người dùng chọn menu "Explore" trên thanh điều hướng.
* **Chức năng**:
  + Hiển thị các bài viết nổi bật, các chủ đề (Topics) đang hot.
  + Tìm kiếm người dùng hoặc nội dung (nếu có thanh search).

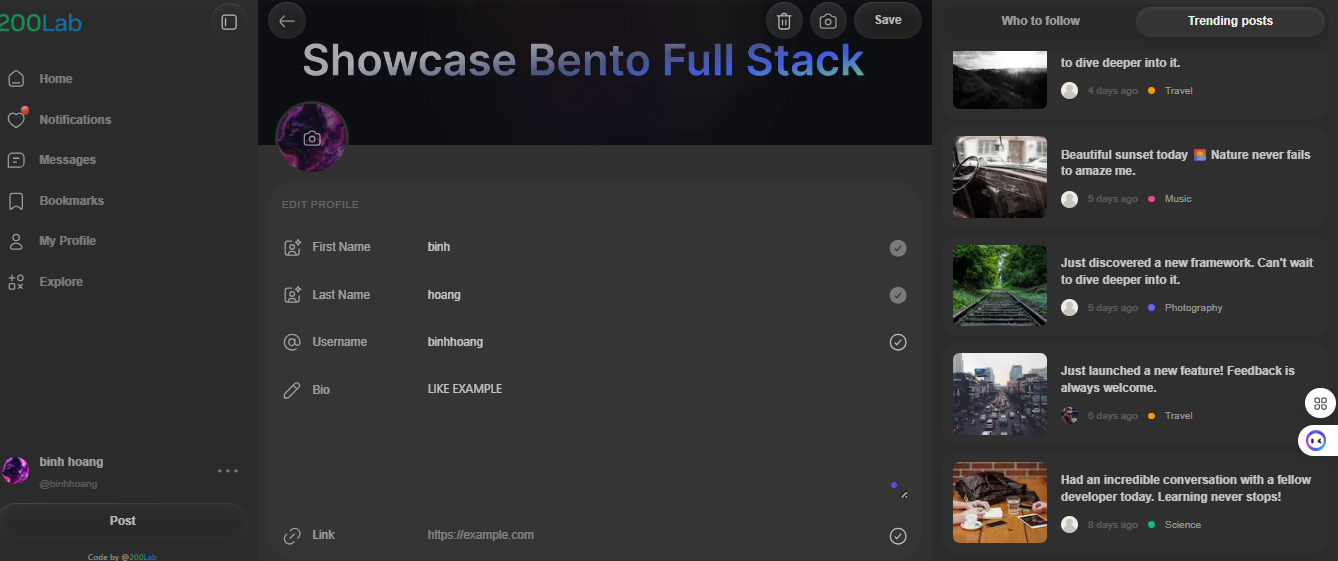
### 3.3.5. Giao diện Thông báo (Notifications)



##### Hình 3.5 Giao diện Thông báo

* **Kích hoạt**: Khi người dùng chọn menu "Notifications"..
* **Chức năng**:
  + Liệt kê danh sách các thông báo mới nhất (ai đã like, comment, follow mình).
  + Phân biệt trạng thái đã đọc/chưa đọc.

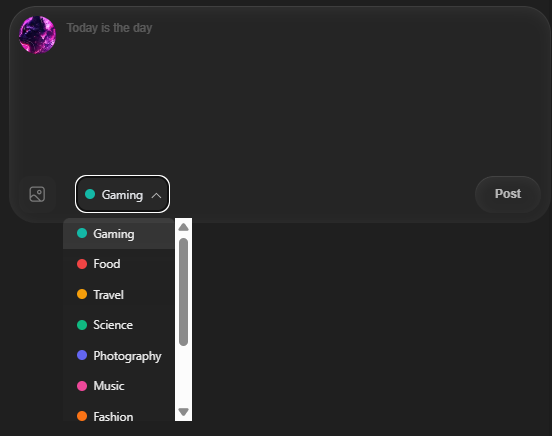
### 3.3.6. Giao diện Trang cá nhân (Profile)

****

##### Hình 3.6 Giao diện Trang cá nhân

* **Kích hoạt**: Khi người dùng click vào Avatar của mình hoặc tên của người khác..
* **Chức năng**:
  + Hiển thị thông tin tổng quan: Avatar, Ảnh bìa (Cover), Tên, Tiểu sử (Bio), Số lượng Follower/Following.
  + Danh sách các bài viết cá nhân của người dùng đó.
  + Các tab chức năng con: Posts, About, Followers, Following.

### 3.3.7. Giao diện Chi tiết bài viết (Post Detail)

****

##### Hình 3.7 Giao diện Chi tiết bài viết

* **Kích hoạt**: Khi click vào một bài viết cụ thể từ Newsfeed hoặc Profile.
* **Chức năng**:
  + Hiển thị toàn bộ nội dung bài viết và hình ảnh kích thước đầy đủ.
  + Hiển thị danh sách bình luận (Comments) và các phản hồi (Replies).
  + Form nhập bình luận ở cuối bài viết.

# KẾT LUẬN

Qua quá trình phân tích và xây dựng, hệ thống **Bento Social** đã hoàn thiện được một nền tảng mạng xã hội cơ bản với đầy đủ các tính năng cốt lõi cần thiết cho một cộng đồng trực tuyến.

* **Về mặt công nghệ:** Hệ thống áp dụng kiến trúc Microservices-ready với việc tách biệt rõ ràng giữa Backend (NestJS) và Frontend (Next.js). Việc sử dụng Docker cho hạ tầng (PostgreSQL, Redis) giúp quá trình triển khai và mở rộng trở nên dễ dàng và linh hoạt. Mã nguồn được tổ chức theo kiến trúc Clean Architecture/Hexagonal Architecture (Ports & Adapters), đảm bảo tính dễ bảo trì và kiểm thử.
* **Về mặt chức năng:** Hệ thống đã đáp ứng tốt các nhu cầu của người dùng từ việc tạo tài khoản, cá nhân hóa hồ sơ (Profile), sáng tạo nội dung (Post), cho đến các hoạt động tương tác xã hội thời gian thực (Like, Comment, Follow, Notification). Cơ chế phân quyền User - Admin cũng đã được thiết lập chặt chẽ để quản lý nội dung và người dùng.

**Hướng phát triển**

Để nâng cao trải nghiệm người dùng và mở rộng quy mô hệ thống trong tương lai, một số hướng phát triển tiềm năng được đề xuất như sau:

1. **Tính năng Chat Real-time (Nhắn tin trực tiếp):**
   * Tích hợp Socket.IO hoặc WebRTC để cho phép người dùng nhắn tin riêng tư (Direct Message) hoặc tạo nhóm chat, hỗ trợ gửi tin nhắn thoại và video call.
2. **Hệ thống Gợi ý thông minh (Recommendation System):**
   * Xây dựng thuật toán AI/Machine Learning để phân tích hành vi người dùng, từ đó gợi ý danh sách Follow phù hợp và sắp xếp Newsfeed theo mức độ quan tâm thay vì chỉ hiển thị theo thời gian thuần túy.
3. **Đa dạng hóa nội dung & Đa phương tiện:**
   * Hỗ trợ đăng tải Video (tích hợp các dịch vụ stream/lưu trữ như AWS S3, Cloudinary).
   * Phát triển tính năng "Story" (tin tự hủy sau 24h) để bắt kịp xu hướng mạng xã hội hiện đại.
4. **Tối ưu hóa Tìm kiếm (Advanced Search):**
   * Tích hợp Elasticsearch để hỗ trợ tìm kiếm toàn văn (Full-text search) nhanh chóng cho bài viết, người dùng và hashtag.
5. **Phiên bản Ứng dụng di động (Mobile App):**
   * Phát triển ứng dụng Native (React Native hoặc Flutter) để tối ưu hóa trải nghiệm trên các thiết bị di động, hỗ trợ thông báo đẩy (Push Notification) tốt hơn.
6. **Nâng cao Bảo mật và Hiệu năng:**
   * Triển khai cơ chế Rate Limiting để chống Spam/DDoS.
   * Sử dụng CDN (Content Delivery Network) để tăng tốc độ tải hình ảnh/video cho người dùng toàn cầu

# TÀI LIỆU THAM KHẢO

[1] **Sam Newman**, *Building Microservices: Designing Fine-Grained Systems*, O’Reilly Media, 2015.

[2] **Martin Fowler**, *Microservices – a definition of this new architectural term*, <https://martinfowler.com/articles/microservices.html>

[3] **George Coulouris, Jean Dollimore, Tim Kindberg**, *Distributed Systems: Concepts and Design*, 5th Edition, Pearson Education, 2011.

[4] **NestJS Team**, *NestJS Official Documentation*, <https://docs.nestjs.com/>